PROCEEDINGS

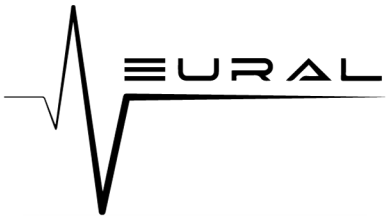
of the

22nd International Conference on Digital Audio Effects

19 DAFx 19

DMT Lab
DMTLAB.BCU.AC.UK

BIRMINGHAM CITY University



soundtoys

Eventide®

Sonnox

ARTURIA®
YOUR EXPERIENCE • YOUR SOUND

ROLI

fxpansion

iZOTOPE

antelope audio

PreSonus

MathWorks®

YAMAHA

SEMANTIC AUDIO LABS

UVI

audiokinetic

Ableton

NATIVE INSTRUMENTS

Proceedings of DAFx 2019

Ryan Stables, Jason Hockman and David Moffat, Eds.

Covers and DAFx-19 logo design by Stefan Adams

ISSN 2413-6700 (Print)
ISSN 2413-6689 (Online)

http://www.dafx.de

Welcome

We are pleased to welcome you to the 22nd International Conference on Digital Audio Effects (DAFx 2019) at Birmingham City University in Birmingham, United Kingdom.

Since its inception in 1998, the conference has been a preeminent annual meeting ground for researchers and practitioners from the fields of audio signal processing, acoustics, and music-related disciplines.

We at Birmingham City University are proud to carry on the DAFx tradition and organise this year's conference and are delighted to host delegates from academia and industry alike in Birmingham. The event has been carefully planned by the local organising committee, which is composed of members of the Digital Media Technology Laboratory (DMT Lab) at Birmingham City University, in collaboration with committee members from Queen Mary University of London, The University of Wolverhampton and Semantic Audio Labs.

The program this year includes 4 tutorials, 3 keynote addresses, and 51 papers, distributed across 7 oral sessions and 3 poster sessions. For the first time in DAFx history, we will also host the 5th Workshop on Intelligent Music Production (WIMP 2019) as a satellite event, in which 3 additional keynotes and 6 more papers will be presented.

There has been a notable subset of this year's papers that have focused on the use of machine learning methods to approximate audio effects and synthesis parameters. The use of machine learning is a noticeable trend in audio technology, and is confluent with most other fields that use digital technologies. The DAFx conferences have also been known for their emphasis on virtual analog modeling. This year, a large subset of the papers are focused on modeling black and greybox systems, with several contributions on physical modeling. Interestingly, there is a cross-over this year, whereby machine learning methods are used to approximate the parameters of complex virtual models.

Following on from the efforts by organisers of previous years' DAFx conferences, the organizing committee is committed to promoting diversity and is pleased to introduce a code of conduct intended to promote a culture of inclusivity. The conference has also offered inclusivity grants intended to assist participants who might struggle to attend for financial reasons.

For our social programme, we have chosen events that we believe provide in-

sight into the Birmingham experience. To this end we have incorporated the first ever Great DAFx Pub Quiz, which promises to be a fun night of audio engineering questions and prizes. Our banquet will be held in The Council House, one of Birmingham's oldest and most prestigious venues, and traditional home to Birmingham's Lord Mayor. To close the event, we will be hosting the first ever DAFx Closing Party event, which will incorporate a trade-show and a night of electronic music at one of Birmingham's most vibrant art gallery spaces.

For DAFx 2019, we have had an overwhelming amount of sponsorship, and we would like to thank all who provided support. The programme would not have been possible without the kind contributions of our sponsors. We would also like to thank last year's organising committee in Aveiro, Portugal for providing us with their excellent LaTeX template for the booklet and Gianpaolo Evangelista for producing these proceedings. Finally, we would like to thank all those in attendance for making this event a success.

The DAFx 2019 Local Organising Committee

Committees

DAFx Board

Daniel Arfib (CNRS-LMA, Marseille, France)
Nicola Bernardini (Conservatorio di Musica "Cesare Pollini", Padova, Italy)
Stefan Bilbao (University of Edinburgh, UK)
Philippe Depalle (McGill University, Montreal, Canada)
Giovanni De Poli (CSC-DEI, University of Padova, Italy)
Myriam Desainte-Catherine (SCRIME, Université de Bordeaux, France)
Markus Erne (Scopein Research, Aarau, Switzerland)
Gianpaolo Evangelista (University of Music and Performing Arts, Vienna, Austria)
Simon Godsill (University of Cambridge, UK)
Robert Höldrich (IEM, University of Music and Performing Arts, Graz, Austria)
Pierre Hanna (Université de Bordeaux, France)
Jean-Marc Jot (DTS, CA, USA)
Victor Lazzarini (National University of Ireland, Maynooth, Ireland)
Sylvain Marchand (L3i, University of La Rochelle, France)
Damian Murphy (University of York, UK)
Søren Nielsen (SoundFocus, Arhus, Denmark)
Markus Noisternig (IRCAM - CNRS - Sorbonne University / UPMC, Paris, France)
Luis Ortiz Berenguer (ETSIS Telecomunicación - Universidad Politécnica de Madrid, Spain)
Geoffroy Peeters (IRCAM, France)
Rudolf Rabenstein (University Erlangen-Nuremberg, Erlangen, Germany)
Davide Rocchesso (University of Palermo, Italy)
Jøran Rudi (NoTAM, Oslo, Norway)
Mark Sandler (Queen Mary University of London, UK)
Augusto Sarti (DEI - Politecnico di Milano, Italy)
Lauri Savioja (Aalto University, Espoo, Finland)
Xavier Serra (Universitat Pompeu Fabra, Barcelona, Spain)
Julius O. Smith (CCRMA, Stanford University, CA, USA)
Alois Sontacchi (IEM, Univ. of Music and Performing Arts, Graz, Austria)
Jan Tro (Norwegian University of Science and Technology, Trondheim, Norway)
Vesa Välimäki (Aalto University, Espoo, Finland)
Udo Zölzer (Helmut-Schmidt University, Hamburg, Germany)

DAFx 19 Local Organizing Committee

Ryan Stables
Jason Hockman
Matthew Cheshire
Brecht De Man
Jake Drysdale
Sean Enderby
Aglaia Foteinou
Nicholas Jillings
David Moffat
Carl Southall
Maciek Tomczak
Xueyang Wang

DAFx 19 Programme Committee

Islah Ali-Maclachlan Birmingham City University
Peter Balazs Acoustics Research Institute, Austrian Academy of Sciences, Vienna
Stefan Bilbao The University of Edinburgh
Marcelo Caetano INESC TEC, Porto
Guilherme Campos University of Aveiro
Vasileios Chatziioannou Institute of Music Acoustics, MdW, Vienna
Matthew Cheshire Birmingham City University
Matthew Davies INESC TEC, Porto
Brecht De Man Semantic Audio Labs, London
Philippe Depalle McGill University, Montreal
Sascha Disch Fraunhofer Institut, Ilmenau
Jake Drysdale Birmingham City University
Michele Ducceschi Univeristy of Edinburgh
Gianpaolo Evangelista University of Music and Performing Arts, Vienna
Sean Enderby Birmingham City University
Gyorgy Fazekas Queen Mary University of London
Bruno Fazenda University of Salford
Federico Fontana University of Udine
Aglaia Foteinou University of Wolverhampton
Brian Hamilton University of Edinburgh
Jason Hockman Birmingham City University
Robert Hoeldrich University of Music and Performing Arts, Graz
Martin Holters Helmut Schmidt University, Hamburg
Nicholas Jillings Birmingham City University
Sylvain Marchand University of La Rochelle
Marius Miron European Commission Joint Research Centre
Dave Moffat Queen Mary University of London
Juan M Navarro Universidad Catolica de Murcia
Julian Parker Native Instruments
Rui Penha INESC TEC / FEUP, Porto
Pedro Pestana UCP , Lisbon
Pavel Rajmic Brno University of Technology
František Rund CTU FEE, Prague
Jiri Schimmel Brno University of Technology
Rod Selfridge Queen Mary University of London
Carl Southall Birmingham City University
Ryan Stables Birmingham City University
Bob Sturm KTH Stockholm
Maciek Tomczak Birmingham City University
Luca Turchet University of Trento
Vesa Valimaki Aalto University
Jose Vieira University of Aveiro
Jeremy Wells University of York
Kurt Werner Queen's University Belfast
Udo Zoelzer Helmut Schmidt University Hamburg
Maarten van Walstijn Queen's University Belfast

Contents

Oral Session 5: Virtual Analog

Oral Session 6: Virtual Analog and Spatial Audio

Oral Session 7: Audio Processing

Poster Session 3

Keynotes

Keynote 1: Magenta

Jesse Engel

Abstract

Magenta is an open source research project exploring the role of machine learning as a tool in the creative process. Our work spans the gamut of pure machine learning research, building developer tools, and using those tools to create new musical instruments and interactions for creatives. In this talk, I will give an overview of a variety of our projects in the realm of polyphonic transcription (Onsets and Frames, MAESTRO, Wav2MIDI2Wav), composition (MusicVAE, GrooVAE, Music Transformer), and audio synthesis (NSynth, GANSynth, RT-NSynth, DDSP). I will also integrate our work on making models adaptive to users in real time (Latent Constraints, MIDI-Me), and implemented in hardware and software platforms (NSynthSuper, magenta.js, Magenta Studio, Fruit Genie). Finally, I'll conclude with a discussion of future directions and lessons learned.

Keynote 2: Encouraging Randomness

Paul Weir

Abstract

Encouraging randomness using generative and procedural audio technology is at the heart of much of Paul's creative work, to achieve this goal, he has helped to develop several original tools. Part opinion piece, part practical demonstration, Paul's presentation will try and bring clarity to definitions surrounding procedural audio to better have an informed discussion and in doing so will present some of the tools that have formed an integral part of his work and lessons learned as part of the development process.

Keynote 3: Intelligent Music Interfaces

Masataka Goto

Abstract

In this keynote I will present intelligent music interfaces demonstrating how end users can benefit from automatic analysis of music signals (automatic music-understanding technologies) based on signal processing and/or machine learning. I will also introduce our recent challenge of deploying research-level music interfaces as public web services and platforms that enrich music experiences. They can analyze and visualize music content on the web, enable music-synchronized control of computer-graphics animation and robots, and provide an audience of hundreds with a bring-your-own-device experience of music-synchronized animations on smartphones. In the future, further advances in music signal analysis and music interfaces based on it will make interaction between people and music more active and enriching.

Tutorials

Tutorial 1: Creative applications of Music and Audio Research

Amélie Anglade and Ryan Groves

Abstract

The definition of Artificial Intelligence (AI) varies wildly depending on the context. In industry, this term can be quite overused and misunderstood. This tutorial attempts to lift the veil on companies who are using AI–or those who are building effective solutions in other ways. We will showcase industry projects (such as apps, platforms, games, performances, pieces of music or audio) that are not necessarily transparent and provide a "forensic" analysis of the research they are using based on our own experience of this field. We will touch on the trend of creativity in the machine learning community, with the challenges it poses to the scientific process. Additionally, we will rely on our own extensive experience as MIR practitioners and consultants in the music and audio creative space to detail the systematic process of creating machine learning products when there is little or no existing research to rely on.

Tutorial 2: Creative Use of Audio Plugins in a Mix: A Live Mixing Session with Grammy-Award Winner Stephen Roessner

Stephen Roessner

Abstract

Join a Grammy-Award Winning Audio Engineer as he uncovers his methods for mixing and use of plugins to achieve creative sounds. The session will be a classroom-style lecture where attendees are free to ask questions about his use of plugins, workflow, and terrible opinions about music – much like what his students endure in his courses.

Tutorial 3: Room Response Equalization

Stefania Cecchi

Abstract

Room response equalization aims at improving the sound reproduction in rooms by applying advanced digital signal processing techniques to design an equalizer on the basis of one or more measurements of the room response. This topic has been intensively studied in the last 40 years, resulting in a number of effective techniques facing different aspects of the problem. This tutorial will review the existing methods following their historical evolution, and discussing pros and cons of each approach with relation to the room characteristics, as well as instrumental and perceptual measures.

Tutorial 4: Principles of Wave Digital Filter Modeling for Virtual Analog: A Tutorial

Kurt Werner

NON-ITERATIVE PHASELESS RECONSTRUCTION FROM WAVELET TRANSFORM MAGNITUDE

Nicki Holighaus, Günther Koliander, Luis Daniel Abreu *

Acoustics Research Institute
Austrian Academy of Sciences
Vienna, Austria
{nholighaus,gkoliander,labreu}@kfs.oeaw.ac.at

Zdeněk Průša

LEWITT GmbH
Vienna, Austria
zprusa@kfs.oeaw.ac.at

ABSTRACT

In this work, we present an algorithm for phaseless reconstruction from magnitude-only wavelet coefficients. The method relies on an explicit relation between the log-magnitude and phase gradients of analytic wavelet transforms and an extension of the Phase-Gradient Heap Integration (PGHI) algorithm recently introduced for Gabor phaseless reconstruction. This relation is exact for a certain family of mother wavelets including Cauchy wavelets of arbitrary order, but only holds approximately otherwise. The presented experiments show that, in practice, the proposed wavelet PGHI method provides competitive quality for various mother wavelets. Furthermore, wavelet PGHI is a non-iterative scheme and thus computational performance is significantly better than established alternate projection methods.

1. INTRODUCTION

The analysis of data utilizing time-frequency or time-scale representations is prevalent in various scientific fields. Prominent examples are medicine [1] and image [2, 3] and audio processing [4, 5, 6]. Although these representations are often visualized by using magnitude-only measurements, they are usually complex-valued, i.e., provide an additional phase component. In general, reconstruction of the signal is only possible from the full complex-valued representation. Since manipulations of the signal are often performed in the magnitude-only representation domain and in some application we can even measure only the magnitudes, there is a need to construct a phase that matches a given magnitude-only representation.

This task is known as phase retrieval or phaseless reconstruction and has been considered from a theoretical [7, 8, 9, 10, 11] as well as an algorithmic [12, 13, 14, 15, 16, 17, 18] viewpoint. While theoretical results mainly deal with the feasibility of phase retrieval, most algorithms are based on iterative projection methods. An important result we will build on, is that although in general the full complex-valued representation is necessary for reconstruction, there are settings where the phase and magnitude components carry almost the same information. The first such case, the STFT with Gaussian generator, was considered by Portnoff [19] and later by Auger and Flandrin [20]. In this setting, the phase gradient and the gradient of the (logarithmically scaled) magnitude are in a one-to-one relationship. Recently, it was shown [21] that for certain mother wavelets $\psi \in \mathbf{L}^2(\mathbb{R})$ this is true for wavelet transforms (WT) as well. More specifically, the Fourier transform of the mother wavelet must satisfy

$$\hat{\psi}(\xi) = \begin{cases} c\xi^{\frac{\alpha-1}{2}} e^{-2\pi\gamma\xi} e^{i\beta\log\xi} & \xi \in \mathbb{R}^+, \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

for some $c \in \mathbb{C}$, $\alpha > -1$, $\beta \in \mathbb{R}$, and $\gamma \in \mathbb{C}$ with $\mathrm{Re}(\gamma) > 0$. Similar to the STFT case with Gaussian generator, the resulting WTs give time-scale representations that are analytic functions for the time-scale parameter pair being interpreted as a single complex variable. This class includes the *Cauchy wavelet* ($\beta = 0$), see [22], for which analyticity of the WT was already shown in [23].

In this paper, we propose an algorithm relying on the phase-magnitude relationship of the WT to perform wavelet phaseless reconstruction. On large scale data, such as audio, this problem has previously been addressed with generic alternating projection methods, such as [16] and its variations. More recently a wavelet-adapted iterative scheme has been proposed in [12, 7]. By combining a discrete approximation of the phase-magnitude relationship with an adaptive integration scheme in the spirit of [24, 25, 26], we can forgo iteration and obtain a phase estimate directly from the magnitude-only coefficients.

We will first recall the results of [21], in particular, the characterization of the WT phase gradient by its log-magnitude gradient for wavelets of the form (1). To motivate the implementation in the discrete domain, we briefly sketch the transition from the continuous to the discrete realm, as well as the invertible WT implementation used in the experiments. Before proceeding to the experiments, which are the main focus of this work and, in contrast to [21], also consider other mother wavelets beside Cauchy wavelets, we formally introduce the wavelet phase gradient heap integration algorithm (WPGHI) for phaseless reconstruction.

Although the phase-magnitude relationship only holds exactly for Cauchy wavelets, we demonstrate that for a broad class of mother wavelets the relations hold approximately and our phase reconstruction algorithm works surprisingly well. At the center of the manuscript is an extensive evaluation that demonstrates the performance of wavelet PGHI under variations of the mother wavelet, its time-frequency resolution trade-off and the oversampling rate. The dependence of reconstruction performance on the parameters is investigated and a comparison with the widely used (fast) Griffin-Lim algorithm [27] is performed. Furthermore, we also consider WPGHI as an initialization for fast Griffin-Lim. In a final experiment, we consider wavelet PGHI for a wavelet with compact support in the time domain. This can be considered the first

* This work was supported by the Austrian Science Fund (FWF): Y 551-N13, I 3067-N30, and P 31225-N32 and the Vienna Science and Technology Fund (WWTF): MA16-053.

step towards a bounded delay implementation of wavelet PGHI in the vein of RTPGHI [28] for the short-time Fourier transform. To complement this experiment, we further present a *causal* variant of wavelet PGHI that processes the wavelet coefficients one time position at a time, assuming that the phase at previous positions is already known. In conjunction with a bounded delay framework for wavelet analysis and synthesis, this algorithm can serve as the central building block for a real-time implementation in the future.

Notation: In this contribution, we consider signals as finite energy functions in a continuous or discrete variable, i.e. $s \in \mathbf{L}^2(\mathbb{R})$ or $s \in \mathbb{C}^L$ for some natural number $L \in \mathbb{N}$. For a differentiable function, we denote partial derivatives by $\frac{\partial}{\partial \bullet}$, where the variable with respect to which we differentiate is substituted for the placeholder $\bullet$. The Fourier transform on $\mathbf{L}^2(\mathbb{R})$ is the unitary operator derived in the usual way from the integral transform $\hat{s}(\xi) = \mathcal{F}(s)(\xi) = \int_{\mathbb{R}} s(t) e^{-2\pi i \xi t}\, dt$ that is defined for integrable signals $s \in \mathbf{L}^1(\mathbb{R})$. Finally, for a complex scalar $z \in \mathbb{C}$, we denote its real and imaginary parts by $\mathrm{Re}(z)$ and $\mathrm{Im}(z)$, respectively.

2. THE PHASE-MAGNITUDE RELATIONSHIP

Fix a function $\psi \in \mathbf{L}^2(\mathbb{R})$ such that its Fourier transform $\widehat{\psi}$ vanishes almost everywhere on $\mathbb{R}^-$. The continuous WT (CWT) of a function (or signal) $s \in \mathbf{L}^2(\mathbb{R})$ with respect to the *mother wavelet* ψ is defined as

$$W_\psi s(x,y) = \langle s, \mathbf{T}_x \mathbf{D}_y \psi \rangle = \frac{1}{\sqrt{y}} \int_{\mathbb{R}} s(t) \overline{\psi\left(\frac{t-x}{y}\right)}\, dt, \quad (2)$$

for all $x \in \mathbb{R}$, $y \in \mathbb{R}^+$. Here, $\mathbf{T}_x$ and $\mathbf{D}_y$ denote the translation and dilation operators, respectively, given by $(\mathbf{T}_x s)(t) = s(t-x)$, and $(\mathbf{D}_y s)(t) = y^{-1/2} s(t/y)$ for all $t \in \mathbb{R}$.

The CWT can be represented in terms of its magnitude $M_\psi^s := |W_\psi s| \geq 0$ and phase $\phi_\psi^s := \arg(W_\psi s) \in \mathbb{R}$ as usual. With this convention, $\log(W_\psi s) = \log(M_\psi^s) + i\phi_\psi^s$.

In [21], it was shown that, with ψ as in (1), the function

$$x + iy \mapsto y^{-\frac{\alpha}{2}} e^{i\beta \log y} W_\psi s\left(x - \frac{\mathrm{Im}(\gamma)}{\mathrm{Re}(\gamma)} y, \frac{y}{\mathrm{Re}(\gamma)}\right). \quad (3)$$

considered as a function in the complex variable $z = x + iy$ ($y > 0$) is analytic, i.e., complex differentiable on the upper half-plane. In this case, the following expressions linking the partial derivatives of the log-magnitude and phase components, hold:

Theorem 1 ([21, Th. 1])**.** *Let $\psi \in \mathbf{L}^2(\mathbb{R})$ be a function that satisfies*

$$\hat{\psi}(\xi) = \begin{cases} \xi^{\frac{\alpha-1}{2}} e^{-2\pi\gamma\xi} e^{i\beta \log \xi} & \xi \in \mathbb{R}^+, \\ 0 & \text{otherwise,} \end{cases} \quad (4)$$

for some $\alpha > -1$, $\beta \in \mathbb{R}$, and $\gamma \in \mathbb{C}$ with $\mathrm{Re}(\gamma) > 0$. Then

$$\frac{\partial}{\partial x}\phi_\psi^s = \frac{\alpha}{2y\,\mathrm{Re}(\gamma)} - \frac{\frac{\partial}{\partial y}\log\left(M_\psi^s\right)}{\mathrm{Re}(\gamma)} + \frac{\mathrm{Im}(\gamma)\frac{\partial}{\partial x}\log\left(M_\psi^s\right)}{\mathrm{Re}(\gamma)} \quad (5)$$

and

$$\frac{\partial}{\partial y}\phi_\psi^s = \frac{\alpha\,\mathrm{Im}(\gamma) - 2\beta}{2y\,\mathrm{Re}(\gamma)} + \frac{|\gamma|^2 \frac{\partial}{\partial x}\log\left(M_\psi^s\right)}{\mathrm{Re}(\gamma)} - \frac{\mathrm{Im}(\gamma)\frac{\partial}{\partial y}\log\left(M_\psi^s\right)}{\mathrm{Re}(\gamma)}. \quad (6)$$

For $\gamma = 1$, these relations simplify to

$$\frac{\partial}{\partial x}\phi_\psi^s(x,y) = -\frac{\partial}{\partial y}\log(M_\psi^s)(x,y) + \frac{\alpha}{2y} \quad (7)$$

and

$$\frac{\partial}{\partial y}\phi_\psi^s(x,y) = \frac{\partial}{\partial x}\log(M_\psi^s)(x,y) - \frac{\beta}{y}. \quad (8)$$

The wavelets ψ specified by (4) are also known as "Klauder wavelets" and minimize a time-scale counterpart of Heisenberg uncertainty [29, Prop. 16]. They are a minor generalization of Cauchy wavelets [22], which are recovered for the choice $\beta = 0$ and $\gamma = 1$. Because a change in γ results only in a scale change, dependent on $\mathrm{Re}(\gamma)$, and a time shift, dependent on $\mathrm{Im}(\gamma)$, we will only consider the case $\gamma = 1$. For most of this contribution, we will in fact consider only Cauchy wavelets.

In the following, ψ will refer to the wavelet specified by (4) with $\gamma = 1$. We can also interpret the wavelet coefficients as time-frequency measurements. More specifically, the Fourier transform of ψ has quick decay around its unique peak (or *center frequency*), located at $\xi_\mathrm{b} = \frac{\alpha-1}{4\pi}$. Considering the $\mathbf{L}^1$-normalized dilation $\tilde{\mathbf{D}}_y s(t) = y^{-1} s(t/y)$, we can define $\tilde{W}_\psi s$ by

$$\tilde{W}_\psi s(x,\xi) = \langle s, \mathbf{T}_x \tilde{\mathbf{D}}_{\xi_\mathrm{b}/\xi} \psi \rangle = \sqrt{\frac{\xi}{\xi_\mathrm{b}}} W_\psi s(x, \xi_\mathrm{b}/\xi).$$

For this form of the WT, straightforward calculations show that the phase-magnitude relations read as follows:

$$\frac{\partial}{\partial x}\tilde{\phi}_\psi^s(x,\xi) = \frac{4\pi\xi^2}{\alpha-1}\frac{\partial}{\partial \xi}\log(\tilde{M}_\psi^s)(x,\xi) + 2\pi\xi, \quad (9)$$

$$\frac{\partial}{\partial \xi}\tilde{\phi}_\psi^s(x,\xi) = -\frac{\alpha-1}{4\pi\xi^2}\frac{\partial}{\partial x}\log(\tilde{M}_\psi^s)(x,\xi) + \frac{\beta}{\xi}, \quad (10)$$

where $\tilde{M}_\psi$ and $\tilde{\phi}_\psi$ denote the magnitude and phase of $\tilde{W}_\psi$, respectively. In the following, we will discretize this form of the phase-magnitude relations to derive a discrete approximation for application within our proposed phaseless reconstruction algorithm.

It is notable that the formulas (9)–(10) almost exactly correspond to the phase-magnitude relations in the STFT case with a dilated Gaussian [24, Sec. III]. In particular, for $\beta = 0$ the only difference is that the constant time-frequency ratio λ is replaced by the frequency-dependent term $\frac{\alpha-1}{4\pi\xi^2}$.

3. PHASELESS RECONSTRUCTION

To perform phaseless reconstruction, we have to assume that the given magnitude coefficients originate from an invertible wavelet system, i.e., a wavelet frame, see [30] and references therein. In this case, we can perform phase estimation followed either by direct synthesis via a dual frame [31, 32, 33, 34] or iterative synthesis via, e.g., conjugate gradient iteration [35, 6].

Although the performance of phaseless reconstruction should be largely independent of the particular implementation of the analysis and synthesis operations, we briefly sketch a potential implementation for illustrative purposes, largely following [36, 37, 38, 6]. We will denote discretizations of continuous signals by brackets, e.g., the discretized signal $s_\mathrm{d}[l] \in \mathbb{C}$ for $l \in \{1, \ldots, L-1\}$ and some $L \in \mathbb{N}$. In this discrete domain, the translation operator acts circularly, i.e., $s_\mathrm{d}[l-m]$ is interpreted as $s_\mathrm{d}[\mathrm{mod}(l-m, L)]$.

We mimic the dilation operator by sampling the continuous frequency response of the mother wavelet $\psi \in \mathbf{L}^2(\mathbb{R}) \cap \mathbf{L}^1(\mathbb{R})$ at the appropriate density: Assuming the sampling rate ξ_s, the frequency response of the wavelet at scale $y = \xi_\mathrm{b}/\xi$ is simply

$$\widehat{\psi_y}[k] = \widehat{\psi}\left(\frac{y\xi_\mathrm{s}k}{L}\right) = \widehat{\psi}\left(\frac{\xi_\mathrm{b}\xi_\mathrm{s}}{L}\frac{k}{\xi}\right),$$

for $k \in \{-\lfloor L/2 \rfloor, \dots, \lceil L/2 \rceil - 1\}$. In practice, we only cover a finite range of scales and to cover the entire frequency range, we introduce an additional low-pass function in the style of [6, Sec. 3.1.2].

The entire wavelet system is characterized by the minimum scale $y_\mathrm{m} \in \mathbb{R}^+$, the scale step $2^{1/B}$, with $B \in \mathbb{R}^+$, the number of scales $K \in \mathbb{N}$, and the decimation[1] factor $a_\mathrm{d} \in \mathbb{N}$, with $a_\mathrm{d} | L$. The corresponding scaled and shifted wavelets are given as

$$\psi_{n,k} = \mathbf{T}_{na_\mathrm{d}} \psi_{2^{k/B} y_\mathrm{m}} \tag{11}$$

for $k \in \{0, \dots, K-1\}$ and $n \in \{0, \dots, L/a_\mathrm{d} - 1\}$. A plateau function $P_\mathrm{lp} \in \mathbb{C}^L$, centered at 0, specifies the low-pass function as

$$\widehat{\psi_\mathrm{lp}} = a_\mathrm{d}^{-1} P_\mathrm{lp} \Psi_\mathrm{lp}, \tag{12}$$

where

$$\Psi_\mathrm{lp} = \sqrt{\max(\Psi) - \Psi}, \quad \Psi = \sum_{k=0}^{K-1} |\widehat{\psi_{0,k}}|^2. \tag{13}$$

An analysis with the constructed system yields LK/a_d complex-valued coefficients for the wavelet scales and additional L/a_d real-valued coefficients for the low-pass function, for a total redundancy of $(2K+1)/a_\mathrm{d}$ when analyzing signals with no negative frequency content. With a slight abuse of terminology, we will from now on refer to the proportional, but simpler quantity K/a_d as the redundancy.

In the following, we will assume that ψ satisfies (4) with $\beta = 0$ and $\gamma = 1$. The adaptation to general β and γ is straightforward, but lowers readability, so we leave it to the reader. For general wavelets, we have to determine the appropriate value of α by comparing the chosen mother wavelet to $\psi^{(\alpha)}$ for varying α and select the best match, see Section 4.

Assume that the continuous-time signal s is approximately band- and time-limited on $[0, \xi_\mathrm{s}[$ and $[0, L/\xi_\mathrm{s}[$, respectively. Then, with $s_\mathrm{d}[l] = s(l/\xi_\mathrm{s})$, for $l \in \{0, \dots, L-1\}$, $a_\mathrm{d} = a\xi_\mathrm{s} \in \mathbb{N}$, and $\xi_k = 2^{-k/B}\xi_\mathrm{b}/y_\mathrm{m}$, we obtain the approximation

$$M_s[n,k] := |\langle s_\mathrm{d}, \psi_{n,k} \rangle| \approx \xi_\mathrm{s} \tilde{M}^s_\psi(na, \xi_k). \tag{14}$$

Using (14), we can formulate a discrete approximation of the phase-magnitude relations (9) and (10). Note that normalization by ξ_s becomes irrelevant after taking the logarithmic derivative in (14). As a substitute for the continuous partial derivatives, we take Δ_n and Δ_k to be an appropriate discrete differentiation scheme. Our implementation relies on (weighted) centered differences:

$$\Delta_n(M_s)[n,k] := \frac{\xi_\mathrm{s}(M_s[n+1,k] - M_s[n-1,k])}{2a_\mathrm{d}}, \tag{15}$$

$$\begin{aligned}\Delta_k(M_s)[n,k] := &\frac{M_s[n,k+1] - M_s[n,k]}{2(\xi_{k+1} - \xi_k)} \\ &+ \frac{M_s[n,k] - M_s[n,k-1]}{2(\xi_k - \xi_{k-1})}.\end{aligned} \tag{16}$$

[1] For simplicity, we restrict here to uniform decimation.

Here, weighted centered differences are used in Δ_k, since the sampling step in the scale coordinate changes depends on k. For border points, i.e., $n \in \{0, N-1\}$ and $k \in \{0, K-1\}$, respectively, the appropriate forward or backward differences are used instead.

If we combine (14) with the phase-magnitude relations (9) and (10), we obtain

$$\begin{aligned}\frac{\partial}{\partial x}\tilde{\phi}^s_\psi(na, \xi_k) &= \frac{4\pi\xi_k^2}{\alpha - 1}\frac{\partial}{\partial \xi}\log(\tilde{M}^s_\psi)(na, \xi_k) + 2\pi\xi_k \\ &\approx \Delta^{\tilde{\phi},x,s}_\psi[n,k] := \frac{4\pi\xi_k^2}{\alpha-1}\Delta_k(\log(M_s))[n,k] + 2\pi\xi_k,\end{aligned} \tag{17}$$

and

$$\begin{aligned}\frac{\partial}{\partial \xi}\tilde{\phi}^s_\psi(na, \xi_k) &= -\frac{\alpha - 1}{4\pi\xi_k^2}\frac{\partial}{\partial x}\log(\tilde{M}^s_\psi)(na, \xi_k) \\ &\approx \Delta^{\tilde{\phi},\xi,s}_\psi[n,k] := -\frac{\alpha-1}{4\pi\xi_k^2}\Delta_n(\log(M_s))[n,k].\end{aligned} \tag{18}$$

Now, from $\Delta^{\tilde{\phi},x,s}_\psi$ and $\Delta^{\tilde{\phi},\xi,s}_\psi$, an estimate of the phase of $\tilde{W}_\psi s$ at the sampling points $\{(na, \xi_k)\}_{n,k}$ can be obtained using a quadrature rule considering the variable sampling intervals. The provided implementation relies on simple 1-dimensional trapezoidal quadrature. This results in the following integration rule on the set of neighbors of (n,k), i.e., $(n_\mathrm{n}, k_\mathrm{n}) \in \mathcal{N}_{n,k} := \{(n \pm 1, k), (n, k \pm 1)\} \cap \{0, \dots N-1\} \times \{0, \dots, K-1\}$.

$$\begin{aligned}&(\tilde{\phi}^s_\psi)_\mathrm{est}[n_\mathrm{n}, k_\mathrm{n}] \\ &= (\tilde{\phi}^s_\psi)_\mathrm{est}[n,k] + \frac{\xi_{k_\mathrm{n}} - \xi_k}{2}\left(\Delta^{\tilde{\phi},\xi,s}_\psi[n,k] + \Delta^{\tilde{\phi},\xi,s}_\psi[n_\mathrm{n}, k_\mathrm{n}]\right) \\ &\quad + \frac{a_\mathrm{d}(n_\mathrm{n} - n)}{2\xi_\mathrm{s}}\left(\Delta^{\tilde{\phi},x,s}_\psi[n,k] + \Delta^{\tilde{\phi},x,s}_\psi[n_\mathrm{n}, k_\mathrm{n}]\right).\end{aligned} \tag{19}$$

When inserting (15) and (16) into (19), the absolute scale of the center frequencies ξ_k and sampling rate ξ_s becomes unimportant and only their ratio enters the quadrature (19). Hence, by considering relative frequencies ξ_k/ξ_s, the algorithm is valid independent of the assumed sampling rate.

However, the integration step itself is not entirely straightforward. As discussed in [7] and [10], phase estimation from magnitude-only measurements is generally highly unstable when the coefficients are close to 0. To avoid these instabilities, the work [24] introduced the Phase Gradient Heap Integration (PGHI) algorithm, originally for Gabor phase reconstruction. The algorithm adaptively applies a given integration rule, starting at coefficients of large magnitude and avoiding areas of low magnitude. In the pseudo-code shown in Algorithm 1 it is assumed that all wavelet coefficients are available at all times, similar to [24] and [26].

Once the phase estimate $(\tilde{\phi}^s_\psi)_\mathrm{est}$ has been computed, it is combined with the magnitude by $W_s := M_s e^{i(\tilde{\phi}^s_\psi)_\mathrm{est}}$ and a time-domain signal is obtained by performing a regular synthesis step.

4. EXPERIMENTS

The performance of wavelet PGHI with Cauchy WTs was extensively evaluated on the EBU SQAM database [40] in [21], see also `http://ltfat.github.io/notes/053/`. Here, we evaluate the performance of wavelet PGHI with mother wavelets differing from the Cauchy wavelet. To this end, we selected a var-

Algorithm 1: Wavelet Phase Gradient Heap Integration

Input: Magnitude M_s of wavelet coefficients, estimates $\Delta^{\tilde{\phi},x,s}_{\psi}$ and $\Delta^{\tilde{\phi},\xi,s}_{\psi}$ of the partial phase derivatives, relative tolerance tol.
Output: Phase estimate $(\tilde{\phi}^s_{\psi})_{\text{est}}$.

$abstol \leftarrow tol \cdot \max(M_s[n,k])$;
Create set $\mathcal{I} = \{(n,k) : M_s[n,k] > abstol\}$;
Assign random values to $(\tilde{\phi}^s_{\psi})_{\text{est}}(n,k)$ for $(n,k) \notin \mathcal{I}$;
Construct a self-sorting max *heap* [39] for (n,k) pairs;
while $\mathcal{I}$ is not $\emptyset$ **do**
  if *heap* is empty **then**
    Move $(n_m,k_m) = \arg\max_{(n,k)\in\mathcal{I}} (M_s[n,k])$ from $\mathcal{I}$ into the *heap*;
    $(\tilde{\phi}^s_{\psi})_{\text{est}}(n_m,k_m) \leftarrow 0$;
  end
  while *heap* is not empty **do**
    $(n,k) \leftarrow$ remove the top of the *heap*;
    foreach $(n_{\text{n}},k_{\text{n}})$ in $\mathcal{N}_{n,k} \cap \mathcal{I}$ **do**
      Compute $(\tilde{\phi}^s_{\psi})_{\text{est}}(n_{\text{n}},k_{\text{n}})$ by means of (19);
      Move $(n_{\text{n}},k_{\text{n}})$ from $\mathcal{I}$ into the *heap*;
    end
  end
end

ied subset of 15 signals from the EBU SQAM database,[2] including signals that were reported as critical in previous contributions [24, 21]. The chosen subset contains synthetic signals, solo instruments, speech, and music.

We performed 3 experiments, derived from the experimental protocol in [21] so that results are comparable between studies. The experiments are described and discussed below. For wavelet analysis and synthesis, we used the filter bank methods in the open source Large Time-Frequency Analysis Toolbox (LTFAT [41], `http://ltfat.github.io/`), where our implementation of *wavelet PGHI* is available by using the `'wavelet'` flag in `filterbankconstphase`. A function to generate the wavelet filters and scripts for generating the individual experiments are provided on `http://ltfat.github.io/notes/055/`, together with audio examples for all experiment conditions. The functionality represented by the supplied code is to be integrated into the next release of the LTFAT Toolbox.

In Experiments I and II, we consider the following wavelet types in addition to the Cauchy wavelet: Morlet wavelets, generalized Morse wavelets [42] with symmetry parameter $\gamma \in \{2,3\}$ and bandlimited wavelets generated in the Fourier domain as cardinal B-spline of order $m \in \{3,5\}$. The latter have previously been called frequency B-spline wavelets [43]. Note that Cauchy wavelets are generalized Morse wavelets with $\gamma = 1$. For $\gamma = 3$, the generalized Morse wavelet is also known as Airy wavelet. In each case, the remaining parameters were adapted to match the bandwidth of Cauchy wavelets of the desired order $(\alpha-1)/2$. In Experiment III, we consider wavelets with compact support in the time domain, namely exponentially modulated B-splines. Here, we fixed the B-spline order $m = 4$. In Table 1, we list all the used wavelet types, their parameters and their Fourier transform $\hat{\psi}$.

The decimation step a_{d} and the number of frequency channels K (without the lowpass filter) were chosen equal to those used for the matched Cauchy wavelet. As quantitative error measure, we employ (wavelet) spectral convergence [44], i.e., the relative mean squared error (in dB) between the wavelet coefficient magnitude of the target signal s_{t} and the proposed solution s_{p}:

$$SC(s_{\text{p}},s_{\text{t}}) = 20\log_{10}\frac{\|M_{s_{\text{p}}} - M_{s_{\text{t}}}\|}{\|M_{s_{\text{t}}}\|}.$$

It should be noted that the wavelet coefficient magnitude in the above formula was computed using the same parameter set for which phaseless reconstruction was attempted. There is no unique method to match the Cauchy wavelet parameter α to another type of mother wavelet. We use a procedure that determines α such that the peak-normalized frequency responses of a given mother wavelet ψ and a Cauchy wavelet $\psi^{(\alpha)}$ of order $(\alpha-1)/2$ with the same central frequency have the same width at a given threshold height $h_{\text{thr}} > 0$. The value of α is computed by the MATLAB function `wpghi_findalpha.m`, supplied on the project webpage `http://ltfat.github.io/notes/055/`.

4.1. Experiment I—Comparison to Fast Griffin-Lim

To study the performance of the proposed algorithm for various mother wavelets and parameter settings, we compare wavelet PGHI to the iterative *fast Griffin-Lim* [16, 27] algorithm. The experimental protocol is similar to Experiment I in [21]. The Cauchy WT serves as a baseline comparison, it is specified by the parameter tuple $(\alpha, a_{\text{d}}, K)$. In this experiment, we considered the following settings: $(30,10,100)$, $(300,24,240)$, and $(3000,40,400)$, leading to a fixed redundancy[3] $K/a_{\text{d}} = 10$. For all settings, the channel center frequencies where geometrically spaced in $\frac{\xi_s}{20} \cdot [2^{-6}, 2^{3.3}]$. Matching the other wavelet types to the given values of α, we obtained the parameters given in Table 2.

We compare three different methods: wavelet PGHI (WPGHI, *proposed*), fast Griffin-Lim with zeros initialization (0-FGLIM, [27]) and fast Griffin-Lim initialized with the result of WPGHI (W-FGLIM). Fast Griffin-Lim was restricted to at most 40 iterations. Nonetheless, it should be noted that the execution time of WPGHI is a small fraction of the time required for either 0-FGLIM or W-FGLIM. Maximum, median, and minimum values for spectral convergence of the three methods are shown in Figure 1 for the different parameter sets $(\alpha, a_{\text{d}}, K)$.

The 0-FGLIM baseline shows the most stable performance across conditions, with little dependence on the mother wavelet or the parameter α. On the tested signals, it also shows the least dependence on the signal content, with 7–10 dB difference in spectral convergence between the best and worst result on any fixed condition. Notably, 0-FGLIM performs slightly better for the lowest value of α, i.e., the wavelets with worst frequency resolution.

The median performance of WPGHI is better than 0-FGLIM for $\alpha \in \{300, 3000\}$, but not for $\alpha = 30$, except when the Cauchy wavelet is used. At low values of α, the Cauchy wavelet is very asymmetric, and thus most different from the other considered mother wavelets. That the Morse wavelet with $\gamma = 3$ performs second best for $\alpha = 30$ gives further indication that the performance of WPGHI depends, as expected, on the closeness of the

[2] The chosen signals are as follows: 01, 02, 04, 14, 15, 16, 27, 39, 49, 50, 51, 52, 53, 54, 70. For each signal, tests were performed on the first 5 seconds of the signal.

[3] In [21], the redundancy $K/a_{\text{d}} = 20$ was used, but Experiment II in [21] suggests that $K/a_{\text{d}} = 10$ still provides excellent performance of all methods.

Wavelet name	Parameters	Fourier transforms $\hat{\psi}(\xi)$
Generalized Morse (M•)	$\alpha > 1, \gamma > 0$	$c_{\alpha,\gamma} \cdot \xi^{(\alpha-1)/2} e^{-2\pi\xi^{\gamma}}$
Morlet (M)	$\sigma > 0$	$c_{\sigma} \cdot (e^{-(\sigma-\xi)^2/2} - e^{-(\sigma^2+\xi^2)/2})$
Frequency B-spline (FB•)	$m \in \mathbb{N}, \xi_{fb} \geq 2$	$c_{m,\xi_{fb}} \cdot B_m(\xi - m\xi_{fb}/4)$
Modulated B-spline (MB•)	$m \in \mathbb{N}, \xi_{fm} \in \mathbb{N}$	$c_{m,\xi_{fm}} \cdot \sin(\pi(\xi - \xi_{fm}))^m / (\pi(\xi - \xi_{fm}))^m$

Table 1: Wavelets used in the experiments. The symbol • is a placeholder for γ or m in the Generalized Morse and B-Spline wavelets, i.e., M3 denotes a Morse wavelet with $\gamma = 3$. For $\gamma = 1$, the generalized Morse wavelet yields the Cauchy wavelet. The parameter σ of the Morlet wavelet is related to the center frequency and controls the time/frequency resolution trade-off. The parameter m of the B-Spline type wavelets denotes the order of the generating B-spline (a B-Spline of order m is a piecewise polynomial of order $m-1$). ξ_{fb} and ξ_{fm} denote the center frequency to bandwidth and center frequency to main lobe width, respectively. The restriction $\xi_{fm} \in \mathbb{N}$ guarantees that the modulated B-spline satisfies $\hat{\psi}(0) = 0$ and is admissible.

Wavelet / Cauchy α	≈ 30	≈ 300	≈ 3000	≈ 1000
M2, $\alpha =$	28.93	298.93	2998.93	999
M3, $\alpha =$	28.45	298.55	2998.45	999
M, $\sigma =$	3.79	12.25	38.78	22.38
FB3, $\xi_{fb} =$	—	4.32	13.70	7.90
FB5, $\xi_{fb} =$	—	3.25	10.30	5.94

Table 2: Wavelets parameters to match a given Cauchy parameter $\alpha \in \{30, 300, 3000\}$ used in Experiment I and $\alpha = 1000$ used in Experiment II. Note that $\alpha \approx 30$ cannot be achieved with the frequency B-Spline wavelet of order $m \in \{3, 5\}$.

mother wavelet to the Cauchy wavelet with the value of α, for which the phase-magnitude relations (17) and (18) are invoked.

As expected, WPGHI performs worse when the Cauchy wavelet is not used, but at large values of α, the difference in median and worst values is small. Over all values of α, the Frequency B-Spline wavelet of order $m = 3$ performs worst, and the Morse wavelet with $\gamma = 2$ performs closest to the Cauchy wavelet. The large range of values for WPGHI corroborates the observations from [21] that WPGHI performance can depend significantly on the signal content, for all chosen mother wavelets. Furthermore, the best performance seems to depend heavily on the chosen mother wavelet and its closeness to the Cauchy wavelet. Initializing fast Griffin-Lim with the result of WPGHI (W-FGLIM) shows significant improvements over either WPGHI or 0-FGLIM in all considered scenarios. In most cases, the final performance of W-FGLIM seems to be proportional to the quality of the WPGHI initialization.

Informal listening mostly confirmed the numerical results. At $\alpha \in \{300, 3000\}$, all methods produce little to no audible distortion, with the exception of 0-FGLIM for simple signals such as synthetic sine waves where the defect is still clearly audible. At $\alpha = 30$, WPGHI sometimes produces results that are perceptually worse when the Cauchy wavelet is not used, but nonetheless, distortions were often more severe in 0-FGLIM, despite contradicting numerical results. The results of W-FGLIM provide excellent quality, even at $\alpha = 30$, where the individual methods may fail to do so.

4.2. Experiment II—Changing the Redundancy

In a second set of experiments, we investigate the influence of the redundancy K/a_{d} on the performance of the proposed methods WPGHI and W-FGLIM. Once more, the experiment follows the protocol established in Experiment II in [21], but with the main aim to compare performance across different mother wavelets. We fix $\alpha = 1000$ and once more match the parameters of all alternative mother wavelets, see Table 2. The considered redundancies are $K/a_{\mathrm{d}} \in \{3, 5, 10\}$. In contrast to [21], we do not consider $K/a_{\mathrm{d}} = 30$, as the results were rather close to the case $K/a_{\mathrm{d}} = 10$ and the same is expected here.

We fix the following parameter sets $(\alpha, a_{\mathrm{d}}, K)$ for the Cauchy wavelet baseline: low redundancy $(1000, 30, 90)$, medium redundancy $(1000, 25, 125)$, high redundancy $(1000, 18, 180)$. Similar to Experiment I, median value, maxima, and minima over the test set are presented in Figure 2 for all parameter sets and mother wavelets. As expected, performance of both proposed methods decreases at lower redundancy, but median performance at low redundancy is still decent. Both median and worst performance shows little dependence on the mother wavelet, due to the chosen large value of α, at which all considered mother wavelets are reasonably close to the Cauchy wavelet. Some influence of the mother wavelet is still apparent in the best values of spectral convergence. Especially for plain WPGHI, the Cauchy wavelet is still at an advantage. At any redundancy, W-FGLIM yields a significant improvement over plain WPGHI, but at low redundancy, the additional Griffin-Lim iteration only marginally improves the reconstruction on signals for which WPGHI performs badly. Perceptual quality is excellent over all redundancies. Only at low redundancy, minor distortions were observed.

4.3. Experiment III—Towards WPGHI with Bounded Delay

The implementation of a bounded delay framework for wavelet analysis and synthesis is more involved than for the short-time Fourier transform and not the objective of this contribution. Nonetheless, we want to indicate some steps that can be taken to enable the use of WPGHI within such a framework. First, we need to show that WPGHI produces good results in conjunction with mother wavelets that are compactly supported in the time domain and thus necessarily $\hat{\psi}(\xi) = 0$ for all $\xi \in \mathbb{R}^-$ cannot be satisfied.

To this end, we repeat Experiment I with a modulated B-spline of fixed order 4 as mother wavelet ψ. Due to the restriction of the parameter ξ_{fm} to positive integers, it is not possible to construct complex-modulated B-spline wavelets that match a Cauchy wavelet of arbitrary order $(\alpha - 1)/2$. Instead we choose values for the center frequency to main lobe width ratio ξ_{fm} and compute the matching Cauchy parameter α. The resulting parameter values are shown in Table 3.

The Fourier transform $\hat{\psi}$ of the modulated B-Spline is symmetric around its peak. Hence, the lowest value of ξ_{fm} corre-

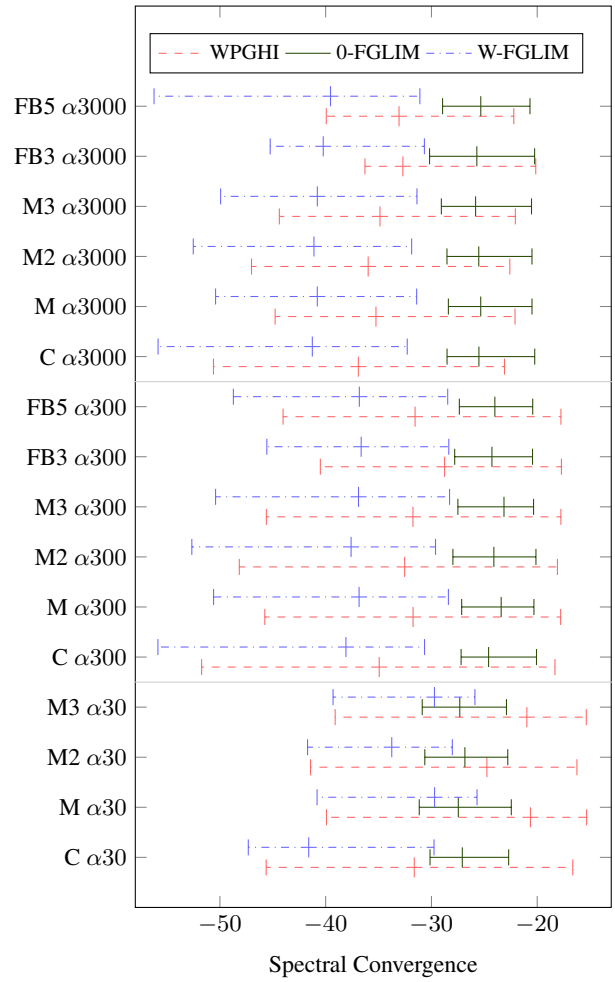


Figure 1: Results of Experiment I for wavelet PGHI and fast Griffin Lim for Cauchy $\alpha \in \{30, 300, 3000\}$ and all tested wavelets. The minimal, median, and maximal spectral convergences over the 15 test signals are depicted.

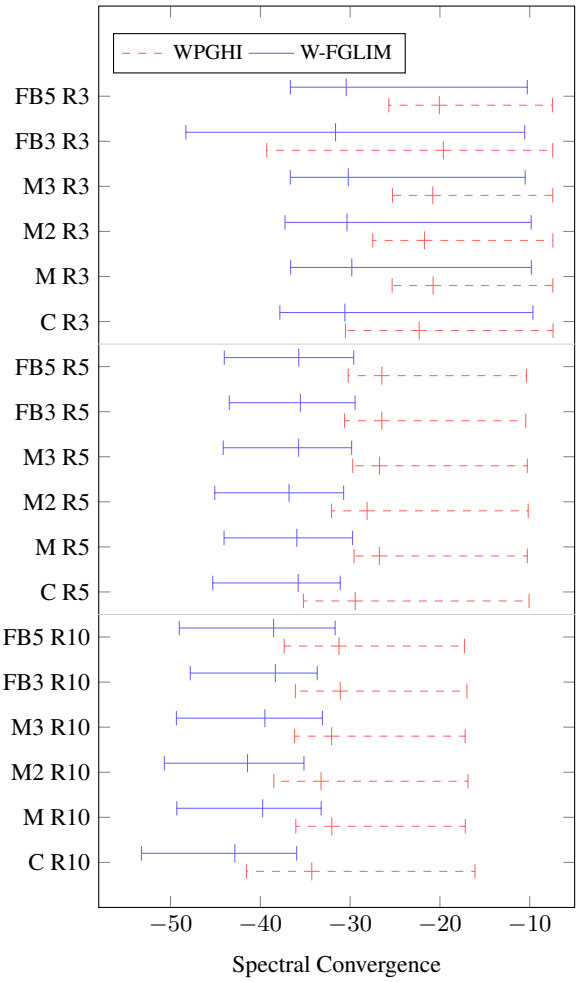


Figure 2: Results of Experiment II for wavelet PGHI and all tested wavelets at redundancies $K/a_d \in \{10, 5, 3\}$. The minimal, median, and maximal spectral convergences over the 15 test signals are depicted.

sponds to a small Cauchy wavelet parameter α, i.e., a rather asymmetric Cauchy wavelet. Thus, the average performance of WPGHI is expected to be inhibited for the modulated B-Spline. While this is certainly apparent in the results, see Figure 3, reconstruction quality is still decent. At higher values of ξ_{fm}, the difference between WPGHI performance for the two tested mother wavelets becomes increasingly negligible and reconstruction performance becomes competitive with or even better than the iterative fast Griffin-Lim algorithm. For the sake of completeness, we also show results for W-FGLIM, which continues to outperform both competing methods. Generally speaking, the results are, not unexpectedly, very similar to those obtained in Experiment I. The same is true for perceptual performance, where differences between the wavelets (in line with numerical results) have been observed almost exclusively for $\xi_{fm} = 1$.

Both 0-FGLIM and W-FGLIM involve iteration relying on many wavelet analysis and synthesis steps. On the other hand, the computations necessary for WPGHI are elementary and can be adapted to a real-time (bounded delay) setting easily. This has previously been shown for the short-time Fourier transform in [28].

We close this contribution with Algorithm 2, which demonstrates the changes necessary to adapt WPGHI to such a setting. This algorithm computes a phase estimate $(\tilde{\phi}^s_\psi)_{\text{est}}[n, \cdot]$ using the phase gradient at time positions $n - 1$ and n and relying on a previously computed phase estimate for $(\tilde{\phi}^s_\psi)_{\text{est}}[n - 1, \cdot]$, for $n = 1, \ldots, N - 1$. Assuming that the phase and phase derivative *at time* -1 are identically 0, it can also be used to initialize the phase estimate for $(\tilde{\phi}^s_\psi)_{\text{est}}[0, \cdot]$ from scratch.

5. CONCLUSION

We have presented a non-iterative method for reconstruction from magnitude-only wavelet coefficients, relying on the phase-magnitude relations for WTs with Cauchy-type mother wavelet, recently introduced in [21]. The resulting algorithm is computationally highly efficient and often performs on par or better than previous iterative schemes, for which it can also serve as an initialization. The latter initialization has been shown to boost the performance of either method used individually. In the presented experiments we showed that the theoretical restriction to Cauchy-type mother wavelets

Cauchy, $\alpha =$	29.7	257.0	2841.7
Mod. B-spline, $m = 4$, $\xi_{fm} =$	1	3	10

Table 3: Wavelet parameters used in Experiment III. Each column lists matching parameters between the Cauchy wavelet and the modulated B-spline of order $m = 4$.

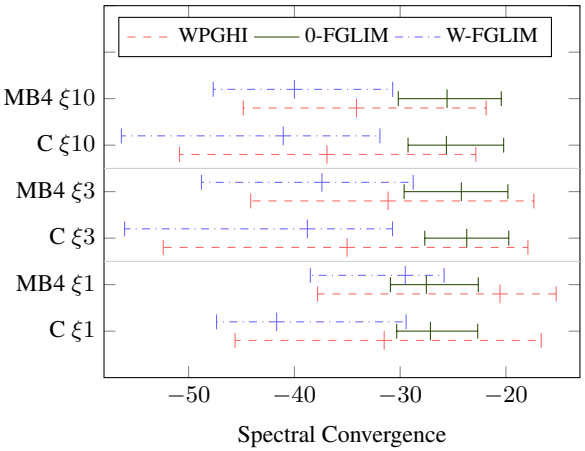


Figure 3: Results of Experiment III for wavelet PGHI and fast Griffin Lim for Cauchy and complex modulated B-spline wavelets. The minimal, median, and maximal spectral convergences over the 15 test signals are depicted.

becomes a soft restriction in practice. In other words, the method can be successfully applied for other types of mother wavelet, provided they are reasonably close to some Cauchy wavelet. Although this *closeness* to a Cauchy wavelet limits the performance of the algorithm when other wavelets are used, the obtained results for a small, but varied corpus of audio data are very promising. Finally, we indicated the steps that are necessary for introducing the proposed method into a bounded delay wavelet analysis/synthesis system, similar to what has been done for the short-time Fourier transform in [28].

Algorithm 2: *Causal* wavelet PGHI for n-th time position

Input: Magnitude $M_s[m, \cdot]$ wavelet coefficients and time-direction phase derivatives $\Delta_{\psi}^{\tilde{\phi},x,s}[m, \cdot]$ for $m \in \{n-1, n\}$. Frequency-direction phase derivatives $\Delta_{\psi}^{\tilde{\phi},\xi,s}[n, \cdot]$, phase estimate $(\tilde{\phi}_{\psi}^{s})_{\text{est}}[n-1, \cdot]$ at time position $n-1$ and relative tolerance tol.

Output: Phase estimate $(\tilde{\phi}_{\psi}^{s})_{\text{est}}[n, \cdot]$ at time position n.

$abstol \leftarrow tol \cdot \max_{m \in \{n-1,n\}} (M_s[n, k])$;
Create set $\mathcal{I} = \{(n, k) : M_s[n, k] > abstol\}$;
Assign random values to $(\tilde{\phi}_{\psi}^{s})_{\text{est}}(n, k)$ for $k \notin \mathcal{I}$;
Construct a self-sorting max *heap* [39] for (n, k) pairs;
for $(n-1, k) \in \mathcal{I}$ **do**
    Move $(n-1, k)$ from $\mathcal{I}$ into the *heap*;
end
while $\mathcal{I}$ is not $\emptyset$ **do**
    if *heap* is empty **then**
        Move $(n_m, k_m) = \arg\max_{(n,k) \in \mathcal{I}} (M_s[n, k])$ from $\mathcal{I}$ into the *heap*;
        $(\tilde{\phi}_{\psi}^{s})_{\text{est}}(n_m, k_m) \leftarrow 0$;
    end
    while *heap* is not empty **do**
        $(n_h, k_h) \leftarrow$ remove the top of the *heap*;
        if $n_h = n - 1$ *and* $(n, k_h) \in \mathcal{I}$ **then**
            Compute $(\tilde{\phi}_{\psi}^{s})_{\text{est}}(n, k_h)$ by means of (19);
            Move (n, k_h) from $\mathcal{I}$ into the *heap*;
        end
        if $n_h = n$ **then**
            foreach $(n_{\text{n}}, k_{\text{n}})$ in $\{(n, k \pm 1)\} \cap \mathcal{I}$ **do**
                Compute $(\tilde{\phi}_{\psi}^{s})_{\text{est}}(n_{\text{n}}, k_{\text{n}})$ by means of (19);
                Move $(n_{\text{n}}, k_{\text{n}})$ from $\mathcal{I}$ into the *heap*;
            end
        end
    end
end

6. REFERENCES

[1] R. D. Nowak, "Wavelet-based Rician noise removal for magnetic resonance imaging," *IEEE Trans. Image Process.*, vol. 8, no. 10, pp. 1408–1419, Oct. 1999.

[2] S. G. Chang, B. Yu, and M. Vetterli, "Adaptive wavelet thresholding for image denoising and compression," *IEEE Trans. Image Process.*, vol. 9, no. 9, pp. 1532–1546, Sep. 2000.

[3] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. Image Process.*, vol. 1, no. 2, pp. 205–220, Apr. 1992.

[4] O. Yilmaz and S. Rickard, "Blind separation of speech mixtures via time-frequency masking," *IEEE Trans. Signal Process.*, vol. 52, no. 7, pp. 1830–1847, Jul. 2004.

[5] S. Chu, S. Narayanan, and C.-C. J. Kuo, "Environmental sound recognition with time–frequency audio features," *IEEE Audio, Speech, Language Process.*, vol. 17, no. 6, pp. 1142–1158, Aug. 2009.

[6] T. Necciari, N. Holighaus, P. Balazs, Z. Průša, P. Majdak, and O. Derrien, "Audlet filter banks: A versatile analysis/synthesis framework using auditory frequency scales," *Appl. Sci.*, vol. 8, no. 1(96), Jan. 2018.

[7] S. Mallat and I. Waldspurger, "Phase retrieval for the Cauchy wavelet transform," *J. Fourier Anal. Appl.*, vol. 21, no. 6, pp. 1251–1309, Dec. 2015.

[8] R. Balan, P. Casazza, and D. Edidin, "On signal reconstruction without phase," *Appl. Comput. Harmon. Anal.*, vol. 20, no. 3, pp. 345–356, May 2006.

[9] A. S. Bandeira, J. Cahill, D. G. Mixon, and A. A. Nelson, "Saving phase: Injectivity and stability for phase retrieval," *Appl. Comput. Harmon. Anal.*, vol. 37, no. 1, pp. 106–125, Jul. 2014.

[10] R. Alaifari, I. Daubechies, P. Grohs, and G. Thakur, "Reconstructing real-valued functions from unsigned coefficients with respect to wavelet and other frames," *J. Fourier Anal. Appl.*, vol. 23, no. 6, pp. 1480–1494, Dec. 2017.

[11] R. Alaifari and P. Grohs, "Phase retrieval in the general setting of continuous frames for Banach spaces," *SIAM J. Math. Anal.*, vol. 49, no. 3, pp. 1895–1911, 2017.

[12] I. Waldspurger, "Phase retrieval for wavelet transforms," *IEEE Trans. Inf. Theory*, vol. 63, no. 5, pp. 2993–3009, May 2017.

[13] J. R. Fienup, "Phase retrieval algorithms: a comparison," *Appl. Opt.*, vol. 21, no. 15, pp. 2758–2769, Aug. 1982.

[14] R. W. Gerchberg and W. O. Saxton, "A practical algorithm for the determination of the phase from image and diffraction plane pictures," *Optik*, vol. 35, no. 2, pp. 237–246, 1972.

[15] E. J. Candes, T. Strohmer, and V. Voroninski, "Phaselift: Exact and stable signal recovery from magnitude measurements via convex programming," *Commun. Pure Appl. Math.*, vol. 66, no. 8, pp. 1241–1274, Aug. 2013.

[16] D. Griffin and J. Lim, "Signal estimation from modified short-time Fourier transform," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 32, no. 2, pp. 236–243, Apr. 1984.

[17] Y. Shechtman, A. Beck, and Y. C. Eldar, "GESPAR: Efficient phase retrieval of sparse signals," *IEEE Trans. Signal Process.*, vol. 62, no. 4, pp. 928–938, Feb. 2014.

[18] Y. Shechtman, Y. C. Eldar, O. Cohen, H. N. Chapman, J. Miao, and M. Segev, "Phase retrieval with application to optical imaging: a contemporary overview," *IEEE Signal Process. Mag.*, vol. 32, no. 3, pp. 87–109, May 2015.

[19] M. R. Portnoff, "Magnitude-phase relationships for short-time Fourier transforms based on Gaussian analysis windows," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, vol. 4, Washington, D. C., USA, Apr 1979, pp. 186–189.

[20] F. Auger, E. Chassande-Mottin, and P. Flandrin, "On phase-magnitude relationships in the short-time Fourier transform," *IEEE Signal Process. Lett.*, vol. 19, no. 5, pp. 267–270, May 2012.

[21] N. Holighaus, G. Koliander, Z. Průša, and L. D. Abreu, "Characterization of analytic wavelet transforms and a new phaseless reconstruction algorithm," *to appear in IEEE Trans. Sig. Process.*, 2019.

[22] I. Daubechies and T. Paul, "Time-frequency localisation operators—a geometric phase space approach: II The use of dilations," *Inverse Prob.*, vol. 4, no. 3, pp. 661–680, Aug. 1988.

[23] G. Ascensi and J. Bruna, "Model space results for the Gabor and wavelet transforms," *IEEE Trans. Inf. Theory*, vol. 55, no. 5, pp. 2250–2259, May 2009.

[24] Z. Průša, P. Balazs, and P. L. Søndergaard, "A noniterative method for reconstruction of phase from STFT magnitude," *IEEE Audio, Speech, Lang. Proc.*, vol. 25, no. 5, May 2017.

[25] Z. Průša and P. L. Søndergaard, "Real-time spectrogram inversion using phase gradient heap integration," in *Proc. of DAFx*, Brno, Czech Republic, Sep. 2016.

[26] Z. Průša and N. Holighaus, "Non-iterative filter bank phase (re)construction," in *Proc. Eur. Signal Process. Conf. EUSIPCO*, Kos island, Greece, Aug 2017, pp. 952–956.

[27] N. Perraudin, P. Balazs, and P. L. Sondergaard, "A fast Griffin-Lim algorithm," in *Proc. IEEE Appl. Sig. Process. Audio Acoustics*, New Paltz, NY, USA, Oct. 2013.

[28] Z. Průša and P. Rajmic, "Toward high-quality real-time signal reconstruction from STFT magnitude," *IEEE Signal Process. Lett.*, vol. 24, no. 6, pp. 892–896, Jun. 2017.

[29] P. Flandrin, "Separability, positivity, and minimum uncertainty in time-frequency energy distributions," *J. Math. Phys.*, vol. 39, no. 8, pp. 4016–4040, Aug. 1998.

[30] O. Christensen, *An Introduction to Frames and Riesz Bases*. Basel, Switzerland: Birkhäuser, 2016.

[31] H. Bölcskei, F. Hlawatsch, and H. G. Feichtinger, "Frame-theoretic analysis of oversampled filter banks," *IEEE Trans. Signal Process.*, vol. 46, no. 12, pp. 3256–3268, Dec. 1998.

[32] Z. Cvetkovic and M. Vetterli, "Oversampled filter banks," *IEEE Trans. Signal Process.*, vol. 46, no. 5, pp. 1245–1255, May 1998.

[33] M. Fickus, M. L. Massar, and D. G. Mixon, "Finite frames and filter banks," in *Finite Frames*, P. G. Casazza and G. Kutyniok, Eds. Basel, Switzerland: Birkhäuser, 2013, pp. 337–379.

[34] T. Strohmer, "Numerical algorithms for discrete Gabor expansions," in *Gabor Analysis and Algorithms*, H. G. Feichtinger and T. Strohmer, Eds. Boston, MA, USA: Birkhäuser, 1998, pp. 267–294.

[35] K. Gröchenig, "Acceleration of the frame algorithm," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3331–3340, Dec. 1993.

[36] P. Balazs, M. Dörfler, F. Jaillet, N. Holighaus, and G. Velasco, "Theory, implementation and applications of nonstationary Gabor frames," *J. Comput. Appl. Math.*, vol. 236, no. 6, pp. 1481–1496, Oct. 2011.

[37] N. Holighaus, M. Dörfler, G. A. Velasco, and T. Grill, "A framework for invertible, real-time constant-Q transforms," *IEEE Audio, Speech, Language Process.*, vol. 21, no. 4, pp. 775–785, Apr. 2013.

[38] C. Schörkhuber, A. Klapuri, N. Holighaus, and M. Dörfler, "A Matlab toolbox for efficient perfect reconstruction time-frequency transforms with log-frequency resolution," in *Proc. AES Conf. Semantic Audio*, London, UK, Jan. 2014.

[39] J. W. J. Williams, "Algorithm 232: Heapsort," *Communications of the ACM*, vol. 7, no. 6, pp. 347–348, Jun. 1964.

[40] "Tech 3253: Sound Quality Assessment Material recordings for subjective tests," Eur. Broadc. Union, Geneva, Tech. Rep., Sept. 2008.

[41] Z. Průša, P. L. Søndergaard, N. Holighaus, C. Wiesmeyr, and P. Balazs, "The large time-frequency analysis toolbox 2.0," in *Sound, Music, and Motion*, M. Aramaki, O. Derrien, R. Kronland-Martinet, and S. Ystad, Eds. Cham, Switzerland: Springer, 2014, pp. 419–442.

[42] S. C. Olhede and A. T. Walden, "Generalized Morse wavelets," *IEEE Trans. Sig. Process.*, vol. 50, no. 11, pp. 2661–2670, Nov. 2002.

[43] R. X. Gao and R. Yan, *Wavelets: Theory and applications for manufacturing*. Springer Science & Business Media, 2010.

[44] N. Sturmel and L. Daudet, "Signal reconstruction from STFT magnitude: A state of the art," in *Proc. Int. Conf. Digital Audio Effects*, Paris, France, Sep. 2011, pp. 375–386.

HIGH-DEFINITION TIME-FREQUENCY REPRESENTATION BASED ON ADAPTIVE COMBINATION OF FAN-CHIRP TRANSFORMS VIA STRUCTURE TENSOR

Maurício do Vale Madeira da Costa *

LTCI, Télécom Paris
Institut Polytechnique de Paris
Paris, France
SMT, PEE/COPPE
Federal University of Rio de Janeiro
Rio de Janeiro, Brazil
mauricio.costa@smt.ufrj.br

Luiz Wagner Pereira Biscainho *

SMT, DEL/Poli & PEE/COPPE
Federal University of Rio de Janeiro
Rio de Janeiro, Brazil
wagner@smt.ufrj.br

ABSTRACT

This paper presents a novel technique for producing high-definition time-frequency representations by combining different instances of short-time fan-chirp transforms. The proposed method uses directional information provided by an image processing technique named structure tensor, applied over a spectrogram of the input signal. This information indicates the best analysis window size and chirp parameter for each time-frequency bin, and feeds a simple interpolation procedure, which produces the final representation. The method allows the proper representation of more than one sound source simultaneously via fan-chirp transforms with different resolutions, and provides a precise reproduction of transient information. Experiments in both synthetic and real audio illustrate the performance of the proposed system.

1. INTRODUCTION

Time-Frequency Representations (TFRs) of audio signals have been used for decades in all sorts of audio processing tasks. Such representations allow to observe the temporal evolution of the frequency content present in a given signal by applying a time-to-frequency mapping, e.g. the Fourier transform, to time frames of the signal [1]. In this context, the availability of an appropriate time-frequency representation for a song recording can improve several tasks in Music Information Retrieval (MIR) [2, 3], e.g. automatic transcription, rhythmic analysis, identification of instruments, sound source separation, etc.

The main problem concerning TFRs is the incapability of representing both time and frequency content with arbitrarily high resolutions simultaneously, which is dictated by the uncertainty principle [1]. For instance, by increasing the length of the analysis window of a spectrogram, a higher frequency resolution is achieved, while the time resolution decreases. Therefore, one should choose the analysis window's length within a compromise.

Having high-definition TFRs is essencial for many MIR tasks [2, 3]. Although the uncertainty principle holds true, the TFR field has been receiving several contributions aiming at providing better time-frequency resolution (i.e. sparser representations) for audio tasks. A common approach is to perform some sort of combination of TFRs, taking advantage of characteristics of audio signals, in an attempt to locally optimize their representation [4, 5, 6, 7, 8, 9].

The Fan-Chirp Transform (FChT) [10, 11], in its discrete short-time version, can provide a sparse representation for signals containing harmonic content with fast fundamental frequency variation. This transform, presented here in Section 2, uses a basis composed of complex exponentials whose frequency varies linearly in time, hence assuming linearity within the excerpts (time frames) under analysis. When the transform properly models the variation present in the signal, its tonal frequency content can be sparsely described. This potentially overcomes the problem of energy smearing observed when a signal with fast frequency variations is represented by a spectrogram.

The main disadvantage of this transform is that it may only well represent one sound source at a time, since the whole exponential basis used follows a unique fundamental frequency variation rate. Unless the sources share the same slope parameter, an only source will be sparsely represented in detriment to the others. Therefore, the problem of dealing with multiple sources can only be tackled by combining multiple instances of Short-Time Fan-Chirp Transform (STFChTs), in a way that the best representations for each time-frequency bin remain in the final TFR.

To the authors' knowledge, unfortunately no TFR found in the literature seems capable of precisely representing polyphonic signals containing fast frequency variation and still preserve a natural and smooth representation of the magnitude of frequency lines. When the standard spectrogram is used for this matter, especially with large analysis windows, a blurred representation results whenever a frequency line presents a steep slope. For instance, this undesired effect is frequently observed in vocal signals, whose frequency content changes considerably fast. Having an appropriate representation is an obvious requirement of tasks involving signals with such characteristics.

In this work, we combine bins of different precomputed TFRs using the information provided by an image processing technique, namely, the structure tensor. This technique (presented in Section 3) allows to compute the predominant orientation angle of edges present in a given image [12, 13, 14, 15]. In the case of musical signals, the frequency lines found in a spectrogram can be interpreted as edges, whose direction can then be estimated. This procedure has two outputs, for each time-frequency bin (pixel): a priority angle, which indicates the direction of the edge; and the anisotropy measure, which informs how relevant is that direction

* The authors thank CAPES and CNPq Brazilian agencies for funding this work.

in terms of edginess. We use the information provided by the structure tensor to choose the best TFRs available (in terms of window size and slope parameter) and combine them bin-wise. This procedure allows one to attain a TFR with high resolution in frequency lines with any desired slope and in sharp transient information.

A flow-chart of the proposed method is depicted in Figure 1. Firstly, the audio signal x is processed to generate a set of different STFChTs using predetermined sets of chirp slopes and analysis window sizes, $\boldsymbol{\alpha}$ and $\mathbf{K}$, respectively. Then, all TFRs are interpolated[1] and assembled in a four-dimensional tensor $\underline{\mathbf{X}}$. From the structure tensor of the standard spectrogram $\mathbf{X}$, parameters $\mathbf{A}$, containing the preferable chirp rates (directional information), and $\mathbf{C}$, containing the preferable window sizes, are computed. Finally, a simple linear combination of the set of TFRs is performed according to $\mathbf{A}$ and $\mathbf{C}$ for each time-frequency bin (as described in Section 4), resulting in the combined TFR $\mathbf{X}^{\mathrm{Comb}}$.

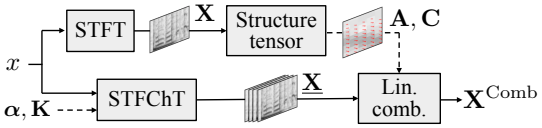


Figure 1: *Flow-chart of the proposed method.*

A set of experiments presented in Section 5 attest the method's performance, and conclusions are drawn in Section 6.

2. TIME-FREQUENCY REPRESENTATIONS

2.1. The Spectrogram

The most used time-frequency transform is the popular spectrogram, which is comprised of the magnitude[2] of the Short-Time Fourier Transform (STFT):

$$X(\tau, f) \triangleq \left| \int_{-\infty}^{\infty} x(t) w(t-\tau) \mathrm{e}^{-j2\pi f t} \mathrm{d}t \right|, \tag{1}$$

where $|\cdot|$ denotes the magnitude of its complex argument, w is a real-valued analysis window, and x is a real valued signal. The window w can be continuously shifted in time, providing a spectrum for each instant τ of the windowed version of $x(t)$.

The discrete version of the spectrogram, $\mathbf{X} \in \mathbb{R}^{K \times M}$, follows the same principle of using a window to focus on some part of the signal. Considering that the time support of the analysis window is limited to K samples, the discrete spectrogram can be described by

$$X_{k,m} \triangleq \left| \sum_{n=0}^{K-1} x_{n-hm} w_n \mathrm{e}^{-j\frac{2\pi}{K}kn} \right|, \tag{2}$$

where $k \in \mathcal{K} \triangleq \{0, 1, 2, ..., K/2\}$ is the frequency index, $m \in \mathcal{M} \triangleq \{1, 2, 3, ..., M\}$ is the time index of the STFT, w_n is the analysis window with K samples used for computation of the spectrogram, and $h \in \mathbb{N}$ is the analysis hop.

As mentioned, this method, as well as any other TFR, has the limitation stated by the uncertainty principle [1]: a signal cannot be represented with arbitrarily high time and frequency resolutions simultaneously. As the length of the analysis window grows, a greater frequency resolution is achieved, as longer excerpts of the signal are projected into the complex exponentials. For the same reason, the time resolution decreases. Parts of the signal with fast variations become blurred in the time-frequency map, for they are integrated with their neighborhood. Another issue is related to frequency variation within the period of the analysis window, which spreads the energy frequency-wise.

In order to address this issue, the fan-chirp transform can be used, allowing for representing harmonic signals whose fundamental frequency varies linearly in time. As long as the analysis window is short enough for the signal to fit this model, a sparse representation of fast frequency variations is attained.

2.2. The Fan-Chirp Transform

In the continuous time domain, the fan-chirp transform $X^{\mathrm{FChT}}(f, \alpha)$ of a given signal $x(t)$ is defined in [11] as

$$X^{\mathrm{FChT}}(f, \alpha) \triangleq \int_{-\infty}^{\infty} x(t) \phi'_\alpha(t) \mathrm{e}^{-\mathrm{j}2\pi f \phi_\alpha(t)} \mathrm{d}t, \tag{3}$$

where $\phi_\alpha(t)$ is a time linear warping function given by

$$\phi_\alpha(t) = \left(1 + \frac{1}{2}\alpha t\right) t, \tag{4}$$

and α is the chirp rate parameter.

This transform can be interpreted as a modification of the Fourier transform (which corresponds to $\alpha = 0$). By applying the variable change $\tau = \phi_\alpha(t)$ to Equation (3), the time domain itself can be warped:

$$X^{\mathrm{FChT}}(f, \alpha) = \int_{-1/\alpha}^{\infty} x(\phi_\alpha^{-1}(\tau)) \mathrm{e}^{-\mathrm{j}2\pi f \tau} \mathrm{d}\tau, \tag{5}$$

where

$$\phi_\alpha^{-1}(t) = -\frac{1}{\alpha} + \frac{\sqrt{1 + 2\alpha t}}{\alpha}. \tag{6}$$

In Equation (5), it is possible to observe that the FChT has the same formulation of the Fourier transform (Equation (1)), with the differences that the input signal $x(t)$ is pre-warped in time, and the inferior integration limit is changed—in order to avoid aliasing, $x(t) = 0$ for $t \leq -1/\alpha$ [10] should be assured. This constrains the usable values of α inside an analysis window with K samples to be within the interval

$$-2F_\mathrm{s}/K \leq \alpha \leq 2F_\mathrm{s}/K. \tag{7}$$

The Short-Time Fan-Chirp Transform (STFChT) $\mathbf{X}^{\mathrm{FChT}}$ is implemented by applying the same windowing procedure employed to compute the spectrogram, after resampling [11] the input signal x, and can be described as

$$X^{\mathrm{FChT}}_{k,m,\alpha} \triangleq \left| \sum_{n=0}^{K-1} \tilde{x}_{\alpha, n-hm} w_n \mathrm{e}^{-j\frac{2\pi}{K}kn} \right|, \tag{8}$$

where $\tilde{x}_n$ is the discrete version of the time warped signal $x(\phi_\alpha^{-1}(\tau))$, as long as the aliasing condition has been satisfied.

[1] The different TFRs must be interpolated not only to have the same dimensions, but also share the same time and frequency axes.

[2] The spectrogram can also be defined as the squared-magnitude of the STFT.

In practice, since $x(t)$ is not available, $\tilde{x}_n$ must be obtained by resampling x_n. With this formulation, the FChT can profit from a fast implementation of the Discrete Fourier Transform (DFT), i.e. an FFT algorithm [11].

By applying this procedure, the input signal is modeled as a sum of linear chirps for each time frame, and to this end, a value of α must be estimated. This step is originally performed via an exhaustive search, in which a predetermined set of values of α is tested, and the choice of the best one is made by searching for the α which provides maximum sparsity. Another reliable and significantly faster way to perform this estimation is proposed in [15], which consists in using the structure tensor [12, 13] technique to estimate priority directions. This procedure can also be used to estimate multiple simultaneous directions, which is useful for signals with more than one sound source [15]. In this paper, the structure tensor will also be used to estimate the target α's; however, differently from what is done in [15], the estimations are performed locally in the TFR, and are used to guide a combination procedure.

3. THE STRUCTURE TENSOR

The structure tensor technique allows the computation of angles of edges present in a given image [12, 13, 14, 15]. The idea is to interpret the spectrogram[3] as an image whose pixels are its time-frequency bins. In this work, the absolute value of the spectrogram compressed by the fourth-root $\hat{\mathbf{X}} = \mathbf{X}^{\frac{1}{4}}$ is used instead of the logarithm, which will be explained latter in this section.

3.1. Computation of the Structure Tensor

Initially, two derivative versions of $\hat{\mathbf{X}}$ are computed by the application of partial derivatives with respect to time index m and frequency index k:

$$\hat{\mathbf{X}}^{\mathrm{m}} = \mathbf{X} * \mathbf{D}, \tag{9}$$

$$\hat{\mathbf{X}}^{\mathrm{k}} = \mathbf{X} * \mathbf{D}^{\mathrm{T}}, \tag{10}$$

where $\mathbf{D}$ is a discrete differentiation operator, more specifically the Sobel-Operator

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \tag{11}$$

and $*$ denotes the 2-dimensional convolution.

Then, $\hat{\mathbf{X}}^{\mathrm{m}}$ and $\hat{\mathbf{X}}^{\mathrm{k}}$ are combined, producing 4 other matrices:

$$\mathbf{T}^{11} = [\hat{\mathbf{X}}^{\mathrm{m}} \odot \hat{\mathbf{X}}^{\mathrm{m}}] * \mathbf{G} \tag{12}$$

$$\mathbf{T}^{12} = \mathbf{T}^{21} = [\hat{\mathbf{X}}^{\mathrm{k}} \odot \hat{\mathbf{X}}^{\mathrm{m}}] * \mathbf{G} \tag{13}$$

$$\mathbf{T}^{22} = [\hat{\mathbf{X}}^{\mathrm{k}} \odot \hat{\mathbf{X}}^{\mathrm{k}}] * \mathbf{G}, \tag{14}$$

where operator $\odot$ denotes the Hadamard product (i.e., point-wise matrix multiplication), and matrix $\mathbf{G}$ is a 2-D Gaussian smoothing filter with standard deviations σ_{m} and σ_{k} in time- and frequency-index directions, respectively, intended to reduce noise interference. Matrix $\mathbf{T}^{11}$ contains information related to temporal (horizontal) variation in the image, $\mathbf{T}^{22}$ contains information about frequency (vertical) variation, and $\mathbf{T}^{12}$ and $\mathbf{T}^{21}$ convey both.

[3] In order to have more consistent results, the input signal is energy-normalized.

Now, each time frequency bin (k, m) has a group of four other values related to it: $T^{11}_{k,m}$, $T^{12}_{k,m}$, $T^{21}_{k,m}$, and $T^{22}_{k,m}$. Together, such bins form a structure tensor element $\mathbf{T}_{k,m}$, which is a 2×2 symmetric and positive semi-definite matrix:

$$\mathbf{T}_{k,m} = \begin{bmatrix} T^{11}_{k,m} & T^{12}_{k,m} \\ T^{21}_{k,m} & T^{22}_{k,m} \end{bmatrix}. \tag{15}$$

This matrix, whose values depend on the time-frequency bin under analysis of the given spectrogram, has interesting properties, since it carries information regarding amplitude variation in different directions. By computing its eigenvalues and eigenvectors, the direction of frequency lines near the analyzed time-frequency bin can be estimated, as well as the anisotropy measure, which indicates the degree of edginess of the given bin, as shown in the following section.

3.2. Computation of Angles and Anisotropy Measure

As mentioned, the information required to compute the angle and the anisotropy of a given time-frequency bin (k, m) is embedded in the eigenvalues and eigenvectors of the structure tensor element $\mathbf{T}_{k,m}$. Consider the eigenvalues $\lambda_{k,m}$ and $\mu_{k,m}$ of $\mathbf{T}_{k,m}$, with $\lambda_{k,m} \leq \mu_{k,m}$, and their respective eigenvectors $\mathbf{v}_{k,m}$ and $\mathbf{w}_{k,m}$. Since $\mathbf{v}_{k,m} = [v^1_{k,m}, v^2_{k,m}]^{\mathrm{T}}$ is related to the smallest eigenvalue, it is pointing in the direction of the smallest change, i.e. parallel to the direction of a frequency line near bin (k, m). Then, the angle of orientation $\theta_{k,m}$, in a horizontal perspective, is given by

$$\theta_{k,m} = \arctan\left(\frac{v^2_{k,m}}{v^1_{k,m}}\right) \in [-\pi/2, \pi/2], \tag{16}$$

with $v^1_{k,m}$ being the horizontal (temporal) component and $v^2_{k,m}$ being the vertical (frequency) component of $\mathbf{v}_{k,m}$.

The eigenvalues can also indicate the edginess of each bin (k, m) by informing how different from each other are the changes in the directions of the eigenvectors. This is called the anisotropy measure $C_{k,m} \in [0, 1]$, defined as

$$C_{k,m} = \begin{cases} \left(\dfrac{\mu_{k,m} - \lambda_{k,m}}{\mu_{k,m} + \lambda_{k,m}}\right)^2, & \mu_{k,m} + \lambda_{k,m} \geq \varepsilon \\ 0, & \text{else,} \end{cases}$$

where $\varepsilon \in \mathbb{R}^+$ is a threshold used to limit the range of what should be considered anisotropic [14], in order to increase robustness against background noise.

Bins within a more homogeneous neighborhood yield smaller values of $C_{k,m}$, while bins close to frequency lines in the spectrogram yield higher values of $C_{k,m}$. Here is where the use of the fourth-root compression is useful, since it presents much reduced dynamic range for amplitude variations at bins with small magnitude, when compared to the logarithm. With the fourth root compression, small magnitude values in the spectrogram, e.g. background noise, will lead to much smaller anisotropy values. A similar result could be obtained by applying an offset of 1 to the magnitude spectrogram before applying the logarithm.

3.3. Computation of α

Since the angles θ are related to the time-frequency bins of the given spectrogram, they live in the discrete time-frequency domain. Nevertheless, the fan-chirp transform is computed using α,

which is related to the analog time-frequency domain; therefore, a transformation must be performed in order to compute the set of α's from a set of θ's.

Let the angle ϑ be the continuous time-frequency domain version of the angle θ, and vector $\boldsymbol{\nu} = [\nu^1, \nu^2]^{\mathrm{T}}$ the continuous time-frequency domain version of vector $\mathbf{v} = [v^1, v^2]^{\mathrm{T}}$. This last conversion can be computed by $\nu^1 = v^1 h/F_{\mathrm{s}}$ and $\nu^2 = v^2 F_{\mathrm{s}}/K$, where F_{s} is the sampling rate, h is the hop-size of the STFT, and K is the number of samples used in the Fourier transform. Figure 2 depicts the geometrical relation between ϑ and α.

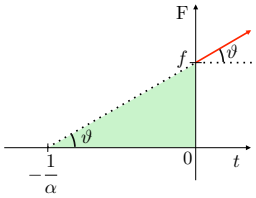


Figure 2: *Geometrical relation between the orientation angle ϑ and variable α, in the continuous time-frequency domain.*

By analyzing the triangle highlighted in green in Figure 2, one can verify that

$$\tan\vartheta = f\alpha, \tag{17}$$

which using the aforementioned conversions is written as

$$\tan\vartheta = \frac{\nu^2}{\nu^1} = \frac{v^2 F_{\mathrm{s}}/K}{v^1 h/F_{\mathrm{s}}} = \tan\theta \frac{F_{\mathrm{s}}^2}{Kh}. \tag{18}$$

Using the relation $f = kF_{\mathrm{s}}/K$,

$$\alpha_{k,m} = \tan\theta_{k,m} \frac{F_{\mathrm{s}}}{hk}. \tag{19}$$

Therefore, by performing this conversion one has a set of α parameters for each time-frequency bin (k, m) of the spectrogram.

4. COMBINATION METHOD

4.1. Principles of the Method

The structure tensor outputs, i.e. the set of angles θ and the set anisotropy measures C, comprise the information of direction and proximity to a straight edge of each time-frequency bin. Figure 3 depicts a small region of the spectrogram of an audio signal with blue arrows representing vectors pointing at direction θ, and having magnitude C. It is possible to observe that the arrows correctly follow the direction of frequency lines, and that the regions presenting only background noise, far from the frequency lines, exhibit no arrows ($C = 0$). In Figure 3, two different regions are highlighted: the arrows inside region 1 present smaller magnitude than the ones inside region 2. This occurs because the latter is surrounded by a much more linear frequency line excerpt than the former, and linear edges provide maximum difference between the eigenvalues. This effect depends on the dimensions of the smoothing filter $\mathbf{G}$ (Section 3.1): a small-dimension $\mathbf{G}$ induces smaller regions, which favor a linear model, and thus decreases the effect. Also, it is worth noting that the magnitudes vary smoothly over the whole time-frequency domain, which will assure smooth transitions between different TFRs in the combined result.

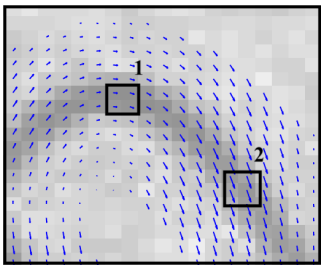


Figure 3: *Vectors in $\theta_{k,m}$ directions with magnitudes $C_{k,m}$.*

Note that the discrete fan-chirp transform models the input signal as a series of harmonically related linear frequency chirps, which means that the resulting TFR will present sparse results when the input signal matches this model within the analysis window period. As a result, using a larger analysis window allows the increase of the number of frequency chirp bins in the transform, providing minimum energy smearing only if the signal under analysis is indeed linearly varying with slope α for a longer period.

Such observations are key to the strategies used in the proposed combination procedure. The idea is to use the anisotropy measure as an indicator of the local linearity of frequency lines, and therefore an indicator of the analysis window length to be applied; and, in order to choose the best fan-chirp representation for each time-frequency bin, parameter α can then be inferred from the angle θ obtained for each bin. In the end, the method consists in performing a linear combination of time-frequency bins of the best candidates among a set of STFChTs with different α's and analysis window lengths K.

4.2. Computation of Tensor $\underline{\mathbf{X}}$

The tensor $\underline{\mathbf{X}}$ comprises TFRs which will be used in the combination. The objective is to span a broad variety of TFRs for audio signals. Three general situations can be observed in musical audio signals: (i) some sort of broadband noise produced, for instance, by blows, brushes in drums, fricative syllables in vocals, or just background noise; (ii) percussive information, as that contained in the attack of a note or a drum hit; and (iii) tonal information, possibly varying continually over time, as in the case of an instrument performing a vibrato. Figure 4 depicts the spectrogram of the onset of a harmonic pulse, zoomed in a region close to the attack. From left to right, it is possible to observe three distinct regions: background noise, the attack, and tonal information. Note that the angles computed by the structure tensor are very close to $\pi/2$ or $-\pi/2$ at the attack, indicating that the energy is distributed vertically.

Since the attacks are much better defined by transforms using short analysis windows, it is useful to define a maximum angle above which transient information should be considered predominant. This angular threshold $\vartheta^{\max}$ is then chosen in order to define two different regions: angles that represent attacks, for which STFTs with short windows will be used in the combination procedure, and angles that indicate the presence of tonal information, which will be represented by STFChTs with proper window length and parameter α. These two angular regions are indicated in Figure 5, similarly to what is done in [14].

For computation of the optimum α's distribution, an equally spaced distribution of angles ϑ is adopted, in order to minimize

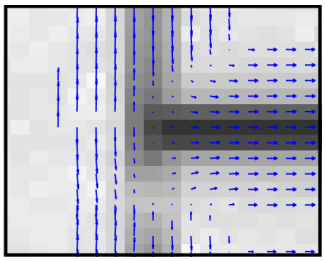


Figure 4: *Spectrogram: onset of a harmonic pulse. Vectors in $\theta_{k,m}$ directions with magnitudes $C_{k,m}$.*

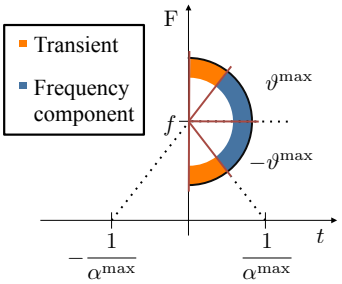


Figure 5: *Angular regions associated to transients and tonal information.*

the energy smearing in tonal regions. Consider that the angular region $[0, \vartheta^{\max}]$ will be divided in I parts. This maximum analog angle is related to an $\alpha^{\max}$ by the same relation described in Equation (17), which indicates that the analog angle ϑ is proportional to $\tan\alpha$. Since parameter α better describes the behavior of varying harmonic frequency content,[4] instead of setting a global maximum angle, it is better to consider a global $\alpha^{\max}$. This parameter can be set, for instance, considering Equation (7), since there is a range of α values that can be used given the analysis window size and the sampling frequency.

Now, angles $\vartheta_{k,m}$ that produce $\alpha_{k,m} > \alpha^{\max}$ will be considered transient information, while the others will be considered tonal information. Considering again the relation in Equation (17),

$$\tan(\vartheta^{\max}) = f\alpha^{\max}. \tag{20}$$

Considering a generic f, e.g. $f = 1$, and given $\alpha^{\max}$ and the number of α's I,

$$\vartheta^{\max} = \arctan(\alpha^{\max}), \tag{21}$$

and

$$\vartheta_i = i\frac{\vartheta^{\max}}{I} = i\frac{\arctan(\alpha^{\max})}{I}. \tag{22}$$

Finally, we can project a linear distribution of ϑ into α by computing α_i as

$$\alpha_i = \tan(\vartheta_i) = \tan(i\arctan(\alpha^{\max})/I), \tag{23}$$

and the set of α's that we shall use to compute the STFChT symmetrically spans this distribution with positive and negative values:

$$\boldsymbol{\alpha} = [-\bar{\alpha}_I, -\bar{\alpha}_{I-1}, \ldots, -\bar{\alpha}_1, \bar{\alpha}_0, \bar{\alpha}_1, \ldots, \bar{\alpha}_{I-1}, \bar{\alpha}_I]. \tag{24}$$

[4] Lower frequencies will have much smaller frequency variation than higher frequencies, for they follow a proportional relation.

For choosing the best distribution of $\mathbf{K} = [K_1, K_2, \ldots, K_J]$, since the FFT algorithm is used, having analysis window lengths of powers-of-two is desirable. This criterion is used to choose the elements of $\mathbf{K}$, optimizing this way the computational cost of this step. The parameters to be set are, then, $\mathbf{K}$ following the aforementioned criterion and the number of α values I. A set of TFRs is then composed of several instances of STFChTs using the combinations of $\mathbf{K}$ and $\boldsymbol{\alpha}$ and an STFT computed with K_1 (for the transients). Note that the sets of STFChTs also include spectrograms, since $\mathbf{X}^{\mathrm{FChT}}_{\alpha=0} = \mathbf{X}$.

Then, all the representations suffer two-dimensional linear interpolation in such a way that the highest time and frequency resolutions are preserved. All representations must have K_J frequency bins after the interpolation, and must be synchronized. In the present implementation, the same hop size is used for computing all TFRs, but this does not guarantee by itself the correct time alignment between analysis windows, reason why the time-wise interpolation (or a previous time shift in x) is also necessary. The set of parameters α and C computed via structure tensor procedure must also be interpolated, generating matrices $\mathbf{A}$ and $\mathbf{C}$, respectively. The best results are obtained when the conversion from θ to α is performed before the interpolation.

The last step is to equalize the energy of the TFRs and store them in a four-dimensional tensor $\underline{\mathbf{X}}$, with the element $X_{k,m;j,i}$ being related to the k-th frequency bin, at the m-th time frame, from a representation that has been computed with an analysis window of length K_j and a chirp rate parameter $\bar{\alpha}_i$. Since the transient information will be represented by a spectrogram computed with K_1, it is allocated at the first and at the last positions in dimension α, and therefore it will not be necessary to compute STFChTs using the first and the last values of α, i. e. $\bar{\alpha}_I$ and $-\bar{\alpha}_I$. Figure 6 depicts the tensor $\underline{\mathbf{X}}$, where groups of TFRs with different α's are illustrated clustered according to the original length of their analysis windows, K_j.

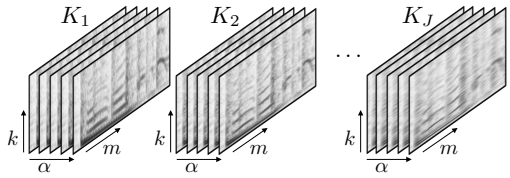


Figure 6: *Scheme of the four-dimensional TFR tensor $\underline{\mathbf{X}}$.*

4.3. Combination Procedure

The combination procedure is independently performed for each time-frequency bin (k, m), using $\alpha_{k,m}$, from $\mathbf{A}$, and $C_{k,m}$, from $\mathbf{C}$. A linear combination is performed using two different weights, one being related to $\alpha_{k,m}$, and another being related to $C_{k,m}$. The idea is to combine the representations that best suit these two parameters by using a simple linear interpolation, which can be represented as triangular complementary functions.

Figure 7 depicts an example of the weights related to the α parameters, λ^{α}, for $I = 2$. The weight λ^{α}_i is applied to the i-th layer of $\underline{\mathbf{X}}$, so the centered weight in the image, λ^{α}_0, in black, is related to the layers in $\underline{\mathbf{X}}$ which were computed with $\bar{\alpha}_0$, the others in blue, λ^{α}_1 and λ^{α}_{-1}, are related to the layers computed with $\bar{\alpha}_1$ and $\bar{\alpha}_{-1}$, and the last ones, λ^{α}_2 and λ^{α}_{-2}, in orange, are related to the STFT, which is used to represent the transients. For this reason, these last curves have a plateau in 1 for representing $\|\alpha\| \geq \bar{\alpha}_2$.

Analogously, λ^C (depicted in Figure 8) will be used for weighting the layers of $\underline{\mathbf{X}}$ along dimension j, which is related to K.

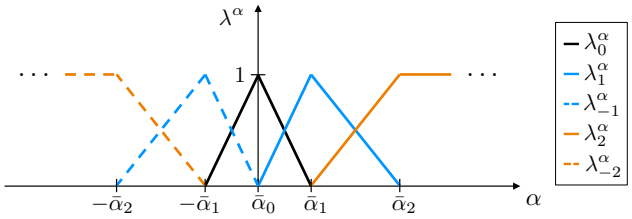


Figure 7: *Example of the weights used for combining TFRs with different α's ($I = 2$).*

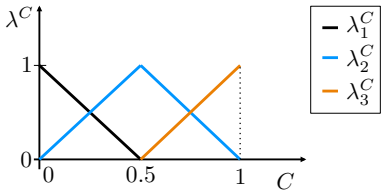


Figure 8: *Example of the weights used for combining TFRs with different K's ($J = 3$).*

The combined TFR $\mathbf{X}^{\text{Comb}}$ is then described by the following two-dimensional interpolation for each time-frequency bin (k, m):

$$X^{\text{Comb}}_{k,m} = \sum_{j=1}^{J} \sum_{i=-I}^{I} \lambda^C_{k,m;j} \lambda^\alpha_{k,m;i} X_{k,m;j,i}. \tag{25}$$

4.4. Practical Considerations

In practice, CPU processing can be saved by using only $\alpha = 0$ for small window sizes, e.g. $K = 1024$ (≈ 21 ms), achieving very similar results. Another practical consideration is regarding the storage of $\underline{\mathbf{X}}$ in memory. Since several TFRs may be stored, it is useful to process the combined TFRs in small excerpts of x, and then concatenate the results, giving a certain time margin to guarantee the proper computation of all TFRs and the structure tensor parameters. Once the combined TFR is processed for that given excerpt, its tensor $\underline{\mathbf{X}}$ is no longer needed, and therefore the memory space can be freed up. Also, in order to reduce backwards smearing of attacks, asymmetrical analysis windows having a longer tail on the left side can be used.

5. EXPERIMENTS

Experiments were conducted in order to assess the performance of the proposed method, using both synthetic and real-world audio signals. All input signals had sampling rate $F_s = 48000$ Hz. The system was set according to the following configuration. In the structure tensor procedure, the analysis windows of the spectrogram had length $K = 1024$ (21.3 ms); in the smoothing two-dimensional filter G, σ_k corresponded to 90 Hz and σ_m to 15 ms; and the threshold used in the anisotropy measure computation was $\varepsilon = 1$. The analysis window sizes for the STFChTs were chosen as $\mathbf{K} = [1024, 2048, 4096]$ (21.3, 42.6 and 85.3 ms); in order to reduce backwards energy smearing, asymmetric[5] analysis windows were used for the computation of STFChTs with K_2 and K_3; $\alpha^{\max} = 23.4$ resulted from the application of Equation (7); and all TFRs were computed with hop size $h = 256$ samples.

5.1. Proof of Concept

As a proof of concept, synthetic signals were selected to asses the method's performance in specific challenging scenarios with regards to time-frequency representations.

First, a pulse comprised of harmonically related sinusoids, with onset at 0.1 s and offset at 0.5 s, contaminated by additive white Gaussian noise (SNR = 50 dB), was used. Figures 9(a) and (b) depict the spectrograms obtained for this signal, using $K_1 = 1024$ and $K_3 = 4096$, respectively the shortest and longest window sizes; and Figures 9(c) and (d) depict the resulting TFRs using the proposed combination procedure, with $I = 1$ and $I = 5$ respectively. Red dashed lines indicate the onset and offset instants to facilitate the visualization. As can be clearly observed, the two TFRs computed with the proposed method yielded nearly identical results, combining the time precision provided by the first spectrogram with the frequency resolution of the second one. Since the frequency lines present in this signal are well represented by an STFChT with $\alpha = 0$ (i.e. a spectrogram), increasing the number of STFChTs available does not affect the result. This could be the case of representing signals of instruments with stable f_0, e.g. piano or harp.

The second example uses a harmonic series whose f_0 varies in a sinusoidal fashion with increasing amplitude, also contaminated by additive white Gaussian noise (SNR = 50 dB). This signal allows one to verify the capability of handling a wide variety of α's. The results are depicted in Figure 10, where it is possible to see the original spectrogram, and three resulting TFRs, computed with $I = 1$, $I = 3$ and $I = 5$. As expected, increasing I also increases the time-frequency resolution, yielding more concentrated and consistent frequency lines. For instance, the results obtained for $I = 3$ and $I = 5$ differ only in the steeper slopes, mainly on the right side of the pictures.

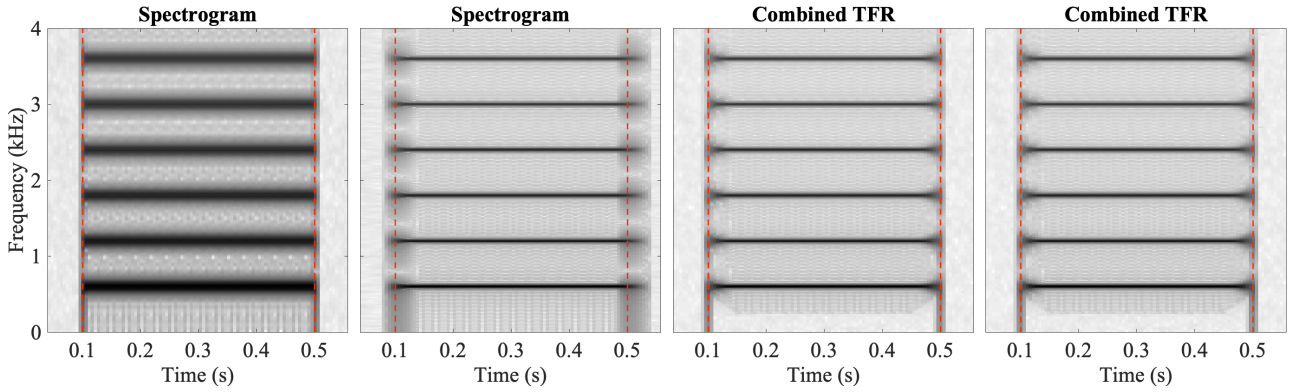
Finally, the last synthetic signal is a sum of two harmonic signals having different sinusoidal variations of f_0, with additive white Gaussian noise (SNR = 50 dB). Figure 11 depicts the spectrogram used for the computation of the structure tensor and the combined TFR, for which $I = 5$ was used. The resulting TFR represents the input signal with a much higher definition, and very smooth transitions can be observed. It is worth highlighting that even at places where more than one frequency line crosses the same bin, the signal is fairly well represented.

5.2. Real-World Signals

The experiments with real-world signals used the MedleyDB [16] dataset. From each track, an excerpt of 10 s containing part of the main melody was selected, so that each song contributed the same amount of data. These signals were divided into 1-s segments, totalling 1210 excerpts. In order to assess the method's performance, the Gini index[6] [17] was chosen as an objective figure-of-merit.

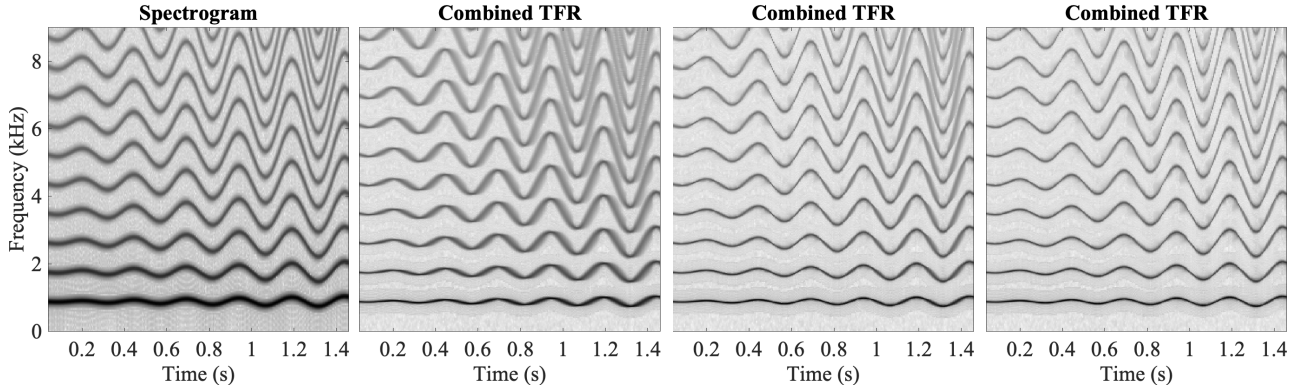
[5] The asymmetric windows are computed by concatenation of the first half of a Hanning window computed with K samples, and the second half of a Hanning window computed with $K/2$ samples.

[6] The Gini index is a measure of sparsity that indicates within the range $[0, 1]$ how concentrated is the energy in a given set of bins.



(a) $K = 1024$. (b) $K = 4096$. (c) $I = 1$. (d) $I = 5$.

Figure 9: *Spectrograms for different K values and combined TFRs with different I values computed for a pulse composed of harmonically related sinusoids. Onset and offset are indicated by the red dashed lines.*



(a) $K = 1024$. (b) $I = 1$. (c) $I = 3$. (d) $I = 5$.

Figure 10: *Varying vibrato: spectrogram and the combined TFRs with different I values.*

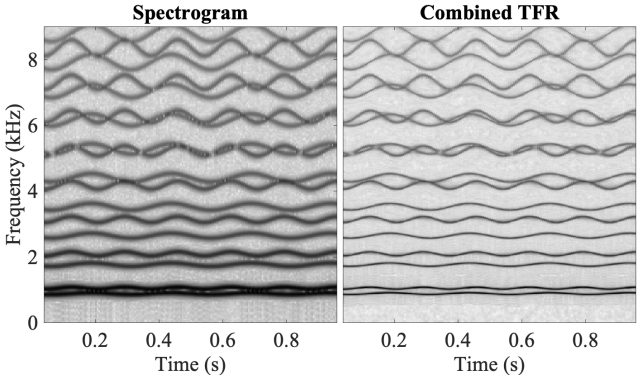


Figure 11: *Simultaneous vibratos: spectrogram ($K = 1024$) and combined TFR ($I = 5$).*

Each audio excerpt was then processed using the proposed method and standard spectrograms (using $K = 1024, K = 2048$ and $K = 4096$), and the Gini index was computed for each resulting TFR.

Figure 12 depicts the percentage of times each representation was ranked in each position according to the Gini index. The combined TFR is by far the most effective in terms of sparsity, ranking first in about 80% of the time. The STFT-1024 accounts for 50% of the second position, the STFT-2048 for 65% of the third, and the STFT-4096 for 50% of the fourth.

Finally, an excerpt from a piano and vocal recording was selected to illustrate how the TFR of a real-world audio signal can be improved by the proposed strategy. Figure 13 depicts its original spectrogram next to its combined TFR. It is possible to verify that in the spectrogram the piano is barely noticeable, while the TFR generated by the proposed method clearly represents both the piano and the singing vocal—which is performing a very fast melisma.

6. CONCLUSIONS

This paper presented a method for producing sparse TFRs by combining different STFChTs using the information provided by the structure tensor. We extract directional information from a spectrogram of the input signal, which guides an interpolation procedure. Experiments comprising synthetic and real recorded audio

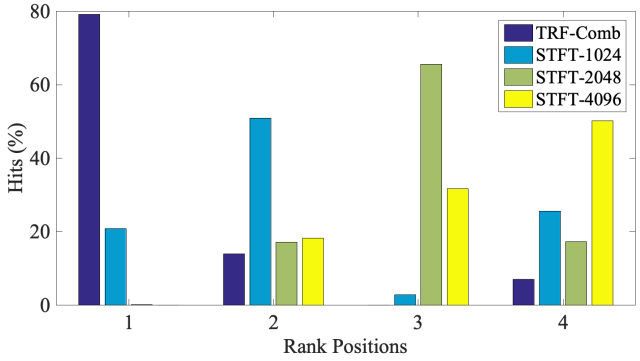


Figure 12: *Rank of different representations in terms of Gini Index for the MedleyDB.*

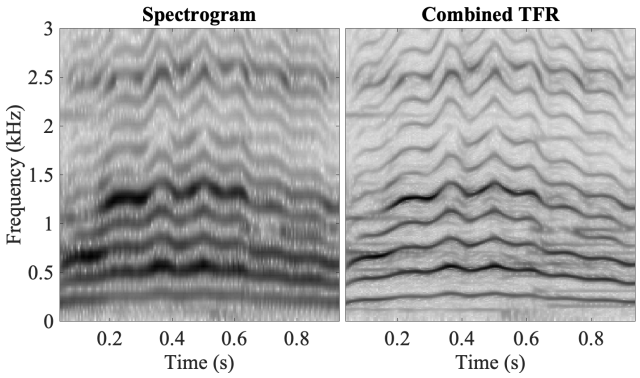


Figure 13: *Vocal and piano: spectrogram ($K = 1024$) and combined TFR ($I = 5$).*

signals suggest that the proposed method provides high-definition TFRs, improving the concentration of frequency lines with various slopes and the definition of transient information, when compared to standard TFRs, e.g. spectrograms. Given the method's ability to provide refined inputs for MIR tasks, such as main melody and multi-pitch extraction, the natural continuation of this research is the reformulation of state-of-the-art methods in MIR to take advantage of such representations in real application scenarios.

7. REFERENCES

[1] L. Cohen, *Time-Frequency Analysis*, Prentice Hall, Englewood Cliffs, USA, 1995.

[2] Emmanouil Benetos, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri, "Automatic music transcription: Challenges and future directions," *Journal of Intelligent Information Systems*, vol. 41, no. 3, pp. 407–434, December 2013.

[3] B. S. Gowrishankar and N. U. Bhajantri, "An exhaustive review of automatic music transcription techniques: Survey of music transcription techniques," in *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*, Paralakhemundi, India, June 2016, pp. 140–152.

[4] Maurício do V. M. da Costa and Luiz W. P. Biscainho, "Combining time-frequency representations for music information retrieval," in *15º Congresso de Engenharia de Áudio da AES-Brasil*, Florianópolis, Brazil, October 2017, pp. 12–18, AES.

[5] Rongping Lin, Chunhui Du, Shan Luo, and Qi Xu, "Performance on a combined representation for time-frequency analysis," in *2nd International Conference on Image, Vision and Computing (ICIVC)*, Chengdu, China, June 2017, pp. 858–862, IEEE.

[6] C. S. Detka, P. Loughlin, and A. El-Jaroudi, "On combining evolutionary spectral estimates," in *IEEE 7th Signal Processing Workshop on Statistical Signal and Array Processing*, Quebec, Canada, June 1994, pp. 243–246, IEEE.

[7] Patrick Loughlin, James Pitton, and Blake Hannaford, "Approximating time-frequency density functions via optimal combinations of spectrograms," *IEEE Signal Processing Letters*, vol. 1, no. 12, pp. 199–202, December 1994.

[8] Alexey Lukin and Jeremy Todd, "Adaptive time-frequency resolution for analysis and processing of audio," in *Audio Engineering Society 120th Convention*, Paris, France, May 2006, AES, Preprint 6717.

[9] François Auger and Patrick Flandrin, "Improving the readability of time-frequency and time-scale representations by the reassignment method," *IEEE Transactions on Signal Processing*, vol. 43, no. 5, pp. 1068–1089, May 1995.

[10] L. Weruaga and M. Képesi, "The fan-chirp transform for non-stationary harmonic signals," *Signal Processing*, vol. 87, no. 6, pp. 1504–1522, June 2007.

[11] Pablo Cancela, Ernesto López, and Martín Rocamora, "Fan chirp transformation for music representation," in *13th International Conference on Digital Audio Effects (DAFx-10)*, Graz, Austria, September 2010, pp. 1–8.

[12] Josef Bigun and Gösta H. Granlund, "Optimal orientation detection of linear symmetry," in *IEEE First International Conference on Computer Vision*, London, UK, June 1987, pp. 433–438.

[13] Hans Knutsson, "Representing local structure using tensors," in *6th Scandinavian Conference on Image Analysis (SCIA)*, Oulu, Finland, June 1989, pp. 244–251.

[14] R. Füg, A. Niedermeier, J. Driedger, S. Disch, and M. Müller, "Harmonic-percussive-residual sound separation using the structure tensor on spectrograms," in *2016 IEEE International Conference on Audio, Speech and Signal Processing (ICASSP)*, Shanghai, China, March 2016, pp. 445–449.

[15] I. F. Apolinário, Maurício do V. M. da Costa, and L. W. P. Biscainho, "Structure tensor applied to parameter estimation in the fan-chirp transform," in *2nd AES Latin American Congress of Audio Engineering*, Montevideo, Uruguay, September 2018.

[16] R. M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. P. Bello, "MedleyDB: A multitrack dataset for annotation-intensive mir research," in *15th International Society for Music Information Retrieval Conference (ISMIR)*, Taipei, Taiwan, October 2014.

[17] Niall Hurley and Scott Rickard, "Comparing measures of sparsity," *IEEE Transactions on Information Theory*, vol. 55, no. 10, pp. 4723–4741, October 2009.

NON-ITERATIVE SOLVERS FOR NONLINEAR PROBLEMS: THE CASE OF COLLISIONS

Michele Ducceschi *

Acoustics and Audio Group
University of Edinburgh
Edinburgh, UK
michele.ducceschi@ed.ac.uk

Stefan Bilbao

Acoustics and Audio Group
University of Edinburgh
Edinburgh, UK
s.bilbao@ed.ac.uk

ABSTRACT

Nonlinearity is a key feature in musical instruments and electronic circuits alike, and thus in simulation, for the purposes of physics-based modeling and virtual analog emulation, the numerical solution of nonlinear differential equations is unavoidable. Ensuring numerical stability is thus a major consideration. In general, one may construct implicit schemes using well-known discretisation methods such as the trapezoid rule, requiring computationally-costly iterative solvers at each time step. Here, a novel family of provably numerically stable time-stepping schemes is presented, avoiding the need for iterative solvers, and thus of greatly reduced computational cost. An application to the case of the collision interaction in musical instrument modeling is detailed.

1. INTRODUCTION

Computer simulation of physical systems is at the core of many disciplines, and physical modeling sound synthesis is no exception. The scope of research into physical modeling has expanded to include the simulation of very complex musical systems, including lumped as well as fully-distributed nonlinearities; musically, many perceptually important phenomena can be viewed as originating from this nonlinear behaviour. A specialised approach is often required at the simulation stage, particularly in ensuring numerical stability—a major concern in the modeling of systems with strong nonlinearities. One approach to the design of numerically-stable schemes is through the use of energy methods. With rare exceptions, however, such designs require the use of iterative numerical schemes, thus increasing computational costs.

Recently, non-iterative numerical integrators for a class of nonlinear ordinary differential equations have been devised through the port-Hamiltonian approach for virtual-analog simulations [1, 2, 3]. The schemes employ a suitable quadratisation of the nonlinear potential [4] which gives, ultimately, an update which can be performed without the use of an iterative method such as, e.g., Newton-Raphson.

In the current work, the possibility of using non-iterative solvers for nonlinear problems is developed further, to the case of a non-invertible potential, and for fully distributed systems described by partial differential equations (PDEs). The focus here will be on collisions, a topic of longstanding attraction for researchers in physical modeling sound synthesis [5, 6, 7, 8, 9, 10, 11, 12, 13]. Collisions here serve as useful and practically important test case, because the method can be applied in the current form to any second-order-in-time system with a non-negative potential energy, and can be extended to higher-order systems. The current method leads to significant speedups, in some cases of one order of magnitude, see [14]. Moreover, with respect to iterative methods, existence and uniqueness of the numerical solution are proven trivially by inspection of the update equation.

The method is described in detail in Section 2. There, an overview is given, along with a numerical experiment dealing with a mass-spring system colliding against a barrier. In Section 3, a case of interest in musical acoustics is studied, i.e. the collision of a musical string against a distributed barrier. Iterative schemes, developed in previous works, are used for bench marking. Novel non-iterative finite difference schemes and modal schemes are developed, showing convergence of all the methods to a common solution. Stability is proven mathematically by energy arguments, and illustrated in a number of numerical experiments.

2. PRELIMINARIES

Before examining a fully distributed system, it is useful to explore the method in the case of a typical lumped system in a mechanical setting, described by a single time-dependent ordinary differential equation (ODE):

$$M\ddot{u} + \phi'(u) = 0 \tag{1}$$

Here, $u = u(t)$ is a displacement of a lumped object of mass M, and as a function of time t. Dots represent derivatives with respect to t. $\phi = \phi(u)$ is a function representing the potential energy of the system, and $\phi' = d\phi/du$. (Note that if $\phi(u) = Bu^2/2$, for some constant $B > 0$, then (1) represents the equation of motion of a simple harmonic oscillator.)

Using the chain rule, it is possible to rewrite (1) as

$$M\ddot{u} + \dot{\phi}/\dot{u} = 0 \tag{2}$$

Conservation of energy may be achieved by directly multiplying both sides of (2) by $\dot{u}$, yielding

$$\frac{d}{dt}\underbrace{\left(\frac{M\dot{u}^2}{2} + \phi\right)}_{\triangleq H(t)} = 0 \tag{3}$$

and hence

$$H(t) = H(0) \triangleq H_0. \tag{4}$$

Non-negativity of the potential energy ϕ reflects a condition for passivity of (1). Under such a condition, the total conserved energy H_0 is non-negative, and one may bound the growth of the state as

$$0 \leq |\dot{u}| \leq \sqrt{2H_0/M} \tag{5}$$

* The author's work was supported by the Leverhulme Trust with an Early Career Fellowship

More restrictive conditions on ϕ (such as radial unboundedness) allow for global asymptotic stability, but the above non-negativity condition will suffice for the present purposes, and allows physically-reasonable solution growth (i.e., inertial drift).

The new schemes proposed in this work are based on an equivalent expression for (1):

$$M\ddot{u} + \psi\psi' = 0 \qquad \text{with} \qquad \psi = \sqrt{2\phi} \tag{6}$$

By means of the chain rule, (6) may be written as

$$M\ddot{u} + \psi\left(\dot{\psi}/\dot{u}\right) = 0 \tag{7}$$

and energy conservation is obtained by multiplication by $\dot{u}$, on both sides, yielding

$$\frac{d}{dt}\underbrace{\left(\frac{M\dot{u}^2}{2} + \frac{\psi^2}{2}\right)}_{\triangleq H(t)} = 0 \tag{8}$$

This form of the energy function is analogous to the form given in [3] developed within the Port-Hamiltonian framework. Clearly, (8) is equivalent to (3), given the definition in (6). Hence, (7) is equivalent to (2), and bounds (5) hold.

Passivity and equivalence of (7) and (2) are possible only under non-negativity of the potential ϕ—an entirely natural requirement. Though various forms of the potential appear in musical acoustics, here, one particular form will be explored, namely:

$$\phi(\eta) = \frac{K}{(\alpha+1)}[\eta]_+^{\alpha+1}, \quad K \geq 0,\ \alpha \geq 1 \tag{9}$$

where $[\eta]_+ \triangleq 0.5(\eta + |\eta|)$ is the *positive part* of η. This potential is clearly non-negative. It has been employed as a penalty-potential in collision models in musical acoustics [15, 8], and it is derived from Hertz's contact law.

2.1. Linear Oscillator With Barrier

In preparation to the fully distributed case, (1) is now equipped with a linear restoring force. Hence

$$M\ddot{u} = -\phi'(\eta) - M\omega_0^2 u \tag{10}$$

with ϕ given by (9), and where

$$\eta = u - b \tag{11}$$

This model describes a mass-spring system, with linear radian frequency $\omega_0 = 2\pi f_0$, with f_0 measured in Hz, colliding from below against a barrier placed at b. This differential equation can be cast in the two equivalent forms, as seen above. These are

$$M\ddot{u} = -\dot{\phi}/\dot{\eta} - M\omega_0^2 u \tag{12a}$$

$$M\ddot{u} = -\psi\left(\dot{\psi}/\dot{\eta}\right) - M\omega_0^2 u \tag{12b}$$

Energy conservation for the above equations reads, respectively,

$$\frac{d}{dt}\left(\frac{M\dot{u}^2}{2} + \frac{M\omega_0^2 u^2}{2} + \phi\right) = 0 \tag{13a}$$

$$\frac{d}{dt}\left(\frac{M\dot{u}^2}{2} + \frac{M\omega_0^2 u^2}{2} + \frac{\psi^2}{2}\right) = 0 \tag{13b}$$

and thus the same bounds as (5) hold in both cases.

2.2. Time Difference Operators

Solutions to (12a) and (12b) are sought by means of appropriate finite difference schemes. Time is discretised by means of a sample rate f_s, yielding a time step $k = 1/f_s$. u^n represents an approximation to the continuous function $u(t)$ at time $t = nk$, for integer n. Finite time difference operators are now introduced.

The identity and temporal shift operators are defined as

$$1u^n = u^n, \quad e_{t+}u^n = u^{n+1}, \quad e_{t-}u^n = u^{n-1} \tag{14}$$

Notice that the similar definitions hold for operators acting on interleaved grid functions [16], as in

$$e_{t+}\psi^{n-1/2} = \psi^{n+1/2}, \quad e_{t-}\psi^{n+1/2} = \psi^{n-1/2} \tag{15}$$

From these, it is possible to define the forward, backward and centred time differences, all approximating a first time derivative, as

$$\delta_{t+} = \frac{e_{t+} - 1}{k}, \quad \delta_{t-} = \frac{1 - e_{t-}}{k}, \quad \delta_{t\cdot} = \frac{e_{t+} - e_{t-}}{2k} \tag{16}$$

An approximation to the second time derivative is constructed by composition of the operators presented above, as

$$\delta_{tt} = \delta_{t+}\delta_{t-} \tag{17}$$

Finally, forward and backward averaging operators may be defined as

$$\mu_{t+} = \frac{e_{t+} + 1}{2}, \quad \mu_{t-} = \frac{1 + e_{t-}}{2} \tag{18}$$

2.2.1. Iterative Conservative Finite Difference Scheme

Following the derivation in [8], a suitable discretisation of (12a) is

$$M\delta_{tt}u^n = -M\omega_0^2 u^n - \frac{\delta_{t+}\phi^{n-1/2}}{\delta_{t\cdot}\eta^n} \tag{19}$$

where

$$\phi^{n-1/2} \triangleq \mu_{t-}\phi(\underbrace{u^n - b}_{\eta^n}) \tag{20}$$

Stability of the scheme may be inferred from energy analysis, after multiplication of both sides of (19) by $\delta_{t\cdot}u^n$. This gives

$$\delta_{t+}\left(\underbrace{\frac{M(\delta_{t-}u^n)^2}{2} + \frac{M\omega_0^2 u^n u^{n-1}}{2} + \phi^{n-1/2}}_{\mathfrak{H}^{n-1/2}}\right) = 0$$

This is clearly a discrete counterpart of (13a). In this case, the non-negativity of the total energy can be assured if and only if [16]

$$\omega_0 < 2f_s \tag{21}$$

Under such condition, one may write

$$0 \leq |\delta_{t-}u^n| \leq \sqrt{2\mathfrak{H}^{1/2}/M} \tag{22}$$

and hence the boundedness of the state follows. Scheme (19) can be written as

$$\underbrace{r - 2u^n + 2u^{n-1} + k^2\omega_0^2 u^n + \frac{k^2}{M}\frac{\phi(r+a) - \phi(a)}{r}}_{G(r)} = 0, \tag{23}$$

where

$$r \triangleq u^{n+1} - u^{n-1}, \quad a \triangleq b - u^{n-1}.$$

Because the unknown r appears implicitly as the argument of ϕ, an iterative root finder (such as Newton-Raphson) must be employed in order to solve $G(r) = 0$ as per (23). The solution to the scheme can be shown to be unique — see [13, 8].

2.2.2. Non-iterative Conservative Finite Difference Scheme

A novel, non-iterative finite difference scheme follows as a suitable discretisation of (12b), as

$$M\delta_{tt}u^n = -M\omega_0^2 u^n - \left(\mu_{t+}\psi^{n-1/2}\right)\frac{\delta_{t+}\psi^{n-1/2}}{\delta_{t\cdot}\eta^n} \tag{24}$$

This may be rewritten as the following system:

$$M\delta_{tt}u^n = -M\omega_0^2 u^n - \left(\mu_{t+}\psi^{n-1/2}\right) g^n \tag{25a}$$

$$\delta_{t+}\psi^{n-1/2} = g^n \delta_{t\cdot}\eta^n \tag{25b}$$

where g^n may be explicitly computed as

$$g^n = \psi'\Big|_{\eta=\eta^n} = \frac{\phi'}{\sqrt{2\phi}}\Big|_{\eta=\eta^n} \tag{26}$$

Here, we may use the analytic expressions for ψ and ϕ directly in the computation of g^n. Stability of the scheme can be deduced from an energy conservation law, obtained after multiplication of (24) by $\delta_{t\cdot}u^n$. Energy is thus conserved according to

$$\delta_{t+}\left(\underbrace{\frac{M(\delta_{t-}u^n)^2}{2} + \frac{M\omega_0^2 u^n u^{n-1}}{2} + \frac{(\psi^{n-1/2})^2}{2}}_{\mathfrak{H}^{n-1/2}}\right) = 0$$

Inspection of the energy function allows to infer the same stability condition as (21). Thus, the same bounds as (22) hold in this case. Using the identity

$$\mu_{t+}\psi^{n-1/2} = \frac{k}{2}\delta_{t+}\psi^{n-1/2} + \psi^{n-1/2} \tag{27}$$

one may insert the value of $\delta_{t+}\psi^{n-1/2}$ from (25b) into (25a), to get

$$Au^{n+1} = v \tag{28}$$

where

$$A = \frac{M}{k^2} + \frac{(g^n)^2}{4}$$

$$v = \frac{M}{k^2}(2u^n - u^{n-1}) - M\omega_0^2 u^n + \frac{(g^n)^2}{4}u^{n-1} - \psi^{n-1/2}g^n$$

This scheme can thus be solved by division, and once u^{n+1} is computed, from (25a), one can update $\psi^{n+1/2}$ using (25b).

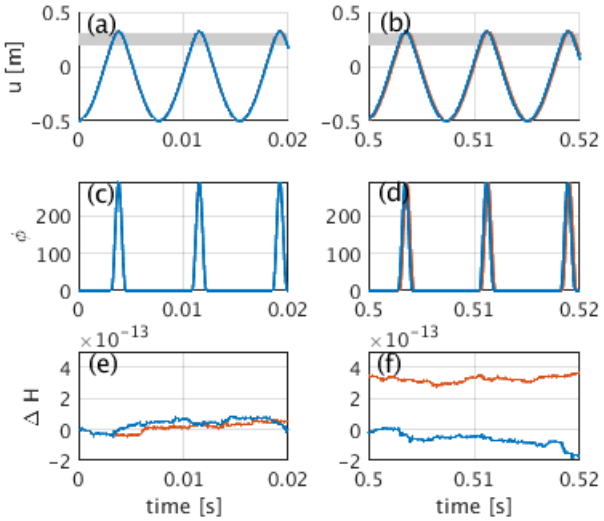


Figure 1: For all panels, the red line is the solution obtained via the iterative scheme (with 20 iterations), and the blue line is the solution obtained via the new scheme. A particle of mass $M = 10$g is launched with initial speed $v_0 = 1$m/s from $u_0 = -0.5$m against a rigid barrier with $\alpha = 1.1$, $K = 5 \cdot 10^4$, located at $b = 0.2$m. The particle is subjected to a linear restoring potential of frequency $f_0 = 10$Hz. The sample rate is chosen as $f_s = 44100$Hz. (a)-(b): displacement vs time, at times indicated. (c)-(d): potential ϕ, at times indicated. (e)-(f): energy variation $\Delta H = (\mathfrak{H}^{n-1/2} - \mathfrak{H}^{1/2})/\mathfrak{H}^{1/2}$, at times indicated.

2.3. Numerical Examples

The finite difference schemes given above are now compared in cases of practical use. In particular, two barriers with different stiffness constants are considered here. Fig. 1 depicts the case with a softer barrier. From panels (a)-(b), it is seen that the two schemes yield a converged solution for a long period of time. The value of the potential ϕ, visible in panels (c)-(d) is also the same over the same period. Finally, panels (e)-(f) show the variation of the total energy.

Fig. 2 considers the same oscillator, with a harder barrier. In this case, the two schemes yield a different solution. For the non-iterative scheme, the excess kinetic energy is converted to potential energy, so to guarantee energy conservation overall, keeping the scheme stable within the bounds (22), as visible in Fig.2(c)-(d). This is reflected in lower-than-expected collision output velocities. The variation of the total energy for the two schemes is visible in Fig.2(e)-(f). As expected, increasing the sample rate yields a converged solution, as can be seen in Fig. (3).

3. DISTRIBUTED SYSTEMS

In this section, the model schemes illustrated in the previous section are applied to cases of interest in musical acoustics, for advanced sound synthesis of string instruments. The system under study is composed of a taut string with stiffness, and of a rigid barrier. The motion of the string, of length L, can be described by

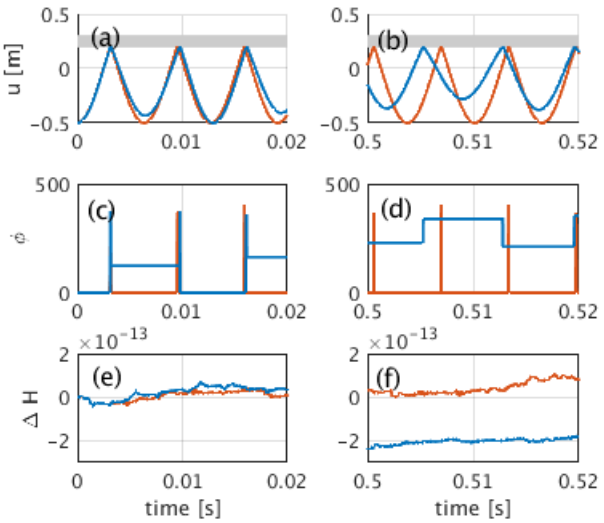


Figure 2: Same as Fig. 1, but with $K = 7 \cdot 10^7$.

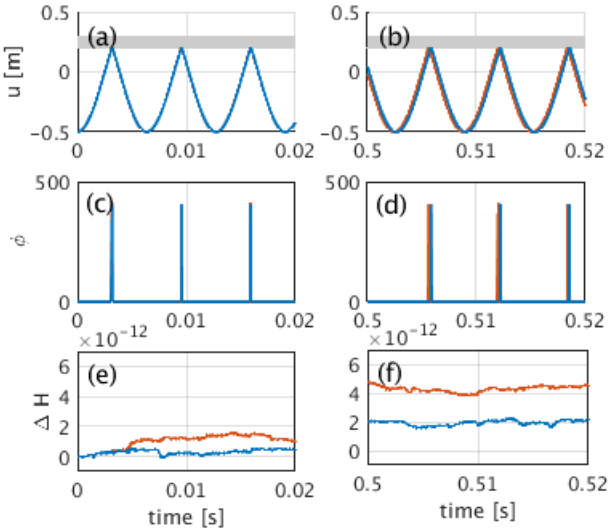


Figure 3: Same as Fig. 2, but with $f_s = 5 \cdot 44100$Hz

either one of the following equivalent equations

$$\rho\partial_t^2 u = \partial_x^2 T_0 u - \partial_x^4 EIu + \frac{\partial_t \phi}{\partial_t \eta} + \delta(x - x_F)F(t) \tag{29a}$$

$$\rho\partial_t^2 u = \partial_x^2 T_0 u - \partial_x^4 EIu + \psi\frac{\partial_t \psi}{\partial_t \eta} + \delta(x - x_F)F(t) \tag{29b}$$

where again

$$\psi = \sqrt{2\phi} \tag{30}$$

In the equations, $u = u(x,t)$ is the displacement of the string, now a function of both time and space. Notice that, having to deal with partial differentiation, the notation for derivatives has changed compared to the lumped case. The string is defined over $\mathcal{D} : x \in [0, L]$. Constants appear as: ρ, the linear density; T_0, the applied tension; E, Young's modulus; I, the moment of inertia of the cross section. The potential ϕ has now units of potential energy per unit length (i.e. energy density) , but its formal definition is again of the form (9), where

$$\eta = b - u \tag{31}$$

In the above, $b = b(x)$ is the height of barrier, supposed unmovable. Finally, $\delta(x - x_F)$ is a Dirac delta function. $F(t)$ will be here expressed as a raised cosine function, i.e.

$$F(t) = \begin{cases} \frac{F_0}{2}\left(1 - \cos\left(\frac{2\pi(t-t_0)}{t_{wid}}\right)\right), & t_0 \le t \le t_0 + t_{wid} \\ 0, & \text{otherwise} \end{cases}.$$

where F_0 and t_{wid} are input parameters controlling, respectively, the maximum amplitude of the forcing, and the contact duration, and t_0 is the activation time.

An inner product of two functions f, g, and the associated norm, are defined as

$$\langle f, g\rangle_{\mathcal{D}} \triangleq \int_0^L f\, g\, dx, \quad \|f\|_{\mathcal{D}}^2 \triangleq \langle f, f\rangle_{\mathcal{D}} \tag{32}$$

Under unforced conditions, i.e. for $F(t) = 0$, energy conservation and boundary conditions can be extracted after taking an inner product of (29a) and (29b) with $\partial_t u$. This gives, respectively,

$$\frac{d}{dt}\left(\underbrace{\frac{\rho\|\partial_t u\|_{\mathcal{D}}^2}{2} + \frac{T_0\|\partial_x u\|_{\mathcal{D}}^2}{2} + \frac{EI\|\partial_x^2 u\|_{\mathcal{D}}^2}{2} + \langle\phi, 1\rangle_{\mathcal{D}}}_{H(t)}\right) = 0 \tag{33a}$$

$$\frac{d}{dt}\left(\underbrace{\frac{\rho\|\partial_t u\|_{\mathcal{D}}^2}{2} + \frac{T_0\|\partial_x u\|_{\mathcal{D}}^2}{2} + \frac{EI\|\partial_x^2 u\|_{\mathcal{D}}^2}{2} + \frac{\|\psi\|_{\mathcal{D}}^2}{2}}_{H(t)}\right) = 0 \tag{33b}$$

where the identities above hold only under a choice of appropriate boundary conditions. Here, conditions of the *simply-supported* kind are enforced, hence

$$u = \partial_x^2 u = 0 \quad \text{at } x = 0, L \tag{34}$$

Given the non-negativity of the energy functions above, one may bound the norm of the state, as

$$0 \le \|\partial_t u\|_{\mathcal{D}} \le \sqrt{2H_0/\rho} \tag{35}$$

where H_0 is the value of the conserved energy (i.e. its initial value).

3.1. Iterative and Non-Iterative Finite Difference Schemes

Finite difference schemes are now constructed for the solution of (29a), (29b). Hence, both time and space are discretised along appropriate grids, and the grid function u_m^n is regarded as an approximation to the solution $u(x,t)$ at the time nk, and at mh, where k is the time step, h is the grid spacing, and m and n are integers. Time difference operators appear as in Section 2.2. Spatial

difference operators have analogous definitions, and hence

$$1u_m^n = u_m^n, \quad e_{x+}u_m^n = u_{m+1}^n, \quad e_{x-}u_m^n = u_{m-1}^n \tag{36}$$

From these, it is possible to define the forward and backward spatial differences, as

$$\delta_{x+} = \frac{e_{x+} - 1}{h}, \quad \delta_{x-} = \frac{1 - e_{x-}}{h} \tag{37}$$

Approximations to the second and fourth spatial differences are constructed from the above, as

$$\delta_{xx} = \delta_{x+}\delta_{x-}, \quad \delta_{xxxx} = \delta_{xx}\delta_{xx} \tag{38}$$

The spatial grid is defined for $\mathbb{M} : m \in \{0, 1..., M\}$. Sets of points lacking at least one end point will also be used: they are $\underline{\mathbb{M}} : m \in \{0..., M-1\}$, and $\overline{\mathbb{M}} : m \in \{1..., M-1\}$. A discrete version of (32), the inner product and associated norm, can be realised using summation. Hence

$$\langle f, g \rangle_{\mathbb{B}} \triangleq \sum_{b \in \mathbb{B}} h\, f_b^n\, g_b^n, \quad \|f\|_{\mathbb{B}}^2 \triangleq \langle u, u \rangle_{\mathbb{B}} \tag{39}$$

Finally, $\delta(x - x_F)$ is approximated by a zeroth-order spreading operator:

$$\mathcal{I}_m(x_F) = \begin{cases} 1/h, & m = m_F = \text{round}\,(x_F/h) \\ 0, & \text{otherwise} \end{cases} \tag{40}$$

3.1.1. Iterative Conservative Finite Difference Scheme

Following the derivation in [8], a conservative, iterative finite difference scheme is constructed as

$$\rho\delta_{tt}u_m^n = \mathfrak{L}u_m^n + \frac{\delta_{t+}\phi_m^{n-1/2}}{\delta_{t\cdot}\eta_m^n} + \mathcal{I}_m(x_F)\, F^n \tag{41a}$$

$$\mathfrak{L}u_m^n = T_0\delta_{xx}u_m^n - EI\delta_{xxxx}u_m^n \tag{41b}$$

$$\eta_m^n = b_m - u_m^n \tag{41c}$$

Numerical boundary conditions, a discrete version of (34), are given as

$$u_m^n = \delta_{xx}u_m^n = 0 \qquad m = 0, M \tag{42}$$

The stability of the scheme can be inferred by energy analysis. Under unforced conditions, taking an inner product of (41a) with $\delta_{t\cdot}u_m^n$ gives the following energy balance

$$\delta_{t+}\left(\underbrace{\mathfrak{H}_k^{n-1/2} + \mathfrak{H}_p^{n-1/2} + \mathfrak{H}_c^{n-1/2}}_{\mathfrak{H}^{n-1/2}}\right) = 0 \tag{43a}$$

$$\mathfrak{H}_k^{n-1/2} = \frac{\rho\|\delta_{t-}u^n\|_{\mathbb{M}}^2}{2} \tag{43b}$$

$$\mathfrak{H}_p^{n-1/2} = \frac{T_0\left\langle \delta_{x+}u^n, \delta_{x+}u^{n-1}\right\rangle_{\mathbb{M}}}{2} + \frac{EI\left\langle \delta_{xx}u^n, \delta_{xx}u^{n-1}\right\rangle_{\overline{\mathbb{M}}}}{2} \tag{43c}$$

$$\mathfrak{H}_c^{n-1/2} = \left\langle 1, \phi^{n-1/2}\right\rangle_{\mathbb{M}} \tag{43d}$$

Because ϕ is non-negative, the total conserved energy will be non-negative under the standard stability condition for the stiff string, i.e. for

$$h^2 \geq \frac{T_0k^2 + \sqrt{T_0^2k^4 + 16EI\rho k^2}}{2\rho} \tag{44}$$

Under such condition, the bounds on the state read

$$0 \leq \|\delta_{t-}u^n\|_{\mathbb{M}} \leq \sqrt{2\mathfrak{H}^{1/2}/\rho} \tag{45}$$

Scheme (41a) can be cast in the following form, resembling the form for the lumped case of section 2.2.1

$$\underbrace{r_m - s_m + \frac{k^2}{\rho}\frac{\phi(-r_m + a_m) - \phi(a_m)}{r_m}}_{G(r_m)} = 0, \; m = 1, ..., M-1$$

where

$$s_m \triangleq 2u_m^n - 2u_m^{n-1} + \frac{k^2}{\rho}\mathfrak{L}u_m^n + \frac{k^2\mathcal{I}_m(x_F)}{\rho}F^n$$
$$r_m \triangleq u_m^{n+1} - u_m^{n-1}, \quad a_m \triangleq b_m - u_m^{n-1}$$

As this is an uncoupled system of nonlinear equations, the scheme can be shown to have a unique solution, as per the lumped system discussed in section 2.2.1. The solution to the system can be found using iterative solvers, such as Newton-Raphson—see [8, 5, 13].

3.1.2. Non-Iterative Conservative Finite Difference Scheme

A novel, non-iterative finite difference scheme arises as a discretisation of (29b). Hence

$$\rho\delta_{tt}u_m^n = \mathfrak{L}u_m^n + \left(\mu_{t+}\psi_m^{n-1/2}\right)\frac{\delta_{t+}\psi_m^{n-1/2}}{\delta_{t\cdot}\eta_m^n} + \mathcal{I}_m(x_F)F^n \tag{46}$$

where $\mathfrak{L}u_m^n$ and η_m^n are as per (41b) and (41c). The stability of the scheme can be inferred by energy analysis. Under unforced conditions, taking an inner product of (46) with $\delta_{t\cdot}u^n$ gives the same energy balance as (43a), where $\mathfrak{H}_k^{n-1/2}$, $\mathfrak{H}_p^{n-1/2}$ are as per (43b), (43c), and where

$$\mathfrak{H}_c^{n-1/2} = \frac{1}{2}\|\psi^{n-1/2}\|_{\mathbb{M}}^2 \tag{47a}$$

Hence, the total energy is non-negative under a choice of the grid spacing h as per (44), in which case the same bounds as (45) hold. As for the lumped case, described earlier in section 2.2.2, an extra equation relating u and ψ is needed. Again, one may conveniently use

$$\frac{\delta_{t+}\psi_m^{n-1/2}}{\delta_{t\cdot}\eta_m^n} = \psi'\Big|_{\eta=\eta_m^n} = \frac{\phi'}{\sqrt{2\phi}}\Big|_{\eta=\eta_m^n} \triangleq g_m^n \tag{48}$$

Making use of the following identity

$$\mu_{t+}\psi_m^{n-1/2} = \frac{k}{2}\delta_{t+}\psi_m^{n-1/2} + \psi_m^{n-1/2} \tag{49}$$

one can cast (46) in the following update form

$$A_m u_m^{n+1} = v_m \tag{50}$$

where

$$A_m = \frac{\rho}{k^2} + \frac{(g_m^n)^2}{4}$$
$$v_m = \frac{2\rho}{k^2}u_m^n - \frac{\rho}{k^2}u_m^{n-1} + \frac{(g_m^n)^2}{4}u_m^{n-1} - g_m^n\psi_m^{n-1/2} + \mathfrak{L}u_m^n$$

The whole system can be solved by a simple vector division. Once u_m^{n+1} is known, one may update $\psi_m^{n+1/2}$, using (48).

3.2. Non-Iterative Conservative Modal Scheme

A modal decomposition is readily available for the string described by (29a). The solution $u(x,t)$ is now expanded onto the eigenmodes for simply-supported boundary conditions, in the following way [17]

$$u(x,t) = \sum_{p=1}^{P} X_p(x) q_p(t) \tag{51}$$

The modal shapes and frequencies are given as

$$X_p(x) = \sin \frac{p\pi x}{L}, \quad \omega_p = \sqrt{\frac{p^2\pi^2}{L^2}\left(\frac{T_0}{\rho} + \frac{EI}{\rho}\frac{p^2\pi^2}{L^2}\right)} \tag{52}$$

Here, for practical purposes, the number of modes has been truncated to P (which is set by stability considerations, as per (61).) Inserting (51) into (29b), and taking an inner product with $X_p(x)$ results in $p = 1, .., P$ projected modal equations, of the form

$$\ddot{q}_p + \omega_p^2 q_p + \frac{X_p(x_F)}{\rho \|X_p\|^2} F(t) + \frac{\langle \psi g, X_p \rangle_{\mathcal{D}}}{\rho \|X_p\|^2} = 0 \tag{53}$$

where

$$g = \frac{\partial_t \psi}{\partial_t \eta}, \quad \eta = b - \sum_{p=1}^{P} X_p(x) q_p(t) \tag{54}$$

It is possible to define a modal energy conservation law, by inserting the modal expansion above into (33b), and by using the fact that $\|X_p\|^2 = \frac{L}{2}\ \forall p$. Hence

$$\frac{d}{dt}\left[\underbrace{\frac{\rho L}{2}\sum_{p=1}^{P}\left(\frac{(\dot{q}_p)^2}{2} + \frac{(\omega_p q_p)^2}{2}\right) + \frac{\|\psi\|_{\mathcal{D}}^2}{2}}_{H(t)}\right] = 0 \tag{55}$$

From the above, it is possible to extract a bound on a single mode p, as

$$0 \leq |\partial_t q_p| \leq \sqrt{4H_0/\rho L} \tag{56}$$

Discretisation of (53) and (54) follows as

$$\delta_{tt} q_p^n + \omega_p^2 q_p^n + \frac{X_p(x_F)}{\rho \|X_p\|^2} F^n + \frac{\left\langle (\mu_{t+}\psi^{n-1/2}) g^n, X_p \right\rangle_{\mathcal{D}}}{\rho \|X_p\|^2} = 0 \tag{57}$$

$$\frac{\delta_{t+}\psi^{n-1/2}}{\delta_{t\cdot}\eta^n} = g^n \triangleq \psi' \Big|_{\eta=\eta^n} \tag{58}$$

Making use of identity (27), one may rewrite the first equation above as

$$\delta_{tt} q_p^n + Q_p + \omega_p^2 q_p^n + \frac{X_p(x_F)}{\rho \|X_p\|^2} F^n + \frac{\left\langle \psi^{n-1/2} g^n, X_p \right\rangle_{\mathcal{D}}}{\rho \|X_p\|^2} = 0 \tag{59}$$

where Q_p is a coupling term, i.e.

$$Q_p = \frac{k}{2\rho \|X_p\|^2} \left\langle (g^n)^2 \delta_{t\cdot}\eta^n, X_p \right\rangle_{\mathcal{D}} \tag{60}$$

The modal coordinates q_p can then be solved using (59), which is in the form of a non-sparse linear system with non-sparse elements given by Q_p. Notice that the resulting non-sparse matrix is in the form of a rank-one perturbation, resolvable very efficiently using the Sherman-Morrison formula [18]. One can then update ψ using (58). The stability of the scheme can once again be understood in terms of its energy-preserving properties. For this scheme, in fact, energy conservation reads

$$\delta_{t+}\left[\underbrace{\frac{\rho L}{2}\sum_{p=1}^{P}\left(\frac{(\delta_{t-}q_p^n)^2}{2} + \frac{\omega_p^2 q_p^n q_p^{n-1}}{2}\right) + \frac{\|\psi^{n-1/2}\|_{\mathcal{D}}^2}{2}}_{\mathfrak{H}^{n-1/2}}\right] = 0$$

and hence, remembering (21), the largest eigenfrequency allowed for this scheme is such that

$$\omega_P < 2 f_s \tag{61}$$

Notice that the contribution of a single mode p to the total energy is non-negative. Hence, boundedness can be stated mode by mode, as

$$0 \leq |\delta_{t-} q_p^n| \leq \sqrt{4\mathfrak{H}^{1/2}/\rho L} \tag{62}$$

3.3. Numerical Examples and Discussion

The three schemes described in the previous sections are now compared in cases of interest in musical acoustics. A first experiment takes into account the case of a point barrier, located at the centre of the domain. Fig. 4 shows a few snapshots of the dynamics of the string, before, during and after contact with the barrier. The three schemes yield consistent solutions, although some differences are also observed.

As for the lumped case of section 2.3, a closer look at the energy components can be revealing. Fig. 5 shows the kinetic, potential and collision energies of the three schemes over time. The non-iterative schemes tend to transform the excess kinetic energy during a collision into extra collision energy, so that bounds (45) are indeed verified. This allows to keep the scheme stable, but it also results in a deterioration of the kinetic and potential energy components. This is particularly evident for the non-iterative finite difference scheme, whereas the modal scheme is somewhat better behaved: this may be a reflection of the modal boundedness property (62). Notice, however, that the total energy of the three schemes is conserved after the forcing vanishes, with the three schemes having the same total energy overall, as expected. Of course, one may increase the sample rate, and observe convergence of the three schemes toward a unique solution, which therefore can be identified as the solution to the original problem. The energy components of Fig. 6 are much more consistent, and the recorded outputs shown in Fig. 7 seem to have converged. Certainly, the solution computed via the classic iterative scheme is characterised by a faster convergence rate. In other words, for the same sample rate, the classic iterative scheme yields a solution closer to the converged solution than the non-iterative schemes. On the other hand, the non-iterative solvers are extremely efficient in this case: the non-iterative finite difference scheme is fully explicit, a rarity in the realm of nonlinear problems. One may exploit this feature in a number of ways, most noticeably using parallel instructions on CPUs and GPUs, alleviating the extra computational burden coming from oversampling. Basic experiments in Matlab show significant speedups: for the experiment of Fig. 8, the new non-iterative finite difference scheme with an oversampling factor

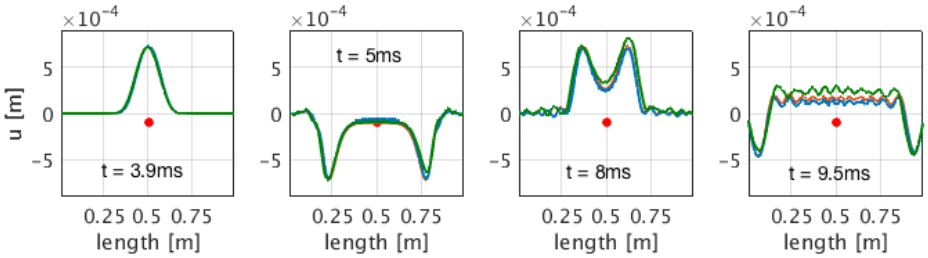


Figure 4: Snapshots of a simulated string colliding against a point barrier. For all panels, the red line is the benchmark (iterative) scheme, the blue line is the non-iterative finite difference method, and the green line is the modal non-iterative method. The point barrier has $K = 5 \cdot 10^6$, $\alpha = 1.4$, $b = 10^{-4}$m and is located at $x = 0.5$m. The string has $L = 1$m, $\rho = 0.063$kg/m, radius $r = 5 \cdot 10^{-4}$m, $T_0 = 500$N, $E = 2 \cdot 10^{11}$Pa. The string is set into motion by a raised cosine input force, with $F_0 = 10$N, and $t_{wid} = 1$ms. The sample rate for this simulation is $f_s = 44100$Hz.

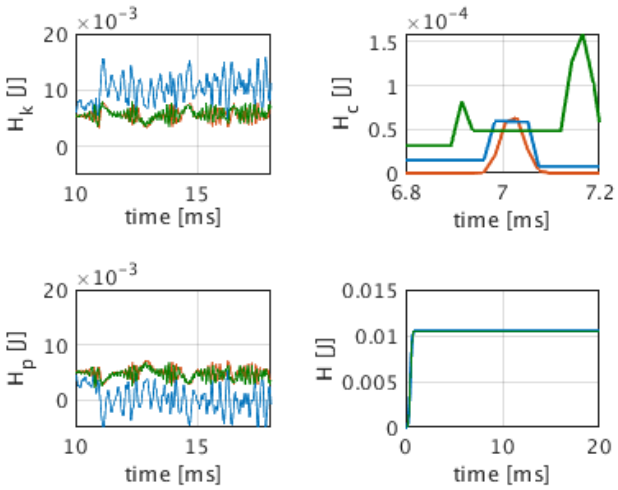


Figure 5: Energy components for the simulations of Fig. 4, where $\mathfrak{H}_k$ stands is the kinetic energy, $\mathfrak{H}_c$ is the collision energy, $\mathfrak{H}_p$ is the potential energy, and $\mathfrak{H}$ is total energy. Colour scheme as Fig. 4

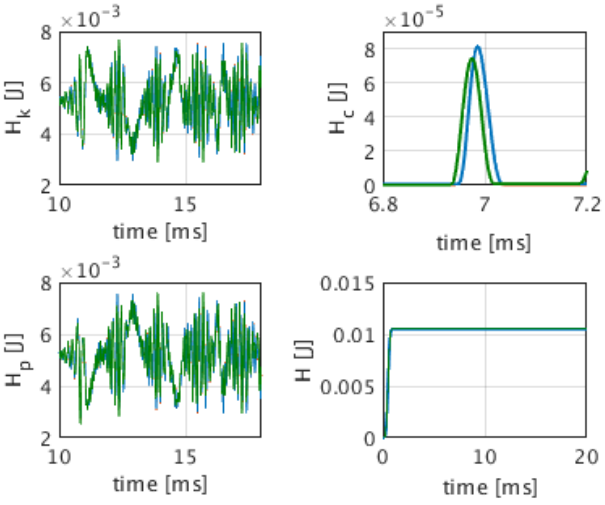


Figure 6: Same as Fig. 5, but with $f_s = 5 \cdot 44100$Hz.

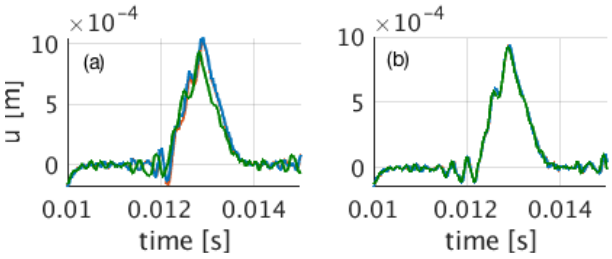


Figure 7: Recorded output at $x_o = 0.9L$. (a): $f_s = 44100$Hz, (b): $f_s = 5 \cdot 44100$Hz. All other parameters as per Fig. 4

of 5 is faster than the iterative finite difference scheme at audio rate (using a fixed number of 20 iterations per collision). A more consistent comparison, in C++, is drawn in the companion paper [14], highlighting time gains of up to an order of magnitude. Assessing the efficiency of the modal non-iterative scheme requires some care. For collisions, modal methods have been successfully implemented in the past. In [12], a non-iterative modal scheme is given, for $\alpha = 1$. In [11], all values of α are allowed, and a spatial grid is used along with an iterative procedure . The novel non-iterative modal schemes presented here can be implemented efficiently, for all values of α, using the Sherman-Morrison formula [18]. Whether or not the non-iterative modal scheme are "faster" than the non-iterative finite difference schemes depends on a number of factors, such as number of modes, sample rate, and number of barrier points. In general, for smaller sizes, the modes can be extremely fast, but in the case of many barrier points, such as the example of Fig. 8, the modal scheme requires the calculation of as many reduced sums, and this has a significant impact of the overall efficiency. The relative efficiency of the schemes proposed will not be discussed further here, but it certainly deserves a closer investigation. As a concluding experiment, the three schemes are used to solve the case of a string colliding against a fairly rigid bent distributed obstacle, as per Fig. 8. This is similar to what happens in tanpuras, and other string instruments, although the simulation parameters are here only meant for illustrative purposes. The three schemes yield the consistent solutions after multiple collisions.

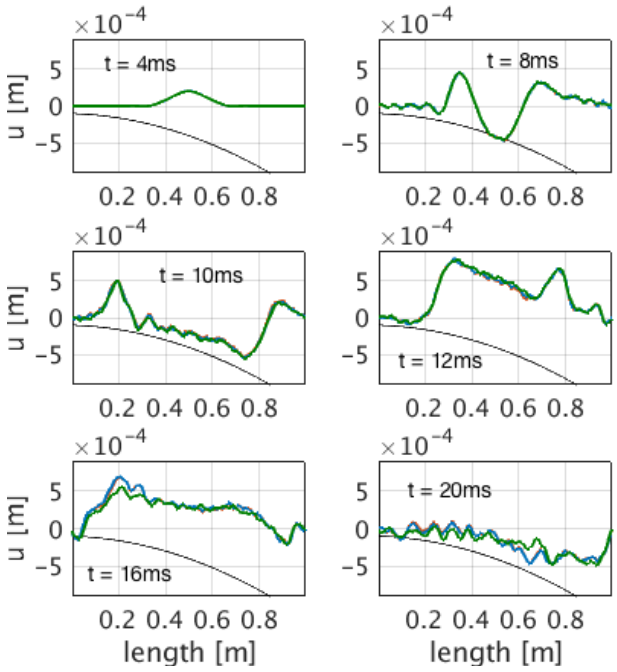


Figure 8: Snapshots of a simulated string colliding against a distributed barrier. For all panels, the red line is the benchmark (iterative) scheme, the blue line is the non-iterative finite difference methods, and the green line is the modal non-iterative method. The barrier has $K = 5 \cdot 10^6$, $\alpha = 1.4$, $b = -10^{-4} - 10^{-4}x - 10^{-3}x^2$. For the modal scheme, the barrier was created by placing point obstacles at locations corresponding to the grid points of the finite difference schemes. The string has $L = 1$m, $\rho = 0.063$kg/m, $r = 5 \cdot 10^{-4}$m, $T_0 = 500$N, $E = 2 \cdot 10^{11}$Pa. The string is set into motion by a raised cosine input force, with $F_0 = 10$N, and $t_{wid} = 1$ms. The sample rate for this simulation is $f_s = 5 \cdot 44100$Hz.

4. CONCLUSIONS

In this work, a novel family of schemes was presented for the solution of collisions in musical acoustics. The case of collisions represents but one of a very large class of nonlinear problems that can be treated within the illustrated framework. The new schemes are non-iterative, and they require at most the solution of a linear system. In particular, for the fully distributed case, a finite-difference non-iterative scheme and a modal non-iterative scheme have been given, with the former being completely explicit. Stability and convergence of the proposed methods have been demonstrated formally, using energy arguments, and via numerical experiments. The new schemes have a slower convergence rate than the benchmark iterative schemes obtained via implicit methods, but they are also much more efficient. Basic experiments in Matlab show that the non-iterative oversampled schemes can be faster than the benchmark iterative schemes run at audio rate. Oversampling was employed here as a corrective measure, but different strategies could and should be investigated in future works.

5. ACKNOWLEDGMENTS

The first author wishes to thank the Leverhulme Trust, who is supporting his research with an Early Career Fellowship.

6. REFERENCES

[1] A. Falaize and T. Hélie, "Passive Guaranteed Simulation of Analog Audio Circuits: A Port-Hamiltonian Approach," *Appl. Sci.*, vol. 6, pp. 273 – 273, 2016.

[2] A. Falaize, *Modélisation, simulation, génération de code et correction de systèmes multi-physiques audios: Approche par réseau de composants et formulation Hamiltonienne À Ports*, Ph.D. thesis, Université Pierre et Marie Curie, Paris, July 2016.

[3] N. Lopes, T. Hélie, and A. Falaize, "Explicit second-order accurate method for the passive guaranteed simulation of port-hamiltonian systems," in *Proc. 5th IFAC 2015*, Lyon, France, July 2015.

[4] X. Yang, "Linear and unconditionally energy stable schemes for the binary fluid-surfactant phase field model," *Comp. Methods Appl. Mech. Eng.*, vol. 318, pp. 1005–1029, 2017.

[5] M. Ducceschi and S. Bilbao, "Modelling collisions of nonlinear strings against rigid barriers: Conservative finite difference schemes with application to sound synthesis," in *Proc. Int. Conf. On Acoust. (ICA 2016)*, Buenos Aires, Argentina, September 2016.

[6] M. Ducceschi, "A numerical scheme for various nonlinear forces, including collisions, which does not require an interative root finder," in *Proc. Int. Conf. On Dig. Audio Eff. (DAFx 2017)*, Edinburgh, UK, September 2017.

[7] V. Chatziioannou, S. Schmutzhard, and S. Bilbao, "On iterative solutions for numerical collision models," in *Proc. Int. Conf. On Dig. Audio Eff. (DAFx 2017)*, Edinburgh, UK, September 2017.

[8] S. Bilbao, A. Torin, and V. Chatziioannou, "Numerical modeling of collisions in musical instruments," *Acta Acust. United Ac.*, vol. 101, pp. 155 – 173, 2015.

[9] V. Chatziioannou and M. van Walstijn, "Energy conserving schemes for the simulation of musical instrument contact dynamics," *J. Sound Vib.*, vol. 339, pp. 262 – 279, 2015.

[10] C. Issanchou, V. Acary, F. Pérignon, C. Touzé, and J-L. Le Carrou, "Nonsmooth contact dynamics for the numerical simulation of collisions in musical string instruments," *J. Acoust. Soc. Am.*, vol. 143, no. 5, pp. 3195, 2018.

[11] C. Issanchou, J.-L. Le Carrou, C. Touzé, B. Fabre, and O. Doaré, "String/frets contacts in the electric bass sound: Simulations and experiments," *Appl. Acoust.*, vol. 129, pp. 217 – 228, 2018.

[12] M. van Walstijn, J. Bridges, and S. Mehes, "A real-time synthesis oriented tanpura model," in *Proc. Int. Conf. On Dig. Audio Eff. (DAFx 2016)*, Brno, Czech Republic, September 2016.

[13] V. Chatziioannou and M. van Walstijn, "An energy conserving finite difference scheme for simulation of collisions," in *Proc. SMAC/SMC 2013*, Stockholm, Sweden, August 2013.

[14] S. Bilbao, M. Ducceschi, and C. Webb, "Large-scale real-time modular physical modeling sound synthesis," in *Proc. Int. Conf. On Dig. Audio Eff. (DAFx 2019)*, Birmingham, UK, September 2019.

[15] K.H. Hunt and F.R.E. Crossley, "Coefficient of restitution interpreted as damping in vibroimpact," *J. Appl. Mech.*, vol. 42, no. 52, pp. 440–445, 1975.

[16] S. Bilbao, *Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics*, Wiley, Chichester, UK, 2009.

[17] M. Ducceschi and S. Bilbao, "Linear stiff string vibrations in musical acoustics: Assessment and comparison of models," *J. Acoust. Soc. Am.*, vol. 140, no. 4, pp. 2445–2454, 2016.

[18] J. Sherman and W. J. Morrison, "Adjustment of an inverse matrix corresponding to a change in one element of a given matrix," *Ann. Math. Stat.*, vol. 21, pp. 124–127, 1950.

GENERALIZATIONS OF VELVET NOISE AND THEIR USE IN 1-BIT MUSIC

Kurt James Werner

The Sonic Arts Research Centre (SARC)
Queen's University Belfast (QUB)
Belfast, United Kingdom of Great Britain and Northern Ireland
kurt.james.werner@gmail.com

ABSTRACT

A family of spectrally-flat noise sequences called "Velvet Noise" have found use in reverb modeling, decorrelation, speech synthesis, and abstract sound synthesis. These noise sequences are ternary—they consist of only the values -1, 0, and $+1$. They are also sparse in time, with pulse density being their main design parameter, and at typical audio sampling rates need only several thousand non-zero samples per second to sound "smooth."

This paper proposes "Crushed Velvet Noise" (CVN) generalizations to the classic family of Velvet Noise sequences including "Original Velvet Noise" (OVN), "Additive Random Noise" (ARN), and "Totally Random Noise" (TRN). In these generalizations, the probability of getting a positive or negative impulse is a free parameter. Manipulating this probability gives Crushed OVN and ARN low-shelf spectra rather than the flat spectra of standard Velvet Noise, while the spectrum of Crushed TRN is still flat. This new family of noise sequences is still ternary and sparse in time. However, pulse density now controls the shelf cutoff frequency, and the distribution of polarities controls the shelf depth.

Crushed Velvet Noise sequences with pulses of only a single polarity are particularly useful in a niche style of music called "1-bit music": music with a binary waveform consisting of only 0s and 1s. We propose Crushed Velvet Noise as a valuable tool in 1-bit music composition, where its sparsity allows for good approximations to operations, such as addition, which are impossible for signals in general in the 1-bit domain.

1. INTRODUCTION

In 2007, Karjalainen and Järveläinen defined a new type of sparse noise sequences which they called "Velvet Noise" [1]. This was later more specifically termed "Original Velvet Noise" (OVN) by Välimäki *et al.* [2]. These noise sequences have some similarities to sparse noise investigated by Schreiber in 1960 [3] and have a few peculiar qualities. First, they are sparse in time—most of their samples are actually zero. Second, the non-zero samples only take the values -1 and $+1$. Specifically, OVN is produced by defining a pulse density, splitting time into equal-length windows, and distributing a single impulse into each window, with both its exact position within the window and its sign ($\pm$) randomized. OVN is clearly not i.i.d. (independent and identically distributed), since the value of each sample within a window is related to the values of all other samples in the window. Despite this, OVN remarkably has a flat magnitude spectrum and an autocorrelation which is nearly zero everywhere except at zero-lag, making it very similar to Gaussian white noise (an i.i.d. process which is neither sparse in time nor limited to particular values). With a sufficiently high pulse density, Velvet Noise can even sound just as "smooth" as Gaussian noise [2].

One fascinating property of Velvet Noise sequences is that they are very efficient to convolve by, since they are very sparse (mostly 0s) and the non-zero values (± 1) don't require an actual multiplication during convolution [4, 5]. These properties have led Velvet Noise to be used in reverb modeling [1, 5, 6, 7, 8, 9], the design of decorrelation filters [10, 11], speech synthesis [12, 13], and abstract sound synthesis [14, 15].

Related to OVN, several other sparse ternary noise sequences have been proposed, including Additive Random Noise (ARN), Totally Random Noise (TRN), Extended Velvet Noise (EVN), Random Integer Noise (RIN) [2]. ARN randomizes the spacing between pulses rather than distributing a single pulse per window. TRN has a random chance of generating a pulse of a random sign for every single sample. EVN takes OVN and restricts the sample location to only a portion of each window, enforcing a second level of sparsity. RIN is identical to ARN, although both the pulse offsets and sign are read from a precomputed table of integer random numbers rather than independent random numbers [2].

In this paper, I propose generalizations to the family of Velvet Noise sequences which are called "Crushed Velvet Noise" (CVN). Specifically, I propose new variants of OVN, ARN, and TRN now called COVN, CARN, and CTRN (the "C" denoting "Crushed" for each). In Crushed Velvet Noise Sequences, the signs of each pulse is not assigned based on a 50% probability, but rather this probability is exposed as a free parameter that can be manipulated along with the pulse density. This small change allows the creation of a variety of spectra with different properties. COVN and CARN both have low-shelf-like Power Spectral Densities (PSDs), with their cuttoff frequency controlled by the pulse density and their shelf attenuation controlled by the free parameter determining the probability of a positive or negative sign for each pulse.

Adjusting this probability to extreme settings gives sequences where either -1s or $+1$s do not appear[1]. Variants of CVN which only have 0s and 1s are of particular use to a niche approach to electronic music composition called "1-bit music," where the only allowable signal levels are 0 and 1. In the end of this paper, I explain how Velvet Noise and the proposed novel variants can be used in 1-bit composition, specifically highlighting their poten-

[1] The case where there is a 100% chance of a $+1$ and no chance of a -1 has already been investigated briefly in [13], where a unipolar variant of OVN is called Unipolar Velvet Noise (UVN). The sparse noise sequence explored by Schreiber [3] was also unipolar. With these in mind, we can say that this paper fills in the gaps between the proposed unipolar variant and the bipolar OVN sequence.

tial for spectral shaping, volume control, and layering of signals. These procedures are not possible *in general* in the 1-bit domain, since even the simplest operations like addition of signals do not exist in that domain, and so it is very hard to do anything LTI (linear and time-invariant). So, being able to control the spectrum and volume of a *particular* type of 1-bit signal, Crushed Velvet Noise, is extremely useful.

The prefix "crushed" refers both to bitcrushing (the classic lo-fi audio effect) and crushed velvet (the soft fabric). Bitcrushing involves reducing the number of possible signal levels of an audio signals, for instance down to $2^8 = 256$ levels for an 8-bit bitcrusher. Velvet Noise's ternary character is related to bitcrushing, and the proposed "crushed" variants, at their extreme settings, further reduce the set of possible sample values down to two. Crushed velvet is a particular kind of velvet fabric whose cut threads have been pressed in different directions in specific ways. Once the reader has understood the construction of Crushed Velvet Noise, a loose metaphorical connection is not hard to imagine.

In the rest of the paper I review the classic definitions of OVN, ARN, and TRN (§2), define and study the novel COVN, CARN, and CTRN sequences (§3), and explain an application to "1-bit music" (§4). §5 concludes and proposes avenues for future work.

2. CLASSIC VELVET NOISE DEFINITIONS

In this section, we will briefly review the classic Velvet Noise sequences: Original Velvet Noise (OVN), Additive Random Noise (ARN), and Totally Random Noise (TRN).

2.1. Original Velvet Noise (OVN)

Original Velvet Noise (OVN), proposed in [1] and termed OVN in [2], is defined by

$$s_{\text{ovn}}(n) = \begin{cases} 2\,\|r_2(m)\| - 1\,, & \text{if } n = k_{\text{ovn}}(m) \\ 0\,, & \text{otherwise} \end{cases}, \qquad (1)$$

where $n = 0, 1, 2, \ldots$ is the discrete-time sample index, $\|\cdot\|$ is a function that rounds to the nearest integer, $r_2(m)$ is a sequence of random numbers uniformly distributed between 0 and $+1$, and k_{ovn} is a sequence of impulse locations defined by

$$k_{\text{ovn}}(m) = \|mT_{\text{d}} + r_1(m)(T_{\text{d}} - 1)\|\,, \qquad (2)$$

where $m = 0, 1, 2, \ldots$ is a discrete pulse index, T_{d} is window width in samples, and $r_1(m)$ is another sequence of random numbers uniformly distributed between 0 and $+1$. T_{d} is related to the pulse density and sampling rate f_{s} by

$$N_{\text{d}} = f_{\text{s}}/T_{\text{d}}\,. \qquad (3)$$

The sampling rate used throughout this paper is $f_{\text{s}} = 96\,\text{kHz}$.

The definition (1)–(2) of OVN is used widely [2, 9, 5, 10]. However, another variant exists [1, 7, 11] and actually precedes (1)–(2) [1]. Its definition of $k_{\text{ovn}}(m)$ is slightly different:

$$k_{\text{ovn,alt}}(m) = \|mT_{\text{d}} + r_1(m)T_{\text{d}}\| = \|(r_1(m) + m)T_{\text{d}}\|\,. \qquad (4)$$

The second definition (4) has the potential to rarely have an impulse in the last sample of one window collide with an impulse in the first sample of the next window, if the pulse should occur on a window boundary.

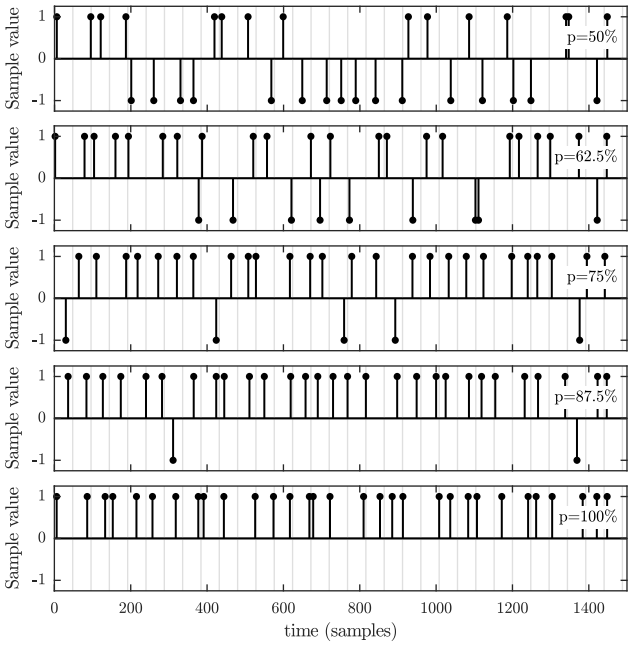


Figure 1: Crushed Original Velvet Noise (COVN), with various probability percentages $p \in \{50\%, 62.5\%, 75\%, 87.5\%, 100\%\}$.

Finally, we can mention that [11] reformulates (4) using a ceiling function $\lceil\cdot\rceil$ rather than $\|\cdot\|$.

An example of an OVN sequence ($N_{\text{d}} = 2000$) is shown in Fig. 1 (top).

2.2. Additive Random Noise (ARN)

Additive Random Noise (ARN) is defined by [2]

$$s_{\text{arn}}(n) = \begin{cases} 2\,\|r_2(m)\| - 1\,, & \text{if } n = \|k_{\text{arn}}(m)\| \\ 0\,, & \text{otherwise} \end{cases}, \qquad (5)$$

where k_{arn} is a sequence of impulse locations defined by

$$\begin{aligned} k_{\text{arn}}(m) = k_{\text{arn}}(m-1) + 1 \ldots \\ + (1-\Delta)(T_{\text{d}} - 1) + 2\Delta(T_{\text{d}} - 1)r_1(m)\,. \end{aligned} \qquad (6)$$

The parameter $\Delta \in [0, 1]$ controls a tradeoff between advancing time by a fixed amount and a random amount.

An example of an ARN sequence ($N_{\text{d}} = 2000$) is shown in Fig. 6 (top).

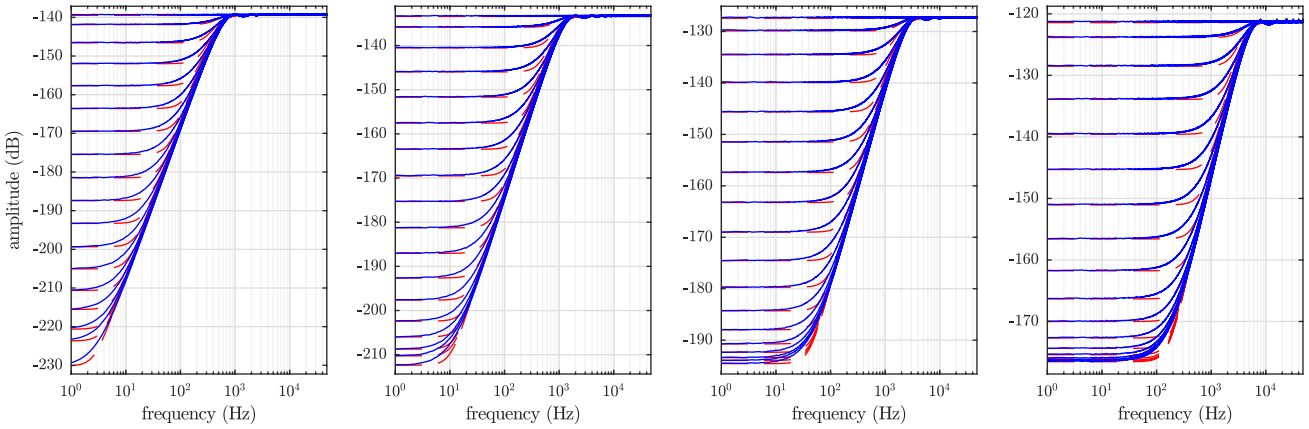
2.3. Totally Random Noise (TRN)

Totally Random Noise (TRN) is defined by [2]

$$s_{\text{trn}}(n) = \left\| \left(\frac{T_{\text{d}}}{T_{\text{d}} - 1} \right) \left(r_1(n) - \frac{1}{2} \right) \right\|\,. \qquad (7)$$

TRN was originally investigated by Rubak and Johansen [16, 17].

An example of a TRN sequence ($N_{\text{d}} = 2000$) is shown in Fig. 9 (top).



(a) COVN, $N_\mathrm{d} = 1000$. (b) COVN, $N_\mathrm{d} = 2000$. (c) COVN, $N_\mathrm{d} = 4000$. (d) COVN, $N_\mathrm{d} = 8000$.

Figure 2: Power Spectral Density (PSD) of COVN at various pulse densities $N_\mathrm{d} \in \{1000, 2000, 4000, 8000\}$.

3. CRUSHED VELVET NOISE (CVN)

In this section, novel "Crushed" variants of OVN, ARN, and TRN are introduced.

3.1. Crushed Original Velvet Noise (COVN)

Crushed Original Velvet Noise (COVN) is defined by

$$s_\mathrm{covn}(n) = \begin{cases} 2 \cdot c(r_2(m), p) - 1\,, & \text{if } n = k_\mathrm{covn}(m) \\ 0 & \text{otherwise} \end{cases}, \qquad (8)$$

which is identical to the traditional OVN sequence except the standard rounding function $\|\cdot\|$ has been replaced by the function $c(x, p)$, defined as

$$c(x, p) = \begin{cases} 1\,, & \text{if } x > p \\ 0\,, & \text{otherwise} \end{cases}. \qquad (9)$$

The pulse timings $k_\mathrm{ovn}(m)$ are given by

$$k_\mathrm{covn}(m) = \|(r_1(m) + m)\, T_\mathrm{d}\|\,, \qquad (10)$$

which is identical to the alternate definition from OVN (4). The reason for basing COVN off of the alternate definition is that the timing equation (2) introduces periodicities into COVN sequences (when $p \neq 0.5$), negatively affecting their noisy character.

Five examples of COVN with pulse density $N_\mathrm{d} = 2000$ and different polarity probabilities $p \in \{0.5, 0.625, 0.75, 0.875, 1.0\}$ are shown in Fig. 1. Notice that $p = 0.5$ is identical to traditional OVN, and that $p = 1.0$ is fully unipolar—it has only 0s and $+1$s, and no -1s. In this figure, the window boundaries are shown with vertical gray lines.

Power Spectral Density (PSD) estimates of COVN sequences with various pulse densities $N_\mathrm{d} \in \{1000, 2000, 4000, 8000\}$ are shown in Fig. 2. In each case, a family of many polarity probabilities p between 0.5 and 1.0 are shown, where the solid blue lines show the PSDs, with $p = 0.5$ on the top, and increasing to $p = 1.0$ below[2]. The polarity probabilties which are plotted follow

[2]In this paper, we will always deal with probabilities between 0.5 and 1.0, biasing the distribution towards $+1$s. Of course, all of the conisderations of the paper would be identical (except for an opposite dc bias) if we instead considered probabilities between 0.5 and 0.0.

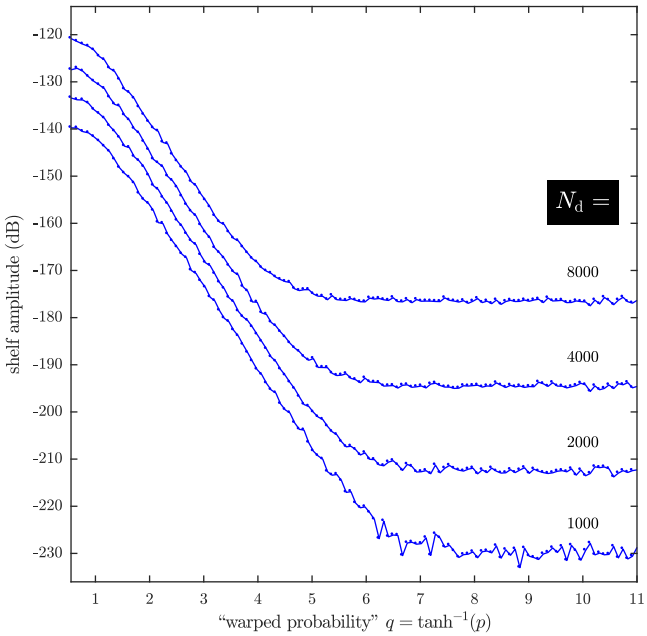


Figure 3: Low shelf amplitude of COVN at various pulse densities $N_\mathrm{d} \in \{1000, 2000, 4000, 8000\}$.

the following pattern:

$$p = \begin{cases} 1/2 & \tau = 0 \\ 1 - (1/2)^\tau & \tau = 1, 2, \ldots 16 \\ 1 & \tau = 17 \end{cases}. \qquad (11)$$

PSD estimates are produced using Welch's method with a window size of $N_\mathrm{FFT} = 2^{18} = 262\,144$ samples, an overlap size of $N_\mathrm{FFT}/2$, and Hamming windows, and are plotted on a $20\log_{10}(\cdot)$ scale rather than a $10\log_{10}(\cdot)$ scale to facilitate comparison with shelf filter magnitude responses. Noise sequences used to generate PSD estimates in this paper are 24 hours long, i.e., $24 \times 60 \times$

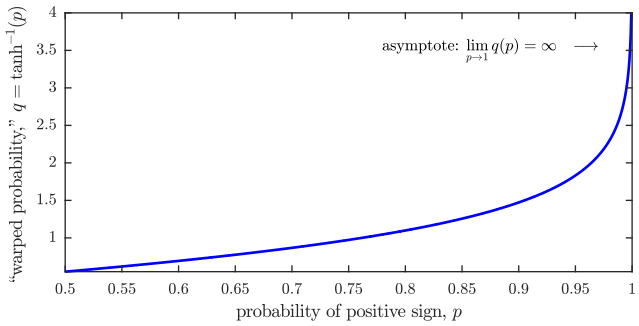


Figure 4: Relationship between probability of positive sign p and "warped probability" q.

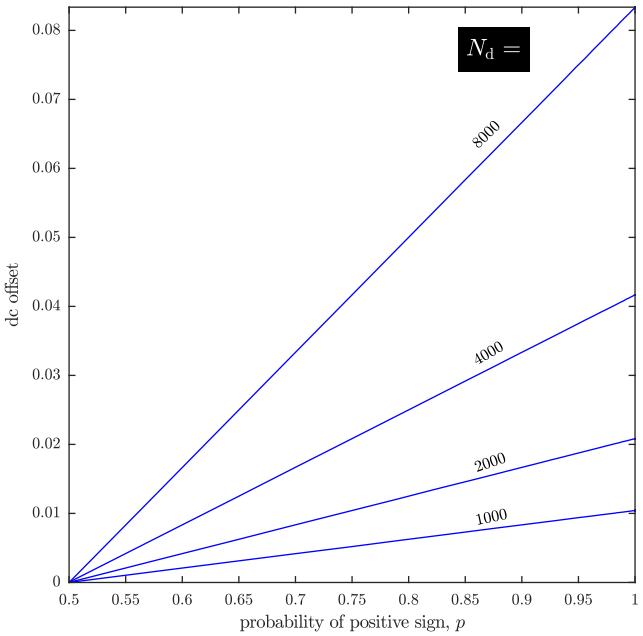


Figure 5: dc offset of COVN at various pulse densities $N_d \in \{1000, 2000, 4000, 8000\}$.

$60 \times f_s = 8.2944 \times 10^9$ samples. The purpose of this length, which may seem excessive, is simply to get better PSD estimates by having more frames to average using Welch's method, i.e., a better estimate of the shape of the PSD curves without using any artificial smoothing.

It is clear from Fig. 2 that, unlike OVN with its flat spectrum, COVN sequences have a *low-shelf* characteristic, where the low shelf has some attenuation but never a boost. These PSDs have a transition band slope of $+12$ dB/octave ($+40$ dB/decade), implying that they are similar in some way to white noise through a second-order shelf filter. There are various second-order shelf-filters defined in the literature [18, 19, 20]. The family of noise spectra produced by the COVN noise are similar to the second-order low-shelf filters proposed by Holters and Zölzer in [19]. They propose low-shelf filters with transfer function $H_{LS}(s)$ defined in pole-zero form on the s-plane by:

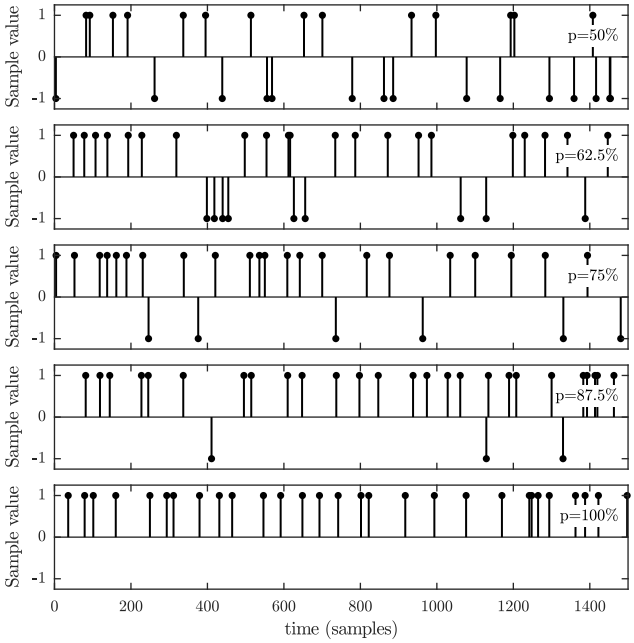


Figure 6: Crushed Additive Random Noise (CARN), with various probability percentages $p \in \{50\%, 62.5\%, 75\%, 87.5\%, 100\%\}$.

$$H_{LS,L}(s) = \prod_{l=1}^{L} \frac{s + \sqrt[L]{g}\omega_c e^{j\left(\frac{1}{2} - \frac{2l-1}{2L}\right)}}{s + \omega_c e^{j\left(\frac{1}{2} - \frac{2l-1}{2L}\right)}}, \tag{12}$$

where $L \geq 1$ is an integer defining the order of the shelf filter, g is the shelf level, and ω_c is the cutoff frequency in radians ($f_c = \omega_c/2\pi$ is the cutoff frequency in Hz). Here we are specifically interested in the 2nd-order case ($L = 2$):

$$H_{LS,2}(s) = \frac{\left(s + \sqrt{g}\omega_c e^{j/4}\right)\left(s + \sqrt{g}\omega_c e^{-j/4}\right)}{\left(s + \omega_c e^{j/4}\right)\left(s + \omega_c e^{-j/4}\right)}. \tag{13}$$

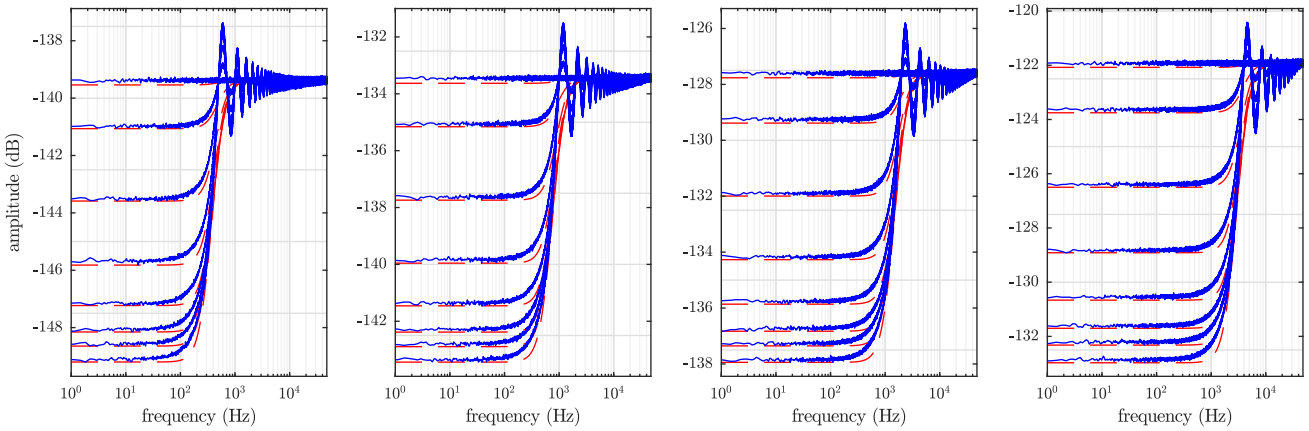
Relating to this family of shelf filter, the cutoff frequency of our PSDs is roughly $f_c = N_d/2$. Shelf filter responses that approximate the COVN sequences are shown on Fig. 2 by dashed red lines.

The shelf attenuation, obviously zero for $p = 0.5$, increases as p approaches 1.0. For the various pulse densities tested, the specific relationship between pulse density and shelf level, g in (13), is shown in Fig. 3. Plotting against p would overly compress the visual display of these traces near $p = 1$, so the horizontal axis is instead a "warped probability scale" q, defined simply as

$$q = \tanh^{-1}(p). \tag{14}$$

The relationship between p and q is shown graphically in Fig. 4.

Finally, it is worth mentioning that disturbing the relative probabilities of -1s and $+1$s using p introduces a small dc offset of $(2p-1)N_d/f_s$ to the signal. Notice that when $p = 0.5$, no dc offset is introduced, and that the maximum offset that can be introduced is $\pm N_d/f_s$. This is shown graphically in Fig. 5.



(a) CARN, $N_\mathrm{d} = 1000$. (b) CARN, $N_\mathrm{d} = 2000$. (c) CARN, $N_\mathrm{d} = 4000$. (d) CARN, $N_\mathrm{d} = 8000$.

Figure 7: Power Spectral Density (PSD) of CARN at various pulse densities $N_\mathrm{d} \in \{1000, 2000, 4000, 8000\}$.

3.2. Crushed Additive Random Noise (CARN)

Crushed Additive Random Noise (CARN) is defined by

$$s_{\mathrm{carn}}(n) = \begin{cases} 2 \cdot c(r_2(m), p) - 1\,, & \text{if } n = \|k_{\mathrm{carn}}(m)\| \\ 0\,, & \text{otherwise} \end{cases}, \quad (15)$$

where k_{carn} is a sequence of impulse locations defined by

$$k_{\mathrm{carn}}(m) = k_{\mathrm{carn}}(m-1) + 1 \ldots + (1-\Delta)(T_\mathrm{d} - 1) + 2\Delta(T_\mathrm{d} - 1) r_1(m)\,, \quad (16)$$

which is identical to the traditional ARN (5)–(6), except that the rounding function $\|\cdot\|$ has again been replaced by the new function (9). In this paper, we will only consider the case $\Delta = 1$, i.e., the case where time advances by purely random steps.

Five examples of CARN with pulse density $N_\mathrm{d} = 2000$ and different polarity probabilities $p \in \{0.5, 0.625, 0.75, 0.875, 1.0\}$ are shown in Fig. 6. Notice that $p = 0.5$ is identical to traditional ARN, and that $p = 1.0$ is again fully unipolar.

PSD estimates of CARN sequences with pulse densities $N_\mathrm{d} \in \{1000, 2000, 4000, 8000\}$ are shown in Fig. 7. Again, a family of many polarity probabilities p between 0.5 and 1.0 are shown, where the solid blue lines show the PSDs, with $p = 0.5$ on the top, and increasing to $p = 1.0$ below. The polarity probabilities which are plotted follow the following pattern:

$$p = \begin{cases} 1/2 & \tau = 0 \\ 1 - (1/2)^\tau & \tau = 1, 2, \ldots 6 \\ 1 & \tau = 7 \end{cases}. \quad (17)$$

As with COVN, shelf filter responses that approximate the CARN sequences are shown on Fig. 7 by dashed red lines.

As with COVN, the PSDs of CARN sequences look a bit like the PSDs of low-shelf-filtered white noise. CARN sequences, however, have a more pronounced ripple around the cutoff frequency, and much less pronounced shelf attenuation than COVN. Again, the sheld attenuation, obviously zero for $p = 0.5$, increases as p approaches 1.0. For CARN, the relationship between p and shelf level is shown in Fig. 8. The dc offset considerations for COVN, which were based purely on pulse density N_d, apply identically to CARN.

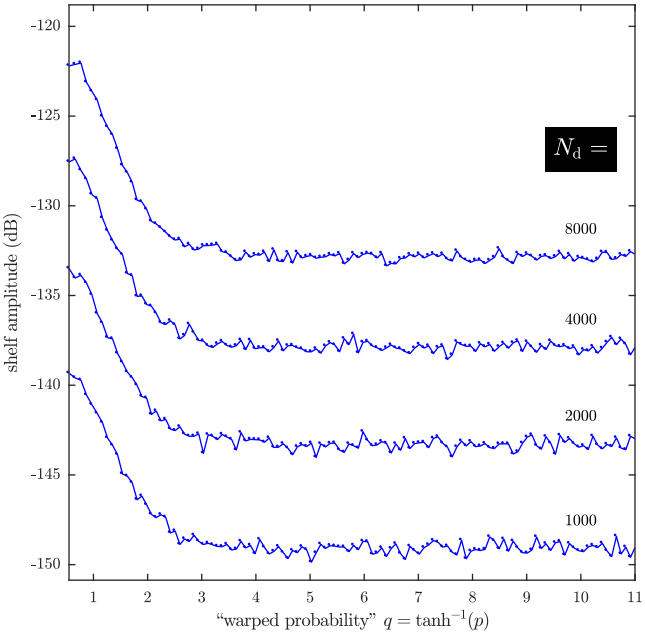


Figure 8: Low shelf amplitude of CARN at various pulse densities $N_\mathrm{d} \in \{1000, 2000, 4000, 8000\}$.

3.3. Crushed Totally Random Noise (CTRN)

Crushed Totally Random Noise (CTRN) is defined by

$$s_{\mathrm{trn}}(n) = c(r_2(n), p) \cdot \left| \left\| \frac{T_\mathrm{d}}{T_\mathrm{d} - 1} \left(r_1(n) - \frac{1}{2} \right) \right\| \right|, \quad (18)$$

where $|\cdot|$ represents the absolute value function, which is identical to traditional TRN (7), except that the polarity is defined by the new function (9) and another random noise sequence $r_2(n)$ rather than the rounding function $\|\cdot\|$.

Five examples of CTRN with pulse density $N_\mathrm{d} = 2000$ and different polarity probabilities $p \in \{0.5, 0.625, 0.75, 0.875, 1.0\}$

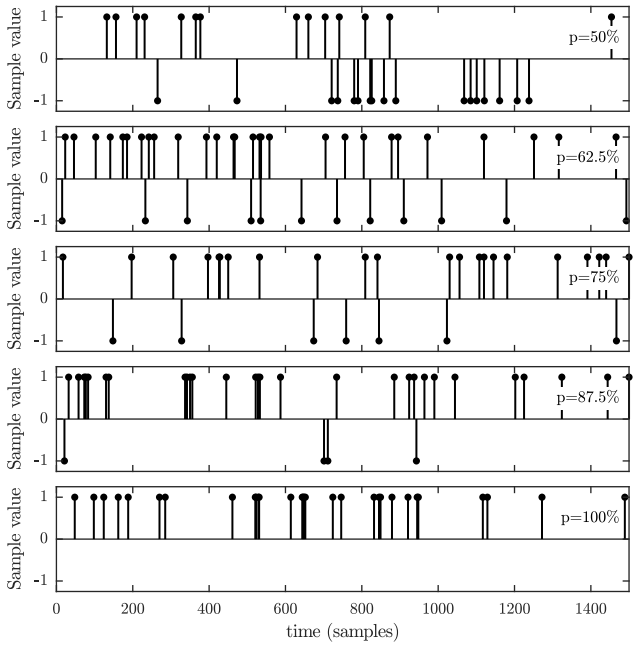


Figure 9: Crushed Totally Random Noise (CTRN), with various probability percentages $p \in \{50\%, 62.5\%, 75\%, 87.5\%, 100\%\}$.

are shown in Fig. 9. Notice that $p = 0.5$ is identical to traditional TRN, and that $p = 1.0$ is fully unipolar—it has only 0s and $+1$s, and no -1s.

PSD estimates of CTRN sequences with pulse densities $N_\mathrm{d} \in \{1000, 2000, 4000, 8000\}$ are shown in Fig. 10. The polarity probabilities which are plotted are the same as for COVN. Unlike COVN and CARN, the PSDs for CTRN seem not to depend on p hardly at all—rather then are all entirely flat, just like traditional TRN.

The dc offset considerations for COVN and CARN apply identically to CTRN.

4. APPLICATIONS TO 1-BIT MUSIC

1-bit music is a style of electronic music production where only waveforms composed of 0s and 1s are allowed. To put it mathematically, an M-second-long 1-bit music composition with a sampling rate of f_s is defined by

$$x(n) \in \{0, 1\}\,, \quad n = 0, 1, 2, \ldots, M \cdot f_\mathrm{s}\,. \tag{19}$$

1-bit music (also called PC "beeper music") has its origins in retro computing platforms like the Apple //e and the Sinclair ZX Spectrum, whose sound systems consisted of a single *digital* CPU pin wired straight to a small speaker or output jack [21, 22, 23]. Conceptually, 1-bit music has some relationship to the idea of composition using "sieves," as developed by Iannis Xenakis [24, 25], digital audio effects and synthesizers based on manipulating binary data [26, 27], and especially to Σ-Δ modulator encoding [28, 29, 30, 31, 32, 33].

Today 1-bit music is still widely created, for instance by musicians such as Richard Hollins (Tufty) [34], utz with his "irrlicht project" [35], Mister Beep [36], and Blake Troise (Protodome). Composer Tristan Perich did a lot for popularizing 1-bit music with his albums "1-bit music" (2004) [37] and the very positively

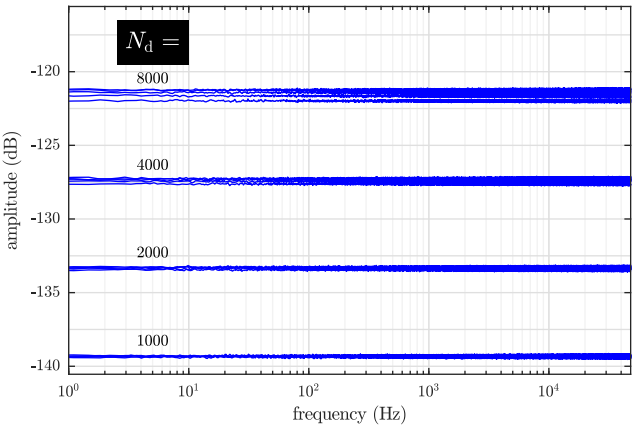


Figure 10: Power Spectral Densities (PSD) of CTRN at various pulse densities $N_\mathrm{d} \in \{1000, 2000, 4000, 8000\}$.

reviewed "1-bit Symphony" [38], both of which create stereo 1-bit music from microcontrollers mounted inside of clear compact disc jewel cases. 1-bit music has even made a recent appearance in popular culture, in the ZX Spectrum software NOHZDYVE featured in the film "Bandersnatch" [39].

At first, it may seem that 1-bit music should have a very limited palette, perhaps consisting solely of square waves, pulse waves, impulse trains, and 1-bit noise. But in fact there are a wide variety of timbres that can be made in the 1-bit domain. Composers have managed to come up with a wide variety of techniques and some synthesisers use oscillators that essentially produce 1-bit signals. For example we can mention the Television Interface Adaptor (TIA) chip from Ataric VCS [21], which produces sounds by logical operations on binary signals, and the Korg MS-20 and ARP Odyssey which use "ring modulators" which are actually XOR chips operating on two square waves. The same technique of "ring-modulating" pulse waves is also used in "metallic noise" generators found in some analog drum machines [40, 41], for instance the Korg KR-55. Some old video game systems produce 1-bit noise, for example the Nintendo NES (Nintendo Entertainment System) and Commodore 64, which produce 1-bit noise using Linear Feedback Shift Register (LFSR) circuits [21].

Nonetheless, composing in the 1-bit domain is still incredibly difficult. Signal addition does not in general exist, which means linear operations like mixing and attenuating signals, filtering, etc. are difficult or impossible to achieve in general and must be accomplished through clever composition, construction of suitable signals to fake them, or other clever means. Here I'll explain how sparse noise sequences, including the unipolar ($p = 1.0$) version of the proposed family of Crushed Velvet Noise sequences, can be particularly useful in the 1-bit music context, and can actually solve a few of these issues.

4.1. (Noise) Problems in 1-bit Music

One of the biggest challenges in 1-bit music is combining multiple signals. Traditionally, this was handled for tonal sounds by using logical operations (e.g. XOR) to combine multiple streams of pulse waves with very narrow pulses (down to a single impulse, i.e. to an impulse train) [42]. Since the pulses would only rarely line up, XOR acts essentially like addition, and you can get the sound of

signals added together without technically being able to perform addition on binary signals in general.

This technique could not be used so well with 1-bit noise, which is not sparse. So, for noise, this problem would be usually handled on the compositional level, leveraging masking phenomenon in human hearing. For instance, it is possible to overwrite very short 1-bit noise bursts onto another 1-bit musical signal without producing a perceived interruption, giving the illusion of two simultaneous musical lines[3]. If the interruption is short enough, the illusion can be very effective. However, this technique is obviously limited to short noise bursts and is no help at all for layering longer noisy sounds.

Another issue with noise in 1-bit music is that its spectrum is not easy to control. Setting each sample to 0 or 1 randomly produces a flat spectrum. This has its uses, but is timbrally limiting. Since LTI signal processing essentially does not exist in the 1-bit domain, it is difficult to filter a white signal like this further, as you would in traditional sound synthesis. However, it is possible to produce a somewhat passable imitation of lowpass-filtered noise by downsampling a 1-bit noise signal. None of these signals will be sparse in time, making them hard to layer with other signals. Furthermore, it's not obvious how to produce an approximation of noise with a highpass quality.

Finally, the lack of any waveform levels beyond 0 and 1 means that there is not usually a good way to control the perceived amplitude of a 1-bit signal. It turns out that this is an issue with many different 1-bit signals, not only noise. For pulse waves, it can be possible to trick your ear into hearing pulse width modulation towards a pulse train as a level shift, but this also introduces a timbre shift and does not generalize well.

4.2. Solution Using Crushed Velvet Noise

These limitations of noise in 1-bit music—limitations on layering with noise, noise spectra, and noise amplitude—can all be addressed somewhat using various Crushed Velvet Noise sequences.

In general, the sparsity of CVN makes it relatively easy to layer with other 1-bit signals using logical operations such as XOR ($\oplus$). This works by the same mechanism as for impulse trains—since the impulses only happen very rarely, they aren't likely to line up with ones from another stream. The $1 \oplus 1 = 0$ case of XOR rarely occurs, so it largely sounds like addition. This is true for all three proposed varieties of CVN: COVN, CARN, and CTRN.

Unlike normal binary noise, two varieties of CVN—COVN and CARN—have controllable spectra. By changing the pulse density N_d, the cutoff frequency of the two types of shelf-filtered noise can be produced. Here the cutoff is linked inextricably to the density of the pulses, hence the signal energy, hence the perceived volume of the signal, which means that the volume of the noise will also increase with increasing pulse density.

That property could be used on its own as a simulacra of volume control. Or, if it was desired to adjust the CVN level independent of the noise spectrum, CTRN signals could be used. CTRN sequences remain largely white regardless of the pulse density. So, they can be used to control the perceived volume of the signal independent of its spectrum, so long as you are happy with the noise spectrum being white.

One limitation of these techniques is that Crushed Velvet Noise sequences cease to sound "smooth" when their densities are too low, presumably similar to the pulse density that is required for standard Velvet Noise. In, e.g., reverb modeling or decorrelation filter design, this would normally be seen as a limitation of Velvet Noise. However, in the context of 1-bit noise where everything sounds a bit raw and grainy, it could perhaps be considered a charming and characteristic quirk. One interesting effect is "sweeping" the pulse density of a COVN sequence downwards. Starting from a high pulse density, this will sound like a shelf filter sweep. However, as the pulse density gets low enough, the "smoothness" dissapears and it sounds like the signal "disintegrating."

An example of using COVN in the context of 1-bit composition can be heard in the author's cover [44] of "Unholy Captives" from the video game "Return of the Obra Din" by Lucas Pope [45].

[3] A phenomenon that is well-known in psychoacoustics, e.g. [43].

5. CONCLUSION

In this paper, new generalizations of Velvet Noise sequences were proposed. These "Crushed" Velvet Noise sequences, which open up the probability of an impulse taking a positive or negative polarity as a free parameter, can be used to make colored noise spectra, as opposed to the white spectra of traditional Velvet Noise. Crushed Velvet Noise (specifically the unipolar case) is a suitable sequence for use in *1-bit music*, alongside classic binary noise sequences like Linear Feedback Shift Register (LFSR) noise. Especially Crushed Original Velvet Noise and Crushed Totally Random Noise should be considered very useful in 1-bit music composition, since their sparsity characteristics make them very easy to mix with other 1-bit signals using logical operations like XOR.

Crushed Velvet Noise leaves open a lot of scope for future work. We've seen in this paper that certain CVN sequences (COVN and CARN) have PSDs that resemble low-shelf-filtered white noise, while others (CTRN) have flat spectra just like their traditional versions. Perhaps future mathematical analysis can reveal how Velvet Noise manages to have a white spectrum without being i.i.d. Future work could also consider trying to create other standard colored noise spectra—e.g. lowpass, bandpass, notch, and high-shelf—from sparse ternary noise. In this paper, only the $\Delta = 1.0$ case of CARN was considered. Future work could consider the more general case of CARN to see if it has any interesting properties, as well as a "Crushed" variant of the related "Extended Velvet Noise" (EVN) [2]. Although the PSD estimates shown in this paper can be taken as very good estimates of the true, underlying PSD of the studied noise sequences, on account of their very long length (24 hours!), we don't need sequences nearly that long to hear the spectral character of different noise sequences. Your ear can guide you; you should find that the spectral character of each noise sequence reveals itself even with very short bursts of Velvet Noise. However, it would be interesting for future work to test our perception of short Velvet Noise bursts.

Finally, it seems clear that 1-bit synthesis, mixing, and audio effects have a close relationship to signal processing of Σ-Δ bit streams [28, 29, 30, 31, 32, 33]. Hopefully future work can gain new insights from that literature.

6. ACKNOWLEDGMENTS

Thank you to Elliot Kermit Canfield-Dafilou and Ólafur Bogason for helpful conversations and to the reviewers for their careful editing and suggestions. Thank you to Fabián Esqueda for bringing the Korg KR-55's metallic noise circuit to my attention.

7. REFERENCES

[1] M. Karjalainen and H. Järveläinen, "Reverberation modeling using velvet noise," in *Proc. 30th Int. Conf. Audio Eng. Soc. (AES)*, Saariselkä, Finland, Mar. 15–17 2007.

[2] V. Välimäki, H.-M. Lehtonen, and M. Takanen, "A perceptual study on velvet noise and its variants at different pulse densities," *IEEE Trans. Audio, Speech, Language Process.*, vol. 21, no. 7, pp. 1481–1488, July 2013.

[3] L. Schreiber, "Was empfinden wir als gleichförmiges Rauschen?," *Frequenz*, vol. 14, no. 12, pp. 399–403, Dec. 1960.

[4] J. Borish, "An efficient algorithm for generating colored noise using a pseudorandom sequence," *J. Audio Eng. Soc. (JAES)*, vol. 33, no. 3, pp. 141–144, Mar. 1985.

[5] B. Holm-Rasmussen, H.-M. Lehtonen, and V. Välimäki, "A new reverberator based on variable sparsity convolution," in *Proc. 16th Int. Conf. Digital Audio Effects (DAFx-13)*, Maynooth, Ireland, Sept. 2–5 2013.

[6] K. S. Lee, J. S. Abel, V. Välimäki, T. Stilson, and D. P. Berners, "The switched convolution reverberator," *J. Audio Eng. Soc. (JAES)*, vol. 60, no. 4, pp. 227–236, Apr. 2012.

[7] V. Välimäki, J. D. Parker, L. Savioja, J. O. Smith, and J. S. Abel, "Fifty years of artificial reverberation," *IEEE Trans. Audio, Speech, Language Process.*, vol. 20, no. 5, pp. 1421–1448, July 2012.

[8] V. Välimäki, J. D. Parker, J. O. Smith, and J. S. Abel, "More than fifty years of artificial reverberation," in *Proc. 60th Int. Conf. Audio Eng. Soc. (AES)*, Leuven, Belgium, Feb. 3–5 2016.

[9] V. Välimaki, B. Holm-Rasmussen, B. Alary, and H.-M. Lehtonen, "Late reverberation synthesis using filtered velvet noise," *Appl. Sci.*, vol. 7, no. 5, 2017, Article #483.

[10] B. Alary, A. Politis, and V. Välimäki, "Velvet-noise decorrelator," in *Proc. 20th Int. Conf. Digital Audio Effects (DAFx-17)*, Edinburgh, UK, Sept. 5–9 2017, pp. 405–411.

[11] S. J. Schlecht, B. Alary, V. Välimäki, and E. A. P. Habets, "Optimized velvet-noise decorrelator," in *Proc. 21st Int. Conf. Digital Audio Effects (DAFx-18)*, Aveiro, Portugal, Sept. 4–8 2018.

[12] H. Kawahara, K.-I. Sakakibara, M. Morise, H. Banno, T. Toda, and T. Irino, "Frequency domain variants of velvet noise and their application to speech processing and synthesis," in *Proc. Interspeech*, Hyderbad, India, Sept. 2–6 2018, pp. 2027–2031.

[13] H. Kawahara, "Application of velvet noise and its variants for synthetic speech and singing," Tech. Rep., IPSJ SIG Tech. Rep., 2018.

[14] V. Välimäki, J. Rämö, and F. Esqueda, "Creating endless sounds," in *Proc. 21st Int. Conf. Digital Audio Effects (DAFx-18)*, Aveiro, Portugal, Sept. 4–8 2018, pp. 32–39.

[15] S. D'Angelo and L. Gabrielli, "Efficient signal extrapolation by granulation and convolution with velvet noise," in *Proc. 21st Int. Conf. Digital Audio Effects (DAFx-18)*, Aveiro, Portugal, Sept. 4–8 2018, pp. 107–112.

[16] P. Rubak and L. G. Johansen, "Artificial reverberation based on a pseudo-random impulse response: part I," in *Proc. 104th Conv. Audio Eng. Soc. (AES)*, Amsterdam, the Netherlands, May 16–19 1998, Conv. paper #4725.

[17] P. Rubak and L. G. Johansen, "Artificial reverberation based on a pseudo-random impulse response ii," in *Proc. 106th Conv. Audio Eng. Soc. (AES)*, Munich, Germany, May 8–11 1999, Conv. paper #4900.

[18] f. harris and E. Brooking, "A versatile parametric filter using an imbedded all-pass sub-filter to independently adjust bandwidth, center frequency, and boost or cut," in *Proc. Asilomar Conf. Signals, Syst. Comput.*, Pacific Grove, CA, Oct. 26–28 1992, pp. 269–273.

[19] M. Holters and U. Zölzer, "Parametric higher-order shelving filters," in *Proc. 14th European Signal Process. Conf. (EUSIPCO)*, Florence, Italy, Sept. 2006.

[20] V. Välimäki and J. D. Reiss, "All about audio equalization: Solutions and frontiers," *Appl. Sci.*, vol. 6, no. 5, 2016, Article #129.

[21] K. B. McAlpine, *Bits and Pieces: A History of Chiptunes*, Oxford Univ. Press, New York, NY, 2019.

[22] K. B. McAlpine, "The sound of 1-bit: Technical constraint and musical creativity on the 48k Sinclair ZX Spectrum," *GAME: Italian J. Game Studies*, vol. 6, no. 1, 2017.

[23] S. N. Goodwin, *Beep to Boom: The Development of Advanced Runtime Sound Systems for Games and Extended Reality*, Focal Press, New York, NY, 2019.

[24] V. Adán, *Discrete time 1-bit music: Foundations and models*, D.M.A. thesis, Columbia Univ., New York, USA, 2010.

[25] C. Ariza, "The Xenakis sieve as object: A new model and a complete implementation," *Comput. Music. J. (CMJ)*, vol. 29, no. 2, pp. 40–60, Summer 2005.

[26] K. J. Werner and M. Sanganeria, "Bit bending: An introduction," in *Proc. 16th Int. Conf. Digital Audio Effects (DAFx-13)*, Maynooth, Ireland, Sept. 2–5 2013.

[27] J. Kleimola, "Audio synthesis by bitwise logical modulation," in *Proc. 11th Int. Conf. Digital Audio Effects (DAFx-08)*, Espoo, Finland, Sept. 1–4 2008, pp. 67–70.

[28] N. M. Casey and J. A. S. Angus, "One bit digital processing of audio signals," in *Proc. 95th Conv. Audio Eng. Soc. (AES)*, New York, USA, Oct. 7–10 1993, Conv. paper #3717.

[29] J. A. S. Angus, "The one-bit alternative for audio processing and mastering," in *Proc. 9th UK Conf. Audio Eng. Soc. (AES)*, York, UK, May 1994.

[30] J. A. S. Angus and N. M. Casey, "Filtering Σ–Δ audio signals directly," in *Proc. 102nd Conv. Audio Eng. Soc. (AES)*, Munich, Germany, Mar. 22–25 1997, Conv. paper #4445.

[31] J. A. S. Angus and S. Draper, "An improved method for directly filtering Σ–Δ audio signals," in *Proc. 104th Conv. Audio Eng. Soc. (AES)*, Amsterdam, the Netherlands, May 16–19 1998, Conv. paper #4737.

[32] J. A. S. Angus, "Direct digital processing "Super Audio CD" signals," in *Proc. 108th Conv. Audio Eng. Soc. (AES)*, Paris, France, Feb. 19–22 2000, Conv. paper #5102.

[33] J. Reiss and M. Sandler, "Digital audio effects applied directly on a DSD bitstream," in *Proc. 7th Int. Conf. Digital Audio Effects (DAFx-4)*, Naples, Italy, Oct. 5–8 2004.

[34] Tufty, "1-bit mechanistic," Compact Disc, 2014.

[35] irrlicht project, "dat fuzz," Compact Disc, 2004.

[36] Mister Beep, "A thousand furious bees," Album, 2011, COUCOU Micromusic Netlabel.

[37] T. Perich, "1-bit music," Circuit in Jewel Case, 2004, Cantaloupe Music.

[38] T. Perich, "1-bit symphony," Circuit in Jewel Case, 2010, Cantaloupe Music.

[39] "Black mirror: Bandersnatch," Film, 2018, Netflix.

[40] K. J. Werner, J. S. Abel, and J. O. Smith, "The TR-808 cymbal: a physically-informed, circuit-bendable, digital model," in *Proc. 40th Int. Comput. Music Conf. (ICMC) / 11th Sound Music Comput. Conf. (SMC)*, Athens, Greece, Sept. 14–20 2014, pp. 1453–1460.

[41] K. J. Werner, J. S. Abel, and J. O. Smith, "More cowbell: a physically-informed, circuit-bendable, digital model of the TR-808 cowbell," in *Proc. 137th Conv. Audio Eng. Soc. (AES)*, Los Angeles, CA, Oct. 9–12 2014, Conv. paper #9207.

[42] K. B. McAlpine, "All aboard the impulse train: An analysis of the two-channel title music routine in Manic Miner," *Comput. Games J.*, vol. 4, no. 3–4, pp. 155–168, Dec. 2015.

[43] G. A. Miller and J. C. R. Licklider, "The intelligibility of interrupted speech," *J. Acoust. Soc. Am. (JASA)*, vol. 22, no. 2, pp. 167–173, Mar. 1950.

[44] K. J. Werner, "'Unholy Captives' from 'Return of the Obra Dinn' (Lucas Pope, 2018): a 1-bit cover/tribute," Soundcloud, 2019, `https://soundcloud.com/kurt-james-werner/unholy-captives-from-return-of-the-obra-dinn-1-bit-covertribute`.

[45] L. Pope, "Return of the Obra Dinn," Video Game, 2018.

STATISTICAL SINUSOIDAL MODELING FOR EXPRESSIVE SOUND SYNTHESIS

Henrik von Coler

Audio Communication Group
TU Berlin
Berlin, Germany
voncoler@tu-berlin.de

ABSTRACT

Statistical sinusoidal modeling represents a method for transferring a sample library of instrument sounds into a data base of sinusoidal parameters for the use in real time additive synthesis. Single sounds, capturing an instrument in combinations of pitch and intensity, are therefor segmented into attack, sustain and release. Partial amplitudes, frequencies and Bark band energies are calculated for all sounds and segments. For the sustain part, all partial and noise parameters are transformed to probabilistic distributions. Interpolated inverse transform sampling is introduced for generating parameter trajectories during synthesis in real time, allowing the creation of sounds located at pitches and intensities between the actual support points of the sample library. Evaluation is performed by qualitative analysis of the system response to sweeps of the control parameters pitch and intensity. Results for a set of violin samples demonstrate the ability of the approach to model dynamic timbre changes, which is crucial for the perceived quality of expressive sound synthesis.

1. INTRODUCTION

A system capable of expressive sound synthesis reacts to dynamic control input with the desired or appropriate changes in sound. In analysis-synthesis systems this means that the perceived timbral qualities of the synthesized sound emulate the behavior of the analyzed instrument as close as possible. Such systems thus need to capture the individual sound of an instrument and allow manipulations based on a limited set of control parameters. In order to achieve this, the synthesis approach presented in this paper, entitled *statistical sinusoidal modeling*, combines a sample based approach with a novel method for sinusoidal modeling.

Sample based synthesis in its basic form is able to capture individual sounds very accurately but does not offer manipulation techniques necessary for an expressive synthesis [1]. Sinusoidal modeling, on the other hand, is capable of wide-ranging means for sound manipulation. A key problem of sinusoidal modeling approaches, however, is the mapping of control parameters to the large amount of synthesis parameters. Statistical sinusoidal modeling can be considered a way of mapping the control parameters *pitch* and *intensity* to the parameters of a sinusoidal model. This reduced set of control parameters is often considered the central input for similar sound synthesis systems.

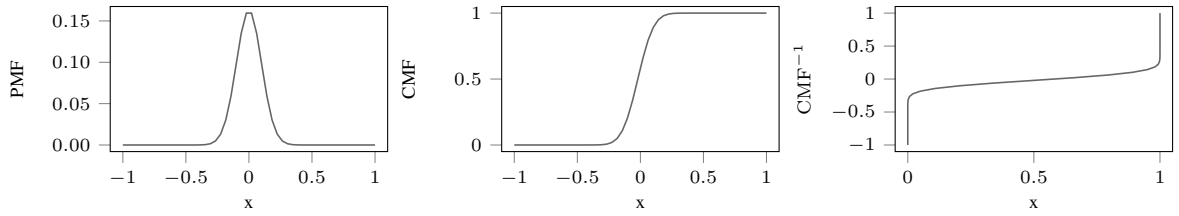
Different approaches aim at improving sample based sound synthesis. Among them are *granular synthesis* and *corpus based concatenative synthesis* [2]. Combined with spectral manipulation techniques, the flexibility of these approaches is further increased. Such combinations have proven to be effective for expressive sound synthesis. Examples include *spectral concatenative synthesis* [3] and *reconstructive phrase modeling* [4].

An *extended source-filter model* has been proposed by Hahn et al. [5, 6]. Partial and noise parameters are modeled in dependency of the control parameters pitch and intensity, by means of tensor-product B-splines. Two separate filters are used, one representing the instrument-specific features by partial index and another one capturing the frequency dependent partial characteristics. Wessel et al. [7] present a system for removing the temporal axis from the analysis data of sinusoidal models by the use of neural networks and memory-based machine learning. These methods are used to learn mappings of the three control parameters pitch, intensity and brightness to partial parameters. A system combining *corpus based concatenative synthesis* with *audio mosaicing* [8] has been proposed by Wager et al. [9]. This approach is able to synthesize an expressive target melody with arbitrary sound material by target-to-source mapping, using the features pitch, RMS energy and the modulus of the windowed Short- Time Fourier Transform.

A key feature in an expressive re-synthesis of many instruments, especially of bowed strings, are the so called spectral envelope modulations (SEM) [10]. The amplitude of each partial is modulated by its frequency in relation to the underlying frequency response of the instrument's resonant body. A vibrato in string instruments thus creates a periodic change in the relative partial amplitudes. At the typical vibrato frequencies of 5-9 Hz this effect is perceived as a timbral quality, rather than a rhythmical feature. This phenomenon, perceptually also referred to as *Sizzle* [11], contributes to the individual sound of instruments to a great extent. Using spectral modeling techniques for manipulations of instrument sounds, this effect is also considered essential for improving the quality [12]. Glissandi also result in spectral envelope modulations in the same way.

Another important effect for an expressive re-synthesis is the connection between intensity and the spectral features of the instrument's sound. Increases in intensity usually cause significant changes in the spectral distribution, respectively spectral skewness and spectral flatness [13], as well as in the tonal/noise ratio.

The proposed system is designed to encompass the above mentioned effects with simple means, enabling an efficient real time implementation. Details on the analysis, sinusoidal modeling and statistical modeling are presented in Section 2. Section 3 explains the statistical sinusoidal synthesis process in detail, followed by the evaluation of synthesis results in Section 4. The conclusion in Section 5 summarizes the findings and lists perspectives for further development.



(a) PMF (b) CMF (c) Inverted CMF

Figure 1: Exemplary probability mass function (PMF) with derived cumulative mass function (CMF) and inverted CMF.

2. ANALYSIS AND MODELING

2.1. Sample Library

The focus of the presented synthesis system rests on excitation-continuous melody instruments, using a violin as source material for the analysis stage. The *TU-Note Violin Sample Library*[14] is used for generating the statistical model. Featuring 336 single sounds and 344 two-note sequences, it has been specifically designed for this purpose. For the use in this project, the single sounds are reduced to a total of 204, consisting of 51 unique, equally spaced pitches, each captured at four dynamic levels. In the remainder, this two-dimensional space will be referred to as the *timbre plane*. It must be noted that, depending on the instrument, additional dimensions for timbre control need to be added to this space. For the violin, limited to standard techniques, the proposed reduction is acceptable. MIDI values from 0 to 127 are used to organize the dimensions pitch and velocity. Pitches range from the lowest violin note at MIDI 55 $\sim$ ***G3*** = 197.3341 Hz (443 tuning frequency) to the note at MIDI 105 $\sim$ ***A7*** = 3544 Hz. The dynamic levels ***pp***, ***mp***, ***mf*** and ***ff*** are captured in the *timbre plane*. The material has been recorded at 96 kHz with 24 bit resolution and can be downloaded using a static repository [15].

2.2. Sinusoidal Analysis

The *sinusoids+noise* model [16] is used for extracting the tonal and noise parameters for each single sound. Analysis and modeling is carried out offline, prior to the synthesis stage. Monophonic pitch tracking is performed using the YIN [17] and SWIPE [18] algorithms. Tests with more recent, real-time capable approaches [19] did not improve the performance. Based on the extracted fundamental frequency trajectories, partial trajectories are obtained by peak picking in the short-time Fourier transform (STFT), using a hop-size of 256 samples (2.76 ms) and a window size of 4096 samples, zero-padded to 8192 samples.

Quadratic interpolation (QIFFT), as presented by Smith et al. [20], is applied for estimating amplitude and frequency of up to 80 partials in each frame. The partial phases φ_i are obtained by finding the argument of the minimum when subtracting each partial with the individual amplitude a_i and frequency f_i from the complete frame x at different phases $\varphi*$:

$$\varphi_i = \arg\min \left[\sum_{n=1}^{L} (x(n) - a_i sin(2\pi f_i t(n) + \varphi^*)) \right], \quad \varphi^* = -\pi \ldots + \pi \tag{1}$$

After all partial parameters are extracted, the residual is created by subtracting the tonal part with original phases from the complete sound in the time domain. Modeling of the residual component is performed with a filter bank, based on the Bark scale, as proposed by Levine et al. [21]. The instantaneous energy trajectories of all bands are calculated using a sliding RMS with the hop-size of the sinusoidal analysis (2.76 ms) and a window length of 21.33 ms.

For each single sound, the analysis results in up to 80 partial amplitude trajectories, 80 partial frequency trajectories, 80 partial phase trajectories and 24 noise band energy trajectories. Since the original phases are not relevant for the proposed synthesis algorithm, they are not used for the further modeling steps.

2.3. Segmentation

The *TU-Note Violin Sample Library* includes manually annotated segmentation labels, based on a transient/steady state discrimination. For the single sounds they define the attack, sustain and release portion of each sound. Trajectories during attack and release segments are stored completely and additionally modeled as parametric linear and exponential trajectories. Details on the modeling and synthesis of attack and release segments are not subject to this paper. The sustain part is synthesized with the statistical sinusoidal modeling approach, explained in detail in the following section.

2.4. Statistical Modeling

After the segmentation, the above obtained trajectories of the partials and noise bands during the sustain portion of the sound are transformed into statistical distributions. Probability mass functions (PMF) with 50 equally spaced bins are created and transformed to cumulative mass functions (CMF):

$$CMF(x) = \sum_{x_i=0}^{x} PMF(x_i) \tag{2}$$

Inverse transform sampling relies on the inverted CMF for generating random number sequences with the given distribution. Figure 1 shows an exemplary PMF with the derived CMF and inverted CMF. For the synthesis algorithm, CMFs and their inversions are calculated and stored for all partial and noise trajectories during the sustain parts. $CMFs$ for the first five partials' amplitudes and frequencies are shown in Figure 2, respectively Figure 3. $CMFs$ for the first five noise band energies are shown in Figure 4. Additionally, the mean, median and standard deviation of all distributions are stored with the model.

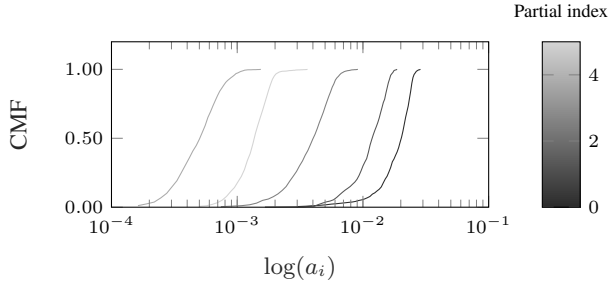


Figure 2: CMFs for the first five partial amplitudes.

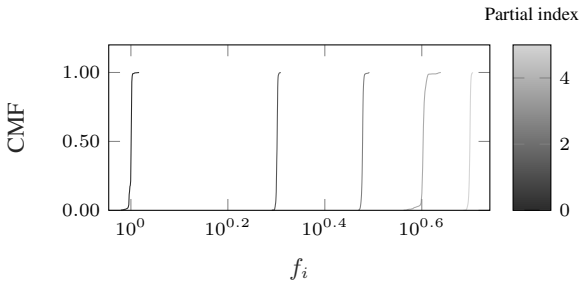


Figure 3: CMFs for the first five partial frequencies.

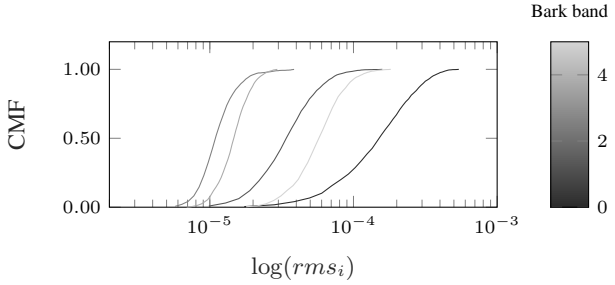


Figure 4: CMFs for the first five bark energy trajectories.

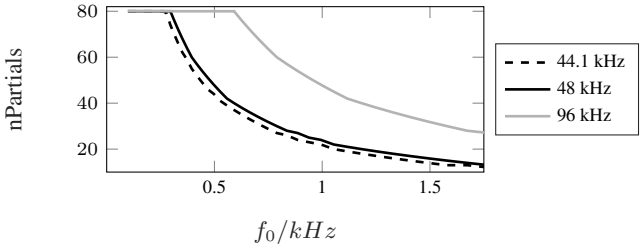


Figure 5: Number of partials synthesized, depending on sampling rate and fundamental frequency.

3. SYNTHESIS

3.1. Algorithm Overview

The implementation of the algorithm is included in a C++ based framework, using the JACK API [22]. Synthesis is performed in the time domain, with a non-overlapping approach and a frame size related to the buffer size of the audio interface. On the test system, a buffer size of 128 samples was used at a sampling rate of $f_s = 48\,\text{kHz}$, which allows a responsive use of the synthesizer. For generating a single sound, a maximum of 160 partial parameters and 24 bark band energies have to be generated each synthesis frame. The full number of 80 partials, however, is only synthesized for pitches below 600 Hz at a sampling rate of 96 kHz, respectively below 300 Hz at 48 kHz. Figure 5 shows the number of synthesized partials in dependency of sampling rate and fundamental frequency.

Listing 1: Pseudo code for the synthesis algorithm.

```
for each frame:

    get control inputs

    for all partials:
        generate random frequency value
        generate random amplitude value
        generate linear amplitude ramp
        synthesize sinusoid
        add sinusoid to output

    for all Bark bands:
        generate random band energy
        generate linear energy ramp
        apply band filter to noise signal
        add band signal to output
```

For each frame of the synthesis output, a new set of support points is generated, as shown in Listing 1. Interpolation trajectories are generated for the connection to the preceding values of partial amplitudes and noise band energies. Partial frequencies are piece-wise constant.

3.2. Statistical Value Generation

The statistical sinusoidal synthesis offers two different modes for generating parameter trajectories. For the three synthesis parameter types (partial amplitude, partial frequency and noise band energy) the mode can be selected, individually.

3.2.1. Mean/Median Mode

In the *mean/median mode*, the individual distribution functions are not used. Support points of the parameter trajectories are generated using the mean or median values stored in the model. For a constant control input, the resulting parameter trajectories remain constant, too. Variations in parameters are thus induced only through modulations of the input parameters.

3.2.2. Inverse Transform Sampling

Inverse transform sampling is a method for generating random number sequences with desired distributions from uniformly distributed random processes [23]. The inverted CMF, as shown in Figure 1c, maps the uniform distribution $\boldsymbol{\mathcal{U}}(0,1)$ to the target distribution. The method can be implemented using a sequential search method [24, p. 85], without actually inverting the distribution functions in advance. For a random value $0 \leq r \leq 1$ from the uniform distribution, the corresponding value $\tilde{r}$ from the target distribution can be obtained as the argument of the minimum of

the difference to the relevant cumulative mass function, as shown in Figure 1b:

$$\tilde{r} = \arg\min\left[CMF(x) - r\right] \tag{3}$$

In the implementation, this is realized using a vector search for Equation 3. Binary search trees can increase the efficiency of this approach and lookup tables or *guide tables* for the individual distributions are even more efficient [24]. For the chosen amount of parameters, the sequential search showed to be efficient enough to run the synthesis smoothly with 80 partials on an *Intel® Core™ i7-5500U CPU at 2.40GHz*.

3.3. Timbre Space Interpolation

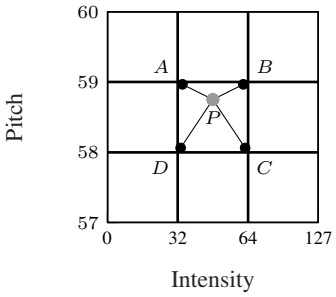


Figure 6: Interpolation in the *timbre plane*.

For the use of expressive input streams for pitch and intensity, arbitrary points in the *timbre plane* need to be synthesized. Interpolation between the support points generated by analyzing the sample library is possible for the mean/median mode as well as for the inverse transform sampling mode. Figure 6 shows a point P located in the square $ABCD$ between four support points. The weights for the parameters at each support point are calculated by the following distance-based formulae:

$$w_A = (1-x)(1-y) \tag{4}$$
$$w_B = x(1-y) \tag{5}$$
$$w_C = x \cdot y \tag{6}$$
$$w_D = (1-x)y \tag{7}$$

In the average mode, the weights w_i can be directly applied to the mean or median values m_i corresponding to the parameter values at the given points A, B, C and D for obtaining the interpolated average $\tilde{m}$:

$$\tilde{m} = w_A m_A + w_B m_B + w_C m_C + w_D m_D \tag{8}$$

In the case of inverse transform sampling, the interpolation is performed as presented in Figure 7. A single random value r is generated from a uniformly distributed random process $\boldsymbol{\mathcal{U}}(0,1)$. This value is then used to generate four random values $\tilde{r}_i$ using the $CMFs$ at the four support points. These resulting values are finally multiplied by the weights from Equations 5–7 and summed to obtain the interpolated random value $\tilde{r}^*$:

$$\begin{aligned}\tilde{r}^* = {} & w_A \tilde{r}(CMF_A) + w_B \tilde{r}(CMF_B) \\ & + w_C \tilde{r}(CMF_C) + w_D \tilde{r}(CMF_D)\end{aligned} \tag{9}$$

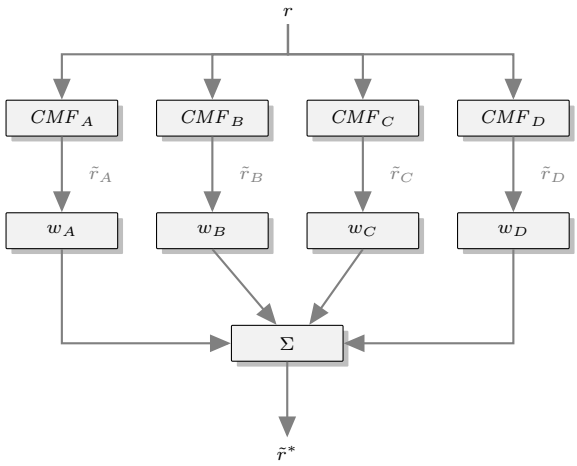


Figure 7: Interpolated inverse transform sampling.

3.4. Smoothing

The inverse transform sampling method as presented above does not consider the recent state for generating new support points. Hence, it does not capture the frequency characteristics of the analyzed trajectories. As a result, rapid changes may occur in the synthesized trajectories, which are not included in the original signals, although the resulting distribution functions are correct. For that reason, an adjustable low-pass filter is inserted after the random number generators for smoothing the trajectories. It should be noted that this filtering process narrows the distribution functions.

4. MEASUREMENTS

For evaluating the ability of the proposed synthesis algorithm to react to expressive control streams, the responses to sweeps in the frequency and intensity dimension are captured and analyzed by qualitative means. Only the deterministic component is used for this evaluation, discarding the noise.

4.1. Frequency Sweeps

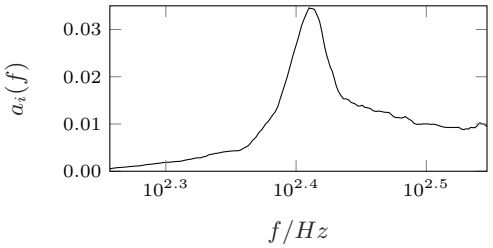
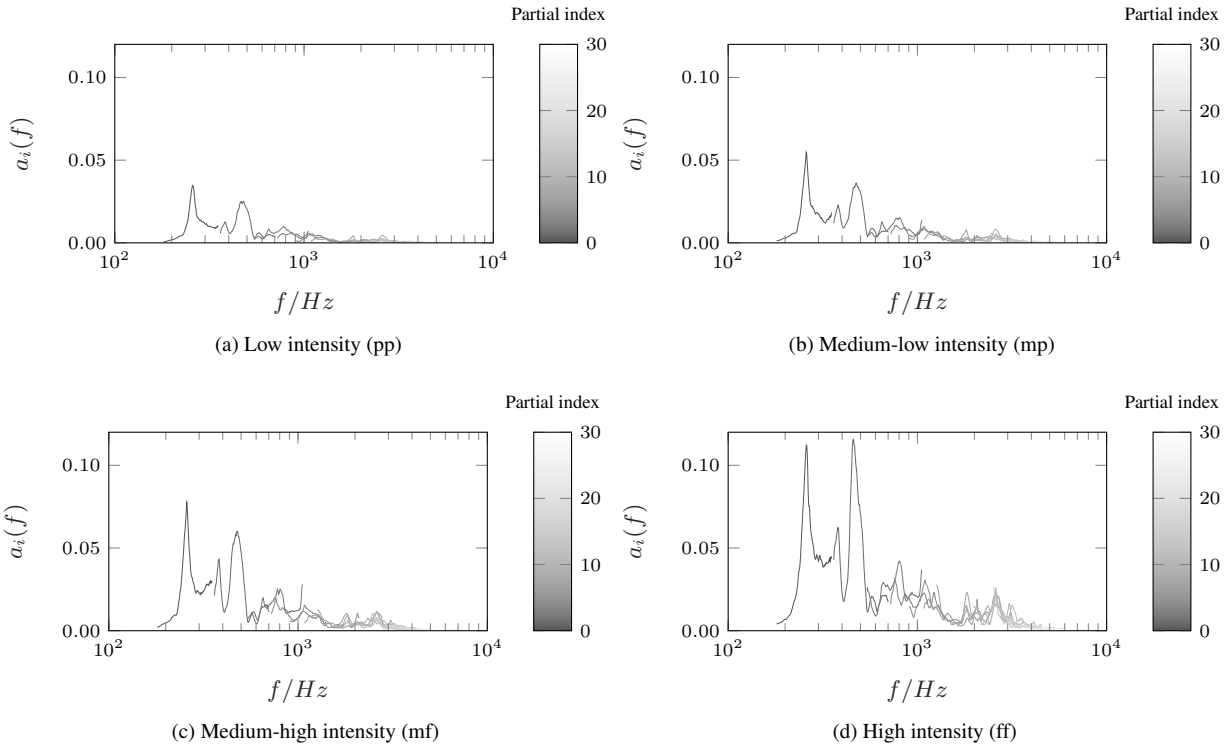


Figure 8: SEM for an octave sweep of the first partial.

For analyzing the effect of the spectral envelope modulations, a frequency sweep of one octave is sent to the synthesis system at four different intensities. The sweep ranges from the lowest tone of MIDI=55 (***G3***, 197.33 Hz) to MIDI=67 (***G4***, 394.67 Hz). The responses of all active partials to the frequency sweeps are recorded as separate signals for an analysis.



(a) Low intensity (pp)

(b) Medium-low intensity (mp)

(c) Medium-high intensity (mf)

(d) High intensity (ff)

Figure 9: Representations of the frequency response through spectral envelope modulations of 30 partials by a one octave sweep.

Figure 8 shows the amplitude of the first partial as a function of the fundamental frequency. The resulting trajectory shows no discontinuities, validating the interpolation process. It further shows a prominent peak at appropriately 257 Hz, caused by the spectral envelope modulations. Joining the amplitude over frequency trajectories of the first 30 partials, the frequency response of the instrument can be visualized through SEM. Results are shown for MIDI intensities 20 (***pp***), 50 (***mp***), 80 (***mf***) and 120 (***ff***) in Figure 9. With increasing partial index, the overlap with the neighboring partial trajectories increases. The approximated frequency responses are thus blurred for higher frequencies.

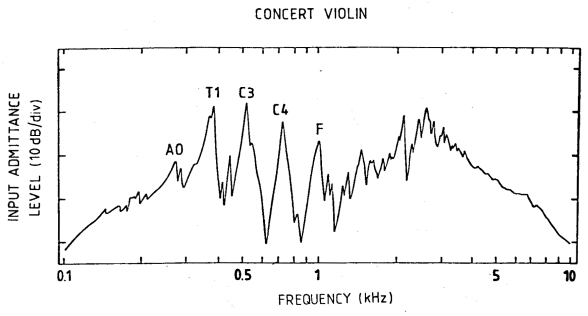


Figure 10: Input admittance at the bass bar side of an Andrea Guarneri violin [25].

All four representations of the frequency responses in Figure 9 show the same prominent peaks. These peaks correspond to the formants typical for violins. For a comparison, the input admittance of a Guarneri violin is shown in Figure 10.

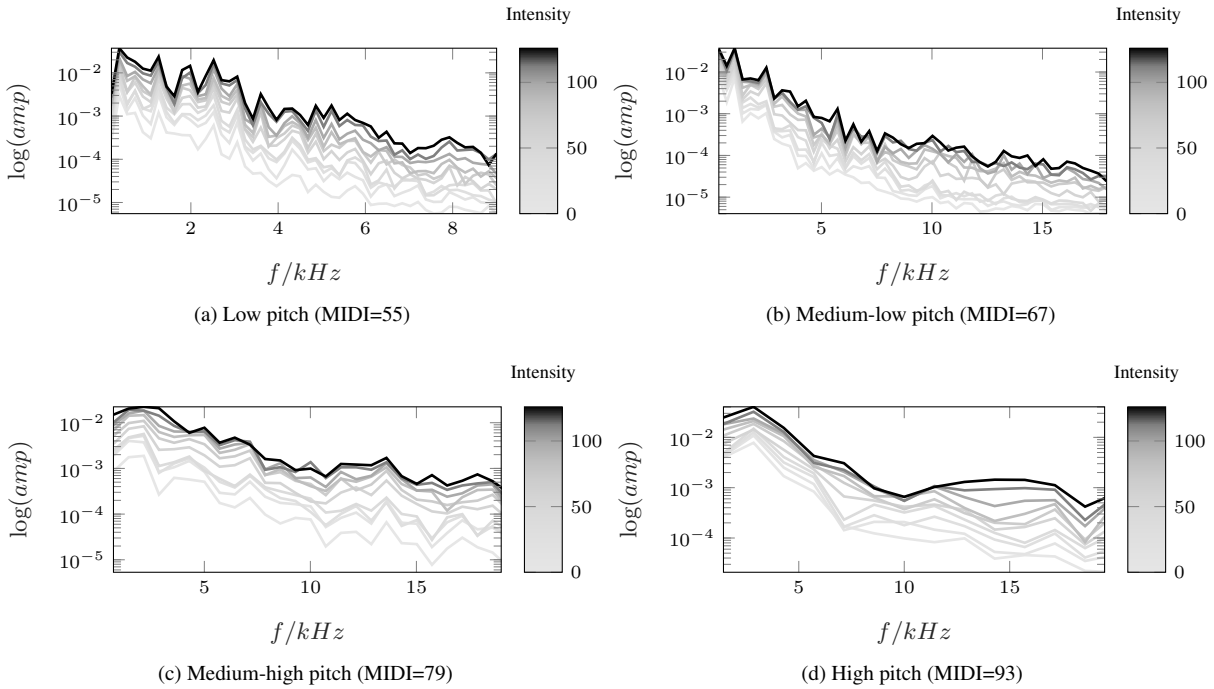
Characteristic resonances of violin bodies have been labeled inconsistently by different researchers. However, referring to Curtin et al. [11], the prominent resonances for Figure 10 are listed in Table 1. Plots 9a – 9d all show the f-hole resonance at 284 Hz and the main wood resonance, respectively the lowest corpus mode at 415 Hz. At higher intensities, the plots show peaks at 709 Hz, 872 Hz and 1170 Hz, related to the upper wood resonances and the lateral air motion. The so called *violin formant* is represented by a region of increased energy between 2000 Hz and 3000 Hz.

Table 1: Main resonances of a violin body [25, 11].

Label	**Frequency**	**Description**
A0:	275 Hz	*f-hole resonance*
C2 (T1):	460 Hz	*main wood*
C3:	530 Hz	*second wood*
C4:	700 Hz	*third wood*
F:	1000 Hz	*lateral air motion*
	2000 - 3000 Hz	*violin formant, bridge hill*

4.2. Intensity Modulations

The response of the synthesis system to changes in intensity is captured at four different pitches. Intensity sweeps from 0 to 127 are used at MIDI pitches 55 (***G3***, 197.33 Hz), 67 (***G4***, 394.67 Hz), 79 (***G5***, 789.33 Hz) and 93 (***A6***, 1772.00 Hz). The plots in Figure 11 show the spectrum of the harmonic signal in dependency of the



(a) Low pitch (MIDI=55)

(b) Medium-low pitch (MIDI=67)

(c) Medium-high pitch (MIDI=79)

(d) High pitch (MIDI=93)

Figure 11: Amplitudes of the first 50 partials in dependency of intensity, captured for four different pitches.

intensity, sampled at the partial frequencies. For higher pitches, the number of partials is reduced, resulting in a lower frequency resolution. An increase in high frequency content is indicated for higher intensities at all pitches.

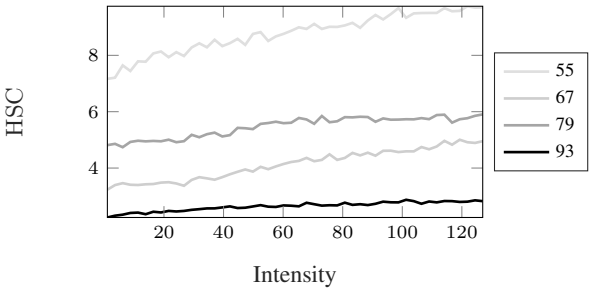


Figure 12: Harmonic spectral centroid as function of intensity for four different MIDI pitches (55, 67, 97, 93).

The harmonic spectral centroid (HSC) is calculated for 50 equally spaced points within all sweeps for analyzing the influence of the intensity on the harmonic component of the signal. Based on the spectral centroid, the HSC regards only the amplitudes a_i of the partials, resulting in a pitch independent measure for the spectral distribution of the partials:

$$\mathrm{HSC} = \frac{\sum_{i=1}^{i=N} i a_i}{\sum_{i=1}^{i=N} a_i} \tag{10}$$

Figure 12 shows the HSC as a function of the intensity for four different pitches. All trajectories show a quasi monotonic increase in the HSC with increasing intensity. Changes in intensity thus result in changes in timbre, respectively in brightness.

5. CONCLUSION

The proposed statistical sinusoidal modeling system is capable of reacting to expressive gestures, using the input parameters pitch and intensity. Evaluations of frequency and intensity sweeps show the desired responses in timbral qualities, validating the interpolated inverse transform sampling. The next important step for improving the algorithm is the implementation of a Markovian inverse transform sampling, considering past values for the random sequence generation and thus preserving the frequency characteristics of the synthesis parameters.

Using the actual inverse cumulative mass functions during runtime could further improve the performance of the algorithm. At the current state the inverse transform sampling requires a search within an unsorted vector, whereas actual inverted functions can be used by simple indexing. The flexibility and compression rate of the model could be increased by using parametric distributions instead of stored distribution functions.

Since the presented approach aims at the synthesis of sustained signals, an integration of parametric transition models [26] and trajectory models for attack and release segments is necessary for completing the synthesis system. Future experiments aim at a perceptual evaluation of synthesized sounds and expressive phrases from the full system. User studies are planned for assessing the applicability of the synthesis approach in a real-time scenario.

6. REFERENCES

[1] Roger B. Dannenberg and Istvan Derenyi, "Combining Instrument and Performance Models for High-Quality Music Synthesis," *Journal of New Music Research*, pp. 211–238, 1998.

[2] Diemo Schwarz, "A System for Data-Driven Concatenative Sound Synthesis," in *Proceedings of the COST-G6 Conference on Digital Audio Effects (DAFx-00)*, Verona, Italy, 2000.

[3] Alfonso Perez, Jordi Bonada, Esteban Maestre, Enric Guaus, and Merlijn Blaauw, "Combining Performance Actions with Spectral Models for Violin Sound Transformation," in *International Congress on Acoustics*, Madrid, Spain, 2007.

[4] Eric Lindemann, "Music Synthesis with Reconstructive Phrase Modeling," *IEEE Signal Processing Magazine [80] March 2007*, 2007.

[5] Henrik Hahn and Axel Röbel, "Extended Source-Filter Model of Quasi-Harmonic Instruments for Sound Synthesis, Transformation and Interpolation," in *Proceedings of the 9th Sound and Music Computing Conference (SMC)*, Copenhagen, Denmark, 2012, pp. 434–441.

[6] Henrik Hahn and Axel Röbel, "Extended Source-Filter Model for Harmonic Instruments for Expressive Control of Sound Synthesis and Transformation," in *Proceedings of the 16th International Conference on Digital Audio Effects (DAFx-13)*, Maynooth, Ireland, 2013.

[7] David Wessel, Cyril Drame, and Matthew Wright, "Removing the Time Axis from Spectral Model Analysis-Based Additive Synthesis: Neural Networks versus Memory-Based Machine Learning," in *Proceedings of the International Computer Music Conference (ICMC)*, Ann Arbor, Michigan, 1998, p. 62–65.

[8] Jonathan Driedger, Thomas Prätzlich, and Meinard Müller, "Let it Bee – Towards NMF-Inspired Audio Mosaicing," in *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Malaga, Spain, 2015, pp. 350–356.

[9] Sanna Wager, Liang Chen, Minje Kim, and Christopher Raphael, "Towards Expressive Instrument Synthesis Through Smooth Frame-by-Frame Reconstruction: From String to Woodwind," in *Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, USA, 2017, pp. 391–395.

[10] Ixone Arroabarren, Miroslav Zivanovic, Xavier Rodet, and Alfonso Carlosena, "Instantaneous Frequency and Amplitude of Vibrato in Singing Voice," in *Proceedings of the 2003 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Hong Kong, China, 2003, p. 537–540.

[11] Joseph Curtin and Thomas D Rossing, "Violin," in *The Science of String Instruments*, Thomas Rossing, Ed., pp. 209–244. Springer, 2010.

[12] Axel Roebel, Simon Maller, and Javier Contreras, "Transforming Vibrato Extent in Monophonic Sounds," in *Proceedings of the 14 th International Conference on Digital Audio Effects (DAFx)*, Paris, France, 2011.

[13] Stefan Weinzierl, Steffen Lepa, Frank Schultz, Erik Detzner, Henrik von Coler, and Gottfried Behler, "Sound Power and Timbre as Cues for the Dynamic Strength of Orchestral Instruments," *The Journal of the Acoustical Society of America*, vol. 144, no. 3, pp. 1347–1355, 2018.

[14] Henrik von Coler, Jonas Margraf, and Paul Schuladen, "TU-Note Violin Sample Library," TU Berlin, `http://dx.doi.org/10.14279/depositonce-6747`, 2018, Data set.

[15] Henrik von Coler, "TU-Note Violin Sample Library – A Database of Violin Sounds with Segmentation Ground Truth," in *Proceedings of the 21st International Conference on Digital Audio Effects (DAFx)*, Aveiro, Portugal, 2018.

[16] Xavier Serra and Julius Smith, "Spectral Modeling Synthesis: A Sound Analysis/Synthesis System Based on a Deterministic Plus Stochastic Decomposition ," *Computer Music Journal*, vol. 14, no. 4, pp. 12–14, 1990.

[17] Alain de Cheveigné and Hideki Kawahara, "YIN, a Fundamental Frequency Estimator for Speech and Music," *The Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1917–1930, 2002.

[18] Arturo Camacho, *Swipe: A Sawtooth Waveform Inspired Pitch Estimator for Speech and Music*, Ph.D. thesis, Gainesville, FL, USA, 2007.

[19] Orchisama Das, Julius Smith, and Chris Chafe, "Real-time Pitch Tracking in Audio Signals with the Extended Complex Kalman Filter," in *Proceedings of the 20th International Conference on Digital Audio Effects (DAFx)*, Edinburgh, UK, 2017.

[20] Julius O. Smith and Xavier Serra, "PARSHL: An Analysis/Synthesis Program for Non-Harmonic Sounds Based on a Sinusoidal Representation," in *Proceedings of the International Computer Music Conference (ICMC)*, Barcelona, Spain, 2005.

[21] Scott Levine and Julius Smith, "A Sines+Transients+Noise Audio Representation for Data Compression and Time/Pitch Scale Modifications," in *Proceedings of the 105th Audio Engineering Society Convention*, San Francisco, CA, 1998.

[22] Henrik von Coler, "A Jack-based Application for Spectro-Spatial Additive Synthesis," in *Proceedings of the 17th Linux Audio Conference (LAC-19)*, Stanford University, USA, 2019.

[23] Luc Devroye, "Sample-based Non-uniform Random Variate Generation," in *Winter Simulation Conference*, Washington, D.C., USA, 12 1986, pp. 260–265.

[24] Luc Devroye, *Non-Uniform Random Variate Generation*, Springer, McGill University, 1986.

[25] J. Alonso Moral and E. Jansson, "Input Admittance, Eigenmodes and Quality of Violins," in *STL-QPSR*, vol. 23, pp. 60–75. KTH Royal Institute of Technology, 1982.

[26] Henrik von Coler, Moritz Götz, and Steffen Lepa, "Parametric Synthesis of Glissando Note Transitions – A user Study in a Real-Time Application," in *Proceedings of the 21st International Conference on Digital Audio Effects (DAFx)*, Aveiro, Portugal, 2018.

REAL-TIME IMPLEMENTATION OF AN ELASTO-PLASTIC FRICTION MODEL APPLIED TO STIFF STRINGS USING FINITE-DIFFERENCE SCHEMES

Silvin Willemsen
Multisensory Experience Lab, CREATE,
Aalborg University Copenhagen
Copenhagen, Denmark
sil@create.aau.dk

Stefan Bilbao
Acoustics and Audio Group
University of Edinburgh
Edinburgh, UK
s.bilbao@ed.ac.uk

Stefania Serafin
Multisensory Experience Lab, CREATE,
Aalborg University Copenhagen
Copenhagen, Denmark
sts@create.aau.dk

ABSTRACT

The simulation of a bowed string is challenging due to the strongly non-linear relationship between the bow and the string. This relationship can be described through a model of friction. Several friction models in the literature have been proposed, from simple velocity dependent to more accurate ones. Similarly, a highly accurate technique to simulate a stiff string is the use of finite-difference time-domain (FDTD) methods. As these models are generally computationally heavy, implementation in real-time is challenging. This paper presents a real-time implementation of the combination of a complex friction model, namely the elasto-plastic friction model, and a stiff string simulated using FDTD methods. We show that it is possible to keep the CPU usage of a single bowed string below 6 percent. For real-time control of the bowed string, the Sensel Morph is used.

1. INTRODUCTION

In physical modelling sound synthesis applications, the simulation of a bowed string is a challenging endeavour. This is mainly due to the strongly non-linear relationship between the bow and the string, through a model of friction. Such friction models can be categorised as static or dynamic; models of the latter type have only recently seen a major effort. As opposed to static friction models, where friction depends only on the relative velocity of the two bodies in contact, dynamic models describe the friction force through a differential equation.

A recently popular dynamic model is the elasto-plastic model, first proposed in [1]. The model assumes that the friction between the two objects in contact is caused by a large ensemble of bristles, each of which contributes to the total friction force. The average bristle deflection is used as an extra independent variable for calculating the friction force. As shown in [2], the elasto-plastic model can be applied to a bowed string simulation and it exhibits a hysteresis loop in the force versus velocity plane due to this multivariable dependency. This is consistent with measurements performed using a bowing machine in [3]. The elasto-plastic model has been thoroughly investigated in a musical context by Serafin et al. in [2, 4, 5].

Regarding bowed string simulations, the first musical non-linear systems, including bowed strings, were presented by McIntyre, et al. in [6]. Smith published the first real-time implementation of the bowed string using a digital waveguide (DW) for the string and a look-up table for the friction model in [7]. Simultaneously, Florens, et al. published a real-time implementation using mass-spring systems for the string and a static friction model for the bow in [8].

The dynamics of musical instruments are generally described by systems of partial differential equations (PDEs). Specialised synthesis methods such as DWs [9] and modal synthesis [10] are derived from particular solutions. Mainstream time-stepping methods such as finite-difference time-domain (FDTD) methods were first proposed in [11, 12, 13], and developed subsequently [14, 15]. In [16] the authors adapted the thermal model proposed by Woodhouse in [3] for real-time applications using a DW for the string implementation and a combination of the DW and FDTD methods for the bowing interaction. In [17, 18], Desvages used FDTD methods for the implementation of the string in two polarizations and a static double exponential friction model introduced in [19]. This was, however, not implemented in real-time. To the best of the authors' knowledge, the only known real-time implementation of any bow model applied to complete FDTD strings was presented in [20] where the soft exponential friction function presented in [14] was used. The current work can be considered an extension of this work.

We are interested in bridging the gap between highly accurate physical models and efficient implementations so that these models can be played in real-time. In this work, we present an implementation of the elasto-plastic friction model in conjunction with a finite-difference implementation of the damped stiff string. Furthermore, we show that it is possible to play the string in real-time using the Sensel Morph controller [21].

This paper is structured as follows. In Section 2, the elasto-plastic bow model in conjunction with a PDE model for a stiff string is described. Discretisation is covered in Section 3, and implementation details appear in Section 4. In Section 5, simulated results are presented and discussed. Some concluding remarks appear in Section 6.

2. ELASTO-PLASTIC BOW MODEL

Consider a linear model of transverse string vibration in a single polarization, where $u(x,t)$ represents string displacement as a function of time $t \geq 0$, in s, and coordinate $x \in [0,\ L]$ (in m) for some string length L (in m). Using the subscripts t and x to denote differentiation with respect to time and space respectively, a partial differential equation describing the dynamics of the damped stiff string is [14]

$$u_{tt} = c^2 u_{xx} - \kappa^2 u_{xxxx} - 2\sigma_0 u_t + 2\sigma_1 u_{txx}. \qquad (1)$$

Here, $c = \sqrt{T/\rho A}$ is the wave speed (in m/s) with tension T (in N), material density ρ (in kg·m^{-3}) and cross-sectional area A (in m^2). Furthermore, $\kappa = \sqrt{EI/\rho A}$ is the stiffness coefficient (in m^2/s) with Young's Modulus E (in Pa) and area moment of inertia I (in m^4). For a string of circular cross section we have radius r (in m), cross-sectional area $A = \pi r^2$ and area moment of inertia $I = \pi r^4/4$. Lastly, $\sigma_0 \geq 0$ (in s^{-1}) and $\sigma_1 \geq 0$ (in m^2/s) are coefficients allowing for frequency-independent and frequency-dependent damping respectively.

In our implementation we assume simply supported boundary conditions, which are defined as

$$u = u_{xx} = 0 \quad \text{where} \quad x = 0, L\,. \tag{2}$$

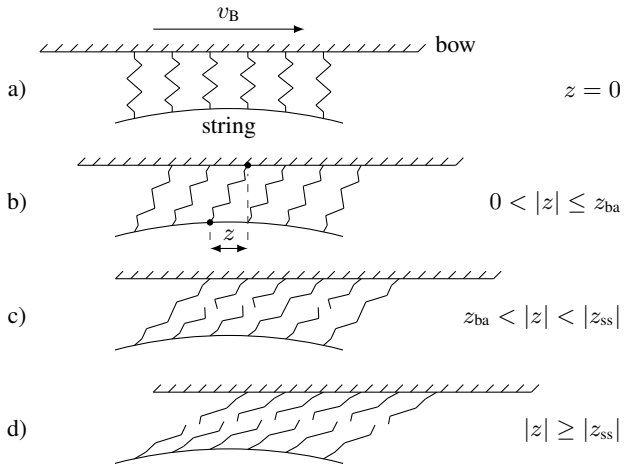


Figure 1: *Microscopic displacements of the bristles between the bow and the string. The bow moves right with a velocity of v_B. a) The initial state is where the average bristle displacement $z = 0$. b) The bow has moved right relative to the string. The purely elastic, or presliding regime is entered (stick). c) After the break-away displacement z_{ba}, more and more bristles start to 'break'. This is defined as the elasto-plastic regime. d) After all bristles have 'broken', the steady state (slip) is reached and the purely plastic regime is entered.*

As mentioned in the introduction, the elasto-plastic bow model assumes that the friction between the bow and the string is due to a large ensemble of bristles, each of which contributes to the total friction force. See Figure 1 for a graphical representation of this. The bristles are assumed to be damped stiff springs and can 'break' after a given break-away displacement threshold. An extra term can be added to (1) to include the bowing interaction

$$u_{tt} = \ldots - \delta(x - x_{\mathrm{B}}) f(v, z)/\rho A. \tag{3}$$

Here, the spatial Dirac delta function $\delta(x - x_{\mathrm{B}})$ (in m^{-1}) allows for the pointwise application of the force f (in N) at externally supplied bowing position $x_{\mathrm{B}}(t)$ (in m).

In the following we will use the definitions found in [1]. The force f is defined in terms of the relative velocity v (in m/s) and average bristle displacement z (in m) (see Figure 1) as

$$f(v, z) = s_0 z + s_1 \dot{z} + s_2 v + s_3 w, \tag{4}$$

where

$$v = u_t(x_{\mathrm{B}}) - v_{\mathrm{B}}, \tag{5}$$

where $v_{\mathrm{B}}(t)$ is an externally supplied bow velocity, s_0 is the bristle stiffness (in N/m), s_1 is the damping coefficient (in kg/s), s_2 is the viscous friction (in kg/s) and s_3 is a dimensionless noise coefficient multiplied onto pseudorandom function $w(t)$ (in N) as done in [4] and adds noise to the friction force. Here, $\dot{z}$ indicates a time derivative of z, and is related to v through

$$\dot{z} = r(v, z) = v\left[1 - \alpha(v, z)\frac{z}{z_{\mathrm{ss}}(v)}\right], \tag{6}$$

where z_{ss} is the steady-state function

$$z_{\mathrm{ss}}(v) = \frac{\mathrm{sgn}(v)}{s_0}\left[f_{\mathrm{C}} + (f_{\mathrm{S}} - f_{\mathrm{C}})e^{-(v/v_{\mathrm{S}})^2}\right], \tag{7}$$

with Stribeck velocity v_{S} (in m/s), Coulomb force $f_{\mathrm{C}} = f_{\mathrm{N}}\mu_{\mathrm{C}}$ and stiction force $f_{\mathrm{S}} = f_{\mathrm{N}}\mu_{\mathrm{S}}$ (both in N). Here μ_{C} and μ_{S} are the dynamic and static friction coefficient respectively and $f_{\mathrm{N}}(t)$ is the normal force (in N) which is, like $v_{\mathrm{B}}(t)$, externally supplied. See Figure 2 for a plot of (7).

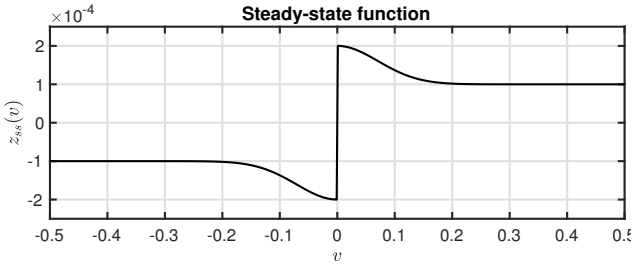


Figure 2: *A plot of the steady-state function $z_{ss}(v)$ with a force of 5 N.*

Furthermore, the adhesion map between the bow and the string is defined as

$$\alpha(v, z) = \begin{cases} \left.\begin{cases} 0 & |z| \leq z_{\mathrm{ba}} \\ \alpha_{\mathrm{m}}(v, z) & z_{\mathrm{ba}} < |z| < |z_{\mathrm{ss}}(v)| \\ 1 & |z| \geq |z_{\mathrm{ss}}(v)| \end{cases}\right\} & \text{if } \mathrm{sgn}(v) = \mathrm{sgn}(z) \\ 0 & \text{if } \mathrm{sgn}(v) \neq \mathrm{sgn}(z), \end{cases} \tag{8}$$

where the transition between the elastic and plastic behaviour is defined as

$$\alpha_{\mathrm{m}} = \frac{1}{2}\left[1 + \mathrm{sgn}(z)\sin\left(\pi\frac{z - \mathrm{sgn}(z)\frac{1}{2}(|z_{\mathrm{ss}}(v)| + z_{\mathrm{ba}})}{|z_{\mathrm{ss}}(v)| - z_{\mathrm{ba}}}\right)\right], \tag{9}$$

with break-away displacement z_{ba}, i.e., where the bristles start to break (see Figure 1 c)). A plot of the adhesion map can be found in Figure 3.[1]

One of the difficulties in working with this model is that, due to the many approximations, the notion of an energy balance, relat-

[1] It is interesting to note is that in the literature on this topic such as [1, 2, 4, 5], a few inaccuracies can be found in the definition of $\alpha(v, z)$: 1) all uses of z_{ss} in (8) and (9) lack the absolute value operator, 2) the multiplications with $\mathrm{sgn}(z)$ in (9) are excluded, 3) $\alpha(v, z)$ is undefined for $|z| = z_{\mathrm{ba}}$ and $|z| = |z_{\mathrm{ss}}(v)|$ (correct in the original paper by Dupont et al. [1]). It can be shown that only with the definitions presented here, is it possible to obtain the curve shown in Figure 3.

ing the rate of stored energy in the system to power input and loss is not readily available. Such energy methods are used frequently in the context of physical modeling synthesis and virtual analog modeling as a means of arriving at numerical stability conditions for strongly nonlinear systems, as is the present case. See, e.g., [14]. This means that we do not have a means of ensuring numerical stability in the algorithm development that follows. This does not mean, however, that an energy balance is not available.

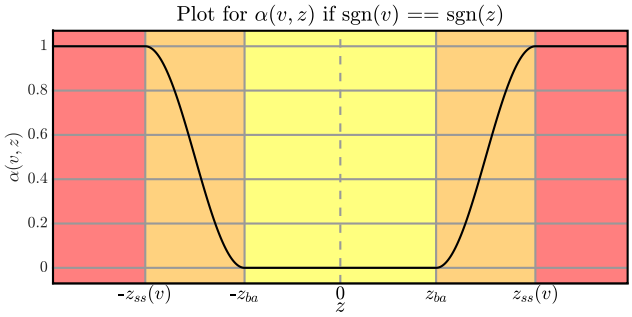


Figure 3: *A plot of the adhesion map $\alpha(v,z)$ plotted against z when the signs of v and z are the same. The different regions of the map are shown with the coloured areas and correspond to Figure 1 according to: yellow - a) & b), orange - c) and red - d).*

3. DISCRETISATION

Finite-difference schemes for the stiff string in isolation are covered by various authors [13, 14].

Equation (1) can be discretised at times $t = nk$, with sample $n \in \mathbb{N}$ and time-step $k = 1/f_s$ (in s) with sample-rate f_s (in Hz) and locations $x = lh$, where grid spacing h (in m) needs to abide the following condition [14]

$$h \geq h_{\min} = \sqrt{\frac{c^2k^2 + 4\sigma_1 k + \sqrt{(c^2k^2 + 4\sigma_1 k)^2 + 16\kappa^2 k^2}}{2}} \tag{10}$$

and grid points $l \in [0, ..., N]$, where $N = \text{floor}(L/h)$ and $N + 1$ is the total number of grid points. It is important to note that the closer h is to $h_{\min}$, the more accurate the scheme will be. Approximations for the derivatives found in (1) are described in the following way [14]:

$$u_t \approx \delta_{t\cdot} u_l^n = \frac{1}{2k}\left(u_l^{n+1} - u_l^{n-1}\right), \tag{11a}$$

$$u_{tt} \approx \delta_{tt} u_l^n = \frac{1}{k^2}\left(u_l^{n+1} - 2u_l^n + u_l^{n-1}\right), \tag{11b}$$

$$u_{xx} \approx \delta_{xx} u_l^n = \frac{1}{h^2}\left(u_{l+1}^n - 2u_l^n + u_{l-1}^n\right), \tag{11c}$$

$$u_{txx} \approx \delta_{t-}\delta_{xx} u_l^n = \frac{1}{hk^2}\left(u_{l+1}^n - 2u_l^n + u_{l-1}^n - u_{l+1}^{n-1} + 2u_l^{n-1} - u_{l-1}^{n-1}\right), \tag{11d}$$

$$u_{xxxx} \approx \delta_{xxxx} u_l^n = \frac{1}{h^4}\left(u_{l+2}^n - 4u_{l+1}^n + 6u_l^n - 4u_{l-1}^n + u_{l-2}^n\right), \tag{11e}$$

with grid function u_l^n denoting a discretised version of $u(x,t)$ at the nth time step and the lth point on the string. Note that in (11d), the backwards time difference operator is used to keep (12) explicit and thus computationally cheaper to update. Using the approximations shown in (11), (3) can be discretised to

$$\begin{aligned}\delta_{tt} u_l^n = {} & c^2\delta_{xx} u_l^n - \kappa^2\delta_{xxxx} u_l^n - 2\sigma_0\delta_{t\cdot} u_l^n \\ & + 2\sigma_1\delta_{t-}\delta_{xx} u_l^n - J(x_{\text{B}}^n) f(v^n, z^n)/\rho A,\end{aligned} \tag{12}$$

where the relative velocity described in (5) can be discretised as

$$v^n = I(x_{\text{B}}^n)\delta_{t\cdot} u_l^n - v_{\text{B}}^n. \tag{13}$$

Here, $I(x_{\text{B}}^n)$ and $J(x_{\text{B}}^n)$ are weighting functions where the former interpolates the string displacement and velocity and the latter distributes the bowing term around time-varying bowing position x_{B}^n (see Figure 4 and [14] for more details on this). Furthermore,

$$f(v^n, z^n) = s_0 z^n + s_1 r^n + s_2 v^n + s_3 w^n \tag{14}$$

is the discrete counterpart of (4) where

$$r^n = r(v^n, z^n) = v^n\left[1 - \alpha(v^n, z^n)\frac{z^n}{z_{\text{ss}}(v^n)}\right] \tag{15}$$

is the discrete counterpart of (6).

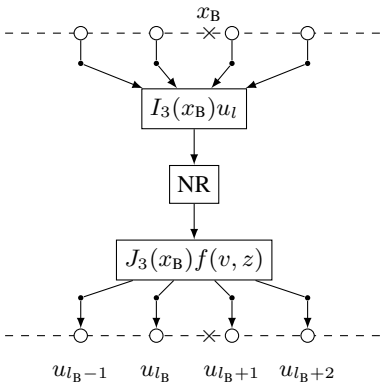


Figure 4: *Cubic interpolation at bowing point x_B. The interpolator I retrieves the values of four grid points which are then used in the Newton-Raphson (NR) solver. This outputs the force function $f(v,z)$ that the spreading function J in turn distributes over the same four grid points. This process happens every single sample.*

At the bowing point we need to iteratively solve for two unknown variables: the relative velocity between the bow and the string v^n and the mean bristle displacement z^n of the bow at sample n. We can solve (12) at x_{B}^n using (13) and identity [14]

$$\delta_{tt} u_l^n = \frac{2}{k}\left(\delta_{t\cdot} u_l^n - \delta_{t-} u_l^n\right) \tag{16}$$

resulting in

$$I(x_{\text{B}}^n)J(x_{\text{B}}^n) f(v^n, z^n)/\rho A + \left(\frac{2}{k} + 2\sigma_0\right)v^n + b^n = 0, \tag{17}$$

where

$$b^n = \frac{2}{k}v_{\mathrm{B}}^n - \frac{2}{k}I(x_{\mathrm{B}}^n)\delta_{t-}u_l^n - c^2 I(x_{\mathrm{B}}^n)\delta_{xx}u_l^n + \kappa^2 I(x_{\mathrm{B}}^n)\delta_{xxxx}u_l^n + 2\sigma_0 v_{\mathrm{B}}^n - 2\sigma_1 I(x_{\mathrm{B}}^n)\delta_{t-}\delta_{xx}u_l^n \tag{18}$$

and can be pre-computed as its terms are not dependent on v^n or z^n. Recalling (4), this can be rewritten to

$$I(x_{\mathrm{B}}^n)J(x_{\mathrm{B}}^n)\left(\frac{s_0 z^n + s_1 r^n + s_2 v^n + s_3 w^n}{\rho A}\right) + \left(\frac{2}{k} + 2\sigma_0\right)v^n + b^n = 0. \tag{19}$$

To obtain the values of v^n and z^n, multivariate Newton-Raphson (NR) is used. If (19) is defined to be $g_1 = g_1(v^n, z^n)$ and

$$g_2(v^n, z^n) = r^n - a^n = 0, \tag{20}$$

with

$$a^n = (\mu_{t-})^{-1}\delta_{t-}z^n \tag{21}$$

(where the operators applied to z^n denote the trapezoid rule [14]) we obtain the following iteration

$$\begin{bmatrix} v_{(i+1)}^n \\ z_{(i+1)}^n \end{bmatrix} = \begin{bmatrix} v_{(i)}^n \\ z_{(i)}^n \end{bmatrix} - \begin{bmatrix} \frac{\partial g_1}{\partial v} & \frac{\partial g_1}{\partial z} \\ \frac{\partial g_2}{\partial v} & \frac{\partial g_2}{\partial z} \end{bmatrix}^{-1} \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}, \tag{22}$$

where i is the iteration number capped by 50 iterations, and the convergence threshold is set to 10^{-7}.

4. IMPLEMENTATION

In this section, we will elaborate on the implementation; the parameters used and the system architecture. The real-time implementation of the discrete-time model shown in the previous section has been done using C++ together with the JUCE framework [22]. The application is shown in Figure 5. The parameters we used can be found in Table 1, most of which are based on implementations by Serafin in [4]. These parameters will be static, i.e., are not user-controlled (except for z_{ba} and s_3 which rely on f_{N}). A demonstrative video can be found in [23]. We use the passivity condition proposed by [24] for our choices of different parameter-values. As this condition applies to the LuGre model first proposed in [25, 26] from which the elasto-plastic model evolved, further investigation is required to conclude whether these conditions are identical for the elasto-plastic model.

4.1. Sensel Morph

As mentioned in Section 1, the Sensel Morph (or Sensel for short) is used as an interface to control the bowed string (see Figure 6). The Sensel is a highly sensitive touch controller containing ca. 20,000 pressure sensitive sensors that allow for expressive control of the implementation [21].

4.2. Interaction

The first finger the Sensel registers is linked to the following parameters: the normal force of the bow f_{N} (finger pressure), the bowing velocity v_{B} (vertical finger velocity) and bowing position x_{B} (horizontal finger position). The parameters are limited by the following conditions: $0 \leq f_{\mathrm{N}} \leq 10$, $-0.3 \leq v_{\mathrm{B}} \leq 0.3$ and

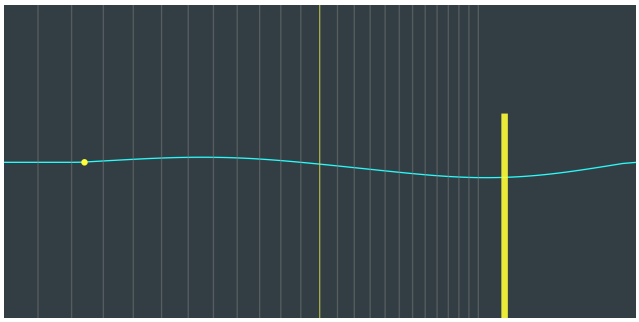


Figure 5: *The elasto-plastic bowed string application. The bow is shown as a yellow rectangle, moves on interaction and its opacity depends on the finger force. The state* $\mathbf{u}^n$ *is visualised using the cyan curve and stopping-finger position is shown as a yellow circle. The grey lines show the 'frets' corresponding to semi-tones as a visual reference for the stopping position and do not influence the model.*

Figure 6: *The Sensel Morph: an expressive touch sensitive controller used for controlling the real-time elasto-plastic bowed string implementation.*

$0 < x_{\mathrm{B}} < L$. The second finger acts as a stopping finger on the string. As done in [20], for a string stopped at location $x_{\mathrm{f}} \in [0, L]$ and $l_{\mathrm{f}} = \mathrm{floor}(x_{\mathrm{f}}/h)$ we use

$$u_l^n = \begin{cases} 0, & l = l_{\mathrm{f}} - 1 \vee l = l_{\mathrm{f}} \\ (1 - \alpha_{\mathrm{f}}^{\epsilon})u_l^n, & l = l_{\mathrm{f}} + 1 \\ u_l^n, & \text{otherwise} \end{cases} \tag{23}$$

where $\alpha_{\mathrm{f}} = x_{\mathrm{f}}/h - l_{\mathrm{f}}$ and $\epsilon = 7$ is a heuristic value that has been found to most linearly alter pitch between grid points.

4.3. System Architecture

Implementation of the scheme shown in (12) starts by expanding the operators shown in (11) and solving for the state at the next sample $\mathbf{u}^{n+1}$ where $\mathbf{u}$ is a vector containing the values for all grid points $l \in [0, ..., N]$.

An overview of the system architecture can be found in Figure 7. The three main components of the application are the Sensel

Table 1: *Parameter values. Values for the fundamental frequency f_0 can be found in Section 5.*

Parameter	Symb. (unit)	Value (notes)
Material Density	ρ (kg·m^{-3})	7850
Radius	r (m)	$5 \cdot 10^{-4}$
String length	L (m)	1
Wave speed	c (m/s)	$2f_0/L$
Young's modulus	E (Pa)	$2 \cdot 10^{11}$
Freq. indep. damping	σ_0 (s^{-1})	1
Freq. dep. damping	σ_1 (m^2/s)	$5 \cdot 10^{-3}$
Coulomb friction	μ_C (-)	$0.3\ (< \mu_S)$
Static friction	μ_S (-)	$0.8\ (> \mu_C)$
Normal force	f_N (N)	10
Bow velocity	v_B (m/s)	0.1
Bow position	x_B (m)	0.25
Stribeck velocity	v_S (m/s)	0.1
Bristle stiffness	s_0 (N/m)	10^4
Bristle damping	s_1 (kg/s)	$0.001\sqrt{s_0}$
Viscous friction	s_2 (kg/s)	0.4
Noise coefficient	s_3 (-)	$0.02f_N$
Pseudorandom func.	w (N)	$-1 < w < 1$
Break-away disp.	z_{ba} (m)	$0.7f_C/s_0\ (< f_C/s_0)$
Sample rate	f_s (Hz)	44,100
Time step	k (s)	$1/f_s$

controlling the application, the violin string class that performs the simulation and the main application class that moderates between these and the auditory and visual outputs. The black arrows indicate instructions that one of these components can give to another and the hollow arrows indicate data flows. Moreover, the arrows are accompanied by coloured boxes, depicting what thread the instruction or data flow is associated with and at what rate this runs.

The graphics thread has the lowest priority, is denoted by the green boxes and runs at 15 Hz. The redraw instruction merely retrieves the current string state $\mathbf{u}^n$ and bow and finger position and visualises this as shown in Figure 5.

The thread checking and receiving data from the Sensel runs at 150 Hz and is denoted by the blue boxes. The parameters that the user interacts with (bowing force, velocity and position) are also updated at this rate.

The highest priority thread is the audio thread denoted by the orange boxes and runs at 44,100 Hz. The violin string class gets updated at this rate and performs operations in the order shown in Algorithm 1.

5. RESULTS AND DISCUSSION

Figure 8 shows the output waveforms for a string with $f_0 = 440$ Hz at different points along the string. The bowing parameters are $f_N = 5$ N and $v_B = 0.1$ m/s. The figure shows the traditional Helmholtz motion, which is the characteristic motion of a bowed string.

To test whether the implementation exhibits a hysteresis loop, the force vs. relative velocity plane was visualised. In Figure 9, this plot can be found for which the same parameters have been used. The figure shows values for 500 samples around $t = 0.5f_s$. As can be seen from the figure, the hysteresis loop is achieved and is similar to the one observed in [19]. The group of values around

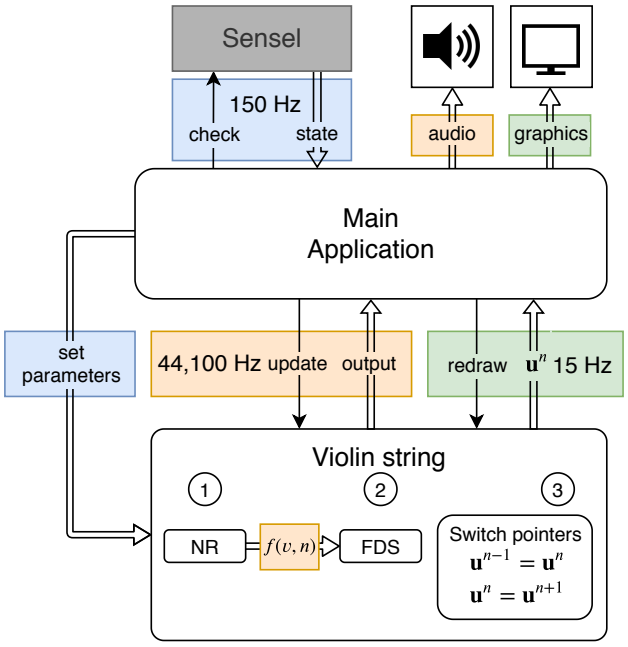


Figure 7: *The system architecture. See Section 4.3 for a thorough explanation.*

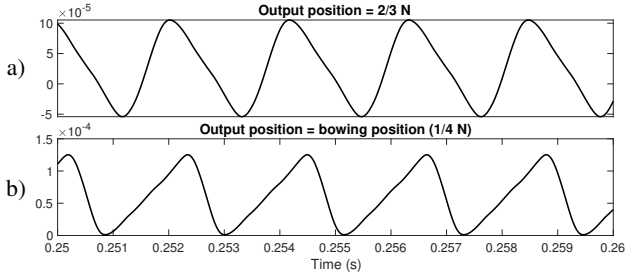


Figure 8: *Output waveforms of the simulation at different positions along the string where N denotes the number of points of the string ($f_0 = 440$ Hz, $f_N = 5$ N and $v_B = 0.1$ m/s).*

$v = 0$ are due to the sticking behaviour, and the others (the loop on the left) to the slipping behaviour.

For testing the speed of the algorithm, a MacBook Pro with a 2.2 GHz Intel Core i7 processor was used. The algorithm was tested using different frequencies according to the violin tuning of empty strings: $f_0 = 196.0$ (G3), 293.66 (D4), 440.0 (A4) and 659.26 (E5) Hz corresponding to $N = 95$, 71, 49, and 33 grid points respectively. The results can be seen in Table 2. When the total number of strings is smaller than 4, always the lowest frequency strings are used.

From Table 2 it can be observed that for one string, the CPU usage is $< 6\%$ with the graphics thread disabled. This is a great result, given the fact that both the bow and the string model are computationally complex. Empirical investigation shows that the NR algorithm converges after ca. 3-4 iterations and the capping of 50 iterations never has to be used. A single string (but also more) could thus safely be used as an audio plugin in parallel to others without the user having to worry about auditory dropouts.

for *t = 1:lengthSound* **do**
  calculate computable part b^n (Eq. (18))
  $\epsilon = 1$
  $i = 0$
  while $\epsilon < tol \wedge i < 50 \wedge f_C > 0$ **do**
    calculate..
    1. $z_{\text{ss}}(v^n_{(i)})$ (Eq. (7) in discrete-time)
    2. $\alpha(v^n_{(i)}, z^n_{(i)})$ (Eq. (8) in discrete-time)
    3. $r(v^n_{(i)}, z^n_{(i)})$ (Eq. (15))
    4. g_1, g_2 (Eqs. (19) and (20))
    5.–9. Compute derivatives of 1.–4. in the same order.
    10. Perform vector NR to obtain $v^n_{(i+1)}$ and $z^n_{(i+1)}$
    11. Calculate ϵ: $\epsilon = \left\| \begin{bmatrix} v^n_{(i+1)} \\ z^n_{(i+1)} \end{bmatrix} - \begin{bmatrix} v^n_{(i)} \\ z^n_{(i)} \end{bmatrix} \right\|$
    12. Increment i: $i = i + 1$
  end
  Repeat 1.–3. using the values for v^n and z^n from the NR iteration.
  Calculate $f(v^n, z^n)$ (Eq. (14))
  Calculate $\mathbf{u}^{n+1}$ (Eq. (12) expanded)
  $\mathbf{u}^{n-1} = \mathbf{u}^n$
  $\mathbf{u}^n = \mathbf{u}^{n+1}$
end

Algorithm 1: Pseudocode showing the order of calculations.

6. CONCLUSIONS

In this paper, we presented a real-time implementation of an elasto-plastic friction model with applications to a bow exciting a string, discretised using a finite-difference approach.

With a single string we are able to keep the CPU usage down to $< 6\%$ making for an efficient implementation that could be used in parallel with other virtual instruments or plugins.

Future work includes parameter design and including an instrument body for more realistic sounding results, as well as listening tests to verify the perceivable differences between simpler friction models versus the elasto-plastic model.

7. ACKNOWLEDGMENTS

Many thanks to the anonymous reviewers for giving their valuable input. This work is supported by NordForsk's Nordic University Hub Nordic Sound and Music Computing Network NordicSMC, project number 86892.

8. REFERENCES

[1] P. Dupont, V. Hayward, B. Armstrong, and F. Altpeter, "Single state elastoplastic friction models," *IEEE Transactions on Automatic Control*, vol. 47, no. 5, pp. 787–792, 2002.

[2] S. Serafin, F. Avanzini, and D. Rocchesso, "Bowed string simulation using an elasto-plastic friction model," *SMAC 03*, pp. 95–98, 2003.

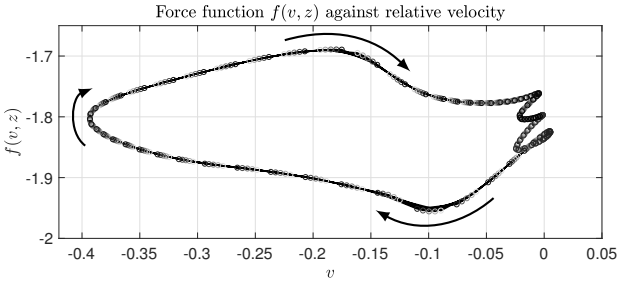


Figure 9: *Hysteresis loop showing 500 values. The values around $v = 0$ are due to sticking behaviour and the loop on the left is due to slipping behaviour.*

Table 2: *CPU usage for different amounts of strings. The values are averages over a 10 s period both for the enabled and disabled graphics thread. All strings are bowed simultaneously (polyphonically).*

Amount of strings	Graphics (%)	No graphics (%)
1	44.8	5.95
2	47.7	9.54
3	52.8	12.1
4	60.9	17.9

[3] J. Woodhouse, "Bowed string simulation using a thermal friction model," *Acta Acustica united with Acustica*, vol. 89, pp. 355–368, 2003.

[4] S. Serafin, *The Sound of Friction: Real-time Models, Playability and Musical Applications*, Ph.D. thesis, CCRMA, 2004.

[5] F. Avanzini, S. Serafin, and D. Rocchesso, "Interactive simulation of rigid body interaction with friction-induced sound generation," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 5, pp. 1073–1080, 2005.

[6] M.E. McIntyre, R.T. Schumacher, and J. Woodhouse, "On the oscillations of musical instruments," *Journal of the Acoustical Society of America*, vol. 74, no. 5, pp. 1325–1345, 1983.

[7] J.O. Smith, "Efficient simulation of the reed bore and bow string mechanics," *ICMC 86*, pp. 275–280, 1986.

[8] J.-L. Florens, A. Razafindrakoto, A. Luciani, and C. Cadoz, "Optimized real time simulation of objects for musical synthesis and animated image synthesis," *ICMC 86*, pp. 65–70, 1986.

[9] J.O. Smith, "Physical modeling using digital waveguides," *Computer Music Journal*, vol. 16, no. 4, pp. 74–91, 1992.

[10] J.D. Morrison and J.-M. Adrien, "Mosaic: A framework for modal synthesis," *Computer Music Journal*, vol. 17, no. 1, pp. 45–56, 1993.

[11] L. Hiller and P. Ruiz, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part I," *Journal of the Audio Engineering Society*, vol. 19, no. 6, pp. 462–470, 1971.

[12] L. Hiller and P. Ruiz, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part II," *Journal of the Audio Engineering Society*, vol. 19, no. 7, pp. 542–550, 1971.

[13] A. Chaigne and A. Askenfelt, "Numerical simulations of struck strings. I. a physical model for a struck string using finite difference methods," *JASA*, vol. 95, no. 2, pp. 1112–1118, 1994.

[14] S. Bilbao, *Numerical Sound Synthesis*, J. Wiley & Sons, 2009.

[15] S. Bilbao, R. Hamilton, B and. Harrison, and A. Torin, "Finite-difference schemes in musical acoustics: A tutorial," in *Springer Handbook of Systematic Musicology*, chapter 19, pp. 349–384. Springer, 2018.

[16] E. Maestre, C. Spa, and J.O. Smith, "A bowed string physical model including finite-width thermal friction and hair dynamics," *Proceedings ICMC|SMC|2014*, pp. 1305–1311, 2014.

[17] C. G. M. Desvages, *Physical Modelling of the Bowed String and Applications to Sound Synthesis*, Ph.D. thesis, The University of Edinburgh, 2017.

[18] C. Desvages and S. Bilbao, "Two-polarisation finite difference model of bowed strings with nonlinear contact and friction forces," *Proceedings of the International Conference on Digital Audio Effects*, 2015.

[19] J.H. Smith and J. Woodhouse, "The tribology of rosin," *Journal of the Mechanics and Physics of Solids*, vol. 48, pp. 1633–1681, 2000.

[20] S. Willemsen, N. Andersson, S. Serafin, and S. Bilbao, "Real-time control of large-scale modular physical models using the sensel morph," *Proc. of the 16th Sound and Music Computing Conference*, pp. 275–280, 2019.

[21] Sensel Inc., "Sensel Morph," Available at https://sensel.com/, accessed April 01, 2019.

[22] JUCE ROLI, "JUCE," Available at https://juce.com/, accessed April 04, 2019.

[23] S. Willemsen, "Elasto-Plastic Bow Model acting on a Finite-Difference Stiff String," Available at https://www.youtube.com/watch?v=-5bDebCW1Qg, accessed April 06, 2019.

[24] K. J. Åström and C. Canudas de Wit, "Revisiting the lugre friction model," *IEEE Control Systems Magazine*, vol. 28, no. 6, pp. 101–114, 2008.

[25] C. Canudas de Wit, H. Olsson, K. J. Åström, and P. Lischinsky, "Dynamic friction models and control design," *Proceedings of the 1993 American Control Conference*, pp. 1920–1926, 1993.

[26] C. Canudas de Wit, H. Olsson, K. J. Åström, and P. Lischinsky, "A new model for control of systems with friction," *IEEE Transactions on Automatic Control*, vol. 40, no. 3, pp. 419–425, 1995.

KEYTAR: MELODIC CONTROL OF MULTISENSORY FEEDBACK FROM VIRTUAL STRINGS

Federico Fontana *

HCI Lab
University of Udine
Udine, Italy
federico.fontana@uniud.it

Andrea Passalenti †

HCI Lab
University of Udine
Udine, Italy
passalenti.andrea@spes.uniud.it

Stefania Serafin ‡

Multisensory Experience Lab
Aalborg University Copenhagen
Copenhagen, Denmark
sts@create.aau.dk

Razvan Paisa §

Multisensory Experience Lab
Aalborg University Copenhagen
Copenhagen, Denmark
rpa@create.aau.dk

ABSTRACT

A multisensory virtual environment has been designed, aiming at recreating a realistic interaction with a set of vibrating strings. Haptic, auditory and visual cues progressively istantiate the environment: force and tactile feedback are provided by a robotic arm reporting for string reaction, string surface properties, and furthermore defining the physical touchpoint in form of a virtual plectrum embodied by the arm stylus. Auditory feedback is instantaneously synthesized as a result of the contacts of this plectrum against the strings, reproducing guitar sounds. A simple visual scenario contextualizes the plectrum in action along with the vibrating strings. Notes and chords are selected using a keyboard controller, in ways that one hand is engaged in the creation of a melody while the other hand plucks virtual strings. Such components have been integrated within the Unity3D simulation environment for game development, and run altogether on a PC. As also declared by a group of users testing a monophonic Keytar prototype with no keyboard control, the most significant contribution to the realism of the strings is given by the haptic feedback, in particular by the textural nuances that the robotic arm synthesizes while reproducing physical attributes of a metal surface. Their opinion, hence, argues in favor of the importance of factors others than auditory feedback for the design of new musical interfaces.

1. INTRODUCTION

Along with the continuous miniaturization and proliferation of digital hardware, the research domain of computer interfaces for musical performance is increasingly taking advantage of the growing market of haptic devices. Traditionally confined to the loudspeaker set, the output channel in these interfaces now often incorporates force and tactile feedback targeting the performer [1] and/or the audience [2]. In spite of the debate around the effects that synthetic tactile cues have on the precision of the execution [3] and in general on the quality of the performance [4], no doubt exists on the impact that haptic feedback has as a conveyor of realism, engagement and acceptability of the interface [5]. This design trend has developed to the point that a specific research line has been recently coined: musical haptics [6].

In parallel to such hardware opportunities, several software architectures have become available. These architectures contain libraries which provide relatively fast access to functionalities that, until few years ago, needed significant labor of researchers interested in programming a new musical interface. Conversely, modern software development environments such as Max/MSP or Unity3D have enough resources in-house for realizing digital instrument prototypes yielding a credible image of the final application. The latter software, in particular, already in its freeware version offers an unprecedented support to existing multimedia devices in the context of a visual or, at the developer's convenience, traditional programming environment. Originally intended for computer game design, Unity3D is being embraced by an increasing community of programmers with interests in virtual and augmented reality, as well as cultural and artistic applications on multimodal technologies.

Among the numerous *assets* Unity3D makes available, basic objects can be found for defining elastic strings which dynamically respond to colliding bodies and, at that moment, play an audio sample or launch a sound synthesis algorithm. Such a core scenario can receive messages by external devices including Musical Instrument Digital Interface (MIDI)-based controllers like a keyboard, and haptic devices like e.g. a robotic arm.

Using these components we have started a project, now called Keytar, aiming at understanding the perceptual importance of haptic, auditory and visual feedback during a point-wise interaction with a virtual string set. This idea is not new: multisensory environments with visual, auditory and haptic feedback have for example been developed for several decades in the context of the Cordis-Anima framework [7]. Our approach, however, aims at an efficient yet accurate simulation in a widely accessible virtual reality environment. We have reproduced a standard and a bass guitar employing different string sets, and then compared sample-based rather than synthesized sounds under different visual viewpoints

* Concept, writing, lab facilities
† Design, development, implementation, writing
‡ Writing, funding dissemination
§ Implementation

of the instruments. Later we have integrated melodic control via MIDI keyboard. This paper reports the current status of the Keytar project, documenting what has been done and what is ongoing based on the lessons learnt so far: Sec. 2 specifies the hardware in use; Sec. 3 describes the software component realizing communication with the devices, haptic workspace, sound synthesis and visualization; Sec. 4 explains how Keytar was first calibrated and then preliminary tested; Sec. 5 concludes the paper.

2. PHYSICAL INTERFACE

The Keytar interface is shown in Fig. 1. While users pluck a string through the robotic arm using their dominant hand, they simultaneously feel its resistance and textural properties. This way, the plucking force results from a natural perception-and-action process involving users and the string. In the meantime, with the other hand they select notes and/or chords through the keyboard controller. The setup is completed by a screen displaying the action of the virtual plectrum against the vibrating strings, as well as by the audio.

Figure 1: *Keytar: desktop configuration.*

Besides the presence of a monitor and a loudspeaker or headphone set, the most interesting physical component of the output interface is certainly the robotic arm (Fig. 2). A SensAble (now 3D Systems) Phantom Omni was available for this project. The limited force its motors exert on the mechanic arm is compensated by extremely silent and low-latency reactions, making this device still an ideal candidate for virtualizing point-wise manipulations that do not need to render stiff materials. Particularly in the case of elastic string simulation, the Phantom Omni revealed ideal characteristics: since only rotation and lift are actuated, the arm renders elastic reactions while leaving users free to select the angular position of the plectrum at their own taste through the stylus. Even more surprisingly, while rendering the textural properties of wound metal strings its motors produced sounds that are very similar to the scraping effect a plectrum creates when it is slid over a bass guitar string. This acoustic by-product coming from the robotic arm added much realism to the multi-sensory scenario, completing the audio-tactile feedback with nuances whose acoustic component would be otherwise particularly difficult to synthesize and keep synchronized with the haptic channel.

Figure 2: *Phantom Omni robotic arm.*

Concerning the input interface, the robotic arm is once more interesting for its ability to instantaneously detect the stylus position and force exerted on it when its motors are resisting to the arm movement. Position was used to detect when the plectrum collided against the strings. Aside of the robotic arm, a Novation ReMOTE 25SL controller was set to acquire standard MIDI notes across its two-octave keyboard. Finally, two buttons located on the arm stylus at immediate reach of the index finger (both visible in Fig. 2) allowed for selecting major (blue button pressed) or minor (grey button pressed) chords instead of single notes through the keyboard. In other words, users could play chords or alternatively solos respectively by pressing either button or by leaving both buttons depressed. Since there is no obvious mapping from piano to guitar keys, rules were adopted to create a link between the two note sets which will be discussed at the end of Sec. 3.2.

3. SOFTWARE ARCHITECTURE

Due to the efficiency of the applications built for many computer architectures, Unity3D is gaining increasing popularity also among musical interface designers. Moreover, this programming environment adds versatility to the development of a software project thanks to its support to a number of peripherals for virtual reality. Finally, software development is made easier by a powerful graphic interface, making it possible to visually program several C# methods and classes that would otherwise need to be traditionally coded through textual input of the instructions. The same interface allows also to preview the application at the point where the project development is.

For the purposes of the Keytar project, Unity3D supported i) communication with the robotic arm and MIDI controller, ii) agile virtual realization of vibrating strings, iii) implementation of collision detection between such strings and a virtual plectrum, iv) access to an algorithm capable of interactively synthesizing string tones, and finally v) relatively easy development of the graphical interface. In the following we browse such components, emphasizing aspects whose knowledge may help researchers in computer music interfaces reproduce the prototype, or take inspiration from Keytar while designing their own instrument.

3.1. Communication with peripherals

Bi-directional communication with the Phantom Omni is made possible by Unity3D's Haptic Plugin for Geomagic OpenHaptics,

developed for Windows and (in beta version) for Linux by the Glasgow School of Art's Digital Design Studio. Through this asset it was possible to get the position of the Phantom's arm and simultaneously send control messages to its motors, hence defining the instantaneous behavior of the robotic device. In particular, Haptic Plugin manages force through higher level parameters: *stiffness*, *damping*, *static friction*, *dynamic friction*, *tangential stiffness*, *tangential damping*, *punctured static friction*, *punctured dynamic friction*, and *mass*. Expansions of this palette appearing in the newer versions of Unity3D have outmatched such parameters, in practice forbidding the Phantom Omni to interact with dynamic (i.e., moving) virtual objects.

Concerning the communication with the keyboard, Keijiro Takahashi's MIDI Jack is an open project[1] enabling Unity3D to acquire and interpret some types of MIDI events coming from a controller that sends messages in this protocol. In particular, it features automatic recognition of input MIDI devices and returns non-zero velocity values of notes when a key is pressed. This software module immediately establishes a communication between the controller and the note selector running in Keytar (see Fig. 3). Though,

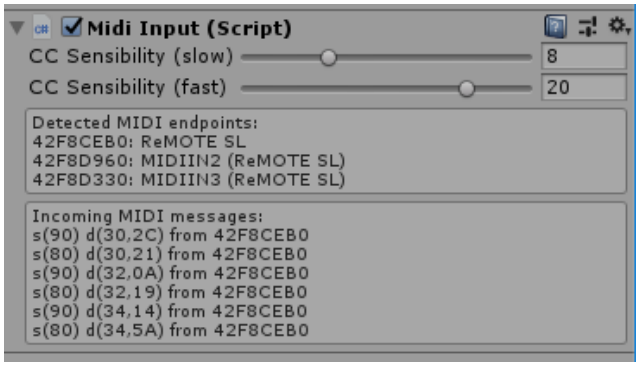


Figure 3: *MIDI input inspector in Unity3D.*

if the application is ported on another PC then care must be taken in keeping the MIDI channel that was chosen during building. Future versions of Keytar will accept this value as an initial argument.

3.2. Strings, plectrum, and collision detection

Each string was created using *Cylinder* objects. Such objects are instantiated by a primitive class of Unity3D once their length, diameter and position in the virtual space are set. The plectrum was instead obtained by putting one passive and one *active* object together: the former gave visual appearance by realizing the traditional shape, while the latter was responsible of the collision against the strings and was instantiated as a *Sphere* object having minimum (i.e. almost point-wise) diameter.

The little sphere was placed on one edge of the plectrum and then made invisible by disabling its display. Like we did also for each string, the sphere activity was enabled by interfacing it with the classes *RigidBody* and *Collider*, containing methods for the real-time detection and management of collisions. Since users often move the plectrum beyond the plane containing the strings, the interaction was made more robust by positioning an active surface just beyond this plane. This object, invisible to the user and having the same role as a fretboard, cannot be crossed by the sphere and hence avoids occasional backward collisions entangling the plectrum behind the strings.

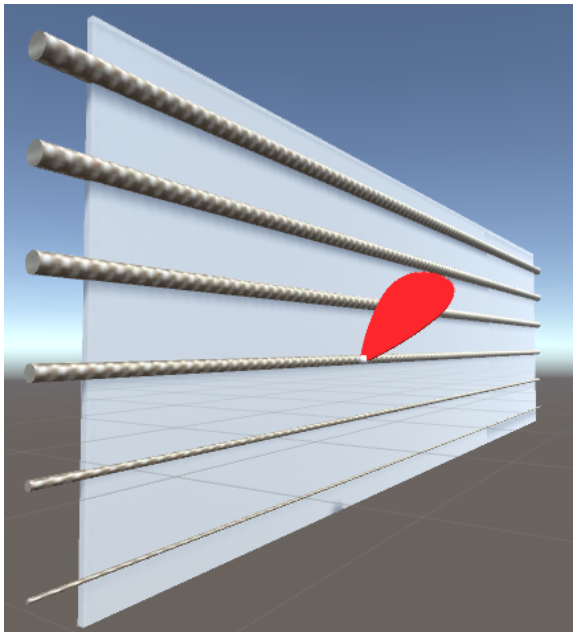


Figure 4: *Haptic workspace in Keytar. The colliding sphere is visible as a white tip on the edge of the plectrum.*

Active objects become part of the haptic workspace, accessible to the robotic arm, if they are labeled as *Touchable*. In this way, their position and shape define areas of the virtual scenario that become inaccessible to the tip of the arm stylus, as obviously no more than one solid object at a time can occupy a single volume of the workspace. In parallel such object surfaces inherit the haptic parameters listed in Sec. 3.1, which are used at runtime while computing the collisions occurring between touchable objects. Fig. 4 shows the final appearance of the haptic workspace. Initially we had labeled the entirely visible plectrum as touchable, i.e. with no distinction between active and passive region. Unfortunately, distributing contact areas across objects having irregular shapes can encumber the computation of the haptic workspace with occasional crashes of the application especially when running on slower machines. This issue was solved in Keytar by tying a tiny touchable sphere to the plectrum.

Clearly, this simplification leads to visuo-haptic mismatching as soon as a passive region of the plectrum intersects with the string. The net effect was that the plectrum often went through strings with no apparent contact. A more subtle visuo-haptic mismatch manifests when the active edge of the plectrum touches, but does not penetrate a string enough: in this case the active objects interact each other as one could clearly see also from the screen, but no collision event is triggered inside the haptic workspace. The latter problem was solved by wrapping each string with an invisible, inactive meanwhile touchable shield, which was set to be as thick as the diameter of the sphere active in the plectrum—see Fig. 5. This shield in fact paired the contact point positions set by the *Touchable* and *Collider* methods. Such string wrappers also attenuate the former problem, as their inclusion made it more difficult to dip the plectrum into the string without production of reactive force from the robotic arm due to collision of the sphere

[1] `https://github.com/keijiro/MidiJack`

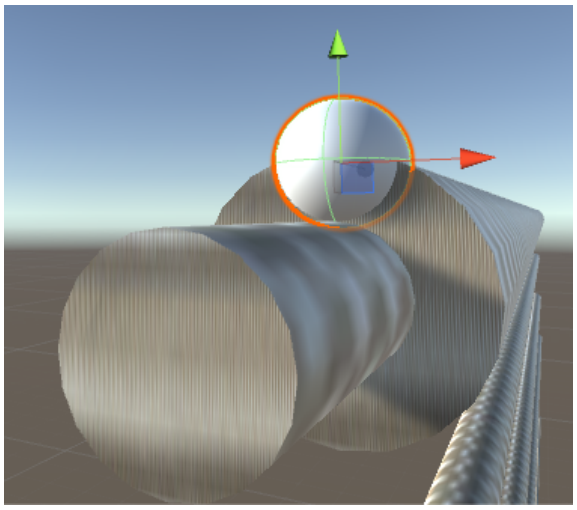


Figure 5: *Mismatching positions of visual and haptic collision point. Solution using string wrappers.*

against the wrapper.

A strategy for estimating each string velocity after a collision was needed. To this regard, Haptic Plugin gives runtime access to the contact point coordinates immediately before (b) and after (a) every collision. We decided to estimate velocity as the distance between such two positions divided by the haptic frame rate, readable from the static property *Time.deltaTime*:

$$\text{velocity} = \frac{\sqrt{(x_a - x_b)^2 + (y_a - y_b)^2 + (z_a - z_b)^2}}{Time.deltaTime}.$$

Velocity was then used to determine the amplitude of vibration in each string.

3.3. Sample-based, interactive, and physical sounds

In Unity3D, sound source and listening point are respectively defined by *AudioSource* and *AudioListener* objects. In the case of a guitar simulation the Keytar workspace contained six sound sources, one for each string. In parallel, a default audio listener was associated to the standard *Camera* object, corresponding to the viewpoint in the virtual scenario. On their way from the sources to the listening point, sounds can be additionally routed in an *AudioMixer* object. On top of mixing down such sources to pick-up points for the audio card, the *AudioMixer* class allows for interconnection of, and interaction at runtime with several sound processing methods including filters, reverberators, compressors, and further digital audio effects. Alone, these methods turn Unity3D in an effective digital audio workstation.

We kept the *AudioMixer* object to the simplest, by just picking up sounds from the strings and mixing them down before reproduction from the virtual listening point. The corresponding console, shown in Fig. 6, was sufficient for reproducing a bank of guitar samples we use in the initial prototype. This prototype had no note selection, and for this reason just six tones corresponding to the freely vibrating strings were recorded to form this bank, from a guitar playing at several dynamic levels.

Later, when the note selection was introduced, we switched to interactive synthesis by including the well-known Karplus-Strong

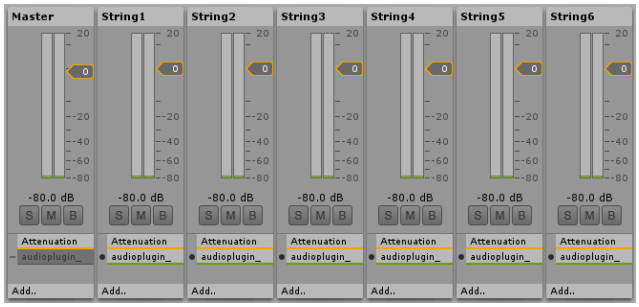


Figure 6: *AudioMixer object in Keytar.*

algorithm[2] in the software architecture [8]. Once imported, this

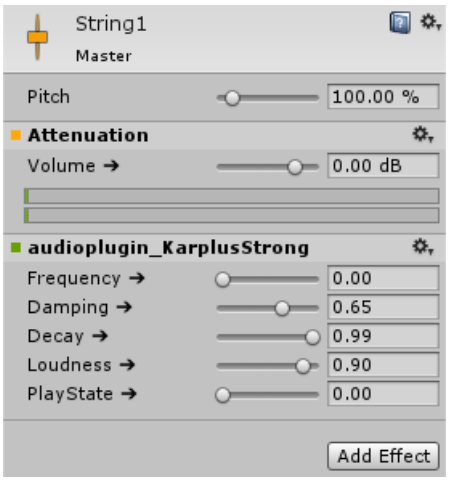


Figure 7: *Karplus-Strong control panel in Unity3D.*

synthesizer exposes the control panel as in Fig. 7, containing five parameters. Except for damping, all such parameters can be set at runtime to values between zero and one. For this reason it was necessary to program a method mapping every tone fundamental frequency into this interval. Apart from the freedom of choice of the notes and their timbre using Unity3D's Karplus-Strong—see in particular the simultaneous presence of *damping* and *decay*, alternatively absent in the traditional algorithm instead—the net result of the switch from sample-based to interactive synthesis was a definitely more reactive instrument with empoverished sound quality. Some preliminary sound engineering using standard guitar effects available in Unity3D, however, suggests that it should not be difficult to synthesize acceptable virtual electric guitar sounds with Keytar.

As anticipated in Sec. 2, Keytar features three different playing modes that can be selected through the stylus buttons: solo, harmonic major and harmonic minor. Any switch of these buttons calls a script that activates the corresponding playing mode. In solo mode, the related script simply runs an instance of the Karplus-Strong algorithm at the time when a collision happens, by tuning the synthesizer to the tone that has been set by the controller key—not below the lowest note the excited string can produce, however. The harmonic modes instead launch several instances of the same

[2] `https://github.com/mrmikejones/KarplusStrong`

algorithm, by building major and minor chords based on the last key pression on the controller. This event dictates the fundamental note of corresponding chord. All chords were standard forms based on the chord voicing music theory. They were formed only by tones belonging to the fundamental tonality.

Last, but not least, the acoustic feedback includes real sounds that are a by-product of the effort the robotic arm makes while reproducing the plectrum-string contact. As it can be seen from Fig. 3, each string results by alternating cylinders of two different diameters. The resulting surface geometry hence reproduces a texture assimilable to that of a wound string coating, as it contains regular discontinuities. While following the longitudinal motion of a plectrum scraping along such a string, the motors simulate a motion similar to friction, producing not only haptic, but also acoustic cues of convincing quality, with substantial improvement of the overall realism of the interaction.

3.4. Visual display

In a scenario populated by elements belonging to a typical guitar playing set, the plectrum and the strings are the only animated objects visible from the standard *camera* viewpoint. Plectrum movements directly reproduce the shifts and rotations of the arm stylus, proportionally to the workspace size. Strings visually vibrate thanks to the *Animator* interface, whose methods implement Unity 3D's Mecanim Animation System. This system concatenates basic *Animation* objects along time, together realizing the visual flow visible from the *Camera* object.

In Keytar each *animation* object models an oscillatory transversal shift having specific frequency and decaying amplitude along time. By concatenating few such objects, each string vibrates up and down with decreasing amplitude. This model demands low computational effort in ways that each string position can be refreshed every 10 milliseconds, that is, about 1.6 times the refresh rate of the video. This ratio creates an effect of occasional semitransparency of the vibrating string, which is beneficial for the realism—see Fig. 8.

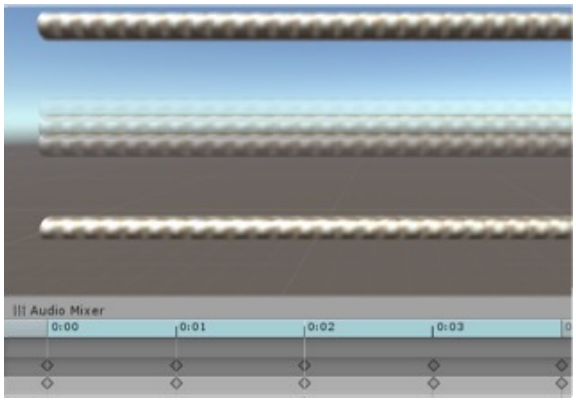


Figure 8: *Semitransparency of one animated string in Keytar.*

In practice, each collision triggers a script that determines the initial vibration amplitude. Then, vibrations are rendered by an *Animator* method that destroys a visual string while cloning it into a shifted instance. This workflow continues until the original string returns at rest or, conversely, a new collision happens against the initial string position. In other words, the clones were not interfaced to the *Collider* class nor did they inherit any *Touchable* property, as the definition of a haptic workspace evolving dynamically at such a fast refresh rate would suffer from the instability mentioned in 3.1. This limit defers any realization of vibratory feedback in Keytar to a version capable of driving vibrotactile devices not using data about the interaction point, but rather higher level information about amplitude and pitch of the string vibrating in contact with the plectrum.

In spite of its efficiency and versatile management of collision events, an accurate visual rendering of the string would require to realize a rectangular surface that shrinks, and becomes progressively less semitrasparent across time until reducing to a solid string at rest position. Finally, the virtual string did not model rigid edges. This approximation results in strings moving up and down as stiff bars would do, with apparent artefacts whose removal will eventually require a different design of the visual string.

4. CALIBRATION AND PRELIMINARY TESTING

The strings' physical parameters were tuned by two students, who regularly perform with their guitar and bass in a pop band. Then, Keytar was preliminary tested during two experiments that have been documented in previous papers [9, 10] when Keytar had no polyphonic keyboard control yet. Both such experiments focused on the realism of the multi-sensory interaction with the strings. Here we report limitedly to the answers participants gave concerning the audio-haptic interaction.

In experiment 1, seven participants with different music experience and knowledge of the technology were asked first to pluck a guitar and a bass guitar using a plectrum. Then, they tested Keytar and answered a questionnaire. Regarding the realism of audio-haptic feedback, participants were asked to rate *roughness* of each string surface, *longitudinal motion friction* while sliding the plectrum, *sound friction* during the same sliding, and finally *string resistance*. While the first three attributes were essentially tactile and depended on the vibratory activity of the arm motors, the fourth attribute was kinaesthetic and linked to the force feedback generated by the robotic arm.

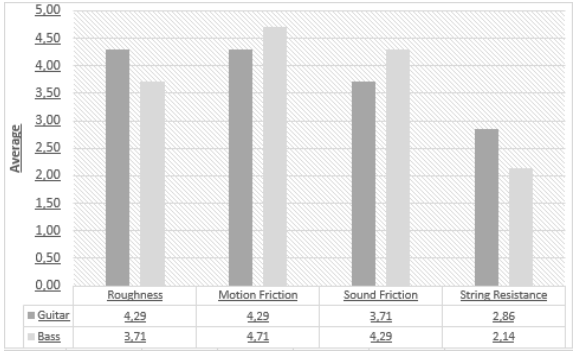


Figure 9: *Experiment 1: Results from haptic evaluation.*

Fig. 9 shows good scores for the first three attributes. Conversely, *string resistance* scored lower. The reason for this dfference might be due to the need to grasp a stylus instead of a plectrum during playing.

Experiment 2 aimed at evaluating the relative importance of

force and vibrotactile feedback for the creation of a realistic experience [10]. For this reason, the arm stylus was modified for the occasion by applying a 3D printed plectrum on its tip. This physical plectrum was embedded with a Haptuator Mark II vibrotactile actuator by Tactile Labs. Each time a collision occurred, the vibrotactile actuator rendered an impact generated using the Sound Design Toolkit for Max/MSP [11]. Twenty-nine subjects were first asked to pluck a real string, then they were exposed to four conditions: no haptic feedback (N), vibrotactile feedback (V, made with the Haptuator), force feedback (F, made with the Phantom Omni), and combination of force and vibrotactile feedback (FV). Finally, subjects reported the perceived realism through a questionnaire. Results are reported, among others, in Fig. 10. Both show that

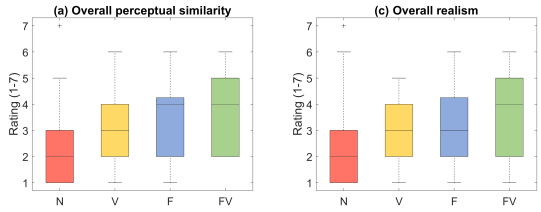


Figure 10: *Experiment 2: Overall perceptual similarity and realism.*

the combined force and vibrotactile feedback was rated as significantly more realistic. This conclusion suggests that expanding the feedback of Keytar with vibrations in both its keyboard and stylus controllers should further increase its realism.

5. CONCLUSIONS

Keytar prototypes a point-wise physical interaction with a set of virtual strings. With relatively low software programming effort, it can model essentially any compact stringed instruments that is played using a plectrum. Ongoing work is oriented to reproduce interactions capable of taking full advantage from alternative prototypes. The lap steel guitar, for instance, could be realized by displaying the strings horizontally below the robotic arm; in parallel the keyboard, if not substituted by a continuous controller, could be used to set the final chord where the steel bar is sliding to. In the longer term, the keyboard (or any other controller) and the plectrum could convey string vibrations through proper actuation, as well as the plectrum position could be used to modulate the spectral content of tones as it happens with real guitars.

6. ACKNOWLEDGMENTS

This work is supported by NordForsk's Nordic University Hub Nordic Sound and Music Computing Network NordicSMC, project number 86892.

7. REFERENCES

[1] Mark T. Marshall and Marcelo M. Wanderley, "Vibrotactile feedback in digital musical instruments," in *Proceedings of the 2006 conference on New interfaces for musical expression*, Paris, France, Jun. 4-8 2006, pp. 226–229.

[2] Stahl Stenslie, Andreas Göransson, Tony Olsson, and David Cuartielles, "Stitchies: towards telehaptic performativity," in *Tei 2014–8th International Conference on Tangible, Embedded and Embodied Interaction, Proceedings*, Stockholm, Sweden, Mar. 18–21 2014, ACM, pp. 327–329.

[3] Mark T. Marshall and Marcelo M. Wanderley, "Examining the effects of embedded vibrotactile feedback on the feel of a digital musical instrument," in *Proc. Int. Conf. on New Interfaces for Musical Expression (NIME)*, Oslo, Norway, May 30–June 1 2011, pp. 399–404.

[4] Joseph Rovan and Vincent Hayward, "Typology of tactile sounds and their synthesis in gesture-driven computer music performance," *Trends in gestural control of music*, pp. 297–320, 2000.

[5] Stephen A. Brewster, "Impact of haptic 'touching' technology on cultural applications," in *Digital Applications for Cultural Heritage Institutions*, J. Hemsley, V. Cappellini, and G. Stanke, Eds., pp. 273–284. Ashgate, 2005.

[6] Stefano Papetti and Charalampos Saitis, Eds., *Musical Haptics*, Springer, 2018.

[7] James Leonard and Claude Cadoz, "Physical modelling concepts for a collection of multisensory virtual musical instruments," in *New Interfaces for Musical Expression 2015*, Baton Rouge, Louisiana, USA, May 31–Jun 3 2015, pp. 150–155.

[8] Kevin Karplus and Alex Strong, "Digital synthesis of plucked-string and drum timbres," *Computer Music Journal*, vol. 7, no. 2, pp. 43–55, 1983.

[9] Andrea Passalenti and Federico Fontana, "Haptic interaction with guitar and bass virtual strings," in *Proceedings of the 15th Sound and Music Computing Conference (SMC 2018)*, Limassol, Cyprus, July 4–7, 2018, pp. 427–432.

[10] Andrea Passalenti, Razvan Paisa, Niels Ch. Nilsson, Nikolaj S. Andersson, Federico Fontana, Rolf Nordahl, and Stefania Serafin, "No strings attached: Force and vibrotactile feedback in a virtual guitar simulation," in *Proceedings of the 16th Sound and Music Computing Conference (SMC 2019)*, Malaga, Spain, May 28–31 2019, pp. 210–216.

[11] Stefano Delle Monache, Pietro Polotti, and Davide Rocchesso, "A toolkit for explorations in sonic interaction design," in *Proceedings of the 5th Audio Mostly Conference: A Conference on Interaction with Sound*, New York, NY, USA, Sep. 15–17 2010, AM '10, pp. 1:1–1:7, ACM.

EXPLORING AUDIO IMMERSION USING USER-GENERATED RECORDINGS

Daniel Gomes, João Magalhães, Sofia Cavaco *

NOVA LINCS, Departamento de Informática
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa
2829-516 Caparica, Portugal
ddl.gomes@campus.fct.unl.pt, {jmag,scavaco}@fct.unl.pt

ABSTRACT

The abundance and ever growing expansion of user-generated content defines a paradigm in multimedia consumption. While user immersion through audio has gained relevance in the later years due to the growing interest in virtual and augmented reality immersion technologies, the existent user-generated content visualization techniques are still not making use of immersion technologies.

Here we propose a new technique to visualize multimedia content that provides immersion through the audio. While our technique focus on audio immersion, we also propose to combine it with a video interface that aims at providing an enveloping visual experience to end-users. The technique combines professional audio recordings with user-generated audio recordings of the same event. Immersion is granted through the spatialization of the user generated audio content with head related transfer functions.

1. INTRODUCTION

Considering a society in transformation and transition to immersive content, such as video games and virtual reality, it is natural to extend the immersion to other content such as user-generated content (UGC). To answer this tendency we propose a technique that uses the audio from UGC to achieve immersion through the audio. By immersion we mean spatial presence, as defined by Madigan [1].

Audio immersion can serve multiple purposes ranging from different areas of interest. As an example, it can be used for education, training and entertainment of blind and visually impaired (BVI) people. *Navmol* is an application that uses audio immersion to help BVI chemistry students to interpret and edit the representation of molecular structures [2]. Once a reference atom is selected, the application uses a speech synthesizer to inform the user about the neighboring atoms. The speech signal is spatialized using head related transfer functions (HRTFs). In this way, users wearing headphones will hear the atoms' descriptions coming from different angles in space. Similarly, immersive audio can be used to train orientation and mobility skills for BVI people. Cavaco, Simões and Silva propose a virtual environment for training spatial perception in azimuth, elevation and distance [3]. The audio is spatialized with HRTFs. Immersive audio has also been used for entertainment of BVI people. *Demor* is a shooting game based in 3D spatialized audio that aims at providing entertainment to both BVI and sighted players [4]. Despite the entertainment component, Cohen *et. al* also attempt to improve BVI people emancipation in the sighted world by training mobility and spatial orientation. The game requires players to localize sounds in space that represent targets to shoot before they reach the player, who is equipped with a laptop, a GPS receiver, a head tracker, headphones and a modified joystick, all attached to a backpack. The kit continuously tracks player location and orientation and updates the sound accordingly.

Immersive audio can also be applied to information delivery. Montan introduced a low cost audio augmented reality prototype for information retrieval [5]. In the study, the author created a headset with a head movement tracker for a use case of museum interactive audio-guides. As the users rotated their heads, the tracker registered head orientation and the system rendered the audio properly. The rendering is performed in real time using HRTFs according to the relative position and orientation of the listener and the emitters. In another study, Guerreiro proposed to take advantage of the cocktail party effect to convey information about digital documents to BVI people using only audio [6]. Instead of using a common text-to-speech system that converts textual information into a speech signal that contains a single voice, the author proposed to use various voices simultaneously at different angles. HRTFs were used to change the speech synthesizer signal.

Here we propose an audio immersion technique for UGC. The technique combines several UGC recordings of the same event, modified with HRTFs, in order to immerse audibly the user. Such recordings are distributed in space and are reproduced from different angles. While the technique focus on audio immersion, we also propose to combine it with a video interface that aims at providing an enveloping visual experience (more details in section 2).

In order to demonstrate and validate the proposed technique, we built a prototype for mobile devices that includes a video player (section 3). The proposed tool is designed to play concert videos, although it it not limited to this single use case scenario. We used this prototype in a user test that validates the proposed technique. The test focuses on three attributes: immersion, sense of space and directional quality. Section 4 describes the tasks performed in more detail, section 4.1 describes the data used in the user test, and section 5 discusses the user test's results.

Since the scope of the presented work required diverse and abundant UGC, musical concerts were chosen as a good use case scenario to draw useful conclusions. The database chosen is composed of multiple events (*i.e.*, audio recordings of several concerts), which in their turn have several recordings. User-generated

* This work was supported by the H2020 ICT project COGNITUS with the grant agreement No 687605 and the Portuguese Foundation for Science and Technology under project NOVA-LINCS (PEest/UID/CEC/04516/2019).

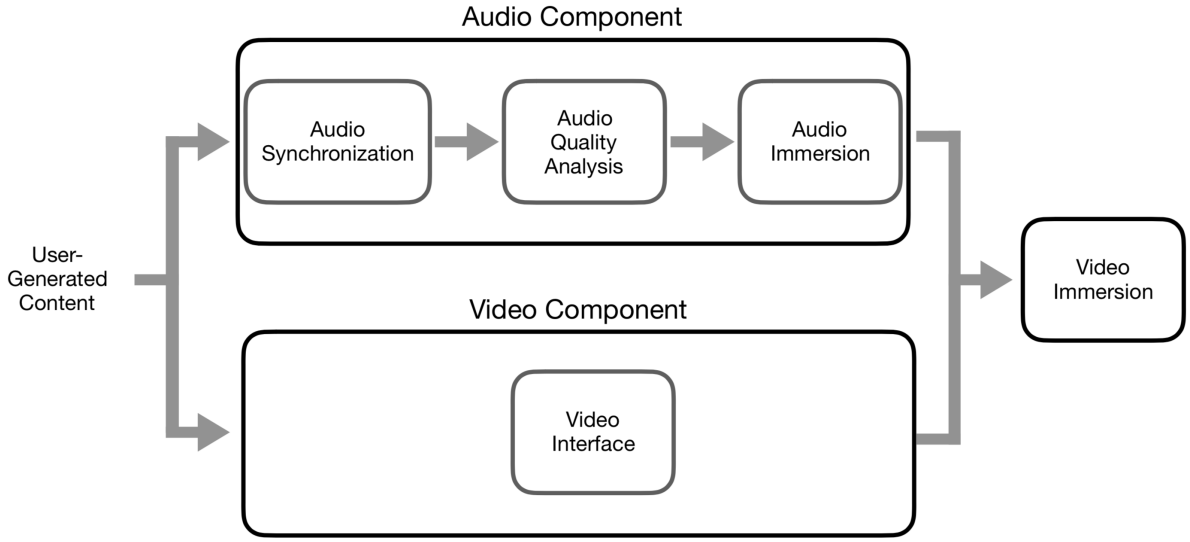


Figure 1: The proposed UGC audio immersion technique scheme.

recordings raise some challenges, in particular noise and a time sparse nature. Different devices have distinct recording qualities which impact in the level of noise captured. Additionally, recordings capture different parts of the event, considering possible overlaps. Thus it is required to deal with these UGC challenges. We propose to categorize recordings of the chosen data set by subjective level of noise and select those with best quality to be played. As explained in section 2, we propose to use a quality analysis and data synchronization technique based on audio fingerprints.

2. THE UGC AUDIO IMMERSION TECHNIQUE

The proposed immersion technique can be seen as the interaction between two main components: the audio and the video components (figure 1). The audio component immerses the user audibly by presenting multiple audio sources distributed in a multidimensional space. The video component consists of a user interface that aims at providing an enveloping visual experience to the end user. Note that we do not aim to achieve video immersion in this prototype.

Since our main focus in this paper is the audio component, in this section we focus only on this component. Nonetheless, we developed two different simple approaches for the video component for validation purposes. These approaches are discussed in section 3.1.

To achieve audio immersion using UGC, we propose to combine several recordings from the same event. Our technique consists of changing the original recordings such that they are reproduced from different angles, and when played together they provide a combined audio signal that can give the perception of immersion.

It is important to highlight that UGC have diverse audio qualities inherent to the different characteristic of the devices used to capture the audio. Also different recordings of the same event (for instance, the same music in a concert), can capture different portions of the event with possible overlapping sections. Thus, even before we process the audio signals to provide a sense of immersion, there are other steps we must perform, namely audio synchronization and analysis of the signals' quality.

2.1. Audio synchronization

Given a data set of recordings from the same event, it is important to understand the chronological order of the events and identify the recordings' overlapping sections. Following our previous work on organization of user generated audio content (UGAC), we propose to use audio fingerprints to create a timeline with the event's recordings [7, 8].

The resistance to noise of fingerprinting techniques is particularly relevant when dealing with low quality music recordings. This characteristic is suitable for our proposed immersion technique because it enables synchronizing samples with quite different quality and noise levels, which is a characteristic of UGAC.

Mordido *et al.* use audio fingerprints to identify common sections between the audio recordings [8]. This technique identifies the overall offsets of all recordings of the same event, as well as the duration of each recording. This gives us information on which recordings cover different portions of the timeline. Thus, we can organize an event with *timeline segments*, such that segments coincide with the time interval of overlapping recordings. The final result is a timeline with all the recordings aligned, such that overlapping sections of different signals are played simultaneously. Figure 2 shows the timeline for a set of five recordings from the same event. The timeline is organized into timeline segments T_1 to T_7.

2.2. Audio quality analysis

Once the signals are chronologically organized, we need to choose which signals to use. Given a set of recordings from the same event, we will choose only a few. More specifically, for each timeline segment, T_i, we choose n_i recordings (we choose a low number, such as three or four at most). In order to choose the n_i recordings for each timeline segment, we start by measuring the quality

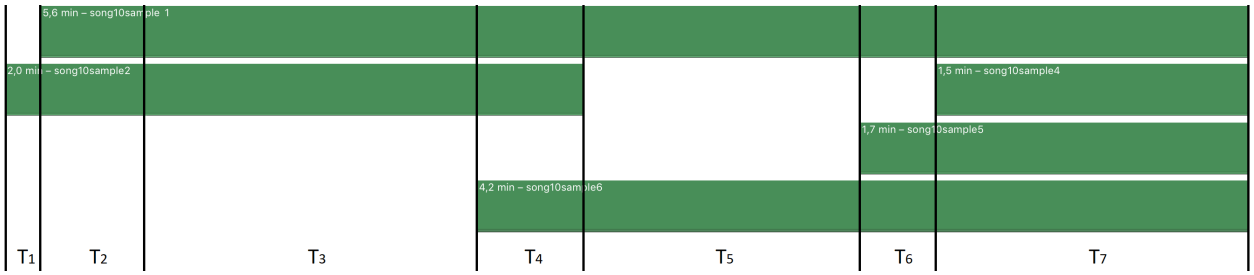


Figure 2: Timeline and segments, T_1 to T_7, for a set of recordings (in green) from the same event.

of all the data in the set, and choose the recordings with the higher quality.

For the recordings audio quality analysis, we propose to use our previous work with audio fingerprints for quality inference of UGAC [7, 8]. Like with the audio synchronization technique explained above, this method uses audio fingerprints to detect overlapping segments between different audio recordings. In addition it assumes that the recordings with more matching landmarks have higher quality. (A landmark is a pair of two frequency peaks and contains information about the frequency for the peaks, the time stamp of the first peak, and the time offset between the two peaks.)

Alternatively, some interesting recordings can also be chosen manually. For instance, let us assume there is a recording with voices or clapping in the audience that we want to use, but has lower quality. While this recording may be ranked as having low quality, it can still be chosen. In fact, we manually chose the recordings used in our prototype because we wanted to have recordings with quite different characteristics.

2.3. Audio immersion

As shown in figure 1, the following step is audio immersion. In this step, we change the original n_i audio signals selected for each timeline segment T_i, such that when heard individually through headphones, they can be perceived as if coming from different directions, and when heard together, they give a sense of space.

Our proposal is to change each original signal s_j with HRTFs. That is, we apply HRTFs to the left and right channel of each signal s_j, such that the modified signal, s'_j is perceived as if coming from angle θ_j. Angle variations are performed in azimuth and elevation. Finally, the timeline built in the audio synchronization phase is used when playing the modified signals. Thus, for each timeline segment T_i, we will play a final signal, S_i, that consists of adding together all selected modified signals from that timeline segment. For instance, let us assume that timeline segment T_i has the overlapping signals s_1, s_2 and s_3. We modify these original signals with HRTFs such that when heard individually, the modified signals s'_1, s'_2 and s'_3 are perceived from directions θ_1, θ_2 and θ_3. Hearing the three signals played simultaneously can give a sense of immersion in which we hear the common music (present in all three recordings) in the surrounding space and we hear the specific individual noises or sounds (like clapping or voices) from each recording as if coming from different directions. Figure 3 illustrates this example.

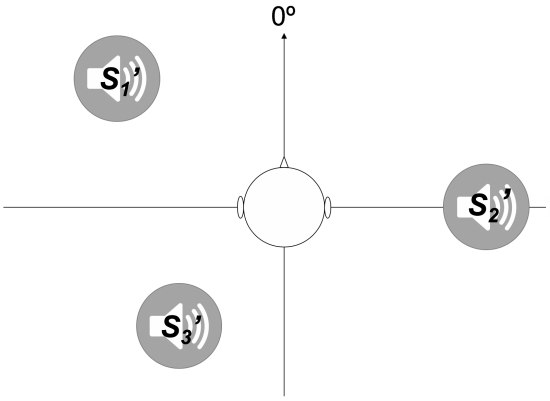


Figure 3: Signals s'_1, s'_2 and s'_3 distributed spatially at directions θ_1, θ_2 and θ_3, respectively. (The users' initial orientation is used to define 0°, which is the direction ahead of the user).

3. THE VIDEO PLAYER PROTOTYPE

In order to validate our UGC audio immersion technique, we developed a prototype that was used in the user tests. This prototype includes an audio component and a video interface.

The prototype's video component was developed with Unity Game Engine. Audio spatialization was granted by Google's Resonance Audio Software Development Kit which uses Sadie HRTF library (University of York SADIE KU100 data set). In the current context, the application was built for iOS devices and requires the use of headphones for audio immersion.

3.1. The video component

The design of our graphical user interface was inspired on the work proposed by Chen, who presented an image-based approach to virtual environment navigation [9]. Chen presented two types of video player: a panoramic and an object player. The first was designed for looking around a space from the inside, while the second was designed to view an object from the outside. Among other features, the panoramic player allows the user to perform continuous panning in the vertical and horizontal directions.

Since, the current state-of-the-art in multimedia content creation by users is from planar smartphone cameras, we developed a graphical user interface that has similarities to the one proposed by Chen. Our user interface does not show the video completely (figure 4). Instead, as shown in the figure, there is a visible region that

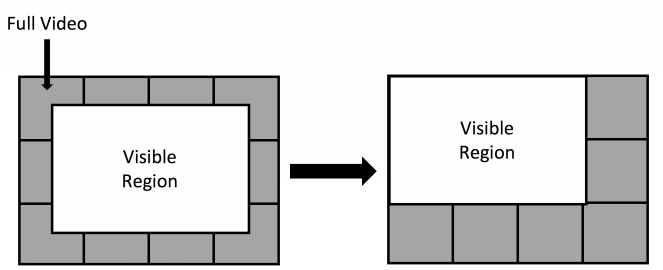


Figure 4: The image display process. The white region is visible, while the dark region is not visible. The user can slide the white rectangle to the dark regions so that the visible region changes. (a) The visible region is in the center of the original video. (b) The visible region changes when the user slides it.

the users can pan continuously through the entire video region (*i.e.*, navigation in horizontal, vertical and diagonal). To implement this visible region we used Unity's orthographic camera projection.

We developed two different approaches for user interaction with the application:

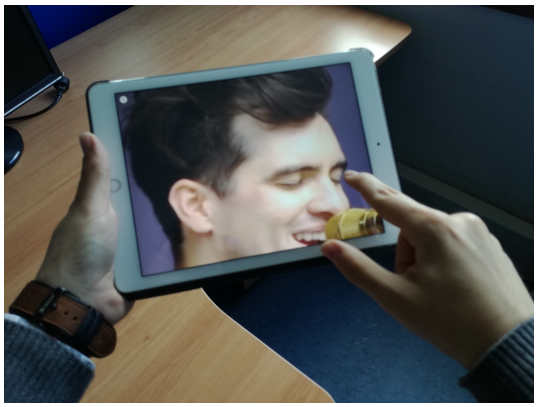
- In the *touch screen approach*, the interaction is processed using the device's touch screen. Users can navigate through the video making use of the device's touch screen to move the visible region around (figure 5.a). For instance, a sliding movement towards the left makes the visible region move to the right.
- The *gyroscope approach* uses the device's gyroscope. Here, users can interact and move the visible region using the gyroscope (figure 5.b). Moving the screen towards the right makes the visible region move to the right. Moving the screen upwards, makes the visible region move upwards.

3.2. The audio component

While developing our prototype, we focused our attention especially in the audio immersion box from figure 1 and the interaction between the video and audio components. The audio recordings used in the prototype were manually selected and synchronized.

As explained above, we use HRTFs to spatialize the original signals s_j, such that each modified signal s'_j is located at a specific direction in space (θ_j) and the final sound S_i for each timeline segment is the combination of the modified signals s'_j. Sliding the visible region into a certain direction, produces changes in each signal s_j, and, as a consequence, the combined signal S_i also changes.

There are two parameters that change for each signal: the intensity and the relative angle to the user. Sliding the visible region into a certain direction, is mapped into head rotations. In other words, when the user slides the visible region (figure 4), the direction of each signal s_j relative to the user changes. This way, when hearing the sounds, the users will perceive changes in the sounds that cause the sensation of having performed head rotations. The changes can be in azimuth and elevation. For instance, when the user moves the visible region to the right, the samples s_j suffer a rotation to the left, as if the user had rotated his/her head clockwise (a change in the azimuth). When the user moves the visible region up, the samples s_j suffer a downward rotation in elevation.


(a)


(b)

Figure 5: The graphical user interface. Navigation in the video using (a) the touch screen approach, and (b) the gyroscope approach.

Let us define $\vec{d}$ as the vector that represents the sliding movement in a 2D space whose x and y-axes are parallel to the screen edges. $\vec{d}$ defines the movement direction and displacement. Let $\vec{d_x}$ be the projection of $\vec{d}$ into the x-axis, and $\vec{d_y}$ be the projection of $\vec{d}$ into the y-axis. For each signal s_j, the rotation in the azimuth is given as a function of $\vec{d_x}$ and the change in elevation is given as a function of $\vec{d_y}$.

The intensity of the signals may also change. We increase the intensity of sounds whose directions θ_j are approximate to the users final orientation, and decrease the intensity of other sounds. Sound intensity changes are described by a linear function of the relative angle to the users' orientation.

4. USER TESTS

In order to evaluate the proposed technique, we run a user test to evaluate spatial sound quality. Pulkki *et al.* propose that spatial sound quality evaluation should consider the evaluation of envelopment, naturalness, sense of space, directional quality and timbre [10]. Among the presented group of attributes, the ones of interest to our study are **directional quality** and **sense of space**. In addition, we introduced the **immersion** factor to be tested.

There were 15 volunteers participating in the study (10 men

on

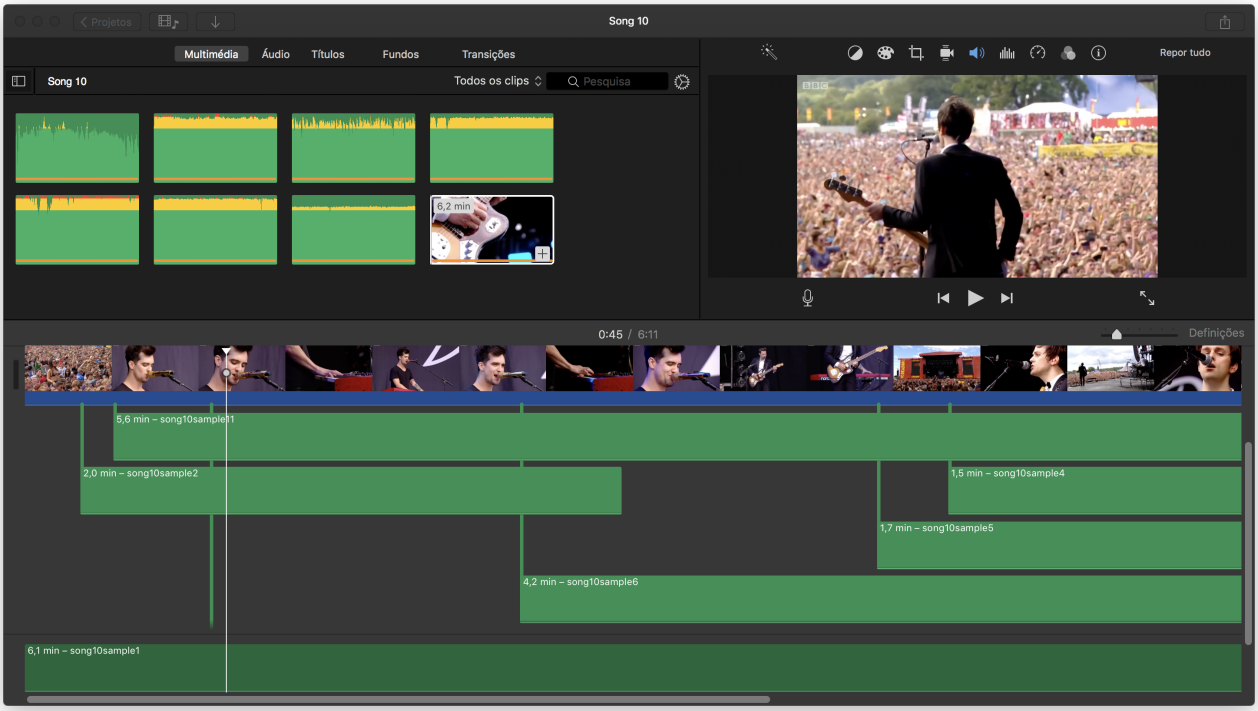


Figure 6: Global timeline of audio and video (developer's view). The green bar for the professional recording (*i.e.*, sample 1) is at the bottom of the image. The UGC recordings are displayed with their respective sample name and length, and starting at their start time t_i. The user's view is presented at the top right corner.

and 5 women) with ages ranging from 19 to 26 years old and all university students. Four of these participants had musical training (either by actively playing an instrument or attending music classes). Only one participant declared having hearing problems and one had a temporary hearing condition. The remaining participants asserted having no hearing problems that could affect their participation.

The user test consisted of five related tasks. For the first four tasks users used a computer while for the last one they used a tablet. The volunteers wore headphones for all tasks and received written instructions and a demonstration for every task. Additionally, at the end of each task, the volunteers were provided with a form in which they were queried about each task. All volunteers were attributed with a numeric reference in order to guarantee data protection.

4.1. Data

The three first tasks tested directional audio quality and used musical instrument sounds generated with iPad's GarageBand application and spatialized according to the technique described in section 2.3. These consisted of:

- A sequence of three sustained piano notes (C, E, G, in the presented order).
- A sequence of three guitar notes (C, G, F, in the presented order).
- A sequence of drum notes from three cymbals (snare, tom high and tom low, at no specific order).

The remaining tasks used recordings from music concerts that were extracted from Mordido's data set [7]. These recordings provide different components under different conditions (*e.g.*, users recording part of a concert in distinct places at different angles to the stage). Our data includes recordings from two events: two musics, each from a different concert. The first music chosen was a cover performed by Panic at the Disco! band of the popular Queen music Bohemian Rhapsody performed at the 2015 Reading Festival. For the second music we chose a live performance of Sing, by Ed Sheeran at the 2014 Glastonbury Festival. For each event, the data set includes a professional recording of the music and two to four user recordings of that music in the same concert.

The professional recordings have higher sound quality and less noise than the remaining samples in the data set. In this group of samples, it is possible to hear the audience singing along, cheering and clapping. Task 4 used the Queen's concert samples, and task 5 used both concerts.

The defined data set is used to produce immersive sound. Professional recordings are combined with UGC using the techniques and timeline explained above. For each event (that is, for each concert), we spatialize the original sound signals such that each modified signal (s'_j) is assigned a different direction (θ_j): (1) The professional recordings are always assigned the same direction: 0°. This direction is determined by the user's initial orientation. (2) The remaining recordings are placed in lateral or rear-user an-

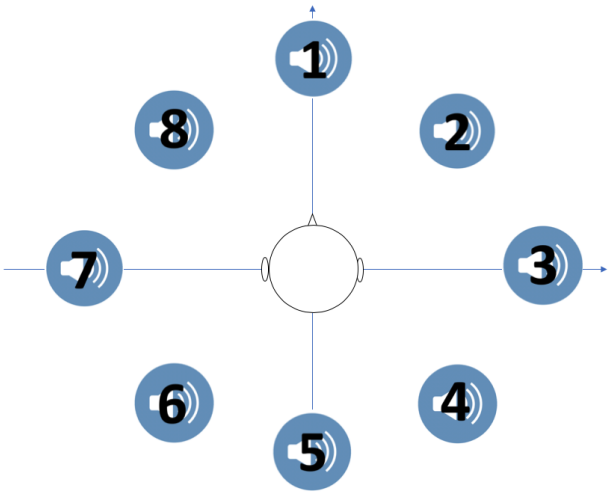


Figure 7: The audio sources' directions used in the user test's tasks 1 to 3.

gles.

For each event's timeline, the professional recording, s_1, starts at $t_1 = 0$ seconds while the start time for each remaining sample s_i is $t_i \geqslant 0$ seconds. Hence, every selected user recording s_i, for $i > 1$, is present in the timeline with $t_i \geqslant 0$ seconds. Figure 6 illustrates an example. This shows the protocol's developer's view, which shows the timeline (bottom blue and green bars).

4.2. Tasks

Task 1 to 3 were used as training tasks but we also used them to assess the proposed technique's audio directional quality. In these tasks, participants train their sound localization ability for the next tasks (task 4 and 5).

Tasks 1, 2 and 3 consist of the reproduction of one, two and three audio sources simultaneously. The first task uses the piano sequence, task 2 uses the piano and guitar sequences and task 3 uses the three instruments sequences (section 4.1). The notes sequences are played several times (16, 9 and 6 times in task 1, 2 and 3, respectively). The directions of the instruments changed randomly. The possible directions are indicated in figure 7.

For every sequence reproduction, the volunteers were asked to register the perceived direction using the numbers provided in figure 7. The volunteers were asked to picture themselves in the center of the referential, with the circle numbered as 1 exactly in front of them, the circle numbered as 3 exactly at their right, etc. In order to better determine the direction of the sequences, the volunteers can simulate head rotations using the mouse and are provided with a button that allows them to return to the original orientation.

These three tasks focused mainly on identifying the audio source locations, in order to test directional audio quality. Therefore, the visual component of those tasks was ignored to keep them simple and have the user focusing on the audio. On the contrary, the fourth and fifth tasks consider both the visual and audio components in the context of the real application's goal.

The main goal of task 4 was to test all parameters simultaneously (*i.e.*, directional quality, sense of space and immersion). In this task, the video player (user's view in figure 6) displayed a concert video with a professional and two UGC recordings with some

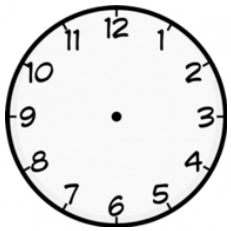


Figure 8: Clock system for audio source location.

Figure 9: Timeline of the audio and video used for the task 4. The blue bar aggregated to video screen shots represent the video. The green bar immediately below the blue bar is the professional recording. Both smaller green bars at the bottom are the UGC recordings.

overlapping and some non-overlapping regions. A clock system was used for sound source location as presented in figure 8. 12 o'clock represents 0° as in figure 3.

The professional recording was located at 12 o'clock, while the UGC recordings were placed at 8 o'clock and at 6 o'clock, by order of appearance respectively. Figure 9 displays the timeline for this task. The recordings in task 4 were played from static different directions. That is, the directions of the three recordings did not change during this task.

In task 5 the directions of the sounds were not static. This task tested if the users perceive a sense of space and directional audio when the directions of the sounds change dynamically.

In this task, participants used the two approaches developed for the video component (section 3.1) in an iPad.

5. RESULTS

In the first three tasks,we used the following classification scheme, where from *error type 1* to *4* the test subject failed to identify the audio source location:
– *Error 0* – the test subject identified successfully the audio source location;
– *Error type 1* – the answer provided was the location at 45 °from the correct audio source location;
– *Error type 2* – the answer provided was the location at 90 °from the correct audio source location;
– *Error type 3* – the answer provided was the location at 135 °from the correct audio source location;
– *Error type 4* – the test subject answered the location in the opposite location (*i.e.*, at 180 °).

Figures 10, 11 and 12 present the results of tasks 1, 2 and 3, respectively. All audio sources for all tasks exceeded more than 70% of right answers, which shows the directional quality of the spacialized sounds obtained with our technique.

The results of task 4 show that the sounds combined and spatialized by our technique give a sense of space and of directional audio. Users perceive that recordings played simultaneously (*i.e.* overlapping recording in the same timeline segment) have different directions. As mentioned above (section 4.2) this task used a

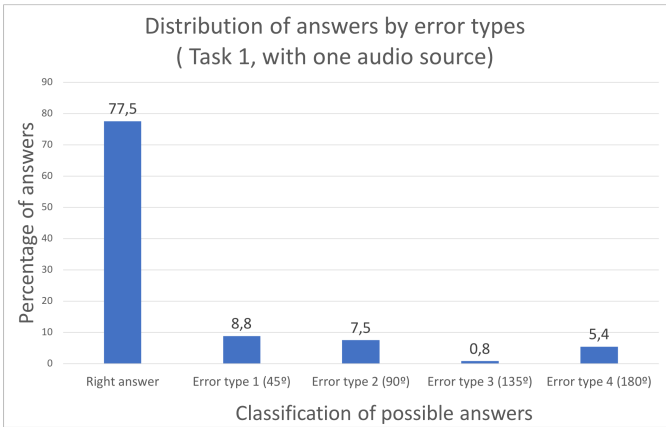


Figure 10: One audio source error distribution.

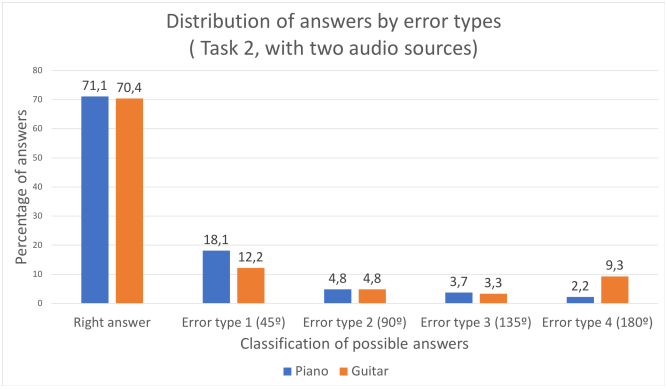


Figure 11: Two audio source error distribution.

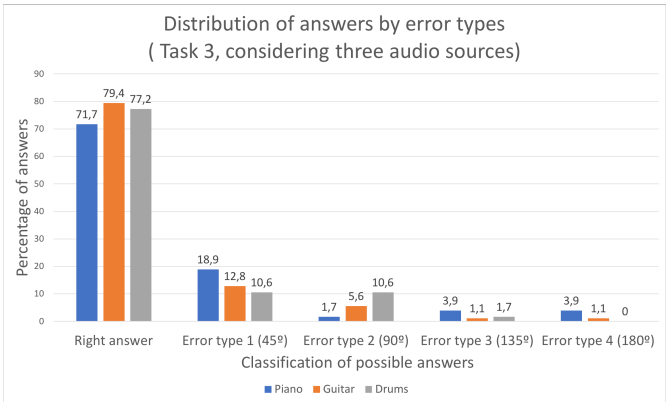


Figure 12: Three audio source error distribution.

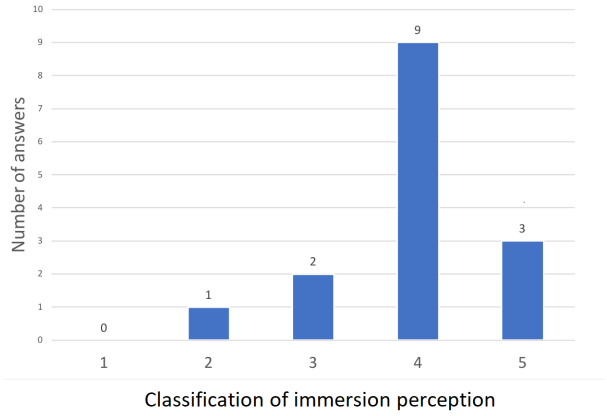


Figure 13: Immersion perception level in task 4.

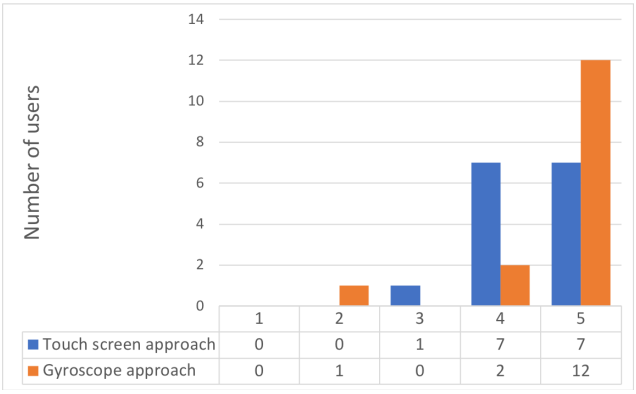


Figure 14: Azimuth perception level with the touch screen approach (approach 1, in blue) and with the gyroscope approach (approach 2, in orange).

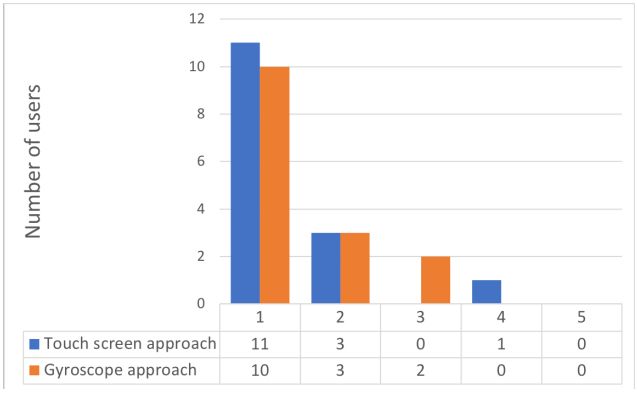


Figure 15: Elevation perception level with the touch screen approach (approach 1, in blue) and with the gyroscope approach (approach 2, in orange).

professional recording, s_1, located at 12 o'clock, one UGC recording, s_2, at 8 o'clock and another, s_3, at 6 o'clock. 93.3% of the subjects localized s_1 correctly, and 73.3% localized s_3 correctly. Only one subject identified the audio source at 8 o'clock correctly but 66.7% of the participants chose the 9 o'clock direction, showing that the perception of the direction of s_2 was close to the real one (and within the 30° localization precision of humans [11]).

The results from task 4 also show that the technique provides the sense of immersion. Subjects were asked to classify the level of audio immersion they felt while performing this task. A 5-point Likert scale was used, where 1 was *the experience was not immersive* and 5 was *the experience was strongly immersive*. The results are shown in figure 13. While only 20% of the subjects chose answer 5, 60% of them chose answer 4 (*the experience was very immersive*), which results in 80% of the subjects judging the experience as very or strongly immersive.

For task 5, users were asked if they perceived direction variations when moving the visible region (figure 4) horizontally and vertically. A 5-point Likert scale was used, where 1 was *no variation perceived* and 5 was *strong direction variation perceived*. The results are shown in figures 14 and 15.

Figure 14 shows that most users perceive variations in azimuth very easily. On the other hand, figure 15 shows that most users did not perceive variations in elevation. This task shows that users can identify variations in azimuth when the directions of the sounds change dynamically, which indicates that the sense of space and directional audio is not lost with dynamic direction changes. On the other hand, variations in elevation remained unnoticed. This was an expected result as humans do not perceive elevation easily. Since the results for the azimuth depend on the interaction approach (touch screen *vs* gyroscope), this task also show that the sense of directional audio is dependent on the type of user interaction with the application.

6. CONCLUSIONS

While the industry has been developing domestic solutions on user immersion that have a particular focus on the visual component, here we focus on audio immersion. We propose a technique that combines user-generated content (and possibly professional recordings) of the same event, to create a final spatialized immersive audio signal that can be combined with video in an interactive tool. The proposed technique spatializes the individual user recordings usings HRTFs, and organizes and synchronizes them with an audio fingerprinting based technique.

We run a user test that showed that the combination of the different recordings from the same event with the proposed technique, where each recording has its own individual characteristics and quality, provides a sense of immersion that the user can experience when listening to the recordings through headphones. The results also show that the proposed technique gives a sense of space and directional audio for azimuth direction changes. Nonetheless, the sense of directional audio is dependent on the type of user interaction with the application.

The current version of the prototype lacks HRTFs individualization. Since different people have different pinnae, the HRTFs set used in our prototype does not adapt equally to all people. As future work, we can extend the prototype to use more HRTFs sets such that it will be possible to choose the HRTF functions that best adapted to each listener, in order to produce more individual and personalized results that best fit the listener.

7. REFERENCES

[1] J. Madigan, *Getting Gamers: The Psychology of Video Games and Their Impact in the Peope who Play Them*, Ebook. Maryland: Rowman Littlefield Publishers, 2015.

[2] I. Rodrigues, R. Teixeira, S. Cavaco, V. Bonifácio, D. Peixoto, Y. Binev, F. Pereira, A. Lobo, and J. Aires-de Sousa, "2D spatial audio in a molecular navigator/editor for blind and visually impaired users," *20th International Conference on Digital Audio Effects*, 2017.

[3] S. Cavaco, D. Simões, and T. Silva, "Spatialized audio in a vision rehabilitation game for training orientation and mobility skills," *18th International Conference on Digital Audio Effects*, 2017.

[4] Y. Cohen, J. Dekker, A. Hulskamp, D. Kousemaker, T. Olden, C. Taal, and W. Verspage, "Demor, location based 3d audiogame," 2004.

[5] N. Montan, "AAR - an audio augmented reality system," M.S. thesis, Department of Microeletronics and Information Technology, KHT, Royal Institute of Technology, Stockholm, 2012.

[6] J. Guerreiro, *Enhancing Blind People's Information Scanning with Concurrent Speech*, Ph.D. thesis, University of Lisbon, Lisboa, Portugal, 2016.

[7] G. Mordido, J. Magalhães, and S. Cavaco, "Automatic organisation, segmentation, and filtering of user-generated audio content," *25th European Signal Processing Conference (EUSIPCO)*, 2017.

[8] G. Mordido, J. Magalhães, and S. Cavaco, "Automatic organisation, segmentation, and filtering of user-generated audio content," *IEEE 19th International Workshop on Multimedia Signal Processing*, 2017.

[9] S. E. Chen, "Quicktime VR: An image-based approach to virtual environment navigation," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. ACM, 1995, pp. 29–38.

[10] V. Pulkki, *Spatial sound generation and perception by amplitude panning techniques*, Ph.D. thesis, 2001.

[11] D. Wang and G. Brown, *Computational auditory scene analysis: Principles, algorithms, and applications*, Wiley-IEEE press, 2006.

[12] J. Rees-Jones and D. Murphy, "A comparison of player performance in a gamified localisation task between spatial loudspeaker systems," .

[13] S. Handel, *Listening: An Introduction to the Perception of Auditory Events*, A Bradford book. MIT Press, 1989.

[14] W.A. Yost and D.W. Nielsen, *Fundamentals of Hearing: An Introduction*, Holt, Rinehart and Winston, 1977.

[15] A. Farina and E. Ugolotti, "Subjective comparison of different car audio systems by the auralization technique," in *Audio Engineering Society Convention 103*. Audio Engineering Society, 1997.

[16] G. Mordido, "Automated organization and quality analysis of user-generated audio content," M.S. thesis, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, 2017.

ANALYSIS AND CORRECTION OF MAPS DATASET

Xuan Gong *
School of Electronic Information and Communications
Huazhong University of Science and Technology
Wuhan, China
M201771839@hust.edu.cn

Wei Xu †
School of Electronic Information and Communications
Huazhong University of Science and Technology
Wuhan, China
xuwei@hust.edu.cn

Juanting Liu
School of Electronic Information and Communications
Huazhong University of Science and Technology
Wuhan, China
M201871969@hust.edu.cn

Wenqing Cheng
School of Electronic Information and Communications
Huazhong University of Science and Technology
Wuhan, China
chengwq@mail.hust.edu.cn

ABSTRACT

Automatic music transcription (AMT) is the process of converting the original music signal into the digital music symbol. The MIDI Aligned Piano Sounds (MAPS) dataset was established in 2010 and is the most used benchmark dataset for automatic piano music transcription. In this paper, error screening is carried out through algorithm strategy, and three data annotation problems are found in ENSTDkCl, which is a subset of MAPS, usually used for algorithm evaluation: (1) there are 342 deviation errors of midi annotation; (2) there are 803 unplayed note errors; (3) there are 1613 slow starting process errors. After algorithm correction and manual confirmation, the corrected dataset is released. Finally, the better-performing Google model and our model are evaluated on the corrected dataset. The F values are 85.94% and 85.82%, respectively, and it is correspondingly improved compared with the original dataset, which proves that the correction of the dataset is meaningful.

1. INTRODUCTION

Automatic music transcription (AMT) is the process of converting acoustic music signals into digital music symbols, which is a challenging task in the field of music signal processing and music information retrieval (MIR) [1]. It consists of several subtasks, including multi-pitch estimation, onset offset detection [2], instrument recognition, beat and rhythm tracking [3]. Automatic music transcription systems can be used in music education, music creation, music production [4], music search [5] and so on. At present, AMT is still considered to be a challenging and open problem, especially for automatic piano music transcription [6]. The overlapping of sound events at the same time often shows harmonic overlap [7], which makes the identification task more difficult.

MIDI Aligned Piano Sounds (MAPS) [8] dataset was established in 2010 and is the most widely used benchmark dataset for piano transcription. MAPS dataset contains about 31GB of audio recordings. There are 9 types of audio recordings corresponding to different piano types and recording conditions, among which 7 types of audio are produced by software piano synthesizers, and 2 subsets ENSTDkAm and ENSTDkCl are recorded by an upright Disklavier piano. In general, ENSTDkCl in MAPS Disklavier dataset is adopted to evaluate. Since the MAPS dataset was established, only Ycart [9] updated it and added rhythm and key information in the annotation.

With the research on the AMT system, some researchers found that there were some problems in the MAPS Disklavier dataset for evaluation. Ewert [10] found in his study of studio piano transcription that in MAPS Disklavier dataset some midi-based annotation deviation exceeded the order of magnitude described in the document. Therefore, he used the greater temporal tolerance which might provide a more realistic impression of the transcription performance. Li [11] found several issues with the MAPS Disklavier dataset, including annotation deviations, omitted notes and recordings consisting entirely of percussive keybed and pedal noises. Hawthorne [12] suggested that some of the low-velocity notes in the annotations were not played during the Yamaha piano playing.

The development of machine learning has led many scholars to apply it to the piano transcription system. In [13]-[14], the authors demonstrated the potential of a single CNN-based acoustic model and an RNN model for polyphonic piano music transcription. The model proposed by Hawthorne [12], which we called the Google model, was a new method to predict the pitch using CNN and LSTM, and achieved the best system performance in 2018. We also designed a CNN-based model for piano transcription [15]. The model consisted of two networks, and an onset-event detector was used to align the pitch onset to a more accurate position. In the end, our model achieved an F1-measure score of 85.15% on the MAPS ENSTDkCl dataset, which is better than Google's system performance.

Data plays an important role in machine learning methods. After analyzing the results of errors in the ENSTDkCl set evaluation, we find that some errors are caused by the dataset itself instead of model identification. We believe that it is necessary to modify the dataset rather than directly modifying the criteria of the evaluation to a broader range. We use the algorithm preselection and manual check to correct the dataset. There are the deviation of midi annotation errors, unplayed note errors and slow start process errors in the data error, whose number is 342, 803 and 1613. The two models are evaluated using the modified dataset, with F1-measure scores 85.94% and 85.82%, respectively, and the modified dataset reflected the performance of the transcription system more accurately. In addition, applying

* This work is supported by The National Natural Science Foundation of China (No. 61877060).
† Corresponding author.

these modified true piano data to the training process may help the network to learn the characteristics of true piano playing better.

2. THE ANALYSIS OF ERROR

The MAPS dataset includes midi files, txt files, and corresponding audio files. The txt file contains information in the midi, including the onset time, offset time and pitch. We analyze the data and find that the data set itself contains three problems: deviation errors of midi annotation, unplayed note errors and slow start process errors.

2.1. Method of analysis

We analyze the data of the ENSTDkCl set. Direct data analysis is a huge task, so we use the Google model [12] and our model [15] for transcription in the current research. We analyze the data based on the common error results of the two models. Common errors in the transcription of the two models include missed detection and surplus detection. According to the general evaluation criteria, the missed detection does not detect the corresponding pitch onset event within ±50 ms of the corresponding time of midi, and the surplus detection is that there is no matching midi annotation within ±50ms of the pitch onset event. There are few errors in surplus inspections and most of them are detection errors of the model. Therefore, the results of the missed detection are analyzed to observe the data problem.

2.2. Reasonable midi annotation

To analyze the data, we first describe the general performance of the note onset. The data of the MAPS subset ENSTDkCl is generated in the real environment. When there is a pitch onset event in the midi, the corresponding piano key that is tapped will generate energy, and the amplitude of the frequency corresponding to the pitch rises. Except the partial bass, the fundamental and second harmonics of the tone contain most of the energy produced by the tap [16]. So we use the fundamental and second harmonics in the figure to show the onset. The spectral transformation of the audio is more conducive to the presentation of its sound characteristics, so this analysis uses the CQT spectral transform.

As shown in Fig. 1 is a pitch onset event, Fig 1.a is a three-dimensional spectrogram, where the three axes are the time, frequency and amplitude of the CQT transform. The height and color represent the amplitude at the same time. The larger the amplitude, the closer the color is to yellow and the higher the height. The white dashed line represents the same time t1, and the white curve represents the case where the center frequency corresponding to the pitch changes with time. The red box in Fig 1.b represents the time and fundamental frequency range corresponding to the note event, and the blue box represents the frequency multiplication range of the pitch event. The abscissa is time, the ordinate is frequency, and the color depth indicates the magnitude of the corresponding time and frequency. The greater the degree of black, the larger the amplitude. The red curve in the lower of Fig 1.c shows the center frequency component of the fundamental frequency range, and the blue curve shows the center frequency component of the second harmonics. The three figures represent the same pitch event from different angles. In Fig 1.a, the midi is marked with time t1=177.574s, and the frequency amplitude of the pitch F5 has a large rise. From the Fig 1.b and the Fig 1.c, the fundamental frequency

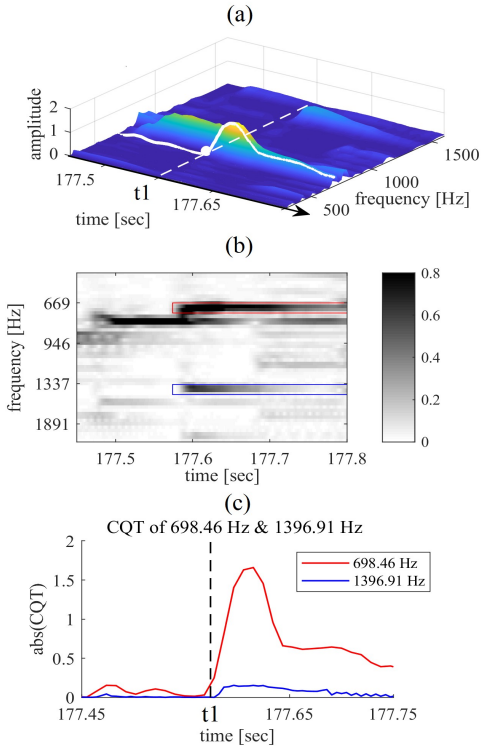


Figure 1: *The spectrum of reasonable midi annotation. (a) is a three-dimensional spectrogram, (b) is the colormap of spectrum, and (c) are the center frequency components of the fundamental frequency and second harmonics.*

amplitude rises rapidly to 1.5 at t1, and the second harmonics also increases correspondingly. Therefore, when a pitch onset event occurs, the amplitude of the base frequency corresponding to the pitch will increase, and the amplitude of second harmonics will also have a certain upward trend.

2.3. The deviation error in midi

The MAPS subset ENSTDkCl is automatically played by the piano based on the information in midi, but we found that there is a case where the playing time in the audio is shifted from the midi annotation by more than 50 ms. The two time points have obvious inconsistency. This deviation in the midi annotation can lead to inaccurate final evaluation results.

As shown in Fig. 2 is a fragment of MAPS_MUS-schuim-1_ENSTDkCl, In Fig 2.a, t1 is the starting time of the midi pitch E4, 112.389s, and the time represented by t2 is 112.489s; in Fig 2.b the red box indicates the time and the fundamental frequency range of the corresponding pitch E4 in the midi; and the c is the curve of the two center frequencies corresponding to the pitch E4 with time. Fig 2.a shows that there is no amplitude increase at t1 and the frequency amplitude rises at t2. We think that t2 should be the onset time of this pitch event according to the general representation of onset event. Fig 2.b shows that all frequencies in the fundamental frequency range have no pitch starting characteristics at time t1. Observing the Fig 2.c, there is certain deviation between t1 and t2, and the distance between them is 0.1s. Therefore, we think that this is a deviation error in midi annotation.

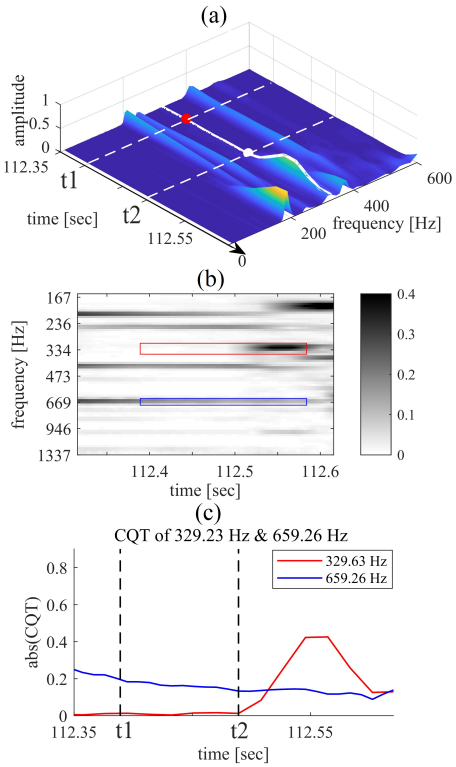


Figure 2: *The spectrum of the deviation error in midi. (a) is a three-dimensional spectrogram, (b) is the colormap of spectrum, and (c) are the center frequency components of the fundamental frequency and second harmonics.*

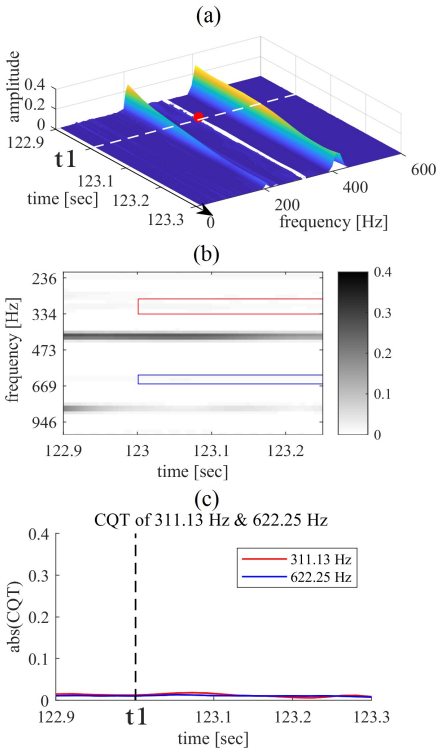


Figure 3: *The spectrum of unplayed note error which is a White Noise. (a) is a three-dimensional spectrogram, (b) is the colormap of spectrum, and (c) are the center frequency components of the fundamental frequency and second harmonics.*

2.4. The unplayed note error

When there is a note onset in the midi, the amplitude rises corresponding to the pitch frequency. It is not normal that there is a note onset event in midi but the frequency is degraded or absent in the spectrum. We believe that this is the case when the note is not played on the piano, and then we describe it with data.

In the first case, the note is not heard in the audio at the start time of the midi. By observing the frequency spectrum, it is found that the frequency component of the corresponding pitch at the labeling time is very small, which is the same as the noise level, so we consider that the pitch is not played.

As shown in the Fig. 3, it is a fragment of the MAPS_MUS-schuim-1_ENSTDkCl, the dotted line shown in Fig 3.a represents the midi annotation time t1, and its time is 123.001s. The white curve represents the change of the amplitude (311.13Hz) of the fundamental central frequency of pitch D#4 over time. In the Fig 3.b, the fundamental frequency and the second harmonics of the pitch D#4 are small and have no upward trend during the marked time period. It can be seen from Fig 3.c that the center frequency of the two frequency ranges are both below 0.1. This is not the performance of normal playing, so we think that the piano does not play D#4 at t1.

The second case is that the beginning of the same pitch marked at two very close times can only be heard once in the audio, so we don't think there is a new play in one of the labels. By analysing the spectrum, we find that this situation has a specific performance.

As shown in the Fig. 4, it is the track MAPS_MUS-mz_570_1_ENSTDkCl, t1 is the midi ann

t2 is 436.920s, the white curve represents the curve of the center frequency component (466.16 Hz) of the pitch A#4, and Fig 4.b shows the spectral characteristics of the fundamental frequency and the frequency doubling corresponding to the pitch A#4, and two marked points are observed from the Fig 4.c. The characteristic is that compared with t1, the fundamental frequency center component of the pitch A#4 of t2 shows a downward trend, and there is no frequency rising process like the normal playing onset. We didn't hear the two playing while listening to the audio, so we thought that the piano did not play the pitch A#4 at t2.

2.5. Slow start process error

As a percussion instrument, when the piano key is pressed, there will be a process in which the frequency component of the key rises. When analysing the data, we find a pitch onset event, whose frequency component rises lasts longer than 100ms. This is not a common performance in piano playing, and the labelling itself is difficult to unify, which is a huge challenge for the recognition task.

As shown in the Fig. 5, the track MAPS_MUS-pathetique_1_ENSTDkCl, in Fig 5.a, the white curve is the change of the fundamental frequency of the pitch G4 with time, and the corresponding midi labeling time t1 is 292.26 s. It can be seen from the figure that this frequency component rises continuously. It can be seen from the Fig 5.c that the amplitude has been rising in the range of 292.15s-292.3s, and the rising process lasts for 150ms. The change process is long, which is difficult to accurately identify in specific tasks.

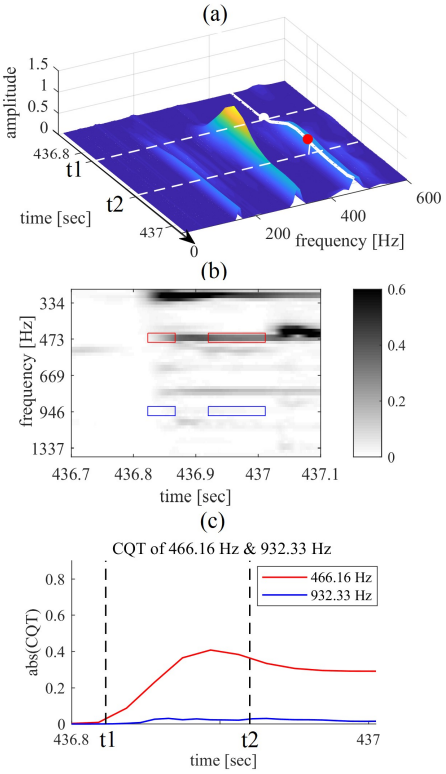


Figure 4: *The spectrum of unplayed note error whose frequency declines. (a) is a three-dimensional spectrogram, (b) is the colormap of spectrum, and (c) are the center frequency components of the fundamental frequency and second harmonics.*

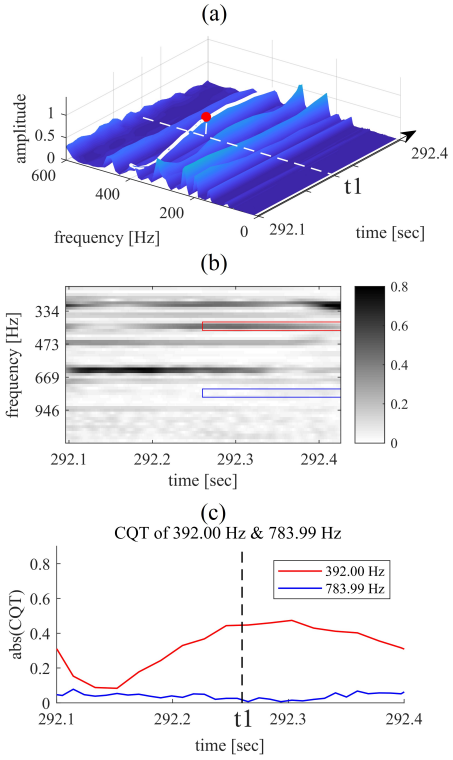


Figure 5: *The spectrum of slow start process error. (a) is a three-dimensional spectrogram, (b) is the colormap of spectrum, and (c) are the center frequency components of the fundamental frequency and second harmonics.*

3. JUDGEMENT AND CORRECTION METHOD

Based on the analysis in above chapter, we make the quantify judgment of three kinds of notes, there are 342 deviation errors, 803 unplayed note errors and 1613 slow starting process errors. We list these notes in Cl-correction-v1.0, the numbers of each kind of notes are listed in Table.1. The complete files are published on https://github.com/itec-hust/MAPS_ENSTDkCl-Dataset-Correction-v1.0.

3.1. Deviated notes correction methods

After data viewing and analysis, we find that deviated note whose label (t1) and true playing time (t2) have spacing over 50ms contain two features. Firstly, the time of spectrum's fastest growth spaces over 50ms from t1, and there is no obvious growth in t1-50ms to t1+50ms. At t2, the spectrum of the notes grows at a very fast rate. Secondly, the evaluation will change if we increase the evaluation tolerance. When the model detection result is consistent with t2, when we set the evaluation tolerance result as 50ms, the note is judged as multiple detection at t2 and miss detection t1. However, if we set the evaluation tolerance result as 150ms, the note will be judged as correct detection at t2.

Based on the first feature, we quantify the certain frequency component rising ratio of time point *i* as *St[i]*, which represents the amplitude rising degree within *10ms* around time point i. When the note event occurs, the corresponding frequency component will rise rapidly, therefore *St[i]* will be a large positive number; when the frequency component decreases, *St[i]* will be 0.

According to the above analysis, we first extract the *St[t1-150ms:t1+150ms]* sequence by extracting the fundamental frequency component and the second harmonic component of all notes.

Based on the second feature, we find out the corresponding notes in the Google model's results and our model's results, calculate the intersection of the two sets. After the spectrum comparison and listening test, we find that the following characteristics exist:

$$\max(St[t1-150ms:t1+150ms]) \geq \max(St[t1-50ms : t1+50ms])*\alpha \quad (1)$$

After statistical analysis, we set α as 4. When St sequence does not satisfy this condition, we believe that the detection error is due to slowly rising or some other reasons instead of deviated label. After the error is detected, we modify the onset time, select the two models at the detection start time of the same pitch event, average the two results as the modified onset, and the corrected end time as the corrected start time plus the duration of the note in the original midi.

In the deviatedNote directory of *Cl-correction-v1.0*, we list all the label deviated notes, and the data is arranged in the following order: (t1, t2, pitch, t3, t4), where (t1, t2, pitch) is the original midi, (t3, t4) are the corrected onset and offset.

Table 1: *Numbers of each kind of error notes in each song.*

Name	deviated	unplayed	slow	Name	deviated	unplayed	slow
alb_se2	0	0	11	mz_331_3	4	4	26
bk_xmas1	3	170	63	mz_332_2	32	41	91
bk_xmas4	2	45	16	mz_333_2	19	19	80
bk_xmas5	1	25	34	mz_333_3	0	2	22
bor_ps6	0	25	37	mz_545_3	1	1	26
chpn-e01	0	0	1	mz_570_1	28	71	238
chpn-p19	2	10	22	pathetique_1	7	25	87
deb_clai	24	24	35	schu_143_3	40	41	125
deb_menu	1	12	13	schuim-1	33	29	244
grieg_butterfly	5	2	14	scn15_11	3	9	16
liz_et6	1	6	25	scn15_12	12	15	33
liz_et_trans5	20	41	78	scn16_3	17	21	43
liz_rhap09	15	68	90	scn16_4	18	32	41
mz_311_1	16	22	41	ty_maerz	13	17	15
mz_331_2	20	18	32	ty_mai	5	8	14

3.2. Unplayed notes judgement

In the analysis in Section 2.3, it can be found that for unplayed note, the spectrum is completely degraded or basically white noise. Based on the first feature, we filter out all the notes whose spectrum is falling near the onset decision time, ie *max(St[onset-50ms : onset+50ms])=0*. We list them in set 1. Based on the second feature, the notes whose spectral components are extremely small and randomly change might not be played in the audio. Therefore, we count the spectral components of the quite period, and set the spectral median β as the threshold of the spectral component of the unplayed note, that means

$$max\left(cqt\left[onset-50ms\ :\ onset+50ms\right]\right) \leq \beta \tag{2}$$

We find all the notes corresponding to the second feature and list them in set 2. After the two sets are determined, we did listening test. In the unplayedNote directory of Cl-correction-v1.0, we list the notes that all tracks are not playing, and the data is arranged in the following order: (t1, t2, pitch), where (t1, t2, pitch) is the original midi.

3.3. Slow start note judgement

The slow-starting notes are characterized by a note that the duration of the note volume from 0 to the final value is significantly longer than other notes, and the spectrum rising process is slow and gentle. Therefore, we have filtered the spectrum growth value. When max(St)<γ and St[onset-50ms : onset+50ms]≥ max(St) * δ, it is defined as a slow start note. Different models may have different results on these notes, so we will list such detected notes, but do not make time correction or note existence judgement, the misjudgment of such notes may not be sufficient to indicate the validity of the research results. After the above judgment, we have screened a total of 155 slow-starting notes. In the slowStartNotes directory of Cl-correction-v1.0, we list these notes, and the data is arranged in the following order: (t1, t2, pitch), where (t1, t2, pitch) is the original midi.

4. EXPERIMENT

Based on the analysis in above chapter, we make the quantify judgment of three kinds of notes, and listing these notes in Cl-correction-v1.0, the numbers of each kind of notes are listed in Table.1, Our correction methods are as follows.

In order to verify the validity of the label correction, we compare the model evaluation results of the original dataset and the corrected dataset. If the corrected label is not detected as an error in the evaluation, and the unplayed notes are not judged to be missing, then this correction is worthwhile. In the corrected dataset, we modified the annotation of the note with the deviated label, deleted the unplayed notes, the data format is consistent with the original label. The corrected dataset is published in the correct_dataset directory of *Cl-correction-v1.0*, and the slow-start notes are listed in the slowStartNote directory.

The mir_eval library [17] is used to calculate the accuracy, recall, and F1 values that are widely used for AMT evaluation. The metrics are defined as follows:

$$P = \frac{N_{TP}}{N_{TP} + N_{FP}} \tag{3}$$

$$R = \frac{N_{TP}}{N_{TP} + N_{FN}} \tag{4}$$

$$F1 = \frac{2 * P * R}{P + R} \tag{5}$$

In Equation3, 4 and 5, P is precision, R is recall and F1 is the f1-measure which is a comprehensive score that considers the precision and recall. And N_{TP} is the number of true positives, N_{FP} is the number of false positives and N_{FN} is the number of false negatives. During the evaluation process, we set the time tolerance to 50ms.

Table 2: *Evaluation result of two models*

	Result of Google model [12]			Result of our model [15]		
	F	P	R	F	P	R
original dataset	84.34%	85.95%	83.05%	85.09%	87.8%	82.83%
corrected dataset	85.94%	89.77%	82.73%	85.82%	88.5%	83.59%

Based on the revised annotations, we performed the accuracy test as shown in Table 2. It can be found that the performance of the two models has increased in the evaluation results of the revised data set. In the process of data error review, we find that most of the errors detected by the model are errors caused by generalization errors, and the number of common errors is also reduced, so we think this data correction is effective.

5. SUMMARY AND FUTURE WORK

In the field of AMT, dataset construction has always been a difficult problem. Synthetic audio can't completely replace the audio of real piano performance. The authenticity of audio and the authenticity of labels cannot be well unified. Therefore, from the perspective of audio spectrum and human listening, we corrected some unreasonable labels, including deviation labels and unplayed labels, and listed the slow-starting notes in the audio. Finally, the validity of the corrected dataset was verified in two different model evaluation. We published our result on the Cl-correction-v1.0, providing a reference for the future work of subsequent researchers. In the future research, we will try to figure out the reasons why the spectrum will have slow rising of the slow-start notes, and summarize some other problems in the dataset.

6. ACKNOWLEDGEMENTS

This work is supported by The National Natural Science Foundation of China (No. 61877060).

7. REFERENCES

[1] A. P. Klapuri and M. Davy, Eds., “Signal Processing Methods for Music Transcription,” New York, NY, USA: Springer, 2006.

[2] C. Duxbury, M. Sandler, and M. Davies, “A hybrid approach to musical note onset detection”, *Proc. Digital Audio Effects Conf. (DAFX,02)*, Hamburg, Germany, pp.33–38, 2002.

[3] X. Shao, M.C. Maddage, C. Xu, and M.S. Kankanhalli, “Automatic music summarization based on music structure analysis,” *IEEE International Conference on Acoustics, Speech, and Signal Processing, Proceedings*, Philadelphia, Pennsylvania, USA, pp.1169–1172, 2005.

[4] M. Müller, D. P. Ellis, A. Klapuri, and G. Richard, “Signal processing for music analysis,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 6, pp. 1088–1110, 2011.

[5] M. Schedl, E. Gómez, and J. Urbano, “Music information retrieval: Recent developments and applications,” *Foundations and Trends in Information Retrieval*, vol. 8, pp. 127–261, 2014.

[6] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri, “Automatic music transcription: challenges and future directions,” *Journal of Intelligent Information Systems*, vol. 41, no. 3, pp. 407–434, Dec. 2013.

[7] N. H. Fletcher and T. D. Rossing, “The Physics of Musical Instruments,” New York, NY, USA: Springer-Verlag, 1991.

[8] V. Emiya, R. Badeau, B. David, “Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1643–1654, 2010.

[9] A. Ycart, E. Benetos, “A-MAPS: Augmented MAPS dataset with rhythm and key annotations,” in *19th International Society for Music Information Retrieval Conference Late-Breaking Demos Session*, 2018.

[10] S. Ewert, M. Sandler, “Piano transcription in the studio using an extensible alternating directions framework,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no.11, pp. 1983-1997, 2016.

[11] S. Li, "Context-Independent Polyphonic Piano Onset Transcription with an Infinite Training Dataset,” arXiv preprint arXiv:1707.08438 (2017).

[12] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, D. Eck, “Onsets and frames: Dual-objective piano transcription,” in *Proc. International Society for Music Information Retrieval Conference*, 2018

[13] R. Kelz, M. Dorfer, F. Korzeniowski, S. Böck, A. Arzt, “On the potential of simple framewise approaches to piano transcription,” arXiv preprint arXiv:1612.05153, 2016.

[14] S. Böck, M. Schedl, "Polyphonic piano note transcription with recurrent neural networks,” *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 121-124, 2012.

[15] K. Sicong, X. Wei, L. Wei, G. Xuan, L. Juanting, C. Wenqing, “Onset-aware polyphonic piano transcription: A CNN-based approach,” in *Proceedings of 2019 the 9th International Workshop on Computer Science and Engineering*, 2019. (In press).

[16] H. Tianqian, "Effect of striking point and striking dynamics on the amplitude spectrum of piano signals,” *Audio Engineering*, 2003.

[17] C. Raffel, B. McFee, “mir_eval: A transparent implementation of common MIR metrics,” In *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*, 2014.

OPTIMIZATION OF AUDIO GRAPHS BY RESAMPLING

Pierre Donat-Bouillud
Music Representation Team
STMS/Ircam/Inria/Sorbonne universite
Paris, France
pierre.donat-bouillud@ircam.fr

Jean-Louis Giavitto
Music Representation Team
STMS/Ircam
Paris, france
giavitto@ircam.fr

Florent Jacquemard
Inria
Paris, France
florent.jacquemard@inria.fr

ABSTRACT

Interactive music systems are dynamic real-time systems which combine control and signal processing based on an audio graph. They are often used on platforms where there are no reliable and precise real-time guarantees. Here, we present a method of optimizing audio graphs and finding a compromise between audio quality and gain in execution time by downsampling parts of the graph. We present models of quality and execution time and we evaluate the models and our optimization algorithm experimentally.

1. INTRODUCTION

Interactive music systems (IMS) [1] are programmable systems that combine audio signal processing with control in real-time. At run time, during a concert, they process or synthesize audio signals in real-time, using various audio effects. For that purpose, they periodically fill audio buffers and send them to the soundcard. They also make it possible to control the sound processing tasks, with aperiodic control (such as changes in a graphical interface) or periodic control (for instance, with a low frequency oscillator). Audio signals and controls are dealt with by an *audio graph* whose nodes represent audio processing tasks (filters, oscillators, synthesizers...) and edges represent dependencies between these audio processing tasks.

Puredata [2] and Max/MSP [3] are examples of IMSs. They graphically display the audio graph, but modifying it at run time as a result of a computation can be complicated. Other IMSs, such as ChucK [4] or SuperCollider [5] are textual programming languages. They are also more dynamic. In Antescofo [6], human musicians and a computer can interact on stage during a concert, using sophisticated synchronization strategies specified in an augmented score, programmed with a dedicated language, that can also specify dynamic audio graphs [7].

Real-time constraints for audio: Audio samples must be written into the input buffer of the soundcard periodically. The buffer size can range from 32 samples for dedicated audio workstations to 2048 samples for some smartphones running Android, depending on the target latency and the resources of the host system. For a samplerate of 44.1kHz, such as in a CD, and a buffer size of 64 samples, the audio period is 1.45 ms. It means that the audio processing tasks in the audio graph are not activated for each sample but for a buffer of samples.

IMSs are not safety critical systems: a failure during a performance is not life-critical, it will not generally result in damages or

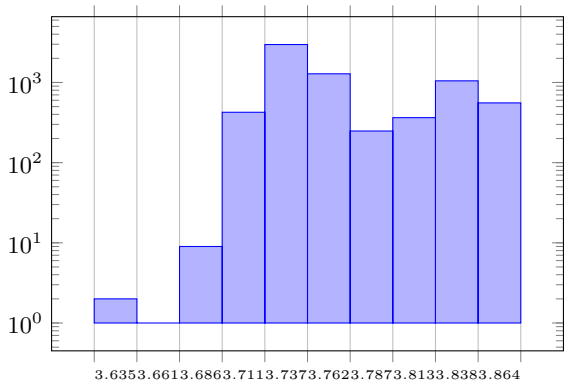


Figure 1: Histogram of relative deadlines for the *audio callback* on a MacBookPro with macOS. We execute a C++ test program generating a sawtooth signal for 10 s. The time budgets range from 3.64 ms to 3.89 ms, *i.e.* a 254 µs jitter, with a mean of 3.78 ms

injuries. However, audio real-time processing has strong real-time constraints. Missing the deadline for an audio task, *i.e.* the time allotted by the audio driver to fill its buffer, is immediately audible:

Buffer underflow The audio driver uses a circular buffer the size of which is a multiple of the size of the soundcard buffer. If the task misses a deadline, it does not fill the buffer quickly enough. Depending on the implementation, previous buffers will be replayed (*gun machine* effect) or silence will be played, which entails cracks or clicks due to discontinuities in the signal. A larger buffer decreases deadline misses but increases latency.

Buffer overflow In some implementations, filling the buffer too quickly can also lead to discontinuities in the audio signal, if audio samples cannot be stored to be consumed later.

On the contrary, in video processing, missing a frame among 24 images per second[1] does not entail a visible decrease in quality so that lots of streaming protocols [8] accept to drop a frame. Therefore, real-time audio constraints are more stringent than for video. Yet, they have not been investigated as much as real-time video processing.

Composers and musicians use IMS on mainstream operating systems such as Windows, macOS or Linux, where a reliable and tight estimation of the worst case execution time (WCET) is difficult to obtain, because of the complex hierarchy of caches of the

[1] Although missing a key frame in a compressed stream can be visible.

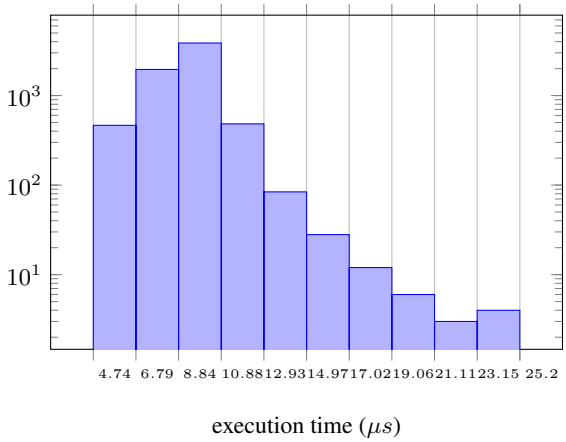


Figure 2: Histogram of the execution time for the same code generating a saw signal for 10 s. Here, we show the execution time at each cycle in the *audio callback* for generating a sawtooth signal on a MacBookPro with macOS. The execution times range from 4.74 µs to 25.20 µs with an average of 9.17 µs. The standard deviation is 1.59 µs.

processor, because there are usually no real-time scheduler or temporal isolation among tasks, and because it is difficult to predict which tasks will be executed at a given time. Mainstream operating systems[2] are not real-time systems and do not offer any strong guarantees on deadlines for audio processing (see Fig. 1) or on the execution times (see Fig. 2). Applications that perform audio computations have to compete for CPU and memory resources with other applications during a typical execution. Those platforms are also often on batteries and often change the frequency of the CPUs in order to save energy, thus totally changing execution times. It follows that we cannot assume that we know the WCET of tasks. Furthermore, IMSs are more and more ported to embedded cards such as Raspberry Pi and have to adapt to limited computation resources on these platforms. On the other hand, composers and sound designers create more and more complex musical pieces, with lots of dynamically interconnected nodes, using a sampling rate up to 96 kHz.

Hard real-time scheduling algorithms that depend on knowing the WCET cannot be applied here, as in practice, IMSs are executed on those operating systems without strong real-time guarantees.

When real-time constraints are not critical, modifying the quality of service (QoS) by partially executing some tasks or even discarding them is an option to consider. In the case of IMSs, tasks are dependent, with dependencies defined with the edges of the audio graph. The quality of a task is itself *position-dependent*: it depends on the position of the audio processing task in a path going from an audio input to an audio output. It means we cannot merely discard or degrade any task in the audio graph, to achieve an optimal QoS adaptation.

To deal with these challenges, IMSs have rather chosen to increase the available computation resources, in particular by increasing the parallelism of the audio graphs in scores to take advantage of the multicore processors. For example, an alternative scheduler for SuperCollider, Supernova [10], can use several cores to synthesize sounds. However, these new schedulers do not automatically parallelize the audio graph but require explicit instructions, such as `ParGroup` for SuperCollider, or `poly` for Max/MSP, and so are difficult to program.

In this paper, we propose an alternative solution where we explore how an audio graph can be optimized by degrading audio processing units without perceiving the degradations (or minimizing the degradation perception). The idea is to generate an *equivalent* but degraded audio graph where the execution time of the graph is decreased while decreasing the quality of the audio processing tasks.

Because of the dependencies, we do not degrade only a single task here, but we have to *choose a whole set of dependent nodes* in an audio graph to degrade. We aim at finding paths to degrade in the audio graph, where the quality of a task is position-dependent. The nodes are seen as *blackboxes* and degradations are made by *resampling* the signal flowing between nodes.

We describe a model of an audio graph based on the dataflow paradigm [11] with models of execution time and quality. We present an *offline* algorithm based on the models that explores exhaustively or randomly the possible degraded versions of an audio graph with constraints on execution time and quality. Heuristics help to select subpaths to degrade in the audio graph at execution time. We evaluate our optimization strategy on three different sets of audio graphs : an exhaustive enumeration of all possible graphs with few nodes, a random sampling of graph with many nodes, and graphs structurally generated from Puredata patches.

2. RELATED WORK

Degrading computations to achieve to respect some time criterion has been dealt with in the *approximate programming* paradigm or in real-time system theory, with *mixed criticality*.

Approximate computing [12] is a paradigm in which errors are allowed in exchange of an improvement of performance. The concept of correctness is relaxed to a correctness with a quantitative error. In [13], a graph is used to represent a *map-reduce* program, with map nodes which compute and reduce nodes which aggregate data. Offline, they generate approximate versions of the graph with a given precision. For that purpose, two kinds of transformations are considered: *substitution transformations* and *sampling transformations*, where the input is randomly downsampled. However, this model is aimed at batch-processing, not real-time audio graphs.

In multimedia systems, a basic strategy [14] related to mixed criticality consists of dividing tasks between a mandatory and an optional part, which can be discarded in case of overload of the processor. Real-time scheduling with this strategy does not entail too much overhead but does not handle dependencies between tasks, in particular for quality estimation. Some mixed criticality approaches address graph-based tasks for mixed-criticality systems, such as in [15]. However, the dependencies between tasks are functional dependencies: all tasks in a graph have the same criticity but it is possible to switch to other graphs with another criticity. Criticity levels are not like *qualities* that would depend on the topology of the graph.

[2]There is an earliest deadline first scheduler [9] on Linux, but it's typically not activated on mainstream distributions.

3. MODEL OF AN AUDIO GRAPH

3.1. Model of an audio graph

In an audio graph, audio streams flow between signal processing nodes at various rates, depending on what the nodes require. The *dataflow* model [11, 16] is well suited to describe these kinds of dependent tasks. In the usual *dataflow* model, no time information is provided, so we enrich this *dynamic dataflow* model with temporal information, such as start times or average execution times (ACET), as in the Time-Triggered Dataflow mode [17]. We also define a quality measure on the graph.

3.2. Dataflow model

The *dataflow* model is *data-oriented*: when there are enough data, seen as a sequence of *tokens*, on a node, the node is fired, consequently, yielding some tokens. More formally, we use the port graph formalism, as in [18]. A *dataflow* quadruplet $G = (V, P, E, \mu)$ where the vertices in V, represent the signal processing nodes and are pairs (I, O) with $I \subset P$ and $O \subset P$, the input and output ports. The edges in $E \subset P \times P$ represent the data flowing between vertices and connect an *output* ports of a vertice to an *input* port of another vertice. We note $p_1 \to p_2$ the edge between ports p_1 and p_2; $v_1 \to v_2$ would denote any edge $v_1.p \to v_2.p'$ between v_1 and v_2. We note $v.p$ the port p of vertice v. The function $\mu : P \to \mathbb{N}$ maps a port to the number of audio samples it consumes or produces.

Distinguished nodes The nodes without input ports are called *inputs* or *sources*. They are typically audio stream generators. The nodes without output ports are the *outputs* or *sinks*, as shown in Fig. 3. They are audio sinks to the soundcard for instance. Nodes that are neither *inputs* nor *outputs* are called *effects*.

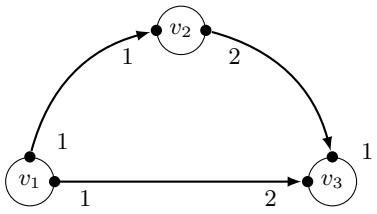


Figure 3: A simple *synchronous dataflow* graph with three nodes v_1, v_2 and v_3 with only one input port and one output port for each of them. Data flows from v_1 to v_3, from v_1 to v_2 and from v_2 to v_3. v_1 yields 1 token per firing, and v_3 requires 2 tokens to be fired. v_1 is a *source* node, v_3 a *sink* node.

3.3. Timed dataflow

A dataflow graph only describes the partial ordering of the firing of its nodes, but not their firing time instants. However, dataflow graphs are often used to describe real-time processing, including audio signal real-time processing. In the following, *input* nodes are given firing dates and *output* nodes, deadlines. Any node v gets an average case execution time (ACET), A_v and a worst case execution time (WCET), W_v. Although we will evaluate the model with the ACET, we could also evaluate with the WCET if we had an accurate WCET. We will note T_v the execution time of a node. We also assume that the graph is acyclic.

The audio processing nodes are periodic and we note $s_v^1 \dots s_v^n$ with $s_v^i < s_v^{i+1}$ the firing times of node v, where $s_v^{i+1} - s_v^i = s_v^2 - s_v^1 = T_v$ with T_v the period of node v. The samplerate f_{p_v} on a port p_v of node v is $\frac{\mu(p_v)}{T_v}$.

Execution time of a path On a path $\pi = v_1 \to \dots v_n$, the mean execution time (resp. worst case) is $\sum_{i=1}^{n} A_{v_i}$ (resp. $\sum_{i=1}^{n} W_{v_i}$).

Execution time of the whole graph For the whole graph G, on a parallel system with enough resources, the execution time is the largest execution time of a path between inputs and outputs, *i.e.* for the ACET, $A_G = \max_{\pi \in \text{Path}}\{A_\pi\}$. For instance, for Fig. 3:

$$A_G = \max\{A_{v_1 \to v_3}, A_{v_1 \to v_2 \to v_3}\}$$

In a sequential execution mode (for instance on an uniprocessor), it is the sum of the execution times of all the nodes:

$$A_{\mathcal{G}} = \sum_{v \in \mathcal{G}} A_v \tag{1}$$

Estimating A_v We measure the average execution time A_v of all the possible nodes that can be part of the audio graph. We do not care about the exact execution time, but rather of an ordering on the execution times between various versions of an audio graph. In addition, the execution time increases monotonically with the buffer input sizes. Thus, we only measure the average execution of a given buffer size. However, some nodes can have a variable number of inputs and outputs, such as for a `mixer` node. We do not want to measure all possible combinations of inputs and outputs. Experimentally, we find that the average case execution time $A_{\text{mixer}}(n_{\text{inputs}}, n_{\text{outputs}})$ of a mixer with n_{inputs} and n_{outputs} is given by:

$$\begin{aligned} A_{\text{mixer}}(n_{\text{inputs}}, n_{\text{outputs}}) = n_{\text{inputs}} \times \underbrace{(A_{\text{mixer}}(2,1) - A_{\text{mixer}}(1,1))}_{\text{cost of adding}} \\ + n_{\text{outputs}} \times \underbrace{(A_{\text{mixer}}(1,2) - A_{\text{mixer}}(1,1))}_{\text{cost of copying one buffer}} \end{aligned} \tag{2}$$

3.4. Quality

The quality of the audio graph is a subjective matter, and relates to psychoacoustics. It depends on the semantics of the nodes. We aim at designing an *a priori* quality measure based on parameters of the audio effects, which is more practical to compute in real-time than analysing the audio signal.

The quality measure should be *compositional*, *i.e.* the quality of the graph $q_G \in [0, 1]$ must be a function of the quality of its nodes and edges. The worst quality is 0 and the best quality is 1. For each node v, we also note $q_v \in [0, 1]$ its quality.

We define the quality $q_{v_1 \to \dots \to v_n}$ on a path $v_1 \to \dots \to v_n$ as $q_{v_1} \otimes \dots \otimes q_{v_n}$ for an operator $\otimes$ with the following properties:

Associativity $q_{v_1} \otimes (q_{v_2} \otimes q_{v_3}) = (q_{v_1} \otimes q_{v_2}) \otimes q_{v_3}$

Decreasing $q_v \otimes q_{v'} \leq q_{v'}$ It means that quality never increases on the path, as the information lost by degrading cannot be rebuilt.

Identity element There is an identity element $1_\otimes$ such that $1_\otimes \otimes v = v \otimes 1_\otimes = v$. Such an element is the quality which preserves for the output the quality of its input.

An obvious choice for an operator fulfilling these desired properties is multiplication on real numbers. For $v \to v'$:

$$q_{v \to v'} = q_v \otimes q_{v'} = q_v \times q_{v'}$$

On the path $v_1 \to \cdots \to v_n$, thanks to associativity:

$$q_{v_1 \to \cdots \to v_n} = \prod_{i=1}^{n} q_{v_i} \tag{3}$$

We also define a *join* operator $\oplus$ that models the quality resulting from joining two paths, such as $v_1 \to v_3$ and $v_1 \to v_2 \to v_3$ on Fig. 3.

$$\begin{aligned} q_G &= q_{v_1 \to v_3} \oplus q_{v_1 \to v_2 \to v_3} \\ &= (q_{v_1} \otimes q_{v_3}) \oplus (q_{v_1} \otimes q_{v_2} \otimes q_{v_3}) \end{aligned} \tag{4}$$

In practice, we choose $\bigoplus_{k=1}^{n} q_k = \frac{1}{n} \sum_{k=1}^{n} q_k$ for n joining paths for mixer-like nodes and $\oplus = \min$ for the other nodes.

4. OPTIMIZATION BY RESAMPLING

We consider a method of degrading the quality which is agnostic of the actual computation node semantics, as it operates on the signal that flows in-between the nodes by resampling parts of the audio graph. The rationale for resampling is based on the following cognitive observations. Above some frequency threshold, no quality improvement is perceived by human beings, according to psychoacoustics studies [19]. Indeed, the human auditory system cannot perceive frequency above 20 kHz. Shannon's theorem implies that the sampling rate must be at least double of the maximum frequency, so about the sampling rate of audio CDs of 44.10 kHz. However, oversampling makes it possible to better handle rounding errors in the signal processing and that is why we also consider frequencies higher than 44.10 kHz.

4.1. Resampling a signal path

4.1.1. Inserting resampling nodes

In a dataflow graph, nodes are fed with samples. The premise here is that a node which receives less samples will take less time to be executed. Changing the number of samples per time unit is a common operation in signal processing, called *resampling*: getting less is *downsampling*, and more, *upsampling*. In order to resample, we insert resampling nodes in the graph. These nodes have the same attributes and properties as other nodes: they can interpolate between samples, copy samples in audio buffers. They have a quality measure and temporal characteristics such as a ACET or WCET, so that we can take into account the overhead due to inserting those nodes. Every resampling node has one input port and one output port.

When a downsampling node is inserted on a path, all the following nodes in the path operate on a downsampled signal. We need to insert an upsampling node if there is a node on that path that enforces a specific samplerate, typically, a sink to the soundcard.

We consider a path $\pi = v_1 \to \cdots \to v_n$ where for a node v_k in path π, $v_k = (\{p^i_{k,j}\}, \{p^o_{k,j}\})$ with p^{ki}_j the input ports and p^{ko}_j the output ports. Thus, the degraded path π' is $\pi' = v_1 \to v'_1 \to \cdots \to v'_n \to v_n$. We want to insert downsampling node v'_1 just after v_1 and an upsampling node v'_n just before v_n, as shown in Fig. 4. The resampling factor of v'_1 is $r \in \mathbb{Q}$ such that $\mu(p^{'i}_1) = \mu(p^o_1)$ and $\mu(p^{'o}_1) = r \times \mu(p^{'i}_1)$ for all edges between, *i.e.* the input of the resampling node is at the same rate at its incoming node, and its output is at the new rate. We do it in the same way when inserting an upsampler. For a downsampler, $r \leq 1$, and for an upsampler, $r \geq 1$. For all $k \in [\![2, n-1]\!]$, the rates of all input and output ports of node v_k are multiplied by the resampling factor r of v'_1.

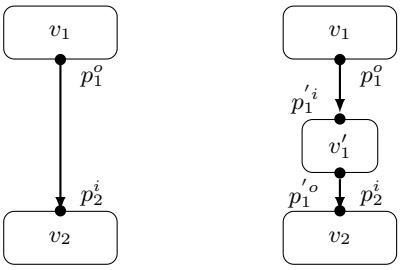


Figure 4: Inserting a downsampling node v'_1 between nodes v_1 and v_2. v_1 has an output port p^o_1, v_2 has an input port p^i_2 and v'_1 has an input port $p^{'i}_1$ and an output port $p^{'o}_1$.

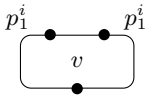


Figure 5: Node v is on path $v_1 \to \cdots \to v_n$. The resampled signal flows on this path through input port p^i_1 with resampling factor q. Node v has another input port, p^i_2. The signal coming into this port must also be resampled with resampling factor r.

In case a node on the path has several input ports and that one of them receives a resampled signal, we also have to resample by the same resampling factor the other input ports, as shown in Fig. 5, by inserting a resampling node connected to this input port.

Also, if an output port p of node v is connected to several input ports $p_1, \ldots, p_n$, it is more efficient to insert the resampler and then a node with n outputs, instead of inserting a resampler on each edge $p \to p_k$, as shown in Fig. 6.

4.1.2. Execution times

We assume that the computation nodes process all their incoming samples and hence, that the complexity of their computations is at least linear. So we can bound execution times of v_k for $k \in [\![2, n-1]\!]$ for a downsampler with resampling factor $1/t$:

$$A'_{v_k} \leq \frac{1}{t} \times A_{v_k} \tag{5}$$

$$W'_{v_k} \leq \frac{1}{t} \times W_{v_k} \tag{6}$$

We can deduce a bound on the whole execution times of the degraded path π':

$$A'_\pi \leq A_{v_1} + A_{v'_1} + \frac{1}{t} \times \sum_{k=2}^{n-1} A_{v_k} + A_{v'_n} + A_{v_n} \tag{7}$$

that is to say for the subpath:

$$A'_\pi \leq A_{v_1} + A_{v'_1} + \frac{1}{t} A_{v_2 \to \cdots \to v_n} + A_{v'_n} + A_{v_n} \tag{8}$$

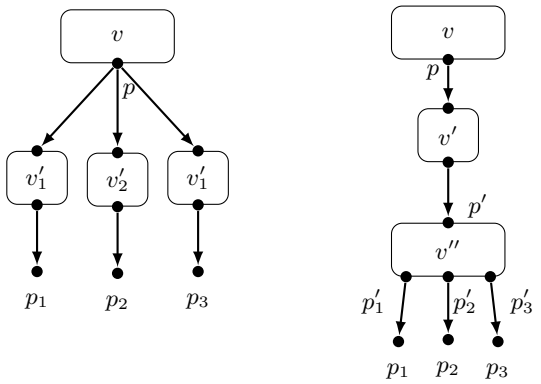


Figure 6: Node v has one output port, p, which is connected to three input ports, p_1, p_2, p_2. On the left, we insert a resampler on each edge $p \rightarrow p_1, p \rightarrow p_2, p \rightarrow p_3$ whereas on the right, we insert a node v'' with one input port p' and 3 outputs p'_1, p'_2, p'_3, and we insert the resampler v' on edge $p \rightarrow p'$.

Note that we take into account the execution times of the inserted nodes, so that the optimization is *overhead-aware*. For small graphs with audio processing nodes with an execution time in the same order of magnitude of the resamplers, the execution time of a degraded graph can be actually larger than the non-degraded one.

4.1.3. Quality

The *a priori* quality measure in the case of resampling is the samplerate: the lower the sampling rate, the lower the quality. In the case of path π, we have $f_{v'_1} = \frac{q \times \mu(p'^{i}_1)}{T_{v'_1}}$. If audio is sent too late to the output buffer, a click can be heard. We consider that it is worse to hear a click because of missing the deadline of the audio driver than to hear a resampled signal. A node that would entail always missing deadlines is given the worst possible quality.The quality q_v of a downsampled node is such that $q_v < 1_\otimes$. The quality of a non-downsampled node is 1.

4.2. Degraded versions

Choosing the nodes to degrade in the audio graph is an optimisation problem under constraints. We can try to maximize the quality of the audio graph given an execution time constraint (a deadline), or minimize the execution time of the audio graph given a minimum target quality.

Our system can enumerate all the possible degraded versions of a graph, or a random sampling of the degraded versions, or use heuristics to compute a useful subset that respects some constraints. The tool is an open source OCaml program[3] that accepts Puredata or MAX/MSP patches, or a custom format as input and can output one optimized version or a set of optimized versions.

4.2.1. Exhaustively

Enumerating all the possible degraded versions of the audio graph is enumerating all the possible subsets of nodes that can be degraded in the audio graph, *i.e.* all nodes except sources and sinks. Then resamplers are inserted so that all the chosen nodes are degraded and the nodes stay isochronous, that is to say have the same sample rate for all input ports and the same sample rate for output ports.

The number of degraded versions is $\mathcal{O}(2^n)$ where n is the number of nodes in the graph and so is impractical when audio graphs start to have a huge number of nodes. When n is large, we randomly choose the possible degraded versions and make sure we have the non-degraded graph and the fully-degraded graph[4] in the set.

4.2.2. Heuristics

The idea is to degrade nodes starting from the outputs, as inserting a downsampling node near the end of the branch impact less nodes than at the beginning.

We choose to have at most one resampled subpath per branch. We also try to minimize the number of inserted nodes with respect to the number of degraded nodes on the resampled subpath, in order to minimize the overhead due to resampling nodes. For that, we need to minimize the number of branches where we resample and that is why we explore the graph branch per branch.

We can use this heuristics to find approximate solutions the optimization problem of maximizing the quality with an execution time constraint. For that, we start from one of the output nodes, and we traverse the graph backwards and see how inserting a downsampling node on a path going to this output node would change the estimated remaining execution time, until the estimation of remaining execution time is lower than the remaining time before the deadline. Other branches can be explored if it is not enough. Then the downsampling and upsampling nodes are inserted on the paths chosen to be resampled.

5. EXPERIMENTAL EVALUATION

In order to evaluate our theoretical models, we need to instantiate our models on a large number of graphs. However, there are no reference benchmarks of audio graphs for IMS. We decided to generate a huge number of graphs, compute the theoretical execution time (in Sect. 3.3) and quality (in Sect. 3.4) of each graph, and then measure the execution time and a quality based on the actual audio signal. We then compare the theoretical values and the measured ones.

The resamplers in use for these experiments are the linear resamplers, and we only resample by 2.

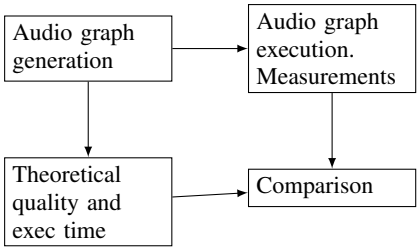


Figure 7: The experimental setup to evaluate the models of quality and execution time.

[3] https://github.com/programLyrique/ims-analysis

[4] The audio graph where all the inner nodes are downsampled. It has the worst possible quality and possibly the shortest execution time.

5.1. Measuring execution time and quality

We developed a open source rust program[5] that can execute the audio graph, monitor execution times of the audio callback, and output the resulting audio. The resampling is handled by libsamplerate,[6]. This is an *opensource* library that makes it possible to downsample down to a 256 ratio, and to upsample up to a 256 ratio. The sampling rate can be changed in real-time. The library provides five resamplers, classified by decreasing order of quality: best, medium, faster sync resamplers (as in [20]), zero order hold resampler, and a linear resampler.

Measuring execution time We execute the audio graph for a large number of cycles and drop the first cycles to take into account processor cache warming.

Measuring quality We compare the audio signal of a degraded graph and the one of the non-degraded graph. Given the spectrum of each signal, we compute their constant-Q transform [21], as we are interested in music signals. We then use a psychoacoustics curve called A-weighting [22] to account for the limited hearing range of human beings (up to 20 kHz). Finally, we compute the distance between the two resulting spectra and normalize it so that a distance of 0 is a quality of 1 and, of $+\infty$, a quality of 0:

$$q_{\mathcal{G}}^{\text{mes}} = \exp(-\|s_{\text{non\_degraded}} - s_{\mathcal{G}}\|) \tag{9}$$

5.2. Comparing the model and the measurements

5.2.1. Graph generation

We generate the structure of the graphs first, *i.e.* vertices and edges, and then pick actual audio processors for each vertex in a node dictionary.

Exhaustive generation for a given number of nodes We enumerate all the non-labelled WCDAGs with n vertices. Non-labelled entails that $a \to b$ and $b \to a$ are isomorphic, are the *same* graphs. Given the set of vertices $V = \{0, \dots, n-1\}$, we undertake the following steps:

1. Compute the set of all the possible edges $\mathcal{E}$ between distinct vertices in one direction. For that, we remark that $\text{pairs}(a_k, \dots, a_n) = \bigcup_{i \in \{k+1,\dots,n\}} \{(a_k, a_i)\} \cup \text{pairs}(a_{k+1}, \dots, a_n)$ It will entail acyclicity.
2. Compute $\mathcal{P}(\mathcal{E})$
3. In a connected graph with n vertices, there are at least $n-1$ edges (*i.e.* a chain graph). So we keep only subsets with more than n edges of $\mathcal{P}(\mathcal{E})$ in our admissible set of set of edges, E.
4. Build the set D of DAGs from E, one graph per subsets.
5. Filter D to remove non weakly-connected graphs, by picking a node and then traverse the undirected version of the graph and counting the vertices. If there are the same numbers as the total number of vertices in the graph, it is weakly connected.

The set of all possible edges from n nodes has size:

$$(n-1) + n - 2 + \dots 1 = \sum_{k=1}^{n-1} k = \mathcal{O}(n^2) \tag{10}$$

Thus the powerset has size $\mathcal{O}(2^{n^2})$. The following operations reduce the number of graphs so we keep that upper bound.

Random generation As the increase in the number of graphs is over-exponential, it becomes untractable when $n > 6$ in practice. For $n = 7$ for instance, there are 3781503 possible DAGs. Hence, for larger number of nodes, we randomly generate graphs using the Erdős–Rényil [23] random graph mode, where a graph can be chosen uniformly at random from the graphs with n nodes and M edges, or with n nodes and a given probability of having an edge between two edges.

From Puredata patches Audio graphs tend to exhibit a particular structure: few incoming and outgoing edges per node, creating long chains in the audio graph, with a few nodes with more inputs that typically mix signals. To take into account this structure, we parsed all the Puredata patches of its tutorial and examples (*i.e.* 133 graphs).

Picking the actual nodes We maintain a database of possible audio processing nodes, with their estimated execution time, their numbers of input ports and output ports, and their possible control parameters. The possible parameter values can be annotated with a range, `[0,1]` for instance for a volume, or a set, for instance `{20,440,1000,2500,6000}` for a set of frequencies.

Given the structure of the graph, for each vertex in it, we can pick (or generate all possible versions) of nodes with the same number input ports as of incoming edges and a number of output ports between 1 and the number of outcoming edges. For the output ports, it is due to *port sharing*, as shown in Fig. 8.

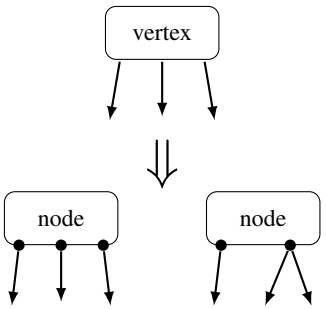


Figure 8: Port sharing entails that a vertex can be replaced by a node with not the same number of output ports, due to port sharing. At the top, it is a vertex with 3 outgoing edges. At the bottom, we show two possible actual nodes from this vertex, one with 3 output ports, and one with 2 output ports and the second port shared by two outgoing edges.

5.2.2. Comparing rankings

After generating the graphs, we can both compute the theoretical qualities and execution times, and measure the actual ones. We obtain two permutations of our set of graphs and we need to know how far the two permutations are.

To compare the similarity of the orderings of the graphs ranked by the theoretical and measured quantities, we use *rank correlation*. The Kendall rank correlation coefficient (Kendall's tau coefficient) [24] is linked to the number of inversions needed to transform one ordering into the other one. The Spearman's rank correlation coefficient (or Spearman's rho) [25] is linked to the distance

[5] `https://github.com/programLyrique/audio-adaptive-scheduling`

[6] `http://www.mega-nerd.com/SRC/`

in positions of a same graph in the two orderings. If the correlation is $+1$, the two orderings are ranked in the same way, *i.e.* the function that transforms one into the other one is monotonic. If it is -1, the order is reversed.

We also compare the position in the two orderings of the worst quality graphs, the shortest execution time and the longest execution time.

5.2.3. Results

Exhaustive enumeration For Fig. 9, we enumerated all the audio graphs with 5 nodes, that is to say, 838 graphs. The average number of versions of a non degraded graph (including the non-degraded graph) is 3.657518 ± 2.066605 and there are 57 graphs without degraded versions. The theoretical models and the measurements both of execution time and quality are well correlated. There are so many correlations in the $0.9 - 1$ bin because many 5 nodes graphs have few degraded versions.

Only 22, *i.e.* 2%, degraded graphs are quicker to execute than their degraded versions here when using nodes with execution time the same order of magnitude as the resamplers. Indeed, in that case, the length of a resampled branch must be large for it to execute quicker than the non-degraded version. However, small graphs do not exhibit long branches. On the contrary, when using nodes an order of magnitude bigger, 275, *i.e.* 33% degraded graphs are quicker than their non-degraded version.

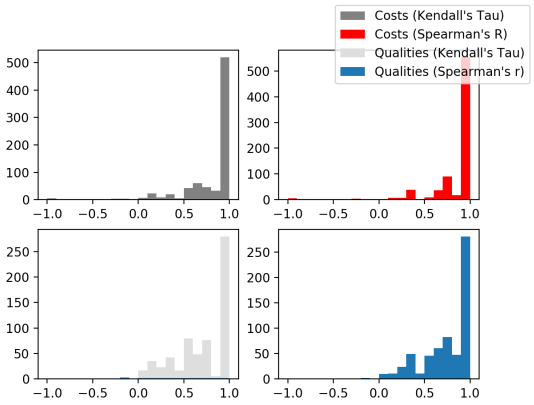


Figure 9: Correlations for an exhaustive enumeration of 5-node audio graphs. At the top, execution times (costs), at the bottom, qualities; on the left, Kendall's Tau, on the right, Spearman's R.

Large random graphs For Fig. 10, we generated 100 random graphs with 10 nodes and a 0.3 edge probability, in order to have nodes with not too many inputs and outputs, as in real audio graphs. The execution time model is rather accurate, with most correlations above 0.5. For the quality, the results are less good, as there are lots of correlations close to 0, but mainly strictly positive. With quick nodes, 15 graphs have degraded versions quicker than their non-degraded versions, whereas with slow nodes, there are 45.

With graphs from Puredata For Fig. 11, we generated 133 audio graphs from 133 Puredata patches. The model of execution time is accurate, which is especially visible with the Spearman correlation coefficient histogram. The model of quality and the measurement of qualities are better correlated than for random

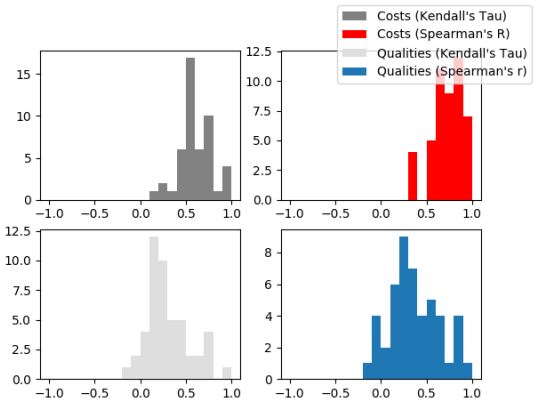


Figure 10: Correlations for random 10-node audio graphs with 0.3 edge probability. At the top, execution times, at the bottom, qualities; on the left, Kendall's Tau, on the right, Spearman's R.

graphs. With quick nodes, 48 graphs, *i.e.* 36%, have degraded versions which are quicker than the non-degraded graph. The structure of Puredata graphs exhibit longer branches and more subsets of the graph that can be resampled with just a few resamplers.

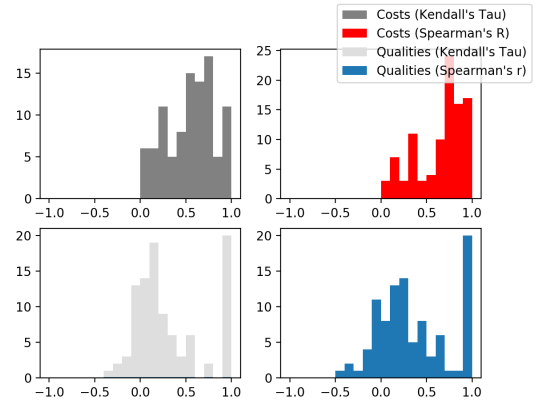


Figure 11: Correlations for audio graphs generated from Puredata patches. Node count can go up to 80. At the top, execution times, at the bottom, qualities; on the left, Kendall's Tau, on the right, Spearman's R.

Hence, large graphs without too many ramifications, and with nodes with execution times at least an order of magnitude higher than the execution time of a resampler take advantage the most of the resampling optimization. Actual audiographs from IMSs, such as the ones from Puredata, have actually these characteristics.

6. CONCLUSION

We proposed a model of an audio graph with execution time and quality, and an optimization algorithm based on the models that can find degraded versions of an audio graph while respecting constraints on execution time or quality. The degradations are based on downsampling parts of the graph.

We evaluated the models and the algorithm on audio graphs that were exhaustively enumerated for graph with few nodes, randomly generated for larger graphs, and generated from Puredata patches. The execution time model is quite accurate. The quality model is well correlated for small graphs (Fig. 9), has to be improved on large random graphs (Fig. 10) and promising results are obtained for graphs (Fig. 11) generated from real Puredata patches. The optimization is particularly effective for actual audio graphs with long audio effects and long branches. Our choice of measures of quality is arbitrary. Another option would be to organize listening tests, but would be impractical considering the huge size of the set of possible graphs and possible degraded graphs.

The optimization program can be integrated to the design of an audio graph, using our custom audio graph format that describes nodes and their connections as input and output and can import Puredata patches but we aim at making it easier to use within MAX/MSP and Puredata. We also want to integrate our optimization as a compiler optimization for Faust programs in the Faust compiler [26]. It would unlock access to many more different audio graphs. As the Faust instructions are both very fine-grained and well-defined, we would be able to refine the execution time and the quality models.

7. ACKNOWLEDGMENTS

This work started during a visit in the group of Prof. Christoph Kirsch in Salzburg, whom we would like to thank for his advice. We also would like to thank the anonymous reviewers for their insights and advice.

8. REFERENCES

[1] Robert Rowe, *Interactive Music Systems: Machine Listening and Composing*, AAAI Press, 1993.

[2] M. Puckette, "Using pd as a score language," in *Proc. Int. Computer Music Conf.*, September 2002, pp. 184–187.

[3] David Zicarelli, "How I learned to love a program that does nothing," *Comput. Music J.*, vol. 26, no. 4, pp. 44–51, 2002.

[4] G. Wang, *The ChucK audio programming language." A strongly-timed and on-the-fly environ/mentality"*, Ph.D. thesis, Princeton University, 2009.

[5] James McCartney, "Supercollider: a new real time synthesis language," in *Proceedings of the International Computer Music Conference*, 1996.

[6] José Echeveste, Arshia Cont, Jean-Louis Giavitto, and Florent Jacquemard, "Operational semantics of a domain specific language for real time musician–computer interaction," *Discrete Event Dynamic Systems*, vol. 23, no. 4, pp. 343–383, 2013.

[7] Pierre Donat-Bouillud, Jean-Louis Giavitto, Arshia Cont, Nicolas Schmidt, and Yann Orlarey, "Embedding native audio-processing in a score following system with quasi sample accuracy," in *ICMC 2016-42th International Computer Music Conference*, 2016.

[8] Saamer Akhshabi, Ali C Begen, and Constantine Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http," in *Proceedings of the second annual ACM conference on Multimedia systems*. ACM, 2011, pp. 157–168.

[9] Dario Faggioli, Fabio Checconi, Michael Trimarchi, and Claudio Scordino, "An edf scheduling class for the linux kernel," in *Proceedings of the Eleventh Real-Time Linux Workshop*. Citeseer, 2009.

[10] T. Blechmann, "Supernova-a multiprocessor aware real-time audio synthesis engine for supercollider," M.S. thesis, Vienna University of Technology, 2011.

[11] Edward A Lee and David G Messerschmitt, "Synchronous data flow," *Proceedings of the IEEE*, vol. 75, no. 9, pp. 1235–1245, 1987.

[12] Swagath Venkataramani, Srimat T Chakradhar, Kaushik Roy, and Anand Raghunathan, "Computing approximately, and efficiently," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*. EDA Consortium, 2015, pp. 748–751.

[13] Zeyuan Allen Zhu, Sasa Misailovic, Jonathan A Kelner, and Martin Rinard, "Randomized accuracy-aware program transformations for efficient approximate computations," in *ACM SIGPLAN Notices*. ACM, 2012, vol. 47, pp. 441–454.

[14] Jane WS Liu, Kwei-Jay Lin, Wei Kuan Shih, Albert Chuang-shi Yu, Jen-Yao Chung, and Wei Zhao, *Algorithms for scheduling imprecise computations*, Springer, 1991.

[15] Pontus Ekberg and Wang Yi, "Schedulability analysis of a graph-based task model for mixed-criticality systems," *Real-time systems*, vol. 52, no. 1, pp. 1–37, 2016.

[16] Edward A Lee and Thomas M Parks, "Dataflow process networks," *Proceedings of the IEEE*, vol. 83, no. 5, pp. 773–801, 1995.

[17] Pau Arumí and Xavier Amatriain, "Time-triggered static schedulable dataflows for multimedia systems," in *Multimedia Computing and Networking 2009*. International Society for Optics and Photonics, 2009, vol. 7253, p. 72530D.

[18] Oana Andrei and Hélene Kirchner, "A rewriting calculus for multigraphs with ports," *Electronic Notes in Theoretical Computer Science*, vol. 219, pp. 67–82, 2008.

[19] Stuart Rosen and Peter Howell, *Signals and systems for speech and hearing*, vol. 29, Brill, 2011.

[20] Julius O Smith and Phil Gossett, "A flexible sampling-rate conversion method," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'84*. IEEE, 1984, vol. 9, pp. 112–115.

[21] Judith C Brown, "Calculation of a constant q spectral transform," *The Journal of the Acoustical Society of America*, vol. 89, no. 1, pp. 425–434, 1991.

[22] Brian CJ Moore, *An introduction to the psychology of hearing*, Brill, 2012.

[23] Paul Erdős and Alfréd Rényi, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci*, vol. 5, no. 1, pp. 17–60, 1960.

[24] Maurice George Kendall, "Rank correlation methods.," 1948.

[25] Wayne W Daniel et al., *Applied nonparametric statistics*, Houghton Mifflin, 1978.

[26] Yann Orlarey, Dominique Fober, and Stephane Letz, "Syntactical and semantical aspects of faust," *Soft Computing*, vol. 8, no. 9, pp. 623–632, 2004.

A REAL-TIME AUDIO EFFECT PLUG-IN INSPIRED BY THE PROCESSES OF TRADITIONAL INDONESIAN GAMELAN MUSIC

Luke M. Craig

Department of Computer Science
Appalachian State University
Boone, NC, USA
lukemcduffiecraig@gmail.com

R. Mitchell Parry

Department of Computer Science
Appalachian State University
Boone, NC, USA
parryrm@appstate.edu

ABSTRACT

This paper presents Gamelanizer, a novel real-time audio effect inspired by Javanese gamelan music theory. It is composed of anticipatory or "negative" delay and time and pitch manipulations based on the phase vocoder. An open-source real-time C++Virtual Studio Technology (VST) implementation of the effect, made with the JUCE framework, is available at github.com/lukemcraig/DAFx19-Gamelanizer, as well as audio examples and Python implementations of vectorized and frame by frame approaches.

1. INTRODUCTION

Gamelan music is the traditional court music of Indonesia, featuring an orchestra of pitched-percussion instruments. It has had a profound influence on Western composers, most famously on Debussy [1, 2]. Gamelanizer is a real-time audio effect plug-in inspired by the music theory of the Javanese variety of gamelan music.

This paper is organized as follows: Section 1.1 gives background information on gamelan music theory. Section 1.2 compares Gamelanizer to similar works and explains the motivation for creating it. Section 2 details our method for achieving this effect as a real-time Virtual Studio Technology (VST). Section 3 details our results. Section 4 discusses the considerations of our user interface, potential applications, and future work.

1.1. Background

There is a large variety of gamelan music theory. Our work is influenced specifically by the Javanese variant because it is arguably the most rule-based [3]. Specifically, we are concerned with the process known as *Mipil*, which translates to "to pick off one by one" [4]. The following is a description of the *Mipil* process. Because there is still debate in the ethnomusicology community concerning gamelan music, we stress that this description is an oversimplification. However, for our audio effect, this oversimplified understanding is still useful.

The numeric notation system used for gamelan music only indicates the *balungan* (melodic skeleton) that the *saron* instrument plays. This is because the notes that the *bonang barung* and *bonang panerus* instruments play can be generated, to a degree, from this base set [5]. In a hypothetical piece, if the first *balungan* notes are those in Table 1 then the elaborating *bonang barung* musicians will know to play the notes in Table 2. However, their notes will be played twice as fast as the *saron* and will occur half a beat earlier.

Table 1: *The balungan, or base melody.*

saron	3	5	2	6

Table 2: *The first level of subdivision.*

bonang barung	3	5	3	5	2	6	2	6

Therefore, the fourth note that the *bonang barung* plays will coincide with the second note that the *saron* plays. The eighth note of the *bonang barung* will coincide with the fourth note of the *saron*. The *bonang panerus*, which is an octave higher than the *barung*, repeats this subdivision process, now playing the notes in Table 3 twice as fast as the *bonang barung* and four times as fast as the *saron*. The combination of these three interlocking parts is shown in Table 4.

1.2. Motivation

There has been previous work related to gamelan and digital audio. Some have focused on synthesizing the unique harmonic structure of the instruments in a gamelan orchestra [6]. Others have tried to infer compositional rules by using real compositions [7]. Others have noted the algorithmic nature of some types of gamelan music [3]. Various computer-aided composition tools have been developed that apply the processes of gamelan music theory to numbered musical notation [8–10].

There is potential to extend concepts from gamelan music theory to audio signals, rather than note numbers, for music production and sound design. Given a digital audio workstation (DAW) with standard time and pitch modification tools, an audio engineer or musician could manually apply the process known as *Mipil* (see Section 1.1) to audio clips. We tested the suitability of the effect by performing this manual operation in a variety of musical mixing contexts. The results were intriguing. While it is a difficult effect to characterize, it is probably most similar to granular delay effects [11], specifically ones that can operate on large grain sizes.

Performing this process manually with the editing tools of a DAW is slow and can become monotonous. Furthermore, if a producer decides to alter a single note in the source, that change must be propagated down the chain of subdivision levels. It is hard to maintain perspective in this situation. Additionally, the process could only be applied to audio clips and thus is not insertable at arbitrary points in the signal chain without rendering audio. If audio

Table 3: *The second level of subdivision.*

bonang panerus	3	5	3	5	3	5	3	5	2	6	2	6	2	6	2	6

Table 4: *All three instruments.*

saron					3				5				2				6					
bonang barung			3		5		3		5		2		6		2		6					
bonang panerus		3	5	3	5	3	5	3	5	2	6	2	6	2	6	2	6					

has to be rendered, the non-destructive editing workflow of digital audio production is lost.

This paper proposes a solution to achieve this digital audio effect without the arduous and creatively limiting manual work. Specifically, a real-time VST plug-in. The effect has three responsibilities:

- time-scaling the quarter notes into eighth, sixteenth thirty-second, and sixty-fourth notes,
- pitch shifting the notes by a user defined amount, and
- placing and repeating the altered notes correctly in time, including earlier than the input signal.

2. METHOD

It is easiest to think of the problem from the perspective of a quarter note (hereafter referred to as a beat). Each beat needs to be pitch shifted and time scaled by the correct factor for each subdivision level and placed and repeated correctly. Then each level, including the base, would need to be delayed by the summation of the duration of one note of each of the levels that are higher pitched than it. Then latency compensation would be used to realign the base level with the input signal and place the subdivision levels earlier than the input.

While this naive method is relatively straightforward, it is not usable for a real-time plug-in. For one audio callback, the processing time of the naive method is often smaller than our proposed method. However, the naive method occasionally has too long of a processing time during a single audio callback, which causes audio dropouts from buffer underflow. Considering the different general audio effect frameworks of [12], the naive method is like a "frame by frame" approach that treats an entire beat as a frame, rather than the usual case of the block size of the audio callback. Our method, outlined in Figure 1, is a "frame by frame" approach that instead treats the much smaller fast Fourier transform (FFT) frames as the fundamental unit of work. In this manner, the computational load is more evenly distributed. We consider the process from the perspective of an individual sample:

1. For each sample, we pass it to each of the subdivision level processors (Section 2.1). These independently perform time compression and pitch shifting through the use of the phase vocoder technique (Section 2.2). They also handle duplicating the notes in the correct positions (Section 2.3).
2. We also copy each sample to a delay buffer for the base level and use latency compensation to make part of the output signal occur earlier than the input signal (Section 2.4).
3. We then update the subdivision levels' write positions if the play head in the DAW would be on a beat boundary (Section 2.5).

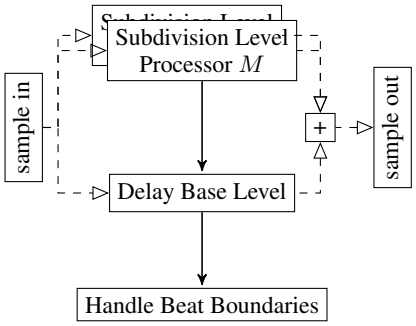


Figure 1: *An overview of our method.*

2.1. Subdivision level processors

The bulk of the work on each input sample is done by each of the subdivision level processors. Each subdivision level processor in the plug-in corresponds to the role of an elaborating instrument in the gamelan orchestra. The pitch shifting parameter of a subdivision level in the plug-in corresponds to the *tessitura* (normal pitch range) of an elaborating instrument. The time scaling factors of a subdivision level in the plug-in correspond to the rhythmic densities of an elaborating instrument.

In each subdivision level processor i, where $i \in \{1, 2 \ldots, M\}$, we perform time compression and pitch shifting via the classical phase vocoder technique [13]. Then we overlap-and-add (OLA) the synthesis frames multiple times in an output buffer channel associated with the subdivision level.

2.2. Phase vocoding

The phase vocoding consists of two stages: pitch shifting by resampling and time scaling. The effective time scaling factors $\mathbf{r}$ of the subdivision levels are

$$r[i] = \frac{1}{2^i}\,. \tag{1}$$

Given pitch shift amounts $\mathbf{c}$, in cents, the frequency scaling factors $\mathbf{p}$ are

$$p[i] = 2^{\frac{c[i]}{1200}}\,, \tag{2}$$

and the actual time scaling factors $\mathbf{v}$ applied after resampling are

$$v[i] = r[i]p[i]. \tag{3}$$

In other words, if the pitch shift amount is an octave (1200 cents), then no actual time scaling needs to be done because the resampling operation of pitch shifting will result in a note length that is $\frac{1}{2}$ the original beat length.

The frame size of the time scaling operation is fixed and independent of $p[i]$ and $v[i]$. If $v[i] \geq 1$, as is the case in the usual operation of our plug-in, the number of samples output from the resampler will be less than the number input to it. Therefore, by ordering the resampler first in the phase vocoders, we reduce the number of analysis-to-synthesis frame conversions that need to be performed per unit of time.

2.2.1. Pitch shifting by resampling

We use a stateful interpolator to accomplish resampling.[1] It needs to be stateful otherwise there would be discontinuities between blocks of data. The phase vocoders operate on overlapping frames that are one analysis hop size, h_a, samples apart

$$h_a = \frac{N}{o_a}\,, \tag{4}$$

where N is the length of the FFT and o_a is a constant analysis overlap factor.[2] Therefore, we always request h_a samples of output from the resampler. Because it is stateful and will be operating at subsample positions, the number of samples consumed by the resampler will vary each time. Therefore, we push the input samples onto a first-in, first-out (FIFO) queue and only pass them to the resampler when there is at least the number needed by the resampler available on the queue. After the resampler finishes, we pop the number of input samples it used off the queue. We then put the h_a samples of resampled audio data into a circular buffer of length N. The buffer is filled one hop at a time and we do not read from it until it is full. Each time we read from it, we will convert the analysis frame it contains to a synthesis frame.

2.2.2. Converting an analysis frame to a synthesis frame

Converting analysis frames to synthesis frames and OLA are the core elements of the phase vocoder technique, which has been written about extensively [13–21]. Normally, the phase vocoder is presented as operations done on a short-time Fourier transform (STFT) matrix [21]. For our real-time implementation, we instead follow the approach of only considering the current and previous length N overlapping frames of audio samples [18, Sec 5.3]. We also follow the common approach of analysis and synthesis windowing with a Hann window of length N [21]. Because we know when the beat boundaries occur, we can take advantage of the scheme proposed in [16, p. 327] of initializing the phases with the first analysis frame of each new beat. To scale the phases of the current frame each subdivision level processor only needs to store the scaled phases of the previous frame, the unmodified phases of the previous frame, and the synthesis overlap factor of the subdivision level i:

$$o_s[i] = \frac{o_a}{v[i]}\,. \tag{5}$$

2.3. Adding synthesis frames to output buffers

The number of samples $s[i]$ in one note of subdivision level i, is given by:

$$s[i] = \frac{s_b}{2^i}\,, \tag{6}$$

s_b being the number of samples per beat in the base level:

$$s_b = \frac{60 f_s}{t}\,, \tag{7}$$

where f_s is the sample rate of the DAW in Hz and t is the tempo in BPM. Figure 2, which we will refer to throughout this section, shows a hypothetical situation with a tempo of 80 BPM and a sample rate of 44.1 kHz. On the right side of Figure 2, $s[i]$ and s_b are visualized by the lengths of the final "D" notes of each subdivision level and the delayed base.

There are M subdivision levels and each one has a channel in the M channel circular buffer $\mathbf{B}$. When playback is begun from sample 0 in the DAW, we initialize $\mathbf{w}$, the lead write head positions in $\mathbf{B}$, with:

$$w[i] = 2s_b + s[M] + \sum_{j=i+1}^{M} s[j]\,, \tag{8}$$

so that the subdivision level M, which has the highest rhythmic density, will have the earliest position in time. Additionally, the beginning of subdivision level $M-1$ will occur with the beginning of the second note of subdivision level M. The initial values of $\mathbf{w}$, each at the beginning of the first note of their associated subdivision level, can be seen on the left side of Figure 2c.

We OLA the N samples from the synthesis frame $\mathbf{u}$ to $\mathbf{B}$, repeating the appropriate number of times for the subdivision level i. This is shown in Algorithm 1. To simplify the notation, the modular indexing of the circular buffer is ignored.

Algorithm 1 OLA while duplicating notes for subdivision level i

1: **for** $j \leftarrow 0$ to 2^i **do** ▷ note number j
2: $\quad w_j \leftarrow w[i] + j(2s[i])$
3: $\quad \mathbf{B}[i, w_j : w_j + N] \mathrel{+}= \mathbf{u}$
4: **end for**

In line 2 of Algorithm 1, the offset $j(2s[i])$ specifies that we skip a note every time we duplicate a synthesis frame $\mathbf{u}$. In Figure 2, the play head in the DAW (the blue triangle at sample 33075 in Figure 2a) has just reached the end of beat "A." The diagonal shading in Figures 2b and 2c shows the data that has been written to the output buffers at this time. Algorithm 1 is what creates the fractal-like pattern of these shaded regions. The gaps between the shaded regions will be filled, one overlapping synthesis frame of length N at a time, as the play head in the DAW moves over the next beat, "B."

Next, the lead write position of the subdivision level i is incremented by its synthesis hop size $h_s[i]$

$$w[i] \leftarrow w[i] + h_s[i]\,, \tag{9}$$

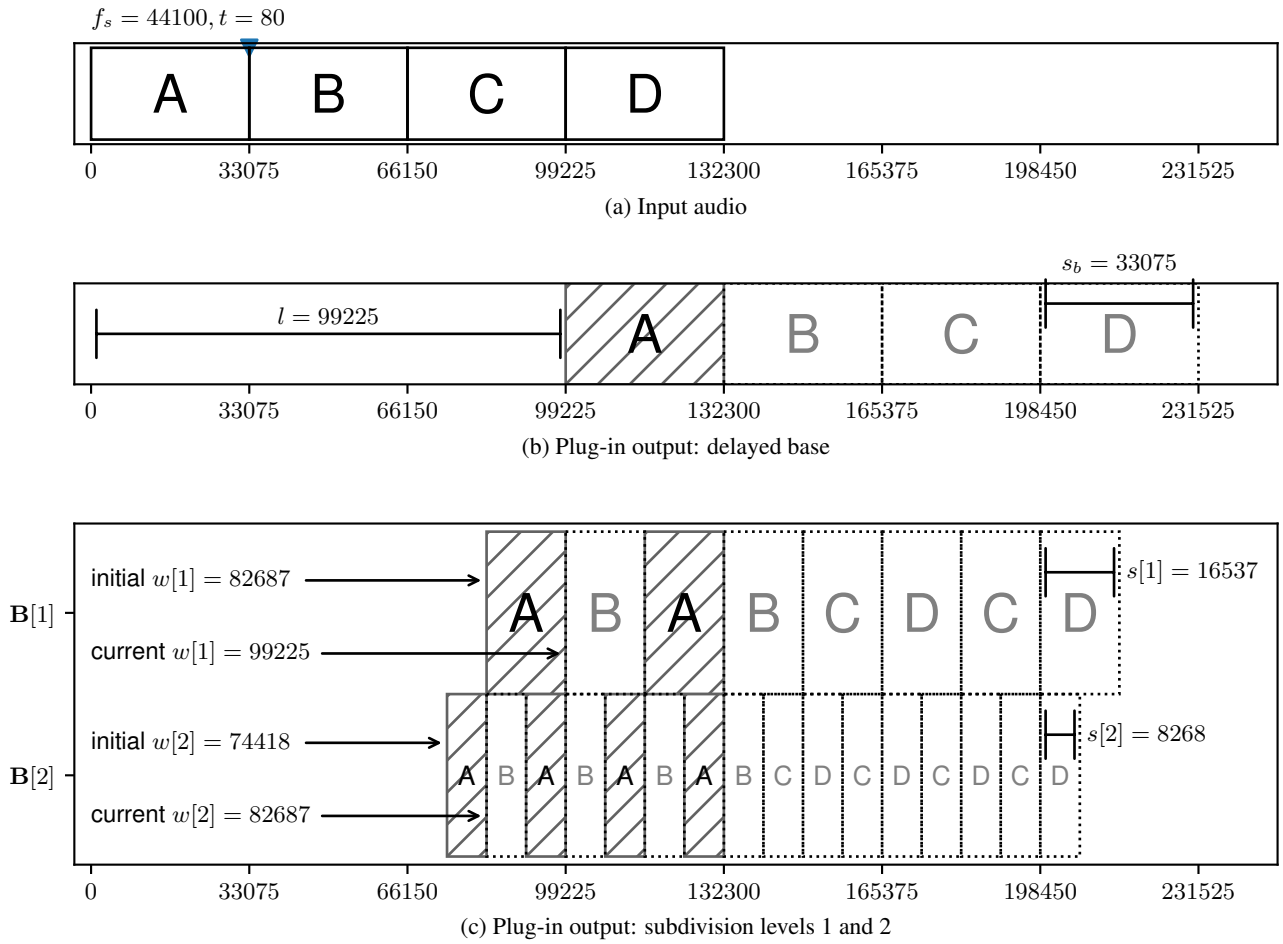
following the standard phase vocoder technique [13]. The synthesis hop size, the number of samples between the beginnings of overlapped synthesis frames, is determined by the analysis hop size h_a and the actual time scaling factor $v[i]$ of the subdivision level i:

$$h_s[i] = h_a v[i]\,. \tag{10}$$

In Figure 2, the write positions have already been incremented many times by equation (9). Their current values, each at the end of the first shaded note for an associated level, are displayed on the left side of Figure 2c.

[1] We implemented this resampling with the CatmullRomInterpolator class from the JUCE framework.

[2] An overlap amount of 75% is suggested in [14] so we set o_a to 4.



(a) Input audio

(b) Plug-in output: delayed base

(c) Plug-in output: subdivision levels 1 and 2

Figure 2: *The output of the plug-in without latency compensation. The play head in the DAW has just reached the beginning of the beat labeled "B" in the input audio (a). The x-axis tick marks indicate the sample positions of the beat divisions at a tempo of 80 BPM. In (b) and (c), the diagonally shaded regions show the data that has been written at this point. The unshaded regions with faded text show where data will be written as the play head progresses. When latency compensation takes effect the base level output (b) will realign with the input (a) and the subdivision level outputs (a) will occur earlier than the input.*

2.4. Delaying the base level

To place time-compressed versions of the entire first beat before the first beat begins in the DAW, we need to request a latency compensation amount that is larger than the initial positions of the lead write heads. Then we need to delay the base level by this same amount to realign it with the input signal and the rest of the tracks in the DAW. The plug-in needs to request l samples of latency compensation from the DAW:

$$l = 3s_b\,, \tag{11}$$

which is also the number of samples we delay the base level by. Given an input of four beats from the DAW, refer to Figure 2 to see what the output from the plug-in would be before latency compensation is applied. The area marked l on Figure 2b shows how delaying the base level aligns it with the subdivision levels in Figure 2c. Keeping this relative alignment and requesting l samples of latency compensation would realign the base level with the input audio (2a) and each subdivision level would begin before the input, as we desire.

2.5. Handling beat boundaries

Whenever the play head of the DAW moves into a new beat, we adjust the lead write heads (Section 2.5.1). If the beat we just finished processing was the second beat in a pair, we adjust the lead write heads further (Section 2.5.2). We also reset the phase vocoders now because the next piece of audio input should be from a new note with unrelated phases, as mentioned in Section 2.2.2.

2.5.1. Adjust lead write heads

When beginning each new beat, we reset $\Delta w[1] \ldots \Delta w[M]$, the number of samples each lead write head has moved during a beat, to zero. Every time a lead write head position $w[i]$ is incremented

with equation (9), we accumulate that change:

$$\Delta w[i] \leftarrow \Delta w[i] + h_s[i]\,, \tag{12}$$

where $0 \leq \Delta w[i] \leq s[i]$. Using $\mathbf{\Delta w}$, when the play head of the DAW[3] is on a beat boundary (for example 33075 or 66150 in Figure 2a), we move all the lead write head positions, $\mathbf{w}$, forward a small amount to the exact starting positions of their next subdivided notes:

$$w[i] \leftarrow w[i] + (s[i] - \Delta w[i])\,. \tag{13}$$

This is necessary because their normal movements are not synchronized, due to the different resampling factors. Moving the write heads like this may leave a small gap between notes. However, it will never cause a "click" artifact from a sample-discontinuity because we are only ever adding windowed and complete synthesis frames.

2.5.2. Handling second beats

Every time we transition to a new beat we also determine if we were on the second beat of a pair. In Figure 2a, these are the beats labeled "B" and "D." If we are finishing with the second beat, then we increment each lead write head position to the starts of their next unwritten regions:

$$w[i] \leftarrow w[i] + s[i](2^{i+1} - 2)\,. \tag{14}$$

Each level has 2^{i+1} notes per pair of beats. After processing the data for a pair of beats, the lead write head of a subdivision level will be two note-lengths, $2s[i]$, past their initial position. That is why we subtract two in equation (14).

For example, as the play head in the DAW reaches the end of beat "B," at sample 66150 in Figure 2a $w[2]$ in Figure 2c will be at the beginning of the second "A" note of level 2, at sample 90956. Accordingly, by moving six notes ahead it will be at the beginning of the first "C" note in level 2, at sample 140568. Due to these increments and the duplication of notes, the plug-in always maintains an equal output rate with the input rate, even though we are consistently time-compressing.

2.6. Starting playback from arbitrary timeline positions

If the user of the DAW starts playback from some arbitrary position, we must determine what the values of $\mathbf{w}$ would be at this position, had the user started from the beginning of the session. Likewise, the user could also change playback positions without stopping the DAW, so we have to check for that as well. Instead of writing a mathematical function to determine the correct values of $\mathbf{w}$, we instead choose to simply run through the main processing method until reaching the play head position of the DAW, skipping any costly operations unrelated to updating $\mathbf{w}$. This way, the software is more flexible to change because we do not have to keep remodeling the behavior.

[3]From the perspective of a plug-in, the reported play head position from the DAW is incrementing an entire block size at a time, rather than sample by sample. Therefore, we have to calculate the expected position of the play head internally, for every sample that we process.

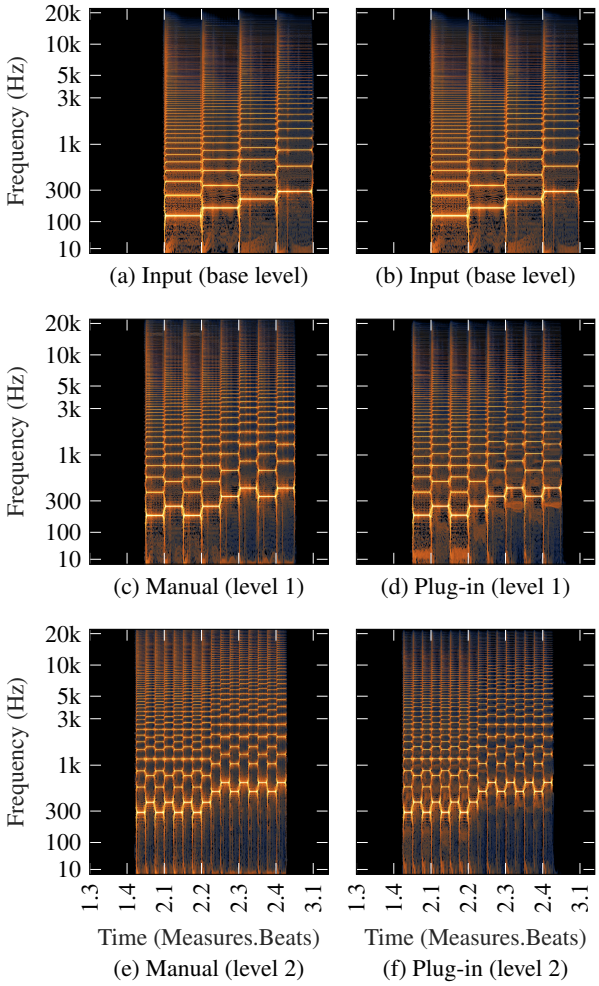
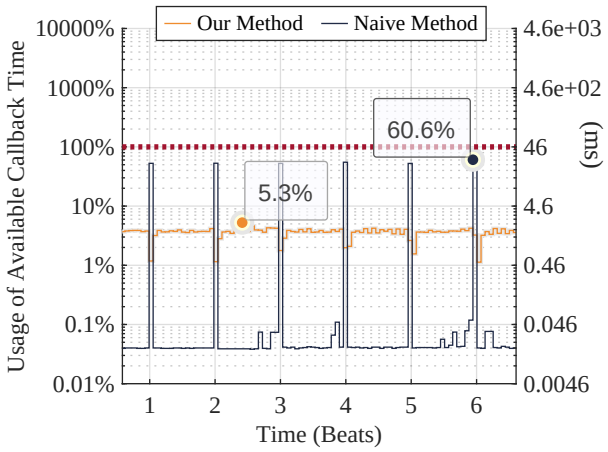


Figure 3: *Spectrograms comparing the outputs of the manual process (our desired result made with editing tools of a DAW) and the plug-in. The input signal (a and b) is four ascending quarter notes at 120 BPM. The first note of the input signal begins on beat 1 of measure 2.*

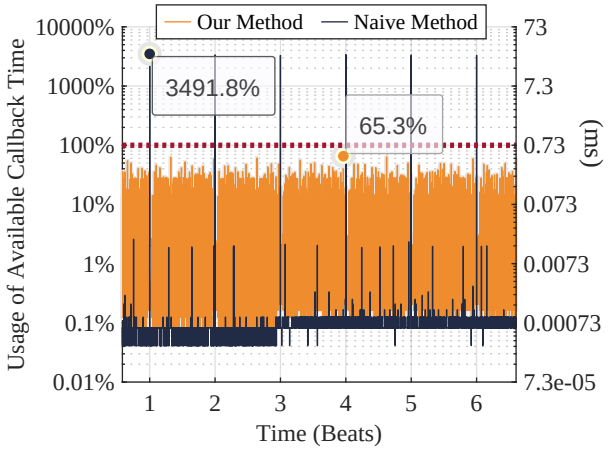
3. RESULTS

Figure 3 shows the output of the manual process, as described in Section 1.2, compared to the output of our real-time plug-in, using the same input signal. One can see the results are similar. Our method successfully time-compressed and placed all the notes in their correct places, including before the input signal. For level 1, the pitch shift factor was seven semitones (700 cents) and, as can be seen on the spectrogram of the plug-in output (3d), it matches the manual output (3c) sufficiently. For level 2, the pitch shift factor was fourteen semitones (1400 cents) and the plug-in output (3f) matches the manual output (3e) sufficiently again.

Figure 4 shows the performance measurements of the naive algorithm that is described in the beginning of Section 2 and our method. The measurements were collected with the plug-in running four subdivision levels with an FFT length of 1024 samples for each phase vocoder. Tests were conducted with block sizes of 32 and 2048 samples at a sample rate of 44.1 kHz. In both



(a) Block size: 2048 samples



(b) Block size: 32 samples

Figure 4: *Performance measurements, relative to available callback time, of four subdivision levels with pitch controls set to stacked fourth and an FFT length of 1024 samples. The tempo was 80 BPM. The four annotated points indicate the maximum time used for processing in the 32 bars that were measured for each test. Points above the dashed line are unacceptable.*

cases, the naive method has dramatic increases in processing time on the beat boundaries. The amount of samples being processed on these boundaries is independent of block size. In this example, 33075 samples are being processed at the beat boundaries because the tempo is 80 BPM. When the block size is 2048 samples, the plug-in has no more than 46 milliseconds (ms) to process these samples, so the naive method is still usable if no other plug-ins are running. However, when the block size is 32 samples, the plug-in has no more than 0.73 ms to do the same amount of work and so it causes buffer underflow and is unusable. Additionally, if the session tempo were slower, even a block size of 2048 may result in buffer underflow due to the increased number of samples per beat.

Our method, in both block size cases, has much better performance. The processing load is decoupled from the number of samples per beat and so there are no longer dramatic time increases at the beat boundaries. In fact, there are decreases in processing time at the beat boundaries because the initialization scheme we use does not alter the first frame of each beat. For the resampling stage of our processor (Section 2.2.1), the processing load scales with the block size, which is good. However, for the other part of the phase vocoding technique (Section 2.2.2), the processing load is independent of the block size, scaling instead with the FFT frame size. Still, because the load is independent of the number of samples per beat, lowering the tempo will not cause the block size of 32 samples to become unusable. If better performance for small block sizes was needed, reducing the FFT frame size would help, at the expense of audio quality.

4. DISCUSSION

4.1. Accompanying Material

The accompanying repository is available at `https://github.com/lukemcraig/DAFx19-Gamelanizer`. Included is the `C++` code and VST builds for macOS and Windows. Audio examples of input and output are also included. Python implementations of vectorized and frame by frame approaches are also available in the repository.

The VST has only been tested to work in the DAW REAPER. Because of the necessity of unusually large amounts of latency compensation, DAWs that limit the amount of latency compensation a plug-in can request will not be able to use the effect, unless the user is comfortable with the subdivision levels occurring after the input. The user could render the output and then drag it into place, which would still be faster than doing it manually.

Another limitation is that the tempo cannot change. Likewise, if the input audio is not perfectly quantized to the grid, the output may sound random, in terms of the rhythmic context. This is because an instrument's note that begins before the beat will have the portion of the signal before the beat treated as if it were the end of the previous note and be moved and repeated in that manner.

4.2. User interface

The user interface is shown in Figure 5. The tempo in the DAW can be specified by the user in the text box at the top. This box is unchangeable during playback. The leftmost vertical slider controls the gain of the delayed input signal (the base level). Sometimes in gamelan music the *balungan* is not played by any instrument but merely implied by the elaborating instruments [22]. Likewise, it can be useful for the user to completely remove the input signal by pressing the mute button (labeled "M"), and only output the subdivision levels. This works well if the first subdivision level has no pitch-shifting applied to it.

Each subdivision level has an identical set of controls that are grouped by a thin rectangle. These can be thought of as channel strips on a mixing console. The largest rotary slider controls the pitch shift factor of that level. The long gray lines inside the rotary arc indicate the octave positions. The small ticks inside the arc indicate the positions of perfect fourths. The small ticks outside the arc indicate the positions of perfect fifths. The user can optionally snap to any of these three intervals by holding different modifier keys while dragging. Changes to pitch shifting are allowed during playback but do not take effect until the subdivision level processor that is being changed is between analysis frames.

Each subdivision level processor also has a gain slider and high and low-pass filters that are applied after pitch-shifting. They

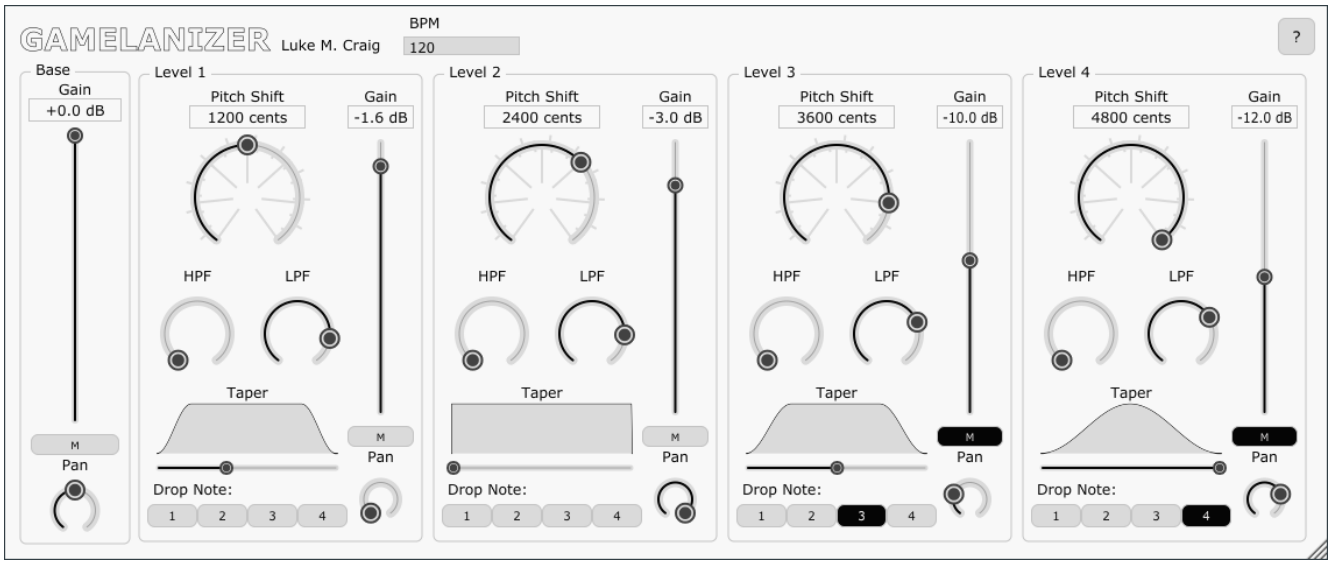


Figure 5: *The Gamelanizer GUI with four subdivision levels. Pitch shifting controls are set to stacked octaves.*

each also have a horizontal slider that specifies the α parameter of a Tukey window [23] that is applied to the input signal before phase vocoding. This can be useful for tapering the input signal to achieve different attack and release characteristics per level. The displays above the horizontal sliders show the resultant Tukey windows.

When any of the four buttons labeled "Drop Note" are toggled, those notes become muted. For instance, if the fourth note was dropped from the fourth subdivision level, then every fourth sixty-fourth-note would become a rest.

Each subdivision level can be output independently or mixed into stereo. By routing them independently the user has more control and can apply other plug-ins and routing to mix them as they wish. When mixed into stereo the pan knobs below the mute buttons can be used.

4.3. Applications

The plug-in is useful for at least two general scenarios. First is the scenario of a composer or musician who is writing new music. If a composer was writing a base melody that they wanted the *Mipil* process to be applied to, they could use the Gamelanizer to get almost instant feedback and make adjustments. In this scenario the plug-in could be considered a computer-aided composition tool similar to those mentioned in Section 1.2. The advantage of using Gamelanizer over those tools is that the composer can transform any sound, such as a human voice, rather than just transforming Musical Instrument Digital Interface (MIDI) input to virtual instruments. Another advantage is that many tech-savvy musicians today compose purely by ear, and using our plug-in would be more intuitive for them than thinking about symbolic representations of notes. For this use case, we suggest starting with only the first and second subdivision levels because higher levels can sound irritating.

Second is the scenario of an audio engineer mixing someone else's music, or creating strange effects for sound design. In this scenario the plug-in is more of an effect that alters the character of the signal without altering the tonal harmony of the music. We recommend pitch shifting with octaves or no pitch shifting to achieve this use case. Filters are useful for adding depth by moving subdivision levels into the background. This can help perceptually separate the effect from the dry signal. Tapering is also useful here for imposing a rhythmic structure on an input signal that does not have one. The third and fourth levels are more useful in this use-case than they were in the first because the rapid rhythmic density of these levels can be mixed to sound less separated from the input source.

4.4. Future work

There are several possible future works, related to gamelan theory, that would improve this software. First, in gamelan music the elaborating instruments often sustain notes instead of playing on top of the second beat in a pair. We have implemented turning the notes into rests but not sustaining the previous notes. This sustain could be achieved by implementing the "spectral freeze" as described in [24]. Similarly, gamelan musicians add variations to their pitch and octave choice [5]. It may make the digital audio effect sound more organic to introduce some stochastic variation to emulate this. Another unimplemented aspect of gamelan music is that the orchestra's instruments are crafted in pairs that are intentionally out of tune to create a shimmering amplitude modulation that is intended to induce a religious trance-like effect [5]. Therefore it would be nice to include a stereo detuning parameter to the plug-in. Lastly, sliding tempo changes are an important part of gamelan music. Our method does not handle tempo changes. We are interested in Celemony Software's new Audio Random Access (ARA) [25] as a means to deal with this.

There are also possible future works unrelated to gamelan theory. One would be making use of beat detection so that the input signal does not have to already be perfectly quantized in time. Perhaps the work in [26] would prove useful for this. Another improvement would be to implement pitch detection on the input notes and limit the shifted notes to only be mapped to MIDI side-

chain input. That way, the tonal harmony of a song being mixed would not become cacophonous. Another improvement we are investigating is performing the pitch shifting without resampling as described in [17]. Another improvement would be to implement something like a noise gate with hysteresis so tails of notes that extend past the beat borders are not repeated. This may be simple to implement given that our method is aware of its location in the beat, and thus should know when to expect low and high amplitudes. Finally, spatialization with head-related transfer functions (HRTF) [27] would be an interesting feature to include.

5. ACKNOWLEDGMENTS

We would like to thank the Department of Computer Science at Appalachian State University, Will Padgett for testing the plug-in and composing example songs, Eyal Amir for the C++ advice, Dr. Virginia Lamothe at Belmont University for the introduction to gamelan music in 2011, and the anonymous reviewers for their feedback.

6. REFERENCES

[1] Richard Mueller, "Javanese influence on Debussy's 'Fantaisie' and beyond," *19th-Century Music*, vol. 10, no. 2, pp. 157–186, 1986.

[2] Neil Sorrell, "Gamelan: Occident or accident?," *The Musical Times*, vol. 133, no. 1788, pp. 66–68, 1992.

[3] Charles Matthews, "Algorithmic thinking and central Javanese gamelan," in *The Oxford Handbook of Algorithmic Music*. Oxford University Press, Feb. 2018.

[4] R. Anderson Sutton, *Traditions of Gamelan Music in Java: Musical Pluralism and Regional Identity*, CUP Archive, Apr. 1991.

[5] Jeff Todd Titon, Ed., *Worlds of music: an introduction to the music of the world's peoples*, Schirmer Cengage Learning, Belmont, CA, 5th ed edition, 2009.

[6] Lydia Ayers, "Merapi: A composition for gamelan and computer-generated tape," *Leonardo Music Journal*, vol. 6, 1996.

[7] Khafiizh Hastuti, Azhari Azhari, Aina Musdholifah, and Rahayu Supanggah, "Building melodic feature knowledge of gamelan music using apriori based on Functions in Sequence (AFiS) Algorithm," *International Review on Computers and Software (IRECOS)*, vol. 11, no. 12, Dec. 2016.

[8] Charles Michael Matthews, *Adapting and applying central Javanese gamelan music theory in electroacoustic composition and performance*, Ph.D thesis, Middlesex University, May 2014.

[9] Khafiizh Hastuti and Khabib Mustafa, "A method for automatic gamelan music composition," *International Journal of Advances in Intelligent Informatics*, vol. 2, no. 1, pp. 26–37, Mar. 2016.

[10] Gerd Grupe, *Virtual Gamelan Graz: Rules, Grammars, Modeling*, Shaker, 2008.

[11] Vincent Verfaille, Catherine Guastavino, and Caroline Traube, "An interdisciplinary approach to audio effect classification," in *Proc. Digital Audio Effects (DAFx-06)*, Montreal, Canada, Sep. 18-20, 2006.

[12] Daniel Arfib, "Different ways to write digital audio effects programs," in *Proc. Digital Audio Effects (DAFx-98)*, Barcelona, Spain, Nov. 19-21, 1998, pp. 188–191.

[13] Mark Dolson, "The phase vocoder: A tutorial," *Computer Music Journal*, vol. 10, no. 4, 1986.

[14] Dan Barry, David Dorran, and Eugene Coyle, "Time and pitch scale modification: a real-time framework and tutorial," in *Proc. Digital Audio Effects (DAFx-08*, Espoo, Finland, Sep. 1-4 2008.

[15] Eric Moulines and Jean Laroche, "Non-parametric techniques for pitch-scale and time-scale modification of speech," *Speech Communication*, vol. 16, no. 2, pp. 175–205, Feb. 1995.

[16] J. Laroche and M. Dolson, "Improved phase vocoder time-scale modification of audio," *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 3, pp. 323–332, May 1999.

[17] J. Laroche and M. Dolson, "New phase-vocoder techniques for pitch-shifting, harmonizing and other exotic effects," in *Proceedings of the 1999 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics. WASPAA'99 (Cat. No.99TH8452)*, New Paltz, NY, USA, 1999, pp. 91–94, IEEE.

[18] Amalia De Götzen, Nicola Bernardini, and Daniel Arfib, "Traditional (?) implementations of a phase-vocoder: The tricks of the trade," in *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-00)*, Verona, Italy, 2000.

[19] Magnus Erik Hvass Pedersen, "The phase-vocoder and its realization," May 2003.

[20] Joshua D. Reiss and Andrew McPherson, "The phase vocoder," in *Audio Effects: Theory, Implementation and Application*. CRC Press, 2015.

[21] Alexis Moinet and Thierry Dutoit, "PVSOLA: A phase vocoder with synchronized overlap-add," in *Proc. Digital Audio Effects (DAFx-11)*, Paris, France, Sept. 2011.

[22] Sumarsam, *Gamelan: Cultural Interaction and Musical Development in Central Java*, University of Chicago Press, Dec. 1995.

[23] F. J. Harris, "On the use of windows for harmonic analysis with the discrete Fourier transform," *Proceedings of the IEEE*, vol. 66, no. 1, pp. 51–83, Jan. 1978.

[24] Jean-François Charles, "A tutorial on spectral sound processing using Max/MSP and Jitter," *Computer Music Journal*, vol. 32, no. 3, pp. 87–102, Sept. 2008.

[25] Celemony, "ARA Audio Random Access | Celemony | Technologies," Available at https://www.celemony.com/en/service1/about-celemony/technologies, Accessed April 18, 2018.

[26] A. M. Stark, M. D. Plumbley, and M. E. P. Davies, "Real-time beat-synchronous audio effects," in *Proceedings of the 7th international conference on New interfaces for musical expression - NIME '07*, New York, New York, 2007, ACM Press.

[27] Durand R. Begault and Leonard J. Trejo, "3-D Sound for Virtual Reality and Multimedia," Technical Report NASA/TM-2000-209606, NAS 1.15:209606, IH-010, NASA Ames Research Center, Aug. 2000.

IMPROVING MONOPHONIC PITCH DETECTION USING THE ACF AND SIMPLE HEURISTICS

Carlos de Obaldía, Udo Zölzer

Department of Signal Processing and Communications
Helmut Schmidt Univeristy
Hamburg, Germany
deobaldia@hsu-hh.de

ABSTRACT

In this paper a study on the performance of the short time autocorrelation function for the determination of correct pitch candidates for non-stationary sounds is presented. Input segments of a music or speech signal are analyzed by extracting the autocorrelation function and a weighting function is used to weight candidates for assessing their harmonic strength. Furthermore, a decision is devised which alerts if there are possible non-related jumps on the fundamental frequency track. A technique to modify the spectral content of the signal is presented to compensate for these jumps, and a heuristic to return a steady fundamental frequency track for monophonic recordings is presented. The system is evaluated with several databases and with other algorithms. Using the compensation algorithm increases the performance of the ACF and outperforms current detection algorithms.

1. INTRODUCTION

Intonation in human perception corresponds to the perceivable tone that is registered by the human brain. This is perceived as pitch, which is in turn related to the fundamental frequency f_0 of a particular sound, that is, the main frequency component of the Fourier series expansion of a signal. In order to extract such information from a signal, several methods have been introduced which help find the perceivable intonation, or pitch, of a particular sound which is usually estimated by its fundamental frequency.

Among the methods used for determining f_0, the autocorrelation function (ACF) has always been of particular interest since it represents the periodicities encountered in a waveform. This is specially suited for determining the fundamental frequency in audio recordings. In this manner, prominent peaks of the ACF will give information about the perceived *pitch* or tone of a sound, particularly when the latter is of a stationary nature.

There are two major study problems at the time of finding pitch tracks in monophonic recordings: The first one is the problem of finding reliable pitch candidates. This can be approached, for example, by analyzing the spectral peaks in the frequency domain, or by determining the autocorrelation lags of a signal segment in a particular way. Several solutions have been proposed in the literature to mitigate ambiguities and perform a correct estimation of the pitch; by analysis of the response to a filter bank, by the use of linear predictive coding to extract pitch candidates [1], by analyzing the spectra with the use of correlation-based functions [2], by the use of spectral weighting functions [3], by applying correlation in the spectral domain [4], with the help of compound weighting functions [5], or by the use of the normalized cross correlation [6], among other methods.

The other issue is the tracking of the fundamental frequency f_0 when several disturbances in the pitch trajectory occur in order to select the correct pitch candidate. Examples of this problem can be when harmonics acquire a higher energy than the fundamental, where subharmonics surpass a decision threshold, or when a particular note is retained after the current note is played or sung. In these cases, candidate pitch lags are taken into consideration and a heuristic rule is created to compensate for shifts and ambiguities. The mitigation of these ambiguities have been approached by different monophonic pitch detection algorithms, but it has also been a matter of study for polyphonic recordings, such by taking different candidates of the autocorrelation-spectrum pairs [7], by statistical analysis of the pitch candidates [8], or by finding other domains where the ambiguity of the estimation can be diminished [9]. Another solution, for example, was considered by [10] in treating the common frequency trajectories based on a graph-solving problem in the spectrogram using continuous Hidden Markov Models (HMMs). Several algorithms also smooth the magnitude function of consecutive spectra to find the most prominent peak. In this work it is shown that the variability of the pitch candidates can be modeled in a simple way, whilst not having to reuse different heuristics for the detection, and a method is presented which provides a way to account for possible ambiguities.

2. BACKGROUND

While cross-correlating two different signals, similarities which exist between the signals at a moment in the present are found based on the history of the signal. Mathematically, the autocorrelation of a stationary discrete signal $x(n)$ is defined as

$$r_{xx}(m) = \sum_{n=m}^{N-1} x(n)x(n-m), \tag{1}$$

where $x(n)$ is a windowed signal of length N_w, and m is the index of the delayed sample, called the lag. This effectively compares an isolated signal segment with a time shifted version of itself. It can be inferred from Eq. 1 that the global maximum for the function appears at $m = 0$. If there exist any maxima in $r_{xx}(m)$ for $m > 0$, the signal is said to be periodic with $T_0 = m$ and will contain local maxima at multiples of the lag m. Since the autocorrelation of the signal at $m = 0$ equals the power in the signal, the height of the local maxima in $r'_{xx}(m)$ for $m > 0$ represents the relative harmonic power of the signal. Thus the ratio of the height

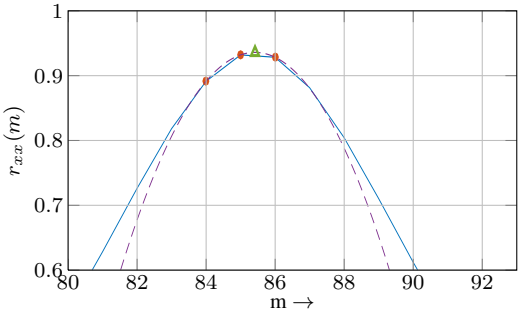


Figure 1: Interpolation of the candidates. The three red points are the anchors of the resulting interpolation shown with the dashed line. The corrected lag is the green triangle

of $r_{xx}(m)$ at specific lags m to the power of the signal,

$$r'_{xx}(m) \equiv \frac{r_{xx}(m)}{r_{xx}(0)}, \tag{2}$$

is considered to be the harmonic strength of the signal and indicates possible periodicities at the lags m whose harmonic strength approaches unity [11].

Since most of real-world signals are in the wide sense non-stationary, Eq.1 can be then calculated over a window $x_b(n)$ of length N_w centered at continuous places in time b spaced by N_h hop samples. This can give estimates of the fundamental frequency $f_0 = 1/T_0$ based on the harmonic strength at each frame b.

Another way to find lag candidates is with the use of the average magnitude difference function (AMDF)[5, 12], where a running sum is performed on the difference of the signal with itself, such that

$$g(m) = \sum_{n=0}^{N-1} (x(n) - x(n+m)). \tag{3}$$

This difference function can alternatively be used as a weighting function to help improve the accuracy of the ACF as a pitch estimator.

The local maxima of $r'_{xx}(m)$ of a particular segment provides a number of candidates which help determine the fundamental frequency in that segment along time. In this paper, several steps for preprocessing at the time of calculating the short time autocorrelation signal are assessed and evaluated, so that the most relevant steps for the calculation of the pitch candidates are taken into account. Furthermore, a method for weighting the harmonic strength of each candidate is presented. Methods to determine voiced and unvoiced parts in the signal are also presented, and a heuristic to determine and follow pitch trajectories is introduced as well. A spectral correction algorithm additionally improves the retrieval of candidates when the harmonic strength of multiples of the fundamental may mask the correct detection. Evaluation results and a comparison with several algorithms is presented and evaluated at the end.

3. PITCH EXTRACTION

Several algorithms which work on the spectral domain smooth the magnitude function of consecutive isolated spectra to find the most prominent frequency peak belonging to a particular frame [13]. In this work it is shown that the variability of the pitch candidates can also be modeled in a straightforward manner.

As it was introduced in Sec.2, the autocorrelation function has a global maximum at $m = 0$, where it represents the overall energy content of the particular isolated segment in which the function is calculated. Periodicities in $x_b(n)$ can then be found by analyzing the places where the pitch lag m is at other local maxima, or $r_{xx}(m_{peak})$ after $m = 0$. The relationship of these peaks to the maximum peak of the ACF would give a good cue of the fundamental period within a predefined interval $[m_{min}, m_{max}]$.

3.1. Pre-processing

For finding the fundamental frequency candidates, an incoming signal is hard - center clipped with a threshold of $\Gamma_c = 1 \times 10^{-3}$ to reduce the influence of other harmonic ratios on the signal [14]. The signal is then high-pass filtered with a butterworth filter of degree six at a cut off frequency of $f_c = 50$ Hz to reduce influences of low frequency noise.

At each windowed frame b of a signal $x(n)$ taken at a hop size $N_h = 0.01s \cdot f_s$, an isolated segment is extracted from $x(n)$ and multiplied with a hanning window

$$w(n) = \sin^2(\frac{\pi n}{N_w + 1}), \tag{4}$$

giving a signal segment $x_b(n)$ for $n = 0, ..., N_w + 1$ where N_w is the length of the window in samples. The autocorrelation of the segment is then calculated using Eq. 1.

The frame is then weighted using Eq. 3 such that

$$\hat{r}_{xx}(m) = \frac{r'_{xx}(m)}{g(m) + \alpha}, \tag{5}$$

where $\alpha = 1$ is taken. The output is then normalized to the maximum of the resulting signal.

3.2. Pitch Candidate Vector

The local maxima are found in the resulting weighted and normalized ACF $\hat{r}_{xx}(m)$ with a peak picking technique to generate a vector of pitch candidates

$$\mathbb{P} = [m_\kappa, ..., m_K], \;\; m_{min} < m_\kappa < m_{max}, \tag{6}$$

where m_{min} and m_{max} are the allowed minimum and maximum lags respectively, and $\kappa = \{1, \; ..., K\}$, where K is the number of detected pitch candidates above the threshold $g_r(m)$. After the previous step, just the positions of $\tilde{r}_{xx}(m)$ which surpass the threshold

$$q_r(m) = \frac{(\ln(f_s) - \ln(m))}{\ln(f_s)} \tag{7}$$

are taken in consideration and sorted in descending order with respect to the distance $\hat{r}_{xx}(m) - q_r(m)$. The resulting candidate vector $\mathring{\mathbb{P}}$ contains the current pitch estimation candidates for that particular frame.

3.3. Spectral Modification

Sometimes and due to the nature of particular sounds, the characteristics of the vocal tract, or the timbre characteristics of some instruments, will give away a formant structure which can detriment the detection performance of the most prominent lag [15].

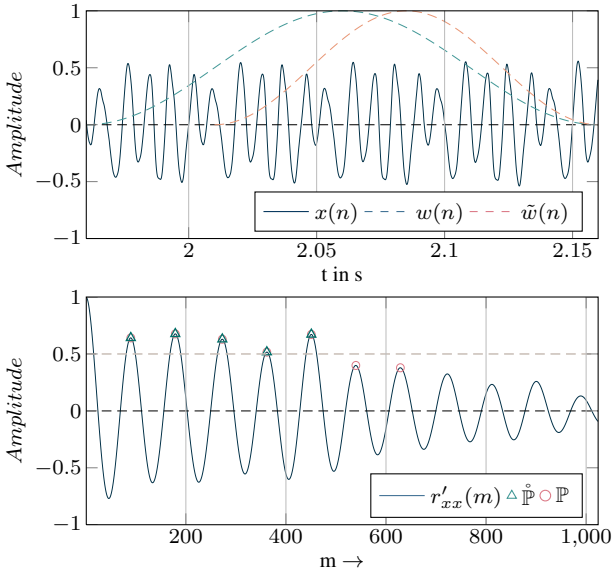


Figure 2: A case for finding the correct candidate with spectral modification. The top frame show the two windows used for calculating the lag. The lower picture shows the ACF of $w(n)$, and the peaks of $\mathring{\mathbb{P}}$ above a threshold.

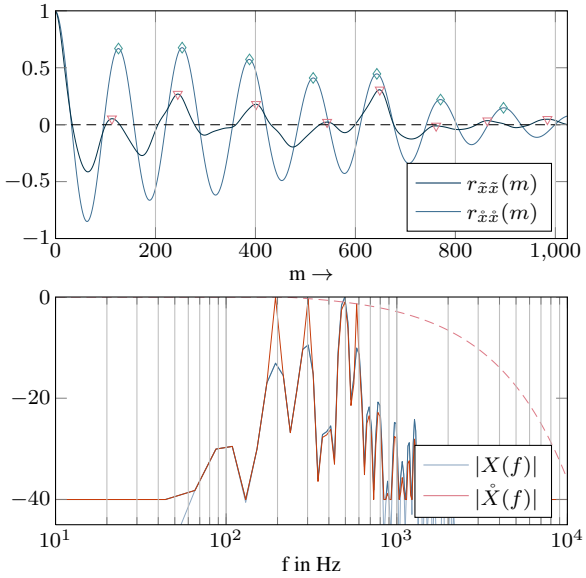


Figure 3: Spectral modification of the second window $\tilde{x}_b(n)$. The upper plot shows the resulting ACFs of $\tilde{x}_b(n)$ and the modified $\mathring{x}_b(n)$. The lower plot shows the spectrum $|\mathring{X}(f)|$ after modification.

To account for this, we introduce a candidate confirmation algorithm which follows the tracked pitch in the ACF, and determines its most plausible position. However some of the fragments may be accompanied by other harmonics which should not be taken in consideration, and the difference of the harmonic content between each frame should be taken into account.

To aid in this problem, the following technique makes use of the fact that if the harmonics of a particular fundamental frequency are equal in amplitude and since their harmonics are equally spaced in frequency, the distance between the resulting peaks of the autocorrelation from this modified waveform will match the fundamental period $T_0 = 1/f_0$.

The local maxima of $r_{xx}(m)$ above $q_r(m)$ are arranged in a vector $\mathring{\mathbb{P}}$, which contains the positions m of the pitch candidates sorted in descending order relative to $q_r(m)$. If the ratio of the harmonic strength of the first two lags $m_1,\ m_2$ in $\mathring{\mathbb{P}}$,

$$\frac{\min\left[\hat{r}_{xx}(m_1), \hat{r}_{xx}(m_2)\right]}{\max\left[\hat{r}_{xx}(m_1), \hat{r}_{xx}(m_2)\right]} \geq \gamma \tag{8}$$

taking γ=0.7, it will indicate that there is a possible mismatch with relation to the harmonic ratio of the frequency components of that particular signal segment. If there are more candidates in the vicinity of the harmonic strength for m_κ, a search is conducted to find a better estimate of the lag. A further window is thus applied on the frame to determine the correct pitch position, so that a second window in

$$\tilde{x}_b(n) = x(n - \frac{N_w}{4})\tilde{w}(n), \tag{9}$$

can be set, where $\tilde{w}(n)$ is a hanning window of length $\frac{N_w}{4}$ according to Eq.4 like in the upper plot of Fig.2. The spectrum of the signal snippet $\tilde{x}_b(n)$ is calculated taking

$$\begin{aligned} X_b(k) &= \mathcal{F}\{\tilde{x}_b(n)\} \\ &= \frac{1}{N_{\text{FFT}}} \sum_{n=1}^{N_{\text{FFT}}} x(n) \exp^{(\frac{-2\pi jkn}{N_{\text{FFT}}})} \forall k \in \{1, ..., N_{\text{FFT}}\}, \end{aligned} \tag{10}$$

where N_{FFT} is the size of the Fast Fourier Transform (FFT) and corresponds to the next power of two of N_w, and where $\tilde{x}_b(n)$ is zero padded accordingly. The spectrum is then normalized to its highest energy seen in any particular bin k, and the normalized power spectral density (PSD) of $\tilde{x}_b(n)$,

$$\tilde{X}_b(k) = \frac{|X_b(k)|^2}{\max[|X_b(k)|]}, \tag{11}$$

is calculated. The peaks of the spectral envelope which are above $-20\,dB$ are set to $0\,dB$ and a noise floor is in turn established at $-20\,dB$. We can then determine the harmonic content that remains in the signal from the frame before, by performing

$$\mathring{X}_b(k) = 2\tilde{X}_b(k) - \tilde{X}_{b-1}(k) \tag{12}$$

to reduce errors in transient regions. Furthermore, $\mathring{X}_b(k)$ is weighted with a roll-off factor of $40\,dB$ per octave starting at the first frequency bin k which is above $-20\,dB$ after Eq. 11, as it is depicted with a dashed line in Fig.3. The corresponding ACF of the delayed frame $\tilde{x}_b(n)$ is then calculated by using

$$r_{\mathring{x}\mathring{x}}(m) = \mathcal{F}^{-1}\{\mathring{X}_b(k)\}, \tag{13}$$

and the maximum peak $\mathring{m}$ in $r_{\mathring{x}\mathring{x}}(m)$ after $r_{\mathring{x}\mathring{x}}(0)$ is extracted. The resulting position in $\mathbb{P}$ which corresponds to the estimated fundamental frequency is found by calculating the closest m_κ in $\mathbb{P}$ with relation to $\mathring{m}$. Fig.3 shows the PSD of the second window in Fig. 2 which is used to determine the candidate m_κ for that frame b.

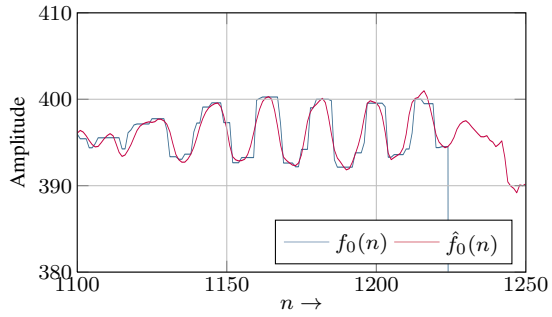


Figure 4: Result of applying the fractional lag calculation using the interpolation procedure for a vibrato around G_4 of a violin stem from Bach's 'Ach Gott und Herr'. The blue line represents the ground truth, $f_0(n)$.

3.4. Tracking Algorithm

Another problem at the time of finding the correct fundamental frequency is when the decay of a particular note is retained after another note starts. This has been shown to produce jumps in the fundamental frequency track, as the harmonic energy of two different fundamentals can be contained in the same frame.

To account for this, a method to provide a continuous tracking of f_0 is used. If there is a difference of more than $\delta_m = 10$ lags between consecutive estimates a decision is performed: If there exists a candidate in the current set $\mathbb{P}_b$ whose difference from $\mathbb{P}_{b-1}$ is at least 5 lags from the previous one, their positions are switched. This will prevent sporadic jumps in the track. Additionally, a guard interval is set such that this decision happens just when $\mathbb{P}_b$ of the ACF of 4 consecutive frames (b, ..., $b-3$) has contained at least one peak above the decision threshold.

3.5. Parabolic Interpolation

The ACF can be seen as a discrete and quantized signal where each pitch lag $m = 1, ..., M$ is an integer, so fractional tones can not be determined. A solution to this is to approximate the natural occurring tone with a parabolic interpolation taking as anchors the lag samples at positions between the local maxima [16]. If we take three points $y_1 = y(x_1)$, $y_2 = y(x_2)$, $y_3 = y(x_3)$ of a parabolic function of the form $y = ax^2 + bx + c$ around its local maximum $y(x_{max})$, and considering that $x_1 = 0$, $x_2 = 1$, $x_3 = 2$ and $y_1 = r_{xx}(m_\kappa - 1)$, $y_2 = r_{xx}(m_\kappa)$, $y_3 = r_{xx}(m_\kappa + 1)$, then

$$\Delta x_{max} = \frac{1}{2} + \frac{1}{2}\frac{(y_1 - y_2)(y_2 - y_3)(y_3 - y_1)}{2y_2 - y_1 - y_3}, \tag{14}$$

will give the relative position of the maximum of the parabola with respect to $y_1 = r_{xx}(m_\kappa - 1)$, so that the lag is now estimated to be at $m_\kappa + \Delta x_{max}$. Although the approximation of the shape of the ACF to a parabola will introduce a bias to the estimation of the frequency, it represents a better approximation of the pitch. Fig. 1 shows the lag positions and the peak resulting from this approximation. Fig.4 shows a result.

3.6. Voiced Activity Detection

For the determination of voiced and unvoiced regions, a simple post processing procedure has been used to clean the track from frames which are falsely labeled as voiced. This has been also a major field of study for pitch detection algorithms, specially with the use of speech signals. A segment is defined as voiced if, based on the source-filter model of speech production, the signal to excite the vocal tract filter is of a periodic nature.

Thus the pitch candidates which lie below the threshold described in Eq. 7 are not taken in consideration. Secondly, candidates whose frame's root mean square (RMS) energy is below a threshold $\Gamma_E = 0.705 \times 10^{-3}$ are also not considered and taken as unvoiced. Lone pitches in frames which are separated more than 20 lags from the previous one are also considered as unvoiced. Moreover, pitch candidates are only considered if the frequency positions in the ACF are separated at least 15 lags. Lone candidates which spawn over three consecutive lags, are also discarded. This results in a continuous pitch track.

In summary, to find voiced and unvoiced regions, a frame b is voiced if

(a) $$\mathrm{RMS}\{x_b(n)\} \geq \Gamma_E, \tag{15}$$

and

(b) $$r_{xx}(m) > q_r(m), \tag{16}$$

and if

(c) $$|m_{b-1} - m_b| \leq 15 \tag{17}$$

which is the maximum lag so that the exchange between the candidates does not surpass this threshold.

Fig. 6 shows the results before and after removal of false positives for a saxophone and a bassoon recording from the Bach 10 dataset.

Finally, the algorithm can be summarized for each particular window at a frame b as follows:

1. Extract a signal segment $\hat{x}_b = x(n) \cdot w(n)$, where $w(n)$ is a hanning window of length N_w.
2. Calculate the autocorrelation function $r'_{xx}(m)$ according to Sec.3.1.
3. Calculate the AMDF as in Eq. 3
4. At each lag m, calculate the weighted function $\hat{r}_{xx}(m)$ as in Eq. 5
5. Normalize the weighted ACF $\hat{r}_{xx}(m)$ with respect to its maximum
6. Get a pitch candidate vector from the autocorrelation peaks and sort it according to Sec.3.2
7. Confirm pitch candidates according to the algorithm described in Sec.3.3, so that a new peak vector

$$\mathring{\mathbb{P}} = \{m_\kappa, \ldots, m_K\}, \tag{18}$$

is obtained, where m_1 corresponds to the estimated pitch lag, so that the estimated f_0 of frame b is $\hat{f}_0 = \frac{f_s}{m_1}$.

8. Run the tracking algorithm of Sec.3.4 for detecting discontinuities.
9. Determine voiced and unvoiced frames.

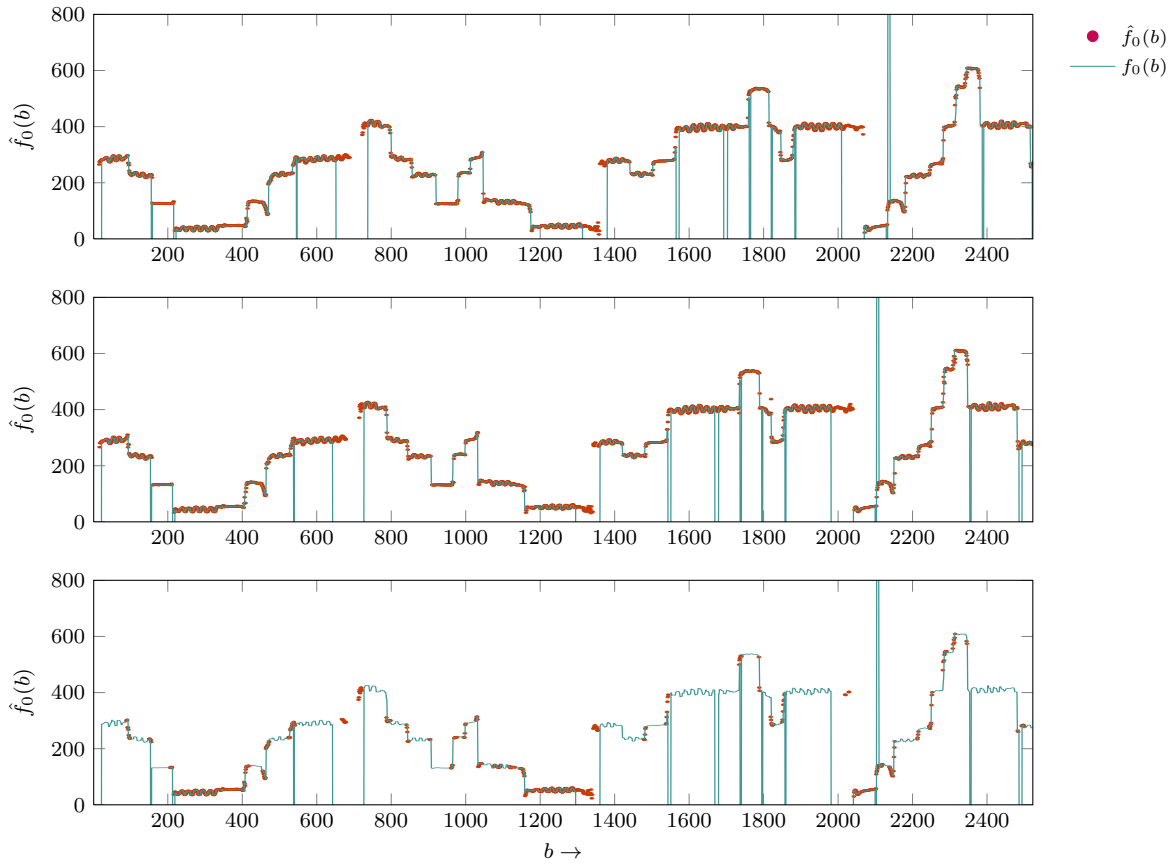


Figure 5: Pitch track for a violin recording of Bach's 'Ach Gott und Herr'. The bottom plot shows a result after selecting the first candidate of the ACF without the spectral modification decision. The middle one shows the same result but with a smaller window $N_w = 1024$. The uppermost plot shows the result with spectral modification with the same parameters as the bottom plot.

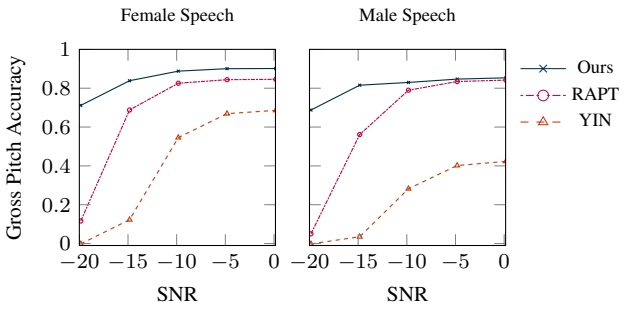


Figure 7: GPA for different SNR levels using the presented algorithm. The result shows an improvement on the pitch accuracy in voiced regions.

4. RESULTS AND EVALUATION

For the evaluation of the performance of the algorithm, two databases are used: The PTDB-TUG database, consisting of 10 female and 10 male speakers, each pronouncing 236 utterances from the TIMIT set, and the Bach 10 Chorales dataset which consists of stems for 10 different Bach songs which are recorded for violin, clarinet, saxophone and bassoon. The database also contains a midi set, but the proposed algorithm is not tested on midi data. The previously presented algorithm is thus evaluated on the monophonic stems of the Bach 10 dataset. For both databases, the original ground truth data is used as evaluation criterium. Results are evaluated with the YIN algorithm and the RAPT and PEFAC implementations found in the voicebox[17].

The overall accuracy is given by the F_0 Frame Error (FFE), which calculates the accuracy given a particular constrain among all the data and all the frames in the signal. For the evaluation, an error of 20%, 8% and $10\,Hz$ within the ground truth is chosen for all cases. Moreover, the results are also compared taking the Gross Pitch Error (GPE), which is the proportion of voiced frames in both the ground truth (GT) and the result, and the Fine Pitch Error (FPE) where the standard deviation of the distribution of relative error values is taken into account [18, 19]. Results are shown for the Bach 10 database in Table 1 and for the PTDB-UG in Table 2.

The algorithm achieves around 97% frame accuracy over the Bach10 dataset and around 90% frame accuracy for the speech examples of the PTDB, relative to 20% of the ground truth values provided for both datasets. For the Bach 10 evaluation a window of $N_w = 1536$ is used which gives a ground pitch error of $3,4\%$ and a F_0 frame error of about $8,4\%$ within 10 Hz for the whole of

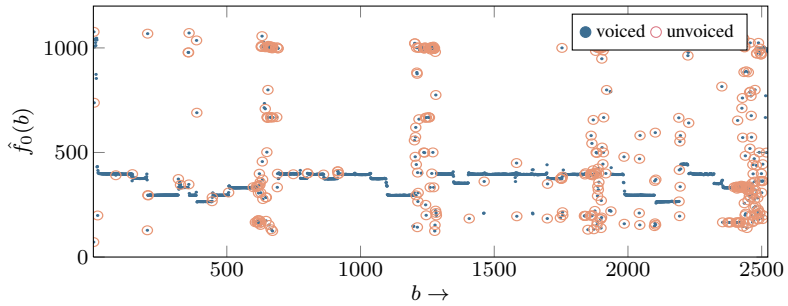


Figure 6: Voiced/Unvoiced detection is performed by applying a simple rule which combines the energy in the frame, the harmonic ratio, and the continuity of the fundamental track.

	violin			clarinet			sax			basoon			average		
Algorithm	GPE	FPE	FFE	GPE	FPE	FFE	GPE	FPE	FFE	GPE	FPE	FFE	GPE	FPE	FFE
Mod.	0.0314	1.7011	0.1107	0.0298	1.1313	0.0662	0.0209	1.1574	0.0478	0.0526	0.7795	0.0955	0.0337	1.1923	0.0801
8%	0.0185	0.5683	0.0199	0.0190	0.6052	0.0180	0.0150	0.6012	0.0155	0.0513	0.5618	0.0559	0.0260	0.5841	0.0273
20%	0.0156	0.8284	0.0179	0.0165	0.8369	0.0166	0.0122	0.8992	0.0142	0.0477	0.9681	0.0546	0.0230	0.8831	0.0258
$N_w = 1536$	0.0315	1.7293	0.1098	0.0301	1.1598	0.0631	0.0256	1.2116	0.0526	0.0508	0.8810	0.1111	0.0345	1.2454	0.0842
8%	0.0170	0.6175	0.0188	0.0183	0.5958	0.0183	0.0180	0.6667	0.0199	0.0492	0.6277	0.0753	0.0256	0.6269	0.0331
20%	0.0131	0.9566	0.0164	0.0151	0.9179	0.0164	0.0119	1.1938	0.0163	0.0421	1.2224	0.0720	0.0205	1.0727	0.0303
RAPT	0.0405	1.8504	0.1277	0.0343	1.1431	0.0711	0.0352	1.2343	0.0612	0.0693	0.9417	0.1090	0.0449	1.2924	0.0922
8%	0.0236	0.6702	0.0210	0.0223	0.6129	0.0198	0.0253	0.7280	0.0225	0.0675	0.6858	0.0602	0.0347	0.6742	0.0309
20%	0.0177	1.1332	0.0157	0.0181	1.0013	0.0161	0.0174	1.3514	0.0154	0.0573	1.4394	0.0512	0.0276	1.2313	0.0246
YIN	0.0555	2.286	0.1118	0.0340	1.3710	0.0561	0.0529	0.9837	0.0696	0.1510	0.4114	0.1631	0.0734	1.2622	0.1001
8%	0.0503	0.5167	0.0449	0.0281	0.5109	0.0250	0.0505	0.4570	0.0449	0.1508	0.2572	0.1344	0.0699	0.4353	0.0623
20%	0.0498	0.6102	0.0444	0.0272	0.6164	0.0241	0.0493	0.6102	0.0438	0.1504	0.3490	0.1340	0.0692	0.5387	0.0616
PEFAC	0.8670	1.7155	0.8055	0.6378	0.9186	0.5784	0.1250	0.9773	0.1338	0.1205	0.7539	0.1306	0.4376	1.0913	0.4121
8%	0.8648	0.7488	0.7698	0.6360	0.4417	0.5658	0.1202	0.5601	0.1069	0.1185	0.5756	0.1058	0.4348	0.5816	0.3871
20%	0.8594	4.478	0.7651	0.6310	1.999	0.5613	0.1083	1.6454	0.0964	0.1074	1.5181	0.0959	0.4265	2.4101	0.3797

Table 1: Results based on absolute error for the Bach 10 database. Comparison is done with the AMDF weighted ACF, with the exchange turned on.

the dataset. The error is higher on the bassoon stems, and improves to about 8% when the spectral modification algorithm is used. The only parameter which is being changed is the window length for the calculation.

For the PTDB database in Table 2, the accuracy is also presented using two different windows, at $N_{w1} = 2048$ and at $N_{w2} = 4096$ with spectral modification. The algorithm performs well over the other for the female speech samples, although for the smaller window size the algorithm finds problems in detecting low frequency fundamentals. For a higher window length, the algorithm performs well with spectral modification achieving over 92% accuracy for the F_0 frame error and around 89% ground pitch accuracy within 10 Hz of the ground truth.

Results under additive white gaussian noise (AWGN) at different signal to noise ratios (SNR) also show an advantage in comparison with the above mentioned algorithms. Fig. 7 show the Gross Pitch Accuracy (GPA) (where GPA $= 1 -$ GPE), for the presented algorithm evaluated under a subset of 20 utterances for the 10 male and 10 female speakers of the PTDB database. By using the presented pitch tracker, an accuracy of over 70% for all the cases, showing an increased improvement in the overall pitch accuracy for voiced regions.

In Fig.5 a result is shown on a violin track for two particular window sizes ($N_{w1} = 1024$ and at $N_{w2} = 2048$). The lower plot shows the detection performed with N_{w2} without spectral modification and the uppermost plot with the decision algorithm. It shows that although the lowest window performs well if the tracking algorithm is used, the accuracy increases using a longer window without diminishing the performance. This also shows the good performance of the spectral modification algorithm at the time of performing a correct detection without the need of further parametrization of the algorithm.

5. CONCLUSION

It has been shown that the algorithm proposed in this paper can be a reliable monophonic pitch detector because it pays attention to several properties in sound signal based on simple heuristics. Unwanted jumps in the pitch track which can occur due to the native timbre characteristics of musical instruments, or due to the resonant frequencies of the vocal tract at the time of uttering particular vowel qualities, can be diminished by the use of a spectral correction function when there exist ambiguities in the output of the weighted autocorrelation signal. Furthermore, the f_0 track is smoothed by the use of a tracking function that resolves the possible disturbances when, for example, a transient between continu-

	female			male		
Method	GPE	FPE	FFE	GPE	FPE	FFE
Mod. $N_{w\,2}$	0.1120	2.2640	0.0750	0.1135	2.1105	0.0805
8%	0.1152	2.1508	0.0715	0.1169	2.0854	0.0798
20%	0.1149	2.1809	0.0715	0.1150	2.1561	0.0797
Mod. $N_{w\,1}$	0.2516	3.8864	0.0997	0.2865	3.8846	0.1251
8%	0.2486	3.7214	0.0960	0.2918	3.8500	0.1243
20%	0.2461	3.8046	0.0959	0.2817	4.0088	0.1241
$N_w = 2048$	0.1063	2.5431	0.0758	0.1329	2.5086	0.1589
8%	0.1069	2.4395	0.0749	0.1375	2.4752	0.1587
20%	0.1057	2.4875	0.0749	0.1346	2.5677	0.1587
YIN	0.3028	2.2002	0.0835	0.5663	1.4291	0.1308
8%	0.3058	2.0779	0.0814	0.5728	1.4050	0.1310
20%	0.3058	2.0781	0.0814	0.5728	1.4057	0.1310
RAPT	0.1523	3.0357	0.1144	0.1523	2.8079	0.0955
8%	0.1520	2.9129	0.1065	0.1536	2.7766	0.0899
20%	0.1503	2.9803	0.1062	0.1496	2.8834	0.0891
PEFAC	0.3931	3.6037	0.2237	0.3586	3.2301	0.1885
8%	0.3827	3.4276	0.2082	0.3609	3.2083	0.1789
20%	0.3819	3.4668	0.2080	0.3562	3.3335	0.1779

Table 2: Performance results of different algorithms for the PTDB-TUG database.

ous notes is present. The fundamental frequency track can thus be reliably extracted by the use of the proposed ACF based algorithm without the need for tuning particular window sizes. Although there exist further possibilities of improvement in detecting the moment where transitions occur and for voiced segment determination, it is possible to find a decision threshold for the application of the presented algorithm. The spectral modification algorithm performs well at the moment of finding these transitions, and can be reliable for improving detection of the fundamental in voiced speech and musical signals.

6. REFERENCES

[1] Bruce Secrest and George Doddington, "An integrated pitch tracking algorithm for speech systems," in *ICASSP'83. IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 1983, vol. 8, pp. 1352–1355.

[2] Li Hui, Bei-qian Dai, and Lu Wei, "A pitch detection algorithm based on AMDF and ACF," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*. IEEE, 2006, vol. 1, pp. I–I.

[3] Sira Gonzalez and Mike Brookes, "PEFAC - a pitch estimation algorithm robust to high levels of noise," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 22, no. 2, pp. 518–530, 2014.

[4] E Chilton and BG Evans, "The spectral autocorrelation applied to the linear prediction residual of speech for robust pitch detection," in *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*. IEEE, 1988, pp. 358–361.

[5] Alain De Cheveigné and Hideki Kawahara, "Yin, a fundamental frequency estimator for speech and music," *The Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1917–1930, 2002.

[6] David Talkin, "A robust algorithm for pitch tracking (rapt)," *Speech coding and synthesis*, vol. 495, pp. 518, 1995.

[7] Sebastian Kraft and Udo Zölzer, "Polyphonic pitch detection by matching spectral and autocorrelation peaks," in *Signal Processing Conference (EUSIPCO), 2015 23rd European*. IEEE, 2015, pp. 1301–1305.

[8] Johannes Böhler and Udo Zölzer, "Monophonic pitch tection by evaluation of individually parameterized phase locked loops," in *19th International Conference on Digital Audio Effects (DAFX16)*, 2016, vol. 68.

[9] Sebastian Kraft, Alexander Lerch, and Udo Zölzer, "The tonalness spectrum: feature-based estimation of tonal components," in *16th International Conference on Digital Audio Effects (DAFx-13), Maynooth, Ireland*, 2013, p. 8.

[10] Gregor Pirker, Michael Wohlmayr, Stefan Petrik, and Franz Pernkopf, "A pitch tracking corpus with evaluation on multipitch tracking scenario," in *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[11] Paul Boersma, "Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound," in *Proceedings of the institute of phonetic sciences*. Amsterdam, 1993, vol. 17, pp. 97–110.

[12] Myron Ross, Harry Shaffer, Andrew Cohen, Richard Freudberg, and Harold Manley, "Average magnitude difference function pitch extractor," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 22, no. 5, pp. 353–362, 1974.

[13] Tom De Mulder, Jean-Pierre Martens, Micheline Lesaffre, Marc Leman, Bernard De Baets, and Hans De Meyer, "An auditory model based transcriber of vocal queries," 2003.

[14] Joseph Carl Robnett Licklider and Irwin Pollack, "Effects of differentiation, integration, and infinite peak clipping upon the intelligibility of speech," *The Journal of the Acoustical Society of America*, vol. 20, no. 1, pp. 42–51, 1948.

[15] Mohan Sondhi, "New methods of pitch extraction," *IEEE Transactions on audio and electroacoustics*, vol. 16, no. 2, pp. 262–266, 1968.

[16] Adrian von dem Knesebeck, *Analyse- und Syntheseverfahren zur automatischen Harmonisierung von Gesang*, PhD dissertation, Helmut Schmidt Universität, 2014.

[17] Mike Brookes et al., "Voicebox: Speech processing toolbox for matlab," *Software, available [Mar. 2011] from www. ee. ic. ac. uk/hp/staff/dmb/voicebox/voicebox. html*, vol. 47, 1997.

[18] Wei Chu and Abeer Alwan, "Reducing f0 frame error of f0 tracking algorithms under noisy conditions with an unvoiced/voiced classification frontend," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2009, pp. 3969–3972.

[19] Sofia Strömbergsson, "Today's most frequently used f0 estimation methods, and their accuracy in estimating male and female pitch in clean speech.," in *INTERSPEECH*, 2016, pp. 525–529.

SOUND TEXTURE SYNTHESIS USING CONVOLUTIONAL NEURAL NETWORKS

Hugo Caracalla

UMR STMS 9912
Sorbonne Université, IRCAM, CNRS,
Paris, France
hugo.caracalla@ircam.fr

Axel Roebel

UMR STMS 9912
IRCAM, Sorbonne Université, CNRS,
Paris, France
axel.roebel@ircam.fr

ABSTRACT

The following article introduces a new parametric synthesis algorithm for sound textures inspired by existing methods used for visual textures. Using a 2D Convolutional Neural Network (CNN), a sound signal is modified until the temporal cross-correlations of the feature maps of its log-spectrogram resemble those of a target texture. We show that the resulting synthesized sound signal is both different from the original and of high quality, while being able to reproduce singular events appearing in the original. This process is performed in the time domain, discarding the harmful phase recovery step which usually concludes synthesis performed in the time-frequency domain. It is also straightforward and flexible, as it does not require any fine tuning between several losses when synthesizing diverse sound textures. Synthesized spectrograms and sound signals are showcased, and a way of extending the synthesis in order to produce a sound of any length is also presented. We also discuss the choice of CNN, border effects in our synthesized signals and possible ways of modifying the algorithm in order to improve its current long computation time.

1. INTRODUCTION

The main difficulties encountered in sound texture synthesis become apparent when trying to properly define them. While examples of textures easily come to mind (e.g. environmental noises such as wind or rain, crowd hubbub, engine sounds, birds singing, etc.), pinpointing their common factors proves much harder: randomness appears to be one, along with a "background" aspect caused by an important number of indistinguishable small audio events happening at once. But this is not all, since we would still tend to call a sound including small occasional events happening in the foreground a texture. Hence, the definition offered by Saint-Arnaud [1], summed up by Schwartz [2], of "a superposition of small audio atoms overlapping randomly while following a higher level organization" is incomplete because it only encompasses completely stationary textures. It can even be argued that in reality no such texture can be observed: a synthesis algorithm strictly following this definition would thus be incomplete and of little practical use.

This means that a sound texture synthesis algorithm needs to be able to synthesize small indiscernible and random events but also singular, recognizable events, both harmonic (e.g. birds chirping) or not (e.g. crowd clapping). This extremely broad range of sounds is precisely what makes texture synthesis complex and why common synthesis algorithms (for instance sinusoidal models) prove ill-suited for it, making it require dedicated ones.

Before presenting an overview of existing sound texture synthesis algorithm, we will first detail more precisely what is expected from them.

In the case of textures, "re-synthesis" is a term as fitting as "synthesis": starting from an existing texture, the goal is usually to create a sound that is different from the original while still being recognizable as the same kind of texture, as if it had been recorded only moments later. Although this is the prime goal of the algorithm, this obviously does not exclude the possibility of manipulating the synthesized texture. For instance, it could be desirable to allow the algorithm to synthesize texture lasting any arbitrary length of time, or to be able to have the synthesis evolve throughout time, altering its properties or progressively turning into another texture.

To achieve such a result, a broad variety of methods have been developed: for the needs of this article, we will split them into 3 different categories.

The first of those is *physics-based synthesis*. It regroups methods which goal is to first emulate the phenomenon at the source of the texture (for instance the impact of a drop of rain on a flat surface) via a physically informed model of it. From there one can simulate any number of events, dimensioning and randomizing them so as to fit the target texture. The result of this is a convincing physical simulation of the texture (see for instance [3]). While this method has the potential of being extremely controllable and allowing the manipulation of synthesis parameters that have a physical meaning, it also has the obvious flaw of not being flexible at all. Each algorithm correspond to one and only one kind of texture: a physical model of the rain will prove poorly suited to synthesize a flock of birds twittering.

The second is *granular synthesis*. It regroups methods in which the original texture is first chopped into milliseconds-long audio particles, then reordered and concatenated to reconstruct a new texture (see for instance [4]). While being quite versatile in the range of texture it is capable of re-synthesizing, such a method is also heavily dependent on the choice of atom size and requires more complex reordering methods when one tries to synthesize a broader array of textures. In particular, reconstructing any foreground event lasting more than the size of an atom proves a hard task.

The last synthesis category, to which our algorithm belongs, is *parametric synthesis*. It regroups synthesis methods in which the general goal is to establish a set of parameters to describe textures with. If those parameters are properly chosen, any two textures which parameters are equal will sound alike without necessarily being identical. From there, one would only need to modify a sound until its parameters are equal to those of a target texture

in order to re-synthesize it. While in theory this method is able to synthesize any and all kind of textures, in practice the quality and flexibility of the synthesis entirely depend on the choice of parameters. Those parameters need to describe the texture so as to hold enough information to re-synthesize a similar one, while not holding too much so as to not over-constraining the synthesis (in which case the only way of reaching a set of parameters would be by creating a perfect copy of the original texture). In addition to this, the parameters must be adapted to the description of the widest possible range of textures.

This paradigm is notably used by McDermott and Simoncelli in [5], where a set of statistics extracted from the critical and modulation bands are used as parameters. This algorithm gives convincing results for a broad range of textures, but fails when trying to synthesize textures containing singular events. Inspired by the work of Gatys et al. [6] who uses the cross-correlations between the feature maps of a trained 2D convolutional neural network (CNN) as parameters to synthesize visual textures, several attempts to convert this approach to audio have been made. In [7], Ulyanov and Lebedev use the same principle applied to spectrograms, with the frequency dimension acting as color depth and using a 1D CNN, to synthesize textures with moderate success. In [8], Antognini et al. add to this approach several constraints aimed at recreating rhythm with a better fidelity and increasing the diversity of results. While giving convincing results, this method requires fine tuning in order to balance the constraints. Since changing target texture implies tuning those synthesis parameters, this makes the algorithm lose in flexibility. In addition to this, both this method and Ulyanov and Lebedev's eventually output a spectrogram: it is then necessary to recover its phase and invert it to retrieve an audible time signal, using methods such as the Griffin and Lim algorithm [9]. This phase recovery step is an added burden to the synthesis as it tends to downgrade the quality of the audio signal, even more so when working with complex sounds such as textures.

Although not directly applied to sound texture synthesis, it may be noted that several works such as those of Grinstein et al. [10], Barry and Kim [11] and Tomczak [12] also use the same parametrization extracted from the feature maps of a CNN as part of audio style transfer processes.

In this work, we present a new parametric texture synthesis based on the method of Gatys et al.[6]. Our synthesis algorithm does not require any fine tuning or spectrogram inversion and works with a wide array of textures, including those presenting strong singular events. We also present a few examples to demonstrate its possibilities and proceed to discuss those results.

2. METHOD

Following the principle of parametric synthesis, we first define a set of parameters to represent a sound texture with.

2.1. Parametrization

2.1.1. Pre-processing

As our method is adapted from the work of Gatys et al. [6] on 2D images, we require a 2D representation of our sound signal.

To this effect we work with log-spectrograms. The log-spectrogram S is computed using the spectrogram X, taken as the magnitude of the short-term Fourier Transform (STFT) of the sound signal:

$$S = \frac{\log(1 + CX)}{\log(1 + C)} \tag{1}$$

where C a factor controlling compression: the larger C is, the more details we will get at low amplitudes. This choice of normalization is made to ensure that the spectrogram is both compressed by the log function and comprised between 0 and 1.

In practice we work with time signals sampled at 22050 Hz, and a choose as STFT parameters a window length of 512 and hop-size of 256. The sound signals all have a length of 262400 samples so that their log-spectrograms are 1024 frames long, and have a bit depth of 16. The compression factor C is set to 1000.

For the rest of this article, any time-frequency matrix will have frequency as first axis, and time as second. For instance, $S(f, t)$ denotes the value of S at the f-th frequency bin and t-th time sample.

2.1.2. Network choice

Seeing as Gatys et al. use a network trained for image recognition, our first intuition was to use an equivalent network for working on spectrograms. As such we initially trained a simple deep 2D CNN on recordings taken from freesound.org in scene recognition in order to use it for synthesis. But in [13], Ustyuzhaninov et al. show that visual textures of the same quality as those obtained by Gatys et al. can be synthesized using a single-layer untrained CNN with various filter size instead of a trained CNN.

This proves to still hold for sound textures: the network we use to synthesize the textures presented in this article is a single-layer untrained 2D CNN using filters of different sizes and ReLU activation. Its single layer is made of 128 square filters of each of the sizes [3, 5, 7, 11, 15, 19, 23, 27] with a stride of (1, 1) and zero-padding so that the differently-sized convolutions can then be stacked, followed by the rectified linear unit (ReLU) activation layer. The weights from the filters are drawn from a uniform distribution between -0.05 and 0.05, and no bias is applied.

For generalization's sake, the rest of the method is nonetheless presented with a network that has K layers (with K being potentially more than 1), although it is still valid when using a single-layer network.

It is worth noting that unlike the methods presented in [7, 8], we use the log-spectrograms as 2D images with time and frequency replacing the two space dimensions and not as a 1D signal with frequency as depth, hence the need for 2D convolutions.

2.1.3. Parameters computation

Let us denote $F^k_{i,(x,y)}$ the value of the i-th feature map of the k-th layer at the position (x, y). In [6], Gatys et al. use the Gram matrices of each layer of the network as parameters. The (i, j) element of the gram matrix G^k of k-th Gram marix is defined as the cross-correlation between the i-th and j-th feature maps of the layer:

$$G^k(i, j) = \sum_{(x,y)} F^k_i(x, y)F^k_j(x, y) \tag{2}$$

The parameter set is chosen as the list of gram matrices from G^1 to G^K. Although this proves a good choice for visual textures, such parameters cannot be directly used in the case of sound

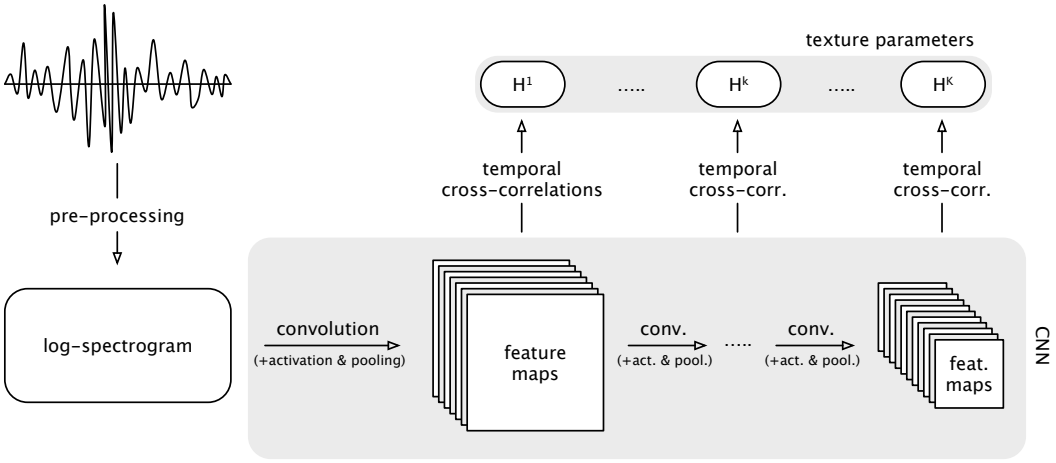


Figure 1: *Computation of the 3-dimensional parameter tensors from the temporal cross-correlations of the feature maps of a CNN, .*

textures. Indeed, those Gram matrices average all spatial information when performing a sum over all positions (x, y), thus implying that the parametrization should be invariant in both directions. This does no translate well to sound, since sound textures behave differently regarding time and regarding frequency: while we wish for a pseudo-stationarity over the time dimension, there is no reason for there to be any invariance to pitch-shifting. As such, we instead use the 3-dimensional tensors H^k defined as:

$$H^k(i, j, x) = \sum_y F_i^k(x, y) F_j^k(x, y) \tag{3}$$

Defined this way, the tensors H^k with $k \in [1, K]$ that we use as parameters do indeed average all information from the time dimension, but keep the information regarding the frequency dimension intact, thus achieving our goal. The parameter extraction process is represented in Figure 1.

2.2. Texture loss

As mentioned in Section 1, the main goal of parametric synthesis is to create a sound which has the same parameters values as a those of a target sound. Seeing as we now have a parameter set, we only need to define a quantitative error function which will then be minimized throughout the synthesis process. To that effect, we use a simple distance function between the two sets similarly to Ustyuzhaninov et al. [13]:

$$\mathcal{L} = \sum_k \frac{\|\tilde{H}^k - H^k\|_2}{\|\tilde{H}^k\|_2} \tag{4}$$

with $\|.\|_2$ denoting the L2 norm, and the tilde denoting the target texture parameters.

2.3. Optimization

The last step of the synthesis process is to create a sound signal which minimizes the texture loss. Since the chain of operations leading to the computation of the texture loss is differentiable (despite passing through the complex domain due to the STFT: see [14] for further insight), we may use any optimization algorithm requiring the gradient of the error function to iteratively modify a sound until it reaches a satisfying minimum of the loss function (similar time domain synthesis can be found in [11, 12].

We observed that performing the optimization on the log-spectrogram is iteration-wise faster than performing the optimization directly on the sound signal (both of them being initialized using white noises). This seems to indicate that the texture loss is simpler to minimize when working in the time-frequency domain rather than when working directly in the time domain. To take advantage of that fact, we first perform a quick synthesis of a log-spectrogram and invert it using a random phase matrix (which would correspond to performing one step of the Griffin-Lim algorithm): while this inversion raises the value of the texture loss, it still makes for a good initialization of the optimization in the time domain. This allows us to skip a major part of the optimization process on the sound signal. Once performed this optimization results in a sound signal which minimizes the texture loss, meaning its parameters values are close to those of the target texture. The synthesis process is illustrated in Figure 2.

As in [6] and [8], we found that the L-BFGS algorithm (introduced in [15]) converges fast and yields good audio results. Starting from a white noise image, we perform 1000 iterations of it in the time-frequency domain to create the initialization of the time domain optimization. The time domain optimization is then performed over 10000 iterations. Using a GeForce GTX 1080 Ti GPU, the whole process takes around an hour.

3. EXPERIMENTS

3.1. Experimental results

The log-spectrograms of both target and synthesized textures are shown in Figure 3 for three sounds: a wildlife scene with crickets chirping in the background and a bird singing in the foreground (recognizable to its inverted "v"-shaped patterns), birds singing both in the background and in the foreground (with one strongly standing out the mid-frequency range), and the hubbub of a crowd chatting. The audio signals of all three are available for listening online [1] along with other textures such as wind, bees and fire sounds.

[1] See `http://recherche.ircam.fr/anasyn/caracalla/dafx19/synthesis/`

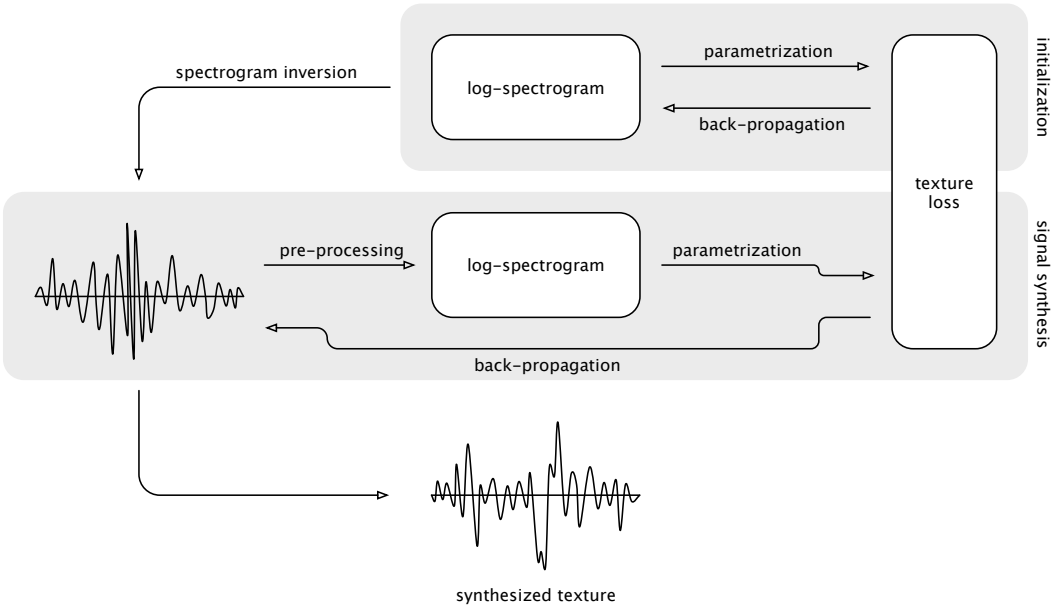


Figure 2: *Organization of the texture synthesis algorithm: a quick optimization on a log-spectrogram is performed in the time-frequency domain and its result is inverted. It then serves as initialization for the main optimization performed on a sound signal.*

3.2. Discussion

3.2.1. Results analysis

The strength of this texture synthesis algorithm lies in the fact that it manages to both synthesize background and foreground events convincingly. The crowd chatter, which includes no singular events, is as well recreated as the bird singing loudly in front of a flock.

Another advantage to it is the absence of any parameter tuning: unlike the method presented by Antognini et al. [8] where three losses need to be balanced through the optimization process and potentially from one texture to another, the texture loss used here is straightforward and requires no tuning. This way the algorithm can effectively be used without needing to take into account which texture is being synthesized.

In addition to this, because the final optimization is performed on the time signal our algorithm does not end with a spectrogram inversion (the harm a phase retrieval process could bring is clearly noticeable when inverting spectrograms of existing texture and comparing the resulting signal to the original).

3.2.2. Untrained vs. trained network

As mentioned in Section 2.1.2, both trained and untrained network have been used in visual texture synthesis with success. The main argument in [6] is that the CNN trained for image recognition has learned filters adapted to common shapes encountered in images. This, coupled with the depth of the network, is what supposedly allows the network to recreate a large array of shapes when trying to reproduce the cross-correlations between the activations of its filters once the network has been fed an image (in this case, a visual texture). This argumentation is challenged by Ustyuzhaninov et al. in [13], who demonstrate that a single-layered untrained CNN can perform as well as a trained network given enough filters. This would tend to imply that given enough random filters, the space of shapes recreated when synchronizing some of those filters is wide enough to compensate for the lack of training.

This translates seamlessly to sound textures: while we first worked with networks trained for acoustics scene recognition in an attempt to emulate the process of [6], experiments with untrained network show that they perform just as well. This being said, it could be interesting to explore the difference between the use of the two further: for instance, trained CNN might require less filters than untrained ones, thus making our parameter tensors lighter and the computations faster. The depth of the trained CNN might also help it capture correlations across distant events in the spectrogram. This would indeed be useful, seeing as birds textures from Figure 3 show that while the algorithm manages to reproduce the local pattern of bird cries well, it fails to reproduce the larger pattern of groups of cries separated by gaps of a few seconds. This is also quite audible when listening to the attempt at (non-texture) singing voice synthesis: since the human voice is rich in harmonics, it spans over a large portion of the frequency spectrum. Because our algorithm does not enforce long-distance correlations, the upper harmonics are not synchronized with the lower ones, thus creating another high-pitched voice speaking on its own. In the fire synthesis, this is also clearly noticeable when looking at impacts: since those short and sharp events span over most frequencies, the algorithm has trouble generating them and mostly manages to recreate impacts that only span over part of the frequency axis, resulting in less convincing synthesized textures. This could potentially be solved even when using untrained network by choosing bigger filters, which would then "see" larger chunks of the log-spectrogram at once.

3.2.3. Border effect

Our texture synthesis presents one intriguing property: at the start and end of the sound, the synthesized texture is identical to the original one (for instance, this is slightly visible at the start of the

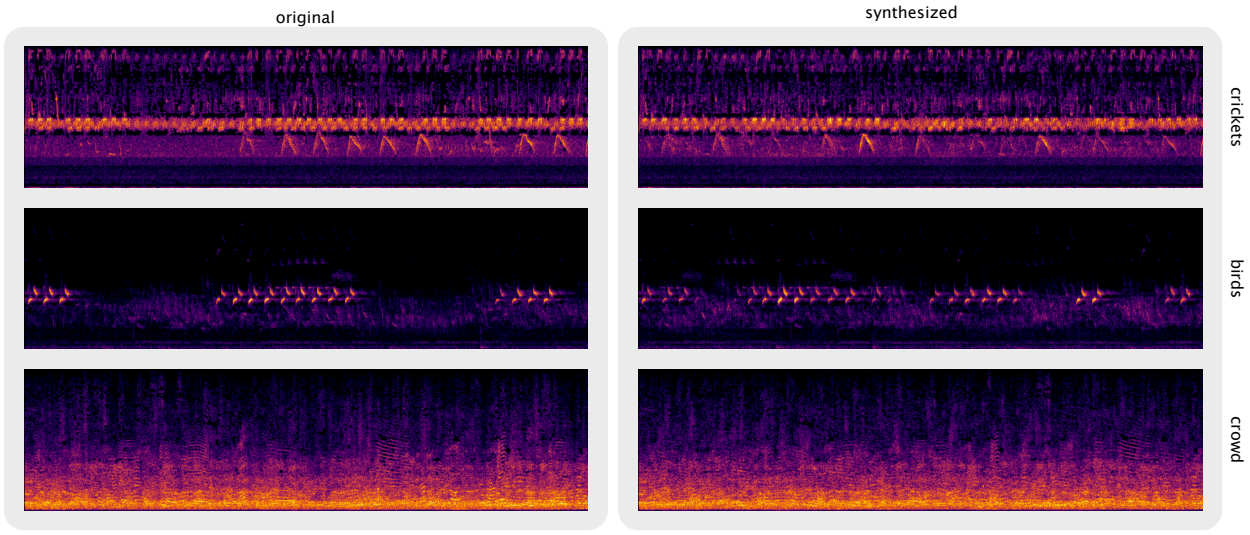


Figure 3: *Log-spectrograms of both originals textures (on the left) and synthesized textures (on the right) using the method presented in this article.*

log-spectrogram of the birds texture from Figure 3). In all of our synthesis, the leftmost and rightmost frames present exactly the same patterns in both original and synthesized textures. Although this effect is interesting, it dissipates quickly and only affects the time dimension: this means that even if we were to not get rid of this artifact, slightly cropping the start and end of the synthesized texture does completely discard it.

Gatys et al. [6] notice a similar effect in his visual texture synthesis where a distinctive part of the image is always reproduced around the same spot, and suggest that this effect originated from the zero-padding used in the convolution layers. To test this theory we experimented with synthesizing textures using only "valid" convolutions (i.e. without any padding): our results still presented the same border artifacts, which would indicate that they do not originate from the zero-padding. We do not have any alternative explanation to present at the moment.

It is worth noticing that this effect lasts around the length of the biggest filter used (in our case, 27 frames): for now, this means that we need to choose a filter size large enough to ensure the good reproduction of correlations between events, while small enough so that the border effect does not spoil too much of the interior of the synthesized texture.

3.2.4. Computation time

As mentioned in Section 2.1.3, computation time is for now far behind real-time since it takes around an hour to synthesize 12 seconds of audio with one GPU. While tedious for now, this process could potentially be alleviated by removing as much redundant information from the target parameters as possible (as of now, we have around $\sim$100M parameters in the parameter tensors H_k).

Using the same network as described in Section 2.1.2, we removed all cross-correlations between filters of different size from the parameter tensor: this dropped the number of parameters to $\sim$16M without altering the quality of synthesized textures. Another lead is to use a trained CNN instead of an untrained one, seeing as trained filters should prove efficient at describing patterns without needing to be as numerous as in an untrained CNN. We believe it should also be possible to drop the number of parameters even lower, for instance by using principal component analysis to select which cross-correlations need to be imposed over the synthesized signal as Gatys et al. [6] did.

Another potential lightening of the algorithm could come from changing signal representation: so far we have used log-spectrograms, but using another time-frequency representation could be greatly beneficial. For instance, using mel bands instead of the raw frequency bins of the spectrogram would reduce the number of parameters while staying perceptually sensible.

3.2.5. Extension

The basis for the texture synthesis having been set, it is also possible to develop on it and add ways of creatively alter the synthesized texture. An example of manipulation is the creation of an indefinitely long sound texture.

In order to do so, we use a principle resembling the "exquisite cadaver" game, where one has to continue the drawing of someone else while only seeing the border of the other's drawing. In our case we first synthesize an initial sound texture from a given target and copy the end of it onto the start of a white noise signal: we then perform another synthesis using this signal as initialization and keeping the same target texture. While doing this, we also prevent the optimization to be performed on the section that was copied from the previous synthesis. This results in a continuous texture seamlessly extending on the copied part, thus being able to perfectly follow where the previous synthesis left off. We only need to concatenate the newly generated texture to the previous one to create a longer sound texture. This process can obviously be repeated any number of times so as to obtain a texture of any desired length. One iteration of this process is shown on figure 4, while an example of such a synthesis is available online.[2]

[2]See `http://recherche.ircam.fr/anasyn/caracalla/dafx19/extended_synthesis/`

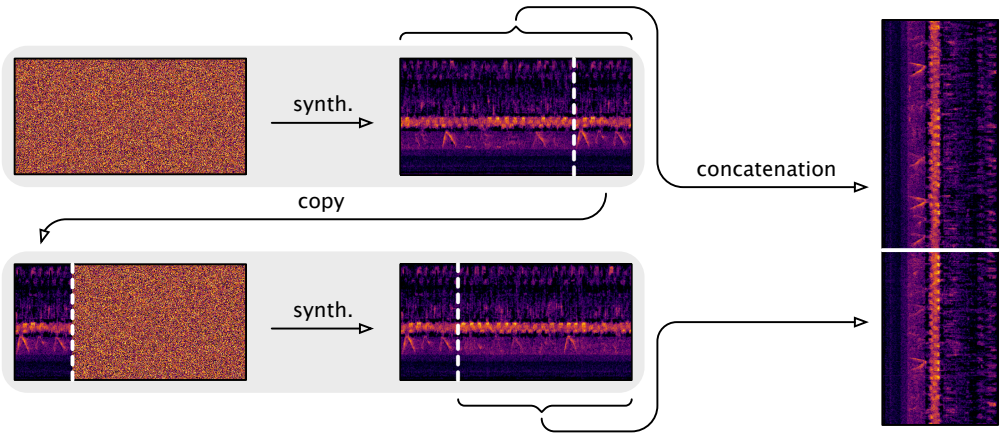


Figure 4: *Extension of a first synthesis by copying its end onto the start of a white noise signal. This signal is then used as initialization of another synthesis, while preventing the common part from being modified. The tiles are then concatenated to form a longer texture. Sound signals are represented by their log-spectrogram for explanation's sake.*

4. CONCLUSIONS

We introduced a new parametric texture synthesis based on the work of Gatys et al. [6] in visual textures: using the temporal cross-correlations between the feature maps of a CNN as parameters, we iteratively modify a sound signal until the values of its parameters reach those of a target texture. While the input of the CNN is the log-spectrogram of the sound, the optimization process is made directly in the time domain so as to avoid any phase recovery step in the synthesis.

The algorithm yields convincing results on a wide array of texture, even if they include singular events in the foreground. It can be straightforwardly applied without requiring the fine tuning of synthesis parameters from one texture to another. Its major flaws as of now lie in its long computation time and its trouble re-synthesizing correlations of events far apart in the log-spectrogram. A number of possible ways to address the first issue have been presented, for instance by subsampling the parameters tensor and altering the time-frequency representation. As for the second, the influence of the CNN architecture, and most notably the shape of its filters, are currently being investigated.

5. REFERENCES

[1] Nicolas Saint-Arnaud, *Classification of sound textures*, Ph.D. thesis, Massachusetts Institute of Technology, 1995.

[2] Diemo Schwarz, "State of the art in sound texture synthesis," in *Digital Audio Effects (DAFx)*, 2011, pp. 221–232.

[3] James F. O'Brien, Chen Shen, and Christine M. Gatchalian, "Synthesizing sounds from rigid-body simulations," in *ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2002, pp. 175–181.

[4] Diemo Schwarz, "Corpus-based concatenative synthesis," *IEEE signal processing magazine*, vol. 24, no. 2, pp. 92–104, 2007.

[5] Josh H McDermott and Eero P Simoncelli, "Sound texture perception via statistics of the auditory periphery: Evidence from sound synthesis," *Neuron*, vol. 71, no. 5, pp. 926–940, 2011.

[6] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge, "Texture synthesis using convolutional neural networks," in *Advances in neural information processing systems*, 2015, pp. 262–270.

[7] Dmitry Ulyanov and Vadim Lebedev, "Audio texture synthesis and style transfer," Available at https://dmitryulyanov.github.io/audio-texture-synthesis-and-style-transfer/, 2016.

[8] Joseph Antognini, Matt Hoffman, and Ron J. Weiss, "Synthesizing diverse, high-quality audio textures," *arXiv preprint arXiv:1806.08002*, 2018.

[9] Daniel Griffin and Jae Lim, "Signal estimation from modified short-time fourier transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 2, pp. 236–243, 1984.

[10] Eric Grinstein, Ngoc Q.K. Duong, Alexey Ozerov, and Patrick Pérez, "Audio style transfer," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 586–590.

[11] Shaun Barry and Youngmoo Kim, "Style transfer for musical audio using multiple time-frequency representations," 2018, Available at https://openreview.net/?id=BybQ7zWCb.

[12] Maciek Tomczak, Carl Southall, and Jason Hockman, "Audio style transfer with rhythmic constraints," in *Digital Audio Effects (DAFx)*, 2018, pp. 45–50.

[13] Ivan Ustyuzhaninov, Wieland Brendel, Leon A. Gatys, and Matthias Bethge, "Texture synthesis using shallow convolutional networks with random filters," *arXiv preprint arXiv:1606.00021*, 2016.

[14] Hugo Caracalla and Axel Roebel, "Gradient conversion between time and frequency domains using wirtinger calculus," in *Digital Audio Effects (DAFx)*, 2017, pp. 234–238.

[15] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal, "Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization," *ACM Transactions on Mathematical Software (TOMS)*, vol. 23, no. 4, pp. 550–560, 1997.

ASSISTED SOUND SAMPLE GENERATION WITH MUSICAL CONDITIONING IN ADVERSARIAL AUTO-ENCODERS

Adrien Bitton, Philippe Esling, Antoine Caillon, Martin Fouilleul

Institut de Recherche et Coordination Acoustique-Musique (IRCAM)
CNRS - UMR 9912, UPMC - Sorbonne Universite
1 Place Igor Stravinsky, F-75004 Paris, France
adrien.bitton@ircam.fr

ABSTRACT

Deep generative neural networks have thrived in the field of computer vision, enabling unprecedented intelligent image processes. Yet the results in audio remain less advanced and many applications are still to be investigated. Our project targets real-time sound synthesis from a reduced set of high-level parameters, including semantic controls that can be adapted to different sound libraries and specific tags. These generative variables should allow expressive modulations of target musical qualities and continuously mix into new styles.

To this extent we train *auto-encoders* on an orchestral database of individual note samples, along with their intrinsic attributes: note class, *timbre domain* (an instrument subset) and *extended playing techniques*. We condition the decoder for explicit control over the rendered note attributes and use *latent adversarial training* for learning expressive style parameters that can ultimately be mixed. We evaluate both generative performances and correlations of the attributes with the latent representation. Our ablation study demonstrates the effectiveness of the *musical conditioning*.

The proposed model generates individual notes as magnitude spectrograms from any probabilistic latent code samples (each latent point maps to a single note), with expressive control of orchestral timbres and playing styles. Its training data subsets can directly be visualized in the 3-dimensional latent representation. Waveform rendering can be done offline with the *Griffin-Lim algorithm*. In order to allow real-time interactions, we fine-tune the decoder with a pretrained magnitude spectrogram inversion network and embed the full waveform generation pipeline in a *plugin*. Moreover the encoder could be used to process new input samples, after manipulating their latent attribute representation, the decoder can generate sample variations as an *audio effect* would. Our solution remains rather light-weight and fast to train, it can directly be applied to other sound domains, including an *user's libraries* with *custom sound tags* that could be mapped to specific generative controls. As a result, it fosters creativity and intuitive audio style experimentations. Sound examples and additional visualizations are available on Github[1], as well as codes after the review process.

[1] https://github.com/acids-ircam/Expressive_WAE_FADER

1. INTRODUCTION

Modern music production techniques rely on large and heterogeneous sound sample libraries along with diverse digital instruments and effects. It opens to a great variety of sound design possibilities and limitless contents to compose with, however principled interactions and scaled visualisations are still lacking in order to efficiently explore such potential and use it to generate *target sound qualities*.

Unsupervised generative models learn an underlying data distribution solely based on the observation of examples, in order to consistently generate novel content. They have been successfully applied to complex computer vision tasks such as processing facial expressions, landscapes, visual styles and paintings. Some solutions to audio emerged more recently, including pioneer musical systems such as *NSynth* (Neural Synthesizer [1]) for real-time high-quality sound synthesis. However, the heavy model architecture and prohibitive training time restrict its dissemination. The learned internal representation remains mostly uninformative and its many generative parameters are still too little correlated to explicit semantic qualities.

In this paper, we develop a high-level sound synthesis system with meaningful data visualisations and explicit musical controls. It is a lighter *non-autoregressive model* that can be trained fast on small datasets, including an user's personal libraries. Our goal is to learn expressive style variables from any sound tags, so that the model fosters creativity and *assists digital interactions* in music production. Considering note samples of orchestral instruments, we could for instance synthesise novel timbres or *playing style hybrids*.

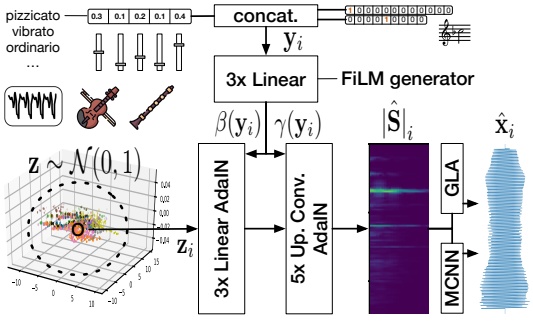


Figure 1: High-level note sample generation from the latent representation and musical conditioning in the decoder with FiLM. Intermediate features are modulated by the note targets and expressive style controls in order to synthesize new timbres and effects.

We train *Wasserstein Auto-Encoders* (WAEs [2][3]) on Mel- spectrogram magnitudes to organise a generative latent representation of individual note samples spanning the *tessitura* of 12 orchestral instruments. The considered database has intrinsic attributes: note classes, playing styles and timbres (each instrument subset), that we wish to control when generating new notes from the latent space. Thus we extend the WAE model with musical conditioning in the decoder and *Adaptive Instance Normalization* (AdaIN [4]). Using *Feature Wise Linear Modulation* (FiLM [5]) and adversarial training with a *Fader latent discriminator* [6], our WAE-Fader model effectively learns these generative controls along with expressive style variables that can be mixed continuously.

We evaluate these features in terms of generative performances and representation. We perform an *ablation study* and show that the model can sustain a good test reconstruction quality while achieving an accurate attribute-conditional generation. The success of the method relies on an attribute-free latent representation so that the decoder is pushed to learn the conditioning. These distributions can be visualized directly in the 3-dimensional latent space where clusters denote an undesired attribute encoding. We measure it with inter-class statistics and latent post-classification. The experiment validates correlations between low attribute encoding and effective conditioning.

We obtain an expressive note sample generator with *3-dimensional* representations of the training sound domains, decoding *probabilistic latent samples* with explicit control over the rendered note qualities. The learned style variables of the orchestra can ultimately be mixed continuously, as *faders* do, in order to intuitively explore new musical effects. Generated spectrogram magnitudes can approximately be inverted to waveform with the *Griffin-Lim* iterative algorithm (GLA [7]). Ultimately we fine-tune the decoder with a pretrained inversion network [8] for *real-time waveform synthesis*. We embed the resulting generative system in a *plugin* allowing for *MIDI* mapping, live exploration and *Digital Audio Workstation* (DAW) integration.

2. STATE-OF-ART

2.1. Generative models and regularized auto-encoders

Generative models aim to find the underlying probability distribution of the data $p(\mathbf{x})$ based on a set of examples in $\mathbf{x} \in \mathbb{R}^{d_x}$. To do so, we consider *latent variables* defined in a lower-dimensional space $\mathbf{z} \in \mathbb{R}^{d_z}$ ($d_z \ll d_x$), a higher-level representation that could have led to generate any given example. The latent variable generative model is defined by the joint probability distribution $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$, where the *prior* $p(\mathbf{z})$ is usually modelled with simpler distributions such as Gaussian or uniform while a complex conditional distribution $p(\mathbf{x}|\mathbf{z})$ maps latent codes to the data space. The model could be evaluated with the maximum marginal likelihood over the considered dataset. However for complex distributions that could model real-world data, integration cannot be computed in closed form.

Regularized auto-encoders have been used to reformulate the problem as an optimization by jointly learning the generative mapping $p_\theta(\mathbf{x}|\mathbf{z}) \in \mathcal{G}$ and an encoding distribution $q_\phi(\mathbf{z}|\mathbf{x}) \in \mathcal{Q}$ from families $\mathcal{G}$, $\mathcal{Q}$ of *approximate densities* both parameterized with neural networks. This was initially proposed through *Variational Inference* in the *Variational Auto-Encoder* (VAE [9]) that maximizes a lower bound of the data log-likelihood:

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\big[\log p_\theta(\mathbf{x}|\mathbf{z})\big] - D_{KL}\big[q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{z})\big] \leq \log p_\theta(\mathbf{x}) \quad (1)$$

This amounts to optimizing the *Evidence Lower Bound Objective* (ELBO) that can be interpreted as follow, the first term is the Negative Log-Likelihood (NLL) data reconstruction cost and the second is the Kullback-Leibler Divergence (KLD) that quantifies the error made by using the approximate $q_\phi(\mathbf{z}|\mathbf{x})$ rather than the true $p_\theta(\mathbf{z})$. This latent regularization pushes the encoder to remain close to the prior latent density and can be weighted with a β parameter that balances these two objectives.

$$\mathcal{L}^{\text{ELBO}}_{\theta,\phi} = -\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\big[\log p_\theta(\mathbf{x}|\mathbf{z})\big] + \beta \cdot D_{KL}\big[q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{z})\big] \quad (2)$$

The VAE is implemented with a *stochastic encoder* that parameterises an isotropic Gaussian latent distribution $q_\phi(\mathbf{z}|\mathbf{x})$ $\sim \mathcal{N}(\mu_\phi(\mathbf{x}), \sigma_\phi(\mathbf{x}))$ regularized against an unit variance prior. These assumptions allow analytical KLD computation and differentiable latent sampling for direct optimization of the ELBO.

The KLD forces each *individual* latent code to resemble the prior, which implicitly matches the whole encoded distribution. However a fitted ELBO value does not always result in an *effective inference*. Since the latent codes of different inputs are individually regularized, the KLD may prevent the encoder from learning any useful features (*posterior collapse* [10]) while the decoder only produces $p_\theta(\mathbf{x})$ regardless of the encoded information. Other conflicting solutions of the ELBO lead to undesired solutions and known limitations of VAEs such as *blurriness* of generated samples or *uninformative latent dimensions* ([11]).

With justifications stemming both from *Likelihood-free Optimization* (InfoVAE [3]) and the theory of *Optimal Transport* (WAE [2]), a more general framework for training regularized auto-encoders was recently proposed and that we call Wasserstein Auto-Encoders (WAEs). Considering a *deterministic decoder* $G_\theta : \mathbf{z} \to \mathbf{x}$ and any family of conditional encoder distribution $Q_\phi(\mathbf{z}|\mathbf{x}) \in \mathcal{Q}$, it is sufficient that the marginal $Q_Z(\mathbf{z}) := \mathbb{E}_X\big[Q(\mathbf{z}|\mathbf{x})\big]$ matches any prior P_Z. In comparisons with VAEs, WAEs can optimize any non-negative cost function C and *any divergence measure* D_Z between latent distributions, without requiring a stochastic encoder nor restricting the latent model to Gaussian prior:

$$\mathcal{L}_{\text{WAE}} := \inf_{Q(\mathbf{z}|\mathbf{x})\in\mathcal{Q}} \mathbb{E}_X \mathbb{E}_{Q(\mathbf{z}|\mathbf{x})}\big[C(\mathbf{x}, G(\mathbf{z}))\big] + \beta \cdot D_Z(Q_Z(\mathbf{z}), P_Z) \quad (3)$$

Thus we set our experiment in the more flexible WAE framework. These regularized auto-encoders are powerful unsupervised representation learning models, rather light-weight and fast to train, performing both inference (encoder) and generation (decoder). They are effective on small datasets (hundreds of training examples), learning a structured latent representation with disentangling capacities encouraged when $\beta > 1$. Once trained, probabilistic samples of the latent prior are consistently decoded into new samples and latent interpolations map to smooth data variations.

2.2. Maximum Mean Discrepancy Regularization

As shown for VAEs, the choice of *latent divergence* heavily impacts the resulting model performances. Since the point-wise KLD has strong intrinsic limitations, a more flexible regularization is required for WAEs. Such *differentiable* divergence on latent distributions was developed in the *Reproducing Kernel Hilbert Space* (RKHS) as a distance between probabilistic moments $\mu_{p,q}$ computed with a non-parametric kernel k:

$$\begin{aligned} ||\mu_p - \mu_q||^2_H &= \langle \mu_p - \mu_q, \mu_p - \mu_q \rangle \\ &= \mathbb{E}_{p,p}k(x,x') + \mathbb{E}_{q,q}k(y,y') - 2\mathbb{E}_{p,q}k(x,y) \end{aligned} \quad (4)$$

It defines the *Maximum Mean Discrepancy* (MMD [12]) between two distributions $x \sim p(x)$ and $y \sim q(y)$, where $\mathbb{E}_{p,q}$ is the expectation that can be evaluated with the *Radial Basic Function* (RBF) kernel of free parameter Σ:

$$k_{\text{(RBF)}}(x, y) = \exp\left(\frac{||x - y||^2}{-2\Sigma^2}\right) \tag{5}$$

To the extent of latent regularization, MMD can be computed between every deterministic mini-batch encoding $\mathbf{z}_{\text{encoder}} = Q(\mathbf{x})$ and random samples from any latent prior $\mathbf{z}_{\text{prior}} \sim P_Z$. Throughout the model optimization, MMD is thus matching the aggregated encoder posterior to the prior rather than regularizing each latent point individually. In comparisons with KLD, WAE-MMD allows for less constrained inference and richer latent representations. For instance, increasing β to two orders of magnitude above the reconstruction cost does not impede the decoder training. Since the WAE objective does not optimize the bounded NLL, the overall generative performances can be improved.
Other kernel functions can be used, which may be more discriminating at the expense of heavier computations. Alternatively to MMD, the WAE-GAN uses an adversarial latent discriminator to assess the divergence, thus optimizing a parametric function that could match even closer the encoder to the prior. However, since we consider a low-dimensional latent space of only 3 dimensions and remain with a simple isotropic Gaussian prior, MMD-RBF is sufficient yet light and stable to train on.

2.3. Conditioning and feature normalizations

Regularization in auto-encoders encourages disentanglement of independent generative factors onto separate latent dimensions that would in turn control the corresponding decoded data variations. However this is only partly achieved on toy datasets (β-VAE [13]) and in most cases the unsupervised latent dimensions are hardly related to explicit generative parameters. An additional supervision signal may be applied to the generative neural network in order to control and render specific attributes of the data. Thus we consider observations $\mathbf{x}$ paired with attribute annotations $\mathbf{y}$, and condition the decoder as $G : \{\mathbf{z}, \mathbf{y}\} \rightarrow \mathbf{x}$.
The simplest conditioning for categorical attributes is to encode them into one-hot vectors that are concatenated to the latent codes before being processed by the decoder. However more advanced conditioning techniques have been developed as for visual style transfer, using full images as conditions (conditional style transfer [14]). In the *Feature-wise Linear Modulation* (FiLM [5]) approach, a separate generator learns a mapping from any style inputs to adaptive biases $\beta_{\text{FiLM}}(\mathbf{y})$ and scales $\gamma_{\text{FiLM}}(\mathbf{y})$ applied to the conditional network computations. This modulation may be placed anywhere within the architecture and proved to be particularly suited to *Adaptive Instance Normalization* (AdaIN [4]). Considering the *l-th* hidden layer output activations $\mathbf{h}^l = g^l(\mathbf{h}^{l-1})$ of a generative neural network, the conditional modulation is thus be computed as:

$$\text{AdaIN}(\mathbf{h}^l, \mathbf{y}) = \gamma^l_{\text{FiLM}}(\mathbf{y})\left[\frac{\mathbf{h}^l - \mu(\mathbf{h}^l)}{\sigma(\mathbf{h}^l)}\right] + \beta^l_{\text{FiLM}}(\mathbf{y}) \tag{6}$$

in which mean and standard deviation $\{\mu,\sigma\}$ are computed across features, independently for each channel and each sample. In the context of style transfer, it can be interpreted as aligning the mean and variance of the content features with those of the style condition. It is a versatile conditioning technique, requiring little additional computations (particularly when applied channel-wise in convolution layers). It also suits well to handling multiple conditions that may more efficiently be mapped throughout the network rather than arbitrarily concatenated to the input. Thus we will use FiLM and AdaIN for conditioning the decoder on both note and style classes. However, such normalization is not suited to classification tasks since content features are individually normalized. In order to preserve its inference power, we will use *Batch Normalization* (BN) on the encoder's hidden activations.

2.4. Adversarial latent training

Adversarial regularization was proposed as an alternative to MMD in the WAE-GAN. For simple low-dimensional latent distributions, the expense of an additional parametric adversarial regularizer is not required. Nonetheless, adversarial latent training remains relevant for expressive conditioning. As detailed in the previous section, adaptive conditioning techniques paired with specific feature normalizations substantially improved feed-forward style transfer. However, in an auto-encoder setting, if the latent space implicitly encodes the attributes of interest, the decoder bypasses the conditioning and does not learn any effective generative controls. This problem was tackled in image generation with the introduction of an adversarial *Fader* latent discriminator $\mathcal{F}$ (Fader Networks [6]) that competes with the non-conditional encoder in order to prevent correlations between attributes and latent distributions. As for the conditional models, we consider annotated data samples $\{\mathbf{x}, \mathbf{y}\}$ and for simplicity, a categorical one-hot representation $\mathbf{y} \in \{0, 1\}^n$ with a single $y_i = 1$ and its opposite $\bar{\mathbf{y}} := \mathbf{1}^n - \mathbf{y}$. Such attribute-free latent representation is implemented in two separate optimization steps, first latent classification of the true attribute $\mathcal{F} : \mathbf{z} \rightarrow \hat{\mathbf{y}} \sim p_\psi(\mathbf{y}|Q(\mathbf{x}))$, then adversarial confusion of the latent classifier at predicting the opposite:

$$\begin{aligned} \mathcal{L}^{\text{class.}}(\psi|\phi) &= -\sum_{\mathbf{x},\mathbf{y}} \log(p_\psi(\mathbf{y}|Q(\mathbf{x}))) \\ \mathcal{L}^{\text{adv.}}(\phi|\psi) &= -\sum_{\mathbf{x},\mathbf{y}} \log(p_\psi(\bar{\mathbf{y}}|Q(\mathbf{x}))) \end{aligned} \tag{7}$$

As the encoder is pushed to remain invariant to attributes, the decoder is forced to learn the conditioning in order to reconstruct every input samples along with their source attributes. Thus it replaces adversarial training in the high-dimensional pixel space with latent attribute confusion in the low-dimensional latent space in order to efficiently learn style transfer variables. Applied to facial expressions, these *Fader* variables can continuously modulate complex visual features such as gender (female $\leftrightarrow$ male) or age (younger $\leftrightarrow$ older). Moreover, in mixing several attributes, one could generate new style qualities.

2.5. Audio synthesis

Neural networks can be trained on spectrogram magnitudes (and other spectral features) for audio analysis purpose. It eases the subsequent modelling task, often involving pattern detection, from a pre-processed structured sound representation. However, for generative purpose, an inversion from magnitude to waveform is required since the complex phase information was discarded. It is

commonly done offline with GLA [7]. Further advances in generative neural networks for audio have targeted raw waveform modelling through specific architecture design. *Wavenet* [15][1] is amongst them the most popular solution. It uses several stacks of dilated causal convolutions in order to aggregate multiple temporal granularities and structure long-term dependencies, which is challenging at the high audio sample rate. The output is a single auto-regressive sample prediction given all the previous sample context $p(x_t|x_1..x_{t-1})$. It results in high-quality real-time audio synthesis. However this sample level modelling requires long training times, heavy architectures that offer little knowledge over their learned features.

The *Multi-head Convolutional Neural Network* (MCNN [8]), a recent alternative for audio waveform modelling, was designed as a feed-forward real-time magnitude spectrogram inversion system that is not restricted to linear frequency scale. It proved to outperform GLA quality for speech. The use of differentiable GPU-based STFT computations enables a faster optimization onto spectral losses, rather than *auto-regressive* sample predictions:

$$\begin{aligned} \mathbf{x} \xrightarrow{|\text{STFT}|} |\mathbf{S}| \xrightarrow{\text{MCNN}} \hat{\mathbf{x}} \xrightarrow{\text{STFT}} \hat{\mathbf{S}} \Longrightarrow \mathcal{L}^{\text{MCNN}}(\mathbf{S}, \hat{\mathbf{S}}) \\ \mathcal{L}^{\text{MCNN}} = \lambda_0 \cdot \mathcal{L}^{\text{SC}} + \lambda_1 \cdot \mathcal{L}^{\text{logSC}} + \lambda_2 \cdot \mathcal{L}^{\text{IF}} + \lambda_3 \cdot \mathcal{L}^{\text{WP}} \end{aligned} \tag{8}$$

where $|\mathbf{S}|$ can be any spectrogram magnitude (including Mel-scaled frequencies). The model is well tailored to audio with multiple heads of 1-dimensional temporal up-sample convolutions. These heads focus on different spectral components and sum into waveform. It remains light-weight and could be adapted in an end-to-end waveform auto-encoder. Four objectives were originally proposed, using the complex STFT for the *Instantaneous Frequency* (IF) and *Weighted Phase* (WP) losses, that we could not optimize successfully. Hence we will only use the *Spectral Convergence* (SC) and log-scale magnitude (logSC) losses:

$$\mathcal{L}^{\text{SC}}(\mathbf{S}, \hat{\mathbf{S}}) = \||\mathbf{S}| - |\hat{\mathbf{S}}|\|_F / \||\mathbf{S}|\|_F \text{ with } \|.\|_F \text{ the Frobenius norm}$$

$$\mathcal{L}^{\text{logSC}}(\mathbf{S}, \hat{\mathbf{S}}) = \|\log(|\mathbf{S}| + \epsilon) - \log(|\hat{\mathbf{S}}| + \epsilon)\|_1 \tag{9}$$

3. METHOD

Our experiment begins with the WAE-MMD, isotropic unit variance Gaussian prior $\mathbf{z}_{\text{prior}} \sim \mathcal{N}(0, 1)$, RBF kernel and BN in both encoder and decoder in order to structure a 3-dimensional generative latent sound representation. Given a magnitude spectrogram $|\mathbf{S}|$ and a corresponding set of annotated attributes $\mathbf{y}$, we are learning $Q : |\mathbf{S}| \rightarrow \mathbf{z}$ and $G : \mathbf{z} \rightarrow |\hat{\mathbf{S}}|$ such as $|\mathbf{S}| \approx |\hat{\mathbf{S}}| = G(Q(|\mathbf{S}|))$ with *Binary Cross-Entropy* (BCE) reconstruction cost:

$$\begin{aligned} \mathcal{L}_{\text{WAE}} = \text{BCE}(|\mathbf{S}|, |\hat{\mathbf{S}}|) + \beta \cdot \text{MMD}_{\text{RBF}}(\mathbf{z}, \mathbf{z}_{\text{prior}}) \\ \text{BCE}(x, \hat{x}) = -\big[x \log \hat{x} + (1 - x) \log(1 - \hat{x})\big] \; ; |x| < 1 \end{aligned} \tag{10}$$

We can sample random codes from the latent prior and consistently decode new magnitude samples, however there is no control on the output features. For the orchestra, we consider $\mathbf{y} = \{\mathbf{y}_{\text{note}}, \mathbf{y}_{\text{style}}\}$ with $\mathbf{y}_{\text{note}} = \{\text{semitone, octave}\}$. We define the timbre attribute as the class of an instrument subset, which comprises the *Ordinario* mode as well as diverse extended playing techniques such as *Staccato*, *Flatterzunge* or *Pizzicato*. When considering a single subset, we thus aim at controlling the playing techniques of the considered instrument as $\mathbf{y}_{\text{style}}$. When considering multiple instruments,

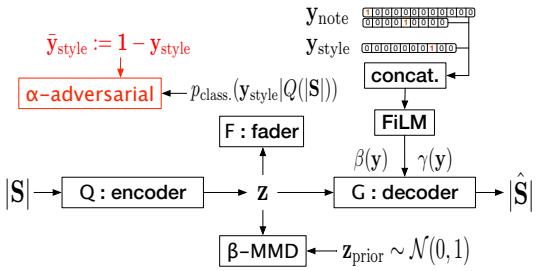


Figure 2: How information flows in the adversarial optimization of the WAE-Fader

instead we aim at controlling the different timbres, either in *Ordinario* or with mixed playing styles within each instrument subset. For explicit controls over the rendered attributes, we condition the decoder as $G : \{\mathbf{z}, \mathbf{y}\} \rightarrow |\hat{\mathbf{S}}|$ using AdaIN. An additional FiLM generator is fed with concatenated one-hot vectors of the three attribute classes (semitone, octave and style). It learns an adaptive mapping to biases $\beta_{\text{FiLM}}(\mathbf{y})$ and scales $\gamma_{\text{FiLM}}(\mathbf{y})$ that are used to modulate the normalized decoder activations. In order to effectively learn the style conditioning and expressively modulate timbres or playing techniques, we use adversarial training with a Fader latent discriminator $\mathcal{F} : \mathbf{z} \rightarrow \hat{\mathbf{y}}_{\text{style}} \sim p_{\text{class.}}(\mathbf{y}_{\text{style}}|Q(|\mathbf{S}|))$ that competes with the non-conditional encoder in classifying the considered styles from latent codes:

$$\begin{aligned} \mathcal{L}_{\text{class.}} &= -\log p_{\text{class.}}(\mathbf{y}_{\text{style}}|Q(|\mathbf{S}|)) \\ \mathcal{L}_{\text{WAE-Fader}} &= \mathcal{L}_{\text{WAE}} - \alpha \cdot \log p_{\text{class.}}(\bar{\mathbf{y}}_{\text{style}}|Q(|\mathbf{S}|)) \end{aligned} \tag{11}$$

with $\bar{\mathbf{y}}_{\text{style}} := \mathbf{1} - \mathbf{y}_{\text{style}}$ and α that weights the adversarial loss in the encoder. Classification is optimized on the NLL with *Softmax* probabilities. The resulting attribute confusion prevents the latent space from implicitly encoding the style distributions, thus the decoder is forced to use the conditioning to reconstruct the source features from the attribute-free code. Ultimately these learned style variables can continuously be mixed, as actual *faders* do. We refer to this final model as WAE-Fader, that still uses MMD regularization. It allows for controlling the strength of each rendered attribute and intuitively exploring hybrid sound effects from any custom tags, here either chosen from extended playing techniques or from diverse orchestral timbre domains.

The resulting generative system maps any latent coordinate $\mathbf{z} \sim \mathcal{N}(0, 1) \in \mathbb{R}^3$ to target note spectrograms with expressive musical style controls. Inversion from spectrogram magnitudes to audio waveforms can be done offline with GLA. Alternatively, we pre-train a MCNN on a larger corpus of musical note samples to allow real-time rendering. In order to improve the final audio quality, we fine-tune the full generative model by freezing the encoder parameters and jointly optimizing the learned decoder with the pretrained MCNN as:

$$\mathbf{x} \xrightarrow{\text{STFT}} \mathbf{S} \xrightarrow{|\text{Mels}|} |\mathbf{S}| \xrightarrow{Q} \mathbf{z} \xrightarrow{G \circ \text{MCNN}} \hat{\mathbf{x}} \xrightarrow{\text{STFT}} \hat{\mathbf{S}} \Longrightarrow \mathcal{L}^{\text{MCNN}}(\mathbf{S}, \hat{\mathbf{S}}) \tag{12}$$

This waveform pipeline $\{G \circ \text{MCNN}\}$ is embed in a plugin for live interactions and DAW integration. Using a MIDI interface, we can for instance trigger target note classes $\mathbf{y}_{\text{note}}$ with keys and map the continuous generative parameters to faders. These are the latent dimensions $\mathbf{z}$, that can also be randomly sampled, and most interestingly the adversarially learned style variables $\mathbf{y}_{\text{style}}$ that can be mixed to explore new sound effects.

4. EXPERIMENT

4.1. Dataset

We use the Studio-On-Line (SOL [16]) library of around 15000 individual note samples, across the tessitura of 12 orchestral instruments grouped in 4 families and with many extended playing techniques, that may be specific or shared across instrument families. These are *Wind* (Alto-Saxophone, Bassoon, Clarinet, Flute, Oboe), *Brass* (English-Horn, French-Horn, Tenor-Trombone, Trumpet), *String* (Cello, Violin) and *Keyboard* (Piano). Notes are consistently tagged with the intrinsic attributes of the dataset: note classes (12 semitones across 9 octaves), several dynamics and playing styles of every instrument. We define two style experiments for the orchestra. If training on a single instrument, we aim for expressive synthesis of its playing styles. If training on multiple instruments, we aim for timbre control. Each instrument subset defines a timbre domain, either in *Ordinario* (its common mode) or with all styles mixed.
Audio files are down-sampled to 22050Hz and pre-processed into Mel-spectrograms with a FFT size of 2048, hop size of 256 and 500 bins ranging the full spectrum. As we consider a generator of *individual* notes, we set a common audio length of 34560 samples (∼1.6s) from the attack which amounts to 128 STFT frames. We choose this duration as a trade-off between input and latent dimensionality, limiting the amount of silence after shorter playing modes (eg. *Pizzicato*) while keeping some sustain for longer notes (from which some sustain and decay may have been cropped). Magnitudes are floored to 1e-3 and log-scaled in [0,1] according to the BCE range. Each playing style subset of each instrument is split into 80% training, 10% validation and 10% test notes. In average each instrument has 10 playing styles and 100 to 200 notes for each.

4.2. Implementation details

Architecture of the WAE-Fader: Our experiments have been implemented in the *PyTorch* environment and our codes will be shared with this dependency. All convolution layers use 2-d. square kernels, an input zero-padding of half the kernel size and are followed by 2-dimensional feature normalization. All fully-connected linear layers are followed by 1-dimensional feature normalization. The non-linear activation used after every normalization is CELU. The deterministic encoder has 5 convolution layers with [12, 24, 48, 96, 128] output channels, kernel size 5 and stride 2, that down-sample the input spectrograms into 128 output maps that are flattened into an intermediate feature vector of size 8192. It is followed with a bottleneck of 3 linear layers of output sizes [1024, 512, 3] mapping to the latent space. For input Mel-spectrograms of size (500,128), it amounts to a dimensionality reduction of more that 5 orders of magnitude. All normalizations are BN. The decoder mirrors this structure with 3 linear layers of output sizes [512, 1024, 8192]. This vector is then reshaped into 128 maps. To avoid the known *checkerboard artifacts* [17] of the transposed convolution, we use *nearest neighbor* up-sampling followed with convolution of stride 1. These maps are processed with 4 up-sampling of ratios 3, the last one directly mapping to the input dimensionality of (500,128), and 5 convolutions with [96, 48, 24, 12, 1] output channels and kernel sizes [5, 5, 7, 9, 7]. All normalizations are AdaIN and the decoder output activation is sigmoid, bounded in [0,1] according to the BCE range. The FiLM conditioning is applied feature-wise at the output of the first two linear layers and channel-wise after. It amounts to 3688 modulation weights computed by an additional FiLM generator of 3 linear layers of output sizes [512, 1024, 3688]. Its output is split into biases and scales of sizes [512, 1024, 128, 96, 48, 24, 12]. The Fader latent discriminator has 3 linear layers of output sizes [1024, 1024, n_{style}] with *LeakyReLU* activations and a dropout ratio of 0.3, mapping latent codes to probabilities of the n_{style} classes.

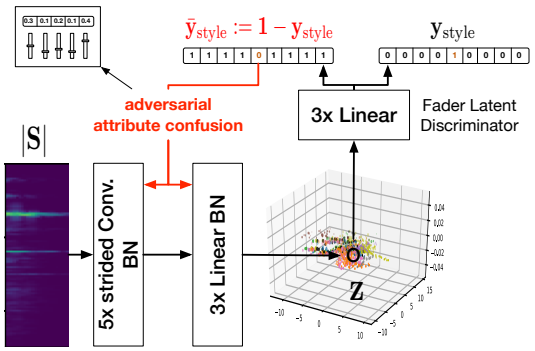


Figure 3: The Fader latent discriminator tries to infer the true style attribute while the encoder adversarially aims at fooling it. It encourages attribute invariance in the latent representation and learning of continuous generative controls in the decoder.

Training parameters: We train our models with the Adam optimizer, an initial learning rate of 5e-4 and a batch size of 90. All model weights are initialized with Xavier uniform distribution. Depending on the considered data subset size, between 200 and 800 epochs are needed. A single instrument (1000-1500 notes) can be modelled in less than 2 hours on one NVIDIA TITAN Xp GPU. Training over all instruments and styles at once (around 11000 notes) takes less than 12 hours. In the first part of the training (30 to 100 epochs), we only optimize the reconstruction and classification objectives. Then we gradually introduce the MMD regularization (β-warmup) and the adversarial feedback in the encoder (α-warmup) until the first half of training epochs. The rest of the training jointly optimizes all training objectives at their target strengths $\beta = 40$ and $\alpha = 4$. These value were estimated in order to approximately balance the gradient magnitudes back-propagated by each loss. However, for the baseline WAE-MMD models we warmup β to 500 since it does not prevent from optimizing the reconstruction cost.

Signal reconstruction: The above described model trains on inputs with 128 frames of Mel-spectrogram, which amount to 34560 waveform samples according to our STFT settings. The generated Mel-magnitudes can be approximated back to the linear frequency scale and iteratively inverted with GLA for 100 to 300 iterations. To allow real-time rendering and a possibly improved audio quality, we reproduce the original MCNN architecture for Mel-spectrogram magnitudes inversion. We use 8 heads, $\lambda_0 = 1$ and $\lambda_1 = 6$. We could not successfully optimize the complex losses, however, we compute these magnitude losses on both the linear and Mel frequency scales. We pretrain this model on a larger dataset of around 50 hours audio comprising SOL and subsets of the *Vienna Symphonic Library* (VSL). Ultimately, we fine-tune the trained decoder with this pretrained MCNN. To do so, we freeze

the encoder weights and optimize $G \circ \text{MCNN}$ on the model train set. In equation (12), the auto-encoder pair G, Q maps to Mel-spectrogram magnitudes $|\mathbf{S}|$ which are inverted to signals by the MCNN. However, the loss computation $\mathcal{L}^{\text{MCNN}}(\mathbf{S}, \hat{\mathbf{S}})$ is not necessarily restricted to this frequency scale. Thus we evaluate and sum $\mathcal{L}^{\text{SC}}$ $\mathcal{L}^{\text{logSC}}$ from equation (9) on both linear and Mel frequency scaled magnitudes.

classified attribute (n_{style})	train set	validation	test
Semitone (12)	1.00	0.99	0.99
Octave (9)	1.00	0.99	1.00
Ordinario timbres (12)	1.00	1.00	1.00
Extended timbres (12)	1.00	1.00	1.00
Violin playing styles (10)	1.00	0.97	0.95
Clarinet playing styles (10)	1.00	0.96	0.94
Piano playing styles (10)	1.00	0.92	0.95
Trumpet playing styles (10)	1.00	0.92	0.94
Alto-Saxophone pl. styles (10)	1.00	0.98	1.00
Tenor-Trombone pl. styles (11)	1.00	0.90	0.90

Table 1: Reference **F1-scores** of the pretrained data classifiers used for the evaluation of conditional note generations

4.3. Evaluations

Generative performances: First, we evaluate the ability of our models to produce accurate spectrograms by computing the reconstruction scores on the test set with *Root-Mean Squared Error* (RMSE) and *Log-Spectral Distance* $\text{LSD} = \sqrt{\sum \left[10 \log_{10}(|\mathbf{S}|/|\hat{\mathbf{S}}|)\right]^2}$. Regarding the conditioning aspects, we first pretrain data classifiers to reliably discriminate the different attribute classes and report their performances in Table 1. These classifiers share the same architecture as the encoder but map to the n_{style} classes of interest. We use them as references to evaluate the effectiveness of the conditioning. Then, we sample an evaluation batch of 1000 random latent points from the prior, along with random semitone and octave targets. This evaluation batch is decoded to each attribute of the model (either playing styles or timbres) and classified with the corresponding reference classifier. A high accuracy means an effective conditioning for the task of musical note generation. We report the average accuracy for all the target conditions, with random octaves both in [0-8] (full orchestral range) or in [3-4] where models train on the overlap of every instrument tessitura.

Latent space structure: The effectiveness of the conditioning relies on learning an attribute-free latent representation of the data. If the attribute distributions are clustered, the decoder may learn their correlations with latent dimensions and bypass the conditioning signal. This phenomenon is alleviated with adversarial training of the non-conditional encoder against a Fader latent discriminator. As we map to 3-dimensional spaces, we can directly visualize this latent organization. We also propose two evaluations of the attribute representations. First, we compute the average inter-class latent statistics with MMD. In this case, low values mean that the attribute distributions blend in the final representation. Second, we also perform a post-classification task by training classifiers at predicting the attributes from the learned latent representation. These models use the same architecture as the Fader discriminator, and we report their final accuracy. In this case, low scores mean that the latent representation did not encode the attributes.

model	test rec.		note cond. acc.				style cond. acc.	
	MSE	LSD	st._{34}	oct._{34}	st._{08}	oct._{08}	style_{34}	style_{08}
Violin playing styles (n_{style}=10) 1475 training note samples								
WAE-MMD	0.76	68.2	NA	NA	NA	NA	NA	NA
WAE-note	0.69	55.4	0.73	0.72	0.47	0.43	NA	NA
WAE-style	0.74	59.6	0.47	0.39	0.30	0.22	0.20	0.17
WAE-Fader	0.80	91.1	**0.96**	**0.77**	0.97	0.48	**0.88**	**0.93**
Ordinario timbres (n_{style}=12) 1784 training note samples								
WAE-MMD	1.04	88.3	NA	NA	NA	NA	NA	NA
WAE-note	0.84	71.6	0.99	0.96	0.62	0.53	NA	NA
WAE-style	0.80	65.7	0.64	0.58	0.30	0.24	0.33	0.19
WAE-Fader	1.01	105	**1.00**	**1.00**	0.94	0.68	**0.95**	**0.70**
Extended timbres (n_{style}=12) >11000 training note samples								
WAE-MMD	0.93	175	NA	NA	NA	NA	NA	NA
WAE-note	0.69	173	0.99	0.98	0.72	0.64	NA	NA
WAE-style	0.65	172	0.84	0.83	0.44	0.39	0.61	0.34
WAE-Fader	1.32	182	**1.00**	**1.00**	0.90	0.71	**0.95**	**0.64**

Table 2: The ablation study confirms the effectiveness of the **WAE-Fader** conditioning, both on target notes and playing styles or timbres. The conditional latent sampling is either performed with random octaves in [3,4] (the overlap of every tessitura) and [0-8] (the full orchestra range), we report the accuracy of the conditioning with respect to the targets $\text{note}_{34,08}$ (st. is semitone classe and oct. is octave classe) and $\text{style}_{34,08}$.

5. RESULTS

5.1. Ablation study

We defined both generative and representation evaluations to assess the effectiveness of our proposed musical conditioning. To study the benefits and compromises of each model feature, we train the base WAE-MMD and compare it with ablations of the WAE-Fader. The incremental model comparisons are WAE-MMD (no conditioning), WAE-note (semitone and octave conditioning), WAE-style (note and style conditioning) and WAE-Fader. In order to simplify the notation, we do not specify the MMD but this regularization is used for all models. We performed this ablation study on the violin subset that has the following annotated playing styles: *Ordinario, Sustained, Short, Non-vibrato, Staccato, Pizzicato-secco, Medium-vibrato-short, Tremolo, Medium-vibrato-sustained* and *Pizzicato-l-vib*. We also compare the WAE-Fader on instrument timbres, either in ordinario or for all extended techniques mixed per instrument subset. Table 2 reports the successive generative performances of the models. Table 3 reports the latent evaluations, showing how the conditioning tasks are reflected in the learned representations. It confirms the effectiveness of the expressive conditioning when the attribute-invariance assumption is achieved.

As we can see, conditioning WAE-note on the semitone and octave classes shows that the WAE-MMD model can partly learn the note controls with FiLM conditioning. Accordingly, the latent space structure does not exhibit strong correlations with the note classes anymore but with the style attributes that become the main unsupervised data feature. We also notice that this additional supervision improves the reconstruction quality. However, when adding the style conditioning in WAE-style, it seems that most performances drop. Indeed, the overall conditioning becomes little effective, both for the target note and style conditions. The final

model	inter-class MMD			post-class. acc.		
	st.	oct.	style	st.	oct.	style
Violin playing styles (n_{style}=10) 1475 training note samples						
WAE-MMD	0.25	0.26	0.30	0.92	0.94	0.56
WAE-note	0.03	0.10	0.50	0.04	0.49	0.82
WAE-style	0.12	0.16	0.35	0.25	0.55	0.59
WAE-Fader	0.02	0.01	0.46	0.08	0.34	0.64
Ordinario timbres (n_{style}=12) 1784 training note samples						
WAE-MMD	0.12	0.38	0.28	0.75	0.87	0.59
WAE-note	0.02	0.28	0.51	0.33	0.57	0.83
WAE-style	0.04	0.40	0.35	0.13	0.60	0.63
WAE-Fader	0.33	0.08	0.03	0.17	0.25	0.23
Extended timbres (n_{style}=12) >11000 training note samples						
WAE-MMD	0.02	0.41	0.12	0.71	0.89	0.48
WAE-note	5e-3	0.30	0.22	0.07	0.49	0.71
WAE-style	4e-3	0.32	0.18	0.07	0.56	0.55
WAE-Fader	3e-3	0.20	0.11	0.11	0.46	0.43

Table 3: The ablation study allows to monitor the latent organization in the different models and throughout their training, as shown in Figure 4. We use both inter-class statistics and latent post-classification to estimate the final attribute invariance in the learned representation.

results show that the adversarial latent training enables the WAE-Fader model to effectively learn the complete conditioning, at the expense of a possible drop in its reconstruction accuracy.
It also seems that the task of modelling the playing styles when learning on a single instrument is more challenging than changing the timbres across multiple instruments. This can be seen in the lower performance of the WAE-style model applied to the violin. This may also be explained by the reduced size of the training data when the learning is restricted to single instrument subsets. These observations are supported by the resulting audio outputs of the conditional note generations. Indeed, it appears that meaningful and expressive variations when switching to any attribute conditions are only achieved with our proposed WAE-Fader model. This is successful for conditioning applied on both timbre attributes or playing styles.

5.2. Expressive note sample generations

In this section, we report additional experiments on the WAE-Fader models when conditioned on the playing styles of different instruments and families. As shown in Table 4, our model seems to train successfully on playing styles in every instrument families, as well as across the 12 instrument timbres of the orchestra as shown in the previous ablation study. This amounts to a great variety of sound qualities spanning extended modes of the orchestra, and let us hypothesize that the model could be applied to other sound domains as long as the tags are consistent with the data. Furthermore, the style variables learned with the Fader latent discriminator are continuous independent controls that can be mixed. Hence, this can allow our system to modulate the strength of rendered styles and create new effects by combining multiple attributes. Our model can also be used for sample modifications, akin to traditional audio effects, by encoding a given sample and manipulating the attribute conditions in order to decode different sample transformations.

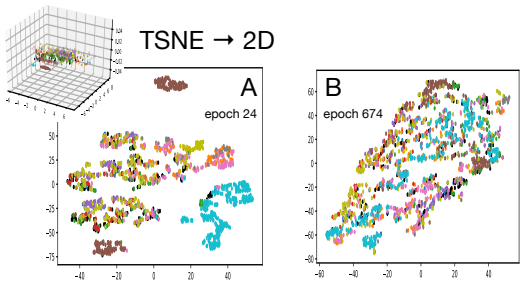


Figure 4: Latent organization as the WAE-Fader model trains on the ordinario timbres, each instrument domain being represented by a separate color. In ***A***, at epoch 24, the encoder does not optimize the adversarial loss yet. Its unsupervised representation exhibits the attribute classes. In ***B***, at epoch 674, the α-warmup is finished and the adversarial latent training had blended the attribute distributions. The 2-dimensional projections are performed with t-Distributed Stochastic Neighbor Embedding (TSNE).

5.3. Audio outputs and plugin development

As discussed previously, our proposed models can generate magnitude spectrograms, while controling their expressive qualities. These spectrograms can be either inverted to waveform offline with GLA or real-time if paired with MCNN. When fine-tuning the learned decoders with the pretrained MCNN on magnitude losses, we obtain a quality almost equivalent to the GLA approximation. We provide audio examples of test set reconstructions and conditional note generations inverted with both GLA and MCNN for individual listening evaluation on the companion webpage. While the audio quality of these results can still be improved, we can already confirm the ability of the model to provide semantic controls. As the learned style variables of WAE-Fader can be mixed continuously, we also provide some sound examples that were generated when modifying multiple orchestral attributes.
Our proposal provides intuitive sound synthesis of target sound qualities with learned style variables that can be modulated and combined. The unsupervised latent dimensions organize remaining data features, which can be directly visualized in a 3-d. space, in order to perform sampling or explicit control. These features allow to generate timbres, playing styles and hybrid effects across multiple attribute combinations through intuitive interactions. We provide a real-time implementation of our models by relying on the fine-tuned $\{G \circ \text{MCNN}\}$ generation. This implementation relies on the *LibTorch* C++ API, which converts trained *PyTorch* models, that we further embed in a *PureData* external. This plugin can be mapped to a MIDI controller or integrated in a DAW for composition and musical performance. This allows to play notes with a keyboard, while using continuous faders to control latent coordinates and mix style conditions.

6. CONCLUSION

We developed an expressive musical conditioning of the Wasserstein Auto-Encoders able to model a collection of orchestral note samples. The model learns effective target semitone and octave

model	test rec.		note cond. acc.				style cond. acc.		inter-class MMD			post-class. acc.		
	MSE	LSD	$st._{34}$	$oct._{34}$	$st._{08}$	$oct._{08}$	$style_{34}$	$style_{08}$	st.	oct.	style	st.	oct.	style
Clarinet	0.87	116	0.96	0.99	0.98	0.45	0.97	0.92	0.05	0.58	0.12	0.15	0.76	0.41
Piano	0.99	113	0.53	0.91	0.47	0.72	0.72	0.64	0.03	0.03	0.08	0.16	0.20	0.43
Trumpet	0.90	107	0.91	0.93	0.96	0.37	0.90	0.87	0.60	0.11	0.02	0.42	0.50	0.29
Alto-Sax.	1.22	131	0.96	0.99	0.98	0.40	0.76	0.71	0.14	0.09	0.50	0.08	0.48	0.48
T. Trombone	0.96	100	1.00	1.00	0.92	0.41	0.83	0.77	0.04	0.14	0.34	0.06	0.55	0.47

Table 4: Additional WAE-Fader results on the playing techniques of instruments in other orchestral families

controls as well as continuous style variables. We considered extended playing techniques and timbre subsets as attributes, and used adversarial latent training to encourage an attribute-invariant representation in the WAE-Fader. Our ablation study validates the effectiveness of style conditioning when this invariance condition is obtained.

We fine-tuned the decoders with a Mel magnitude spectrogram inversion network that allows real-time waveform rendering and are currently working on refining the audio quality. This results in a note sample generator with meaningful data visualizations and intuitive controls of audio styles. These parameters can be mixed, as faders, in order to explore hybrid sound effects. Our final generative model is embed in a plugin for MIDI mapping and live interactions. This system provides assisted music production and fosters creative sound experimentations. We provide sound examples from our orchestral models, either inverted offline with GLA or with the fine-tuned waveform generation pipeline. These sounds allow for subjective evaluation of both semantic and audio qualities of our solution.

Although we used clearly defined metadata attributes pertaining to instrumental playing styles, the model can potentially be applied to any sound domain. For instance, a user library with custom tags could be mapped to sound synthesis parameters. Furthermore, as the architecture is rather light and scales to small datasets, it could be trained on user libraries. Future experiments will target the quality of the waveform modelling systems for variable note lengths and real-time synthesis. Ultimately, our models could be implemented as a standalone instrument with physical controls that can be mapped to pretrained style variables. This would allow an intuitive and creative exploration across a vast amount of sound variations with a reduced set of adaptive parameters.

7. ACKNOWLEDGEMENTS

This work was supported by the MAKIMOno project 17-CE38-0015-01 funded by the French ANR and the Canadian NSERC (STPG 507004-17), the ACTOR Partnership funded by the Canadian SSHRC (895-2018-1023) and an NVIDIA GPU Grant. We acknowledge and thank the initial contributions of Jean-Baptiste Dakeyo and Geoffroy Thibault, as well as the further collaboration of Antoine Caillon and Martin Fouilleul.

8. REFERENCES

[1] J. Engel et al., "Neural audio synthesis of musical notes with wavenet autoencoders," *International Conference on Machine Learning (ICML)*, 2017.

[2] I. Tolstikhin et al., "Wasserstein auto-encoders," *International Conference on Learning Representations*, 2018.

[3] S. Zhao et al., "Infovae: Information maximizing variational autoencoders," *arXiv:1706.02262*, 2017.

[4] X. Huang et al., "Arbitrary style transfer in real-time with adaptive instance normalization," *International Conference on Computer Vision (ICCV)*, 2017.

[5] E. Perez et al., "Film: Visual reasoning with a general conditioning layer," *AAAI Conference*, 2018.

[6] G. Lample et al., "Fader networks: Manipulating images by sliding attributes," *Conference on Neural Information Processing Systems (NIPS)*, 2017.

[7] D. W. Griffin and J. S. Lim, "Signal estimation from modified short-time fourier transform," *IEEE Transactions on acoustics, speech, and signal processing*, 1984.

[8] G. Lample et al., "Fast spectrogram inversion using multi-head convolutional neural networks," *IEEE Signal Processing Letters, vol. 26, no. 1*, 2019.

[9] D. P. Kingma et al., "Auto-encoding variational bayes," *International Conference on Learning Representations*, 2014.

[10] S. Zhao et al., "Towards deeper understanding of variational autoencoding model," *arXiv:1702.08658*, 2017.

[11] S. Yeung et al., "Tackling over-pruning in variational autoencoders," *ICML workshop*, 2017.

[12] A. Gretton et al., "A kernel two-sample test," *Journal of Machine Learning Research*, vol. 13, 2012.

[13] I. Higgins et al., "beta-vae: Learning basic visual concepts with a constrained variational framework," *International Conference on Learning Representations (ICLR)*, 2017.

[14] G. Ghiasi et al., "Exploring the structure of a real-time, arbitrary neural artistic stylization network," *British Machine Vision Conference (BMVC)*, 2017.

[15] A. Van Den Oord et al., "Wavenet: A generative model for raw audio," *arXiv:1609.03499*, 2016.

[16] G. Ballet et al., "Studio online 3.0: An internet "killer application" for remote access to ircam sounds and processing tools," *Journee Informatique Musicale (JIM)*, 1999.

[17] A. Odena et al., "Deconvolution and checkerboard artifacts," *Distill*, 2016.

UNIVERSAL AUDIO SYNTHESIZER CONTROL WITH NORMALIZING FLOWS

Philippe Esling[1], Naotake Masuda[1], Adrien Bardet[1], Romeo Despres[1], Axel Chemla–Romeu-Santos[1,2]

[1] IRCAM - CNRS UMR 9912
Sorbonne Université, Paris, France
esling@ircam.fr

[2] Laboratorio d'Informatica Musicale (LIM)
UNIMI, Milano, Italy
axel.chemla@unimi.it

ABSTRACT

The ubiquity of sound synthesizers has reshaped music production and even entirely defined new music genres. However, the increasing complexity and number of parameters in modern synthesizers make them harder to master. Hence, the development of methods allowing to easily create and explore with synthesizers is a crucial need.

Here, we introduce a novel formulation of audio synthesizer control. We formalize it as finding an organized latent audio space that represents the capabilities of a synthesizer, while constructing an invertible mapping to the space of its parameters. By using this formulation, we show that we can address simultaneously *automatic parameter inference*, *macro-control learning* and *audio-based preset exploration* within a single model. To solve this new formulation, we rely on Variational Auto-Encoders (VAE) and Normalizing Flows (NF) to organize and map the respective *auditory* and *parameter* spaces. We introduce a new type of NF named *regression flows* that allow to perform an invertible mapping between separate latent spaces, while steering the organization of some of the latent dimensions. We evaluate our proposal against a large set of baseline models and show its superiority in both parameter inference and audio reconstruction. We also show that the model disentangles the major factors of audio variations as latent dimensions, that can be directly used as *macro-parameters*. Finally, we discuss the use of our model in creative applications and its real-time implementation in Ableton Live[1].

1. INTRODUCTION

Synthesizers are parametric systems able to generate audio signals ranging from musical instruments to entirely unheard-of sound textures. Since their commercial beginnings more than 50 years ago, synthesizers have revolutionized music production, while becoming increasingly accessible, even to neophytes with no background in signal processing.

[1] All code, supplementary figures, results and plugins are available on a supporting webpage: https://acids-ircam.github.io/flow_synthesizer/

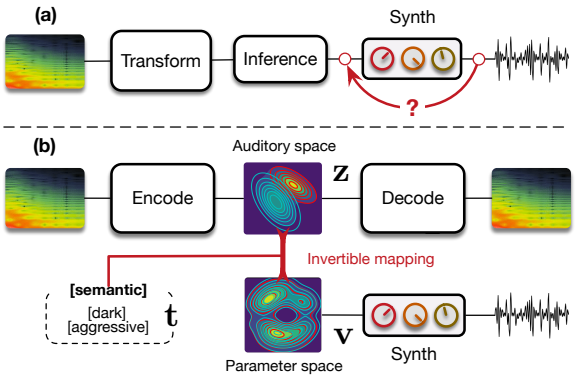


Figure 1: *Universal synthesizer control*. (a) Previous methods perform direct inference from audio, which is limited by non-differentiable synthesis and lacks high-level control. (b) Our novel formulation states allows to learn an organized latent space $\mathbf{z}$ of the synthesizer's audio capabilities, while mapping it to the space $\mathbf{v}$ of its synthesis parameters.

While there exists a variety of sound synthesis types [1], they all require an *a priori* knowledge to make the most out of a synthesizer possibilities. Hence, the main appeal of these systems (namely their versatility provided by large sets of parameters) also entails their major drawback. Indeed, the sheer combinatorics of parameter settings makes exploring all possibilities to find an adequate sound a daunting and time-consuming task. Furthermore, there are highly non-linear relationships between the parameters and the resulting audio. Unfortunately, no synthesizer provides intuitive controls related to perceptual and semantic properties of the synthesis. Hence, a method allowing an intuitive and creative exploration of sound synthesizers has become a crucial need, especially for non-expert users.

A potential direction taken by synth manufacturers, is to propose *macro-controls* that allow to quickly tune a sound by controlling multiple parameters through a single knob. However, these need to be programmed manually, which still requires expert knowledge. Furthermore, no method has ever tried to tackle this *macro-control learning* task, as this objective appears unclear and depends on a variety of unknown factors. An alternative to manual parameter setting would be to infer the set of parameters that could best

reproduce a given *target sound*. This task of *parameter inference* has been studied in the past years using various techniques. In Cartwright et al. [2], parameters are iteratively refined based on audio descriptors similarity and relevance feedback provided by the user. However, this approach appears to be rather inaccurate and slow. Garcia et al. [3] proposed to use genetic programming to directly grow modular synthesizers to solve this problem. Although the approach is appealing and appears accurate, the optimization of a single target can take from 10 to 200 hours, which makes it unusable. Recently, Yee-king et al. [4] showed that a bi-directional LSTM with highway layers can produce accurate parameters appproximations. However, this approach does not allow for any user interaction. All of these approaches share the same flaws that (i) though it is unlikely that a synthesizer can generate exactly any audio target, none explicitly model these limitations, (ii) they do not account for the non-linear relationships that exist between parameters and the corresponding synthesized audio. Hence, no approach has succeeded in unveiling the true relationships between these *auditory* and *parameters* spaces. Here, we argue that it is mandatory to organize the parameters and audio capabilities of a given synthesizer in their respective spaces, while constructing an invertible mapping between these spaces in order to access a range of high-level interactions. This idea is depicted in Figure 1

The recent rise of *generative models* might provide an elegant solution to these questions. Indeed, amongst these models, the *Variational Auto-Encoder* (VAE) [5] aims to uncover the underlying structure of the data, by explicitly learning a *latent space* [5]. This space can be seen as a high-level representation, which aims to disentangle underlying variation factors and reveal interesting structural properties of the data [5, 6]. VAEs address the limitations of control and analysis through this latent space, while being able to learn on small sets of examples. Furthermore, the recently proposed *Normalizing Flows* (NF) [7] allow to model highly complex distributions in the latent space. Although the use of VAEs for audio applications has only been scarcely investigated, Esling et al. [8] recently proposed a perceptually-regularized VAE that learns a space of audio signals aligned with perceptual ratings via a regularization loss. The resulting space exhibits an organization that is well aligned with perception. Hence, this model appears as a valid candidate to learn an organized audio space.

In this paper, we introduce a radically novel formulation of audio synthesizer control by formalizing it as the general question of finding an invertible mapping between two learned latent spaces. In our case, we aim to map the audio space of a synthesizer's capabilities to the space of its parameters. We provide a generic probabilistic formalization and show that it allows to address simultaneously the tasks of *parameter inference*, *macro-control learning*, *audio-based preset exploration* and *semantic dimension discovery* within a single model. To elegantly solve this formulation, we introduce *conditional regression flows*, which map a latent space to any given target space, while steering the organization of some dimensions to match target distributions. Our complete model is depicted in Figure 2.

Based on this formulation, *parameter inference* simply consists of encoding the audio target to the latent audio space that is mapped to the parameter space. Interestingly, this bypasses the well-known blurriness issue in VAEs as we can generate directly with the synthesizer. We evaluate our proposal against a large set of baseline models and show its superiority in parameter inference and audio reconstruction. Furthermore, we show that our model is the first able to address the new task of automatic *macro-control learning*. As the latent dimensions are continuous and map to the parameter space, they provide a natural way to learn the perceptually most significant macro-parameters. We show that these controls map to smooth, yet non-linear parameters evolution, while remaining perceptually continuous. Furthermore, as our mapping is invertible, we can map synthesis parameters back to the audio space. This allows intuitive *audio-based preset exploration*, where exploring the neighborhood of a preset encoded in the audio space yields similarly sounding patches, yet with largely different parameters. Finally, we discuss creative applications of our model and real-time implementation in *Ableton Live*.

2. STATE-OF-ART

2.1. Generative models and variational auto-encoders

Generative models aim to understand a given set $\mathbf{x} \in \mathbb{R}^{d_x}$ by modeling the underlying probability distribution of the data $p(\mathbf{x})$. To do so, we consider *latent variables* defined in a lower-dimensional space $\mathbf{z} \in \mathbb{R}^{d_z}$ ($d_z \ll d_x$), a higher-level representation that could have led to generate a given example. The complete model is defined by the joint distribution $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$. Unfortunately, real-world data follow complex distributions, which cannot be found analytically. The idea of *variational inference* (VI) is to solve this problem through *optimization* by assuming a simpler approximate distribution $q_\phi(\mathbf{z}|\mathbf{x}) \in \mathcal{Q}$ from a family of approximate densities [9]. The goal of VI is to minimize the difference between this approximation and the real distribution, by minimizing the Kullback-Leibler (KL) divergence between these densities

$$q_\phi^*(\mathbf{z}|\mathbf{x}) = \text{argmin}_{q_\phi(\mathbf{z}|\mathbf{x}) \in \mathcal{Q}} \mathcal{D}_{KL}\big[q_\phi\left(\mathbf{z}|\mathbf{x}\right) \parallel p\left(\mathbf{z}|\mathbf{x}\right)\big]$$

By developing this KL divergence and re-arranging terms (the detailed development can be found in [5]), we obtain

$$\begin{aligned}\log p(\mathbf{x}) - D_{KL}\big[q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z}|\mathbf{x})\big] \\ = \mathbb{E}_{\mathbf{z}}\big[\log p(\mathbf{x}|\mathbf{z})\big] - D_{KL}\big[q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z})\big] \quad (1)\end{aligned}$$

This formulation describes the quantity we want to model $\log p(\mathbf{x})$ minus the error we make by using an approximate

q instead of the true p. Therefore, we can optimize this alternative objective, called the *evidence lower bound* (ELBO)

$$\mathcal{L}_{\theta,\phi} = \mathbb{E}\big[\log p_\theta(\mathbf{x}|\mathbf{z})\big] - \beta \cdot D_{KL}\big[q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{z})\big] \quad (2)$$

The ELBO intuitively minimizes the reconstruction error through the likelihood of the data given a latent $\log p_\theta(\mathbf{x}|\mathbf{z})$, while regularizing the distribution $q_\phi(\mathbf{z}|\mathbf{x})$ to follow a given prior distribution $p_\theta(\mathbf{z})$. We can see that this equation involves $q_\phi(\mathbf{z}|\mathbf{x})$ which *encodes* the data $\mathbf{x}$ into the latent representation $\mathbf{z}$ and a *decoder* $p_\theta(\mathbf{x}|\mathbf{z})$, which generates $\mathbf{x}$ given a $\mathbf{z}$. This structure defines the *Variational Auto-Encoder* (VAE), where we can use parametric neural networks to model the *encoding* (q_ϕ) and *decoding* (p_θ) distributions. VAEs are powerful representation learning frameworks, while remaining simple and fast to learn without requiring large sets of examples [10].

However, the original formulation of the VAE entails several limitations. First, it has been shown that the KL divergence regularization can lead both to uninformative latent codes (also called *posterior collapse*) and variance overestimation [11]. One way to alleviate this problem is to rely on the *Maximum Mean Discrepancy* (MMD) instead of the KL to regularize the latent space, leading to the WassersteinAE (WAE) model [12]. Second, one of the key aspect of VI lies in the choice of the family of approximations. The simplest choice is the *mean-field* family where latent variables are mutually independent and parametrized by distinct variational parameters $q(z) = \prod_{j=1}^{m} q_j(z_j)$. Although this provide an easy tool for analytical development, it might prove too simplistic when modeling complex data as this assumes pairwise independence among every latent axis. Normalizing flows alleviate this issue by adding a sequence of invertible transformations to the latent variable, providing a more expressive inference process.

2.2. Normalizing flows

In order to transform a probability distribution, we can rely on the *change of variable* theorem. As we deal with probability distributions, we need to *scale* the transformed density so that it still sums to one, which is measured by the determinant of the transform. Formally, let $\mathbf{z} \in \mathcal{R}^d$ be a random variable with distribution $q(\mathbf{z})$ and $f : \mathcal{R}^d \to \mathcal{R}^d$ an invertible smooth mapping. We can use f to transform $\mathbf{z} \sim q(\mathbf{z})$, so that the resulting random variable $\mathbf{z}' = f(\mathbf{z})$ has the following probability distribution

$$q(\mathbf{z}') = q(\mathbf{z}) \left| \det \frac{\partial f^{-1}}{\partial \mathbf{z}'} \right| = q(\mathbf{z}) \left| \det \frac{\partial f}{\partial \mathbf{z}} \right|^{-1} \quad (3)$$

where the last equality is obtained through the inverse function theorem. We can perform any number of transforms to obtain a final distribution $\mathbf{z}_k \sim q_k(\mathbf{z}_k)$ given by

$$\begin{aligned} q_k(\mathbf{z}_k) &= q_0(f_1^{-1} \circ ... \circ f_k^{-1}(\mathbf{z}_k)) \prod_{i=1}^{k} \left| \det \frac{\partial f_i^{-1}}{\partial \mathbf{z}_i} \right| \\ &= q_0(\mathbf{z_0}) \prod_{i=1}^{k} \left| \det \frac{\partial f_i}{\partial \mathbf{z}_{i-1}} \right|^{-1} \end{aligned} \quad (4)$$

This series of transformations, called a *normalizing flow* [7], can turn a simple distribution into a complicated multimodal density. For practical use of these flows, we need transforms whose Jacobian determinants are easy to compute. Interestingly, *Auto-Regressive* (AR) transforms fit this requirement as they lead to a triangular Jacobian matrix. Hence, different AR flows were proposed such as *Inverse AR Flows* (IAF) [13] and *Masked AR Flows* (MAF) [14]

Normalizing flows in VAEs. Normalizing flows allow to address the simplicity of variational approximations by complexifying their posterior distribution [7]. In the case of VAEs, we parameterize the approximate posterior distribution with a flow of length K, $q_\phi(\mathbf{z}|\mathbf{x}) = q_K(\mathbf{z}_K)$, and the new optimization loss can be simply written as an expectation over the initial distribution $q_0(\mathbf{z})$

$$\begin{aligned} \mathcal{L} &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log q_\phi(\mathbf{z}|\mathbf{x}) - \log p(\mathbf{x}, \mathbf{z})\right] \\ &= \mathbb{E}_{q_0(\mathbf{z}_0)}\left[\ln q_0(\mathbf{z}_0)\right] - \mathbb{E}_{q_0(\mathbf{z}_0)}\left[\log p(\mathbf{x}, \mathbf{z}_K)\right] \\ &- \mathbb{E}_{q_0(\mathbf{z}_0)}\left[\sum_{i=1}^{k} \log \left| \det \frac{\partial f_i}{\partial \mathbf{z}_{i-1}} \right|\right] \end{aligned} \quad (5)$$

The resulting objective can be easily optimized since q_0 is still a Gaussian from which we can easily sample. However, the final samples $\mathbf{z}_k$ used by the decoder are drawn from a more complex distribution.

3. OUR PROPOSAL

3.1. Formalizing synthesizer control

Considering a set of audio samples $\mathcal{D} = \{\mathbf{x}_i\}, i \in [1, n]$ where the $\mathbf{x}_i \in \mathbb{R}^d$ follow an unknown distribution $p(\mathbf{x})$, we can define latent factors $\mathbf{z} \in \mathbb{R}^z$ to model the joint distribution $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x} \mid \mathbf{z})p(\mathbf{x})$ as detailed in Section 2.1. In our case, some $\bar{\mathbf{x}} \in \mathcal{D}_s \subset \mathcal{D}$ inside this set have been generated by a given synthesizer. This synthesizer defines a generative function $f_s(\mathbf{v}; p, i) = \bar{\mathbf{x}}$ where $\mathbf{v} \in \mathbb{R}^s$ is a set of parameters that produce $\bar{\mathbf{x}}$ at a given pitch p and intensity i. However, in the general case, we know that if $\mathbf{x}_j \notin \mathcal{D}_s$, then $\mathbf{x}_j = f_s(\mathbf{v}) + \epsilon$ where ϵ models the error made when trying to reproduce any audio $\mathbf{x}_i$ with a given synthesizer. Finally, we consider that some audio examples are annotated with a set of *categorical semantic tags* $\mathbf{t}_i = \{0, 1\}^t$, which define high-level perceptual properties that separate *unknown* latent factors $\mathbf{z}$ and *target* factors $\mathbf{t}$. Hence, the complete generative story of a synthesizer can be defined as

$$p(\mathbf{x}, \mathbf{v}, \mathbf{t}, \mathbf{z}) = p(\mathbf{x}|\mathbf{v}, \mathbf{t}, \mathbf{z})p(\mathbf{v}|\mathbf{t}, \mathbf{z})p(\mathbf{t}|\mathbf{z})p(\mathbf{z}) \quad (6)$$

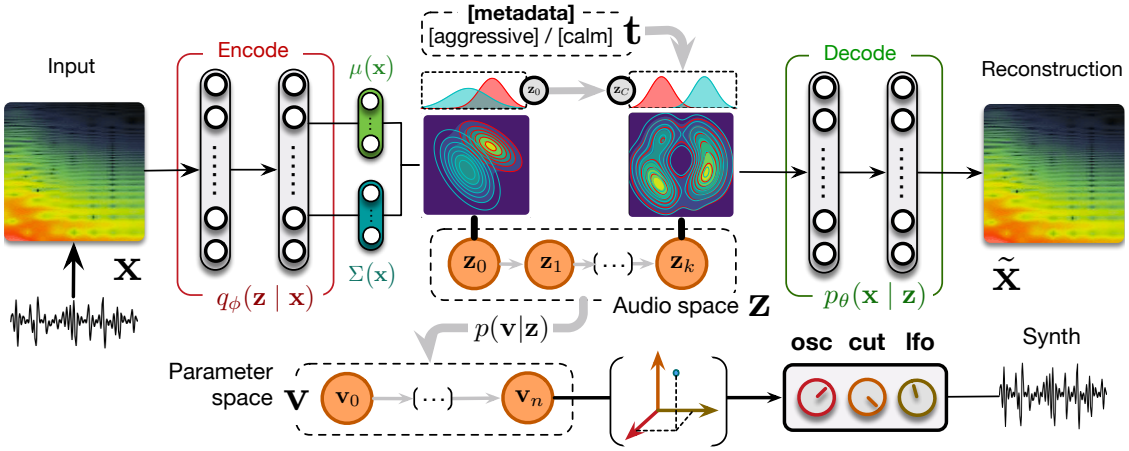


Figure 2: *Universal synthesizer control*. We learn an organized latent audio space $\mathbf{z}$ of a synthesizer capabilities with a VAE parameterized with NF. This space maps to the parameter space $\mathbf{v}$ through our proposed *regression flow* and can be further organized with metadata targets $\mathbf{t}$. This provides sampling and invertible mapping between different spaces.

This very general formulation entails our original idea that we should uncover the relationship between the latent audio $\mathbf{z}$ and parameters $\mathbf{v}$ spaces by modeling $p(\mathbf{v}, \mathbf{z})$. The advantage of this formulation is that the reduced dimensionality $\mathbb{R}^z \ll \mathbb{R}^x$ of the latent $\mathbf{z}$ simplifies the problem of parameters inference, by relying on a more adequate and smaller input space. Furthermore, this formulation also provides a natural way of learning *macro-controls* by inferring $p(\mathbf{v}|\mathbf{z})$ in the general case, where separate dimensions of $\mathbf{z}$ are expected to produce smooth auditory transforms. Although we provide a complete formalization, due to space constraints, we do not detail the use of metadata $\mathbf{t}$ in our model. We provide this information in the supporting webpage and companion article. Here, we consider that tags $\mathbf{t}$ are included in the latent factors $\mathbf{z}$ and define the model as

$$p_\theta(\mathbf{x}, \mathbf{v}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{v}, \mathbf{z})p_\theta(\mathbf{v}|\mathbf{z})p_\theta(\mathbf{z}) \tag{7}$$

3.2. Mapping latent spaces with regression flows

In order to map the latent $\mathbf{z}$ and parameter $\mathbf{v}$ spaces, we first separate our formulation so that

$$\log p_\theta(\mathbf{x}, \mathbf{v}, \mathbf{z}) = \log(p_\theta(\mathbf{x}|\mathbf{v}, \mathbf{z})p_\theta(\mathbf{z})) + \log p_\theta(\mathbf{v}|\mathbf{z}) \tag{8}$$

This allows to separately model the variational approximation detailed in Section 2.1, while solving separately the inference problem $p_\theta(\mathbf{v}|\mathbf{z})$. To address this inference, we need to find the optimal parameters $\boldsymbol{\psi}$ of a transform $f_{\boldsymbol{\psi}}$ so that $\mathbf{v} = f_{\boldsymbol{\psi}}(\mathbf{z}) + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{C}_v)$ models the inference error as a zero-mean additive Gaussian noise with covariance $\mathbf{C}_v$. Here, we assume that the covariance decomposes into $\mathbf{C}_v^{-1} = \sum_i exp(\lambda_i)\mathbf{Q}_i$, where $\mathbf{Q}_i$ are fixed basis functions and λ are hyperparameters. Therefore, the full joint likelihood that we need to optimize is given by

$$\mathcal{L}_{f_{\boldsymbol{\psi}},\lambda} = \log\left[p_\theta(\mathbf{v}|f_{\boldsymbol{\psi}}, \lambda, \mathbf{z})p_\theta(f_{\boldsymbol{\psi}}|\mathbf{z})p_\theta(\lambda|\mathbf{z})\right] \tag{9}$$

If we know the optimal transform $f_{\boldsymbol{\psi}}$ and parameters λ, the likelihood of the data can be easily computed as

$$p_\theta(\mathbf{v} \mid f_{\boldsymbol{\psi}}, \lambda, \mathbf{z}) = \mathcal{N}(\mathbf{v}; f_{\boldsymbol{\psi}}(\mathbf{z}), \mathbf{C}_v) \tag{10}$$

However, the two posteriors $p_\theta(f_{\boldsymbol{\psi}}|\mathbf{z})$ and $p_\theta(\lambda|\mathbf{z})$ remain intractable in the general case. In order to solve this issue, we rely again on variational inference by defining an approximation $q_\phi(f_{\boldsymbol{\psi}}, \lambda|\mathbf{v}, \mathbf{z})$ (see Section 2.1) and assume that it factorizes as $q(f_{\boldsymbol{\psi}}, \lambda|\mathbf{v}, \mathbf{z}) = q(f_{\boldsymbol{\psi}}|\mathbf{v}, \mathbf{z})q(\lambda|\mathbf{v}, \mathbf{z})$. Therefore, our final inference problem is

$$\begin{aligned}\mathcal{L}_{f_{\boldsymbol{\psi}},\lambda} &= \log\left[p_\theta(\mathbf{v}|f_{\boldsymbol{\psi}}, \lambda, \mathbf{z})\right] \\ &+ \mathcal{D}_{\mathrm{KL}}\left[q_\phi(f_{\boldsymbol{\psi}}|\mathbf{z}, \mathbf{v})\|p_\theta(f_{\boldsymbol{\psi}}|\mathbf{z})\right] \qquad (11)\\ &+ \mathcal{D}_{\mathrm{KL}}\left[q_\phi(\lambda|\mathbf{z}, \mathbf{v})\|p_\theta(\lambda|\mathbf{z})\right] \qquad (12)\end{aligned}$$

Hence, we can optimize our approximations through the KL divergence if we find a closed form. To solve for λ, we use a Gaussian distribution for both the prior $p_\theta(\lambda|\mathbf{z}) = \mathcal{N}(\lambda, \mu_\lambda, C_\lambda)$ and posterior $q_\phi(\lambda|\mathbf{z}, \mathbf{v}) = \mathcal{N}(\lambda, \mu_q, C_q)$. To solve this issue, we introduce the idea of *regression flows*. This allows to obtain a simple analytical solution. However, the second part of the objective might be more tedious. Indeed, to perform an accurate inference, we need to rely on a complicated non-linear function, which cannot be assumed to be Gaussian. To address this issue, we introduce the idea of *regression flows*. We consider that the transform $f_\theta(\mathbf{z})$ is a normalizing flow (see Section 2.1) and provides two different way of optimizing the approximation.

Posterior parameterization. First, we follow a reasoning akin to the original formulation of normalizing flows by parameterizing the posterior $q_\phi(f_{\boldsymbol{\psi}}|\mathbf{z}, \mathbf{v})$ with a flow $q_k(\mathbf{v}_k)$.

Hence, by developing the KL expression, we obtain

$$\mathcal{D}_{\mathrm{KL}}\left[q_\phi(f_{\boldsymbol{\psi}}|\mathbf{z},\mathbf{v})\|p(f_{\boldsymbol{\psi}}|\mathbf{z})\right] = \mathbb{E}_{q_0}\left[\log q_0(\mathbf{v}_0)\right] - \mathbb{E}_{q_0}\left[\log p(\mathbf{v}_k)\right] - \mathbb{E}_{q_0}\left[\sum_{i=1}^{k}\log\left|\det\frac{\partial f_i}{\partial \mathbf{v}_{i-1}}\right|\right] \quad (13)$$

Hence, we can now safely rely on Gaussian priors for $q_0(\mathbf{v}_0)$ and $p(\mathbf{v}_k)$. This formulation allows to consider $\mathbf{v}$ as a transformed version of $\mathbf{z}$, while being easily invertible as $\mathbf{z} = f^{-1}_{[k,1]}(\mathbf{v})$. We denote this version as $Flow_{post}$.

Conditional amortization. Here, we consider that the parameters $\boldsymbol{\psi}$ of the flow are random variables that are optimized by decomposing the posterior KL objective as

$$\mathcal{D}_{\mathrm{KL}}\left[q_\phi(f_{\boldsymbol{\psi}}|\mathbf{z},\mathbf{v})\|p(f_{\boldsymbol{\psi}}|\mathbf{z})\right] = \mathcal{D}_{\mathrm{KL}}\left[q_\phi(\boldsymbol{\psi}|\mathbf{z})\|p(\boldsymbol{\psi}|\mathbf{z})\right] + \mathbb{E}_{q_0(\mathbf{v}_0)}\left[\sum_{i=1}^{k}\log\left|\det\frac{\partial f_i}{\partial \mathbf{v}_{i-1}}\right|\right] \quad (14)$$

As we rely on Gaussian priors for the parameters, this additional KL term can be computed easily. In this version, denoted $Flow_{cond}$, parameters of the flow are sampled from their distributions before computing the resulting transform.

4. EXPERIMENTS

4.1. Dataset

Synthesizer. We constructed a dataset of synthesizer sounds and parameters, by using an off-the-shelf commercial synthesizer *Diva* developed by U-He[2]. It should be noted that our model can work for any synthesizer, as long as we obtain couples of (audio, parameters) as input. We selected Diva as (i) almost all its parameters can be MIDI-controlled, (ii) large banks of presets are available and (iii) presets include well-organized semantic tags pairs. The factory presets for Diva and additional presets from the internet were collected, leading to a total of roughly 11k files. We manually established the correspondence between synth and MIDI parameters as well as the parameters values range and distributions. We only kept continuous parameters and normalize all their values to $[0, 1]$. All other parameters are set to their fixed *default* value. Finally, we performed PCA and manual screening to select increasing sets of the most used 16, 32 and 64 parameters. We use *RenderMan*[3] to batch-generate all the audio files by playing the note for 3 sec. and recording for 4 sec. to capture the release of the note. The files are saved in 22050Hz and 16bit floating point format.

Audio processing. For each sample, we compute a 128 bins Mel-spectrogram with a FFT of size 2048 with a hop of 1024 and frequency range of $[30, 11000]$. We only keep the magnitude of the spectrogram and perform a log-amplitude transform. The dataset is randomly split between a training (80%), validation (10%) and test (10%) set before each training. We repeat the training k times to perform k-fold cross-validation. Finally, we perform a corpus-wide zero-mean unit-variance normalization based on the train set.

[2]https://u-he.com/products/diva/

[3]https://github.com/fedden/RenderMan

4.2. Models

Baseline models. In order to perform an objective evaluation of our proposal, we implemented several recent high-capacity models similar to [4]. We implement a 5-layers MLP with 2048 hidden units per layer, Exponential Linear Unit (ELU) activation, batch normalization and dropout with $p = .3$. The final layer is a sigmoid activation. We implement a gated variant of this model, denoted MLP_g. We implement a convolutional model composed of 5 layers with 128 channels of strided dilated convolutions with kernel size 7, stride 2 and an exponential dilation factor of 2^l with batch normalization and ELU activation. The convolutions are followed by a 3-layers MLP identical to the previous model. We also implement the gated variant denoted CNN_g. Finally, we implemented a variant of *Residual Networks*, with parameters settings identical to CNN and denote this model $ResCNN$.

Our models. We implemented various *AE architectures to evaluate different aspects of our proposal. To perform a fair comparison, we rely on the same setup as before, but halve the number of parameters to obtain roughly the same capacity as the baselines. First, we implement a simple deterministic AE without regularization. We implement the VAE by adding a KL regularization to the latent space and the WAE by replacing the KL by the MMD. Finally, we implement a VAE_{flow} by adding a normalizing flow composed of 16 successive IAF transforms to the VAE latent posterior. All AEs map to a latent space of dimensionality equal to the number of synthesis parameters. For probabilistic models, we perform *warmup* [10] by linearly increasing the regularization β from 0 to 1 for 100 epochs. We also apply the same weight annealing for the regression loss. We first evaluate all these models by using a simple 2-layers MLP to predict the parameters based on the latent space. Finally, we evaluate our *regression flows* by adding them to the VAE_{flow}, with an IAF of length 16.

Optimization. We train all models for 500 epochs with the ADAM optimizer, initial learning rate of 0.0002, Xavier initialization and a scheduler that halves the learning rate if the validation loss stalls for 20 epochs. With this setup, the most complex VAE_{flow} with regression flows only needs 5 hours to complete training on a NVIDIA Titan Xp GPU.

5. RESULTS

5.1. Parameters inference

We compare the accuracy of our proposal with all baseline models on the *parameters inference* task. To do so, we evaluate the distance between predicted parameters and their real values in the test dataset, by computing the magnitude-

	Parameter		Audio			
	MSE_n		SC		MSE	
MLP	0.236	± .44	6.226	± .13	9.445	± 3.1
MLP_g	0.195	± .45	1.731	± .31	7.787	±1.9
CNN	0.171	± .43	1.372	± .29	6.329	±1.9
CNN_g	0.174	± .44	1.245	± .28	6.496	±2.0
$ResCNN$	0.191	± .43	1.004	± .35	6.422	±1.9
AE	0.181	±.40	0.893	±.13	5.557	±1.7
VAE	0.182	±.32	0.810	±.03	4.901	±1.4
WAE	**0.159**	**±.37**	0.787	±.05	4.979	±1.5
VAE_{fl}	0.199	±.32	0.838	±.02	4.975	±1.4
$Flow_{post}$	0.197	±.31	**0.752**	**±.05**	**4.409**	**±1.6**
$Flow_{cond}$	0.199	±.31	1.085	±.02	6.303	±2.1

Table 1: Comparison between baseline, *AEs with MLP regression and our proposed *regression flows* on the test set. Parameters accuracy is evaluated with normalized MSE and the audio with Spectral Convergence (SC) and MSE.

normalized *Mean Square Error* (MSE_n). We average these results across k-folds and report across-runs variance. More importantly, we evaluate the distance between the audio synthesized based on these inferred parameters and the original audio through the *Spectral Convergence* (SC) and *MSE* distances, where SC is the Frobenius norm normalized over time and frequency. The results are displayed in Table 1.

As we can see, baseline models are able to perform an accurate approximation of the parameter vectors, with the CNN providing the best inference. Based on this parameter distance criterion solely, the best results are obtained by the deterministic WAE model that outperforms traditional approaches. Although it would seem, at first, that our formulation only provides a marginal improvement on the parameters inference task, and that our proposal is even outperformed by baseline models, the analysis of the corresponding synthesized audio tells an entirely different story. Indeed, all AEs approaches strongly outperform the baseline models when it comes to audio accuracy, with the best results obtained with our probabilistic formulation $Flow_{post}$. These results show that, even though AE models do not provide an exact approximation of parameter vectors, they are able to account for the importance of these different parameters on the corresponding audio result. An even more interesting observation is that our proposed $Flow_{post}$ is outperformed by most baseline models on the parameters accuracy distance. However, it strongly outperforms all other methods on the resulting audio approximation accuracy. This supports our original hypothesis that learning the latent space of the synthesizer audio capabilities is a crucial component to understand its behavior. Furthermore, this might imply that our model can provide closely-sounding parameter settings based on the audio latent space, even though the parameters are quite different. This assumption is evaluated in the following section. Finally, this analysis also seems to be supported by the across-run variance, where probabilistic models obtain a more consistent accuracy, indicating that they provide a better generalization.

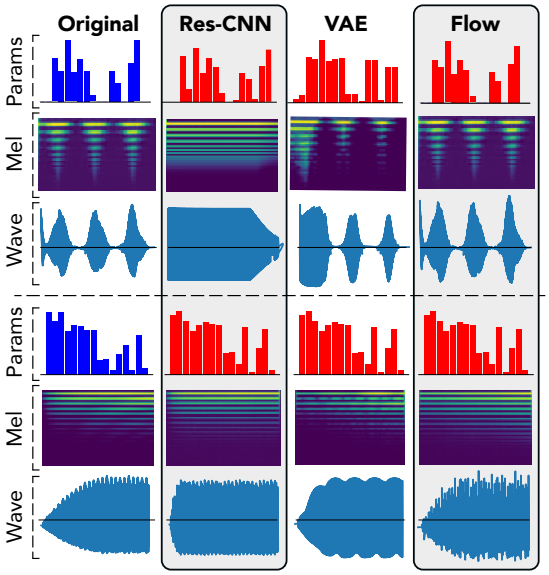


Figure 3: *Reconstruction analysis*. Comparing parameters inference and corresponding synthesized audio on the test dataset between the best performing models.

5.2. Reconstructions and latent space

We provide an in-depth analysis of the relations between inferred parameters and corresponding synthesized audio to support our previous claims. First, we selected two samples from the test set and compare the inferred parameters and synthesized audio in Figure 3.

As we can see, although the CNN provides a close inference of the parameters, the synthesized approximation completely misses important structural aspects, even in simpler instances as the slow ascending attack in the second example. This confirms that direct inference models are unable to assess the *relative impact* of parameters on the audio. Indeed, the errors in all parameters are considered equivalently, even though the same error magnitude on two different parameters can lead to dramatic differences in the synthesized audio. Oppositely, even though the parameters inferred by the VAE are quite far from the original preset, the corresponding audio is largely closer. This indicates that the latent space provides knowledge on the *audio-based neighborhoods* of the synthesizer. Therefore, this allows to understand the impact of different parameters in a given region of the latent audio space.

To evaluate this hypothesis, we encode two distant presets in the latent audio space and perform random sampling around these points to evaluate how local neighborhoods are organized. We also analyze the latent interpolation between those examples. The results are displayed in Figure 4.

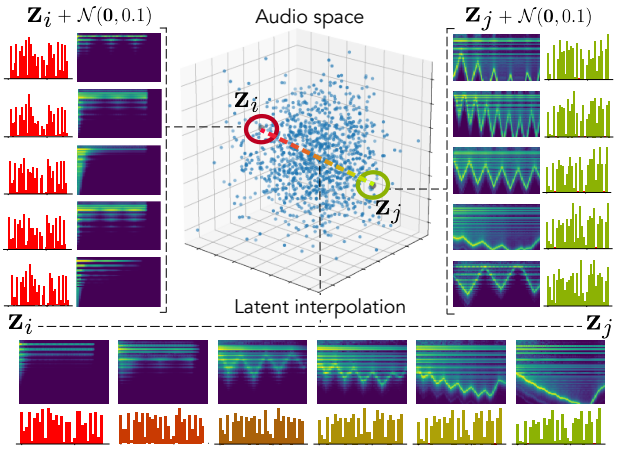


Figure 4: *Latent neighborhoods*. We select two examples from the test set that map to distant locations in the latent space $\mathbf{z}$ and perform random sampling in their local neighborhood to observe the parameters and audio. We also display the latent interpolation between those points.

As we can see, our hypothesis seems to be confirmed by the fact that neighborhoods are highly similar in terms of audio but have a larger variance in terms of parameters. Interestingly, this leads to complex but smooth non-linear dynamics in the parameters control.

5.3. Macro-parameters learning

Our formulation is the first to provide a continuous mapping between the audio $\mathbf{z}$ and parameter $\mathbf{v}$ spaces of a synthesizer. As latent VAE dimensions has been shown to disentangle major data variations, we hypothesized that we could directly use $\mathbf{z}$ as *macro-parameters* defining the most interesting variations in a given synthesizer. Hence, we introduce the new task of *macro-parameters learning* by mapping latent audio dimensions to parameters through $p(\mathbf{v}|\mathbf{z})$, which provides simplified control of the major audio variations for a given synthesizer. This is depicted in Figure 5

We show the two most informative latent dimensions $\mathbf{z}$ based on their variance. We study the traversal of these dimensions by keeping all other fixed at $\mathbf{0}$ to assess how $\mathbf{z}$ defines smooth macro-parameters through the mapping $p(\mathbf{v}|\mathbf{z})$. We report the evolution of the 5 parameters with highest variance (top), the corresponding synthesis (middle) and audio descriptors (bottom).

First, we can see that latent dimension corresponds to very smooth evolutions in terms of synthesized audio and descriptors. This is coherent with previous studies on the disentangling abilities of VAEs [6]. However, a very interesting property appear when we map to the parameter space. Although the parameters evolution is still smooth, it exhibits more non-linear relationships between different parameters. This correlates with the intuition that there are lots of complex interplays in parameters of a synthesizer. Our formulation allows to alleviate this complexity by automatically providing *macro-parameters* that are the most relevant to the audio variations of a given synthesizer. Here, we can see that the $\mathbf{z}_3$ latent dimension (left) seems to provide a *percussivity* parameter, where low values produce a very slow attack, while moving along this dimension, the attack becomes sharper and the amount of noise increases. Similarily, $\mathbf{z}_7$ seems to define an *harmonic densification* parameter, starting from a single peak frequency and increasingly adding harmonics and noise.

5.4. Creative applications

Our proposal allows to perform a direct exploration of presets based on audio similarity. Indeed, as the flow is *invertible*, we can map parameters to the audio space for exploration, and then back to parameters to obtain a new preset. Furthermore, this can be combined with *vocal sketch control* where the user inputs vocal imitations of the sound that he is looking for. This allows to quickly produce an approximation of the intended sound and then exploring the audio neighborhood of the sketch for intuitive refinement. We embedded our model inside a *MaxMSP* external called `flow_synth~` by using the *LibTorch* API and further integrate it into *Ableton Live* by using the *Max4Live* interface.

6. CONCLUSION

In this paper, we introduced several novel ideas including reformulating the problem of synthesizer control as matching the two latent space defined as the *user perception space* and the *synthesizer parameter space*. We showed that our approach outperforms all previous proposals on the seminal problem of *parameters inference*. Our formulation also naturally introduces the original tasks of *macro-control learning*, *audio-based preset exploration* and *semantic parameters discovery*. This proposal is the first to be able to simultaneously address most synthesizer control issues at once.

Altogether, we hope that this work will provide new means of exploring audio synthesis, sparkling the development of new leaps in musical creativity.

7. ACKNOWLEDGEMENTS

This work was supported by MAKIMOno project (ANR:17-CE38-0015-01 and NSERC:STPG 507004-17) and the ACTOR Partnership (SSHRC:895-2018-1023).

8. REFERENCES

[1] Miller Puckette, *The theory and technique of electronic music*, World Scientific Publishing Co., 2007.

[2] Mark Cartwright and Bryan Pardo, "Synthassist: an audio synthesizer programmed with vocal imitation,"

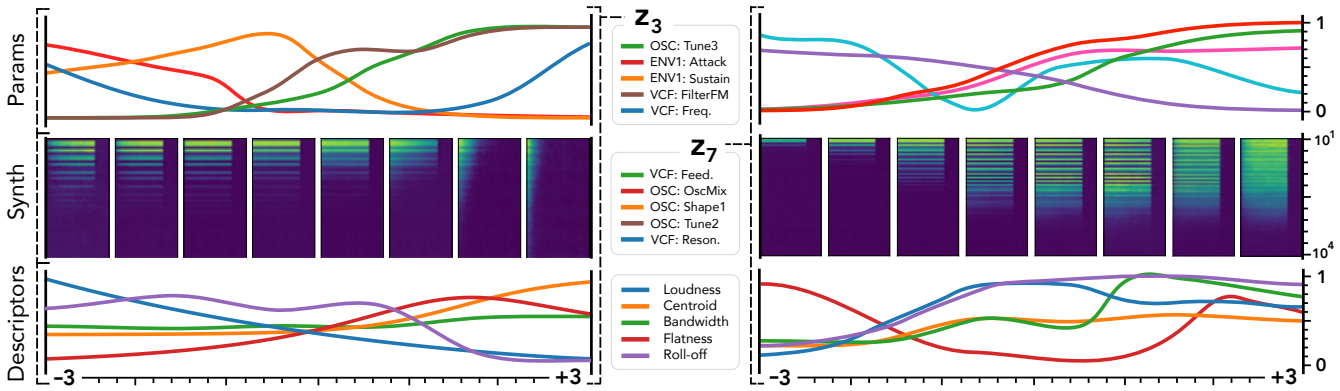


Figure 5: *Macro-parameters learning*. We show two of the learned latent dimensions $\mathbf{z}$ and compute the mapping $p(\mathbf{v}|\mathbf{z})$ when traversing these dimensions, while keeping all other fixed at $\mathbf{0}$ to see how $\mathbf{z}$ define smooth macro-parameters. We plot the evolution of the 5 parameters with highest variance (top), the corresponding synthesis (middle) and audio descriptors (bottom). (Left) $\mathbf{z}_3$ seems to relate to a *percussivity* parameter. (Right) $\mathbf{z}_7$ defines an *harmonic densification* parameter.

in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 741–742.

[3] Ricardo A Garcia, "Automatic design of sound synthesis techniques by means of genetic programming," in *Audio Engineering Society Convention 113*, 2002.

[4] Matthew John Yee-King, Leon Fedden, and Mark d'Inverno, "Automatic programming of vst sound synthesizers using deep networks and other techniques," *IEEE Transactions on ETCI*, vol. 2, no. 2, 2018.

[5] Diederik P. Kingma and Max Welling, "Auto-encoding variational bayes," *arXiv:1312.6114*, 2013.

[6] Irina Higgins, Loic Matthey, Arka Pal, Shakir Mohamed, and Alexander Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," *ICLR*, 2016.

[7] Danilo Rezende and Shakir Mohamed, "Variational inference with normalizing flows," in *International Conference on Machine Learning (ICML)*, 2015.

[8] Philippe Esling, Adrien Bitton, and Axel Chemla-Romeu-Santos, "Generative timbre spaces with variational audio synthesis," *21st International DaFX Conference, arXiv:1805.08501*, 2018.

[9] Christopher M. Bishop and Tom M. Mitchell, "Pattern recognition and machine learning," 2014.

[10] Casper K. Sønderby, Tapani Raiko, Lars Maaløe, Søren K. Sønderby, and Ole Winther, "How to train deep variational autoencoders and probabilistic ladder networks," *arXiv preprint arXiv:1602.02282*, 2016.

[11] Xi Chen, Diederik P Kingma, Tim Salimans, Ilya Sutskever, and Pieter Abbeel, "Variational lossy autoencoder," *International Conference on Learning Representations (ICLR)*, 2016.

[12] Ilya Tolstikhin, Olivier Bousquet, and Bernhard Schölkopf, "Wasserstein auto-encoders," *International Conference on Learning Representations*, 2017.

[13] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling, "Improved variational inference with inverse autoregressive flow," in *Advances in NIPS*, 2016, pp. 4743–4751.

[14] George Papamakarios, Theo Pavlakou, and Iain Murray, "Masked autoregressive flow for density estimation," in *NIPS*, 2017, pp. 2338–2347.

CROSS-MODAL VARIATIONAL INFERENCE FOR BIJECTIVE SIGNAL-SYMBOL TRANSLATION

Axel Chemla–Romeu-Santos[1,2], Stavros Ntalampiras[1], Philippe Esling[2], Goffredo Haus[1], Gérard Assayag[1]

[1] Laboratorio d'Informatica Musicale (LIM)
UNIMI, Milano, Italy
axel.chemla@unimi.it
stavros.ntalampiras@unimi.it
goffredo.haus@unimi.it

[2] IRCAM - CNRS UMR 9912
Sorbonne Université, Paris, France
esling@ircam.fr assayag@ircam.fr

ABSTRACT

Extraction of symbolic information from signals is an active field of research enabling numerous applications especially in the Musical Information Retrieval domain. This complex task, that is also related to other topics such as pitch extraction or instrument recognition, is a demanding subject that gave birth to numerous approaches, mostly based on advanced signal processing-based algorithms. However, these techniques are often non-generic, allowing the extraction of definite physical properties of the signal (pitch, octave), but not allowing arbitrary vocabularies or more general annotations. On top of that, these techniques are one-sided, meaning that they can extract symbolic data from an audio signal, but cannot perform the reverse process and make symbol-to-signal generation. In this paper, we propose an bijective approach for signal/symbol translation by turning this problem into a density estimation task over signal and symbolic domains, considered both as related random variables. We estimate this joint distribution with two different variational auto-encoders, one for each domain, whose inner representations are forced to match with an additive constraint, allowing both models to learn and generate separately while allowing signal-to-symbol and symbol-to-signal inference. In this article, we test our models on pitch, octave and dynamics symbols, which comprise a fundamental step towards music transcription and label-constrained audio generation. In addition to its versatility, this system is rather light during training and generation while allowing several interesting creative uses that we outline at the end of the article.

1. INTRODUCTION

Music Information Retrieval (MIR) is a growing domain of audio processing that aims to extract information (labels, symbolic or temporal features) from audio signals [1, 2]. This field embeds both musical and scientific challenges paving the way to a large variety of tasks. Such abundant industrial and creative applications [3] have attracted the interest of a large number of researchers with plentiful results. Among the diverse sub-tasks included in MIR, *music transcription* comprises an active research field [4, 5] which is not only interesting by itself but finds generic applicability as a sub-task for other MIR objectives (cover recognition, key detection, symbolic analysis). Music transcription can be described as associating symbols to audio signals composed of one or more musical instruments. Thus, this field embeds pitch and multi-pitch estimation tasks but also other musical dimensions, such as dynamics. Currently, most pitch estimation techniques are based on fundamental frequency detection [6]. However, such approaches may prove insufficient in multi-pitch contexts, where the need for more sophisticated approaches appears crucial.

In parallel, the recent rise of *generative systems* provided interesting alternatives to supervised machine learning approaches focusing on classification [7]. These unsupervised learning models aim to discover the inner structure of a dataset based on a reconstruction task. Such methods are usually defined as probabilistic density estimation approaches, Bayesian inference and auto-encoding structures. Among those, the *Variational Auto-Encoders* (VAE) provides a powerful framework, which explicitly targets the construction of a *latent space* [8]. Such spaces are high-level representations with the ability to reveal interesting properties about the inner structure of different types of data [8][9], and also more recently in audio [10]. Such learning procedures can be mixed with supervised learning to perform label extraction and conditional generation, showing the flexibility and the efficiency of this approach. Last but not least, latent spaces can also be explicitly shared by several systems acting on different data domains, providing an elegant way of performing domain-to-domain translation or multi-modal learning [11].

In this article, we propose a generative modeling approach to musical transcription by formulating it as a density estimation problem. Our approach allows to directly model pairs $(\mathbf{x}, \mathbf{y})$, where $\mathbf{x}$ represents the spectral features and $\mathbf{y}$ represents the corresponding musical annotations. Following a multi-modal approach inspired by Higgins & al. [12], we train two different VAEs on these separate domains whose latent representations are progressively shared through explicit distribution matching. In addition to providing a Bayesian formulation of musical transcription compatible with arbitrary vocabularies, our method also naturally handles the reverse audio generation process, and thus allows both *signal-to-symbol* and *symbol-to-signal* inference. Furthermore, direct data/symbol generation is also available by latent space exploration, providing an interesting method for creative audio synthesis. Finally, we bind our transcription approach with a novel source-separation approach, based on explicit source decomposition with disjoint decoders. The idea behind our method is to use the knowledge previously acquired on individual instruments in order to ease their recognition in the mixture signal. A novel form of inference network is trained on the product space of the decoders latent space, with additional latent dimensions that per-

forms Bayesian inference directly over mixture coefficients.

2. STATE OF THE ART

Here, we provide a brief state-of-the-art of the most common approaches for musical transcription. Then, we introduce *variational auto-encoders* and detail their use for cross-modal inference and generation.

2.1. Automatic music transcription

Automatic music transcription (AMT) aims at closing the gap between acoustic music signals and their corresponding musical notation. The main problem in AMT is detecting multiple and possibly overlapping in time pitches. Classical approaches for pitch and multi-pitch extraction are mostly based on spectral or spectral analysis using fundamental harmonics localization [13], such as the Yin algorihtm [6]. As these methods were originally conceived for monophonic signals, their extension to multi-pitch estimation contexts often implies recursive processes (multi-fundamental recognition, harmonic subtraction) that reduce their efficiency. In parallel, other methods relying on spectrogram factorization have been proposed. These are based on the decomposition of the spectrogram into a linear combination of non-negative factors. These include Non-negative Matrix Factorisation (NMF) [14] or probabilistic latent component analysis (PLCA) [15]. However, spectrogram factorization methods usually fail to identify a global optima, a limitation which led many researchers to hypothesize the need for supplementary external knowledge to attain more accurate decompositions [16, 17].

Recently, deep learning approaches have been proposed to address the multi-pitch detection problem. For instance, piano transcription task has been tackled via a variety of neural networks in [18, 19, 20, 21]. Interestingly, the MusicNet dataset [22] includes multi-instrument music conveniently structured to address polyphonic music transcription. Finally, a method based on convolutional neural networks is presented in [23], which aims at learning meaningful representations allowing accurate pitch approximation in polyphonic audio recordings.

2.2. Generative models and variational auto-encoders

2.2.1. Variational inference

Generative models define a class of unsupervised machine learning approaches aiming to recover the probability density $p(\mathbf{x})$ underlying a given dataset. This density is usually conditioned on another set of random variables $\mathbf{z}$, called *latent variables*. This set acts as a higher-level representation that controls the generation in the data domain. Formally, generative models can be described as modeling the joint probability $p(\mathbf{z}, \mathbf{x}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$, where $p(\mathbf{z})$ acts as a Bayesian *prior* over the latent variables. The *generative process* takes a latent position $\mathbf{z}$ to produce the corresponding probability density $p(\mathbf{x}|\mathbf{z})$ in the data domain. Conversely, we also want to estimate the *posterior* distribution $p(\mathbf{z}|\mathbf{x})$, that gives the latent distribution corresponding to a data sample $\mathbf{x}$. Retrieving this posterior distribution from a given generative process is called *Bayesian inference*, and is known to be a very robust inference framework. Unfortunately, this inference is generally intractable for complex distributions or requires limiting assumptions on both generative and inference processes. *Variational inference* (VI) is a framework that overcomes this intractability by turning Bayesian inference to an optimization problem [24]. To do so, variational inference posits a parametric distribution $q(\mathbf{z}|\mathbf{x})$ that can be freely designed, and optimizes this distribution to approximate the real posterior $p(\mathbf{z}|\mathbf{x})$. This optimization is performed thanks to the following bound

$$\log p(\mathbf{x}) \geq \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\big[p(\mathbf{x}|\mathbf{z})\big] + D_{KL}\big[q(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})\big] = \mathcal{L}_{\mathrm{ELBO}}(q) \tag{1}$$

where D_{KL} denotes the Kullback-Leibler divergence. We can see that maximizing the right term of this inequality inherently optimizes the evidence $p(\mathbf{x})$ of our model. This bound, called the *Evidence Lower-BOund* (ELBO), can be interpreted as the sum of a likelihood term $p(\mathbf{x}|\mathbf{z})$ and of a divergence term that enforces the approximated posterior $q(\mathbf{z}|\mathbf{x})$ to match the prior $p(\mathbf{z})$. This variational formulation is less restrictive than direct Bayesian inference, as it only requires the tractability of these two terms. Thus, we are able to model complex dependencies between $\mathbf{x}$ and $\mathbf{z}$ for both $p_\theta(\mathbf{x}|\mathbf{z})$ and $q_\phi(\mathbf{z}|\mathbf{x})$ while retaining the benefits of a Bayesian formulation [25].

2.2.2. Variational auto-encoder and cross-modal learning

To define the approximate distribution, we can model both generative and inference models as normal distributions

$$q(\mathbf{z}|\mathbf{x}) = \mathcal{N}\big(\boldsymbol{\mu}_q(\mathbf{x}), \boldsymbol{\sigma}_q^2(\mathbf{x})\big)$$
$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}\big(\boldsymbol{\mu}_p(\mathbf{z}), \boldsymbol{\sigma}_p^2(\mathbf{z})\big)$$

such that parameters $(\boldsymbol{\mu}_q, \boldsymbol{\sigma}_q^2)$ and $(\boldsymbol{\mu}_p, \boldsymbol{\sigma}_p^2)$ are respectively obtained by deterministic functions $f_\theta(\mathbf{x}; \theta)$ and $g_\phi(\mathbf{z}; \phi)$. When these functions are parametrized as neural networks, we obtain the original *Variatonal Auto-Encoder* (VAE) formulation proposed by Kingma & al. [8]. The prior is usually defined as an isotropic normal distribution $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, which acts as a regularizer to enforce the independence of latent dimensions. Similar to auto-encoding architectures, $f_\theta(\mathbf{x}; \theta)$ and $g_\phi(\mathbf{z}; \phi)$ are respectively called the *encoder* and the *decoder* of the system. These functions are jointly trained until convergence on parameters $\{\theta, \phi\}$ with a back-propagation algorithm. Despite the apparent simplicity of its formulation, this system allows very expressive encoding and generative processes while providing a highly structured latent space, whose smoothness is provided by the D_{KL} reconstruction term.

3. CROSS-MODAL VAE FOR MUSIC TRANSCRIPTION

3.1. Signal/symbol transfer through shared latent spaces

In this paper, we propose to reformulate the audio transcription problem as the estimation of a joint probability density $p(\mathbf{x}, \mathbf{y})$, where $\mathbf{x}$ represents the spectral information of the analyzed audio signal and $\mathbf{y}$ represents the corresponding set of symbolic information. Previous works showed the efficiency of VAEs for audio processing when used on spectral frames, in terms of both representational and generative abilities [10, 26]. However, we intend here to estimate not only the probability density $p(\mathbf{x})$, but also the joint probability density $p(\mathbf{x}, \mathbf{y})$. Considering $\mathbf{y}$ as label information, some approaches proposed to include an additional discriminator on the latent space, that is jointly trained during the learning process [27]. Here, we take inspiration from the SCAN approach proposed by Higgins & al., that trains a mirrored VAE on symbolic

data whose latent representation is constrained to match the latent space obtained from the signal VAE [12]. Hence, modelling our symbolic information as binary vectors $\mathbf{y} = [y_1, ..., y_L]$, we can train this VAE over the label space

$$q(\mathbf{z}|\mathbf{y}) = \mathcal{N}\big(\boldsymbol{\mu}_q(\mathbf{y}), \boldsymbol{\sigma}_q^2(\mathbf{y})\big)$$
$$p(\mathbf{y}|\mathbf{z}) = \prod_{i=1}^{L} \mathcal{B}\big(\mu_{p,i}(\mathbf{z})\big) \text{ or } \prod_{i=1}^{L} \text{Cat}(\boldsymbol{\mu}_{p,i}(\mathbf{z}))$$

where $\mathcal{B}\big(\mu_{p,i}(\mathbf{z})\big)$ denotes a Bernouilli distribution of mean $\mu_{p,i}$ for binary symbols, and $\text{Cat}(\boldsymbol{\mu}_{p,i}(\mathbf{z}))$ denotes a Categorical distribution of classwise probabilities $\boldsymbol{\mu}_{p,i}$ in the case of multi-label symbols. We enforce its latent representation to fit the one obtained with the signal VAE by adding a term to the ELBO

$$\mathcal{L}_{\text{scan}}(q) = \mathcal{L}(q) + D_{KL}\big[q(\mathbf{z}|\mathbf{x}) \| q(\mathbf{z}|\mathbf{y})\big] \tag{2}$$

such that the latent distributions provided by the two inference processes match for a given pair $(\mathbf{x}, \mathbf{y})$. The ordering of terms in the Kullback-Leibler divergence is chosen such that the distribution $q(\mathbf{z}|\mathbf{y})$ is forced to cover the whole mass of $q(\mathbf{z}|\mathbf{x})$. Hence, the correct label for a given $\mathbf{x}$ is encouragqed even for low-probability areas of $q(\mathbf{z}|\mathbf{x})$. Both VAEs are jointly trained, so that the latent representation obtained is a compromise between both auto-encoder performances. It should be noted that, as both signal and symbolic VAEs are independent, we are still able to perform semi-supervised learning for incomplete pairs $(\mathbf{x}, \mathbf{y})$ by training only one of the two auto-encoders.

3.2. Bidirectional signal-to-symbol mappings

Our approach extends the multi-pitch detection problem on several aspects. First, our model is inherently bi-directional as we can recover symbolic inference with the process

$$p(\mathbf{y}|\mathbf{x}) = p(\mathbf{y}|\mathbf{z})q(\mathbf{z}|\mathbf{x})$$

This can be understood as a Bayesian formulation of audio semantic labeling. Hence, multi-pitch transcription is simply a special case of our formulation, where $\mathbf{y}$ is defined as being solely the pitch information. Furthermore, we can also naturally handle *signal generation from symbolic constraints*, by taking the reverse process

$$p(\mathbf{x}|\mathbf{y}) = p(\mathbf{x}|\mathbf{z})q(\mathbf{z}|\mathbf{y})$$

such that we can recover the appropriate spectral distribution from the symbolic data, as depicted in Fig. 1. Another interesting property of our method is its applicability to arbitrary symbols. In this paper, we model symbolic information $\mathbf{y}$ as a triplet `[pitch class, octave, dynamics]`, where we add dynamics estimation to the pitch estimation task. Thus, we have $p(\mathbf{y}|\mathbf{x}) = p(\mathbf{y}^p|\mathbf{z})p(\mathbf{y}^o|\mathbf{z})p(\mathbf{y}^d|\mathbf{z})$, where each $p(\mathbf{y}^{\cdot}|\mathbf{z})$ is defined as a categorical distribution. We use this property to extend this method to multi-pitch applications, where $\mathbf{x}$ is a mixture signal with M different sources. Hence, we formulate the symbolic information as a product $p(\mathbf{y}|\mathbf{z}) = p(\mathbf{y_1}|\mathbf{z})...p(\mathbf{y_M}|\mathbf{z})$, where each $p(\mathbf{y}_{\cdot}|\mathbf{z})$ follows the previous specification. In addition to performing multi-pitch estimation, it also specifies the corresponding instrument if a given symbolic ordering is held during training. Finally, our formulation can be extended to polyphonic instruments in a straightforward manner. In this case, we simply replace the above conditioning by $p(\mathbf{y}|\mathbf{z}) = p(\mathbf{y}^p|x)p(\mathbf{y}^o|\mathbf{z})$, where we define $p(\mathbf{y}^p|\mathbf{z})$ to be a Bernoulli distribution over a one-hot pitch vector.

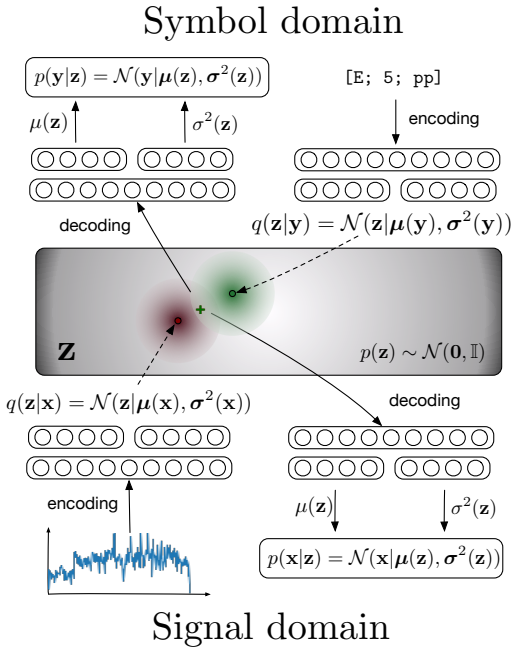


Figure 1: Multi-modal variational auto-encoding process. Our model is based on two separate variational auto-encoder, one in the signal domain and one in the symbolic domain, sharing a common latent space.

4. EXPERIMENTS

4.1. Datasets

To evaluate our approach we use the Studio One Line (SOL) [28], a database that contains solo instrument recordings for every note across their tessitura. Each note is recording over a range of different dynamics (*ff,mf,pp*). Here, we selected five instruments: violin, alto-sax, flute, C-trumpet and piano, for a total amount of 800 files. First, audio files are all resampled to a sample rate of 22050Hz. Then, we transform the raw audio data to the spectral domain by using a *Non-Stationary Gabor Transform* (NSGT) [29]. Interestingly, this multi-resolution spectral transform allows to define custom frequency scales, while remaining invertible. Here, we use a constant-Q scale with 48 bins per octave. For each model training, we split our dataset with 80% as training and 20% as test sets. As our dataset is composed of monophonic signals, we randomly create instrument signal mixtures during training such that every combination is seen during the training.

4.2. Models

To show the efficiency of our proposal, we rely on VAEs with very simple architectures. Nevertheless, depending on the complexity of the input data, we adjust the dimensionality of both the latent space and hidden layers. For single-instrument models we use 32 dimensions for the latent space, and define both encoding and decoding functions for the signal VAE as 2-layers multi-layer perceptrons (MLP) with 2000 hidden units. For the symbolic auto-encoder, encoding and decoding MLPs have 2 layers and 800 hid-

Table 1: Signal reconstruction and transfer performances

	$-\log p(\mathbf{x}\|\mathbf{z})$	ISD	$-\log p(\mathbf{x}\|\mathbf{y})$	*ISD*
Alto-Sax (Sax)	-694.1	0.093	-416.6	0.177
Violin (Vn)	-671.4	0.104	-551.1	0.151
Trumpet-C (TpC)	-706.9	0.073	276.71	0.35
Flute (Fl)	-706.2	0.076	-379.2	0.147
Piano (Pn)	-813.5	0.044	-361.13	0.112
Sax + Vn	-358.71	0.364	-27.37	0.852
Sax + Vn + Fl	-268.7	0.624	692.4	3.813

den units. For mixtures of two different instruments, the number of hidden units for the signal encoders/decoders are set to 5000. For the mixture of three instruments, hidden layers have 5000 and 1500 units for the signal and the symbolic encoders / decoders respectively. All models are trained using the ADAM optimizer, and we use the warm-up procedure that slowly brings the regularization from 0 to 1 during the first 100 epochs. As recommended by Higgins & al., the additional term presented in (2) is scaled up to a factor 10. The learning rate is first set to 1e-3, and is increasingly reduced as the derivative of the error decreases.

4.3. Evaluation

In addition to performing a standard evaluation on the test set, we also evaluate our model on a separate dataset containing recordings of flute arpeggios, scales and melodies [30] with source audio files and aligned MIDI files. Unfortunately, this dataset does not provide information about symbolic dynamics, so we do not evaluate the dynamics inference on this set. We compare the efficiency of our model with results obtained from a baseline approach. To this end, we rely on an architecture similar to our model, but designed in a supervised way to emphasize the gain provided by our model. This baseline classifier first performs a Principal Component Analysis (PCA) from the signal data to perform dimensionality reduction, mocking the compression between the input data and the latent space. Then, we use a 2-layer MLP with the same amount of hidden units than the corresponding symbolical decoder, to output the desired labels. The whole system is trained on a standard cross-entropy loss. The classifier is trained until convergence with the same optimization strategy.

5. RESULTS

In this section, we present the results of our methods. The source code, audio examples and additional figures and results are available on our support page `https://domkirke.github.io/latent-transcription/`.

5.1. Signal reconstruction and transfer performances

First, we analyze the results obtained on the SOL examples. Signal reconstruction and transfer scores are provided in Table 1, relying on two evaluation metrics. The first metric is the log-likelihood of the original spectrum with respect to the distribution decoded by the model. The second is the Itakura-Saito Divergence (ISD), a metric that reflects the perceptual dissimilarity between the original and reconstructed spectrum [31]. Both scores are presented for signal-to-signal reconstruction (left) and symbol-to-signal inference (right). In addition to these scores, reconstruction examples

Table 2: Symbolic inference reconstruction and classification results (successively pitch, octave and dynamics). Scores without parenthesis are reconstruction scores obtained within the symbolic domain, while scores in parenthesis are obtained when performing transfer from the signal domain

		$-\log p(\mathbf{y}\|\mathbf{z})$	Success Ratio (%)	*loose* (%)	Baseline (*loose*)
Sax	p	-1.0 *(-1.0)*	100% *(100%)*	-	*94%*
	o	-1.0 *(-1.0)*	100% *(100%)*	-	*97%*
	d	-1.0 *(-1.0)*	100% *(100%)*	-	*46%*
Vn	p	-1.0 *(-1.0)*	100% *(100%)*	-	*89.8%*
	o	-1.0 *(-1.0)*	100% *(100%)*	-	*99.0%*
	d	-1.0 *(-1.0)*	100% *(100%)*	-	*35.3%*
TpC	p	-1.0 *(-1.0)*	99.9% *(100%)*	-	*76.1%*
	o	-1.0 *(-1.0)*	100% *(100%)*	-	*99.8%*
	d	-0.998 *(-1.0)*	99.7% *(100%)*	-	47.8%
Fl	p	-1.0 *(-1.0)*	100% *(100%)*	-	*52.4%*
	o	-1.0 *(-1.0)*	100% *(100%)*	-	*81.8%*
	d	-1.0 *(-1.0)*	100% *(100%)*	-	*41.4%*
Pn	p	-1.0 *(-1.0)*	100% *(100%)*	-	51.6%
	o	-1.0 *(-1.0)*	100% *(100%)*	-	63.9%
	d	-1.0 *(-0.999)*	99.9% *(100.0%)*	-	40.0%
Sax + Vn	p	-0.534 *(-0.871)*	54.0% *(87.9%)*	62.6% *(81.6%)*	*65.3%*
	o	-0.782 *(-0.980)*	84.6% *(99.2%)*	94.9% *(88.7%)*	*79.1%*
	d	-0.712 *(-0.939)*	74.4% *(95.9%)*	82.4% *(66.3%)*	*52.0%*
Sax + Vn + TpC	p	-0.381 *(-0.725)*	38.6% *(75.0%)*	62.6% *(84.5%)*	*56.6%*
	o	-0.377 *(-0.641)*	42.4% *(67.8%)*	79.3% *(88.7%)*	*62.3%*
	d	-0.347 *(-0.616)*	34.6% *(62.4%)*	66.9% *(69.5%)*	*41.2%*

are depicted in Fig. 2. We can see that performances in both signal reconstruction and transfer decrease with the number of instruments, as the complexity of the incoming signal increases. Both reconstruction and signal-to-transfer scores are almost perfect in the case of solo instruments, providing convincing and high-quality sound samples generation. In the case of mixtures of two or more instruments, reconstruction scores maintain an acceptable performance, but symbol-to-signal transfer scores clearly decrease. This observation correlates with the decrease of performance observed in the symbolic domain, as discussed in the following sub-section.

5.2. Symbolic inference performances

Here, we evaluate the performances of our model in the symbolic domain. We provide in Table 2 four different classification scores, separately for each family of labels: octave, pitch class and dynamics. In the case of multi-instrument mixtures, these losses are averaged over every instrument of the mixture. Every column (except for the baseline) show two scores : the first are the scores obtained symbol-to symbol (reconstruction), and the second within parenthesis are the ones obtained signal-to-symbol (transfer).

The first loss, written $-\log p(\mathbf{y}|\mathbf{z})$, denotes the *likelihood* of the true labels with respect to the distributions decoded by the symbolic part of the VAE. The percent scores located at the right of the likelihood correspond to classification scores, obtained by taking the highest probability of the categorical distribution and obtaining the corresponding ratio of well-classified symbols. The first column, called *success ratio*, denotes the classification score obtained by the symbolical VAE. The second column, called *loose ratio*, is specific in the case of mixed instruments, considering a label to be correct regardless of the instrument (we will come back to the motivation behind this score). Finally, the last column display the scores obtained by our baseline classifier, that does not have symbol-to-symbol scores.

We note that symbolic reconstruction and signal-to-symbol scores are almost perfect in the case of single-source signals, outperforming the equivalent baseline system. We argue that is due to two main aspects of the proposed approach. First, thanks to the re-

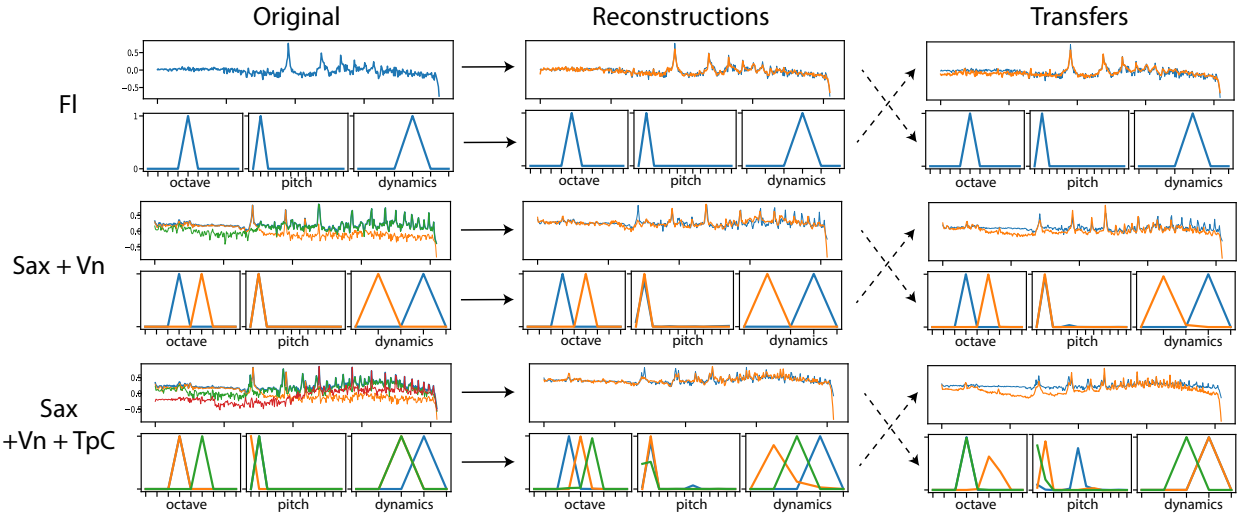


Figure 2: Signal & symbolic reconstruction samples in 1-instrument, 2-instruments and 3-instruments settings. The first column shows the original spectral & symbolic contents : for spectra, the blue line represents the final mixture, and the thinner lines the different components of the mixture. For symbols, labels are grouped by family and are symbolized by a peak at the correct label. The second column represents reconstruction results. For spectra, the orange line represents the reconstructed spectra (the original spectra in blue is left for comparison). Regarding the symbolic reconstructions, the corresponding categorical distributions are displayed right to the original one-hot vectors. The third column finally shows the transfer results, where we decode a latent position given by the encoder of the other domain.

construction task, the construction of the latent space is organized to reflect the inner structure of both signal and symbol domains. The latent space can be thus understood as a feature space, carrying higher-level information that allow signal/symbolic coupling to be more efficient. Second, the Bayesian approach matching the latent spaces allows a smoother and more efficient mapping than a deterministic approach, that would just provide pairwise mappings between incoming examples.

In the case of instrument mixtures, scores are decreasing as the complexity of both the spectrum and symbolic distributions increase. While the system still performs convincingly with two instruments mixtures, it struggles with mixtures of more than three instruments. We argue that this is partly due to a combinatorial problem, as can be seen when analyzing the *loose classification ratio* : indeed, it becomes harder for the model to accurately affect a label to the corresponding instrument as the number of different possible sources increase. This can be seen with the *loose ratio* in table 2 : where the classification ratio significantly increase if the label is considered correct regardless the affected instrument. This effect can also be seen figure 2, where some peaks are correct but unfortunately distributed to the wrong instrument. A more subtle strategy to tackle this effect has to be considered ; we leave this to a future work.

5.3. Monophonic flute transcription

Here, we analyze the results obtained with an external dataset of flute recordings, as depicted in Table 3. Performances in symbolic inference is still convincing, showing that our model does not suffer from strong over-fitting. Compared to the results obtained with the reference dataset 1, the reconstruction results obtained here have decreased. This is due to several points : first, we have trained the model solely on the stationary part of each instrument signals, such that the attack and release of the signal are not understood by our model. This anomaly is clearly perceptible when listening to the reconstructions. Second, a more subtle comparison between the reference dataset and this dataset showed important differences in terms of harmonic content. Specifically, a 1-octave lower harmonic is globally present in this dataset, and not in the reference one. This may explain the important decrease in octave classification, and may indicate that an increased amount of various instruments of the same type may be required to enforce the generalization of the model.

Table 3: Results obtained on an external flute dataset

Likelihood	ISD	Class. Ratio	Baseline
2648 (*1057*)	1.065 (*0.632*)	65.4% 81.9%	63.8% 76.8%

6. DISCUSSION AND FUTURE WORKS

6.1. Performance aspects

We think that the efficiency of the proposed approach mainly relies on the hybridization of its learning process, that combines both unsupervised and supervised learning. Indeed, while each encoder learns to extract domain-dependant features in an unsupervised manner, latent spaces are matched by enforcing a supervised coupling of signal/symbol pairs. This process thus intends to learn transferable features, that are then used by each decoder to project them back into their respective data domains. Furthermore, this

process allows the model to train on incomplete data, such that each domain's encoding / decoding functions can still be trained individually even if some signal / symbol couplings are missing. This means that the training method is scalable to bigger datasets where some symbolic information may be absent, such that incomplete data can yet be used to reinforce the reconstruction abilities of the system.

However, in spite of the strengths of the proposed approach, the actual state of the model suffers some issues, that we aim to tackle in the future. The first main issue is that, while the system performs well in the single-instrument case, its performance weakens with two instruments and clearly fails when applied on more. We think that this falls to several reasons. First, we think this is due to the capacity of the model, as we still use very simple systems even for complex signals like the 3-instruments case. Secondly, the complexity of the problem is such that, as we showed when comparing the loose and non-loose version of the classification ratio, the system struggles to correctly allocate the good label to the good instrument. We think that the incoming signal representation may be not precise enough to alleviate some ambiguities, as for examples in the case of octaves or fifths where instrument identification may be hard to disentangle. Furthermore, the model does not prevent instrument-wise symbolic outputs to focus on the same spectral components, and thus to perform redundant symbol predictions, and thus may also lead to a permutation problem.
Finally, another issue with the proposed model is that the temporal evolution is not considered by the system. Including temporal features could bring decisive enhancements : in addition to allow full-sound generation and increased pitch and dynamics inference, it may even be mandatory for applying our model to custom symbolic dictionaries (playing modes, temporal symbols such as trills...) and provide a substantial advantage over more casual pitch-detection methods.

6.2. Creative aspects

Finally, an important aspect of the proposed model is the diversity of creative applications it provides (see figure 3). As generation of both symbolic and signal content is both based on the latent space, one may use it as a continuous control space and meaningfully explore it in either an unsupervised or semi-supervised fashion. Indeed, this space can be explored in a fully unsupervised manner by direct interaction: both signal and symbol information are then generated, such that the user can have a direct symbolic feedback on the data he is generating. Alternately, it can also be used in a semi-supervised fashion, constraining the navigation to the distribution inferred by a given symbol or a given sound. For example, in our case, we can directly generate a note with given pitch, octave and dynamics by inferring a distribution with the symbolic encoder, and then navigate inside it to access the diversity of signals retained under the corresponding label information. This allows us, translating first MIDI information in pitch/octave/dynamics pairs, and then transferring this symbolic information in the signal domain, to generate audio content from a MIDI file. We list below various use cases that can be carried out by our model:

- **sequence generation**: we can use a sequence of labels to recover the corresponding distribution in the latent space that we can freely sample and/or navigate,
- **spectral morphing**: we can take two latent target distributions, and draw a trajectory that we can sample regularly to obtain a smooth transformation between the two target sounds,
- **free trajectory**: take a totally free trajectory in the latent space,
- **symbol extraction**: we can infer symbolic information from an incoming signal, and still train the corresponding signal encoder/decoder with the incoming data. This could be a particularly interesting feature especially in real-time contexts.

Corresponding examples for each of the above navigation strategies are given at `support webpage`. Finally, also note that, in our example, the vocabulary is easy to learn, such that retrieving the underlying distribution $p(\mathbf{y})$ of the symbolic data itself is not really useful. Indeed, the different labels are all independent, and are approximately equally distributed in their own domain. However, our system can also learn on much more complex vocabularies where learning the underlying distribution in the symbolic domain itself has an interest, and thus open additional perspectives for its use in creative and/or MIR applications.

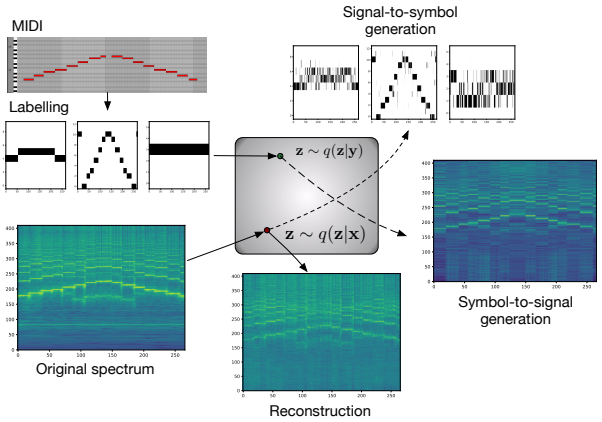


Figure 3: Diagram showing in-domain and cross-domain inference. If we give a spectrum to the signal encoder and draw from the corresponding distribution a latent position $\mathbf{z}$, we can decode it with the signal generation network (in-domain) and obtain the reconstruction. Alternatively, we can decode it with the symbol generation network (cross-domain) to perform *signal-to-symbol* generation, that here is equivalent to transcription. Reversely, if we draw a latent position from the symbol inference network with a given set of labels, we can decode it with the signal decoder to perform *symbol-to-signal* generation, that allows us to "play" a MIDI file with our model.

7. CONCLUSION

In this paper, we proposed a novel formulation for bijective signal/symbol translation, based on the latent space matching of domain-wise variational architectures. We studied the benefits and drawbacks of the proposed system, and concluded that while improvable this model performed well and proposed a very interesting alternative to signal-symbol algorithms, and furthermore provided

additional applications that were not possible in previous models. Indeed, our method is bi-directional, and performs well in both audio-to-symbol and symbol-to-audio prediction. Furthermore, our method is compatible with any kind or arbitrary symbolic information, and is then opened to user-defined vocabularies. Besides, as our model is based on a latent space that can be considered as a continuous control space, it is also opened to diverse creative uses as sequence generation, sound interpolation, or free navigation, whether in an supervised manner or semi-supervised manner, guided with symbolic information. For future work, we plan to solve the symbolic ambiguities that raise in the case of numerous instruments, to incorporate temporal features to allow dynamical features extraction, and to design user interfaces to make our model compatible with artistic practises.

8. REFERENCES

[1] J Stephen Downie, "Music information retrieval," *Annual review of information science and technology*, vol. 37, no. 1, pp. 295–340, 2003.

[2] F. Simonetta, S. Ntalampiras, and F. Avanzini, "Multimodal music information processing and retrieval: Survey and future challenges," in *2019 International Workshop on Multilayer Music Representation and Processing (MMRP)*, Jan 2019, pp. 10–18.

[3] Michael A Casey, Remco Veltkamp, Masataka Goto, Marc Leman, Christophe Rhodes, and Malcolm Slaney, "Content-based music information retrieval: Current directions and future challenges," *Proceedings of the IEEE*, vol. 96, no. 4, pp. 668–696, 2008.

[4] Anssi Klapuri and Manuel Davy, *Signal processing methods for music transcription*, Springer Science & Business Media, 2007.

[5] Emmanouil Benetos, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri, "Automatic music transcription: challenges and future directions," *Journal of Intelligent Information Systems*, vol. 41, no. 3, pp. 407–434, 2013.

[6] Alain De Cheveigné and Hideki Kawahara, "Yin, a fundamental frequency estimator for speech and music," *The Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1917–1930, 2002.

[7] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent, "Generalized denoising auto-encoders as generative models," in *Advances in Neural Information Processing Systems*, 2013, pp. 899–907.

[8] Diederik P Kingma and Max Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[9] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," *arXiv preprint arXiv:1401.4082*, 2014.

[10] Philippe Esling, Axel Chemla-Romeu-Santos, and Adrien Bitton, "Generative timbre spaces with variational audio synthesis," *CoRR*, vol. abs/1805.08501, 2018.

[11] Ming-Yu Liu, Thomas Breuel, and Jan Kautz, "Unsupervised image-to-image translation networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 700–708.

[12] Irina Higgins, Nicolas Sonnerat, Loic Matthey, Arka Pal, Christopher P Burgess, Matthew Botvinick, Demis Hassabis, and Alexander Lerchner, "Scan: Learning abstract hierarchical compositional visual concepts," *arXiv preprint arXiv:1707.03389*, 2017.

[13] Thomas Drugman and Abeer Alwan, "Joint robust voicing detection and pitch estimation based on residual harmonics," in *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[14] Daniel D. Lee and H. Sebastian Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, oct 1999.

[15] Madhusudana Shashanka, Bhiksha Raj, and Paris Smaragdis, "Probabilistic latent variable models as nonnegative factorizations," *Computational Intelligence and Neuroscience*, vol. 2008, pp. 1–8, 2008.

[16] G. Grindlay and D. P. W. Ellis, "Transcribing multi-instrument polyphonic music with hierarchical eigeninstruments," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 6, pp. 1159–1169, Oct 2011.

[17] G. J. Mysore and P. Smaragdis, "Relative pitch estimation of multiple instruments," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, April 2009, pp. 313–316.

[18] Rainer Kelz, Matthias Dorfer, Filip Korzeniowski, Sebastian Böck, Andreas Arzt, and Gerhard Widmer, "On the potential of simple framewise approaches to piano transcription.," *CoRR*, vol. abs/1612.05153, 2016.

[19] Siddharth Sigtia, Emmanouil Benetos, and Simon Dixon, "An end-to-end neural network for polyphonic piano music transcription," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 24, no. 5, pp. 927–939, May 2016.

[20] Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck, "Onsets and frames: Dual-objective piano transcription," *arXiv preprint arXiv:1710.11153*, 2017.

[21] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck, "Enabling factorized piano music modeling and generation with the maestro dataset," *arXiv preprint arXiv:1810.12247*, 2018.

[22] John Thickstun, Zaid Harchaoui, and Sham M. Kakade, "Learning features of music from scratch," in *International Conference on Learning Representations (ICLR)*, 2017.

[23] Rachel M. Bittner, Brian McFee, Justin Salamon, Peter Li, and Juan Pablo Bello, "Deep salience representations for f0 estimation in polyphonic music," in *ISMIR*, 2017.

[24] Tommi S Jaakkola and Michael I Jordan, "Bayesian parameter estimation via variational methods," *Statistics and Computing*, vol. 10, no. 1, pp. 25–37, 2000.

[25] Christopher M Bishop, *Pattern Recognition and Machine Learning*, Springer edition, 2006.

[26] Adrien Bitton, Philippe Esling, and Axel Chemla-Romeu-Santos, "Modulated variational auto-encoders for many-to-many musical timbre transfer," *arXiv preprint arXiv:1810.00222*, 2018.

[27] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling, "Semi-supervised learning with deep generative models," in *Advances in neural information processing systems*, 2014, pp. 3581–3589.

[28] Guillaume Ballet, Riccardo Borghesi, Peter Hoffmann, and Fabien Lévy, "Studio online 3.0: An internet" killer application" for remote access to ircam sounds and processing tools," *Journée dâInformatique Musicale (JIM)*, 1999.

[29] Gino Angelo Velasco, Nicki Holighaus, Monika Dörfler, and Thomas Grill, "Constructing an invertible constant-q transform with non-stationary gabor frames," *Proceedings of DAFX11, Paris*, pp. 93–99, 2011.

[30] Elena Agullà Cantos, "Flute audio labelled database for Automatic Music Transcription," Sept. 2018.

[31] C. Févotte, N. Bertin, and J. Durrieu, "Nonnegative matrix factorization with the itakura-saito divergence: With application to music analysis," *Neural Computation*, vol. 21, no. 3, pp. 793–830, March 2009.

[32] Emmanouil Benetos, Anssi Klapuri, and Simon Dixon, "Score-informed transcription for automatic piano tutoring," in *Signal processing conference (eusipco), 2012 proceedings of the 20th european*. IEEE, 2012, pp. 2153–2157.

[33] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103.

[34] Estefanía Cano, Derry FitzGerald, Antoine Liutkus, Mark D Plumbley, and Fabian-Robert Stöter, "Musical source separation: An introduction," *IEEE Signal Processing Magazine*, 2018.

[35] Aditya Arie Nugraha, Antoine Liutkus, and Emmanuel Vincent, "Multichannel audio source separation with deep neural networks.," .

[36] Aaron van den Oord, Yazhe Li, and Oriol Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

EXPLORING THE SOUND OF CHAOTIC OSCILLATORS VIA PARAMETER SPACES

Georg Essl

University of Wisconsin - Milwaukee
Milwaukee, Wisconsin, U.S.A.
essl@uwm.edu

ABSTRACT

Chaotic oscillators are exciting sources for sound production due to their simplicity in implementation combined with their rich sonic output. However, the richness comes with difficulty of control, which is paramount to both their detailed understanding and in live musical performance. In this paper, we propose perceptually-motivated parameter planes as a framework for studying the behavior of chaotic oscillators for musical use. Motivated by analysis via winding numbers, we extend traditional study of chaotic oscillators by using local features that are perceptually inspired. We illustrate the framework on the example of variations of the circle map. However, the framework is applicable for a wide range of sound synthesis algorithms with nontrivial parametric mappings.

1. INTRODUCTION

The use of nonlinear and chaotic oscillators for sound synthesis poses a trade-off. On the one hand, many of them have very simple algorithmic realizations and very diverse and rich sonic outcomes. On the other hand, the output becomes complex with increased nonlinearity. This complexity is often referred to as chaos and is characterized by drastic changes in output in response to minor parametric changes and a sensitivity to initial values. Because of their desirable properties, chaotic oscillators have repeatedly found their way into sound synthesis research and musical practices [1, 2, 3, 4, 5, 6, 7, 8, 9] and recently has found a renewed interest with in the context of live musical performance [10].

Despite this longstanding interest, there are a number of areas which are central to understanding chaotic oscillators for musical use which remain under-explored. Individual chaotic oscillators are usually studied on specific parameter examples [11] while making it difficult to predict outcomes under change of parameters. A large set of algorithms that exhibit chaotic oscillation exist [10], but their relationship is poorly, if at all, understood. This makes it difficult to chose oscillators with intention.

In live music performance, the ability of the musician to navigate the possibilities of the sound synthesis algorithm can be critical. This relationship of synthesis parameters to performer control is recognized as a central problem in musical instrument design known as the *mapping problem* [12]. However the mapping relationship in this case is complex, calling for strategies that support the performer's ability to have a sense of predictability of the performance choices made.

The purpose of this paper is to propose a framework for making the relationship of parametric choice and sonic outcome of chaotic oscillators accessible visually. More specifically, in this paper we propose perceptually motivated parametric planes (sometimes also called "atlases" [13]). These are parametric spaces which visualize features of the chaotic oscillator over a range of performance parameters. The visual space helps us understand dependence of parameters to outcomes, and the selection of features allows us to probe different aspect of the outcome potential.

2. RELATED WORK

Visualizations have long played an important role in the study of nonlinear oscillators. Perhaps closest to our proposed approach are two forms of visualization: (1) Arnold Tongues [14] and (2) Spectral Bifurcation Diagrams [15, 16].

Originally, Arnold tongues referred to stability regions of the circle map rendered as a plane with linear frequency Ω of the map as one coordinate, and increasing nonlinearity k as the other [14]. Since then, Arnold tongues have been more broadly referred to as regions of stability representing mode-locking in chaotic dynamical systems in general [17, 18, 19, 20, 13]. The literature discussing aspects of Arnold tongues is, in fact, so vast that it cannot be sensibly included here. The pervasiveness of this approach to studying nonlinear dynamics is one of the main motivations of using it as a starting point for our proposed approach. This should allow comparison to the vast existing body of research on nonlinear oscillators. Our proposed work can be understood to generalizing the parameter planes of Arnold tongue to depict behavior that is perceptually motivated in nature and in particular studies the use of spectral content.

Spectral Bifurcation Diagrams constitute another visualization approach for nonlinear phenomena proposed in the context of studying their acoustics [15, 16]. They are rendered by drawing the short-time spectrum of the iterative map with increasing iterative steps. Hence it is a method for depicting the onset of chaos over time within an oscillatory system. Our approach also uses spectral content. However, we are looking for different parametric relationships, and primarily those useful for control changes.

Within the realm of musical use of chaotic oscillators, this work is related to recent proposals to modify chaotic oscillators by injecting delay lines to make them more friendly to control by musical performers and the associated emergence of a large number of proposed nonlinear oscillators [10] . However, our understanding of the detailed aspects of the effect of delay lines to diverse nonlinear oscillators as well as the relationships and differences between these oscillators is currently preliminary. Our work looks to provide a concrete framework to study nonlinear oscillators for musical use and supplement their performance visually, hence is intended to help us systematically probe these open questions.

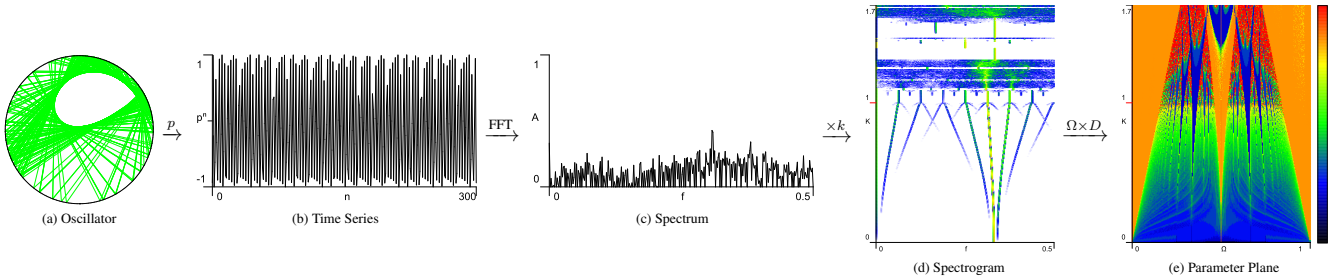


Figure 2: Visual representations of a nonlinear oscillator depicting intermediate steps of constructing parameter planes: (a) Circle Map as example of an iterative map, (b) projection of the map into a time series, (c) amplitude spectrum of the time series, (d) spectrogram of the time series over increased nonlinearity k with fixed linear frequency Ω, (e) parameter plane computing the PeakSparsity feature over a range of Ω.

3. CIRCLE MAPS AS NONLINEAR OSCILLATOR

While the proposed framework is not limited to a specific nonlinear oscillator, a concrete example will facilitate the discussion of the method. For this reason we will utilize circle maps [8, 11].

The most general form of circle maps refers to all mappings from the circle to itself [8]. Here we will restrict this to a perturbative form of the linear oscillator defined as follows:

$$y_{n+1} = \left(y_n + \Omega - \frac{k}{2\pi} f(y_n) \right) \mod 1 \tag{1}$$

Here $f(\cdot)$ refers to a nonconstant function that reflects a chosen type of nonlinearity. k is the strength of the nonlinearity. If k is 0 then the map is linear. Ω is a linear increment on the circle and reflects the distance traveled by a linear oscillator for one time step. y_n is the iterative position on the circle. y_0 is an initial value of the position. Technically the circle map hence has three parameters, Ω, k, and y_0. Finally the choice of the nonlinear function $f(\cdot)$ provides a further source of variation. One of the most widely studied circle map uses a sine function (equation (7)) as nonlinear perturbation and we will call this particular instance of the circle map the *sine circle map* [21]. In Figure 2(a) successive iterations of a sine circle map are connected by straight lines.

3.1. Choice of Projection

There is no set way to arrive at a time series from an iterative map such as the circle map. For our purpose, we follow [8] and define a *projection* $p(\cdot)$ onto an orthogonal axis:

$$p_n = \sin(2\pi y_n) \tag{2}$$

This choice of projection is justified because it mimics the projection used to construct a linear discrete sinusoidal oscillator from a phase function. In this case a constant increment Ω in the phase yields a sinusoidal output. Hence we can interpret the circle map equation (1) as an iterative nonlinear phase function, where Ω corresponds to the constant phase increase of a linear oscillator, and where k corresponds to the strength of a nonlinear contribution $f(\cdot)$ to phase change.

It is important to note that this choice of projection is arbitrary. One could use $\cos$ or in fact any other orthogonal projection of the circle for $p(\cdot)$. One could also simply interpret the y_n as time series. A wide range of further choices are also possible. We do believe that the $\sin$ is a natural choice as $y_0 = 0$ corresponds to a zero phase sinusoidal oscillation, and because it allows the interpretation of the circle map as a one-parameter nonlinear perturbation of a linear oscillator, hence grounding the map in a well-understood base configuration.

This particular justification is specific to this nonlinear dynamical system, and other dynamical systems may justify different projection functions.

4. PARAMETER PLANES AS FRAMEWORK FOR ANALYSIS

For the purpose of this paper we refer to *parameter planes* as two-dimensional visual renderings of behavior of a sound synthesis algorithm under the variation of two control parameters. This is different from other dense visual renderings such as a spectrogram, where typically the coordinate parameters are not necessarily directly related to control parameters, though the content of the visualization may well be responsive to parameter changes.

Assume any synthesis algorithm with two or more control parameters. The parameter plane would be the rendering along two control parameters (or combinations of control parameters that reduces their number to two). The local visual content is the output of the synthesis algorithm reduced by some feature process to a one-dimension that is then rendered as color gradient. Hence, parameter planes constitute the relationship of control parameter to some feature-specific output.

A pertinent established example of a parameter plane in the study of nonlinear oscillators are Arnold tongues. Arnold tongues are rendered by varying the two control parameters (Ω, k) of equation (1) or comparable comparable parameters of other nonlinear oscillators. The local feature traditionally used are winding numbers, which allow for the classification of the type of winding, and was used to discover mode-locking for mild nonlinearities in nonlinear oscillators [14].

Our proposed use of parameter planes can be understood as a generalization of Arnold tongue planes, but with the computation of the local value changed from the winding number to some other local feature of interest that is more suitable for understanding of audio properties.

The overall process through intermediary steps is depicted in Figure 2. We are given a nonlinear oscillator (Figure 2(a)). The choice of connecting the nonlinear oscillator to a time-series (Figure 2(b)) that is taken to be a sampled audio signal is not obvious. Hence we require a choice of function (*projection*) $p(\cdot)$ that establishes this relationship. Over a certain range of time-steps, we

compute local features. In our examples all newly introduced local features are spectral in nature, hence we compute the *FFT* (Figure 2(c)). It is important that this and the next step are flexible. One could choose a non-spectral feature (such as zero-crossings). In the spectral case, one can compute a further intermediary step of arranging a spectrogram over one control parameter (Figure 2(d)). This step is also not strictly required but we found it to be a very useful intermediate visualization to probe aspects of spectral evolution under parameter changes. Finally, a *feature mapping* $D(\cdot)$ derives from the spectrum a one-dimensional local representation that is arranged into the plane (Figure 2(e)) over the control parameter which we label here k and Ω for consistency with our chosen example of the circle map (section 3).

The specific aspects that are of interest for investigation and performance dictate further choices in this process. Oscillators can have initial transients. While these can be interesting subjects on their own right, for the purpose of this paper we have sought to capture steady state behavior. For this reason all figures in this paper, unless otherwise noted, compute the first 1000 iterations without consider them for rendering[1] The length of the time series, and the size of the FFT are further choices that can impact fidelity of the rendering and the information that will be drawn out. The following process was used to compute spectral information: A fast Fourier transform (FFT) was performed with a length of 8192 bins. Time series intervals were rendered for half that length and zero padded. The signal was weighted using a Blackman window function. Only the amplitude spectrum was considered. Neighboring bins were averaged to reduce the number of bins to the figure widths rendered in the paper (300 points). All aggregate feature computations were computed on the 300 bin averaged spectrum to create maximum relation between visual representations of spectra, spectrograms, and parameter planes. We do not propose this as a necessary aspect of the process. In other applications it may be desirable to compute feature on larger (or smaller number of bins).

Finally, a color gradient is chosen to render the one-dimensional feature D. This relationship can be linear or be further modified by a transfer function. It is common to use the logarithm on one-dimensional audio measures. In this paper all figures based on spectral information were color rendered on a logarithmic scale, normalized to the range of the color gradient.

Figure 2(e) shows the chosen color column to the right. The normalized color gradient ranged from zero to one and is composed of four equal intervals of length $1/4$. It is computed as linear interpolation of RGB colors black $(0, 0, 0)$, blue $(0, 0, 255)$, green $(0, 255, 0)$, yellow $(255, 255, 0)$, and red $(255, 0, 0)$. Hence black reflecting a value of 0, pure green reflecting a value of 0.5, and pure red reflecting a value of 1. For spectrogram computations we replaced black with white to achieve a white background look, with the color gradient otherwise unchanged.

Our Java implementation of the process for all features discussed in the next section took between 30 seconds to a few minutes depending primarily on the spectral window size used. In order to facilitate interactive use (for musical purposes or in interactive demonstration) we stored the resulting array of features in a file for instantaneous recall.

[1] 1000 was chosen after it was observed that this was sufficient to capture rare long transitions in parameter ranges of interest. The vast majority of transitions are very short and do not exceed just a few steps.

5. LOCAL FEATURES

A key component of computing parameter planes is finding local features that fill the plane to inform about the underlying behavior of the synthesis algorithm at a parameter point (Ω, K). The choice of this local feature carries reduces all of the behavior of the synthesis algorithm to a number, with the goal for this number to represent information of interest. By spanning a plane that allows one to inspect the evolution of local behavior under parameter change.

In order to understand that choice of feature we first discuss a widely established example in the study of nonlinear dynamical systems, which will motivate why we need more measures to study the sound of these systems.

5.1. Winding Numbers and Arnold Tongues

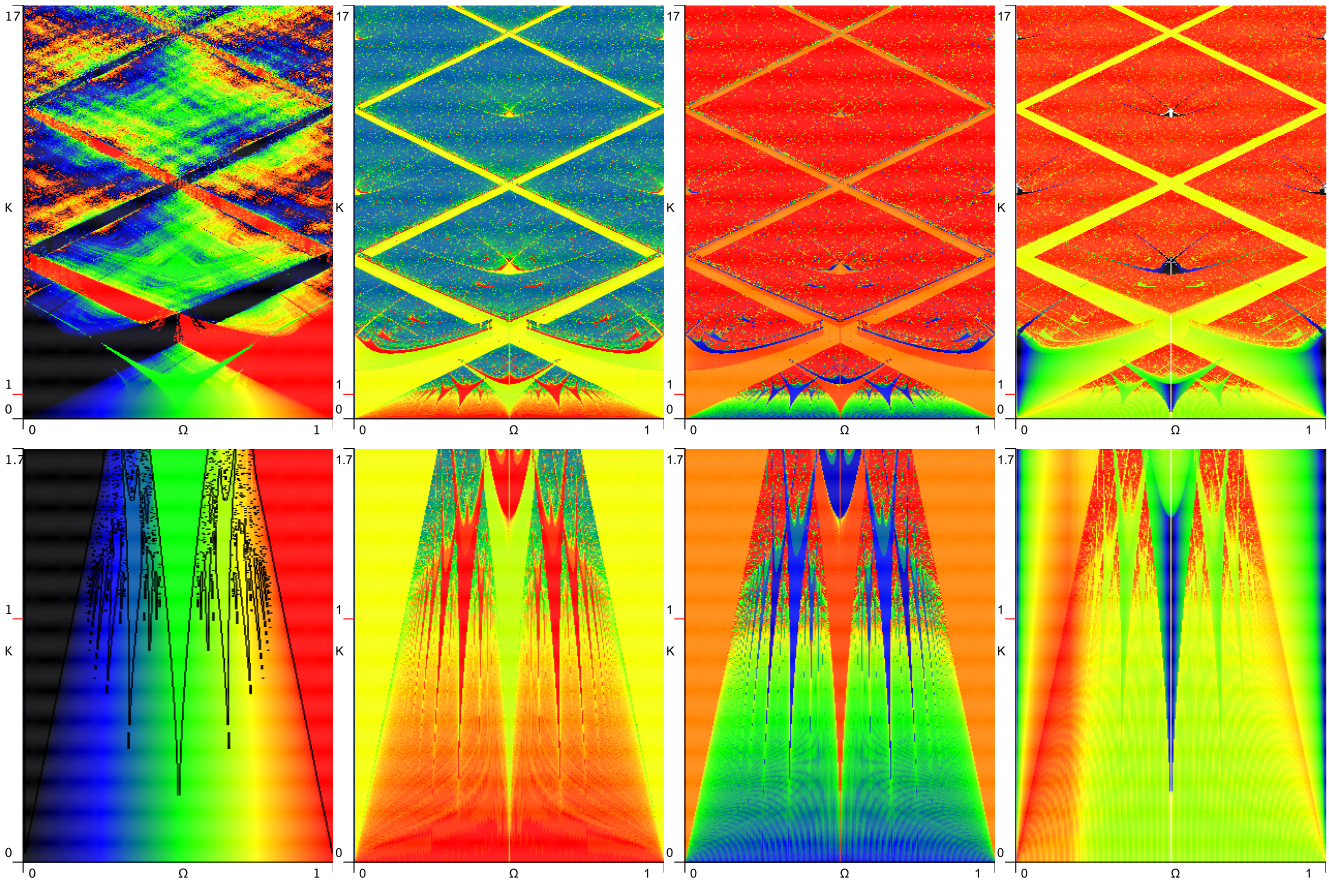
The construction of Arnold tongues is an example of a parameter plane visualization. Arnold tongues are areas in the parameter plane which exhibit mode locking [14, 17, 18, 22, 23] as well as provide a range of information about the transition into chaotic regimes. The *winding number* W (interchangably also referred to as rotation number [18]) is the local feature used for their construction. Given an iterator producing incremental states y_n, the winding number W is the average long-term map increment, hence describes the long-time average phase of the map. For iterations that are not subject to periodicity induced by modulo operations, they can be computed as [8]:

$$W = \lim_{n\to\infty} \frac{y_n - y_0}{n} \tag{3}$$

In practical computation the length of winding n is finite but sufficiently large to overcome initial transients. We chose 1000 for our depictions of Arnold tongues throughout this paper, matching our assumed maximum transients.

Computing the winding number over parameters of nonlinear oscillators led to the discovery of mode-locking (though phase-locking [24] is probably a more precise term in the case of the circle map). With mild increase in nonlinearity one finds that nonlinear oscillators tend to form areas of constant winding number even though the underlying parameters are changing. If depicted as a function of constant nonlinearity these form a staircase shape, which have been called "devil's staircase". When plotted over increasing nonlinearity, the family of devil's staircases widen, forming tongue-like shapes in the plane. We see depictions of Arnold tongues for the sine circle map in the bottom part of Figure 3(a). For low nonlinearity $k < 1$ we observe that widening regions of locked modes occur.

As we will see, Arnold tongue-like properties are very persistent features, even if we change the local feature we are computing. However, the winding number is still not a great feature to use to study sonic qualities of nonlinear dynamics. The winding number following equation (3) can be interpreted as a very-long-time averaged behavior of the state of the dynamical system. In fact, the equation can be formulated such that it is identical to the computation of the mean with the length of the mean being extended to infinity [24]. From this we observe that the winding number does not capture more localized properties of the dynamics, such as local periodicity, short-term spectral changes and so forth. Furthermore, given that we project the circle map into a discrete sine oscillator, giving us the interpretation of a phase function, we realize that the winding number here is really the long-term average



(a) Winding Number (b) MeanBalance (c) PeakSparsity (d) Entropy

Figure 3: Comparison of features computed in the parameter plane: (a) winding number (Arnold Tongues), (b) Spectral Mean Balance, (c) Spectral Peak Sparsity, (d) Spectral Entropy. (top) $k = [0, 13.\dot{3})$ (bottom) $k = [0, 1.3\dot{3})$ Arnold tongues use black contours to emphasize mode-locking transitions.

phase. While the precise response to phase in sound is a complicated phenomena, and it can play a role in specific settings [25], our ear generally does not associate perceptible qualities to phase. Hence a feature that primarily looks at phase is not the best feature to consider unless one is trying to capture very specific effects.

There are numerous aspects of iterative maps that could replace the winding number. For example, one can compute initial transient times by computing the winding number incrementally and checking when the computed winding numbers of successive steps fall below a value ϵ. The iteration count n then provides a measure of the duration of the transient. One can also probe for other mathematically interesting properties such as fixed points [11]. These would correspond to DC signals and help identify regions of silence, which is easily captured as part of richer and more broadly descriptive features.

5.2. Perceptually Motivated 1-D Features

Understanding the results of a synthesis algorithm is eminently perceptual, motivating the need for visual guidance of complex synthesis methods to predict perceptually interesting aspects of the expected sonic outcomes.

Perceptual cues are a particular form of features derived from audio time series. A wide array of audio features have been proposed [26] for they play important roles in all forms of audio analysis, including in creating meaningful starting points for machine classification. The body of audio features are dominated by needs of understanding human vocalization and musical signals. While nonlinear oscillators produce resonant spectra quite well suited by this body of work, especially at higher nonlinearities the sounds exhibit a range of differing noisy outcomes [11]. The literature dealing with noise is substantially smaller [27, 28]. This may go along with our understanding of human sound perception having substantially more detailed results for resonant spectral content than otherwise [29]. For the purpose of presenting the method, we were looking for features that would give clear visual differences over the same parameter space. After initial experimentation, we decided to present here three features that we believe to have a particular motivation given the output of nonlinear oscillators. Nonlinear oscillators have a range of effects: At low nonlinearities they tend to behave similar to wave shaping [8] in that the nonlinearity introduces spreading resonant spectra. At higher nonlinearity complicated noise-like patterns can emerge. Hence we were interested in features that could carry information about both resonant and noisy content.

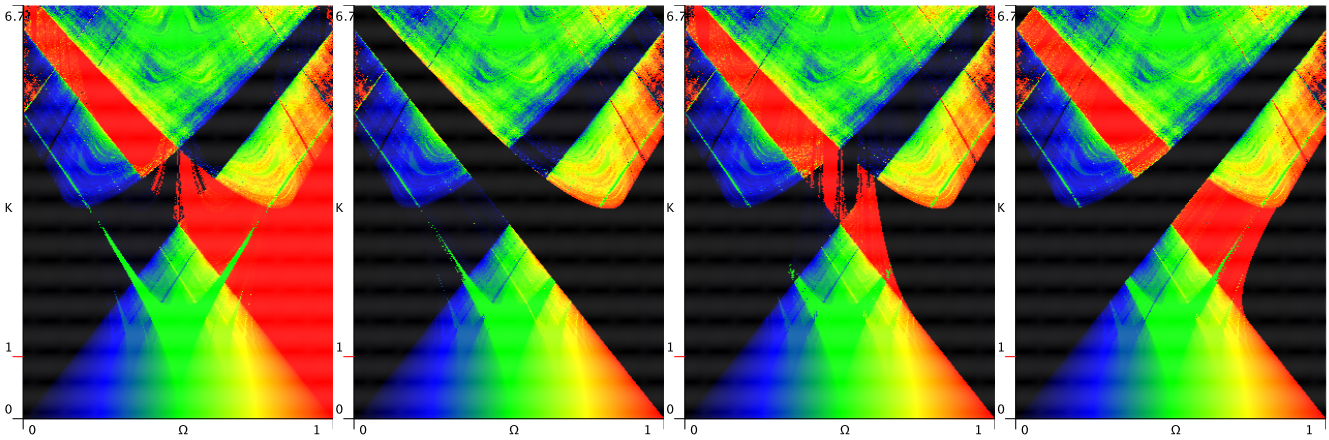


Figure 4: Arnold Tongues (winding numbers) of the sine circle map for initial values $y_0 = 0.0, 0.25, 0.5$, and 0.75.

5.2.1. *MeanBalance Feature*

The MeanBalance feature is quantifying if a spectrum is dominated by contributions that lie above or below the mean of the spectrum. The intuition is that a flat but noisy spectrum will be balanced with respect to the number of values that lie above and below the mean. For a spectrum of distinct and sparse narrow peaks, most values lie below the mean due to the sparsity of the peaks. A spectrum characterized that occasionally dips in the spectrum would, conversely, have most values lie above the mean. More precisely, the feature is computed as:

$$\text{mean} = \frac{1}{N}\sum_{n=1}^{N} f_n \qquad X[c] = \begin{cases} 0, & \text{if } c \text{ is false} \\ 1, & \text{otherwise} \end{cases}$$

$$b = \sum_{n=1}^{N} X[b > \text{mean}] \qquad a = \sum_{n=1}^{N} X[b < \text{mean}]$$

$$D_{\text{MB}} = \begin{cases} b/a & \text{if } a > 0 \\ 1 & \text{if } a = 0 \text{ and } b = 0 \\ b & \text{otherwise} \end{cases} \tag{4}$$

mean is the customary mean over all FFT bins f_n. b is the count of FFT bins that lie below the mean of the spectrum, and a is the count of FFT bins above the mean of the spectrum. If $a = 0$, we define $D_{\text{MB}} = b$. If all bins are equal to the mean $D_{\text{MB}} = 1$.

We expect D_{MB} to return high values for sparse peak spectra, return average values for flat spectra, and return low values for spectra with occasionally dips.

5.2.2. *PeakSparsity Feature*

The PeakSparsity feature takes an integral approach to estimating how peak-dominated a spectrum is. The core idea is that the area underneath the spectrum, characterized by the mean, can be arrived at in various ways. If the spectrum is peaky, we expect that there is more area under a peak. Hence high peaks will occupy more of the area than a flat spectrum. The spectrum sorts the peaks in the spectrum and then counts how many sorted bins are needed to arrive at the mean. If lots of area in within a few very high peaks it will take only a few bins to reach the mean. If the spectrum is flat, we expect to have to count all bins to arrive the mean. Hence PeakSparsity is a measure of sparsity peaks in the spectrum. It is computed as follows:

$$\text{area}_n = \sum_{m=1}^{n} \text{sort}_m(f) \qquad \text{sort}_n(f) \text{ are sorted FFT bins}$$

$$\text{c}_n = \begin{cases} 1 & \text{if } \sum_{m=1}^{n} \text{area}_m \leq \text{mean} \\ 0 & \text{otherwise} \end{cases}$$

$$D_{\text{PS}} = \sum_{n=1}^{N} c_n \tag{5}$$

5.2.3. *Spectral Entropy Feature*

Spectral entropy [27] is the computation of the Shannon entropy over all FFT bins:

$$D_{\text{Ent}} = -\frac{1}{\log N}\sum_{n=1}^{N} f_n \log(f_n) \tag{6}$$

f_n are the FFT bins and N is the number of bins up to half Nyquist frequency. Description by Shannon entropy leads to information-theoretic interpretation of the measure. High entropy corresponds to the requirement to describe more information content. Hence it is a kind of measure of information-theoretic complexity of the spectrum. We expect sparse spectra to contain less information than dense spectra, but per chance have some information-theoretic differentiation between different forms of dense spectra.

6. PARAMETER SPACES FOR CIRCLE MAPS

We are now ready to proceed to give concrete illustrations of parameter planes using these chosen features on variations of the circle maps introduced in section 3.

6.1. Comparisons of features

Figure 3 compares different choices of local features computed in the (Ω, k) parameter plane for the sine circle map. The figure shows both a weak nonlinearity regime ($k \in [0, 1.7)$) in the bottom row as well as a strong nonlinearity region ($k \in [0, 17)$) in

the top column. We include the winding number as classical local feature on the left.

Mild nonlinearity: Each of the four illustrated features highlight very different structures in the very low nonlinearity regime $k \in [0, 0.3)$. The winding number does not show the emerging resonant spectral aspects that emerge in this region. Specifically the PeakSparsity feature shows an overall increase in resonant peaks with additional fine structure. While Entropy displays a different structure than any other feature it shares with the winding number the lack of detail in this very mild nonlinearity regime.

nonlinearity near invertibility: The sine circle map becomes non-invertible at $k \geq 1$ [8]. This point is marked on the k-axis with a red marker. We expect more pronounced nonlinear effects to occur after this point, and early cascades into chaos are possible. Overall we see that both the MeanBalance and the PeakSparsity measure show substantial fine structure in the region $k \in [1, 1.3\dot{3})$. While the winding number exhibits a lot of structure it does not align well with these transitional properties. Entropy shows little change until chaos onset, when we see a smooth transition.

Persistent macroscopic properties and their variation: Arnold Tongue shapes are visible with all chosen features which make them very robust properties. This means that they do not merely correspond to mode-locking, but also have spectral and entropic effects. Unsurprisingly, fixed point regions (crossing diagonal regions) can be identified with all features as they correspond to constant value, and DC component spectral content. It is noteworthy however that the Entropy feature provides additional information in fixed point regions because it differentiates information content by the level of the DC component. These feature as not perceptually relevant, suggesting that unmodified entropy may not be an ideal feature for perceptually motivated modeling.

Strong nonlinearity: For a parameter range of $k \gg 1$ chaotic behavior, potentially interspersed with fixed-point regions, that then re-cascade into chaos, are dominating the behavior of the sine circle map for all features. Here the winding number presents the most diversity in value ranges, but is poor at resolving some visible trends that emerge using the other features. There are some subtle differences which of these trends are highlighted how strongly between MeanBalance, PeakSparsity and Entropy. For example using Entropy we see trend lines at the half-way point between fixed point diagonal regions. MeanBalance shows more structure that follows central fixed point and stability regions around $\Omega = 0.5$.

6.2. Initial Value Sensitivity

As illustrated on a number of examples [11], the circle map exhibits initial value sensitivity, a feature that is known to be typical of chaotic systems, and a widely propagated narrative in popular culture involving butterfly wings. While known examples [11] make clear that observable and clearly perceptible differences due to initial value differences can occur through the parameter plane, inspection of the parameter plane allows us to look for initial value effects more broadly. Arnold Tongues exhibit initial value sensitivity as discernible differences in the plane, in particular in areas that approach recurrent chaotic regimes, as well as in the behavior of crossing fixed point regions.

Figure 4 exhibits this effect. We see that for increasing initial values, certain regions change. With respect to fixed point regions, we see that for initial values below 0.5, one side overlaps the other, while this directionality is flipped for initial values above 0.5.

In order to understand if these differences are perceptually relevant, we observe initial value conditions on a perceptual feature instead of the winding numbers used in Arnold tongues (Figure 3). We see that the features at the top end of fixed point diagonals in the winding number plane is not captured by any of the other measures. Some of these effects can perceptually be treated as phase effects and hence are not usually perceptually relevant. Mathematically they do not appear in our features due to using only the amplitude spectrum.

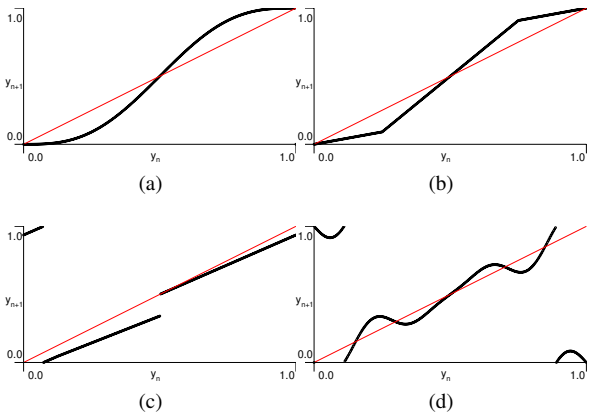


Figure 5: Four nonlinearities functions explored: (a) Sine, (b) Triangle, (c) Piecewise linear cardiorespiratory coupling model [20], (d) truncated Fourier-series. Figure from [11].

7. VARIATION OF NONLINEAR FUNCTION

To explore the impact of the variation of the nonlinear function $f(\cdot)$ we use four exemplar functions discussed in [11]:

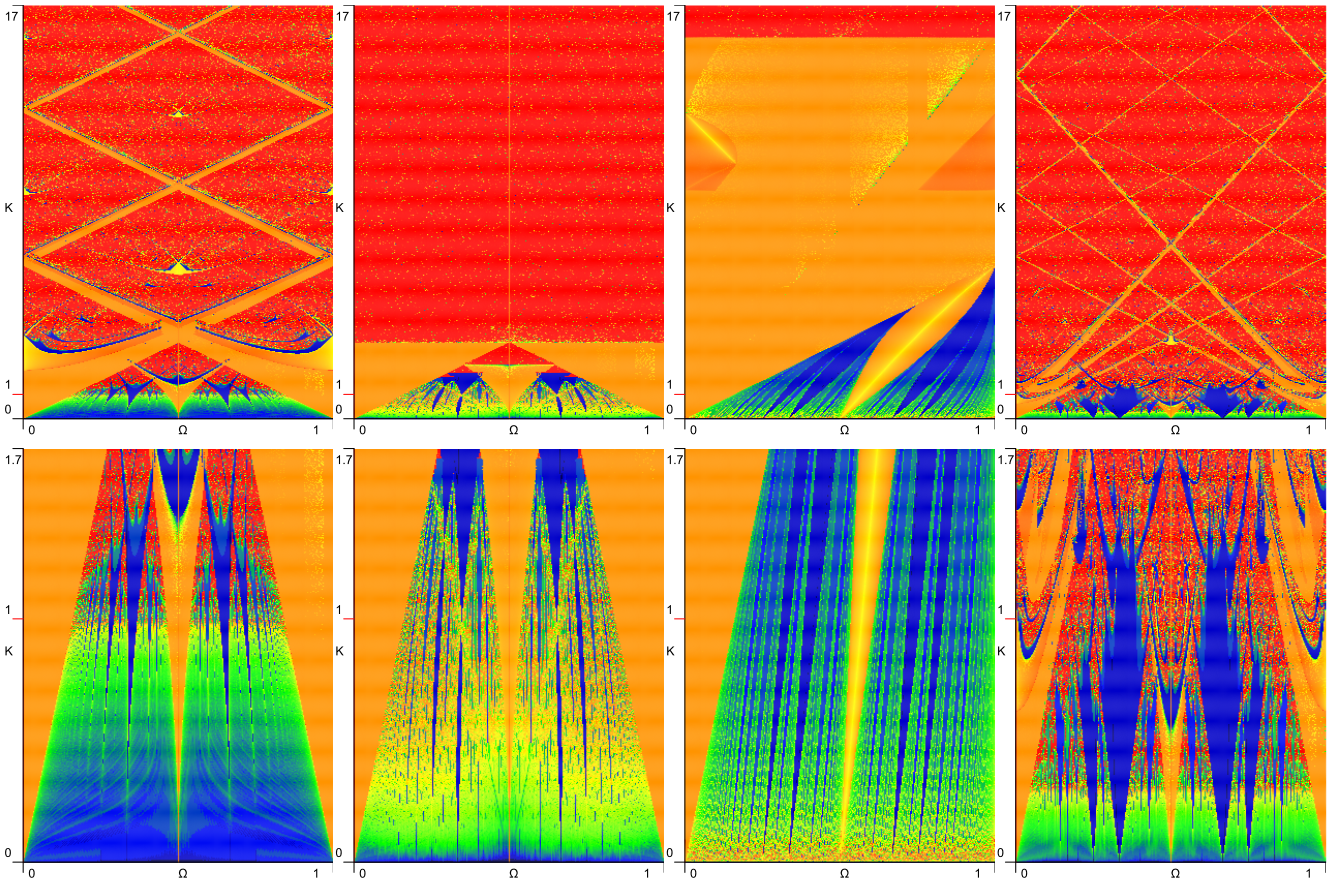
$$f(y_n) = \sin(2\pi y_n) \tag{7}$$

$$f(y_n) = \begin{cases} 4 \cdot y_n & \text{if } 0 \leq y_n < 1/4 \\ (1/4 - y_n) \cdot 4 + 1 & \text{if } 1/4 \leq y_n < 3/4 \\ (y_n - 3/4) \cdot 4 - 1 & \text{otherwise.} \end{cases} \tag{8}$$

$$f(y_n) = \begin{cases} \frac{y_n + T}{1 + 2 \cdot \epsilon \cdot T} & \text{if } 0 \leq y_n < B \\ \frac{y_n + (1 - 2 \cdot ep) \cdot T}{1 - 2 \cdot \epsilon \cdot T} & \text{if } b \leq y_n < 1 - T \\ \frac{y_n + T - 1}{1 + 2 \cdot \epsilon \cdot T} & \text{if } 1 - T \leq y_n \leq B + 1 \\ \frac{y_n + (1 - 2 \cdot \epsilon) \cdot T - 1}{1 - 2 * \epsilon \cdot T} & \text{otherwise.} \end{cases} \tag{9}$$

$$f(y_n) = \frac{1}{A} \sum_{m=1}^{4} a_m \sin(2\pi m y_n) \tag{10}$$

With $T = 0.5$, $\epsilon = 0.25$ and $B = 0.5 + (\epsilon - 1) \cdot T$ in (9) and with $a_m = \{1, \frac{1}{2^2}, \frac{1}{3^2}, \frac{1}{4^2}\}$ and $A = a1 + a2 + a3 + a4$ in (10). Equation (8) is a triangle function. Equation (9) is a piecewise linear function from the biomedical literature [20] and equation (10) is a Fourier-series composition with four terms. The functions are shown in Figure 5. The variation of nonlinear function is depicted in Figure 6 using the PeakSparsity feature. Obviously the variation of the nonlinear function has substantial large scale impact on the response, though Arnold tongue-like features are persistent in all cases. For the continuous piecewise linear (triangle) function, we observe a well-known effect of tongues pinching together with increase nonlinearity. This effect has been called "Sausages" [30].



(a) Sine (b) Triangle (c) Cardiorespiratory (d) Fourier

Figure 6: Variation of nonlinear perturbation function in the circle map using the PeakSparsity spectral value in the (k,Ω)-plane (top) mild nonlinearity with $k \in [0, 1.3\dot{3})$ (bottom) strong nonlinearity with $k \in [0, 13.\dot{3})$ (a) sine, (b) continuous rectilinear, (c) discontinuous rectilinear, (d) mixed three sine.

The asymmetric non-continuous piecewise linear case (cardiorespiratory model) shows that symmetry can be broken as well as that fixed point regions can be very substantially extended. Some of this behavior can be explained by understanding occurance of fixed points by the slope of the function against an intersection with the identity map $y_n = y_{n+1}$ depicted in Figure 5 as a red diagonal line. If the angle is shallow it creates fixed points. For the triangle case we observe no fixed points due to the choice of slope, while in the cardiorespiratory case we see an abundance of fixed points for the choice of slope and intersection with the identity.

Overall we note that while there are persistent structure between nonlinear function, the choice of nonlinearity plays a substantial role in the sound of a chaotic oscillator. This suggests that the study of their variation is an interesting subject of further investigation.

8. CONCLUSIONS

The study of chaotic oscillators is an extremely rich topic ripe for long-standing research. The proposed framework described in this paper offers a strategy to further our understanding of their behavior for sound synthesis.

The present framework is applicable to many more cases than the ones discussed in this paper. A comparative analysis of wave shaping, modulation techniques, feedback, and circle maps is forthcoming [31]. Yet a detailed study on a wide range of chaotic oscillators [10] would help crystallize criteria for picking algorithms and their respective similarities and differences.

The parameter space itself could be subjected to further analysis. For example, similarity measures [32] could be applied to related different points and regions in the parameter plane and hence provides a form of reparametrization that replaces parametric closeness with perceptual closeness.

Finally feature discovery and a deeper understanding of the perception of differing noisy sounds are key aspects that can help us get further understanding of nonlinear oscillators. We see the use of feature discovery by maximum discrimination as a promising strategy for detailed study of existing features [33] and also as a possible pathway towards understanding what aspects of the signal are poorly captured by the current feature landscape. Deepening our understanding of perception of noise is less straightforward and will require new perceptual experimentation with novel models ideas about what could constitute perceptually relevant mechanisms that discriminate difference versions of noise.

9. REFERENCES

[1] A. Di Scipio, "Composition by exploration of non-linear dynamic systems," in *Proceedings of the International Computer Music Conference*, 1990, pp. 324–327.

[2] B. Truax, "Chaotic non-linear systems and digital synthesis: an exploratory study," in *Proceedings of the International Computer Music Conference*, 1990, pp. 100–103.

[3] M. Gogins, "Iterated functions systems music," *Computer Music Journal*, vol. 15, no. 1, pp. 40–48, 1991.

[4] R. Bidlack, "Chaotic systems as simple (but complex) compositional algorithms," *Computer Music Journal*, vol. 16, no. 3, pp. 33–47, 1992.

[5] J. Mackenzie and M. Sandler, "Modelling sound with chaos," in *Proceedings of IEEE International Symposium on Circuits and Systems - ISCAS '94*, May 1994, pp. 93–96.

[6] R. Dobson and J. Fitch, "Experiments with chaotic oscillators," in *Proceedings of the International Computer Music Conference*, Banff, Canada, 1995, pp. 45–48.

[7] D. Slater, "Chaotic sound synthesis," *Computer Music Journal*, vol. 22, no. 2, pp. 12–19, 1998.

[8] G. Essl, "Circle maps as a simple oscillators for complex behavior: I. Basics," in *Proceedings of the International Computer Music Conference (ICMC)*, New Orleans, November 2006.

[9] R. Holopainen, *Self-organised sound with autonomous instruments: Aesthetics and experiments*, Ph.D. thesis, University of Oslo, 2012.

[10] E. Berdahl, E. Sheffield, A. Pfalz, and A. T. Marasco, "Widening the razor-thin edge of chaos into a musical highway: Connecting chaotic maps to digital waveguides," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, L. Dahl, D. Bowman, and T. Martin, Eds., Blacksburg, Virginia, USA, June 2018, pp. 390–393, Virginia Tech.

[11] G. Essl, "Circle maps as a simple oscillators for complex behavior: II. Experiments," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, Montreal, September 18-20 2006.

[12] E. R. Miranda and M. Wanderley, *New Digital Musical Instruments: Control And Interaction Beyond the Keyboard (Computer Music and Digital Audio Series)*, A-R Editions, Inc., Madison, WI, USA, 2006.

[13] S. P. Kuznetsov et al., "Two-parameter analysis of nonlinear systems. atlas of charts of dynamical regimes," `http://www.sgtnd.narod.ru/science/atlas/eng/`, Accessed: 2019-03-24.

[14] V. I. Arnold, "Small denominators, I: mappings of the circumference into itself," *Am. Math. Soc. Transl. (2)*, vol. 46, pp. 213–284, 1965.

[15] W. Lauterborn and E. Cramer, "Subharmonic route to chaos observed in acoustics," *Phys. Rev. Lett.*, vol. 47, pp. 1445–1448, Nov 1981.

[16] W. Lauterborn and E. Suchla, "Bifurcation superstructure in a model of acoustic turbulence," *Phys. Rev. Lett.*, vol. 53, pp. 2304–2307, Dec 1984.

[17] H. Mori, G.C. Paquette, and Y. Kuramoto, *Dissipative Structures and Chaos*, Springer Berlin Heidelberg, 2nd edition, 1998.

[18] J. M. T. Thompson and H. B. Stewart, *Nonlinear Dynamics and Chaos*, Wiley, 2nd edition, 2002.

[19] AP Kuznetsov, LV Turukina, AV Savin, IR Sataev, JV Sedova, and SV Milovanov, "Multi-parameter picture of transition to chaos," *Izvestija Vuzov. Applied Nonlinear Dynamics*, vol. 10, no. 3, pp. 80, 2002.

[20] M. McGuinness and Y. Hong, "Arnold Tongues in Human Cardiorespiratory Systems," *Chaos*, vol. 14, no. 1, pp. 1–6, 2004.

[21] Bai-Lin Hao and Wei-Mou Zheng, *Applied symbolic dynamics and chaos*, World scientific, 2nd edition, 2018.

[22] N. Y. Ivankov and S. Kuznetsov, "Complex periodic orbits, renormalization, and scaling for quasiperiodic golden-mean transition to chaos," *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol. 63, pp. 046210, 05 2001.

[23] V. S. Anishchenko, V. Astakhov, A. Neiman, T. Vadivasova, and L. Schimansky-Geier, *Nonlinear Dynamics of Chaotic and Stochastic Systems: Tutorial and Modern Developments (Springer Series in Synergetics)*, Springer-Verlag, Berlin, Heidelberg, second edition, 2007.

[24] L. Glass and J. Sun, "Periodic forcing of a limit-cycle oscillator: Fixed points, arnold tongues, and the global organization of bifurcations," *Physical Review E*, vol. 50, no. 6, pp. 5077, 1994.

[25] O. Santala, *Perception and auditory modeling of spatially complex sound scenarios*, Ph.D. thesis, Aalto University, 2015.

[26] F. Alías, J. Socoró, and X. Sevillano, "A review of physical and perceptual feature extraction techniques for speech, music and environmental sounds," *Applied Sciences*, vol. 6, no. 5, pp. 143, May 2016.

[27] F. Alías and J. C. Socoró, "Description of anomalous noise events for reliable dynamic traffic noise mapping in real-life urban and suburban soundscapes," *Applied Sciences*, vol. 7, no. 2, pp. 146, 2017.

[28] X. Sevillano, J. C. Socoró, F. Alías, et al., "Dynamap–development of low cost sensors networks for real time noise mapping," *Noise Mapping*, vol. 3, no. 1, 2016.

[29] B.C.J. Moore, *Introduction to the Psychology of Hearing*, Academic Press, 4th ed. edition, 1995.

[30] W.-M. Yang and Hao B.-L., "How the arnold tongues become sausages in a piecewise linear circle map," *Communications in Theoretical Physics*, vol. 8, no. 1, pp. 1–15, jul 1987.

[31] G. Essl, "Connecting Circle Maps, Waveshaping, and Phase Modulation via Iterative Phase Functions and Projections," in *Proceedings of the 14th International Symposium on Computer Music Multidisciplinary Research (CMMR)*, 2019.

[32] J.-J. Aucouturier, F. Pachet, et al., "Music similarity measures: What's the use?," in *Proceedings of ISMIR*, 2002, pp. 13–17.

[33] T. Jebara and T. Jaakkola, "Feature selection and dualities in maximum entropy discrimination," in *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 2000, pp. 291–300.

LARGE-SCALE REAL-TIME MODULAR PHYSICAL MODELING SOUND SYNTHESIS

Stefan Bilbao
Acoustics and Audio Group
University of Edinburgh, Edinburgh, UK
sbilbao@ed.ac.uk

Michele Ducceschi *
Acoustics and Audio Group
University of Edinburgh, Edinburgh, UK
michele.ducceschi@ed.ac.uk

Craig J. Webb
Physical Audio Ltd.,
London, UK
craig@physicalaudio.co.uk

ABSTRACT

Due to recent increases in computational power, physical modeling synthesis is now possible in real time even for relatively complex models. We present here a modular physical modeling instrument design, intended as a construction framework for string- and bar-based instruments, alongside a mechanical network allowing for arbitrary nonlinear interconnection. When multiple nonlinearities are present in a feedback setting, there are two major concerns. One is ensuring numerical stability, which can be approached using an energy-based framework. The other is coping with the computational cost associated with nonlinear solvers—standard iterative methods, such as Newton-Raphson, quickly become a computational bottleneck. Here, such iterative methods are sidestepped using an alternative energy conserving method, allowing for great reduction in computational expense or, alternatively, to real-time performance for very large-scale nonlinear physical modeling synthesis. Simulation and benchmarking results are presented.

1. INTRODUCTION

One goal of physical modeling synthesis is the emulation of existing musical instruments, perhaps with extensions in terms of design and functionality not easily realisable in the real world. Another is the design of new instruments, without a real-world referent. The hope is that through adherence to the laws of physics, synthetic sound of a natural acoustic character can be produced. In this setting, modularity is an important concept—the idea is to give the user control over instrument design through the interconnection of semi-independent modules. In this contribution, a modular network, consisting of a set of string- or bar-like elements coupled through an auxiliary nonlinear connection network is presented.

Modularity in physical modeling synthesis is not new. The earliest attempt at a complete synthesis system, due to Cadoz [1], had such a modularity principle explicitly built in—the basic elements were masses and springs, through the interconnection of which more elaborate instruments could be constructed. The MOSAIC (later Modalys) system [2] uses the same notion, now employing modal synthesis [3]. Modular networks have also been proposed using wave-based methods [4] and time-stepping schemes—see, e.g., [5, 6]. The main difficulty is in simulating a network connected in a feedback configuration—a problem compounded when nonlinearities, essential for any musically-interesting sound output, are present. The complications are similar to those which occur in virtual analog modeling, but now in a mechanical setting, where individual elements have a distributed character.

Two main technical concerns emerge: one is ensuring numerical stability, as numerical methods for nonlinear systems are prone to explosive unstable solution growth. Another is computational efficiency; iterative methods (such as, e.g., Newton-Raphson [7]) commonly used in nonlinear ODE/PDE solvers, can lead to large increases in computational cost. For the problem of numerical stability, energy techniques are probably the only known method. Such methods, in different guises, have been popular in virtual analog applications (port-Hamiltonian approaches [8], wave digital filtering [9, 10], and used in PDE solvers [11]). For general nonlinearities, however, including those involving collision or intermittent contact, energy methods are available, but usually require the use of iterative methods. See, e.g., [12, 13, 14].

Recently, a new class of methods mas been proposed in the context of virtual analog modeling, relying on energy quadratisation [15], leading to numerical methods which are resolvable without recourse to iterative methods, and which maintain the notion of an energy balance leading to a numerical stability guarantee—see [16, 17], as well as [18] in the context of audio systems. Such an approach reduces computational costs by as much as an order of magnitude, and allows for the simulation of relatively complex nonlinear systems in real time on standard hardware.

In Section 2, a modular physical modeling synthesis system is presented, consisting of an interconnection of string- or bar-like primitives, and with nonlinear connections which allow intermittent contact. A numerical discretisation scheme is presented in Section 3, allowing for energy-stable non-iterative simulation. Simulation results, illustrating characteristic behaviour of such a modular synthesis system appear in Section 4. Performance results for a C implementation are presented in Section 5, demonstrating the possibility of real-time operation of relatively complex instrument designs. Some perspectives appear in Section 6.

2. A MODULAR INSTRUMENT MODEL

The canonical building block here is the linear bar or string (referred to henceforth here as a stiff string), of circular cross section, assumed to vibrate transversely in a single polarisation. The basic equation of motion (see, e.g., [11], as well as other closely-related forms [19, 20]), under unforced conditions, is of the form

$$\mathcal{L}u = 0 \tag{1}$$

where the linear partial differential operator $\mathcal{L}$ is defined as

$$\mathcal{L} = \rho A \partial_t^2 - T\partial_x^2 + EI\partial_x^4 + 2\rho A\sigma_0\partial_t - 2\rho A\sigma_1\partial_t\partial_x^2 \tag{2}$$

Here, $u(x,t)$ is the transverse displacement of the stiff string, in m, as a function of time $t \geq 0$, in s, and spatial coordinate $x \in$

* M. Ducceschi was supported by an Early Career Fellowship from the Leverhulme Trust.

$\mathcal{D} = [0,\ L]$, for some string length L in m. ∂_t and ∂_x represent partial differentiation with respect to t and x, respectively.

The various parameters which define the stiff string are the material density ρ, in kg$\cdot$ m^{-3}, Young's modulus E in Pa, tension T in N, and string radius r in m, from which follow the cross-sectional area $A = \pi r^2$ and moment of inertia $I = \frac{1}{4}\pi r^4$. σ_0 and σ_1 are loss parameters, allowing for a simple frequency-dependent loss characteristic (though much more realistic loss models are available [21]). The symbol $\boldsymbol{\gamma}$ is used here to represent the set of parameters defining a particular string; its minimal form is:

$$\boldsymbol{\gamma} = \{\rho, E, r, T, \sigma_0, \sigma_1, L\} \tag{3}$$

Two boundary conditions must be supplied at each end of the string; many choices are possible (see [11]), but in this work, simply supported conditions will be enforced, so that

$$u = \partial_x^2 u = 0 \qquad \text{at} \qquad x = 0, L \tag{4}$$

The equation (1) requires two initial conditions: $u(x, 0)$ and $\partial_t u(x, 0)$. In a synthesis setting, initial conditions are usually set to zero, but will be maintained here in order to examine energetic behaviour under unforced conditions.

System (1) satisfies an energy balance. Multiplying (1) by $\partial_t u$, and integrating over $\mathcal{D}$ gives

$$\int_{\mathcal{D}} \partial_t u \mathcal{L} u \, dx = 0 \tag{5}$$

Defining the L_2 norm of a function $f(x)$ over the domain $\mathcal{D}$ as

$$\|f\|_{\mathcal{D}} = \sqrt{\int_{\mathcal{D}} f^2 dx} \tag{6}$$

and using integration by parts, as well as the boundary conditions (4) leads to the energy balance

$$d\mathcal{H}_s/dt + \mathcal{Q}_s = 0 \tag{7}$$

where

$$\mathcal{H}_s = \frac{\rho A}{2}\|\partial_t u\|_{\mathcal{D}}^2 + \frac{T}{2}\|\partial_x u\|_{\mathcal{D}}^2 + \frac{EI}{2}\|\partial_x^2 u\|_{\mathcal{D}}^2 \tag{8a}$$

$$\mathcal{Q}_s = 2\rho A \sigma_0 \|\partial_t u\|_{\mathcal{D}}^2 + 2\rho A \sigma_1 \|\partial_x \partial_t u\|_{\mathcal{D}}^2 \tag{8b}$$

$\mathcal{H}_s$ is the total stored energy for the string, and $\mathcal{Q}_s$ is the power loss. Given that $\mathcal{H}_s \geq 0$ and $\mathcal{Q}_s \geq 0$, under unforced conditions, this implies that $d\mathcal{H}_s/dt \leq 0$, and thus energy is monotonically non-increasing, and may be used to bound the growth of the state u itself—such a balance may be employed in discrete time to arrive at energy-based numerical stability conditions. See Section 3.

2.1. Excitation

For system (1), an excitation may be added as

$$\mathcal{L} u = \delta(x - x_e) f_e(t) \tag{9}$$

where here, $f_e(t)$ is an externally-supplied force signal (in N), and $\delta(x - x_e)$, is a spatial Dirac delta function selecting the excitation location $x_e \in [0,\ L]$. The pointwise character of the excitation can be easily extended to the case of distributed contact (as in, e.g., the case of a piano hammer [19] or bow [22]). Simple choices of the excitation function, if intended to model a strike or pluck, are raised cosine distributions of the form

$$f_e(t) = \begin{cases} \frac{1}{2} f_{e,max} \left(1 - \cos\left(\frac{q\pi(t - t_e)}{T_e}\right)\right), & t_e \leq t \leq t_e + T_e \\ 0, & \text{otherwise} \end{cases} . \tag{10}$$

where t_e is the starting time of the excitation, T_e is the duration, $f_{e,max}$ is the maximum force, in N, and where $q = 1$ for a pluck and $q = 2$ for a strike. See Figure 1.

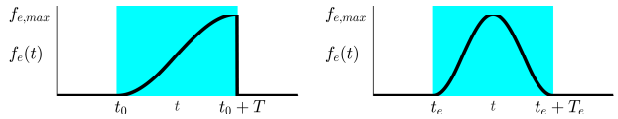


Figure 1: Excitation functions $f_e(t)$ for a pluck (left) and strike (right), as per the form given in (10).

From an energetic standpoint, one may again multiply (9) by $\partial_t u$, and then integrate over $\mathcal{D}$ to obtain the energy balance:

$$\frac{d}{dt}\mathcal{H}_s + \mathcal{Q}_s = \underbrace{\partial_t u(x_e) f_e}_{\triangleq \mathcal{P}} \tag{11}$$

where $\mathcal{H}_s$ and $\mathcal{Q}_s$ are as in (8), and where $\mathcal{P}$ is the instantaneous input power due to the excitation. Thus the rate of growth of energy in the system may be bounded in terms of supplied power.

2.2. Connection to a Lumped Object

Consider now a coupled system of a string under excitation, and in pointwise contact, at location $x = x_c$, with a lumped object of mass M which is constrained to travel parallel to the plane of polarisation of the string, and with displacement $w = w(t)$:

$$\mathcal{L} u = -\delta(x - x_c) f_c + \delta(x - x_e) f_e \qquad M \frac{d^2 w}{dt^2} = f_c \tag{12}$$

Here, $f_c = f_c(\eta)$ is the connection force, assumed dependent on

$$\eta = u(x_c, t) - w(t) \tag{13}$$

the relative displacement between the lumped object and the string at the connection point. See Figure 2.

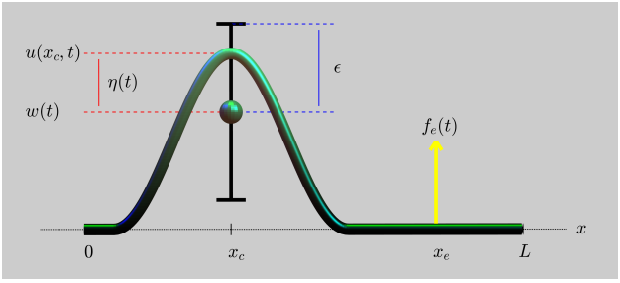


Figure 2: Stiff string, under an excitation force and coupled to a lumped object, as per (12).

Many choices for the interaction force f_c are possible. A basic linear/cubic restoring force served as a basis for the modular network presented in [5]. A more general choice, allowing for intermittent loss of contact (or collisions), and considerably widening

the range of possible sound output, is the following choice:

$$f_c(\eta) = K[|\eta| - \epsilon]_+^\alpha \operatorname{sgn}(\eta) \tag{14}$$

Such a force law is parameterised by $K \geq 0$, a stiffness constant, $\alpha \geq 1$, a nonlinearity exponent inspired by models in contact dynamics [23, 19], and $\epsilon \geq 0$, an effective length of the lumped object, which is now able to rattle, and which allows for a dead zone where no force is exerted by the lumped object on the string and vice versa. The notation $[\cdot]_+$ indicates the "positive part of." See Figure 3. Note that when $\epsilon = 0$, so that the lumped mass and string are always in contact, the force (14) reduces to a linear connection when $\alpha = 1$, and a cubic nonlinear force when $\alpha = 3$. It is also possible to go further here, and introduce a model of contact loss, as per similar models in contact dynamics (see, e.g., [23]), and employed in musical instrument modeling [13].

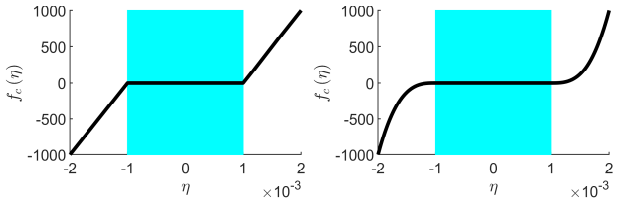


Figure 3: Nonlinear force characteristics of the form given in (14). Left: with $K = 10^6$, $\alpha = 1$ and $\epsilon = 0.001$, and right: with $K = 10^{12}$, $\alpha = 3$ and $\epsilon = 0.001$. The dead zone, over which there is no connection force, is shown as a blue shaded region.

At this point, we introduce the potential $\phi(\eta)$, such that

$$\phi(\eta) \geq 0 \qquad f_c = d\phi/d\eta \tag{15}$$

and note that, through the chain rule,

$$d\phi/dt = f_c d\eta/dt \tag{16}$$

Given the non-negativity of ϕ, it is also possible to define ψ as

$$\psi = \sqrt{2\phi} \qquad f_c = \psi d\psi/d\eta \tag{17}$$

This quadratisation of the potential energy has been employed in the context of virtual analog modeling and musical acoustics in [17, 24, 16], and also, recently, in much more general numerical settings, where it is referred to as *invariant energy quadratisation*—see, e.g., [15]. In the case of the force characteristic given in (14), the corresponding forms of ϕ and ψ are

$$\phi(\eta) = \frac{K}{\alpha+1}[|\eta| - \epsilon]_+^{\alpha+1} \quad \psi(\eta) = \sqrt{\frac{2K}{\alpha+1}}[|\eta| - \epsilon]_+^{\frac{\alpha+1}{2}} \tag{18}$$

Though the forms in (15) and (17) are equivalent, in the numerical context, they lead to distinct energy-conserving methods—in particular, the form in (15) leads to a form requiring iterative solution, and that in (17) to one which may be resolved explicitly. For a direct comparison of this distinction in a simplified case, see the companion paper [18].

From an energetic standpoint, multiplying the first of (12) by $\partial_t u$ and integrating over $\mathcal{D}$ yields

$$d\mathcal{H}_s/dt + \mathcal{Q}_s = \mathcal{P} - f_c \partial_t u(x_c) \tag{19}$$

Multiplying the second by dw/dt gives

$$\frac{d}{dt}\left(\frac{M}{2}\left(\frac{dw}{dt}\right)^2\right) = \frac{dw}{dt} f_c \tag{20}$$

Adding (19) and (20) gives, then

$$\frac{d}{dt}\mathcal{H}_s + \mathcal{Q}_s + \frac{d}{dt}\left(\frac{M}{2}\left(\frac{dw}{dt}\right)^2\right) + f_c\left(\partial_t u(x_c) - \frac{dw}{dt}\right) = \mathcal{P} \tag{21}$$

But, using

$$d\eta/dt = \partial_t u(x_c) - dw/dt \tag{22}$$

as well as (16) leads to the energy balance

$$d\mathcal{H}/dt + \mathcal{Q}_s = \mathcal{P} \tag{23}$$

where

$$\mathcal{H} = \mathcal{H}_s + \mathcal{H}_c \geq 0 \qquad \mathcal{H}_c = \frac{M}{2}\left(\frac{dw}{dt}\right)^2 + \phi \geq 0 \tag{24}$$

Thus the total energy $\mathcal{H}$ of the system may be decomposed into the energy of the stiff string $\mathcal{H}_s$ and that of the connection mechanism $\mathcal{H}_c$, and as before is non-negative—its rate of growth may again be bounded in terms of the input power $\mathcal{P}$. Such a connection mechanism is passive (and furthermore lossless).

2.3. A Complete Modular Network

The step to constructing an arbitrary network is relatively straightforward, using the model above as a starting point.

Suppose a network is defined with N_s stiff string elements, each characterised by a parameter set $\boldsymbol{\gamma}^{(q)}$, $q = 1, \ldots, N_s$, of the form given in (3). For each element, the associated transverse displacement is $u^{(q)} = u^{(q)}\left(x^{(q)}, t\right)$, for $x^{(q)} \in \mathcal{D}^{(q)} = [0,\ L^{(q)}]$, and the associated partial differential operator is $\mathcal{L}^{(q)}$. Suppose also that there are N_m lumped objects, of mass $M^{(j)}$ kg, and of displacement $w^{(j)}(t)$, $j = 1, \ldots, N_m$. See Figure 4.

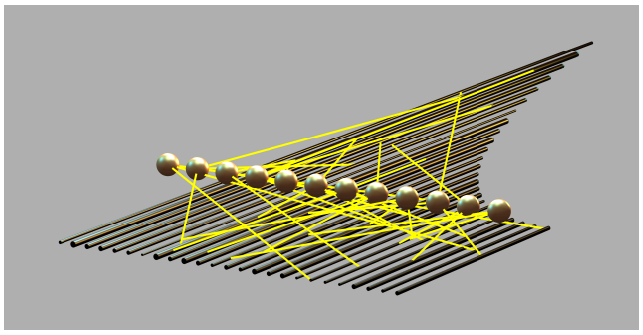


Figure 4: Modular network of stiff strings and lumped masses, with connections indicated as yellow lines.

Now suppose that there are N_c individual connections, each associated with a force $f_c^{(\nu)}$, $\nu = 1, \ldots, N_c$. The νth connection links the q_νth stiff string, at location $x^{(q_\nu)} = x_c^{(\nu)}$ with the j_νth lumped object, where $q_\nu \in \{1, \ldots, N_s\}$ and $j_\nu \in \{1, \ldots, N_m\}$. The force $f_c^{(\nu)}$ will be of the form

$$f_c^{(\nu)} = f_c^{(\nu)}\left(\eta^{(\nu)}\right) \quad \eta^{(\nu)} = u^{(q_\nu)}\left(x_c^{(\nu)}\right) - w^{(j_\nu)} \tag{25}$$

The force $f_c^{(\nu)}\left(\eta^{(\nu)}\right)$ is assumed to be related to an underlying potential energy $\phi^{(\nu)}\left(\eta^{(\nu)}\right) \geq 0$ as

$$f_c^{(\nu)} = d\phi^{(\nu)}/d\eta^{(\nu)} \tag{26}$$

In this case, $\phi^{(\nu)}$ will be chosen to be of the form (18), but any non-negative form is possible. Furthermore, define the index sets

$$\mathbb{I}_c^{(q)} = \{\nu \in \{1,\ldots,N_c\} \,|\, q_\nu = q\} \tag{27a}$$

$$\mathbb{I}_c^{(j)} = \{\nu \in \{1,\ldots,N_c\} \,|\, j_\nu = j\} \tag{27b}$$

Finally, assume N_e excitation functions $f_e^{(\xi)}$, $\xi = 1,\ldots,N_e$, acting on the stiff strings with index q_ξ where $q_\xi \in \{1,\ldots,N_s\}$ at locations $x_e^{(\xi)}$. Define also the index set

$$\mathbb{I}_e^{(q)} = \{\xi \in \{1,\ldots,N_e\} \,|\, q_\xi = q\} \tag{28}$$

The excitation functions $f_e^{(\xi)}$ could be chosen as in (10).

The complete system is:

$$\mathcal{L}^{(q)}u^{(q)} = -\sum_{\nu\in\mathbb{I}_c^{(q)}} f_c^{(\nu)}\delta\left(x^{(q)} - x_c^{(\nu)}\right) + \sum_{\xi\in\mathbb{I}_e^{(q)}} f_e^{(\xi)}\delta\left(x^{(q)} - x_e^{(\xi)}\right) \tag{29a}$$

$$M^{(j)}\frac{d^2w^{(j)}}{dt^2} = \sum_{\nu\in\mathbb{I}_c^{(j)}} f_c^{(\nu)} \tag{29b}$$

Energy analysis is an extension of the case of the single string. Multiplying (29a) by $\partial_t u^{(q)}$, and integrating over $\mathcal{D}^{(q)}$ gives

$$\frac{d}{dt}\mathcal{H}_s^{(q)} + \mathcal{Q}_s^{(q)} + \sum_{\nu\in\mathbb{I}_c^{(q)}} f_c^{(\nu)}\partial_t u^{(q)}\left(x_c^{(\nu)}\right) = \mathcal{P}^{(q)} \tag{30}$$

where $\mathcal{H}_s^{(q)}$ and $\mathcal{Q}_s^{(q)}$ are stored energy and power loss for stiff string q, as defined in (8), with parameter set $\boldsymbol{\gamma}^{(q)}$, and where $\mathcal{P}^{(q)}$ is of the form given in (21). Summing over all strings gives

$$\frac{d}{dt}\mathcal{H}_s + \mathcal{Q}_s + \sum_{q=1}^{N_s}\sum_{\nu\in\mathbb{I}_c^{(q)}} f_c^{(\nu)}\partial_t u^{(q)}\left(x_c^{(\nu)}\right) = \mathcal{P} \tag{31}$$

where

$$\mathcal{H}_s = \sum_{q=1}^{N_s}\mathcal{H}_s^{(q)} \quad \mathcal{Q}_s = \sum_{q=1}^{N_s}\mathcal{Q}_s^{(q)} \quad \mathcal{P} = \sum_{q=1}^{N_s}\mathcal{P}^{(q)} \tag{32}$$

are the total stored energy, dissipated power and supplied power for the set of stiff strings. Recognising that the sets $\mathbb{I}_c^{(q)}$ form a partition of the set $\{1,\ldots,N_c\}$, one has, simply,

$$\frac{d}{dt}\mathcal{H}_s + \mathcal{Q}_s + \sum_{\nu=1}^{N_c} f_c^{(\nu)}\partial_t u^{(q_\nu)}\left(x_c^{(\nu)}\right) = \mathcal{P} \tag{33}$$

Similarly, multiplying (29b) by $dw^{(j)}/dt$, and summing over $j \in \{1,\ldots,N_m\}$, again using the partitioning property of $\mathbb{I}_c^{(j)}$, gives

$$\frac{d}{dt}\sum_{j=1}^{N_m}\left(\frac{M^{(j)}}{2}\left(\frac{dw^{(j)}}{dt}\right)^2\right) - \sum_{\nu=1}^{N_c} f_c^{(\nu)}\frac{d}{dt}w^{(j_\nu)} = 0 \tag{34}$$

Adding (33) and (34) gives, using (25) and (26),

$$d\mathcal{H}/dt + \mathcal{Q}_s = \mathcal{P} \tag{35}$$

where

$$\mathcal{H} = \mathcal{H}_s + \mathcal{H}_c \geq 0 \qquad \mathcal{H}_c = \sum_{j=1}^{N_m}\frac{M}{2}\left(\frac{dw^{(j)}}{dt}\right)^2 + \sum_{\nu=1}^{N_c}\phi^{(\nu)} \geq 0 \tag{36}$$

3. DISCRETE-TIME SIMULATION

Finite difference schemes for the stiff string are covered in detail in [11], and will be reviewed only briefly here.

Assume first a sample rate f_s (and associated time step $k = 1/f_s$), to be employed uniformly across all components in the network. Consider now a single stiff string as defined by (1). The grid function u_l^n represents an approximation to $u(x,t)$ at $t = nk$ and $x = lh$, for integer n and l, and where h is the grid spacing. In particular, $0 \leq l \leq N$, where $h = L/N$, for integer N. The discrete domains d, $\bar{d}$ and $\underline{d}$ are defined as

$$d = \{0,\ldots,N\} \quad \bar{d} = \{0,\ldots,N-1\} \quad \underline{d} = \{1,\ldots,N-1\} \tag{37}$$

A discrete inner product and norm may be defined over a discrete domain $b \subset \mathbb{Z}$ with grid spacing h (such as d, $\bar{d}$ or $\underline{d}$ in (37) above), for grid functions f_l^n, g_l^n, as

$$\langle f^n, g^n\rangle_b = \sum_{l\in b} h f_l^n g_l^n \qquad \|f^n\|_b = \sqrt{\langle f^n, f^n\rangle_b} \tag{38}$$

3.1. Shift and Difference Operators

The forward and backward time-shift operators e_{t+} and e_{t-} may be defined, with regard to the grid function u_l^n, as

$$e_{t+}u_l^n = u_l^{n+1} \qquad e_{t-}u_l^n = u_l^{n-1} \tag{39}$$

Forwards, backwards and centered approximations to a first time derivative may be defined in terms of shifts as

$$\delta_{t+} = \frac{e_{t+}-1}{k} \quad \delta_{t-} = \frac{1-e_{t-}}{k} \quad \delta_{t\circ} = \frac{e_{t+}-e_{t-}}{2k} \tag{40}$$

and averaging operators μ_{t+} and μ_{t-} and an approximation δ_{tt} to a second derivative may be written as

$$\mu_{t+} = \frac{e_{t+}+1}{2} \qquad \mu_{t-} = \frac{1+e_{t-}}{2} \qquad \delta_{tt} = \delta_{t+}\delta_{t-} \tag{41}$$

Similarly, spatial shifts e_{x+} and e_{x-} may be defined as

$$e_{x+}u_l^n = u_{l+1}^n \qquad e_{x-}u_l^n = u_{l-1}^n \tag{42}$$

and approximations to the first and second spatial deriavtaives as

$$\delta_{x+} = \frac{e_{x+}-1}{h} \quad \delta_{x-} = \frac{1-e_{x-}}{h} \quad \delta_{xx} = \delta_{x+}\delta_{x-} \tag{43}$$

3.2. Basic Scheme

A basic explicit finite difference scheme for an uncoupled linear stiff string, as in (1) may be written, for $u = u_l^n$, $l \in d$ as

$$\underbrace{\left(\rho A \delta_{tt} - T\delta_{xx} + EI\delta_{xx}^2 + 2\rho A\sigma_0 \delta_{t\circ} - 2\rho A\sigma_1 \delta_{t-}\delta_{xx}\right)}_{\ell} u = 0 \tag{44}$$

Numerical boundary conditions corresponding to (4) are

$$u_l^n = \delta_{xx} u_l^n = 0 \qquad l = 0, N \tag{45}$$

and allow (44) to be well-defined even when applied at or adjacent to the domain end points.

The scheme (44) satisfies an energy balance of the form

$$\delta_{t+} \mathfrak{h}_s^{n-1/2} + \mathfrak{q}_s^n = 0 \tag{46}$$

where $\mathfrak{h}_s^{n-1/2}$ and $\mathfrak{q}_s^n$ are the discrete-time energy storage function and power loss, respectively, defined as

$$\begin{aligned} \mathfrak{h}_s^{n-\frac{1}{2}} &= \frac{\rho A}{2}\|\delta_{t-} u^n\|_d^2 + \frac{T}{2}\langle \delta_{x+} u^n, e_{t-}\delta_{x+} u^n\rangle_{\bar{d}} \qquad (47a) \\ &\quad + \frac{EI}{2}\langle \delta_{xx} u^n, e_{t-}\delta_{xx} u^n\rangle_{\underline{\bar{d}}} - \rho A\sigma_1 k\|\delta_{t-}\delta_{x+} u\|_{\bar{d}}^2 \\ \mathfrak{q}_s^n &= 2\rho A\sigma_0\|\delta_{t\circ} u\|_d^2 + 2\rho A\sigma_1\|\delta_{t\circ}\delta_{x+} u\|_{\bar{d}}^2 \qquad (47b) \end{aligned}$$

The power loss $\mathfrak{q}_s^n$ is non-negative, but the stored energy $\mathfrak{h}_s$ is only non-negative under the condition

$$h^2 \geq \frac{1}{2}\left(\frac{Tk^2}{\rho A} + 4\sigma_1 k + \sqrt{\left(\frac{Tk^2}{\rho A} + 4\sigma_1 k\right)^2 + 16\frac{EIk^2}{\rho A}}\right) \tag{48}$$

which is a numerical stability condition for scheme (44) in isolation. Under this condition, the state size may be bounded in terms of supplied energy.

In implementation, scheme (44) is explicit—if the state u_l^n is written as a column vector $\mathbf{u}^n = [u_0^n, \ldots, u_N^n]^T$, (44) becomes

$$\mathbf{u}^{n+1} = \mathbf{B}\mathbf{u}^n + \mathbf{C}\mathbf{u}^{n-1} \tag{49}$$

where $\mathbf{B}$ and $\mathbf{C}$ are sparse $(N+1) \times (N+1)$ matrices. See [11] for the construction of these matrices in terms of the various spatial difference operators.

3.3. String with Nonlinear Connection and Excitation

Consider now the simple case of a stiff string under an excitation force and a nonlinear connection, as per (12). A discrete form is

$$\ell u = -f_c \frac{1}{h}\mathcal{I}(x_c) + f_e \frac{1}{h}\mathcal{I}(x_e) \qquad M\delta_{tt} w = f_c \tag{50}$$

Here, $f_c = f_c^n$ is a time series representing an approximation to $f_c(t)$, of a nature to be described shortly, and $f_e = f_e^n$ is an approximation to the externally-supplied function $f_e(t)$, perhaps obtained through sampling. The grid functions $\mathcal{I}_l(x_0)$, selects a given location x_0. Many choices are possible, but the simplest, and that which will be employed subsequently here, is certainly

$$\mathcal{I}_l(x_0) = \begin{cases} 1, & l = l_0 = \text{round}(x_0/h) \\ 0, & \text{otherwise} \end{cases} \tag{51}$$

When multiplied by $1/h$, as in (50) above, approximations to the Dirac delta functions from (12) result, where $x_0 = x_c, x_e$.

The force f_c^n will be dependent on the relative displacement η of the string and lumped mass at the connection grid index $l = l_c$:

$$\eta^n = u_{l_c}^n - w^n \tag{52}$$

Consider now the inner product of the first of (50) with $\delta_{t\circ} u$ over d—this leads to the energy balance

$$\delta_{t+}\mathfrak{h}_s^{n-1/2} + \mathfrak{q}_s^n + f_c^n \delta_{t\circ} u_{l_c}^n = \underbrace{f_e^n \delta_{t\circ} u_{l_e}^n}_{\mathfrak{p}^n} \tag{53}$$

where $\mathfrak{h}_s^{n-1/2}$ and $\mathfrak{q}_s^n$ are as defined in (47), and where $\mathfrak{p}^n$ is the discrete input power due to the excitation.

Similarly, multiplying the second of (50) by $\delta_{t\circ} w^n$ leads to

$$\delta_{t+}\left(\frac{M}{2}(\delta_{t-} w^n)^2\right) - f_c \delta_{t\circ} w^n = 0 \tag{54}$$

Adding (53) and (54), and using (52) leads to the balance

$$\delta_{t+}\left(\mathfrak{h}_s^{n-1/2} + \frac{M}{2}(\delta_{t-} w^n)^2\right) + \mathfrak{q}_s^n + f_c^n \delta_{t\circ}\eta^n = \mathfrak{p}^n \tag{55}$$

What is lacking is a definition of f_c^n. Here are two:

$$f_c^n = \frac{\delta_{t+}\phi^{n-1/2}}{\delta_{t\circ}\eta} \qquad f_c^n = \mu_{t+}\psi^{n-1/2}\frac{\delta_{t+}\psi^{n-1/2}}{\delta_{t\circ}\eta^n} \tag{56}$$

Here, $\phi^{n-1/2}$ is an approximation to ϕ at time $t = (n-1/2)k$, such as $\phi^{n-1/2} = \mu_{t-}\phi(\eta^n) \geq 0$, and $\psi^{n-1/2}$ is an approximation to ψ at time $t = (n-1/2)k$. When inserted in (55) above, either form in (56) leads to the energy balance

$$\delta_{t+}\mathfrak{h}^{n-1/2} + \mathfrak{q}_s^n = \mathfrak{p}^n \qquad \mathfrak{h}^{n-1/2} = \mathfrak{h}_s^{n-1/2} + \mathfrak{h}_c^{n-1/2} \tag{57}$$

where, for the two choices of the force f_c^n from (56),

$$\mathfrak{h}_c^{n-1/2} = \frac{M}{2}(\delta_{t-} w^n)^2 + \begin{cases} \phi^{n-1/2} \\ \frac{1}{2}\left(\psi^{n-1/2}\right)^2 \end{cases} \geq 0 \tag{58}$$

Under the further condition (48), then $\mathfrak{h}_s \geq 0$ implying that $\mathfrak{h} \geq 0$, and as before, the system as a whole will be numerically stable.

3.4. Implementation: Iterative vs. Non-iterative Methods

When written in vector/matrix form, the scheme (50) has the form

$$\begin{aligned} \mathbf{u}^{n+1} &= \mathbf{B}\mathbf{u}^n + \mathbf{C}\mathbf{u}^{n+1} - \theta \mathbf{j}_c f_c^n + \theta \mathbf{j}_e f_c^n \qquad (59a) \\ w^{n+1} &= 2w^n - w^{n-1} + \frac{k^2}{M} f_c^n \qquad (59b) \end{aligned}$$

where $\theta = k^2/(\rho A h(1+\sigma_0 k))$, and where $\mathbf{j}_e$ and $\mathbf{j}_c$ are $N+1$ element column vectors, all zero except for a 1 in locations l_e and l_c respectively.

Now, using the fact that

$$\delta_{t\circ}\eta^n = \frac{1}{2k}\left(\mathbf{j}_c^T\left(\mathbf{u}^{n+1} - \mathbf{u}^{n-1}\right) - w^{n+1} + w^{n-1}\right) \tag{60}$$

and the updates (59) leads to the affine relationship between $\delta_{t\circ}\eta^n$ and f_c^n:

$$\delta_{t\circ}\eta^n + \zeta f_c^n + \Xi^n = 0 \tag{61}$$

where Ξ^n consists of previously computed (known) values of $\mathbf{u}$ and samples of the excitation f_e^n, and ζ is a constant.

Consider now the first energy-conserving definition of f_c^n, from (56). In combination with (61), this leads to the nonlinear equation:

$$\underbrace{r + 2k\zeta \frac{\phi\left(r + \eta^{n-1}\right) - \phi\left(\eta^{n-1}\right)}{r}}_{F(r)} + 2k\Xi^n = 0 \tag{62}$$

where $r = \eta^{n+1} - \eta^{n-1}$ is the unknown. Such a form has appeared in earlier works regarding energy-conserving collision simulation [12, 13], and requires, in general, an iterative solution through, e.g., Newton-Raphson; the number of iterations depends strongly on the particular parameters defining the nonlinearity.

The second form of f_c given in (56) leads to a distinct update. Define

$$g^n = \frac{\delta_{t+}\psi^{n-1/2}}{\delta_{t\circ}\eta^n} \triangleq \frac{d\psi}{d\eta}\,|_{t=nk} \tag{63}$$

The time series g^n can be computed directly from known values of η^n and the functional form of ψ. Given g^n, the following pair of updates results from (61) and (63):

$$\delta_{t\circ}\eta^n + \zeta g^n \mu_{t+}\psi^{n-1/2} + \Xi^n = 0 \qquad \delta_{t+}\psi^{n-1/2} = g^n \delta_{t\circ}\eta^n \tag{64}$$

These can be consolidated into a single update yielding η^{n+1} from previously computed values of η and ψ:

$$\eta^{n+1} = \Theta\left(\eta^n, \eta^{n-1}, \psi^{n-1/2}\right) \tag{65}$$

and, given η^{n+1}, the second of (64) may be updated to yield $\psi^{n+1/2}$, at which point the simulation advances to the next time step. For more details, see [18].

3.5. A Complete Discrete-time Modular Network

Returning to the complete modular system presented in Section 2.3, the numerical scheme corresponding to (29) is

$$\begin{aligned} \ell^{(q)} u^{(q)} &= -\sum_{\nu \in \mathbb{I}_c^{(q)}} f_c^{(\nu)} \frac{1}{h^{(q)}} \mathcal{I}\left(x_c^{(\nu)}\right) + \sum_{\xi \in \mathbb{I}_e^{(q)}} f_e^{(\xi)} \frac{1}{h^{(q)}} \mathcal{I}\left(x_e^{(\xi)}\right) \\ \delta_{tt} w^{(j)} &= \frac{1}{M^{(j)}} \sum_{\nu \in \mathbb{I}_c^{(j)}} f_c^{(\nu)} \end{aligned} \tag{66}$$

Here, $u^{(q)} = u_l^{(q),n}$ is the grid function corresponding to the qth string, with grid spacing $h^{(q)}$, and defined over $l \in \{0, \ldots, N^{(q)}\}$ and $\ell^{(q)}$ is the difference operator derived from $\mathcal{L}^{(q)}$. $w^{(j)} = w^{(j),n}$ is the time series corresponding to the displacement of the jth lumped object, $j = 1, \ldots, N_m$ and $f_c^{(\nu)}$, $\nu = 1, \ldots, N_c$ to the associated connection forces. In (3.5) above, the time step n accompanying all dependent variables is suppressed for brevity.

Each of the N_c connection forces $f_c^{(\nu)} = f_c^{(\nu),n}, \nu = 1 \ldots, N_c$ is assumed dependent on a displacement

$$\eta^{(\nu),n} = u_{l_c^{(\nu)}}^{(q_\nu),n} - w^{(j_\nu),n} \tag{67}$$

There is assumed an underlying discrete potential $\phi^{(\nu),n-1/2}$ (and accompanying function $\psi^{(\nu),n-1/2}$), such that, for the non-iterative algorithm,

$$f_c^{(\nu),n} = \mu_{t+}\psi^{(\nu),n-1/2} \frac{\delta_{t+}\psi^{(\nu),n-1/2}}{\delta_{t\circ}\eta^{(\nu),n}} \tag{68}$$

Full details of the implementation and energy analysis will not be presented here, because of space limitations, but follow in almost all respects from the analysis of the single string/connection described in the previous section. A demonstration of energy conservation appears in Section 4.1, and of the benefit of a non-iterative algorithm, in terms of computation time, in Section 5.

4. SIMULATION RESULTS

In this section, we present some numerical results illustrating numerical and musical aspects of the modular system described here. All simulations are run at 44.1 kHz.

4.1. Energy Conservation

Consider first the energy conservation property, for a full network of 25 strings (ranging in pitch from C2 to C4), and using 24 two-sided connections, where the non-iterative scheme is employed. A worst case is assumed here, where the strings are lossless, and the initial conditions of the strings and lumped elements are randomised. In Figure 5, the energy partition as a function of time between strings and connections is shown, alongside the normalised energy variation, illustrating energy conservation to machine accuracy. The availability of such a measure allows for an excellent approach to debugging modular synthesis codes.

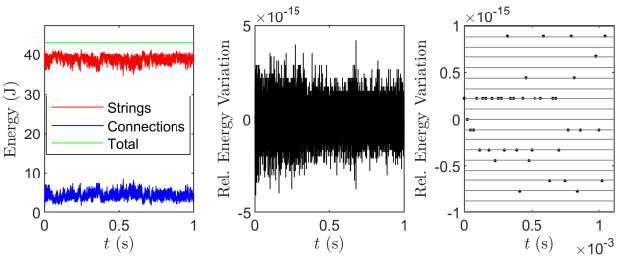


Figure 5: Left: energy partition, showing energy in strings, connections, and the total. Middle: normalised sample to sample energy variation, and right: a detail showing energy conservation to machine accuracy.

4.2. Single String/Lumped Object

An illustration of the interaction between a single string and a lumped object in intermittent contact is shown in Figure 6. Even in this extremely simple case, a wide variety of timbres is possible. In Figure 7, spectrograms of sound output are shown where string parameters are chosen according to a high E string on a guitar.

4.3. Large network

Consider now a network of 25 strings with 24 double-sided connections, representing approximately the largest possible in real time at 44.1 kHz. Snapshots of the time evolution of this network are shown in Figure 8.

5. COMPUTATIONAL ANALYSIS

This section details the computational costs of the element types (stiff string and connection) from which the system is built. We

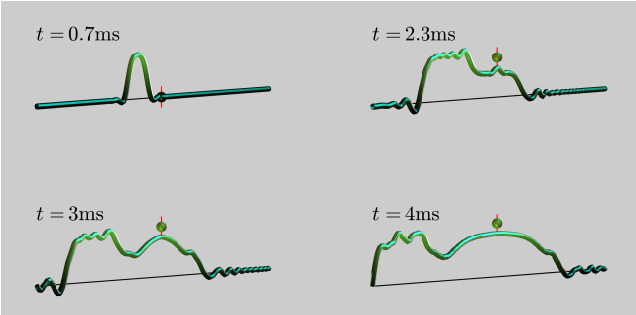


Figure 6: Interaction of a single stiff string with a lumped colliding object, at times as indicated.

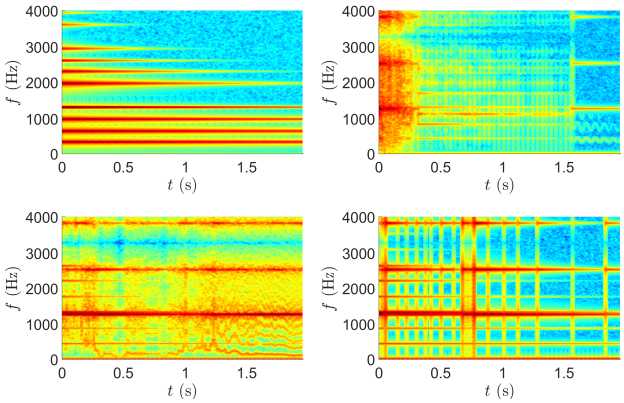


Figure 7: Spectrograms of sound output for a high E guitar string connected to a single lumped element. Top left: free vibration. Top right, under a single connection, illustrating abrupt changes in timbre over a long time scale. Bottom left: illustrating low-frequency "warbling" effects. Bottom right: for a high mass lumped object, leading to distinct bounces or temporal events.

start with a set of strings, and then compare an individual nonlinear connection solver in both iterative and non-iterative form. Finally we assess a full system of strings and connecting elements, as could be used in a real-time audio plug-in. All simulations are written in C++, and timings are produced by running at a sample rate of 44.1 kHz (in double precision) over 441,000 time steps and then dividing by 10 to give an average performance for 44,100 steps. The test CPU is a 6-core Intel Xeon E5-1650 v2, and the LLVM compiler was used. Whilst we will not consider multithreading here (as this presents problems for real-time usage) we do make use of vectorization, namely with Intel's SSE and AVX intrinsics [25].

The stiff string is simulated using the scheme (44), which is explicit, and amenable to spatial parallelization. The state size and computational cost can become large, but not unmanageably so: for the case of 25 low-pitched strings tuned from C2 at 65Hz to middle C4, the total state size required is 2,550 values. Table 1 shows results using various compiler optimisation levels, and also using manually written vector intrinsics which lead ultimately to a speedup by a factor of four. Performance is faster than real time.

Consider now a single nonlinear connection. Two forms of code were tested: first a form which uses an iterative Newton-

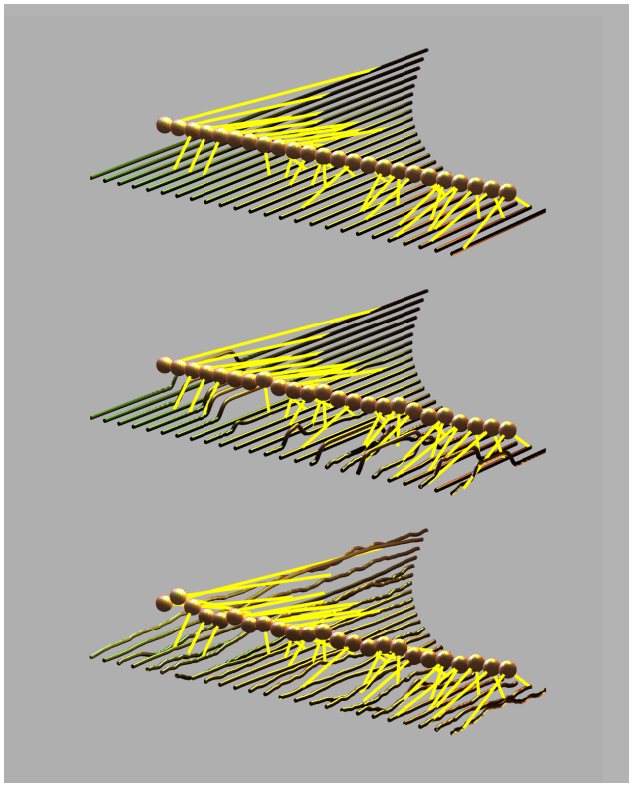


Figure 8: Large modular network, at rest (top), under initial excitations (middle) and later (bottom).

Table 1: *Computation times for 25 strings, tuned from C2 to C4, for 1 s output. Total state size: 2,550 grid points.*

Optimisation Level	Time (s)
-O0	0.59
-O3 SSE only enabled	0.18
Manual AVX	0.13

Raphson solver, and then a form which uses the new non-iterative solver. For the purposes of this testing the Newton-Raphson solver was run for 10 iterations at each time step, starting from a cached solution. This is a usable "average" figure for the simulation, although in practice could be larger, and also possibly reduced by testing for a residual.

The algorithm for solving for each connection force consists of some initial arithmetic operations and two calls to the `power()` function with non-integer exponents (using `std::pow()` from the cmath library). Then the Newton-Raphson solver is launched. This requires a further two calls of the `power()` function at each iteration. Finally there are some further arithmetic calculations to compute the forces at each end of the connection. The non-iterative form is considerably simpler, requiring just two `power()` functions in total. Table 2 shows the resulting computation times. The benefit of the non-iterative form is clear, giving a 11× speed up over the iterative form. This is mostly due to the reduction in the number of calls to the `power()` function, from 22 in the iterative form to just two in the latter. Further optimisation may be

Table 2: *Compute times for an individual connection for 1 s.*

Optimisation	Iterative (ms)	Non-Iterative (ms)	Speedup
-O0	79.2	7.1	11.2×
-O3 SSE	73.5	6.3	11.6×

achieved by implementing a manually vectorized power function across groups of connection calculations.

The final test simulation involved two octaves of strings with 24 two-sided rattle connections, such that each string was connected to another through a lumped element—a single strike to any string will excite the entire network, as in Figure 8. At each time step the strings are updated first, followed by a loop over the connections which then adds back the calculated forces into the strings at the connection points. Table 3 shows the results for the full system.

Table 3: *Computation times: 25 strings/24 connections for 1 s.*

Optimisation	Iterative (s)	Non-Iterative (s)	Speedup
-O0	2.53	0.81	3.1×
-O3 SSE	1.93	0.35	5.5×
Manual AVX	1.84	0.28	6.6×

The version computed using an iterative solver results in a best time of 1.84 s for 1 s sound output, which is clearly well short of the real time threshold. The non-iterative solver results in a 6.6× speedup, and the resulting time of 0.28 seconds is just within the scope of usability for a real-time application. Note that performance levels may vary according to other factors such as the CPU-specific cache, and other compiler details.

6. CONCLUDING REMARKS

A form of this system available as a real-time plug-in is under development through Physical Audio [26], called *net2*. It is rewritten from a previous version (called *net1*), which ran offline in multicore using an iterative solver. The ability to sidestep such iterative methods, and maintain a numerically-stable synthesis algorithm has allowed this move to real time through the approximately 10× speedup for the nonlinear part of the simulation. Though only stiff strings have been shown here, the extension to systems including multiple vibrating plates is immediate—see, e.g., [5]. Also, only one type of nonlinearity has been presented here, but the non-iterative algorithm presented here is fully general.

A major consideration, at the level of the user experience, is in the UI design, particularly when there are potentially many stiff string/connection elements to manage—this has been partly dealt with during the design of the simpler Derailer system, also from Physical Audio. A deeper issue is that of exploring the design parameter space—for the system illustrated here with 25 strings and 24 connections, for example, there are 319 parameters to set. Some heuristics can be employed to make this more manageable (by, say, restricting string tunings), but clearly some more general strategy for finding interesting regions of the parameter space (perhaps invoking methods from machine learning) is necessary.

7. REFERENCES

[1] C. Cadoz, A. Luciani, and J.-L. Florens, "Responsive input devices and sound synthesis by simulation of instrumental mechanisms," *Comp. Music J.*, vol. 8, no. 3, pp. 60–73, 1983.

[2] D. Morrison and J.-M. Adrien, "Mosaic: A framework for modal synthesis," *Comp. Music J.*, vol. 17, no. 1, pp. 45–56, 1993.

[3] J.-M. Adrien, "The missing link: Modal synthesis," in *Representations of Musical Signals*, G. DePoli, A. Picialli, and C. Roads, Eds., pp. 269–297. MIT Press, Cambridge, Massachusetts, 1991.

[4] R. Rabenstein, S. Petrausch, A. Sarti, G. De Sanctis, C. Erkut, and M. Karjalainen, "Block-based physical modeling for digital sound synthesis," *IEEE Sig. Proces. Mag.*, vol. 24, no. 2, pp. 42–54, 2007.

[5] S. Bilbao, "A modular percussion synthesis environment," in *Proc. Int. Digital Audio Effects Conf.*, Como, Italy, 2009, pp. 321–328.

[6] S. Bilbao, A. Torin, P. Graham, J. Perry, and G. Delap, "Modular physical modeling synthesis on gpu," in *Proc. Int. Comp. Music Conf.*, Athens, Greece, 2014.

[7] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, Cambridge, UK, 1992.

[8] R. Müller and T. Hélie, "Power-balanced modelling of circuits as skew gradient systems," in *Proc. Int. Conf. Digital Audio Effects*, Aveiro, Portugal, 2018, pp. 1–8.

[9] K. Werner, V. Nangia, J. O. Smith, and J. Abel, "A general and explicit formulation for wave digital filtres with multiple/multiport nonlinearities and complicated topologies," in *Proc. Workshop Appl. Signal Proces. Audio Acoust.*, Mohonk, NY, 2015, pp. 1–5.

[10] A. Bernardini, K. Werner, A. Sarti, and J. O. Smith, "Modeling nonlinear wave digital elements using the lambert function," *IEEE Trans. Circ. Syst. I*, vol. 63, no. 8, pp. 1231–1242, 2016.

[11] S. Bilbao, *Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics*, John Wiley and Sons, Chichester, UK, 2009.

[12] V. Chatziioannou and M. van Walstijn, "An energy conserving finite difference scheme for simulation of collisions.," in *Proc. Stockholm Musical Acoust. Conf.*, Stockholm, Sweden, 2013.

[13] S. Bilbao, A. Torin, and V. Chatziioannou, "Numerical modeling of collisions in musical instruments," *Acta Acustica u. with Acustica*, vol. 101, no. 1, pp. 155–173, 2015.

[14] M. Ducceschi and S. Bilbao, "Modelling collisions of nonlinear strings against rigid barriers: Conservative finite difference schemes with application to sound synthesis," in *Proc. Int. Conf. On Acoust. (ICA 2016)*, Buenos Aires, Argentina, September 2016.

[15] X. Yang, "Linear and unconditionally energy stable schemes for the binary fluid-surfactant phase field model," *Comp. Methods Appl. Mech. Eng.*, vol. 318, pp. 1005–1029, 2017.

[16] N. Lopes, T. Hélie, and A. Falaize, "Explicit second-order accurate method for the passive guaranteed simulation of port-hamiltonian systems," in *Proc. 5th IFAC 2015*, Lyon, France, July 2015.

[17] A. Falaize, *Modélisation, simulation, génération de code et correction de systèmes multi-physiques audios: Approche par réseau de composants et formulation Hamiltonienne À Ports*, Ph.D. thesis, Université Pierre et Marie Curie, Paris, July 2016.

[18] M. Ducceschi and S. Bilbao, "Non-iterative solvers for nonlinear problems: The case of collisions," 2019, Under review, 22nd Int. Conf. Digital Audio Effects.

[19] A. Chaigne and A. Askenfelt, "Numerical simulations of struck strings. I. A physical model for a struck string using finite difference methods," *J. Acoust. Soc. Am.*, vol. 95, no. 2, pp. 1112–1118, 1994.

[20] P. Ruiz, "A technique for simulating the vibrations of strings with a digital computer," M.S. thesis, University of Illinois, 1969.

[21] C. Vallette, "The mechanics of vibrating strings," in *Mechanics of Musical Instruments*, A. Hirschberg, J. Kergomard, and G. Weinreich, Eds., pp. 116–183. Springer, New York, New York, 1995.

[22] C. Desvages and S. Bilbao, "Two-polarisation physical model of bowed strings with nonlinear contact and friction forces, and application to gesture-based sound synthesis," *Appl. Sci.*, vol. 6, pp. 135, 2016.

[23] K. Hunt and F. Crossley, "Coefficient of restitution interpreted as damping in vibroimpact," *ASME J. Appl. Mech.*, pp. 440–5, June 1975.

[24] A. Falaize and T. Hélie, "Passive Guaranteed Simulation of Analog Audio Circuits: A Port-Hamiltonian Approach," *Appl. Sci.*, vol. 6, pp. 273 – 273, 2016.

[25] N Firasta, M. Buxton, P. Jinbo, K. Nasri, and S. Kuo, "Intel AVX: New frontiers in performance improvements and energy efficiency," *Intel White Paper*, 2008.

[26] "Physical audio," `www.physicalaudio.co.uk`, Accessed: 2019-03-26.

A PERCEPTUALLY INSPIRED GENERATIVE MODEL OF RIGID-BODY CONTACT SOUNDS

James Traer
Dept. of Brain and Cognitive Sciences
MIT
Cambridge, MA, USA
jtraer@mit.edu

Maddie Cusimano
Dept. of Brain and Cognitive Sciences
MIT
Cambridge, MA, USA
mcusi@mit.edu

Josh H. McDermott *
Dept. of Brain and Cognitive Sciences
MIT
Cambridge, MA, USA
jhm@mit.edu

ABSTRACT

Contact between rigid-body objects produces a diversity of impact and friction sounds. These sounds can be synthesized with detailed simulations of the motion, vibration and sound radiation of the objects, but such synthesis is computationally expensive and prohibitively slow for many applications. Moreover, detailed physical simulations may not be necessary for perceptually compelling synthesis; humans infer ecologically relevant causes of sound, such as material categories, but not with arbitrary precision. We present a generative model of impact sounds which summarizes the effect of physical variables on acoustic features via statistical distributions fit to empirical measurements of object acoustics. Perceptual experiments show that sampling from these distributions allows efficient synthesis of realistic impact and scraping sounds that convey material, mass, and motion.

1. INTRODUCTION

The sounds that enter the ear are collectively determined by the physical processes that generate the acoustic waveform. Sound generation by rigid bodies is a classic physics problem and the processes by which material parameters (e.g. material, mass, motion) affect acoustic waveforms have been well characterized [11, 15, 27, 31]. Typically, physical sound synthesis is done by modelling in detail the relevant processes which lead to the generation of a sound. For example, rigid bodies are modelled as a mesh-grid of masses on springs [4, 5, 6, 28, 38, 41], or decomposed into small segments over which wave equations can be solved by Finite-Element or Boundary-Element-Methods (FEM/BEM) [3, 16, 23]. These models yield a set of resonant modes from which contact sounds can be synthesized. In practice such models require computing physical interactions at very small spatiotemporal scales, and are thus computationally expensive.

Humans perceive sounds in terms of physical variables [12, 34], and these perceptual abilities might inform sound synthesis approaches. When we hear the sound of a fork dropped upon a wooden table, we can make judgments about the size [7, 14, 37], material [2, 13, 17] and motion of the fork [19]. However, our discrimination abilities are limited. It is not clear that humans can tell a fork from a knife in such a case, for instance, let alone the detailed geometry of the fork. Indeed, perceptual experiments indicate that humans can infer broad material differences (e.g. metal vs wood) from contact sounds, but are less accurate for more precise judgments (e.g. distinguishing metal from glass) [13].

The coarse-grained nature of human material judgments suggest material perception is insensitive to mode properties within some tolerance. Exactly what tolerance remains an open question, but it suggests that synthetic modes need not have a detailed correspondence to those of an actual object to yield compelling sounds. We hypothesize that the auditory system infers coarse-grained material parameters from statistical properties of modes, rather than their precise details. For example, consider again the sound of a fork dropped upon a table. Although fine-grained features (e.g. the thickness of the handle, the length of the tines, the narrowing of the neck, etc.) may affect individual modes, we see little evidence that humans infer such subtle features. However, coarse-grained physical features, which are crucial to inferring scene properties like material and size, will affect all the modes and thus are likely to be reflected in the modal statistics.

Rather than attempt to simulate the physical process in fine-grained detail, we measure statistics of modes from real-world impact sounds and use these distributions as the building blocks for sound synthesis via a source-filter model (in which a time-varying force is convolved with the object impulse response). We synthesize sounds from both impacts and sustained frictional forces (Fig. 1). As with our statistical model of modes, the impact forces are parametrized only by coarse-grained properties: mass, stiffness, and velocity. For scraping sounds, the force is generated through a texture quilting algorithm [10], reflecting listeners' perception of summary statistics as opposed to fine-grained temporal detail in sound textures [25].

Our approach yields compelling renditions of sounds via a fast and efficient process. As with other similar approaches [1, 29], it is thus ideal for use in physics engines used in modern computer games and simulations. Such engines store a set of attributes for rigid-bodies to compute how they will move (e.g. mass, elasticity, frictional coefficients, a grid-model of the geometry, etc.) and to compute their appearance under lighting (e.g. diffuse and specular reflectance profiles, visual surface statistics, etc.). As conventional sound synthesis is slow, current engines rely on memory intensive sample banks of pre-recorded or pre-computed sounds to be played on contact. However, our synthesis model only requires a simple texture model and low-dimensional representations of coarse physical features, such as are already encoded for motion and visual appearance. From these crude features and a sample bank of mode distributions (e.g. wood, metal, plastic, ceramic, etc.), our synthesis algorithm can rapidly generate a range of realistic and unique contact sounds. Here we show that impact sounds generated in this

* Work supported by the Center for Brains, Minds and Machines and The MIT-IBM Watson AI Lab

way convey mass and material to listeners as well as recordings of real sounds. Scraping sounds derived from these mode distributions are also realistic and convey motion trajectories.

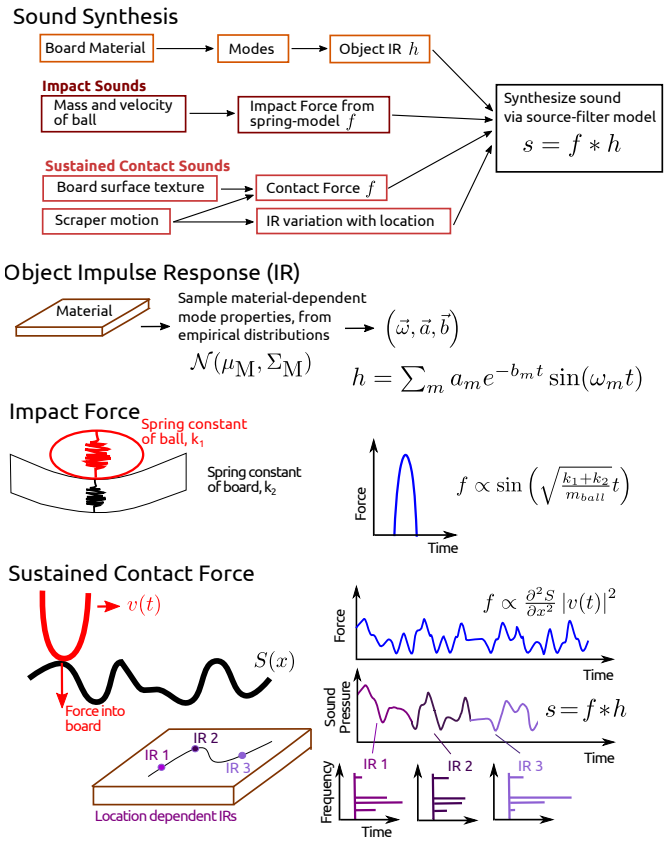


Figure 1: *We synthesize sounds by (top) a generative model of impact and sustained contacts. (Upper-middle) Object Impulse Responses are synthesized by sampling modes from empirical distributions. (Lower-middle) Impact forces are modelled via a spring model. (Bottom) Sustained contacts are modelled via measured surface textures and location-dependent IRs.*

2. SOURCE-FILTER MODEL OF IMPACTS

Our model is inspired by the well-known source-filter model [8]

$$s(t) = f(t) * [h_1(t) + h_2(t)] \quad , \tag{1}$$

where $s(t)$ is the sound entering a listener's ear, $f(t)$ is the contact-force between two objects and $h_j(t)$ is the impulse response (IR) of the $j$the object. Past sound synthesis techniques have computed high-resolution IRs with large grid models such as finite-element or boundary-element techniques [16, 24, 30], solved analytically for the resonant modes of an object of known material and shape [17, 21, 22, 38], or fit parameteric models of mode parameters to measured impacts[1]. The grid solutions are flexible but require significant computational power. The analytical modal solutions allow fast synthesis but only apply to a small subset of rigid bodies.

We approximate, as have others before [33], object IRs via the summation of a broadband transient "click" and a set of exponentially decaying sinusoids corresponding to the resonant modes of the object

$$h(t) = h_T(t) + \sum_m^M 10^{(a_m - b_m t)/20} \cos(\omega_m t) \quad , \tag{2}$$

where h_T is the transient, and (a_m, b_m, ω_m) are the onset power, decay rate and angular frequency of the mth mode. The transient can be described via a set of decaying noise-bands:

$$h_T(t) = \sum_n^N 10^{(\alpha_n - \beta_n t)/20} \nu_n(t) \quad , \tag{3}$$

where ν_n is a time-series of random noise filtered by the nth Equivalent-Rectangular-Bandwidth (ERB) filter of a cochleagram decomposition, and (α_n, β_n) are the onset power and decay rates of this channel. Under our model, an object IR can be completely described by $2N + 3M$ parameters, to precisely determine the shape of the transient and the modes. Throughout this work we use N=30 and M=15, which we found to be sufficient for compelling resynthesis.

Our preliminary experiments suggest several broad perceptual trends: (1) perception of material properties is dominated by a small number of powerful modes; (2) changes to the properties of weaker modes are barely noticeable; (3) slight changes to the most powerful modes are detectable, but the resulting sound is perceived as a different exemplar of a similar object or the the same object struck in a different location; (4) altering the transient but not the modes, has a minimal effect on perceived material. All of these perceptual trends suggest that human perception of object properties (i.e. material, size, shape) are primarily predicated upon the statistics of the most powerful object resonant modes.

2.1. Modal synthesis of object Impulse Responses (IRs)

To test our hypothesis that human judgments of object properties are based on mode statistics, we seek to synthesize impact sounds which match the modal statistics of real-world impacts, but are otherwise unconstrained (such that the exact mode parameters are different). We began by measuring the mode statistics from real-world objects.

To measure resonant modes, we recorded the sounds of a large number of materials being struck by small pellets. We estimated the resonant modes of each impact via an iterative procedure of spectrogram matching: (1) we obtained the frequency channel of the spectrogram of the impact sound with the maximum power; (2) we synthesized an initial synthetic impact with an exponentially decaying sinusoid at that frequency; (3) we adjusted the mode properties (frequency, onset power and decay rate) to minimize the mean-squared error between the spectrograms of the recording and the synthetic; (4) we subtracted the synthetic spectrogram from the original (removing the mode we just measured). We then repeated the procedure 14 times, yielding parameters for the 15 most powerful modes. After fitting the modes we repeat this procedure using exponentially decaying noise-bands instead of sinusoidal modes to fit the properties of the transient.

For each material, we recorded multiple impacts at different locations on multiple objects. We pooled together modes from multiple objects and characterized the mode statistics by fitting a multivariate Gaussian distribution to the resulting collection. We similarly fit distributions to the transient decay parameters.

To generate a synthetic IR, we sample both mode and and transient properties from our empirically measured distributions:

$$\begin{aligned}(\vec{a}, \vec{b}, \vec{\omega}) &= \mathcal{N}(\mu_{\mathrm{M}}, \Sigma_{\mathrm{M}}) \\ (\vec{\alpha}, \vec{\beta}) &= \mathcal{N}(\mu_{\mathrm{T}}, \Sigma_{\mathrm{T}}) \quad , \end{aligned} \tag{4}$$

where (μ_M, Σ_M) are the mean and covariance of the mode properties, conditioned upon the required object or material, and (μ_T, Σ_T) are the analogous mean and covariance of the transient subband properties. We used rejection sampling to ensure that the average frequency spacing between sampled modes was within 10% of that measured from recordings of the material. Because the mode statistics are computed offline prior to synthesis, all that needs to be encoded at time of sound synthesis are material labels which index distributions of IR properties.

To simulate multiple contacts of the same object we sample from the distributions once, and then randomly perturb mode onset powers (standard deviation=20% mean mode power) for each later impact. This emulates the fact that impacts in different locations differentially excite the same modes. We found empirically that either sampling from the distribution twice or repeating the exact same set of mode parameters produced unrealistic sounds [20].

2.2. Effect of impact physics

To synthesize an impact sound, we also need to compute the contact force, to be convolved with the object IR [Eq. (1)]. We approximate the contact force using a simple spring-model, in which the force acting on either object is proportional to the displacement of the surface at the point of contact. This yields the force between two objects as a half-wavelength of a sinusoid

$$f(t) = \begin{cases} \sin\left(\sqrt{\frac{k}{m}}t\right) & \forall \;\; 0 < t < \frac{\pi m}{k} \\ 0 & \text{otherwise} \end{cases} \quad , \tag{5}$$

where v is the velocity at impact, m the mass of the pellet and k a spring constant determined by the materials of the board and ball. Note that as the mass tends to zero, the time of contact between the two materials tends to zero and the contact force tends towards a Dirac-delta function. This observation partly justifies the use of small pellet impact recordings to approximate the object impulse response.

To synthesize impact sounds, we convolve a synthesized IR from Eq. (4) with the contact force described in Eq. (5). All that needs to be encoded at the time of impact are labels of object mass, velocity, and material labels, which determine both the spring constants and the distributions from which modes are sampled. Except for parameters of the mode distributions, these features are already included in physics engines.

3. PERCEPTION OF SYNTHETIC IMPACTS

To assess our impact synthesis model we played both recorded and synthesized sounds to listeners and asked them to judge: (1) realism; (2) material; and (3) mass of the colliding objects. All perceptual experiments were conducted over Amazon's Mechanical Turk platform. A standardized test was used to ensure participants were wearing headphones [40].

3.1. Experiment 1. Realism of synthetic impact sounds

We first sought to test whether our synthetic sounds were compelling renditions of real-world impacts. If our synthesis method neglected sound features to which the brain is sensitive, the synthetic sounds should be recognizable as fake.

Participants were presented with a pair of impact sounds and identified which was the real recording. In all trials, one sound was a real-world recording of a ball dropped on a resonant object, and one a synthetic impact generated via our model or a model that was 'lesioned' in some way, by omitting the transient component of the IR, or by omitting the modes from the IR. The conditions of the experiment were (1) full synthetic model; (2) Modes only, without transient; (3) Transient only, without modes; (4) Time-reversed synthetics. The sound in the final condition were clearly synthetic, which serves to ensure task comprehension.

The results (Fig. 2) show that listeners could not distinguish sounds from either the full or lesioned models from real-world recordings, demonstrating that our method of impact sound synthesis yields plausible sounds. The chance performance for the lesioned models presumably reflects the fact that the resulting sounds remained realistic even though the lesion altered the quality of the sounds. As participants were good at identifying the Time-Reversed sounds it is clear they understood the task. Poor performance in the other conditions thus reflects the success of the synthesis.

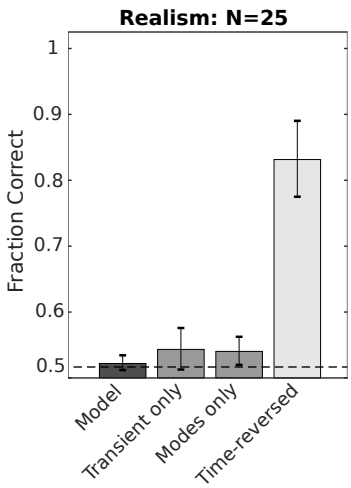


Figure 2: *Discrimination of real vs. synthetic impact sounds (Exp 1). Dashed line denotes chance performance.*

3.2. Experiment 2. Perception of material

Having demonstrated that our synthetic impact sounds are realistic, we sought to test whether they convey appropriate physical parameters to listeners. We first tested whether listeners can recognize the material of a struck resonant object.

Participants heard a single impact sound and were asked to identify the material of the struck object from one of four possible categories: metal, ceramic, wood or cardboard. Participants were told that the striking mallet was effectively noiseless and that many different objects of each material class were used, of a range of

different sizes, shapes and sub-material (i.e. metal contained steel, tin, aluminium etc.; wood contained poplar, pine, oak etc.)

With real-world recordings, participants were excellent at distinguishing hard materials (metal or ceramic) from soft materials (wood or cardboard) but made errors within the hard or soft categories (Fig. 3). This result is consistent with prior studies [13]. Sounds from our synthesis model - both with and without the transient - yielded a similar pattern of success and failures. Without modes, or with shortened modes, human judgments were strongly biased towards to softer materials. With lengthened modes, judgments were biased towards harder materials, particularly metal. This demonstrates that our model - particularly the mode statistics - have captured the acoustical features that humans use to judge material classes from impact sounds. The correlation of the confusion matrices for the full model and recorded sounds was 0.72.

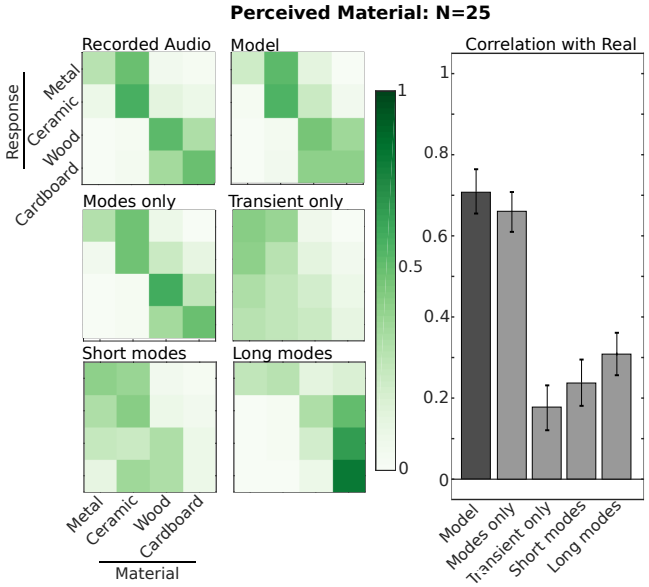


Figure 3: *Material discrimination from synthetic impact sounds (Exp 2). Left: Confusion matrices of the presented material and participant responses. Right: Correlation of the confusion matrices of various synthetic sounds with that of the recorded impacts.*

3.3. Experiment 3. Perception of mass

We next sought to test whether our synthetic sounds convey the mass of the striking mallet to listeners. Participants heard two impact sounds, one of a small wooden pellet (0.7 g) dropped onto an object, and one of a larger wooden ball (7.6 g) dropped onto the same object. Participants were asked to identify which of the two balls was heavier. To generate synthetic sounds the synthetic IRs were convolved with two different contact forces to emulate different ball masses, as shown in Eq. (5). The impact levels were not normalized but retained the relative variation in power level induced by the difference in impact force (i.e. the coefficient in Eq. (5) and the amplitude of the IR). All recordings and simulations were made with balls dropped from the same height (8 cm), but participants were not explicitly told this.

Since we do not know k, the spring constant, we cannot compute the contact force [Eq. (5)]. Instead we estimate k from the recorded impact sounds. Since both balls are the same material, we assume $k_{\text{large}} = k_{\text{small}}$, which means the ratio between the contact times for the two balls is $m_{\text{large}}/m_{\text{small}}$. We set the contact time of the larger ball to be 10.9 times that of the smaller ball. We then iteratively adjusted the contact time of the smaller ball, until it produced a match between the average spectral centroid of the synthetic sounds and of the corresponding impact recordings.

The results (Fig. 4) show that humans perform very well at this task, both with real-world recordings and with synthetic sounds. This demonstrates that humans are sensitive to the filtering effect described by the contact force and can use this acoustic information to estimate the mass of the striking mallet. Participants showed a small performance decrement in the conditions where modes were shortened or excluded altogether, suggesting that humans are using modes, in addition to the sound level and spectral centroid, to estimate mass. The results suggest that our synthetic sounds convey mass as well as real-world recordings.

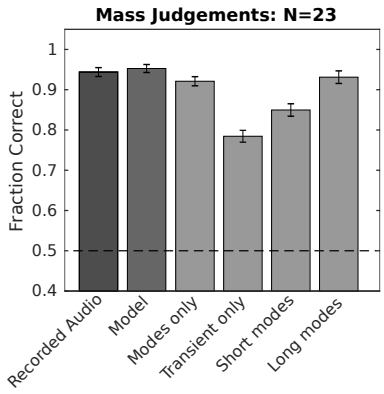


Figure 4: *Mass discrimination with real and synthetic impact sounds (Exp 3).*

4. SUSTAINED CONTACTS

To test the generalizability of our impulse response distributions, we next consider sustained contacts such as made by two objects scraping across each other. Similar to [32], we again use the source-filter model of Eq. (1) but both the force and object IRs are more complicated than for impact sounds. The contact force $f(t)$ is generated by a series of small collisions as the scraper moves across the surface of the scraped object, and is thus a function of the downward force applied to the scraper, the surface texture depth, and the scraper speed (Fig. 1, bottom). The object IR changes with scraper position $x(t)$, and thus, as the scraper moves across the surface, the IR becomes a time-varying function $h_{\text{surface}}(x(t))$. We describe these models of force and IR in more detail below. Despite the simplicity of this model, our results suggest that it yields plausible scraping sounds which convey motion of the scraper.

4.1. Contact force for sustained contacts

To model the force between scraper and surface we start with several simplifying assumptions: that the external force applied to the scraper F_{p} is constant and applied vertically downwards, and that the probe follows the surface exactly without any slip or bounce,

such that the probe height $z(t)$ at time t, is given by the surface elevation $S(x)$ at the probe location x. For now we consider a transect across the surface so x is a one-dimensional variable, though the following analysis applies easily to a 2D treatment.

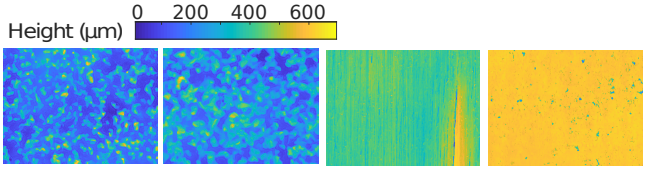


Figure 5: *Everyday textures measured with the confocal microscope. Surface area is 7.3 mm by 10 mm. From-left: 100 grit sandpaper; 60 grit sandpaper; wood; vinyl tile.*

We first consider the vertical component of the force. Under our assumptions, the change in vertical force applied to the surface can be derived from the vertical acceleration of the probe, which, as the probe follows the surface, is given by

$$\begin{aligned} f_{\mathrm{v}}(t) &= m_{\mathrm{p}}\ddot{z} \\ &= m_{\mathrm{p}}\frac{\partial^2 S}{\partial x^2}\,|v(t)|^2 \quad , \end{aligned} \tag{6}$$

where m_{p} is the mass of the probe and $v(t)$ is the horizontal velocity of the probe.

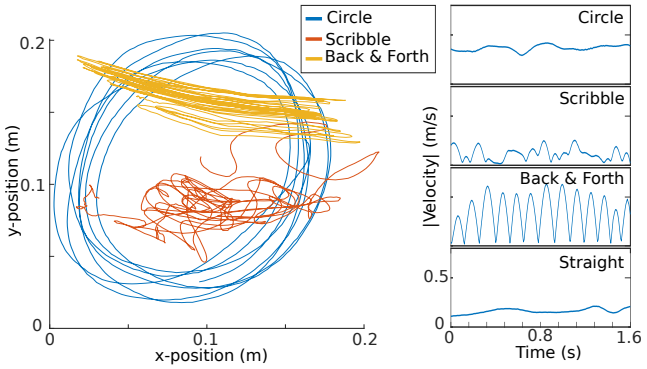


Figure 6: *Scraping motions. Left: Measured position traces of scraper over surface, for three different types of motion. Right: Absolute velocity measurements.*

We next consider the frictional force tangential to the surface. We model this as proportional to the probe speed raised to the power of an exponential factor γ, giving

$$f_{\mathrm{h}}(t) \propto \left(\left|v(t)\frac{\partial S}{\partial x}\right|\right)^{\gamma} \quad , \tag{7}$$

where the partial derivative with respect to x accounts for the difference between the speed of scraping across the surface, which is the important factor, and the horizontal speed $|v(t)|$. The total force imparted onto the object is then given by

$$\begin{aligned} f(t) &= f_{\mathrm{v}} + f_{\mathrm{h}} \\ &= m_{\mathrm{p}}\frac{\partial^2 S}{\partial x^2}v(t)^2 + A\left(\left|v(t)\frac{\partial S}{\partial x}\right|\right)^{\gamma} \quad , \end{aligned} \tag{8}$$

where A and γ are unknown constants which titrate the importance of shear friction versus vertical forcing. We explore the role of these factors by listening to synthetic scrape sounds from a range of values. We have neglected the constant downward force term F_{p} which, though present, does not create any sound.

To obtain S, we measured the surface texture of several real objects using a scanning confocal microscope (Keyence VK-X260K). In these experiments, we used a micro-scale depth map of a small section of a wood block (Fig. 5). These are relatively small matrices (1600 pixels by 2300 pixels), which render the surface with horizontal resolution of 5.6 μm and vertical resolution of 0.1 nm. Based on perceptual results concerning auditory texture perception, we expect that the perceptually important properties of such textures are statistical [26]. Therefore, to define S, we use one-dimensional quilting to generate a texture from a measured depth map [10], sampling a series of single rows and concatenating them. In future work, we plan to synthesize these surfaces statistically from coarse-level variables, in the same spirit as our distribution over impulse responses.

In addition to a depth map, the synthetic scraping force requires ecologically plausible velocity profiles of scraping motions. To probe the mechanics of typical human scraping movements, we measured the velocity and position profiles of several scraping movements using an optical tracking system (OptiTrack V120:Trio; Fig. 6). We use these recorded trajectories in the reported synthesis. However, in informal experiments, we found that the quality of sound synthesis was not heavily dependent on a precise match to the recorded data. Future work will include simple statistical models of these trajectories.

4.2. Variation of IRs over contact location

Object IRs depend upon the location being struck, and thus to simulate scraping we model this variation of modal properties with probe location. To informally assess the variability of mode properties as a function of impact location, we compared impact recordings we had made with different strike locations and found the variation in mode properties to be moderate. To emulate such changes with synthetic IRs, we synthesized a single canonical IR from our model [Eq. (4)], with properties $(\mathbf{a}_o, \mathbf{b}, \omega)$, and simulated a number of location specific IRs by adding some noise to the mode powers

$$(\vec{a}, \vec{b}, \vec{\omega}) = (\vec{a_{\mathbf{o}}} + \vec{\epsilon}, \vec{b}, \vec{\omega}) \quad , \tag{9}$$

where $\vec{a_{\mathbf{o}}}$ is the original vector of mode powers sampled from our model and $\vec{\epsilon}$ is a Gaussian noise vector sampled with zero-mean and a standard-deviation set to 20% the mean mode onset power. This gives a set of IRs with similar but varying modes, which crudely emulate an object of arbitrary shape struck in various locations.

We assign these sampled IRs to points along a motion trajectory, and interpolated between them in waveform space to give a smoothly varying surface IR, $h_{\mathrm{surface}}(x(t))$. When the scraper was at a position between the defined centerpoints, the impulse response was a linear combination of the impulse responses with weights proportional to the relative distances from the scraper to the centerpoints. We ignore the contribution of the scraper to the impulse response, assuming that it is damped by the hand in which it is held.

5. PERCEPTION OF SYNTHETIC SCRAPING

To assess the efficacy of our scraping synthesis model, we played both recorded and synthesized sounds to listeners and asked them to judge: (1) realism; and (2) the shape of the scraper's position trajectory. As in section 3, all experiments were conducted online using Amazon's Mechanical Turk platform, and a standardized test was used to ensure participants were wearing headphones [40]. In each experiment, in addition to testing lesioned forms of our own synthesis model, we compare our model to the one other scraping synthesis method that we are aware has been tested psychophysically [35]. Thoret et al. generated low-pass filtered white noise whose amplitude and filter cutoff increased with increasing velocity, and showed that several motion trajectories could be accurately judged from the resulting sounds.

5.1. Experiment 5. Realism of synthetic scraping sounds

Participants were played a pair of scraping sounds and asked to identify which was the real recording. In all trials, one sound was a real-word recording of chopstick scraping a board, and one a synthetic scrape generated via our model or a lesioned version thereof. The synthetic conditions of the experiment were generated via (1) the full model, using measurement-based surface textures and varied IRs; (2) measured depth map and just a single IR; (3) pink noise depth map and varied IRs; (4) white noise with varied filter cutoff from [35]; and (5) velocity-gated white noise, which is silent when the chopstick moves more slowly than a threshold, but otherwise constant. Condition (5) is clearly synthetic and serves to ensure the participants understand the task.

The results (Fig. 7) show that the full synthesis model, though not perfectly realistic, frequently fools listeners. However, using a time-varying impulse response does not improve realism over filtering with a single synthetic impulse response. A synthetic noise depth map also produced comparably realistic sounds. Our sounds were less obviously synthetic than those of [35], but one caveat is that the comparison recordings were produced by a narrow scraping probe. We suspect that condition (4), with its flat broadband spectrum, may be more appropriate for modeling scrapes produced by heavier objects with large contact surface area (e.g. pushing a heavy box over tile). The gated white-noise is easily recognized as synthetic by the participants, demonstrating that they understood the task.

5.2. Experiment 6. Perception of motion

Participants were presented with a single scraping sound and asked to choose the scraping trajectory from four choices: "circular", "back-and-forth", "scribble", or "straight". Participants heard both real-world recordings and synthetic sounds derived from a real-world motion. The motion trajectories used to generate synthetic scrapes were matched in speed to the scrapers used to make the real-world recordings.

As shown in Fig. 8, motion judgments for synthetic scrapes were similar to those for real-world scrape recordings. In both cases participants were correct most of the time, but misjudged "straight" motions to be "circular", both of which have velocity profiles without zero points. When judging either "back-and-forth" or "scribble" sounds, the full model and its lesioned variants led to more "scribble" judgments. This result could reflect the greater scattering of contact position around the surface in scribbling compared to other motions. Although we attempted to simulate this positional change with changing IRs, the full model and the constant IR model were comparable for both realism and motion, suggesting that we did not successfully capture this informative spatial variation.

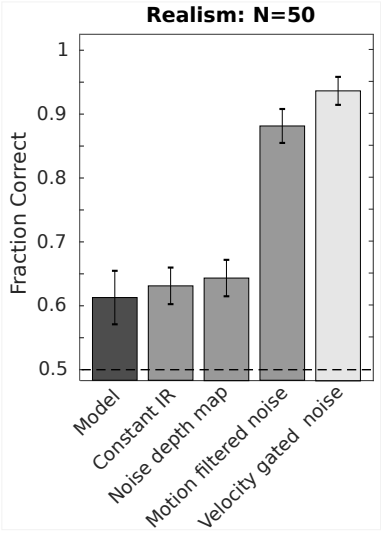


Figure 7: *Discrimination of real vs. synthetic impact sounds scraping sounds (Exp 5). Dashed line indicates chance performance.*

6. DISCUSSION

Our synthesis model is fast because it only models the effects of a small number of physical variables (material, mass, velocity etc.). It is evident from daily life that humans can infer more than just the variables we have described from contact sounds. Impact sounds contain cues to shape, size, and hollowness, as well as to the environmental reverberation [36]. Some physical variables explored in our impact model can also be conveyed by frictional sounds (e.g. material) but this remains to be explored in future work. Furthermore, friction sounds are not limited to scraping, but rather include other interactions such as rubbing, brushing and sliding. Future investigations into how these interactions produce sound, and into human sensitivity to their properties, will hopefully suggest extensions to a better and more nuanced synthesis algorithm.

The current version of our synthesis model requires some physical measurements of real-world objects: statistical distributions of object IRs conditioned upon material parameters; and surface structures. In future we hope to be able to synthesize these intermediate representations from physical variables. Our impact experiments with altered IRs demonstrated that lengthening or shortening the resonant modes caused listeners to rate the synthetic materials as"harder" or "softer" materials, consistent with physical models [13, 17, 21], but did not diminish their realism. This suggests that we should be able to synthesize IRs for novel objects without having to measure them first, permitting sound synthesis for a much larger range of objects. Similar generalizations should be possible for the forcing functions used to generate scraping sounds. As with perception of acoustic textures [25], it is likely that humans are insensitive to the fine-grained temporal details of the contact force we use to synthesize scrapes. Presumably we can synthesize such a contact force directly from a texture model [26],

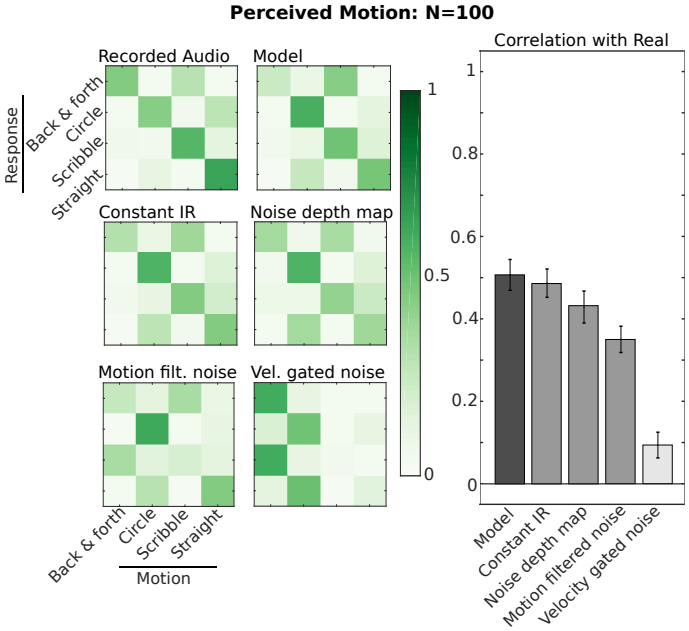


Figure 8: *Motion discrimination from synthetic scrape sounds (Exp 6). (Left) Confusion matrices of presented motion pattern and the human responses. (Right) Correlations of the confusion matrices of synthetic sounds with the correlation matrix of recorded sounds.*

enabling sound synthesis for a wider and more diverse range of objects without costly and time-consuming measurements.

Our impulse response model, while derived from statistics of impact sounds, can successfully contribute to the synthesis of relatively realistic scraping sounds. However, it appears that this model does not accurately capture the spatial covariance between impulse responses over a surface. Our full model and lesioned model with a single IR perform equally well, and neither are yet on par with real recordings, both in terms of realism and in the conveyed motion (Fig. 7, Fig. 8). Future investigations will include measurements and modelling of this variation in impulse responses based on position, as well as comparing modes measured from scraping sounds with those from impacts. The other component of the scraping synthesis is an excitation force based on quilted textures of measured depth maps. Several authors have treated scraping as a noisy source paired with a modal filter. Some model the friction force as $1/f^{\beta}$ noise [9, 32, 39], while others use a statistical model of densely-packed impact events [18]. In the experiments explored here, the utilization of real-world measurements did not improve realism or motion inference. However, it remains possible that constraining a more sophisticated model of surface texture with these measurements could be useful, particularly in judgments of material and surface roughness.

The model we have presented is similar in some respects to that of Conan et al. [8], who used statistics of contact forces to synthesize rolling sounds. We also utilize a statistical approach, but model the sounds of impacts and scraping, using statistics of the resonant modes of objects. We also found that we could use a linear model for contact forces. By contrast, Conan et al. found that a non-linearity in impact force (namely that the duration of impact should change with impact force) was required to induce realistic rolling sounds. In the future, we plan to investigate whether there are perceptual benefits to sound synthesis with more realistic impact forces.

7. CONCLUSION

We have presented a fast and efficient method for synthesis of contact sounds - inspired by both physics and perception. The method generates object IRs by sampling resonant modes from distributions fitted to empirical measurements from example impact sounds. The method then convolves the IRs with contact force simulated with a simple physics model of either impacts or sustained scrapes. Despite the simplicity of the model, perceptual listening tasks demonstrate that the synthetic sounds are realistic and convey basic physical information as well as recorded sounds. These results suggest that our model has captured many of the acoustic features that matter for perception of physical contact sounds, despite neglecting a great deal of physical information about the sound sources.

8. REFERENCES

[1] Mitsuko Aramaki, Mireille Besson, Richard Kronland-Martinet, and Sølvi Ystad. Controlling the perceived material in an impact sound synthesizer. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(2):301–314, 2010.

[2] Federico Avanzini and Davide Rocchesso. Controlling material properties in physical models of sounding objects. In *ICMC*, 2001.

[3] Stefan D Bilbao. *Numerical sound synthesis*. Wiley Online Library, 2009.

[4] Claude Cadoz. *Synthèse sonore par simulation de mécanismes vibratoires. Applications aux sons musicaux*. PhD thesis, Institut national polytechnique de Grenoble, 1979.

[5] Claude Cadoz, Annie Luciani, and Jean-Loup Florens. Responsive input devices and sound synthesis by stimulation of instrumental mechanisms: The cordis system. *Computer music journal*, 8(3):60–73, 1984.

[6] Claude Cadoz, Annie Luciani, and Jean Loup Florens. Cordis-anima: a modeling and simulation system for sound and image synthesis: the general formalism. *Computer music journal*, 17(1):19–29, 1993.

[7] Claudia Carello, Krista L Anderson, and Andrew J Kunkler-Peck. Perception of object length by sound. *Psychological science*, 9(3):211–214, 1998.

[8] Simon Conan, Olivier Derrien, Mitsuko Aramaki, Sølvi Ystad, and Richard Kronland-Martinet. A synthesis model with intuitive control capabilities for rolling sounds. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(8):1260–1273, 2014.

[9] Simon Conan, Etienne Thoret, Mitsuko Aramaki, Olivier Derrien, Charles Gondre, Solvi Ystad, and Richard Kronland-Martinet. Intuitive synthesizer of continuous-interaction sounds: Rubbing, scratching, and rolling. *Computer Music Journal*, 38(4):24–37, 2014.

[10] Alexis A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. *SIGGRAPH: Computer Graphics*, 2001.

[11] Neville H Fletcher and Thomas D Rossing. *The physics of musical instruments*. Springer Science & Business Media, 2012.

[12] William W Gaver. What in the world do we hear?: An ecological approach to auditory event perception. *Ecological psychology*, 5(1):1–29, 1993.

[13] Bruno L Giordano and Stephen McAdams. Material identification of real impact sounds: Effects of size variation in steel, glass, wood, and plexiglass plates. *The Journal of the Acoustical Society of America*, 119(2):1171–1181, 2006.

[14] Massimo Grassi. Do we hear size or sound? balls dropped on plates. *Perception & psychophysics*, 67(2):274–284, 2005.

[15] Hermann LF Helmholtz and Alexander J Ellis. On the sensation of sound in general. 1875.

[16] Doug L James, Jernej Barbič, and Dinesh K Pai. Precomputed acoustic transfer: output-sensitive, accurate sound generation for geometrically complex vibration sources. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 987–995. ACM, 2006.

[17] Roberta L Klatzky, Dinesh K Pai, and Eric P Krotkov. Perception of material from contact sounds. *Presence: Teleoperators & Virtual Environments*, 9(4):399–410, 2000.

[18] Mathieu Lagrange, Gary Scavone, and Philippe Depalle. Analysis/synthesis of sounds generated by sustained contact between rigid objects. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3):509–518, 2010.

[19] Guillaume Lemaitre and Laurie M Heller. Auditory perception of material is fragile while action is strikingly robust. *The Journal of the Acoustical Society of America*, 131(2):1337–1348, 2012.

[20] D Brandon Lloyd, Nikunj Raghuvanshi, and Naga K Govindaraju. Sound synthesis for impact sounds in video games. In *Symposium on Interactive 3D Graphics and Games*, pages PAGE–7. ACM, 2011.

[21] Robert A Lutfi. Human sound source identification. In *Auditory perception of sound sources*, pages 13–42. Springer, 2008.

[22] Robert A Lutfi and Christophe NJ Stoelinga. Sensory constraints on auditory identification of the material and geometric properties of struck bars. *The Journal of the Acoustical Society of America*, 127(1):350–360, 2010.

[23] Dinesh Manocha and Ming C Lin. Interactive sound rendering. In *2009 11th IEEE International Conference on Computer-Aided Design and Computer Graphics*, pages 19–26. IEEE, 2009.

[24] Stephen McAdams, Vincent Roussarie, Antoine Chaigne, and Bruno L Giordano. The psychomechanics of simulated sound sources: Material properties of impacted thin plates. *The Journal of the Acoustical Society of America*, 128(3):1401–1413, 2010.

[25] Josh H McDermott, Michael Schemitsch, and Eero P Simoncelli. Summary statistics in auditory perception. *Nature neuroscience*, 16(4):493, 2013.

[26] Josh H McDermott and Eero P Simoncelli. Sound texture perception via statistics of the auditory periphery: evidence from sound synthesis. *Neuron*, 71(5):926–940, 2011.

[27] Philip McCord Morse and K Uno Ingard. *Theoretical acoustics*. Princeton university press, 1986.

[28] James F O'Brien, Chen Shen, and Christine M Gatchalian. Synthesizing sounds from rigid-body simulations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 175–181. ACM, 2002.

[29] Laurent Pruvost, Bertrand Scherrer, Mitsuko Aramaki, Sølvi Ystad, and Richard Kronland-Martinet. Perception-based interactive sound synthesis of morphing solids' interactions. In *SIGGRAPH Asia 2015 Technical Briefs*, page 17. ACM, 2015.

[30] Nikunj Raghuvanshi and Ming C Lin. Interactive sound synthesis for large scale environments. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 101–108. ACM, 2006.

[31] John William Strutt Baron Rayleigh. *The theory of sound*, volume 1. Macmillan, 1896.

[32] Zhimin Ren, Yeh Hengchin, and Ming C. Lin. Synthesizing contact sounds between textured models. University of North Carolina at Chapel Hill, 2010.

[33] Zhimin Ren, Hengchin Yeh, and Ming C Lin. Example-guided physically based modal sound synthesis. *ACM Transactions on Graphics (TOG)*, 32(1):1, 2013.

[34] Davide Rocchesso and Federico Fontana. *The sounding object*. Mondo estremo, 2003.

[35] Etienne Thoret, Mitsuko Aramaki, Richard Kronland-Martinet, Jean-Luck Velay, and Solvi. Ystad. From sound to shape: auditory perception of drawing movements. *Journal of Experimental Psychology: Human Percept ion and Performance*, 40(3):983–994, 2014.

[36] James Traer and Josh H McDermott. Statistics of natural reverberation enable perceptual separation of sound and space. *Proceedings of the National Academy of Sciences*, 113(48):E7856–E7865, 2016.

[37] Simon Tucker and Guy J Brown. Investigating the perception of the size, shape and material of damped and free vibrating plates. *University of Sheffield, Department of Computer Science Technical Report CS-02-10*, 2002.

[38] Kees van de Doel and Dinesh K Pai. Synthesis of shape dependent sounds with physical modeling. Georgia Institute of Technology, 1996.

[39] Kees van den Doel, Paul G. Kry, and Dinesh K. Pai. Foleyautomatic: Physically-based sound effects for interactive simulation and animation. University of British Columbia, 1996.

[40] Kevin JP Woods, Max H Siegel, James Traer, and Josh H McDermott. Headphone screening to facilitate web-based auditory experiments. *Attention, Perception, & Psychophysics*, 79(7):2064–2072, 2017.

[41] Changxi Zheng and Doug L James. Toward high-quality modal contact sound. In *ACM Transactions on Graphics (TOG)*, volume 30, page 38. ACM, 2011.

MODELLING EXPERTS' DECISIONS ON ASSIGNING NARRATIVE IMPORTANCES OF OBJECTS IN A RADIO DRAMA MIX

Emmanouil Theofanis Chourdakis *

Queen Mary University of London
London, UK
e.t.chourdakis@qmul.ac.uk

Lauren Ward †

University of Salford
Salford, Manchester, UK
l.ward7@edu.salford.ac.uk

Matthew Paradis

BBC Research & Development
London, UK
matthew.paradis@bbc.co.uk

Joshua D. Reiss

Queen Mary University of London
London, UK
joshua.reiss@qmul.ac.uk

ABSTRACT

There is an increasing number of consumers of broadcast audio who suffer from a degree of hearing impairment. One of the methods developed for tackling this issue consists of creating customizable object-based audio mixes where users can attenuate parts of the mix using a simple complexity parameter. The method relies on the mixing engineer classifying audio objects in the mix according to their narrative importance.

This paper focuses on automating this process. Individual tracks are classified based on their music, speech, or sound effect content. Then the decisions for assigning narrative importance to each segment of a radio drama mix are modelled using mixture distributions. Finally, the learned decisions and resultant mixes are evaluated using the Short Term Objective Intelligibility, with reference to the narrative importance selections made by the original producer. This approach has applications for providing customizable mixes for legacy content, or automatically generated media content where the engineer is not able to intervene.

1. INTRODUCTION

Hearing loss is estimated to affect one in six people in the United Kingdom (UK) and North America [1, 2]. This figure is likely to rise given an aging demographic and the prevalence of age-related hearing loss [3]. Further to this, 2017 audience statistics indicate that those over 50 years old in the United States of America and those over 55 in the UK watch more television on average than any other age demographic [4, 5]. Therefore listeners with some degree of hearing loss make up an increasing proportion of television audiences.

Object-based audio offers the potential for personalizable content, and may significantly improve the broadcast experience for this segment of television audiences. Complex auditory scenes, such as those found in television, radio dramas and other genres, contain some objects that are essential to narrative comprehension (e.g., dialogue and certain sound effects), and others that facilitate increased immersion (e.g., background sounds and reverberation). For people with some hearing loss, the latter sounds can impair their comprehension of the narrative [6].

In order to mitigate the issue, an approach to accesible audio has been developed which allows users to control the complexity of those scenes based on their listening needs, using a single dial control [7, 8, 9]. This is achieved by the mix engineer assigning each audio object a rank based on its narrative importance (NI) so that each track can then be amplified or attenuated according to those assignments, as well as the user's desired complexity. This method is currently limited to newly-authored object-based content or old content remixed into an object-based format. For legacy content, this manual process is arduous and prohibitively costly.

This paper investigates an approach to alleviate the issue. It first models the decision processes of mixing engineers when assigning narrative importances to a track in a radio programme mix, then recreates those decisions in unseen tracks and mixes. To do this, features that inform decisions are identified, methods for extracting those features are developed and mixture models for assigning narrative importances are used together with the audio effect developed in [7, 8, 9]. The efficacy of this model is then evaluated using an automatic speech intelligibility metric, the Short Term Objective Intelligibility (STOI) criterion [10]. This provides a first indication of the method's merit. The main contributions of this paper are twofold: we provide a model for discriminating between music/speech and sound effects, and we demonstrate a method for learning to automatically assign narrative importances to older mixes without such metadata. The latter may underpin future tools which enable hard of hearing users greater access to the large amount of legacy content.

2. PREVIOUS WORK

Personalising the balance of audio elements at the user end utilising object-based audio methods has been explored for both normal hearing listeners in noise [11, 12] and hard of hearing listeners [13, 14]. At its most simple, this personalisation provides the end-user with the ability to control the balance between background

* This work was supported by the EPSRC Programme Grant S3A: Future Spatial Audio for an Immersive Listener Experience at Home (EP/L000539/1) and the BBC as part of the BBC Audio Research Partnership.

† Lauren Ward is funded by the General Sir John Monash Foundation.

and foreground elements [11, 12]. A more nuanced approach explored for hard of hearing listeners gives end-users the ability to control four categories of sound: dialogue, foreground sound effects, background sound effects, and music [13]. Whilst feedback for such an approach was overwhelmingly positive, it lacked the ease of use required for large scale adoption.

A single dial control based on narrative importance metadata has been developed to combine powerful user personalisation with ease of access [7, 8, 9]. NI metadata categorises the audio objects within a soundtrack hierarchially, based on the role each sound plays in conveying the narrative. Each object is assigned an NI value in metadata between 0 (essential) and 3 (least important). Metadata is currently generated and auditioned by the producer in an audio effect plugin [9], in order to ensure that the producer's intent for the content is maintained. Gain adjustments are then applied to each sound category based on the level selected by the user on a single dial control. The control transitions smoothly between a fully immersive mix and a mix containing only the narratively important elements. This effectively allows users to adjust the complexity of the reproduced audio mix based on their needs, whilst ensuring comprehension of the narrative is always maintained. Full details of this implementation can be found in [7]. Early work on this has shown qualitative improvements in intelligibility for hard of hearing listeners whilst maintaining the creative integrity of the producer's work.

Ranking an object according to its NI can be seen as a type of automatic mixing based on gain adjustment, where a gain function of the user's preferences is chosen for each track based on its NI. Automatic gain adjustment works have existed since 1975, initially just for speech [15], and more recently in the generic context of music production [16, 17]. Here the authors optimized gains for ratios of loudness between different tracks in a multitrack live music mix. Our work differs in that we consider that individual track gains have been chosen, e.g. by one of the cited methods or a mixing engineer, and then we apply an additional post-fader gain which is a function of an individual user's preference. A similar approach, which takes individual preference into consideration can be seen in [18] where the proposed method allowed users to adapt the behaviour of a dynamic range compressor to their listening conditions.

Our method learns and models the choices of mixing engineers when ranking an audio object based on importance, as well as important features that can characterize such decisions. The latter is similar to work in [19] where the authors included important musical features as well as domain expert rules for guiding music production decisions using a probabilistic expert system. Finally, we evaluate using the Short Term Objective Intelligibility criterion [10].This metric only indicates objective intelligibility of the resulting mix, rather than the subjective comprehension of the content. However it yields an initial indication of the efficacy of the approach and whether subsequent subjective testing is warranted. A relevant work which used the same criterion to control a dynamic range compressor can be found in [20].

3. METHODOLOGY

Our goal is to assign a narrative importance d to an object based on various features extracted from the object and its role in the mix. We approached the issue as a classification task where the training data comes from a web audio listening experiment where mixing experts assigned NI values to audio objects in a radio drama.

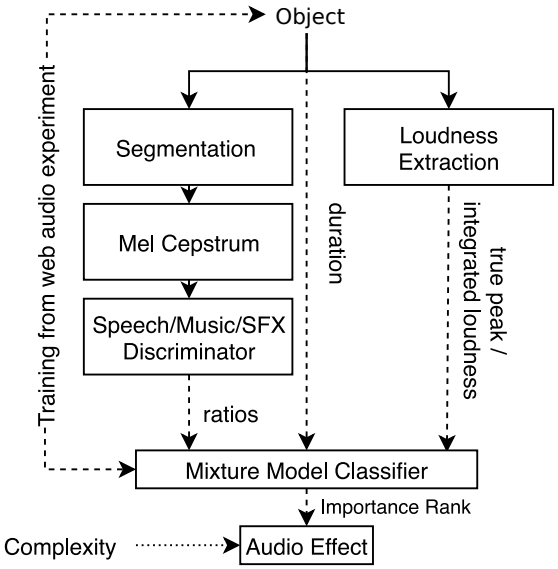


Figure 1: The outline of the process. An object from a radio mix is split into segments and each segment is classified as music/speech or sound effects. The duration of the object and its loudness characteristics are also extracted. Data coming from experiments is used to train a mixture model classifier to assign importances to similar objects. Finally, an audio plugin assigns this importance's gain level based on narrative complexity chosen by the user.

Given similar content, the goal is to make similar decisions. Below we describe each session in developing of the effect an outline of which can be seen in Figure 1.

3.1. Data Acquisition

The data used in this experiment was collected from 34 individuals who identified as audio production or mixing professionals. The majority of participants worked in television production (44%) followed by radio (35%) and film (27%). Most participants work freelance (53%) or for a national broadcaster (35%). Documentary, drama and music were the most common genres they worked in and most identified as a dubbing mixer or sound mixer (58%). 41% of respondents had worked on an object-based production before, and a further 32% were familiar with the concept of object-based audio. On average they had 22.7 years experience (median 21.5 years).

They completed the online task over their own headphones. First they were asked to listen to the radio drama *The Turning Forest* in its entirety [21]. Then they were given a segment of the drama (100s in duration) for which they were asked to assign narrative importance for its constituent 23 audio objects. The web audio interface allowed them to audition their choices at the extremes of the complexity control (fully immersive and narrative only) as well as at the 50% point to ensure their decisions produced mixes they were happy with.

In addition to this, a workshop was undertaken with the original sound designer using an early iteration of the narrative impor-

Layer Type	Layer Shape	Activation function
Convolutional	$64 \times 96 \times 64$	$relu$
Max pooling	$64 \times 48 \times 32$	–
Convolutional	$128 \times 48 \times 32$	$relu$
Max pooling	$128 \times 24 \times 16$	–
Convolutional	$256 \times 24 \times 16$	$relu$
Convolutional	$256 \times 24 \times 16$	$relu$
Max pooling	$256 \times 12 \times 8$	–
Convolutional	$512 \times 12 \times 8$	$relu$
Convolutional	$512 \times 12 \times 8$	$relu$
Max pooling	$512 \times 6 \times 4$	–
Fully-connected	256	$relu$
Fully-connected	3	$softmax$

Table 1: Shapes of the layers of the network. The network is represented with the top layer being the input layers and the bottom giving the output. The first dimension of the convolutional layers refers to the number of extracted features from that layer, and the next two to the shape of those filters. With $softmax$ we denote the softmax function which converts the output of the layer to a discrete probability distribution and with $relu$ the rectified linear unit which allows the network to model non-linearities. Since we do transfer learning, we only train the last two layers.

tance metadata acquisition tool. The sound designer was encouraged to develop her own workflow for authoring the metadata for the entirety of the *The Turning Forest*. Metadata changes could be auditioned by the sound designer in real-time using the full range of the NI control interface. Whilst an objective *ground truth* for the narrative importance assignments is not possible as it is inherently subjective, the assignments by the original producer provide the point of reference for this investigation.

3.2. Features informing decisions

Observing the decisions made by the mixing engineers, an initial hypothesis could be formed based on the type of content of the individual objects. We mainly deal with 3 classes of content; speech, music, and sound effects [22]. We developed a Convolutional Neural Network (CNN) for classification to the above three classes. CNNs have been successfully used for fast classification of images, video, or spectral representations of audio since they have many fewer parameters than fully connected neural networks and can thus be trained much faster [23]. For the task of classification, they are usually constructed using building blocks called "Convolutional Layers" which extract useful features from an image-like input, "pooling" layers which select part of the resulting features, "fully-connected" layers which combine those features, and a final classification layer [23, 24]. Such networks have previously been used to successfully distinguish between speech and music [25]. To develop our network, we used VGGish [26] as a starting point and we applied transfer learning to make it classify between speech, music, and sound effects. Transfer learning is a technique where a network trained for a task can be trained for a different task with minimal computational effort [24]. VGGish is a CNN originally trained to distinguish between 632 classes found in AudioSet [27] which is a dataset consisting of the soundtracks of 8 million Youtube videos. Since speech, music, and sound effects are among those tracks, we can achieve good performance in discriminating between those three "superclasses" by retraining the model to only discriminate between those. We therefore train the model by keeping its convolutional layers with their AudioSet-trained weights intact, since this is the part of the network that does feature extraction, and replacing the fully connected layers with a layer of size 256. Finally, we add a classification fully connected layer of 3 classes with the softmax activation function which converts the output of that layer to probabilities of the input being in one of the three classes. The shapes of the individual layers are listed in Table 1.

Inputs to the CNN are fed into the top convolutional layer in Table 1. Each audio object's track is split into non-overlapping segments of 960ms where each segment consists of the magnitudes of 64 bands of the mel cepstrum computed using a frame size of 25ms and a hop size of 10ms. Training was done by freezing the weights of the convolutional layers and only training the last two feed-forward layers. We used the GTZAN music/speech discrimination dataset[1] which contains 120 tracks with 30 minutes of speech, and 30 minutes of music as 22kHz 16bit audio files to adapt the new model to our task. In addition, we added 30 minutes of randomly sampled sound effects from the recently released online BBC SFx library[2] reformatted to match the examples in the other two classes. To make sure a specific class of sound effects is not over-represented, we used stratified sampling to select the sound effects by sampling first the class of sound effects, and then the sound effect audio file. After training using the augmented dataset we have a model that can classify the content of the object into music, speech, or sound effects and use this classification as a feature which informs the NI assignment.

Track loudness and duration is also measured. We expect that important sound effects which require the attention of the listener to have a high peak-to-integrated loudness ratio [22] as well as short duration. For example the clinking sound of two glasses toasting will signify a more important effect than the sound of frogs croaking repeatedly in the background. For this reason we use both peak-to-integrated-loudness ratio and total duration as features.

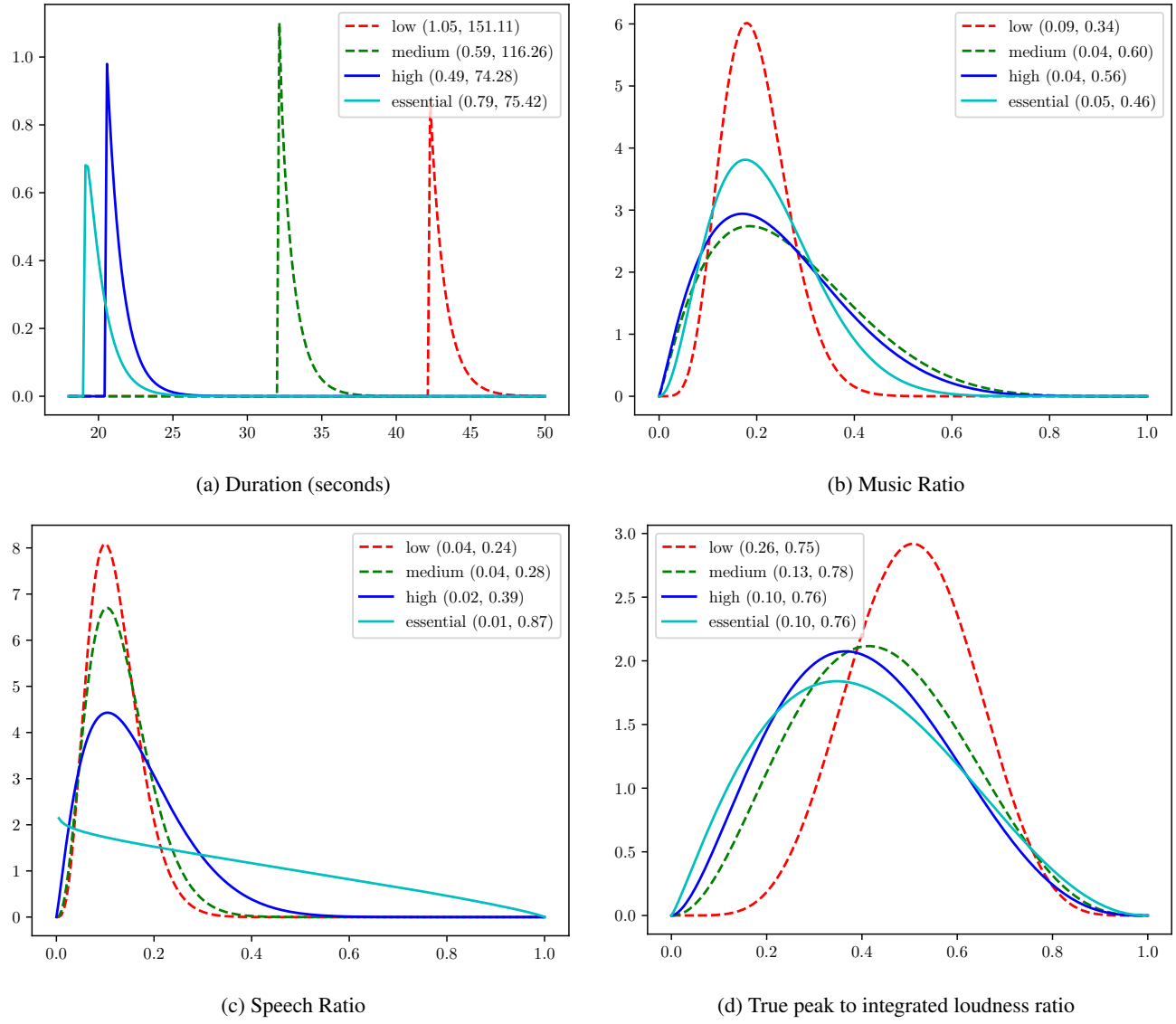
3.3. Decision Modeling

Our goal was to create a model based on the decisions from Section 3.1. An inherent challenge in the data described in Section 3.1 is that the practitioners would disagree quite a lot when ranking objects according to their importances. To quantify this disagreement, we can use Fleiss' kappa [28]:

$$\kappa = \frac{\hat{P} - \hat{P}_c}{1 - \hat{P}_c} \tag{1}$$

$\hat{P}$ is the degree of agreement between raters and $\hat{P}_c$ the degree of agreement attributed to chance. It is defined in the interval $[0, 1]$ where $\kappa = 1$ signifies total agreement. We found $\kappa = 0.008$ which denotes a low degree of agreement. Despite the low level of agreement, from Figure 3 we observe that for most objects, importance assignments are concentrated around 2 neighbouring values. This can be particularly observed for non-narration objects containing speech (*Girl Voice*, *Boy Voice*)). We decided to use a mixture model to treat this uncertainty as stochasticity in the model's decisions. In order to use such a model, we need to determine appropriate features that can give correct decisions and define their

[1] `http://marsyas.info/downloads/datasets.html`
[2] `http://bbcsfx.acropolis.org.uk/`



(a) Duration (seconds)

(b) Music Ratio

(c) Speech Ratio

(d) True peak to integrated loudness ratio

Figure 2: Estimated densities for four different narrative importance levels. In parentheses are the 95% lowest and highest respectively confidence intervals.

probability densities. From the features selected in Section 3.2, we found that ratio of speech ($p \ll 0.05$) and music ($p < 0.05$) in an object, true-peak-to-integrated-loudness ratio ($p \ll 0.05$) and total duration ($p \ll 0.05$) are good features. If we represent the values of the features above as x_{sr}, x_{mr}, x_{tpti}, and x_{dur}, the goal of decision modelling is to decide an importance d given those values. If we furthermore assume that each feature is a sample from a respective independent feature distribution, this decision can be given as:

$$\begin{aligned} d &= \arg\max_i \Pr\left(I = i | x_{sr,i}, ..., x_{dur,i}, \boldsymbol{\theta}_{sr,i}, ..., \boldsymbol{\theta}_{dur,i}\right) \\ &= \arg\max_i \Pr\left(I = i\right) \prod_{\mu \in \{sr,mr,tpti,dur\}} \Pr\left(x_{\mu,i}, \boldsymbol{\theta}_{\mu,i} | I = i\right) \\ &= \arg\max_i \Pr\left(I = i\right) \prod_{\mu \in \{sr,mr,tpti,dur\}} \Pr\left(x_{\mu,i} | I = i, \boldsymbol{\theta}_{\mu,i}\right) \\ &\quad \cdot \Pr\left(\boldsymbol{\theta}_{\mu,i} | I = i\right) \end{aligned} \tag{2}$$

Where $\boldsymbol{\theta}_\mu$ is the parameter vector for the distribution that corresponds to $x_{\mu,i}$ and i is a level of narrative importance ($essential$, $high$, $medium$, low). Music x_{mr}, speech x_{sr}, and true-peak-to-loudness x_{tpti} ratios are defined in the interval $[0, 1]$ and thus

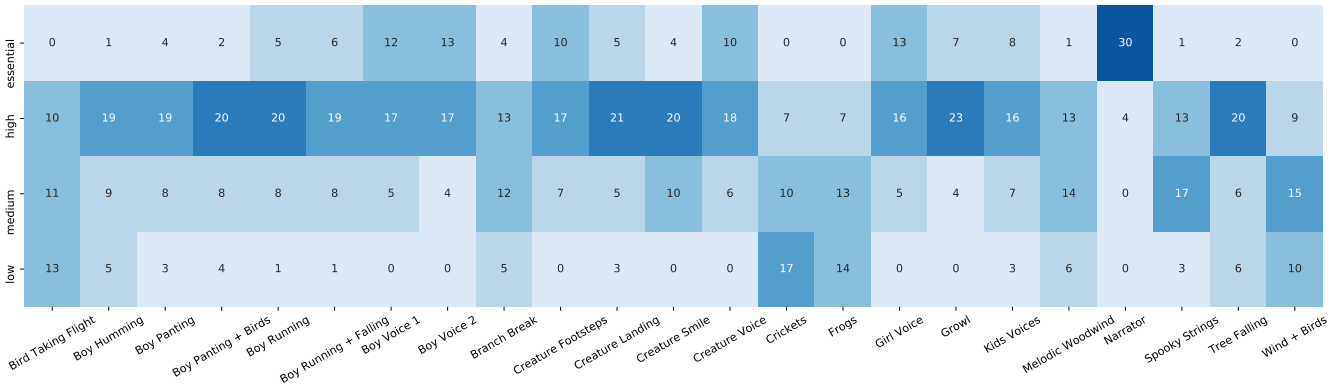


Figure 3: Heatmap of importance assignment decisions made by the experts for *The Turning Forest* radio drama. Horizontal axis shows the object names and vertical axis the assigned importances. The numbers inside individual cells correspond to the frequency of each importance value assigned to each object where darker cells correspond to higher frequencies. We observe that for most objects, there are 1 or 2 "most preferred" importance assignments.

make Beta distributions suitable for modelling their values. On the other hand the Gamma distribution is suitable for modelling total duration x_{dur} since it is defined in positive numbers. Both Beta and Gamma distributions are defined by two parameters α and β. Finally, the prior distribution of importances $\Pr(I = i)$ can be modelled as a categorical distribution, since it can take one of 4 distinct values:

$$x_{\nu,i} \sim \text{Beta}(\alpha_{\nu,i}, \beta_{\nu,i}), \quad \nu \in \{sr, mr, tpti\} \tag{3}$$
$$x_{dur,i} \sim \text{Gamma}(\alpha_{dur,i}, \beta_{dur,i}) \tag{4}$$
$$i \sim \text{Categorical}(4) \tag{5}$$

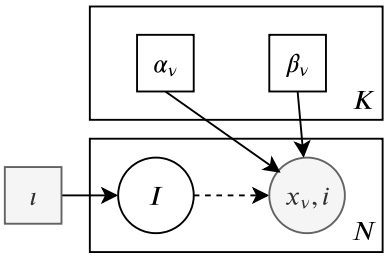


Figure 4: The mixture model used. In the diagram above, circles represent random variables, squares parameters of those variables. Shaded shapes represent observed variables. $K = 4$ represents the number of mixtures which is the same as the assigned importances, and N is the number of samples in the training data. The goal of the estimation process is to estimate the parameters represented in the non-shaded boxes (parameters of feature densities) given the observations represented in the shaded boxes and given from a training dataset (importance assignment and feature values).

The model for each feature can be seen as a diagram in Figure 4 and its distributions in Figure 2. Finally, the Beta and Gamma distributions in Eq. 3 are defined according to:

$$\text{Gamma}(\alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}, x > 0 \tag{6}$$
$$\text{Beta}(\alpha, \beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}\Gamma(\alpha+\beta)}{\Gamma(\alpha)\cdot\Gamma(\beta)}, 0 \leq x \leq 1 \tag{7}$$
$$\Gamma(t) = \int_0^{+\infty} dx \cdot x^{t-1} e^{-x} \tag{8}$$

What the above essentially mean is that when we know the importance of an object (differently stylized lines in Figure 2) we expect the values of the features given in Section3.2 to be samples from the distributions given in Eqs. 6, and 7. What is left is to decide on the exact shapes of those distributions, which are defined by parameters α and β above. We estimate those by using a training set of observations of importances and corresponding features and using Stochastic Variational Inference [29]. After having defined the model that model the values of the object features, the decision can be taken as in Eq. 2 where in this case:

$$\boldsymbol{\theta}_{\mu,i} = \begin{bmatrix} \alpha_{\mu,i} \\ \beta_{\mu,i} \end{bmatrix} \tag{9}$$

In this case, Eq. 2 gives a decision on an importance level d that maximizes the probability that an object belongs to that importance level given the values of its features.

3.4. Applying gains

The importance assignment process in the previous sections controls the audio effect described in [7]. This is a mixing effect with 4 stereo inputs and 2 stereo outputs:

$$\boldsymbol{y}_n = \underbrace{\begin{bmatrix} 1 & \dots & 0 \\ 0 & \dots & 1 \end{bmatrix}}_{\text{Downmixes to 2 channels}} \begin{bmatrix} \boldsymbol{I}_3(c) & \dots & & 0 \\ \vdots & 1 & & \vdots \\ & & \boldsymbol{I}_{-12}(c) & \\ 0 & & \dots & \boldsymbol{I}_{-48}(c) \end{bmatrix} \boldsymbol{x}_n \tag{10}$$

where $\boldsymbol{x}_n$ is the 8 channel input at time n corresponding to the inputs at the 4 importance levels, $\boldsymbol{y}_n$ the corresponding output, $\boldsymbol{I}_3$, 1, $\boldsymbol{I}_{-12}$, and $\boldsymbol{I}_{-48}$ are the mixing coefficients corresponding to

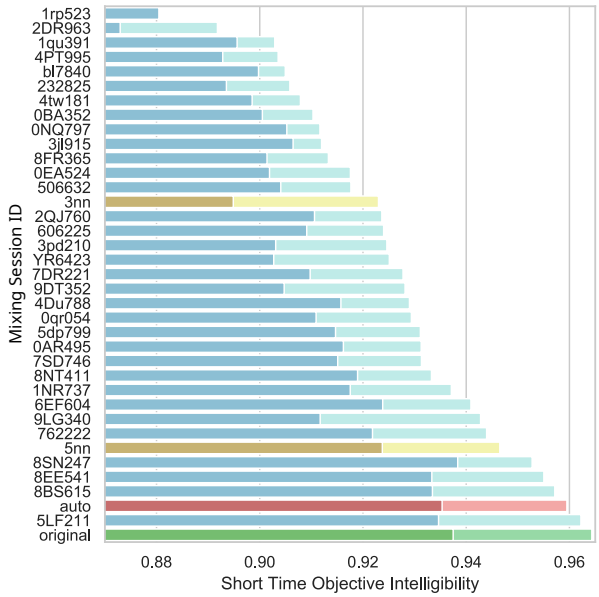


Figure 5: Results using the Short Term Objective Intelligibility measure. With lighter colors a complexity of 0 has been selected and with darker colors a complexity of 50. STOI values of the mixing sessions of the practitioners (Section 3.1) are shown as cyan colored bars. STOI of automatic assignment of narrative importances is labeled as 'auto', and STOI of the mix done by the author of the radio drama as 'original'. STOI values for decisions made using K-Nearest Neighbour classifiers for $K = 3$ and 5 are also included as '3nn' and '5nn'.

essential, *high*, *medium*, and *low* respectively (the subscripts are the gain values for each importance level in dB):

$$\boldsymbol{I}_M = \begin{bmatrix} 10^{M\left(\frac{-c}{2000}+20\right)} \\ 10^{M\left(\frac{-c}{2000}+20\right)} \end{bmatrix} \tag{11}$$

where c is a complexity number between 0 and 100 chosen by the user during playback.

4. RESULTS

For the discriminator model, 70% of the data was used as training/validation and 30% was kept aside for testing. The split was performed with a predefined random seed to guarantee reproducibility. For evaluating the model, we calculated precision p, Recall r, and f_1 on the test set:

$$p = \frac{tp}{tp+fp} \tag{12}$$

$$r = \frac{tp}{tp+fn} \tag{13}$$

$$f_1 = 2\frac{p \cdot r}{p+r} \tag{14}$$

where tp is the number of true positives, fp the number of false positives and fn the number of false negatives. The calculated metrics can be seen in Table 2.

We tested how the importance assignment model works compared to the data acquired from practitioners (Section 3.1) as well as the original radio drama author. We also included a K-Nearest neighbour classifier with $K \in \{3, 5\}$. We evaluated according to the Short Term Intelligibility criterion [10] and more specifically the PySTOI implementation[3]. In Figure 5 we can see that when fully attenuating non-essential narrative elements our model outperformed all but one of the mixings done using the online platform (0.959 vs 0.962) and scored close to the mix by the original author (0.964). A more interesting result is when choosing a complexity value of 50, which keeps some less-important narrative elements as well, the STOI is equal to the original author's mix (0.937), even if the latter was not included in the training set. This suggests that our classifier managed to model "good" decisions from the practitioners despite the high level of disagreement. In comparison, the K-NN classifiers which do not account for uncertainty performed worse, although the classifier using the 5 nearest neighbours was still ranked above the third quartile regarding STOI.

class	precision	recall	f_1
music	1.00	0.97	0.99
speech	0.98	0.98	0.98
sfx	0.96	0.99	0.98

Table 2: Results on the test set for the music/speech/sfx discriminator model

5. DISCUSSION

This paper presented a method for modelling decisions made by mixing engineers with the goal of allowing the listener to alter the complexity of a radio drama while retaining speech intelligibility. The method relies on modelling the mixing engineers' behaviour using mixture models even when those have a large degree of disagreement. The model was tested against decisions made by the original mixing engineer of a radio drama mix and it was found that it could perform comparatively well when evaluated with an objective intelligibility metric. In the process we developed a simple music/speech/sound effects discriminator that works well for this application and is provided freely to those interested [4] and a plugin based on the VISR [30] environment is planned in order to automate the process. The current work is limited however to a single radio drama and also to a single intelligibility metric. More metrics should be considered that also measure quality and immersion. We also assume that each object is assigned a single importance value that does not change for the duration of the drama. This is an assumption that does not necessarily hold. For example we expect the footsteps of a monster approaching the main character to have higher narrative importance than the footsteps of a monster when it is further away doing something irrelevant to the story. Using our method however those two different scenarios would be ranked the same. A simple solution to this issue employed in the current work is to manually assigns the footsteps in the two senarios in distinct objects. Further work could also consider characteristics of an object that change throughout the duration of the drama when ranking them based on importance. In this paper we also consider gains after the fader stage in the mix. Gain effects

[3] https://github.com/mpariente/pystoi.git

[4] The models and other supplementary material can be found at: https://github.com/bbc/audio-dafx2019-automatic/

that synergize with our current work in applying appropriate gains pre-fader can also be examined as well as other automated mixing techniques such as in EQ [16, 31], Compression [18], or Reverberation [32]. Finally, subjective listening tests should be undertaken such that the overall quality and comprehension of the automatically assigned mixes can be evaluated by human subjects.

6. ACKNOWLEDGMENTS

We would like to thank the reviewers for their suggestions in clarifying parts of this paper. We would also like to thank Andrew Mason for his advice on measuring loudness and also his help in proofreading the final draft of the paper. This work was supported by BBC R&D as part of BBC Audio Research Partnership. Lauren Ward is supported by the General Sir John Monash Foundation.

7. REFERENCES

[1] Action on Hearing Loss, "Hearing Matters Report," November 2015.

[2] Y. Agrawal, E. A. Platz, and J. K. Niparko, "Prevalence of hearing loss and differences by demographic characteristics among US adults: data from the National Health and Nutrition Examination Survey, 1999-2004," *Archives of internal medicine*, vol. 168, no. 14, pp. 1522–1530, July 2008.

[3] Office for National Statistics, "National population projections: 2014-based statistical bulletin," October, 2015.

[4] The Nielsen Company (US), "The total audience report q1, 2017.," November 2017.

[5] Broadcasters Audience Research Board, "Trends in television viewing 2017," February 2018.

[6] O. Strelcyk and G. Singh, "Tv listening and hearing aids," *PloS one*, vol. 13, no. 6, June 2018.

[7] L. Ward, B. Shirley, and J. Francombe, "Accessible object-based audio using hierarchical narrative importance metadata," in *Audio Engineering Society Convention 145*, New York, USA, October 2018.

[8] L. A. Ward, "Accessible broadcast audio personalisation for hard of hearing listeners," in *Adjunct Publications of the 2017 ACM International Conference on Interactive Experiences for TV and Online Video*, Hilversum, Netherlands, June 2017, pp. 105–108.

[9] B. Shirley, L. A. Ward, and E. T. Chourdakis, "Personalization of object-based audio for accessibility usingnarrative importance.," in *ACM International Conference on Interactive Experiences for Television and Online Video, Workshop on In-Programme Personalisation*, Manchester, UK, June 2019.

[10] C. H. Taal et al., "A short-time objective intelligibility measure for time-frequency weighted noisy speech," in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, Dallas, USA, March 2010, pp. 4214–4217.

[11] T. Walton, M. Evans, D. Kirk, and F. Melchior, "Exploring object-based content adaptation for mobile audio," *Personal Ubiquitous Comput.*, vol. 22, pp. 707–720, August 2018.

[12] W. Bleisteiner et al., "D5.6: Report on audio subjective tests and user tests," July 2018.

[13] B. Shirley, M. Meadows, F. Malak, J.S. Woodcock, and A. Tidball, "Personalized object-based audio for hearing impaired tv viewers," *J. Audio Eng Soc.*, vol. 65, no. 4, pp. 293–303, April 2017.

[14] H. Fuchs and D. Oetting, "Advanced clean audio solution: Dialogue enhancement," *SMPTE Motion Imaging J.*, vol. 123, no. 5, pp. 23–27, July 2014.

[15] D. Dugan, "Automatic microphone mixing," *J. Audio Eng Soc.*, vol. 23, no. 6, pp. 442–449, August 1975.

[16] E. Perez-Gonzalez and J.D. Reiss, "Automatic gain and fader control for live mixing," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New York, USA, October 2009, pp. 1–4.

[17] D. Ward, J. D. Reiss, and C. Athwal, "Multitrack mixing using a model of loudness and partial loudness," in *Audio Engineering Society Convention 133*, October 2012.

[18] A. Mason, N. Jillings, Z. Ma, J. D. Reiss, and F. Melchior, "Adaptive audio reproduction using personalized compression," in *AES 57th Conf. on The Future of Audio Entertainment Technology*, Hollywood, California, USA, March 2015.

[19] G. Bocko, M. F. Bocko, D. Headlam, J. Lundberg, and G. Ren, "Automatic music production system employing probabilistic expert systems," in *Audio Engineering Society Convention 129*, San Fransisco, USA, November 2010.

[20] H. Schepker, J Rennies, and S. Doclo, "Speech-in-noise enhancement using amplification and dynamic range compression controlled by the speech intelligibility index," *J. Acoustical Soc. of America*, vol. 138, no. 5, pp. 2692–2706, November 2015.

[21] J. Woodcock et al., "Presenting the s3a object-based audio drama dataset," in *Audio Engineering Society Convention 140*, Paris, France, June 2016.

[22] "Guidelines for production of programmes in accordance with EBU R 128," Tech. Rep., European Broadcasting Union, January 2016.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, December 2012, pp. 1097–1105.

[24] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, MIT press, 2016.

[25] M. Papakostas and T. Giannakopoulos, "Speech-music discrimination using deep visual feature extractors," *Expert Systems with Applications*, vol. 111, pp. 334–344, December 2018.

[26] S. Hershey et al., "CNN architectures for large-scale audio classification," in *Int. Conf. on acoustics, speech and signal processing*, New Orleans, USA, March 2017, pp. 131–135.

[27] Jort F. G. et al., "Audio set: An ontology and human-labeled dataset for audio events," in *Int. Conf. on acoustics, speech and signal processing*, New Orleans, USA, March 2017.

[28] J. L. Fleiss, "Measuring nominal scale agreement among many raters.," *Psychological bulletin*, vol. 76, no. 5, pp. 378, November 1971.

[29] M. D. Hoffman et al., "Stochastic variational inference," *The Journal of Machine Learning Research*, vol. 14, no. 1, pp. 1303–1347, January 2013.

[30] A. Franck and F. M. Fazi, "VISR – a versatile open software framework for audio signal processing," in *AES International Conference on Spatial Reproduction*, Tokyo, Japan, July 2018.

[31] Y. Tang and M. Cooke, "Optimised spectral weightings for noise-dependent speech intelligibility enhancement," in *13th Annual Conference of the International Speech Communication Association*, September 2012.

[32] E. T. Chourdakis and J. D. Reiss, "A machine-learning approach to application of intelligent artificial reverberation," *Journal of the Audio Engineering Society*, vol. 65, no. 1/2, pp. 56–65, February 2017.

SPEECH DEREVERBERATION USING RECURRENT NEURAL NETWORKS

Shahan Nercessian and Alexey Lukin

iZotope, Inc.
Cambridge, MA, USA
shahan@izotope.com, alex@izotope.com

ABSTRACT

Advances in deep learning have led to novel, state-of-the-art techniques for blind source separation, particularly for the application of non-stationary noise removal from speech. In this paper, we show how a simple reformulation allows us to adapt blind source separation techniques to the problem of speech dereverberation and, accordingly, train a bidirectional recurrent neural network (BRNN) for this task. We compare the performance of the proposed neural network approach with that of a baseline dereverberation algorithm based on spectral subtraction. We find that our trained neural network quantitatively and qualitatively outperforms the baseline approach.

1. INTRODUCTION

Reverberation is an effect that can be created naturally when a source sound is reflected off various surfaces before reaching an observer (e.g. a microphone). The characteristics of this reverberation are defined by its acoustic environment (the dimensions and objects in a space, their material properties, etc.), as well as the positions of the source and observer in this environment. Though desirable in some creative and musical contexts, reverberation has the overall effect of reducing intelligibility, which may be particularly undesirable for speech applications, such as telecommunications, automated voice systems, and dialogue editing in post-production. *Dereverberation* is the process of automatically removing reverberation from audio signals. It is an extremely difficult problem, as neither the source signal nor the characteristics of the acoustic environment are known a priori. Moreover, moving sources or microphones can induce time-varying reverberation effects and further complicate its removal.

Standard algorithms for speech dereverberation either exploit properties of speech, or attempt to blindly estimate the reverberant channel [1]. In the former case, algorithms explicitly track harmonic content or leverage linear predictive coding (LPC) of speech to estimate components of the underlying direct-path speech signal [2]. In contrast, blind channel estimation methods generally involve an explicitly parameterized model of the reverberation process, techniques for estimating its parameters, and finally, a reverb removal step based on this parameter estimation through some form of inverse filtering. Moreover, when multiple microphones are available, these algorithms leverage beamforming techniques to further improve reverb cancellation [3], though this is not expected to be the case for most practical applications. The performance of existing dereverberation algorithms is limited by the flexibility of their model assumptions and the ability to accurately estimate parameters over a wide range of applicable scenarios. They are also laborious to develop and may require a fair amount of intensive hand-tuning.

Source separation techniques using deep learning have become increasingly popular, particularly for the application of separating speech from non-stationary noise [4]. Arguably the most common source separation approaches involve time-frequency masking, wherein models are trained to estimate the amount of speech and noise present in each spectrogram bin, and accordingly create a time-varying masking filter to separate speech from noise through Wiener filtering [5]. To this end, recurrent neural network (RNN) architectures have been shown to be extremely effective, as they are a natural choice for modeling sequential data [6]. Bidirectional RNNs (BRNNs) can further improve separation quality by performing a forward and backward pass over the data, thus incorporating future temporal context at the cost of offline (non-realtime) operation [7].

In this paper, we propose the adaptation of blind source separation techniques using deep learning for single-channel dereverberation. One of the main advantages of this method, in addition to its performance, is that we no longer require learning an explicit reverberation profile, but instead simply learn to distinguish dry and reverberant signal components through several synthesized examples observed during network training. With a targeted interest in dialogue editing for post-production, we deliberately limit ourselves to speech signals.

The rest of this paper is structured as follows: We review a baseline dereverberation algorithm based on spectral subtraction [8], and outline our proposed neural network approach in Section 2. We compare the performance of our neural network solution against the baseline algorithm in terms of reverberation reduction and speech intelligibility in Section 3. Finally, we draw conclusions and allude to future work in Section 4.

2. DEREVERBERATION ALGORITHMS

We consider monaural speech signals and model their reverberation by the convolution $y(t) = h(t) * s(t)$, where $y(t)$ is the observed reverberated signal, $h(t)$ is the impulse response of the acoustic environment, factoring the position of the source and observer in said environment, and $s(t)$ is the direct-path speech signal. The dereverberation process is a blind deconvolution in which we attempt to find an estimate $\hat{s}(t)$ from $y(t)$. Rather than operating on the time-domain waveform, both the baseline and proposed neural network solutions operate primarily on the short-time Fourier transform (STFT) magnitude spectrogram of $y(t)$, denoted as $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_T] \in \mathbb{R}^{d \times T}$, where d is the number of frequency bins and T is the number of STFT time frames.

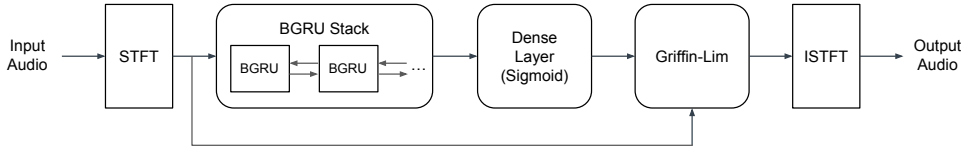


Figure 1: *Speech dereverberation neural network architecture.*

2.1. Baseline Algorithm

The baseline dereverberation algorithm uses short-time spectral attenuation with directly calculated spectral masks. In this case, the multi-path signal is modeled as a first-order recursive filter that "smears" the spectrogram in time, defined as

$$\mathbf{r}_t = \alpha \mathbf{s}_t + (1-\alpha)\mathbf{r}_{t-1}, \tag{1}$$

where t is the STFT frame index, $\mathbf{s}_t$ is the magnitude spectrum of the direct path signal, $\mathbf{r}_t$ is the magnitude spectrum of the multi-path signal, and α is a coefficient related to the reverberation time (RT_{60}). This implies an exponential model for the reverberation process, which is considered to be an adequate assumption for a large number of reverberation types. The observed reverberant signal $\mathbf{y}_t$ at time t is a mixture of the direct and multi-path signals defined as

$$\mathbf{y}_t = \mathbf{s}_t + \beta \mathbf{r}_t, \tag{2}$$

where β is the wet-to-dry ratio coefficient controlling the relative amount of reverberation. Assuming that the reverberation parameters α and β are known, we can easily invert (1) and (2) to compute an estimate of the dry signal spectral magnitude $\hat{\mathbf{s}}_t$ from $\mathbf{y}_t$. The obtained signal-to-noise ratio $\hat{\mathbf{s}}_t/\mathbf{y}_t$ can be used in a spectral attenuation algorithm with extra time-frequency smoothing for reduction of "musical noise" artifacts [8]. In our implementation, α is frequency-independent, while β is independently estimated in 4 frequency bands. Parameter estimation is an offline process which analyzes an entire audio waveform. The parameter α is estimated from a histogram of spectral decay rates, while β is estimated in each frequency band such that the resulting spectral subtraction maximally reduces the energy of $\mathbf{y}_t$ while remaining non-negative.

2.2. Neural Network Algorithm

For any practical application, we can consider impulse responses $h(t)$ normalized such that $h(0) = 1$. This allows us to reformulate the reverberation process as

$$y(t) = s(t) + h_0(t) * s(t) = s(t) + n(t), \tag{3}$$

where

$$h_0(t) = \begin{cases} 0, & t = 0, \\ h(t), & \text{otherwise} \end{cases} \tag{4}$$

and $n(t)$ is the multi-path signal. As such, we have converted the problem of dereverberation into one of blind source separation, and can leverage advances in deep learning which we have successfully used to this end [7, 9]. Note that, in contrast to (2), the frequency-dependent wet-to-dry ratio β is simply embedded in $n(t)$ and its respective magnitude spectrum $\mathbf{n}_t$ (e.g. $\mathbf{n}_t = \beta \mathbf{r}_t$). In general, one of the attractive features of a deep learning-based approach is that we can forego the need to define an explicit parameterization of the reverberation process, easing design and giving flexibility to perform well over a wider range of possible reverberation types.

Given a training set with examples of isolated direct-path and multi-path speech signals, we create mixtures with known ground truth to learn a mapping that estimates the direct-path signal $\hat{s}(t)$ from the reverberated mixture $y(t)$. We use the magnitude ratio mask as a time-varying filter for separating dry and reverberated speech, which is defined as

$$\mathbf{m}_t = \frac{\mathbf{s}_t}{\mathbf{s}_t + \mathbf{n}_t}, \tag{5}$$

where the division operation in (5) is performed element-wise. Because magnitude spectra $\mathbf{s}_t$ and $\mathbf{n}_t$ are nonnegative, the mask elements $\mathbf{m}_t$ are in the interval $[0, 1]$. The output of our neural network is $\hat{\mathbf{m}}_t$, which we use to obtain estimated magnitude spectra for separated direct-path and multi-path speech, i.e.,

$$\hat{\mathbf{s}}_t = \hat{\mathbf{m}}_t \odot \mathbf{y}_t, \tag{6}$$

$$\hat{\mathbf{n}}_t = (\mathbf{1} - \hat{\mathbf{m}}_t) \odot \mathbf{y}_t, \tag{7}$$

where $\odot$ represents an element-wise product. For inference, we use (6) to obtain the estimated time-domain waveform $\hat{s}(t)$ through the inverse STFT. This usually involves using phase information taken from the noisy mixture $y(t)$, which can introduce some noticeable artifacts. To improve upon this, we use the mixture phase as our initial estimate of the direct-path phase, and apply a few iterations of the Griffin-Lim algorithm [10]. Though not strictly necessary for inference (unless we, for some reason, want to retain a portion of the multi-path signal in the processed output), we still make use of (7) for network training.

We estimate $\hat{\mathbf{m}}_t$ using a bidirectional recurrent neural network architecture as depicted in Figure 1. At 48 kHz, spectrograms are computed with Hann-windowed FFTs of size 2048 and a stride of 512. A stack of bidirectional gated recurrent units (BGRU) [11] take the reverberated mixture spectrogram as input, and produce outputs which incorporate temporal context from both past and future spectrogram frames. We opt for a non-causal architecture because their lookahead capabilties allow them to perform better than their causal counterparts, and because the baseline approach already required an initial offline learning pass. The output of the BGRU stack is projected to the appropriate number of frequency bins by means of a dense layer, whose sigmoid activation ensures that spectral masks are in the desired [0, 1] range. Several studies on neural network-based speech separation have shown the utility of using the error in the estimated spectrum $\hat{\mathbf{s}}_t$ (as opposed to the error in the estimated mask $\hat{\mathbf{m}}_t$) as the network training objective [4, 5, 12]. To this end, a common objective function for source separation compares $\hat{\mathbf{s}}_t$ and $\mathbf{s}_t$ in a mean-squared sense. As

Table 1: *(*SDR/SI-SNR*) metrics as a function of SNR.*

Method	−5dB	0dB	5dB	10dB	15dB
Mixture	−4.2/ − 5.3	0.4/ − 0.2	5.4/4.9	10.3/ 9.9	15.3/15.0
Baseline	−2.3/ − 3.6	2.6/ 1.7	7.0/6.2	11.1/10.4	15.7/15.1
Proposed	**1.3/ − 0.4**	**5.6/ 4.5**	**9.6/8.7**	**13.7/13.1**	**17.6/17.1**
Oracle	4.8/ 3.1	8.1/6.9	11.6/10.7	15.3/14.5	19.1/18.5

Table 2: ΔSTOI *metric as a function of SNR.*

Method	−5dB	0dB	5dB	10dB	15dB
Baseline	4.5	2.9	0.6	0.4	0.3
Proposed	**16.2**	**9.3**	**3.8**	**1.5**	**0.5**
Oracle	38.6	17.9	7.3	2.8	1.1

in [4, 7], we use the modified mean-squared error function

$$J = \frac{1}{T}\sum_{t=1}^{T}\Big(||\hat{\mathbf{s}}_t - \mathbf{s}_t||_2^2 + ||\hat{\mathbf{n}}_t - \mathbf{n}_t||_2^2 - \gamma||\mathbf{s}_t - \hat{\mathbf{n}}_t||_2^2 - \gamma||\mathbf{n}_t - \hat{\mathbf{s}}_t||_2^2\Big), \quad (8)$$

where the parameter γ provides a trade-off between interference and artifacts caused by the source separation process. We found that the addition of cross-term penalties helped to improve dereverberation performance at lower wet-to-dry ratios.

3. EXPERIMENTAL RESULTS

3.1. Dataset description

While there are several speech datasets available for machine learning research, most of them are band-limited (usually sampled at 16 kHz), and are insufficient for the full audio rate processing needs of post-production. With a target sampling rate of 48 kHz in mind, we have opted to use speech from the pitch tracking corpus in [13], the processed speech from the DAPS experiments [14], and the TSP speech dataset [15]. We have also supplemented our clean speech training with several hours of audio from iZotope tutorial videos. While these are not truly anechoic speech recordings, they were found to be representative enough to serve as "ideal" dereverberated outputs of our system. In addition to the speech dataset, we have gathered reverb impulse responses (RIRs) from many open sources. We have also developed a RIR generator factoring different reverb characteristics (RT_{60}, wet-to-dry ratio, etc.), and added hundreds of simulated RIRs to our dataset. We considered reverberation types and parameter ranges that resemble the naturally occurring environments that the system was targeted for.

3.2. Performance assessment

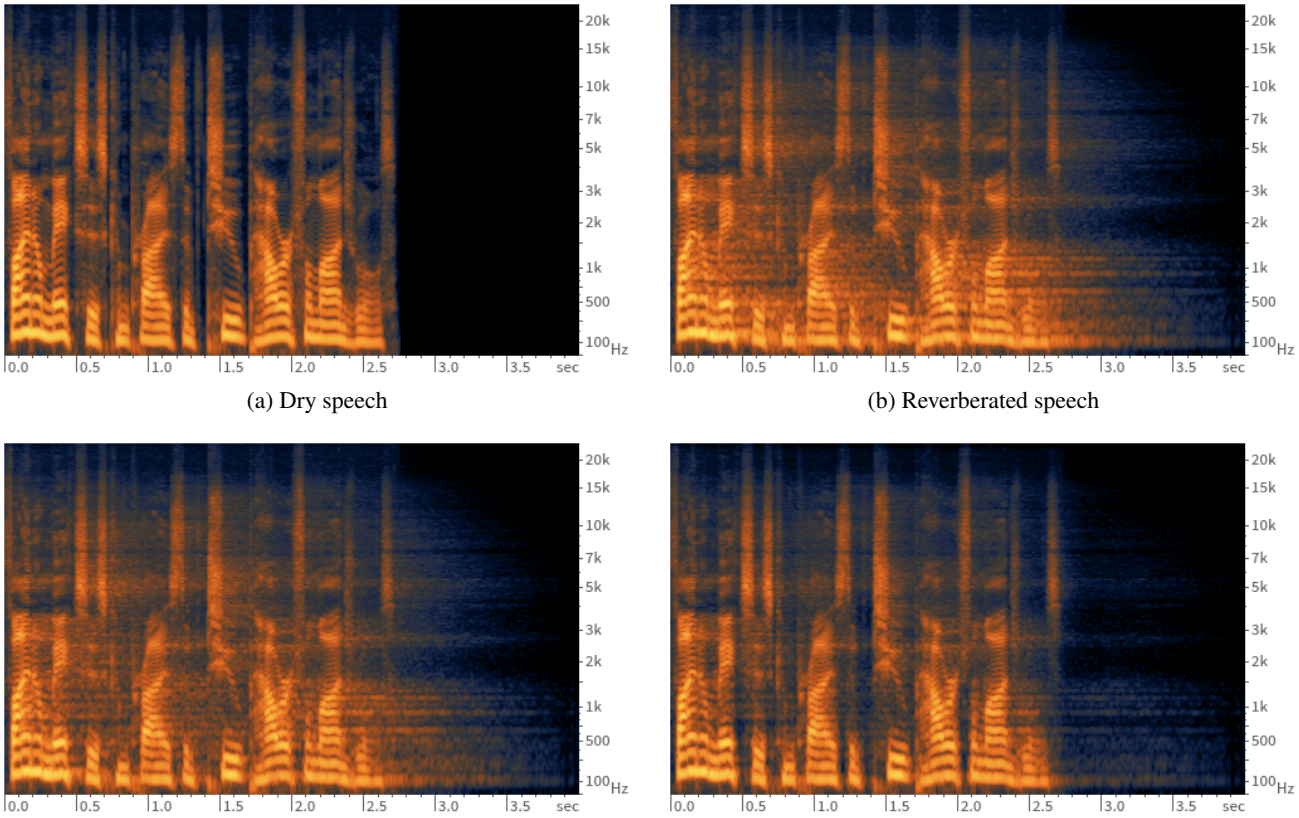
We quantitatively compared the performance of the baseline and our proposed neural network dereverberation algorithms on a test dataset consisting of audio from held-out speakers and RIRs not seen during training. Algorithm performance was evaluated at a number of different SNRs (i.e. dry-to-wet ratios), ranging from -5 to $+20$ dB. To evaluate performance in terms of reverberation reduction, we used the signal-to-distortion ratio (SDR) and signal-invariant signal-to-noise ratio (SI-SNR) [16]. For completeness, we computed metrics on the original mixture signals, as well as the results of speech separation using oracle (ground truth) magnitude ratio masks, essentially specifying the expected lower and upper performance bounds for our methods. Additionally, we used the difference of the short-time objective intelligibility measure (STOI) [17] between the processed output and the original reverberant mixtures converted to a percentage ΔSTOI. This measures the overall percent improvement in speech quality and intelligibility relatively to the original reverberant mixtures. Our choice of metrics attempt to quantify algorithm performance through both "standard" and perceptually-driven means.

Table 1 and 2 illustrates our quantitative performance evaluation in terms of reverb reduction and speech quality improvement, respectively. We can observe that the baseline approach clearly improves upon the original reverberant mixture. Our proposed neural network solution consistently outperforms the baseline approach, and as to be expected, performs a few dB worse than the oracle mask solution. The ΔSTOI confirms that speech intelligibility is not as degraded at higher SNRs. These results suggest that our proposed neural network solution can recover about half of the possible of improvement in speech intelligibility relative to the oracle solution, and it outperforms the baseline approach by a large margin in this regard.

We performed informal listening tests, both on our synthesized evaluation set and on real-world speech signals that are naturally reverberated. We observe that our neural network solution can reduce more reverberation than the baseline approach, while remaining rather transparent in its processing and reducing "pumping" artifacts often heard in the baseline approach. The spectrograms in Figure 2 provide a visual comparison between the baseline and proposed approaches on a speech sample with synthetically applied reverberation. We can see that while the baseline approach improves upon the reverberated speech, the proposed approach yields an output that more closely resembles the underlying dry speech. For audio examples and additional spectrograms, please visit http://www.izotope.com/tech/dafx_dereverb .

3.3. Generalization to non-speech signals

Though we have explicitly trained our dereverberation network on speech signals, we have informally noticed that the system can generalize to some classes of non-speech signals. This is particularly fortuitous for our post-production application, where there may be other sound effects, laughter, etc. that may be desirable to salvage in a given performance.



(a) Dry speech

(b) Reverberated speech

(c) Dereverberation of (b) using the baseline approach

(d) Dereverberation of (b) using the proposed approach

Figure 2: Speech dereverberation comparison.

4. CONCLUSIONS

In this paper, we proposed a novel application of source separation to the problem of speech dereverberation, and trained a BRNN to this end. Our proposed solution outperformed a baseline approach based on spectral subtraction through both qualitative and quantitative means. In addition to its improved performance, a benefit of our deep learning approach is that we no longer need to formulate an explicit model for reverberation, and can avoid hand-tuned estimation of reverb parameters altogether. In the future, we would like to research effective low-latency solutions, and additionally consider time-domain architectures which may be able to more accurately observe and remedy early reflections.

5. ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers, whose comments greatly improved the quality of this manuscript.

6. REFERENCES

[1] P. Naylor and N. D. Gaubitch, Eds., *Speech Dereverberation*, Springer, New York, NY, USA, 2010.

[2] B. Yegnanarayana and P. Satyanarayana, "Enhancement of reverberant speech using lp residual signal," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 8, pp. 267–281, May 2000.

[3] E. A. P. Habets and J. Benesty, "A two-stage beamforming approach for noise reduction and dereverberation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 5, pp. 945–957, May 2013.

[4] P.S. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis, "Joint optimization of masks and deep recurrent neural networks for monaural source separation," *IEEE/ACM Transactions on Speech and Language Processing*, vol. 23, pp. 2136–2147, Dec. 2015.

[5] F. Weninger, J. Le Roux, J. R. Hershey, and B. Schuller, "Discriminatively trained recurrent neural networks for single-channel speech separation," in *IEEE GlobalSIP 2014 Symposium on Machine Learning Applications in Speech Processing*, 2014, pp. 577–581.

[6] Z. Q. Wang and D Wang, "Recurrent deep stacking networks for supervised speech separation," in *International Conference on Acoustics, Speech and Signal Processing*, New Orleans, LA, USA, Mar. 5-9 2017.

[7] G. Wichern and A. Lukin, "Removing lavalier microphone rustle with recurrent neural networks," in *Proc. Digital Audio Effects (DAFx-2018)*, Aviero, Portugal, Sept. 4-8 2018, pp. 19–25.

[8] A. Lukin and J. Todd, "Suppression of musical noise artifacts in audio noise reduction by adaptive 2d filtering," in *123rd Audio Engineering Society Convention*, New York, NY, USA, Jan. 2007.

[9] G. Wichern and A. Lukin, "Low-latency approximation of bidirectional recurrent networks for speech denoising," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, USA, Oct. 15-18 2017.

[10] D. Griffin and J. Lim, "Signal estimation from modified short-time fourier transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 2, pp. 236–243, Apr. 1984.

[11] K. Cho et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Conference on Empirical Methods in Natural Language Processing*, 2014.

[12] Y. Wang, A. Narayanan, and D. L. Wang, "On training targets for supervised speech separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, pp. 1849–1858, 2014.

[13] G. Pirker, M. Wohlmayr, S. Petrik, and F. Pernkopf, "A pitch tracking corpus with evaluation on multipitch tracking scenario," in *Interspeech*, 2011, pp. 1509–1512.

[14] G. J. Mysore, "Can we automatically transform speech recorded on common consumer devices in real-world environments into professional production quality speech? A dataset, insights, and challenges," *IEEE Signal Processing Letters*, vol. 22, pp. 1006–1010, 2015.

[15] P. Kabal, "TSP speech database," Tech. Rep., Department of Electrical & Computer Engineering, McGill University, Montreal, Quebec, Canada, 2002.

[16] J. Le Roux, J. R. Hershey, S. T. Wisdom, and H. Erdogan, "SDR: half-baked or well done?," Tech. Rep., Mitsubishi Electric Research Laboratories, Cambridge, MA, USA, 2018.

[17] J. Chen et al., "Large-scale training to increase speech intelligibility for hearing-impaired listeners in novel noises," *Journal of the Acoustical Society of America*, vol. 139, pp. 2604–2612, 2016.

NOTES ON THE USE OF VARIATIONAL AUTOENCODERS FOR SPEECH AND AUDIO SPECTROGRAM MODELING

Laurent Girin, Thomas Hueber
Univ. Grenoble Alpes, CNRS,
Grenoble INP, GIPSA-lab
Grenoble, France
laurent.girin@grenoble-inp.fr
thomas.hueber@grenoble-inp.fr

Fanny Roche *
Arturia
Meylan, France
fanny.roche@arturia.com

Simon Leglaive
Inria Grenoble Rhône-Alpes
Grenoble, France
simon.leglaive@inria.fr

ABSTRACT

Variational autoencoders (VAEs) are powerful (deep) generative artificial neural networks. They have been recently used in several papers for speech and audio processing, in particular for the modeling of speech/audio spectrograms. In these papers, very poor theoretical support is given to justify the chosen data representation and decoder likelihood function or the corresponding cost function used for training the VAE. Yet, a nice theoretical statistical framework exists and has been extensively presented and discussed in papers dealing with nonnegative matrix factorization (NMF) of audio spectrograms and its application to audio source separation. In the present paper, we show how this statistical framework applies to VAE-based speech/audio spectrogram modeling. This provides the latter insights on the choice and interpretability of data representation and model parameterization.

1. INTRODUCTION

Autoencoders (AEs) are a specific type of deep neural networks (DNNs) that can learn from data a non-linear projection of the signal space into a low-dimensional latent space (encoding step), followed by inverse non-linear transformation of the latent coefficients into the original signal space (decoding step) [1]. AEs have been essentially used as an unsupervised technique for data dimension reduction. More recently, variational autoencoders (VAEs) were proposed as a probabilistic/generative extension of AEs [2]: Instead of deterministically mapping the input vector $\mathbf{x}$ into a unique vector of latent coefficients $\mathbf{z}$, as done in AEs, the VAE *encoder network* maps $\mathbf{x}$ into the parameters of a conditional distribution $q_\phi(\mathbf{z}|\mathbf{x})$ of $\mathbf{z}$. Similarly, the *decoder network* maps a vector of latent coefficient $\mathbf{z}$ into the parameters of a conditional distribution $p_\theta(\mathbf{x}|\mathbf{z})$ of $\mathbf{x}$. A VAE decoder is thus intrinsically a (non-linear and deep) generative model of $\mathbf{x}$, conditioned on the latent variable $\mathbf{z}$ (which is itself conditioned on the input when decoding follows encoding). VAEs thus combine the modeling power of DNNs with the flexibility of generative models.

VAEs have recently received a strong interest for speech and audio processing, more specifically for modeling, transformation and synthesis of speech signals [3, 4, 5, 6], for music sound synthesis [7, 8], and for single-channel [9, 10, 11, 12] and multi-channel [13, 14, 15] speech enhancement and separation. In all those papers, VAEs are used to process a sequence of vectors encoding the short-time Fourier transform (STFT) spectrogram extracted from speech or music signals. For synthesis/transformation applications, the output audio signal is reconstructed using the decoded magnitude spectrogram, after possible modification of the latent coefficients, and either the phase of the original signal or some reconstructed phase more coherent with the decoded magnitude spectrogram. For speech enhancement application, the decoder of the VAE is used as a supervised generative model of the speech signal in the STFT domain, which is exploited in a probabilistic enhancement/separation method.

A keypoint is that in most of these papers, very few justification is given about the precise choice of the encoder and decoder conditional distributions, or the corresponding cost function used for VAE training. These distributions are generally chosen as Gaussian for convenience, but the choice for their parameters is not clearly justified. The same about the related issue of data representation: It is chosen a bit arbitrarily, without clear theoretical support, possibly more considering DNN training issues rather than fundamental signal processing ones.

Yet, this theoretical framework exists. In fact, it has been extensively presented and discussed in the seminal papers [16] and [17]. Those papers describe the statistical framework underlying the decomposition of audio magnitude/power spectrograms using Nonnegative Matrix Factorization (NMF) [18]. These developments have then been extensively used for audio source separation, see e.g. among many others [19, 20, 21, 22, 23, 24, 25]. In the present paper, we show how this theoretical statistical framework applies to the VAE model. Based on [16, 17], we describe the three main cases encountered in practice, with three modeling cost functions corresponding to three signal statistical models. We show how this provides interesting insights on the choice and interpretability of data representation and loss function for speech/audio spectrogram modeling with VAEs.

The remainder of this paper is organized as follows. Section 2 presents the VAE framework. In Section 3, we discuss the way VAEs are currently used to model speech/audio signals in the literature, and raise a set of related questions. In Section 4 we present the nonnegative representation and underlying signal statistical models as a general framework, of which NMF is a particular case, and we show how this framework also applies to VAE-based spectrogram modeling. Section 5 illustrates this discussion with some experiments on speech/audio analysis-synthesis with VAEs. Section 6 draws a series of conclusions and perspectives.

* This work is supported by a CIFRE PhD Grant funded by ANRT

2. VARIATIONAL AUTOENCODERS

As mentioned in the introduction, a VAE can be seen as a probabilistic autoencoder. In the original formulation of the seminal paper [2], a VAE delivers a parametric model of data distribution:

$$p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z}), \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^F$ is a vector of observed data, $\mathbf{z} \in \mathbb{R}^L$ is a corresponding vector of latent data, with $L \ll F$, and θ denotes the set of distribution parameters. The likelihood function $p_\theta(\mathbf{x}|\mathbf{z})$ plays the role of a probabilistic decoder which models how the generation of observed data $\mathbf{x}$ is conditioned on the latent data $\mathbf{z}$. The prior distribution $p_\theta(\mathbf{z})$ is used to structure (or regularize) the latent space. Typically a standard Gaussian distribution is used: $p_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I}_L)$, where $\mathbf{I}_L$ is the identity matrix of size L. This encourages the latent coefficients to be orthogonal and with similar range. Note that this prior actually lacks parameters. The likelihood $p_\theta(\mathbf{x}|\mathbf{z})$ is usually defined as Gaussian:

$$p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_\theta(\mathbf{z}), \boldsymbol{\sigma}^2_\theta(\mathbf{z})), \tag{2}$$

where $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ denotes the probability density function (pdf) of the multivariate Gaussian distribution which is defined in the Appendix, and $\boldsymbol{\mu}_\theta(\mathbf{z}) \in \mathbb{R}^F$ and $\boldsymbol{\sigma}^2_\theta(\mathbf{z}) \in \mathbb{R}^F_+$ are the outputs of the decoder network. The parameter set θ is composed of the weights of this decoder network. Note that the entries of $\mathbf{x}$ are assumed independent as common in VAEs, so the vector $\boldsymbol{\sigma}^2_\theta(\mathbf{z})$ contains the diagonal coefficients of a diagonal covariance matrix.

The exact posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ corresponding to the above model is intractable. It is approximated with a tractable parametric model $q_\phi(\mathbf{z}|\mathbf{x})$ that plays the role of the corresponding probabilistic encoder. This model generally has a form similar to the decoder:

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \tilde{\boldsymbol{\mu}}_\phi(\mathbf{x}), \tilde{\boldsymbol{\sigma}}^2_\phi(\mathbf{x})), \tag{3}$$

where $\tilde{\boldsymbol{\mu}}_\phi(\mathbf{x}) \in \mathbb{R}^L$ and $\tilde{\boldsymbol{\sigma}}^2_\phi(\mathbf{x}) \in \mathbb{R}^L_+$ are the outputs of the encoder network. The parameter set ϕ is composed of the weights of this encoder network. As before, $\tilde{\boldsymbol{\sigma}}^2_\phi(\mathbf{x})$ is a vector containing the diagonal entries of a diagonal covariance matrix.

Training of the VAE model, i.e. estimation of θ and ϕ, is made by optimizing a lower-bound of the marginal log-likelihood $\log p_\theta(\mathbf{x})$ computed from a large training dataset of vectors $\mathbf{x}$. It is shown in [2] that the marginal log-likelihood for an individual vector $\mathbf{x}$ writes:

$$\log p_\theta(\mathbf{x}) = d_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})|p_\theta(\mathbf{z}|\mathbf{x})) + \mathcal{L}(\phi, \theta, \mathbf{x}), \tag{4}$$

where $d_{\text{KL}} \geq 0$ denotes the Kullback-Leibler (KL) divergence and $\mathcal{L}(\phi, \theta, \mathbf{x})$ is the variational lower bound (VLB) given by:

$$\mathcal{L}(\phi, \theta, \mathbf{x}) = \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]}_{\text{reconstruction accuracy}} \underbrace{- d_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})|p_\theta(\mathbf{z}))}_{\text{regularization}}. \tag{5}$$

We can see that the VLB is the sum of two terms. The first term represents the average reconstruction accuracy. The second term acts as a regularizer encouraging the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ to be close to the prior $p_\theta(\mathbf{z})$. Since the expectation taken with respect to $q_\phi(\mathbf{z}|\mathbf{x})$ in the reconstruction accuracy term is analytically intractable, it is approximated using a Monte Carlo estimate with R samples $\mathbf{z}^{(r)}$ independently and identically drawn from $q_\phi(\mathbf{z}|\mathbf{x})$:

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] \approx \frac{1}{R}\sum_{r=1}^{R} \log p_\theta(\mathbf{x}|\mathbf{z}^{(r)}). \tag{6}$$

In practice a training dataset $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^{N_{tr}}$ is used for the training of the VAE. Under the hypothesis of independent and identically distributed (i.i.d.) training vectors, the VAE training is done by maximizing the total VLB, which is the sum of individual VLBs over the training vectors. If we consider only one Monte Carlo sample per training vector (which is common practice provided that the batch size is sufficiently large [2]), or if we consider several Monte Carlo samples as additional training data, we can write the total VLB as:

$$\begin{aligned}\mathcal{L}(\phi, \theta, \mathbf{X}) = &\sum_{n=1}^{N_{tr}} \log p_\theta(\mathbf{x}_n|\mathbf{z}_n) \\ &- \sum_{n=1}^{N_{tr}} d_{\text{KL}}(q_\phi(\mathbf{z}_n|\mathbf{x}_n)|p_\theta(\mathbf{z}_n)).\end{aligned} \tag{7}$$

For the present case of Gaussian likelihood (2) and Gaussian encoding distribution (3), the VLB in (7) becomes:

$$\begin{aligned}\mathcal{L}(\phi, \theta, \mathbf{X}) = &-\sum_{n=1}^{N_{tr}}\sum_{f=0}^{F-1}\Big(\log \sigma^2_{\theta,f}(\mathbf{z}_n) + \frac{(x_{fn} - \mu_{\theta,f}(\mathbf{z}_n))^2}{2\sigma^2_{\theta,f}(\mathbf{z}_n)}\Big) \\ &+ \frac{1}{2}\sum_{n=1}^{N_{tr}}\sum_{l=1}^{L}\Big(\log \tilde{\sigma}^2_{\phi,l}(\mathbf{x}_n) - \tilde{\mu}_{\phi,l}(\mathbf{x}_n)^2 - \tilde{\sigma}^2_{\phi,l}(\mathbf{x}_n)\Big)\end{aligned} \tag{8}$$

where the subscript f or l denotes the f-th or l-th entry of a vector. Maximization of the total VLB is done by using the usual back-propagation technique and gradient-based optimization, which are not detailed in this paper. For more technical details that are not relevant here, the reader is referred to [2].

3. VAES FOR SPECTROGRAM MODELING: FACTS AND QUESTIONS

In this section, we analyze how VAEs are generally used for speech and audio spectrogram modeling in the recent literature. Although some of the points discussed below may seem trivial, they rise a series of fundamental questions that are poorly discussed in these papers and that we will address in the following.

3.1. Audio signal representation in the STFT domain

As shortly stated in the introduction, the processing is generally carried out in the STFT domain. Let $\mathbf{S} = [s_{fn}]_{f=0,n=1}^{F-1,N} \in \mathbb{C}^{F\times N}$ denote the STFT of a speech/audio signal, where f is the frequency bin index and n is the time frame index. Let $\mathbf{X} = [x_{fn}]_{f=0,n=1}^{F-1,N} \in \mathbb{R}_+^{F\times N}$ denote the corresponding real-valued and nonnegative *magnitude* or *power* spectrogram, i.e. $\mathbf{X} = |\mathbf{S}|$ or $\mathbf{X} = |\mathbf{S}|^2$, where $|.|$ and $.^2$ are to be understood as entry-wise operators. Note that we use the same notation as in the previous section on purpose, since the VAE modeling will precisely be applied on speech/audio spectrograms. Note also that $\mathbf{X} = |\mathbf{S}|^2$ is a sampled power spectrogram, aka a periodogram, i.e. an estimate of the power spectral density (PSD) $\mathbb{E}[|\mathbf{S}|^2]$ built from a single observation of the data in each time-frequency bin (and the same for the magnitude spectrogram).

3.2. Data representation, pre-processing and normalization

A VAE considers vectors as input and output. Hence an STFT spectrogram is processed as a sequence of successive spectral vectors $\mathbf{x}_n = [x_{fn}]_{f=0}^{F-1} \in \mathbb{R}_+^F$, each vector representing an STFT frame. Note that all x_{fn} are assumed independent across frequency bins and time frames, which is not to be confused with possible time-frequency structuration of the distribution parameters. An important practical question in VAEs is the choice of the audio STFT data representation. We did not observe any consensus in the literature.

For synthesis and transformation applications, e.g. [6], the observed/generated vector at time frame n generally corresponds to the short-term magnitude or power spectrum. There may be two explanations for that: (i) the original VAE formulation of [2] (i.e. the Gaussian models in (2) and (3)) considers real-valued and not complex-valued vectors, but in that case what about the non-negativity? and (ii) the magnitude or power spectrogram is the primary information used in the synthesis/transformation applications considered in the referenced papers (the phase spectrogram being processed separately).

For speech enhancement applications, the VAE speech model is generally plugged in a more general statistical framework including a noise model and a speech + noise mixture model, e.g. [9, 10]. In this framework, the original (real-valued) formulation of the VAE has been extended to model the complex-valued STFT vector $\mathbf{s}_n = [s_{fn}]_{f=0}^{F-1} \in \mathbb{C}^F$. This has been done by replacing the Gaussian distribution over real-valued vectors in (2) with the circularly symmetric complex Gaussian distribution that is widely used in speech enhancement and source separation probabilisitic methods [26, 27]. This important point is poorly commented in the referenced papers. Moreover, although $\mathbf{s}_n$ is here modeled by the VAE decoder, $\mathbf{x}_n$ as a short-term magnitude or power spectrum is still considered at the input of the encoder during VAE training.[1] The possible consequences (or absence of consequences) of this input/output mismatch are not discussed either. Note that here also, all s_{fn} are assumed independent across frequency bins and time frames, as is usually done in the speech enhancement and source separation literature.

It is important to note that in practice, the encoder input vector can contain magnitudes or squared magnitudes as discussed above, but also log-magnitudes as in [4], or actually any vector encoding a magnitude spectrum, possibly pre-processed and normalized in different manners. Normalization is a typical example of DNN-driven process, it has no theoretical justification from the signal processing point-of-view but it is known as helping a DNN training in general. So it is applied very frequently, and actually on purpose in VAEs. Also, the encoder input vector can be of different nature than the VAE decoder output vector, which is composed of probability distribution parameters; not to be confused with the output of the VAE as a generative model. Some of the output parameters may be homogeneous to the input data, e.g. mean vectors, and some others may not be, e.g. variance parameters. Moreover, data normalization can also be applied to output data, and the normalization/denormalization can be conducted in different manners at the input and at the output. Then, does data representation, pre-processing and normalization have any consequence on the theoretical foundations of the model?

[1] For speech enhancement applications, the encoder is only used for VAE training. During the speech signal inference process, only the decoder is used.

3.3. Statistical modeling and implications for VAE training

The choice of the reconstruction term of the loss function for the VAE training is often poorly discussed in papers dealing with VAE-based spectrogram modeling. A typical yet poorly justified approach could be: Let us choose a data representation that is appropriate for the considered application, for example a magnitude spectrum vector $\mathbf{x}_n$, and let us apply some normalization that is appropriate for DNNs. Then systematic application of the Gaussian model (2) is the easy way, leading to the weighted squared error form in the reconstruction term of (8). If we further set the variance parameters $\sigma^2_{\theta,f}(\mathbf{z}_n)$ to an arbitrarily fixed value σ^2 (i.e. we consider only the mean parameters $\mu_{\theta,f}(\mathbf{z}_n)$ as the free VAE outputs), then (8) becomes (up to an additive constant factor):

$$\mathcal{L}(\phi,\theta,\mathbf{X}) = -\frac{1}{\sigma^2}\sum_{n=1}^{N_{tr}}\sum_{f=0}^{F-1}\frac{1}{2}\big(x_{fn}-\mu_{\theta,f}(\mathbf{z}_n)\big)^2 + \frac{1}{2}\sum_{n=1}^{N_{tr}}\sum_{l=1}^{L}\Big(\log\tilde{\sigma}^2_{\phi,l}(\mathbf{x}_n) - \tilde{\mu}_{\phi,l}(\mathbf{x}_n)^2 - \tilde{\sigma}^2_{\phi,l}(\mathbf{x}_n)\Big) \quad (9)$$

This means that using the basic mean squared error (MSE) as the reconstruction term of the VAE loss function amounts to maximize the likelihood function under the present "fixed-variance free-mean" Gaussian model, hence providing some nice theoretical interpretation of the process. Yet this interpretation is poorly discussed in the papers. Does this approach have limitations? Does it make sense to model normalized magnitude vectors with a Gaussian distribution? Do other strategies exist? And what is the link with the problem of data representation?

As briefly mentioned in the introduction, a consistent theoretical framework exists that enables one to justify and interpret the choice of data representation, likelihood function and reconstruction term of the loss function, and how those points are related. This is what we present in the next section.

4. LINKING NMF AND VAE

In this section, we build on the existing statistical framework related to nonnegative representations, in particular Nonnegative Matrix Factorization (NMF), and its application to the modeling of speech/audio spectrograms. Most of the technical material presented here is extracted from [16] and [17]. We first shortly present the principle of NMF decomposition, then we go to the major point of this section which is to show that the underlying statistical framework directly applies to the VAE model, and can thus be used to give a solid theoretical interpretation of VAE-based modeling of speech/audio spectrograms. We finally report the three major NMF-based generative models considered in [16] and [17] and give their VAE counterparts.

4.1. The NMF model

NMF consists in modeling a matrix $\mathbf{V} = [v_{fn}]_{f,n} \in \mathbb{R}_+^{F\times N}$ of nonnegative entries as the product of two nonnegative matrices $\mathbf{W} = [w_{fk}]_{f,k} \in \mathbb{R}_+^{F\times K}$ and $\mathbf{H} = [h_{kn}]_{k,n} \in \mathbb{R}_+^{K\times N}$. In other words we have $\mathbf{V} \approx \widehat{\mathbf{V}} = \mathbf{WH}$, or equivalently $\widehat{v}_{fn} = (\mathbf{WH})_{fn} = \sum_{k=1}^{K} w_{fk}h_{kn}$. A low-rank approximation of $\mathbf{V}$, represented with a reduced number of parameters, is obtained by setting K such that $K(F+N) \ll FN$. In the speech/audio processing literature, $\widehat{\mathbf{V}}$ is typically used to model the signal ("true" or

"theoretical") PSD $\mathbb{E}[|\mathbf{S}|^2]$ based on the observed power spectrogram $\mathbf{X} = |\mathbf{S}|^2$ (or the same for the "true" magnitude spectrogram based on the observed magnitude spectrogram $\mathbf{X} = |\mathbf{S}|$). The interest of this approach is thus to provide a model of the signal PSD in each time-frequency bin with a very reasonable number of parameters (if K is chosen properly).

Calculating $\widehat{\mathbf{V}}$ from a given observed nonnegative matrix $\mathbf{X}$ is done by minimizing over $\mathbf{W}$ and $\mathbf{H}$ the following error under a non-negativity constraint:

$$D(\mathbf{X}|\widehat{\mathbf{V}}) = \sum_{n=1}^{N} \sum_{f=0}^{F-1} d(x_{fn}|\hat{v}_{fn}), \tag{10}$$

where $d(\cdot|\cdot)$ is a scalar divergence. The three most popular cost functions are the squared Euclidian distance $d_{\text{EUC}}(x|y) = 0.5(x-y)^2$, the generalized Kullback-Leibler (KL) divergence $d_{\text{KL}}(x|y) = x\log(x/y) - x + y$, and the Itakura-Saito (IS) divergence $d_{\text{IS}}(x|y) = x/y - \log(x/y) - 1$. For each of them, a set of algorithms have been proposed to solve the above minimization problem. Their presentation is out of the scope of this paper, where we focus on the link with the VAE and the underlying statistical models. For the same reason, we do not deal with the interpretation of NMF as a model of composite signals [16, 17], which is of primary importance in the source separation literature.

4.2. Linking NMF- and VAE-based spectrogram modeling

Now the major point of the present paper is the following: *The minimization of the global cost function* (10)*, the choice of the scalar cost function in* (10)*, the choice of data representation, and the interpretation in terms of underlying statistical model are problems that are all common to NMF and VAE.* In other words, *a common framework exists where* $\widehat{\mathbf{V}}$ *may as well be an NMF model* $\widehat{\mathbf{V}} = \mathbf{WH}$ *or the concatenation of successive (nonnegative) output vectors of a VAE*, e.g. $\widehat{\mathbf{V}} = [\boldsymbol{\sigma}_\theta^2(\mathbf{z}_1), \boldsymbol{\sigma}_\theta^2(\mathbf{z}_2), \cdots, \boldsymbol{\sigma}_\theta^2(\mathbf{z}_N)]$, which is the case for VAE-based spectrogram modeling. Indeed, as will be detailed below, for both NMF and VAE models, (10) is nothing but a reformulation of the negative log-likelihood function of the underlying generative model. More specifically, if $\widehat{\mathbf{V}}$ is the output of a VAE, the reconstruction accuracy in (7) and the cost function (10) are identical up to a constant multiplicative positive factor α, sign, and a constant additive factor. In short, (7) can be rewritten as:

$$\begin{aligned}\mathcal{L}(\phi, \theta, \mathbf{X}) = -\alpha &\sum_{n=1}^{N_{tr}} \sum_{f=0}^{F-1} d(x_{fn}|\hat{v}_{fn}) \\ &- \sum_{n=1}^{N_{tr}} d_{\text{KL}}(q_\phi(\mathbf{z}_n|\mathbf{x}_n)|p_\theta(\mathbf{z}_n)).\end{aligned} \tag{11}$$

In the VAE model framework, minimization of (10) *thus amounts to optimal estimation of the VAE parameters in the maximum-likelihood (ML) sense*. Let us temper a bit: (10) only concerns the VAE decoder, and the complete VAE is actually optimized by maximizing (7) (or (11)), i.e. the combination of (10) with the VLB regularization term. This latter is important to differentiate a VAE from a deterministic AE. Let us note that in the VAE framework, ML estimation of $\widehat{\mathbf{V}}$ is to be understood as a shortcut for ML estimation of θ, the decoder parameters, which requires the joint estimation of the encoder parameters ϕ during the VAE training. Finally, let us also note that α plays the role of balancing factor between reconstruction and regularization, and quite interestingly, it is very similar to the β factor of the β-VAE model proposed in [28] in an ad-hoc manner, for the same aim (though β is applied to the regularization term instead of the reconstruction term).

Although all these points may sound trivial to readers familiar with the statistical interpretation of NMF spectrogram modeling, to our knowledge they have never been pointed out in the literature on VAE-based speech/audio processing. One reasonable explanation for this may be that NMF studies often start with the cost function formulated as (10), and the interpretation in terms of underlying generative model comes in second (when it comes), whereas VAE studies start with a generative model then go to the cost function formulated as (7).

4.3. Practical cases

We now apply the above considerations to the three major cases considered in [16] and [17], which correspond to different divergences $d(\cdot|\cdot)$ in (10) and (11).

Euclidian distance case In the NMF context, it has been shown in [16, 17] that choosing and minimizing the squared Euclidian distance between $\mathbf{X}$ and $\widehat{\mathbf{V}} = \mathbf{WH}$ corresponds to ML estimation of $\mathbf{W}$ and $\mathbf{H}$ under the assumption of the Gaussian model

$$x_{fn} \sim \mathcal{N}(x_{fn}; \widehat{v}_{fn}, \sigma^2), \tag{12}$$

with $\widehat{v}_{fn} = (\mathbf{WH})_{fn} = \sum_{k=1}^{K} w_{fk}h_{kn}$. Similarly, in the VAE case, choosing and minimizing the squared Euclidian distance between x_{fn} and $\widehat{v}_{fn}$ in (11), with $\widehat{v}_{fn} = \mu_{\theta,f}(\mathbf{z}_n)$, corresponds to ML estimation of $\widehat{v}_{fn}$ under the assumption of the Gaussian model (2), with a fixed variance $\sigma^2_{\theta,f}(\mathbf{z}_n) = \sigma^2, \forall(f,n)$. Actually this is what we have already done at the end of Section 3, and formalized in (9). In both NMF and VAE cases, we have the following underlying model:

$$x_{fn} = \widehat{v}_{fn} + e_{fn}, \tag{13}$$

where e_{fn} is an i.i.d. additive white Gaussian noise, i.e. $e_{fn} \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2)$. Moreover, identifying (11) and (9) leads to $\alpha = 1/\sigma^2$, hence σ^2 plays the role of balancing factor between reconstruction and regularization.

A Gaussian model is often favored because of its generality and its nice features in mathematical derivations. For instance, it has been used for VAE-based speech spectrogram modeling in [4, 6, 11, 29]. However, although this approach could work quite well in many settings, it suffers from what is referred to as an interpretation ambiguity in [16]: Although x_{fn} represents a magnitude or power spectrum, $\mathcal{N}(x_{fn}; \mu_{\theta,f}(\mathbf{z}_n), \sigma^2)$ may produce negative data (even if we somehow enforce $\mu_{\theta,f}(\mathbf{z}_n) \geq 0$). This problem may be partly fixed by appropriate data normalization (e.g. min-max rescaling within $[-1, 1]$) and/or with log-scaling. However, it is subject to discussion if the distribution of log-magnitude spectra of real-world speech and audio signals has a Gaussian shape or not.

Itakura-Saito divergence case Alternately, it was shown and largely discussed in [17] that using the IS divergence in (10) corresponds to maximizing the log-likelihood function under the assumption of a Gamma distribution for x_{fn}. More precisely, the statistical model is:

$$x_{fn} \sim \mathcal{G}(x_{fn}; \alpha, \alpha/\widehat{v}_{fn}), \tag{14}$$

where $\mathcal{G}(.;a,b)$ is the Gamma distribution with shape parameter $a>0$ and rate parameter $b>0$, and whose pdf is defined in the Appendix. In the NMF framework we have $\widehat{v}_{fn}=(\mathbf{WH})_{fn}=\sum_{k=1}^{K}w_{fk}h_{kn}$, but this result is still valid in the VAE framework where we now have $\widehat{v}_{fn}=\sigma^2_{\theta,f}(\mathbf{z}_n)$. In both NMF and VAE cases, we have the following underlying model:

$$x_{fn}=\widehat{v}_{fn}e_{fn}, \tag{15}$$

where e_{fn} is an i.i.d. multiplicative Gamma noise, i.e. $e_{fn}\overset{i.i.d.}{\sim}\mathcal{G}(e_{fn};\alpha,\alpha)$.

Importantly, it was also shown in [17] that if x_{fn} corresponds to a linear-scale squared magnitude, minimizing the IS divergence corresponds to ML estimation of $\widehat{v}_{fn}$ under a circularly symmetric complex Gaussian model for the STFT coefficients $s_{fn}\in\mathbb{C}$ corresponding to $x_{fn}=|s_{fn}|^2\in\mathbb{R}_+$, with a variance $\mathbb{E}[|s_{fn}|^2]$ equal to $\widehat{v}_{fn}$. In short, $s_{fn}\sim\mathcal{N}_c(s_{fn};0,\widehat{v}_{fn})$, where the pdf of the complex Gaussian distribution $\mathcal{N}_c$ is defined in the Appendix. This interpretation is quite important since this model and associated ML fitting procedure have been used extensively in speech enhancement and speech/audio source separation, in combination with NMF, e.g. [19, 21, 23], or not, e.g. [26, 20, 30]. Indeed, in such applications, we are interested in inferring the complex-valued source STFT coefficients s_{fn} from corrupted observations. Again, this result is valid for both NMF and VAE frameworks: In IS-based NMF, we have $\mathbb{E}[|s_{fn}|^2]=\mathbb{E}[x_{fn}]=\widehat{v}_{fn}=\sum_{k=1}^{K}w_{fk}h_{kn}$. In IS-based VAE, we have $\mathbb{E}[|s_{fn}|^2]=\mathbb{E}[x_{fn}]=\widehat{v}_{fn}=\sigma^2_{\theta,f}(\mathbf{z}_n)$ and the mean parameters $\mu_{\theta,f}(\mathbf{z}_n)$ are simply disregarded since (2) is implicitly replaced with the above Gamma model of x_{fn}. Note that IS-VAE was shown to outperform IS-NMF for speech enhancement in [10].

Generalized Kullback-Leibler divergence case Finally, minimizing the KL divergence between x_{fn} and $\widehat{v}_{fn}$ corresponds to ML estimation of $\widehat{v}_{fn}$ under the assumption of a Poisson distribution for x_{fn}:

$$x_{fn}\sim\mathcal{P}(x_{fn};\widehat{v}_{fn}), \tag{16}$$

where $\mathcal{P}(.;\lambda)$ is the Poisson distribution with scale parameter $\lambda>0$ and whose pdf is defined in the Appendix. Note that there is here no equivalent model in terms of additive or multiplicative noise. In theory, the Poisson distribution is defined for nonnegative integer-valued random variables, but this issue can be fixed by considering high-resolution fixed-point quantization of the spectrograms. As above, this result is valid for both NMF and VAE models. Here, $\widehat{v}_{fn}$ plays the role of a scale parameter, hence in principle the output of a KL-based VAE is a vector of scale parameters $\widehat{v}_{fn}=\sigma_{\theta,f}(\mathbf{z}_n)$ for $f=0,...,F-1$. Although, as stated above, arbitrary normalization and corresponding denormalization can be applied. Historically, KL-based NMF has been applied on (linear-scale) magnitude spectra instead of power spectra, see the seminal papers [31, 32], but in fact there is no underlying model on the complex-valued STFT coefficients s_{fn} to support this principle. In other words, in most papers on KL-based NMF, $\widehat{v}_{fn}$ is a scale parameter over magnitude spectra, because x_{fn} is a magnitude spectra, but it could as well be a scale parameter over a different representation. Of course, the same remark applies to a KL-based VAE.

In summary, in the speech/audio spectrogram NMF modeling framework, we had:

- EUC-NMF: $p_\theta(\mathbf{X}|\mathbf{Z})=\prod_{f,n}\mathcal{N}(x_{fn};(\mathbf{WH})_{fn},\sigma^2)$;
- IS-NMF: $p_\theta(\mathbf{X}|\mathbf{Z})=\prod_{f,n}\mathcal{G}(x_{fn};\alpha,\alpha/(\mathbf{WH})_{fn})$ and $p_\theta(\mathbf{S}|\mathbf{Z})=\prod_{f,n}\mathcal{N}_c(s_{fn};0,(\mathbf{WH})_{fn})$ with $x_{fn}=|s_{fn}|^2$;
- KL-NMF: $p_\theta(\mathbf{X}|\mathbf{Z})=\prod_{f,n}\mathcal{P}(x_{fn};(\mathbf{WH})_{fn})$.

In the VAE framework we have:

- EUC-VAE: $p_\theta(\mathbf{X}|\mathbf{Z})=\prod_{f,n}\mathcal{N}(x_{fn};\mu_{\theta,f}(\mathbf{z}_n),\sigma^2)$;
- IS-VAE: $p_\theta(\mathbf{X}|\mathbf{Z})=\prod_{f,n}\mathcal{G}(x_{fn};\alpha,\alpha/\sigma^2_{\theta,f}(\mathbf{z}_n))$ and $p_\theta(\mathbf{S}|\mathbf{Z})=\prod_{f,n}\mathcal{N}_c(s_{fn};0,\sigma^2_{\theta,f}(\mathbf{z}_n))$ with $x_{fn}=|s_{fn}|^2$;
- KL-VAE: $p_\theta(\mathbf{X}|\mathbf{Z})=\prod_{f,n}\mathcal{P}(x_{fn};\sigma_{\theta,f}(\mathbf{z}_n))$.

4.4. A practical note on the implementation of the VAE loss function

The above considerations have a practical consequence in the coding of the loss function when implementing a VAE with a deep learning library. Indeed, in practice, as stated above, input/output data are often pre-processed (e.g. log-scaled) and/or normalized to facilitate the VAE training. For the statistical interpretation considered in this paper to hold, the reconstruction term of the VAE loss function, as implemented in a deep learning toolkit, must have the form of the log-likelihood function $\log p_\theta(\mathbf{x}|\mathbf{z})$, and the data used in this loss function must be consistent with the model, i.e. if they have been previously normalized, then *they must be denormalized*. Using the normalized data would break the consistency of the underlying statistical model.

Let us give an example, by considering the Gamma model in (14) for the squared STFT magnitudes $x_{fn}=|s_{fn}|^2$. This model *implies that we have to* use the IS divergence in the reconstruction term of the loss function in (11). At training time, the VAE is fed with pre-processed/normalized data $x_{fn}^{\text{norm}}=g(x_{fn})$ and it provides pre-processed/normalized scale parameters $\widehat{v}_{fn}^{\text{norm}}=\tilde{g}(\widehat{v}_{fn})$. Note that the pre-processing/normalization of data and parameters may be different, as denoted by the different $g(\cdot)$ and $\tilde{g}(\cdot)$ functions. Then the implementation of the reconstruction term of the loss function based on the IS divergence and "applied to" x_{fn}^{norm} and $\widehat{v}_{fn}^{\text{norm}}$ should be of the form:

$$\frac{g^{-1}(x_{fn}^{\text{norm}})}{\tilde{g}^{-1}(\widehat{v}_{fn}^{\text{norm}})}-\log\frac{g^{-1}(x_{fn}^{\text{norm}})}{\tilde{g}^{-1}(\widehat{v}_{fn}^{\text{norm}})}-1=d_{\text{IS}}(x_{fn}|\widehat{v}_{fn}). \tag{17}$$

The denormalized outputs $\widehat{v}_{fn}=\tilde{g}^{-1}(\widehat{v}_{fn}^{\text{norm}})$ are then "automatically" homogeneous to scale parameters. In contrast, using directly the normalized values in the above reconstruction term (i.e. calculating $d_{\text{IS}}(x_{fn}^{\text{norm}}|\widehat{v}_{fn}^{\text{norm}})$) or using another distance (e.g. the MSE) on either the normalized or denormalized data would not be consistent with the Gamma model considered in this example.

5. EXPERIMENTS

In this section, we briefly present the results of experiments that were conducted to illustrate our discussion. We processed VAE-based analysis-synthesis of sound spectrograms for the three cases described in Section 4. Waveform resynthesis was done by combining the output magnitude spectrogram with the phase spectrogram of the original signal. We applied this on speech signals (TIMIT dataset [33], 10 utterances $\times$ 462 speakers in the training set, for a total of about 4h, and 10 different utterances $\times$ 168 different speakers in the test set, for a total of about 1.5h) and music

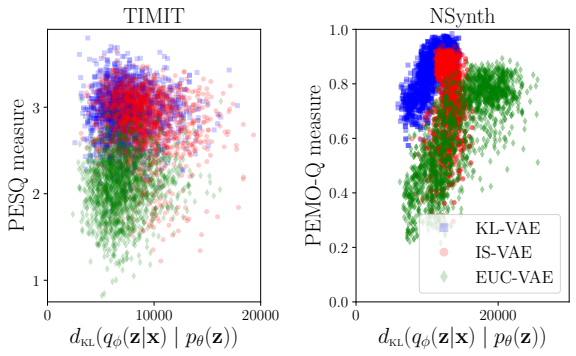


Figure 1: Audio quality as a function of the regularization term of (5).

signals (a subset of the large NSynth dataset [34], 88 notes with 4 different velocities from 17 instruments for the training set and 3 instruments for the test set all from the acoustic keyboards family, for a total of 9h of signals) at a 16 kHz sampling rate. The STFT was computed using a 64-ms sine window ($F = 513$) and a 75% overlap.

The VAE decoder network contains three layers of size [64, 128, 513] and the encoder network is the symmetric. Both networks use $\tanh$ and identity activation functions for the hidden and output layers respectively. The output of the encoder and decoder networks are thus real-valued, and as proposed in the original paper on VAEs [2], we output the logarithm of variance/scale parameters for the IS-VAE and KL-VAE cases. At the input of the encoder, we provide either magnitude spectrograms (KL-VAE and EUC-VAE) or power spectrograms (IS-VAE).

The results are plotted in Fig. 1. In order to measure the quality of the reconstructed signal independently of the nature of the cost function, PESQ scores [35] (for speech) and PEMO-Q scores [36] (for music) were calculated on the resynthesized signals in the test set. These scores are plotted in Fig. 1 as a function of the regularization term of (5). Each point represents either a utterance (left) or a music note (right) from the dataset. We set $\alpha = 0.1$ in (7) for the IS-VAE, and $\alpha = 1$ for both EUC-VAE and KL-VAE. This was to ensure (i) to keep a sufficiently small regularization term in the loss function so that VAEs are not turning into a deterministic autoencoders, and (ii) to obtain the same range of regularization term values for the 3 cost functions, so that the performance can be fairly compared in terms of reconstruction quality. We can see in Fig. 1 that for music signals (PEMO-Q scores) KL-VAE globally performs the best, followed by IS-VAE (with an overlapping zone of equal performance). For speech signals (PESQ scores), KL-VAE and IS-VAE are providing similar results. EUC-VAE generally provides lower scores.

6. CONCLUSION

We can now draw the following conclusions:

- The three presented cost functions usable for NMF or VAE modeling all correspond to an underlying statistical model of processed spectrogram $\mathbf{X} = [\mathbf{x}_n]_{n=1}^N$. For all three cases, training the VAE with data $\mathbf{X}$ corresponds to ML estimation of VAE decoder parameters under the corresponding statistical model of $\mathbf{X}$.
- Among these three cases, only one (IS-case) has an underlying statistical model of the speech/audio signal STFT coefficient s_{fn} (circularly symmetric complex Gaussian), which has proven to be of great interest for speech enhancement and source separation applications.
- The reconstruction accuracy and regularization of the VAE can be weighted using the α factor in (11). For EUC-VAE and IS-VAE this factor is naturally emerging as a parameter of the underlying statistical model, which provides a nice alternative (or interpretation) to the ad-hoc definition of the similar β factor introduced in [28]. This is not the case for KL-VAE, where $\alpha = 1$. For the interpretation of IS-VAE in terms of complex Gaussian model on s_{fn} to hold, we must also have $\alpha = 1$.
- In our experiments, KL-VAE and IS-VAE perform better than EUC-VAE according to perceptually-motivated objective measures.
- Although we necessarily presented this extension in the context of nonnegative representations, VAEs are not limited to nonnegative data. They can be applied to any real-valued data. This is what is done when processing log-scale spectrograms such as in [4]. The IS and KL divergences and associated Gamma and Poisson models are limited to nonnegative data, but the Euclidian distance and associated Gaussian model are not.
- In practice, input/output data are often pre-processed and/or normalized. If the pre-processed/normalized data are used in the VAE practical implementation, then the loss function should include denormalization and inverse pre-processing.
- All the points considered in this paper are valid for recurrent VAEs [37], which are likely to become popular in speech/audio processing as well. Also, generalization of NMF to more general divergences and corresponding statistical interpretation exist, e.g. [38, 39]. It is likely to be relevant for VAEs.

We spent time and effort to understand the correct form that a VAE loss function should have in a deep learning library to be consistent with a sounded signal statistical model. We believe that sharing the content of this paper (and code if the paper is accepted) with the speech/audio processing community can help colleagues to take VAEs into hand faster and in a principled manner. Also, we believe that the bridge we built in this paper can benefit to both the speech enhancement / source separation community and the musical sound processing community.

A. PROBABILITY DISTRIBUTIONS

A.1. Gaussian distributions

Let $\mathcal{N}(x; \mu, \sigma^2)$ denote the Gaussian distribution for a random variable $x \in \mathbb{R}$ with mean $\mu \in \mathbb{R}$ and variance $\sigma^2 \in \mathbb{R}_+$. Its probability density function (pdf) is defined by:

$$\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right). \qquad (18)$$

Note that for simplicity we use the same notation to denote a probability distribution and its pdf.

Let $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ denote the multivariate Gaussian distribution for a real-valued random vector $\mathbf{x} \in \mathbb{R}^F$ of mean vector $\boldsymbol{\mu} \in \mathbb{R}^F$,

and with statistically independent entries such that $\boldsymbol{\sigma}^2 \in \mathbb{R}_+^F$ is the vector of variances (covariance terms are zero and thus omitted in the parametrization for simplicity). Its pdf is therefore equal to the product of univariate Gaussian pdfs:

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2) = \prod_{f=0}^{F-1} \mathcal{N}(x_f; \mu_f, \sigma_f^2), \tag{19}$$

where v_f denotes the f-th entry of a vector $\mathbf{v}$.

Let $\mathcal{N}_c(x; \mu, \sigma^2)$ denote the proper complex Gaussian distribution for a random variable $x \in \mathbb{C}$ with mean $\mu \in \mathbb{C}$ and variance $\sigma^2 \in \mathbb{R}_+$. Its pdf is defined by:

$$\mathcal{N}_c(x; \mu, \sigma^2) = \frac{1}{\pi\sigma^2} \exp\left(-\frac{|x-\mu|^2}{\sigma^2}\right). \tag{20}$$

This distribution is circularly symmetric (i.e. invariant to a phase shift for x) if $\mu = 0$

A.2. Gamma distribution

Let $\mathcal{G}(x; a, b)$ denote the Gamma distribution for a random variable $x \in \mathbb{R}_+$ with shape and rate parameters $a > 0$ and $b > 0$ respectively. Its pdf is defined by:

$$\mathcal{G}(x; a, b) = \frac{b^a}{\Gamma(a)} x^{a-1} \exp(-bx), \tag{21}$$

where $\Gamma(\cdot)$ is the Gamma function.

A.3. Poisson distribution

Let $\mathcal{P}(x; \lambda)$ denote the Poisson distribution for a random variable $x \in \mathbb{N}$ with rate parameter $\lambda > 0$. Its pdf is defined by:

$$\mathcal{P}(x; \lambda) = \exp(-\lambda)\frac{\lambda^x}{x!}. \tag{22}$$

7. REFERENCES

[1] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[2] D.P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Int. Conf. Learning Representations (ICLR)*, 2014.

[3] M. Blaauw and J. Bonada, "Modeling and transforming speech using variational autoencoders.," in *Conf. of the Int. Speech Comm. Association (Interspeech)*, San Francisco, CA, 2016.

[4] C.C. Hsu, H.T. Hwang, Y.C. Wu, Y. Tsao, and H.M. Wang, "Voice conversion from non-parallel corpora using variational auto-encoder," in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, 2016, pp. 1–6.

[5] W.-N. Hsu, Y. Zhang, and J. Glass, "Learning latent representations for speech generation and transformation," in *Conf. of the Int. Speech Comm. Association (Interspeech)*, Stockholm, Sweden, 2017.

[6] K. Akuzawa, Y. Iwasawa, and Y. Matsuo, "Expressive speech synthesis via modeling expressions with variational autoencoder," in *Conf. of the Int. Speech Comm. Association (Interspeech)*, Hyderabad, India, 2018.

[7] F. Roche, T. Hueber, S. Limier, and L. Girin, "Autoencoders for music sound modeling: A comparison of linear, shallow, deep, recurrent and variational models," in *Sound and Music Computing Conference (SMC)*, Málaga, Spain, 2019.

[8] P. Esling, A. Chemla-Romeu-Santos, and A. Bitton, "Bridging audio analysis, perception and synthesis with perceptually-regularized variational timbre spaces," in *Int. Society for Music Information Retrieval Conf. (ISMIR)*, Paris, France, 2018.

[9] Y. Bando, M. Mimura, K. Itoyama, K. Yoshii, and T. Kawahara, "Statistical speech enhancement based on probabilistic integration of variational autoencoder and non-negative matrix factorization," in *IEEE Int. Conf. on Acoustics, Speech and Signal Process. (ICASSP)*, Calgary, Canada, 2018.

[10] S. Leglaive, L. Girin, and R. Horaud, "A variance modeling framework based on variational autoencoders for speech enhancement," in *IEEE Int. Workshop on Machine Learning for Signal Processing (MLSP)*, Aalborg, Denmark, 2018.

[11] L. Pandey, A. Kumar, and V. Namboodiri, "Monaural audio source separation using variational autoencoders," in *Conf. of the Int. Speech Comm. Association (Interspeech)*, Hyderabad, India, 2018.

[12] S. Leglaive, U. Şimşekli, A. Liutkus, L. Girin, and R. Horaud, "Speech enhancement with variational autoencoders and alpha-stable distributions," in *IEEE Int. Conf. on Acoustics, Speech and Signal Process. (ICASSP)*, 2019.

[13] K. Sekiguchi, Y. Bando, K. Yoshii, and T. Kawahara, "Bayesian multichannel speech enhancement with a deep speech prior," in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, 2018, pp. 1233–1239.

[14] L. Li, H. Kameoka, and S. Makino, "Fast MVAE: Joint separation and classification of mixed sources based on multichannel variational autoencoder with auxiliary classifier," in *IEEE Int. Conf. on Acoustics, Speech and Signal Process. (ICASSP)*, Brighton, UK, 2019.

[15] S. Leglaive, L. Girin, and R. Horaud, "Semi-supervised multichannel speech enhancement with variational autoencoders and non-negative matrix factorization," in *IEEE Int. Conf. on Acoustics, Speech and Signal Process. (ICASSP)*, Brighton, UK, 2019.

[16] C. Févotte and A.T. Cemgil, "Nonnegative matrix factorizations as probabilistic inference in composite models," in *European Signal Processing Conference (EUSIPCO)*, Glasgow, Scotland, 2009.

[17] C. Févotte, N. Bertin, and J.-L. Durrieu, "Nonnegative matrix factorization with the itakura-saito divergence: With application to music analysis," *Neural computation*, vol. 21, no. 3, pp. 793–830, 2009.

[18] D.D. Lee and H.S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, pp. 788–791, 1999.

[19] A. Ozerov and C. Févotte, "Multichannel nonnegative matrix factorization in convolutive mixtures for audio source separation," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 18, no. 3, pp. 550–563, 2010.

[20] N.Q. Duong, E. Vincent, and R. Gribonval, "Under-determined reverberant audio source separation using a full-rank spatial covariance model," *IEEE Trans. Audio, Speech, Language Process.*, vol. 18, no. 7, pp. 1830–1840, 2010.

[21] T. Gerber, M. Dutasta, L. Girin, and C. Févotte, "Professionally-produced music separation guided by covers," in *Int. Society for Music Information Retrieval Conf. (ISMIR)*, Porto, Portugal, 2012.

[22] P. Smaragdis, C. Fevotte, G. Mysore, N. Mohammadiha, and M. Hoffman, "Static and dynamic source separation using nonnegative factorizations: A unified view," *IEEE Signal Processing Magazine*, vol. 31, no. 3, pp. 66–75, 2014.

[23] D. Kounades-Bastian, L. Girin, X. Alameda-Pineda, S. Gannot, and R. Horaud, "A variational EM algorithm for the separation of moving sound sources," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, NJ, USA, 2015.

[24] S. Leglaive, R. Badeau, and G. Richard, "Multichannel audio source separation with probabilistic reverberation priors," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 12, pp. 2453–2465, 2016.

[25] S. Leglaive, R. Badeau, and G. Richard, "Student's t Source and Mixing Models for Multichannel Audio Source Separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 6, pp. 1150–1164, 2018.

[26] Y. Ephraim and D. Malah, "Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 6, pp. 1109–1121, 1984.

[27] A. Liutkus, R. Badeau, and G. Richard, "Gaussian processes for underdetermined source separation," *IEEE Transactions on Signal Processing*, vol. 59, no. 7, pp. 3155–3167, 2011.

[28] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "β-vae: learning basic visual concepts with a constrained variational framework.," in *Int. Conf. on Learning Representations (ICLR)*, 2017.

[29] Y. Jung, Y. Kim, Y. Choi, and H. Kim, "Joint learning using denoising variational autoencoders for voice activity detection," in *Conf. of the Int. Speech Comm. Association (Interspeech)*, Hyderabad, India, 2018.

[30] X. Li, L. Girin, and R. Horaud, "An EM algorithm for audio source separation based on the convolutive transfer function," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, NJ, USA, 2017.

[31] P. Smaragdis and J.C. Brown, "Non-negative matrix factorization for polyphonic music transcription," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, NJ, 2003.

[32] T. Virtanen, "Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 3, pp. 1066–1074, 2007.

[33] J.S. Garofolo, L.F. Lamel, W.M. Fisher, J.G. Fiscus, D.S. Pallett, N.L. Dahlgren, and V. Zue, "TIMIT acoustic phonetic continuous speech corpus," in *Linguistic data consortium*, 1993.

[34] J. Engel, C. Resnick, A. Roberts, S. Dieleman, D. Eck, K. Simonyan, and M. Norouzi, "Neural audio synthesis of musical notes with wavenet autoencoders," *arXiv preprint arXiv:1704.01279*, 2017.

[35] A.W. Rix, J.G. Beerends, M.P. Hollier, and A.P. Hekstra, "Perceptual evaluation of speech quality (PESQ): A new method for speech quality assessment of telephone networks and codecs," in *IEEE Int. Conf. on Acoustics, Speech and Signal Process. (ICASSP)*, 2001, pp. 749–752.

[36] R. Huber and B. Kollmeier, "PEMO-Q: A new method for objective audio quality assessment using a model of auditory perception," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 6, pp. 1902–1911, 2006.

[37] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio, "A recurrent latent variable model for sequential data," in *Advances in Neural Information Processing Systems*, 2015, pp. 2980–2988.

[38] S. Sra and I.S. Dhillon, "Generalized nonnegative matrix approximations with bregman divergences," in *Advances in Neural Information Processing Systems*, 2006, pp. 283–290.

[39] A. Cichocki, S. Cruces, and S. Amari, "Generalized alpha-beta divergences and their application to robust nonnegative matrix factorization," *Entropy*, vol. 13, no. 1, pp. 134–170, 2011.

MODELLING OF NONLINEAR STATE-SPACE SYSTEMS USING A DEEP NEURAL NETWORK

Julian D. Parker, Fabián Esqueda, André Bergner

Native Instruments GmbH.
Berlin, Germany
firstname.lastname@native-instruments.de

ABSTRACT

In this paper we present a new method for the pseudo black-box modelling of general continuous-time state-space systems using a discrete-time state-space system with an embedded deep neural network. Examples are given of how this method can be applied to a number of common nonlinear electronic circuits used in music technology, namely two kinds of diode-based guitar distortion circuits and the lowpass filter of the Korg MS-20 synthesizer.

1. INTRODUCTION

Virtual analog (VA) modelling is a well-established and active field of research within musical signal processing that focuses on the digital emulation of electrical or electro-mechanical systems used in music production. Previous VA research has studied a wide variety of music systems, such as analog filters and oscillators like those used in subtractive synthesis [1–5], guitar effects pedals [6–8], and guitar amplifiers [9], to name but a few examples. VA modelling can generally be separated into two broad categories, 'white-box' modelling and 'black-box' modelling.

White-box modelling, so called because it requires full knowledge of the structure of the device under study (e.g. via circuit schematics), focuses on deriving the underlying differential equations governing a system and then discretizing them to generate a numerical solution. For simple circuits this process can be performed manually, which typically allows for a tailored solution to the problem [10–12]. However, for more complicated systems this approach can quickly become unwieldy and the use of an automated general-purpose framework is generally preferred. Examples of these frameworks include state-space models [13, 14], the wave digital filter (WDF) formalism [15,16], and port-Hamiltonian systems [17].

Black-box techniques, on the other hand, focus on measuring the system which is being modelled, and then using these measurements to provide parameters or coefficients to a standard modelling structure. Prominent forms of black-box modelling include Volterra series [18], dynamic convolution [19, 20] and Wiener-Hammerstein models [21, 22]. Some work has focused on automatically tuning a hand-designed model system for a specific class of systems, e.g. a guitar amplifier, using measurements from a specific example of such a system [23, 24]. This kind of approach is commonly named 'grey-box' as it requires some insight into the construction of the system.

Recent applications of Machine Learning (ML), specifically Convolutional Neural Networks (CNNs), to the topic of VA modelling [25] are promising but computationally expensive and limited to systems with no autonomous or higher-order nonlinear behaviour. Other recent work has examined the modelling of nonlinear time-series data by using a compound neural network structure including an autoencoder to estimate the internal state of the process and then model the evolution within this inferred space [26]. Both of these approaches do not consider measurements inside the system, and work with only input/output data. They are broadly therefore in the black-box category.

In this paper we present a new structure consisting of a deep neural network, embedded within a discrete-time state-space system. We call this structure a State Trajectory Network (STN). We show how this structure can be trained to approximate a number of the common circuits of music electronics using not just input and output measurements, but also measurements of signals within the circuits.

This paper is structured as follows. In Sec. 2 we describe the principles behind the method. In Sec. 3 we describe the proposed Artificial Neural Network (ANN) structure, and discuss both how it can be trained to fit a system, and how it can be applied to process signals. In Sec. 4 we apply the technique to a number of circuits and discuss the results. Sec.5 gives concluding remarks.

2. METHOD

The state-space (also known as phase-space) approach is a powerful mathematical formalism that can describe any system which can be characterised by a system of ODEs [27]. In continuous-time, it can be written as:

$$\dot{\mathbf{x}}(t) = f(\mathbf{u}(t), \mathbf{x}(t)) \tag{1}$$

$$\mathbf{y}(t) = g(\mathbf{u}(t), \mathbf{x}(t)) \tag{2}$$

where $\mathbf{u}$ is the vector of inputs to the system, $\mathbf{y}$ is the vector of outputs and $\mathbf{x}$ are the 'states' of the system. This can also be written in a single equation form, by concatenating the states with the input or output:

$$\begin{matrix}\dot{\mathbf{x}}(t) \\ \mathbf{y}(t)\end{matrix} = f_a \begin{pmatrix}\mathbf{u}(t) \\ \mathbf{x}(t)\end{pmatrix}. \tag{3}$$

The function f_a therefore completely describes the behaviour of the system. Note that in cases where the state of a system is directly taken as the output, there is no need for any $\mathbf{y}$ variables. This is quite common in the types of systems we will be considering. The discrete-time analogue (given constant time-steps, where $n = \tau t$ with τ being the sampling interval) is given by:

$$\begin{matrix}\mathbf{x}_{n+1} \\ \mathbf{y}_n\end{matrix} = f_d \begin{pmatrix}\mathbf{u}_n \\ \mathbf{x}_n\end{pmatrix} \tag{4}$$

which allows the behaviour of the system to be calculated iteratively based on its inputs and previous states.

Much of the literature on VA modelling is concerned with deriving a set of equations in the form described by (3) which define the continuous time system of a particular circuit, and then using various methods to discretize it into the corresponding discrete-time state-space system to allow replication of its behaviour in a digital device like a computer. The discretization problem therefore becomes essentially the problem of transforming the function f into the function f_a.

2.1. Approximating f_d

If we have access to the system being modelled, we generally can also access the input and output signals $\mathbf{u}(t)$ and $\mathbf{y}(t)$. In many practical situations, e.g. when modelling an electrical circuit, we will also have access to the signals $\mathbf{x}(t)$ corresponding to the states of the system. If we feed the system with test signals $\mathbf{u}(t)$, and sample the state and output signals $\mathbf{x}(t)$ and $\mathbf{y}(t)$, respectively, we now have a large set of data-points describing the input and output relationship of the function f_d. If we want to replicate the behaviour of the system for arbitrary input, we can use this data to produce a new function $\hat{f}_d$ that approximates f_d to an appropriate degree of accuracy.

This is essentially a regression task, and could be tackled via any number of standard techniques. However, it is well suited to the application of an artificial neural network (ANN) which are known to be universal approximators, i.e. they are capable of approximating any continuous function of N real variables arbitrarily well [28, 29]. The system can then be simulated iteratively as usual, but using the approximated function $\hat{f}_d$ instead of an f_d derived analytically through discretization. The system can then be written as

$$\begin{matrix}\mathbf{x}_{n+1} \\ \mathbf{y}_n\end{matrix} = \hat{f}_d \begin{pmatrix}\mathbf{u}_n \\ \mathbf{x}_n\end{pmatrix}. \tag{5}$$

The choice of what quantities to take as the states of the system is not as strict as it might first appear. Whilst the real states of the system are the quantities that store energy, and hence information (e.g. capacitors), other quantities in the system will be related to those states by a mapping (linear in the best case, but likely nonlinear). This means that as long as we have sufficient measured quantities compared to the number of states of the system, we should be able to learn its dynamics. The formal upper bound on the number of independent measurements needed is given by the Whitney embedding theorem as $2m + 1$, where m is the number of states [30]. Additionally Taken's theorem [31] allows us to replace some (or all) of these measurements with time-delayed versions of another measurement. In practice, the dynamics in this transformed state-space may be more complicated, and hence it is preferable to use measurements as close as possible to the real states of the system.

3. NEURAL NETWORK ARCHITECTURE

The formulation of the problem in state-space terms is advantageous, as it means that the function f_d we need to approximate is purely a static, i.e. memoryless, mapping. Consequently, network structures with embedded memory, such as recurrent neural networks (RNNs) [32], in particular LSTM networks [33] or

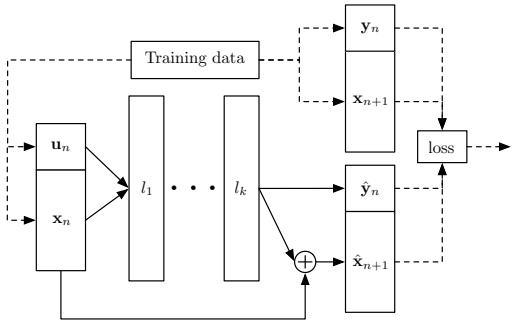


Figure 1: Proposed network structure, as it appears during training

echo state networks [34], which are standard candidates for sequence modelling, are unnecessary. The proposed method is also distinct from autoregressive modelling which predicts the next (often scalar) output based on a sequence of past observations [35], e.g. the celebrated WaveNet architecture [36]. Instead of considering only the input-output behavior of the system, we map from the state space into itself. Speaking in terms of dynamical systems, this means we are learning the piece-wise flow along the trajectory of the system [37]. The input $\mathbf{u}$ can be considered as a parameter that influences this flow.

The core of the network is a Multilayer Perceptron (MLP), i.e. series of k densely connected layers with an activation function. The number of layers and the layer width is tuned to suit the particular system being modelled, with small systems potentially needing only small networks. The activation function can be changed to fit the system, but generally saturating nonlinearities such as $\tanh$ produce good results. This intuitively makes sense, as the type of systems we are examining - electronic circuits, generally contain nonlinearities of the saturating type. Simpler activations such as Rectified Linear Units (ReLU) can be used, with the caveat that the size of the network will then generally need to be larger.

The states $\mathbf{x}_n$ and their values at the new time step $\mathbf{x}_{n+1}$ are likely to be closely related. We therefore structure this part of the network in a residual fashion using a skip-layer connection. Skip-layer connections have been successfully applied in different domains [36, 38]. The implication of this is that the network is learning a residual of the state signals compared to their previous value, rather than the new values directly. This consistently improved training speed and accuracy in the case of the considered systems.

The proposed network structure is shown in Fig. 1. As the network is able to iteratively move along the trajectories of the learned state-space, we call this structure a State Trajectory Network (STN).

3.1. Training

In initial experiments with the structure, a Mean Squared Error (MSE) loss function was used for training:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2, \tag{6}$$

where Y and $\hat{Y}$ are respectively the desired output and the actual output of the neural network [1], and n is the size of the training batch.

Using this loss function, the network structure showed some difficulty with training accurately. Investigation showed that the cause of this difficulty was large variance in the state residuals needed to fit the training data. This meant that MSE was prioritising the accuracy of the large residuals, and often leaving the smaller residuals completely incorrect. To combat this, a normalized version of MSE was used for the initial 10 epochs of the training:

$$\text{MSE}_{\text{norm}} = \frac{1}{n}\sum_{i=1}^{n}\frac{(Y_i - \hat{Y}_i)^2}{Y_i^2}. \tag{7}$$

This seemed to provide a better foundation for further training using MSE, resulting in better overall accuracy. The Nesterov Adam (Nadam) optimizer was used, with learning rate and other meta-parameters set as recommended by Dozat [39]. Training and verification of the models was conducted using Keras [40] with TensorFlow [41] as a backend.

3.2. Application to sound synthesis and processing

Fig. 2 shows the structure of the network when used for processing or generating audio signals. The feedback connection representing the iterative calculation of the modelled system behaviour is shown. Also shown is an additional gain element h_τ used to scale the state residual calculated by the network. This element highlights another advantage of skip-level structure. Having direct access to the residuals is useful, as the residuals are entirely responsible for the time-evolution of the modelled system. Multiplying the residuals by an arbitrary gain allows us to effectively alter the time-scale of the simulated system. With some caveats, this allows the model to be run at sampling periods other than the one used to train the model. This gain can be defined as:

$$h_\tau = \frac{\tau_p}{\tau_t} = \frac{F_t}{F_p}, i \tag{8}$$

where τ_p, τ_t are the time-steps used for processing and for training respectively, and equivalently F_p and F_s are the sampling frequencies used for processing and for training.

Care must be taken when using this facility to run the system at a different sampling frequency than the one which it was trained for. Whilst the time-scale will be correctly altered, this does not make the system equivalent to one trained at the new sampling frequency. Running the system at a lower sampling frequency than used for the training raises the possibility of aliasing being introduced by elements of the learned behaviour that exceed the new Nyquist limit. Running the system at a higher sampling rate than it was trained at poses less risks, although the modelled system will potentially lack high-frequency behaviours that might have been present if the system had been trained at a higher sampling frequency. Oversampling compared to audio-rate is recommended in most situations, as the system does not utilize any specific anti-aliasing and hence will produce aliasing in the same situations as a normal nonlinear state-space system.

[1] Note that the neural network output Y should not to be confused with $\mathbf{y}$, the output of the system being modelled.

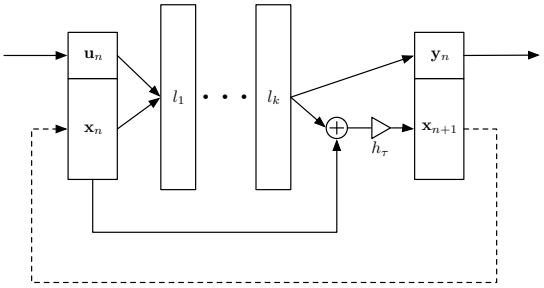


Figure 2: Proposed network structure in form used for processing signals.

4. MODELLING MEASURED SYSTEMS

In order to test the performance of the proposed method, a number of test circuits were built. A measurement signal was applied to the circuit, and the state and output signals recorded using an analog-to-digital converter (ADC). Unity gain op-amp buffers were used to isolate the measurement points from the load of the ADC. In some situations, node voltages were recorded instead of differential voltages across capacitors (i.e. the required states). In these cases, the differential voltage was recovered in post-processing by linear combination with other recorded node voltages.

For each circuit several models were trained, and then two selected for presentation here - one smaller and one larger network. All of the data referenced below is available on the accompanying website[2], including verification signals and model outputs.

4.1. Measurement Signal Design

Theoretically, any signal can be used to generate the measurements used to train the network. In practice, some important considerations have to be made. In contrast to other black-box modelling techniques, measurements with varying frequency input, e.g. a sine-sweep, are not strictly necessary to capture the behaviour of the system under consideration. This is because behaviour at all time-scales is captured in the learnt state trajectories. However, the chosen measurement signal does have a number of important impacts on the process.

The measurement signal defines the subset of the state space which will be sampled. In complicated systems, particular inputs could potentially be needed in order to access particular parts of the state space. A simple example of this could be a resonant filter circuit. In this case, an input signal at the resonant frequency would excite the circuit much more strongly than signals at other frequencies, potentially revealing nonlinear behaviours that only apply at high state magnitudes.

Care must also be taken to restrict the upper frequency of the measurement signal to significantly below the Nyquist frequency at the sampling rate used for the measurements. This is because the process of sampling the state signals and outputs is bandlimited. Components that exceed the Nyquist will be removed during the sampling process. Hence, harmonics produced in the real system may be lost. This means that state signals recorded in response to too high of a frequency will no longer accurately reflect the true state-space trajectory. This leads to measurements which appear to violate the constraint that each point in space space can correspond

[2] https://github.com/julian-parker/DAFX-NLML

to only one trajectory (effectively there are now further states in the bandlimiting filter of the sampling apparatus, which are not reflected in our system model).

In practice, the selection of a measurement signal is a balance between these two constraints, and can be tailored to the system under consideration. Typical input signals for the system can work well (e.g. guitar playing if modelling a distortion pedal), as can carefully specified sweeps. In this paper we used 10-second logarithmic sine-sweeps combined with low-level (-20dB) white noise. The noise was bandlimited to 22 kHz. This combined signal was then increased in amplitude linearly from zero to unity over half of the length of the measurement signal, in order to ensure the capture of sufficient data-points near the origin of the state-space. The minimum frequency of the sweep was taken as 20 Hz, and the maximum frequency was chosen to be 10 kHz in order to cover a sufficient amount of the state-space without corrupting the data with fictional trajectories. A spectrogram of the measurement signal is shown in Fig. 3. A sampling rate of 192 kHz was used for measurement and training.

In all the examples presented here, training was done over 300 epochs, with a batch size of 256. Due to the small size of the networks under consideration, a GPU is not needed for the training process (and is in-fact slower than training on a CPU with SIMD functionality).

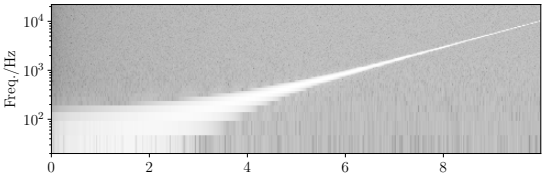


Figure 3: Spectrogram of sine-sweep and noise signal used to generate training data from the systems.

For time-domain verification of the results, two signals were chosen - a 500 Hz sawtooth wave, and a short passage of electric guitar. Short extracts from the sawtooth and guitar signals are shown in Fig. 4. The guitar signal contains a variety of signal levels, and the sawtooth was chosen to have a peak voltage of 2 V, so as to stress the nonlinear behaviours in the circuits. Verification was primarily conducted in the time-domain, but a 1 kHz sinusoid with a a peak voltage of 2 V was also used to check frequency-domain behaviour of the models.

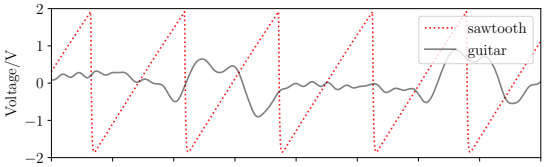


Figure 4: Extracts of the signals used for verification of the modelled circuits

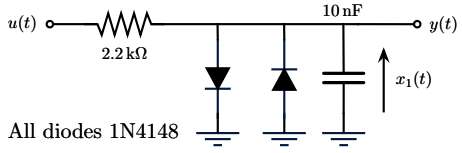


Figure 5: Schematic for the first-order diode clipper.

4.2. First-order diode clipper

The single-capacitor diode clipper, shown in Fig. 5, is a common object of examination in the VA literature, starting from the work of Yeh [10]. For small input signals, (i.e. peak voltages below approximately 0.6 V) the current flowing through the diodes is close to zero and the circuit reduces to a simple RC lowpass filter. For larger signal values the diodes will cause the output to saturate and the circuit becomes a nonlinear lowpass filter, with its instantaneous cutoff frequency rising as the circuit is driven harder.

The circuit looks simple, but has proven somewhat challenging for white-box techniques due to its inherent stiffness. A conventional discretization of this circuit will generally need to employ an implicit numerical scheme, with iterations necessary at each time-step in order to find the correct state value [42]. In this circuit, the output y and the single state x_1 are the same quantity, i.e. the voltage across the single capacitor.

4.2.1. Training

Selecting training data is simple in this case. There is only one capacitor and hence state, and the value of this state is the output of the system. Fig. 6 shows a visualization of the training data. We can clearly see the saturating behaviour of the system as the prominent dark s-shape. Whilst the shape of the space may seem relatively simple, smaller networks had trouble fitting the curve shape exactly - even when using saturating activation functions.

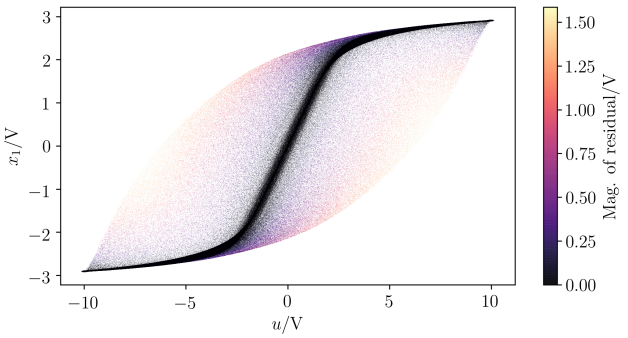


Figure 6: Visualization of a subset of the state-space data used to train diode clipper model.

4.2.2. Results

The MSE for the chosen models, calculated over the entirety of the verification signals is:

Model struct.	Sawtooth	Guitar
2x128 ReLU	0.237 mV	0.276 mV
2x8 tanh	0.079 mV	0.135 mV

Fig. 7 shows extracts from the verifications signals, processed with the chosen models and with the real circuit. The match appears to be good on both the sawtooth and guitar signal. Surprisingly, the larger ReLU based model shows lower accuracy. The frequency domain verification also shows a very close match to the real output, with odd and even harmonic levels matched very closely apart from a small amount of error in high-frequency even harmonics. The model output is indistinguishable from the output of the real circuit in casual listening.

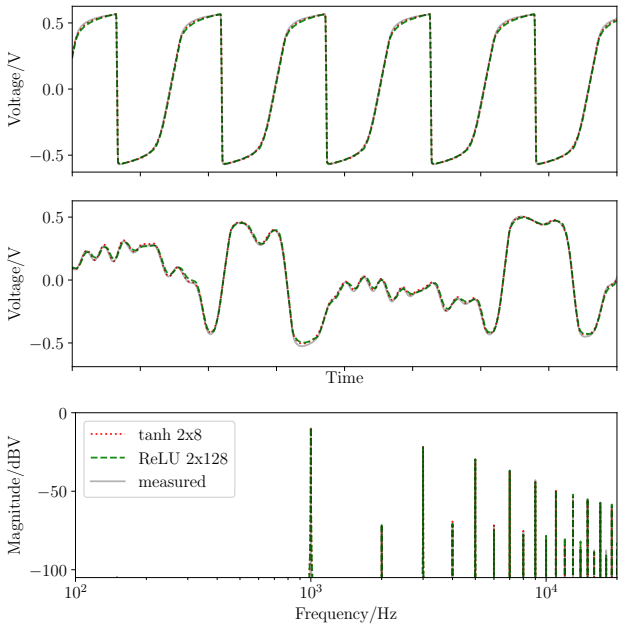


Figure 7: Results of processing the verification signals with two different trained diode-clipper models, compared to the output measured from the real circuit.

4.3. Second-order diode clipper (Boss DS-1)

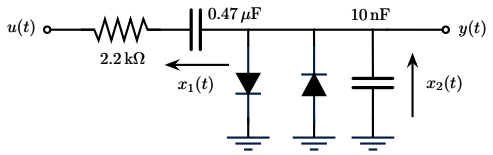


Figure 8: Schematic for the second-order diode clipper. Figure adapted from [43].

The next circuit considered in this study is the second-order diode clipper shown in Fig. 8. This circuit adds a capacitor between the input and the diodes, and is a simplified version of the core distortion section in the well-known Boss DS-1 pedal [43]. In this study we have omitted the preceding op-amp gain stage and the subsequent tone control. The behaviour of this circuit is similar to that of the first-order clipper, but more complex. The series capacitor introduces additional highpass filtering which in turn can cause asymmetry in the overall clipping behaviour [44].

This circuit exhibits stiffness properties similar to those of the first-order clipper when discretized using standard numerical techniques. The voltages across both capacitors were chosen as the states of the circuits. The first state, x_1, was retrieved by measuring the node between the input resistor and series capacitor, and computing its difference with x_2, which was measured directly at the output node y.

4.3.1. Training

Again, it is valuable to visualize the measured training data to gain insight into the system. This can be seen in Fig. 9, where we see a projection of the data onto the input u vs x_2 plane. We can still see the characteristic s-shape of the single diode-clipper. If we view the data in 3d, with x_1 providing the third dimension, we see that the s-shape is actually a manifold rotated around the x_2 axis. Training proceeded as in the single-capacitor case.

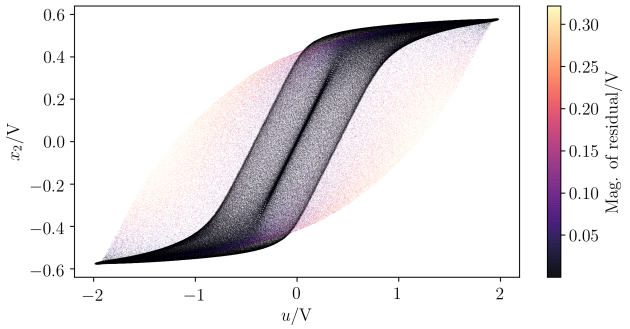


Figure 9: Visualization of state-space data used to train DS-1 model.

4.3.2. Results

The MSE for the models, calculated over the entirety of the verification signals is:

Model struct.	Sawtooth	Guitar
2x128 ReLU	0.082 mV	0.191 mV
3x4 tanh	0.232 mV	2.67 mV

Again, we can see extracts from the verification signals in Fig. 10. The match is again very good, with a small advantage in accuracy from the larger ReLU based model. The frequency domain results show a very close match for odd harmonics, with even harmonics being a worse fit. The ReLU model in particular seems to produce stronger even harmonics than needed. The model output is again indistinguishable from the output of the real circuit in casual listening.

4.4. Sallen-Key Filter (Korg MS-20 REV2)

The last system examined is an adapted version of the lowpass filter circuit from the Korg MS-20, a classic monophonic analog

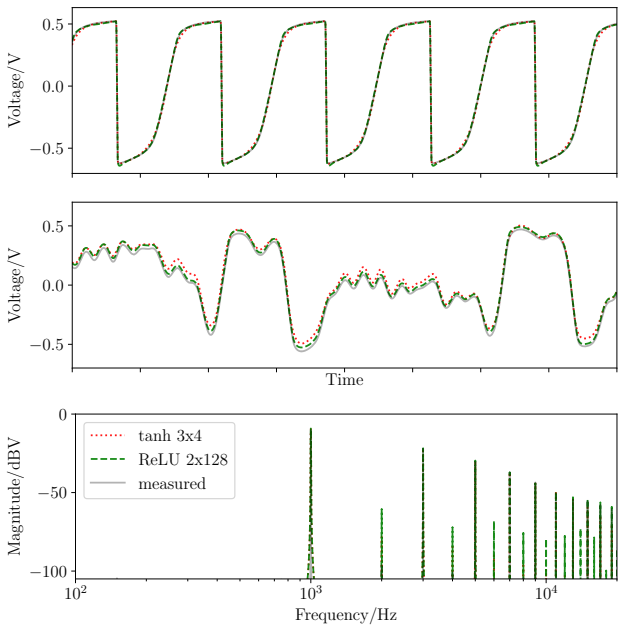


Figure 10: Results of processing the verification signals with two different trained DS-1 models, compared to the output measured from the real circuit.

synthesiser released by Korg in 1978. The particular version examined in this study corresponds to the second revision of the circuit, commonly known as 'REV2'. This filter is a Sallen-Key topology, and utilizes operational transconductance amplifiers (OTAs) as current-controlled gain elements used to set the cutoff of the filter. An op-amp-based non-inverting amplifier with clipping diodes is used in the feedback path as a resonance control [45]. These diodes are responsible for the characteristic distorted sound of the filter. The schematic for this circuit is shown in Fig. 13. For clarity, the cutoff control section has been omitted and ideal buffers have been used to represent the transistor buffers contained within the LM13700 OTAs [46].

This circuit was chosen for modelling as it exhibits strongly nonlinear self-oscillatory behaviour. This behaviour would not be possible to model using existing black-box techniques. For the purpose of this work, the parametric elements were set to fixed values. This meant feeding a constant control current I_{ctl} to the OTAs to fix the cutoff, and fixing the resonance potentiometer at a particular gain. As in the DS-1 circuit, x_1 was computed by measuring the voltages at the outputs of the first OTA (IC 1) and the feedback amplifier, and computing their difference in post-processing. The second state of the circuit x_2 is simply the voltage difference between the output of the second OTA (IC 2) and ground. Since this node is connected to y by a unit-gain buffer, it can be measured directly at the output filter. For this circuit, the peak amplitude of the input sweep as seen by the circuit was adjusted to be 1.1 V.

4.4.1. Training

The MS-20 filter exhibits much more complex behaviour than the simpler clipper circuits. A plot of the training data is shown in Fig. 11, which again shows a projection of the data onto the u vs x_2 plane. The 3-dimensional structure of the state space is already quite visible in this plot. Two main components can be seen, a saturating s-shaped manifold similar to that seen in the clipper circuits, along with a closed orbit corresponding to the self-oscillating behaviour.

Despite the more complex seeming behaviour of the circuit, training of the model networks proceeded more quickly than with the clipper circuits.

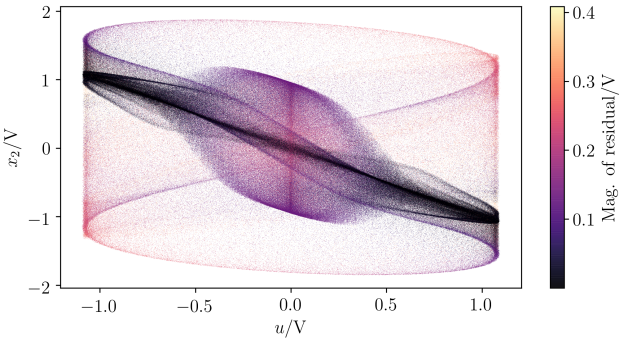


Figure 11: Visualization of state-space data used to train MS-20 filter model.

4.4.2. Results

The MSE for the chosen models, calculated over the entirety of the verification signals is given by:

Model struct.	Sawtooth	Guitar
2x32 ReLU	2.01 mV	684 mV
3x4 tanh	1.29 mV	794 mV

The error values are somewhat higher than seen in the case of the clipper circuits, especially in the case of the guitar verification signal. As can be seen in Fig. 12, the behaviour of the circuit seems to be replicated rather well by the models with self-oscillation occurring at the correct frequency and being damped with increased input level as expected. The major difference between the measured and modelled results in the case of the guitar signal appears to be the phase of the self-oscillation. This can be explained by the fact that even very small errors in the learned state-space trajectory will accumulate to produce a phase difference when the dynamics are dominated by free self-oscillation. The phase of the oscillation isn't important perceptually in this case, so we conclude that the poor MSE numbers do not imply a poor model. Listening to the verification signals processed through the models confirms this, as they appear indistinguishable from the verification signals processed by the real circuit. The frequency-domain results also show a very close correspondence between the models and the real circuit, with small differences seen only in some higher harmonics.

4.5. Computational Complexity

We computed the number of floating point operations by counting all multiplications, additions, and nonlinear function evaluations in the network. For the activations functions we assume three operations for ReLUs (abs, add, mul) and an average of 30 operations for tanh. Assuming that the models are run at the same sampling

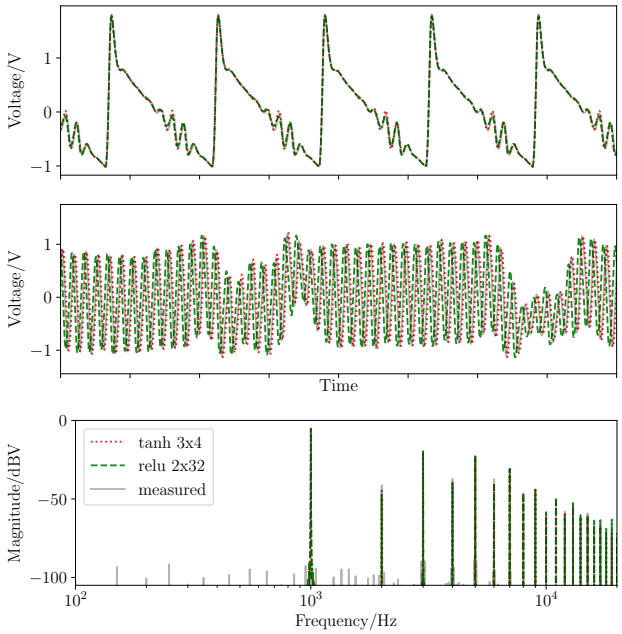


Figure 12: Results of processing the verification signals with two different MS20 filter models, compared to the output measured from the real circuit for the same input.

frequency used for training, 192 kHz, we can then extrapolate this into a value in floating point operations per second (FLOPS):

Model	ops/sample	GFLOPS
DS-1 2x128 ReLU	34822	6.7
Clipper 2x128 ReLU	34307	6.6
MS-20 2x32 ReLU	2620	0.50
Clipper 2x8 tanh	686	0.13
MS-20 3x4 tanh	524	0.10
DS-1 3x4 tanh	524	0.10

The values lie well within the bounds of real-time operation on a modern computer. However, the values should only be taken as a rough guideline for performance. The actual performance of the algorithm will depend strongly on the processor architecture used.

5. CONCLUSIONS

In this work we introduced a new technique for modelling the behaviour of nonlinear state-space systems, using a discrete-time state-space system with an embedded neural network. The network learns trajectories in state-space from measurements in a residual manner, and can be used to reproduce these trajectories in response to arbitrary input. We call the network structure a State Trajectory Network (STN). We showed how this structure can be used to accurately model the behaviour of a number of nonlinear circuits, using measurements from within the circuit to train the model. We also showed that the produced models are of sufficiently low computational complexity to be run for real-time audio usage.

Future work will concentrate on extending the STN to work with larger systems, and those with parametric control. We believe the STN may be an interesting structure for application to the modelling of stateful systems outside of the virtual-analog domain, and also intend to investigate this avenue.

6. ACKNOWLEDGMENTS

Many thanks to Nikolaj Andersson for providing the guitar recording used for evaluation in this work, as well as participating in a previous very early incarnation of the research presented here.

7. REFERENCES

[1] T. Stilson and J. Smith, "Alias-free digital synthesis of classic analog waveforms," in *Proc. Int. Computer Music Conf.*, Hong Kong, Aug. 1996, pp. 332–335.

[2] A. Huovilainen, "Non-linear digital implementation of the Moog ladder filter," in *Proc. 7th Int. Conf. Digital Audio Effects (DAFx-04)*, Naples, Italy, Oct. 2004, pp. 61–64.

[3] V. Välimäki, J. Pekonen, and J. Nam, "Perceptually informed synthesis of bandlimited classical waveforms using integrated polynomial interpolation," *J. Acoust. Soc. Am.*, vol. 131, no. 1, pp. 974–986, Jan. 2012.

[4] F. Fontana and M. Civolani, "Modeling of the EMS VCS3 voltage-controlled filter as a nonlinear filter network," *IEEE Trans. Audio, Speech, Language Process.*, vol. 18, no. 4, pp. 760–772, Apr. 2010.

[5] M. Rest, J. D. Parker, and K. J. Werner, "WDF modeling of a Korg MS-50 based non-linear diode bridge VCF," in *Proc. 20th Int. Conf. Digital Audio Effects (DAFx-17)*, Edinburgh, UK, Sept. 2017, pp. 145–141.

[6] R. C. D. de Paiva, S. D'Angelo, J. Pakarinen, and V. Välimäki, "Emulation of operational amplifiers and diodes in audio distortion circuits," *IEEE Trans. Circ. Syst. II: Express Briefs*, vol. 59, no. 10, pp. 688–692, Oct. 2012.

[7] F. Eichas, M. Fink, M. Holters, and U. Zölzer, "Physical modeling of the MXR Phase 90 guitar effect pedal," in *Proc. 17th Int. Conf. Digital Audio Effects (DAFx-14)*, Erlangen, Germany, Sept. 2014, pp. 153–158.

[8] B. Holmes and M. van Walstijn, "Physical model parameter optimisation for calibrated emulation of the Dallas Rangemaster treble booster guitar pedal," in *Proc. 19th Int. Conf. Digital Audio Effects (DAFx-16)*, Brno, Czech Republic, Sept. 2016, pp. 47–54.

[9] W. R. Dunkel, M. Rest, K. J. Werner, M. J. Olsen, and J. O. Smith III, "The Fender Bassman 5F6–A family of preamplifier circuits–A wave digital filter case study," in *Proc. 19th Int. Conf. Digital Audio Effects (DAFx-16)*, Brno, Czech Republic, Sept. 2016, pp. 263–270.

[10] D. T. Yeh, J. S. Abel, and J. O. Smith, "Simulation of the diode limiter in guitar distortion circuits by numerical solution of ordinary differential equations," in *Proc. 10th Int. Conf. Digital Audio Effects (DAFx-07)*, Bordeaux, France, Sept. 2007, pp. 197–204.

[11] S. D'Angelo and V. Välimäki, "Generalized Moog ladder filter: Part II–explicit nonlinear model through a novel delay-free loop implementation method," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 22, no. 12, pp. 1873–1883, Dec. 2014.

[12] F. Esqueda, H. Pöntynen, J. D. Parker, and S. Bilbao, "Virtual analog models of the Lockhart and Serge wavefolders," *Appl. Sci.*, vol. 7, no. 12, Dec. 2017.

[13] D. T. Yeh, J. S. Abel, and J. O. Smith, "Automated physical modeling of nonlinear audio circuits for real-time audio effects—Part I: Theoretical development," *IEEE Trans. Audio, Speech Lang. Process.*, vol. 18, no. 4, pp. 728–737, May 2010.

[14] M. Holters and U. Zölzer, "A generalized method for the derivation of non-linear state-space models from circuit schematics," in *Proc. European Signal Processing Conference (EUSIPCO-15)*, 2015, pp. 1073–1077.

[15] G. De Sanctis and A. Sarti, "Virtual analog modeling in the wave-digital domain," *IEEE Trans. Audio Speech Lang. Process.*, vol. 18, no. 4, pp. 715–727, May 2010.

[16] K. J. Werner, *Virtual analog modeling of audio circuitry using wave*

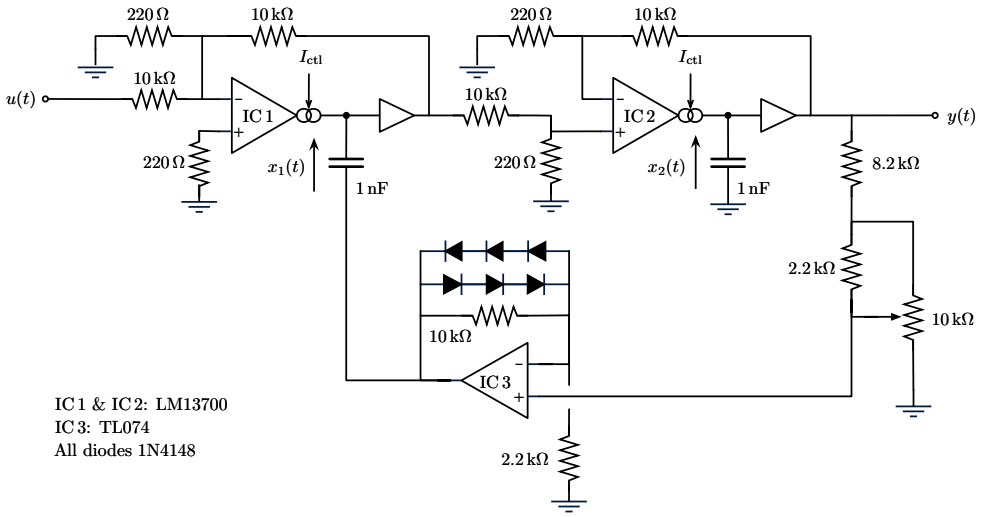


Figure 13: Simplified schematic of the Korg MS–20 filter, adapted from [46].

digital filters, Ph.D. thesis, Stanford University, Stanford, CA, USA, Dec. 2016.

[17] A. Falaize and T. Hélie, "Passive guaranteed simulation of analog audio circuits: A port-Hamiltonian approach," *Appl. Sci.*, vol. 6, no. 10, Apr. 2016.

[18] T. Hélie, "On the use of Volterra series for real-time simulations of weakly nonlinear analog audio devices: Application to the Moog ladder filter," in *Proc. 9th Int. Conf. Digital Audio Effects (DAFx-06)*, Montreal, Canada, Sept. 2006, pp. 7–12.

[19] M. J. Kemp, "Analysis and simulation of non-linear audio processes using finite impulse responses derived at multiple impulse amplitudes," in *Proc. Audio Eng. Soc. 106th Conv.*, Munich, Germany, May 1999.

[20] A. Primavera, S. Cecchi, L. Romoli, M. Gasparini, and F. Piazza, "Approximation of dynamic convolution exploiting principal component analysis: Objective and subjective quality evaluation," in *Proc. Audio Eng. Soc. 133th Conv.*, San Francisco, USA, Oct. 2012.

[21] A. Novák, L Simon, F. Kadlec, and P. Lotton, "Nonlinear system identification using exponential swept-sine signal," *IEEE Trans. Instrum. Meas.*, vol. 59, no. 8, pp. 2220–2229, Oct. 2009.

[22] F. Eichas and U. Zölzer, "Black-box modeling of distortion circuits with block-oriented models," in *Proc. 19th Int. Conf. Digital Audio Effects (DAFx-16)*, Brno, Czech Republic, Sept. 2016, pp. 39–45.

[23] F. Eichas, S. Möller, and U. Zölzer, "Block-oriented gray box modeling of guitar amplifiers," in *Proc. 20th Int. Conf. Digital Audio Effects (DAFx-17)*, Edinburgh, UK, Sept. 2017, pp. 184–191.

[24] C. Kemper, "Musical instrument with acoustic transducer," Patent No. US 8,796,530, 12 June 2008.

[25] E.-P. Damskägg, L. Juvela, E. Thuillier, and V. Välimäki, "Deep learning for tube amplifier emulation," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Brighton, UK, May 2019.

[26] D. Masti and A. Bemporad, "Learning nonlinear state-space models using deep autoencoders," in *Proc. 57th IEEE Conf. Dec. Control (CDC 2018)*, Miami, FL, USA, Dec. 2018, pp. 3862–3867.

[27] E. R. Scheinerman, *Invitation to dynamical systems*, Dover Publications, 1996.

[28] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, 1989.

[29] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[30] H. Whitney, "Differentiable manifolds," *Annals of Mathematics*, pp. 645–680, 1936.

[31] F. Takens, "Detecting strange attractors in turbulence," in *Dynamical systems and turbulence*, pp. 366–381. Springer, 1981.

[32] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, MIT press, 2016.

[33] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[34] H. Jaeger, "Adaptive nonlinear system identification with echo state networks," in *Advances in Neural Inf. Proc. Systems*, 2003, pp. 609–616.

[35] R. J. Frank, N. Davey, and S. P. Hunt, "Time series prediction and neural networks," *Journal of intelligent and robotic systems*, vol. 31, no. 1-3, pp. 91–103, 2001.

[36] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[37] J. M. Ginoux, *Differential geometry applied to dynamical systems*, vol. 66, World Scientific, 2009.

[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. 29th IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR 2016)*, Las Vegas, NV, USA, 2016, pp. 770–778.

[39] T. Dozat, "Incorporating Nesterov momentum into Adam," in *Proc. 4th Int. Conf. Learn. Repres. (ICLR 2016)*, San Juan, Puerto Rico, May 2016.

[40] F. Chollet et al., "Keras," `https://keras.io`, 2015.

[41] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., "Tensorflow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Oper. Sys. Des. Implement. (OSDI'16)*, 2016, pp. 265–283.

[42] F. G. Germain, "Fixed-rate modeling of audio lumped systems: A comparison between trapezoidal and implicit midpoint methods," in *Proc. 20th Int. Conf. Digital Audio Effects (DAFx-17)*, Edinburgh, United Kingdom, Sept. 2017, pp. 168–175.

[43] Roland Corp., "*Boss DS-1 Service Notes*," Dec. 1994.

[44] D. T. Yeh, J. S. Abel, and J. O. Smith, "Simplified, physically-informed models of distortion and overdrive guitar effects pedals," in *Proc. 10th Int. Conf. Digital Audio Effects (DAFx-07)*, Sept. 2007, pp. 189–196.

[45] T. E. Stinchcombe, "A study of the Korg MS10 and MS20 filters," Aug. 2006.

[46] Korg Inc., "*MS-20 Monophonic Synthesizer Service Manual*," 1978.

REAL-TIME BLACK-BOX MODELLING WITH RECURRENT NEURAL NETWORKS

*Alec Wright, Eero-Pekka Damskägg, and Vesa Välimäki**

Acoustics Lab, Department of Signal Processing and Acoustics
Aalto University
Espoo, Finland
alec.wright@aalto.fi

ABSTRACT

This paper proposes to use a recurrent neural network for black-box modelling of nonlinear audio systems, such as tube amplifiers and distortion pedals. As a recurrent unit structure, we test both Long Short-Term Memory and a Gated Recurrent Unit. We compare the proposed neural network with a WaveNet-style deep neural network, which has been suggested previously for tube amplifier modelling. The neural networks are trained with several minutes of guitar and bass recordings, which have been passed through the devices to be modelled. A real-time audio plugin implementing the proposed networks has been developed in the JUCE framework. It is shown that the recurrent neural networks achieve similar accuracy to the WaveNet model, while requiring significantly less processing power to run. The Long Short-Term Memory recurrent unit is also found to outperform the Gated Recurrent Unit overall. The proposed neural network is an important step forward in computationally efficient yet accurate emulation of tube amplifiers and distortion pedals.

1. INTRODUCTION

Virtual analog modelling is an active area of research, which seeks to create software that can accurately emulate popular music hardware, such as instruments, audio effects or amplifiers [1]. Nonlinear systems with memory, such as guitar amplifiers and distortion pedals are particularly challenging to emulate [2, 3]. Generally, the approaches to virtual analog modelling fall into three categories, "white-box" [2, 4, 5, 6], "grey-box" [7, 8], and "black-box" [9, 10] modelling. This paper is concerned with black-box modelling of tube amplifiers and distortion pedals using a recurrent neural network (RNN).

This is a very timely topic, as the first attempts to model nonlinear audio circuits with a Long Short-Term Memory (LSTM) neural network were published last year [11, 12]. Schmitz and Embrechts used a hybrid neural network consisting of a convolutional layer in front of an RNN [12]. Zhang *et al*. tested tube amplifier modelling using an LSTM neural network with many hidden layers but only a few units per layer. They reported that the sound quality of the emulation was not good enough, as there were clear audible differences with respect to the real amplifier [11].

In recent works [13, 14], we adapted the WaveNet convolutional neural network to model nonlinear audio circuits such as tube amplifiers and distortion pedals. In [14] it was shown that the WaveNet model of several distortion effects was capable of running in real time. The resulting deep neural network model, however, was still fairly computationally expensive to run.

In this paper, we propose an alternative black-box model based on an RNN. We demonstrate that the trained RNN model is capable of achieving the accuracy of the WaveNet model, whilst requiring considerably less processing power to run. The proposed neural network, which consists of a single recurrent layer and a fully connected layer, is suitable for real-time emulation of tube amplifiers and distortion pedals.

The rest of this paper is organized as follows. Section 2 discusses the two nonlinear circuits, which are used as target devices in this study, as well as the creation of the training data. Section 3 introduces the neural network architecture that we use for modelling. Section 4 focuses on the training of the RNNs. The real-time implementation of the RNN is presented in Section 5. Section 6 reports experiments and comparisons we have conducted to validate this work. Section 7 concludes the paper.

2. MODELLED DEVICES

Two commonly used types of nonlinear audio circuits are distortion pedals and guitar amplifiers. As such, for this study, we chose to model the Electro-Harmonix Big Muff Pi distortion/fuzz pedal and the Blackstar HT-1 combo guitar amplifier.

The Big Muff is a famous guitar pedal, whose first version was released by Electro-Harmonix in 1969 [15]. Since then, numerous versions of the pedal have appeared, each with slight differences to the original circuit. The Big Muff is known for its heavily distorted sound, which is produced by its two cascaded diode clipping stages. The pedal has a "sustain" knob for controlling the pre-gain, i.e. the gain applied to the signal before the clipping stages, a "volume" knob for controlling the post-gain, as well as a "tone" knob for controlling the shape of the filter in the tone stage. Several researchers have studied the digital modelling of the Big Muff distortion pedal prior to this work [16, 17, 14].

The Blackstar HT-1 is a small 1-Watt vacuum tube amplifier [18]. It has two channels: a high-gain and a low-gain channel. In this work, the high-gain channel was used, as it introduces more distortion to the signal. The amplifier has an unconventional tone stage. The "ISF" tone knob on the amplifier allows for continuous shifting between two distinct tone settings, which the manufacturer describes as the "American" and "British" tones. The amplifier also has a "gain" knob and a "volume" knob, which control the pre-gain and the post-gain respectively.

* This research belongs to the activities of the Nordic Sound and Music Computing Network—NordicSMC (NordForsk project number 86892).

2.1. Training Data

One issue which is common to many neural network applications is the collection of training data. Often this is a very labour intensive process, as a large collection of training examples needs to be gathered and labelled. For this application the process is relatively straightforward, as training data can be generated simply by inputting audio signals to each of the devices being modelled. The resulting output can then be recorded from the devices and used as the ground truth.

The input audio files were taken from the guitar and bass guitar datasets[1] [2] described in [19, 20], respectively. The full dataset used consists of 8 minutes and 10 seconds of audio. The dataset was split into a training set of 5 min and 42 s, a validation set of 1 min and 24 s and a test set of 1 min and 4 s. The audio was split so that each data subset contained approximately equal proportions of guitar and bass recordings. All of the training data audio used during this study was recorded at a sampling rate of 44.1 kHz.

The recording was carried out using a MOTU UltraLite-mk3 USB audio interface. One output of the audio interface was connected to the input of the device being measured. The output of the device being measured was recorded by connecting it to one of the inputs of the audio interface. The direct signal coming out of the audio interface was also measured by connecting one of its outputs to one of its inputs. The test audio was then output from both interface output channels and recorded through both input channels. The recorded direct signal from the audio interface and the recorded output signal from the device being measured makes up the input/output pairs which were used during network training and testing.

This process was adapted slightly for the HT-1 amplifier. The HT-1 amplifier has an emulated line out, which applies a speaker cabinet simulation to the signal. For the purposes of this study we are not interested in modelling this, so the speaker output of the amplifier was used instead. To allow for connection from the amplifier speaker output to the audio interface input, a Bugera PS1 power attenuator was used. The HT-1 amplifier speaker output was connected to the power attenuator and the line-out of the power attenuator was connected to the input of the audio interface. Additionally the speaker output of the power attenuator was also connected to a speaker cabinet.

The authors acknowledge that the choice of load will affect the output of the amplifier. Specifically whether the load is resistive or reactive has been shown to influence the output of triode tubes [21]. Whilst the amplifier is connected, indirectly, to a reactive load (the speaker cabinet), it is still thought that the output will be influenced by the presence of the power attenuator. Ideally the amplifier could be connected to a speaker cabinet, and the amplifier output could be recorded directly. This is difficult to achieve in practice. One option is to record the output of the speaker cabinet using a microphone, however this introduces a number of additional effects to the amplifier output. Namely the speaker cabinet nonlinearities and frequency response [22], as well as the coloration of the signal introduced by the microphone and its placement [23]. For these reasons, we chose to use the power attenuator for the collection of the training data.

For each device, the entire dataset was processed five times. Each time the user control being modelled was adjusted. For the HT-1 the control being modelled was the "ISF" control, and for the Big Muff it was the "Tone" control. For each measurement the control was set to one of five equally spaced settings, from the minimum (0) to the maximum (10) possible value.

For all recordings the HT-1 "gain" and "volume" knobs were set to 5 and 10 respectively. For the Big Muff the "volume" and "sustain" knobs were set to 10 and 5 respectively.

[1] www.idmt.fraunhofer.de/en/business_units/m2d/smt/guitar.html

[2] www.idmt.fraunhofer.de/en/business_units/m2d/smt/bass_lines.html

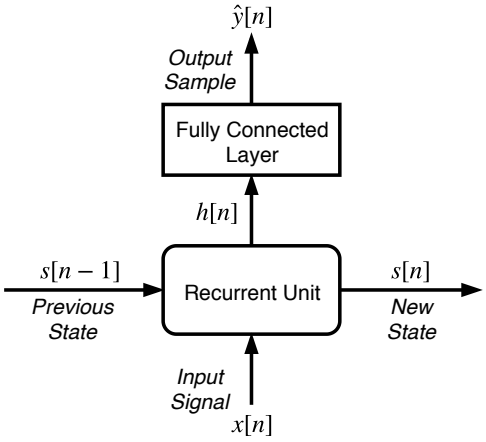


Figure 1: *Proposed neural network architecture, where $x[n]$ is the input signal, $s[n]$ and $s[n-1]$ are recurrent unit states, $h[n]$ is the recurrent unit output and $\hat{y}[n]$ is the neural network's predicted output sample.*

3. RECURRENT NEURAL NETWORK MODEL

The proposed model is a gated RNN based on either LSTM or Gated Recurrent Units (GRUs). The precise behaviour of LSTM units and GRUs is described in Sections 3.1 and 3.2, this section describes the general behaviour of the model and applies whether the recurrent unit selected is an LSTM or a GRU.

The proposed model consists of two hidden layers, a recurrent unit, followed by a fully connected layer. This architecture is shown in Figure 1. Unlike feedforward neural network layers, recurrent units have a *state*, which is used in the computation of the output and then updated at each time step. The model can be thought of as a function which is trained to predict the current output sample value, based on the current input signal value, the recurrent unit's state and the model's learned parameters:

$$\hat{y}[n] = f(x[n], s[n-1], \theta), \tag{1}$$

where n is the discrete time index, θ are the model's learned parameters and $s(n-1)$ is the recurrent unit's state at the previous time step. In this study, $\hat{y}[n]$ represents the model's prediction of a single output sample and $x[n]$ the unprocessed input signal.

At each time step an output is produced by the recurrent unit and fed into the fully connected layer. The size of the recurrent unit's output is determined by its *hidden size*, which is a model parameter defined by the user. Generally a greater hidden size produces a model capable of emulating more complex behaviour, but also requiring increased computational resources to train and run. The fully connected layer therefore consists of a single neuron, with a number of inputs equal to the recurrent unit hidden size. The output of this layer represents the networks predicted

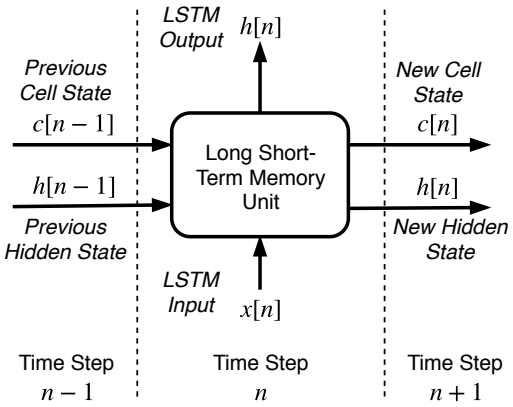


Figure 2: *Diagram of an LSTM unit, where c represents the cell state, h the hidden state and $x[n]$ is the input signal at time step n.*

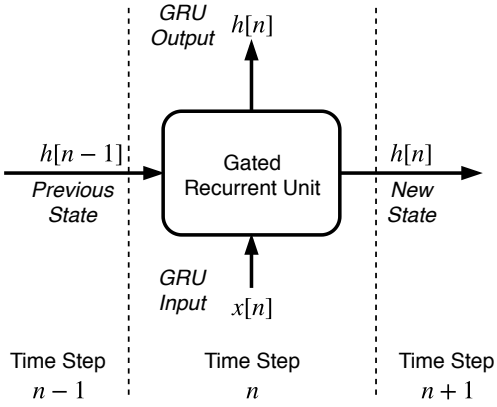


Figure 3: *Diagram of a GRU, where h represents the hidden state and $x[n]$ is the input signal at time step n.*

sample value for that time step. In addition to this, the recurrent unit's state is also updated at each time step, based on the previous state, current input and the model's learned parameters. As each state update is a function of the previous state, this means that the model has a potentially unlimited memory and in practice learns through training the effective memory required for the system being modelled.

The neural network model can also include conditioning, which allows for the emulation of the target device's user controls. To add conditioning, the input signal is extended to include an additional value representing the user control's setting, as suggested in [13]. At each time step the input signal will then be a vector containing the input sample and one or more conditioning values, depending on how many controls are included in the model.

It is possible to add additional recurrent layers before the final fully connected layer, however, preliminary testing indicated that this had little influence on the resulting model's accuracy. As such, the model was limited to just a single recurrent layer for this study.

3.1. Long Short-Term Memory

In an LSTM [24], the unit's state consists of two vectors, the *cell state*, c, and the *hidden state*, h. At each time step, the inputs are the current time step input, $x[n]$, the initial cell state, $c[n-1]$, and the initial hidden state, $h[n-1]$. The LSTM produces two outputs, the updated hidden state, $h[n]$, and the updated cell state, $c[n]$. An LSTM unit is depicted in Figure 2. The outputs are produced according to the following functions:

$$i[n] = \sigma(W_{ii}x[n] + b_{ii} + W_{hi}h[n-1] + b_{hi}), \quad (2)$$

$$f[n] = \sigma(W_{if}x[n] + b_{if} + W_{hf}h[n-1] + b_{hf}), \quad (3)$$

$$\tilde{c}[n] = \tanh(W_{ic}x[n] + b_{ic} + W_{hc}h[n-1] + b_{hc}), \quad (4)$$

$$o[n] = \sigma(W_{io}x[n] + b_{io} + W_{ho}h[n-1] + b_{ho}), \quad (5)$$

$$c[n] = f[n]c[n-1] + i[n]\tilde{c}[n], \quad (6)$$

$$h[n] = o[n]\tanh(c[n]), \quad (7)$$

where $i[n]$ is the input gate, $f[n]$ is the forget gate, $\tilde{c}[n]$ is the candidate cell state, $o[n]$ is the output gate, $\tanh(.)$ is the hyperbolic tangent function and $\sigma(.)$ is the logistic sigmoid function.

The LSTM unit state consists of the hidden state and the cell state. The state update of the LSTM unit is determined by the eight weight matrices and eight bias vectors, denoted by W and b respectively in the above equations. It should be noted that the biases in the neural network are simply constant offsets applied during the unit computation, and are in no way related to bias in electronic circuits. These weights and biases make up the learnable parameters of the LSTM unit, which are learned during training. The size of the weight matrices and bias vectors is determined by the input size and the LSTM's hidden size. The updated cell state, $c[n]$, and the updated hidden state, $h[n]$, are used as the initial state for the next time step, $n+1$. The updated hidden state is also used as the output of the LSTM for the current time step, n, which is input into the fully connected layer in our model.

3.2. Gated Recurrent Unit

A GRU [25] is an alternative recurrent unit, which can be used in the proposed architecture shown in Figure 1. In a GRU, the unit state consists of a single *hidden state* vector, h. At each time step, the inputs are the current time step input, $x[n]$ and the initial hidden state, $h[n-1]$. The GRU produces a single output, the updated hidden state, $h[n]$. A GRU unit is depicted in Figure 3. The hidden state is calculated according to the following functions:

$$r[n] = \sigma(W_{ir}x[n] + b_{ir} + W_{hr}h[n-1] + b_{hr}), \quad (8)$$

$$z[n] = \sigma(W_{iz}x[n] + b_{iz} + W_{hz}h[n-1] + b_{hz}), \quad (9)$$

$$\tilde{h}[n] = \tanh(W_{ih}x[n] + b_{ih} + r[n](W_{hh}h[n-1] + b_{hh})), \quad (10)$$

$$h[n] = (1 - z[n])\tilde{h}[n] + z[n]h[n-1], \quad (11)$$

where $r[n]$ is the reset gate, $z[n]$ is the update gate and $\tilde{h}[n]$ is the candidate hidden state.

The GRU unit state consists of the hidden state vector, which is determined by the six weight matrices and six bias vectors, denoted by W and b respectively in the above equations. These weights and biases are the learnable parameters of the GRU. The size of the weight matrices and bias vectors is determined by the input size and the GRU's hidden size. The updated hidden state, $h[n]$, is used as the initial state for the time step, $n+1$, as well as being used as the output of the GRU for the current time step n.

3.3. Fully Connected Layer

The fully connected layer, which can be seen in Figure 1, is not followed by an activation function, so the output sample predicted by the network is simply an affine transformation of the hidden state vector:

$$\hat{y}[n] = W_{fc}h[n] + b_{fc}, \tag{12}$$

where W_{fc} and b_{fc} are the fully connected layer's weight matrix and bias vector respectively and $h[n]$ is the recurrent unit hidden state at time n. As the model outputs a single value at each time step, the fully connected layer consists of just a single neuron. As such the weight matrix reduces to a vector and the bias vector consists of a single value in this case.

4. TRAINING

Recurrent Neural Networks are generally considered to be harder to train effectively in comparison to feedforward networks [26]. As such this section describes, in detail, the training process and techniques used in the completion of this study. All the RNN models produced for this paper were trained using the Adam optimizer [27], with an initial learning rate of 5×10^{-4}.

4.1. Loss Function

The loss function used during training is based on the error-to-signal ratio (ESR) with pre-emphasis filtering, as used in [13, 14]. The ESR is the squared error divided by the energy of the target signal. A pre-emphasis filter was applied to the output and target signals before computing the ESR. The pre-emphasis filter helps the network learn to model the high-frequency content. For a segment of training signal of length N, the pre-emphasised ESR is given by:

$$\mathcal{E}_{\text{ESR}} = \frac{\sum_{n=0}^{N-1} |y_p[n] - \hat{y}_p[n]|^2}{\sum_{n=0}^{N-1} |y_p[n]|^2}, \tag{13}$$

where y_p is the pre-emphasised target signal, and $\hat{y}_p$ is the pre-emphasised neural network output. The pre-emphasis filter was chosen to be a first-order high-pass filter with transfer function

$$H(z) = 1 - 0.85z^{-1}. \tag{14}$$

Additionally, a DC term was added to the loss to reduce the DC offset in the model outputs. The DC loss term is given by:

$$\mathcal{E}_{\text{DC}} = \frac{|\frac{1}{N}\sum_{n=0}^{N-1}(y[n] - \hat{y}[n])|^2}{\frac{1}{N}\sum_{n=0}^{N-1}|y[n]|^2}. \tag{15}$$

The DC term was introduced as early tests showed the model outputs contained a DC offset. The final loss function used for training is given by:

$$\mathcal{E} = \mathcal{E}_{\text{ESR}} + \mathcal{E}_{\text{DC}}. \tag{16}$$

4.2. Truncated Back-Propagation Through Time

When training RNNs it is possible to update the network parameters at each time step, however this comes with a very high computational cost. An alternative approach is to allow the RNN to process an entire sequence and then update the parameters. This involves backpropagating through the entire sequence to find the required gradients, as such this is often referred to as *Backpropagation Through Time* (BPTT) [28]. When modelling very long sequences, updating once per sequence results in slow training as the network parameters are updated infrequently. In addition to this, each update requires backpropagation through the entire sequence, which is computationally expensive.

Truncated BPTT [29] is a method in which parameter updates are carried out during the processing of a sequence. This means that updates can be carried out frequently and the recurrent unit state can persist between updates. The frequency of parameter updates and the number of time steps to run BPTT through for each update are training hyperparameters that are chosen by the user [26]. Our model is trained on audio sampled at a rate of 44.1 kHz, so the sequence length is very long even for a second of audio. As such truncated BPTT was used during the training of the RNNs in this study. The parameter updates were carried out every 2048 samples, with BPTT being run over 2048 time steps each update, as this was found to be an effective update frequency during our early experiments.

4.3. Batch-Processing

The training dataset was split into half-second segments. This was done for two reasons, firstly, by creating a large number of separate sequences, the short sequences can be processed in parallel, greatly reducing the time required to process the entire dataset. Secondly, having short segments allows the dataset to be shuffled at each epoch, which is known to improve network convergence rates [30]. The training data segments were shuffled at the beginning of each epoch and processed in mini-batches of 40 segments. At the start of each mini-batch the recurrent unit's initial state is set to 0. The first 1000 samples are then processed without updating the network parameters, to allow the recurrent unit state to initialise. The remaining samples are then processed, with backpropagation and parameter updates being carried out every 2048 samples. This is repeated until the entire training dataset is processed. The training dataset is then shuffled for the next training epoch.

5. REAL-TIME IMPLEMENTATION

A real-time implementation of the RNN was developed in C++. The implementation was built using the JUCE framework and the Eigen library for matrix and vector operations. JUCE can be used to build real-time audio plugins in the common VST, AU, and AAX formats supported by modern digital audio workstations. The developed plugin can be used to process audio through the trained models.

5.1. Recurrent Layer Computations

Algorithm 1 shows how the state update for the LSTM, as given by Equations (2)–(7), is carried out in practice. The eight matrix multiplications in Equations (2)–(5), which involve the input x and hidden state h, are performed as two bigger matrix multiplications. Furthermore, the eight bias terms in these equations can be combined to a single bias term.

The result of these matrix multiplications and the bias addition are stored in the vector v, which contains the non-activated values of the input gate $i[n]$, forget gate $f[n]$, output gate $o[n]$, and the candidate cell state $\tilde{c}[n]$. The hyperbolic tangent and logistic sigmoid activation functions are applied to v elementwise

Algorithm 1 LSTM State Update

Require: Layer input x, hidden state h, cell state c, hidden size N, input weight W_i, state weight W_h, bias b, [conditioning term b_{cond}]

$v \leftarrow W_i x + W_h h + b$
if b_{cond} was given **then** $v \leftarrow v + b_{\text{cond}}$
for each i in $[0, N[$ **do**
    $c[i] \leftarrow \sigma(v[N+i])c[i] + \sigma([v[i])\tanh(v[2N+i])$
    $h[i] \leftarrow \sigma(v[3N+i])\tanh(c[i])$
return h, c

when computing the new cell state c and the new hidden state h. The GRU computations are carried out in a similar fashion.

In the real-time implementation, the conditioning is not given to the layer in the input vector x in the same way as it was described in Section 3. Instead, since the conditioning is not typically updated at audio rate, processing power is saved when the effect of the conditioning in the layer activation is computed separately, and stored into a vector b_{cond}, which is then added to v at each time step.

5.2. Computational Load

The computational load of the implementation was tested with different RNN configurations. Models using LSTM units and GRUs were tested with different hidden state sizes. The results are shown in Figure 4. The processing speed is reported in terms of compute time required to process one second of audio, which was estimated by running the models on an Apple iMac with an 2.8 GHz Intel Core i5 processor. The GRU models run faster than LSTM models using the same hidden size. Using a single recurrent layer, an LSTM network runs faster than real time up to a hidden size of 160, whereas a GRU runs faster than real time up to a hidden size of 192.

6. COMPARISON OF MODELS

Models of the devices were created using two neural network architectures, the RNN model described in this paper, and the convolutional WaveNet-like neural network described in [13, 14].

For the RNN model, both the LSTM and GRU recurrent unit types were tested, with the hidden size ranging from 32 to 96. Each model was trained for 20 hours on a Nvidia Tesla V100 Graphics Processing Unit (GPU). The validation error was calculated every other epoch. Once the training was complete, the test loss was calculated using the model parameters from the epoch with the lowest validation loss. An example of the validation and training loss during training is shown in Figure 5. It can be seen that around epoch 250 there is a large increase in both the training and validation loss. It is not entirely clear why this sudden increase in loss occurs, however this was found to be fairly common during the training of the RNN models. As the spike includes both the training and validation loss, it does not indicate that the network is overfitting to the training dataset.

For the WaveNet model, the three configurations presented in [14] were used. The models vary in the number of convolutional layers and in the number of channels in the convolutional layers, i.e. the hidden size. All models use gated activations. Training the WaveNet models took approximately two to three hours on the GPU. The processing speed of the WaveNet models was estimated using the C++ implementation presented in [14].

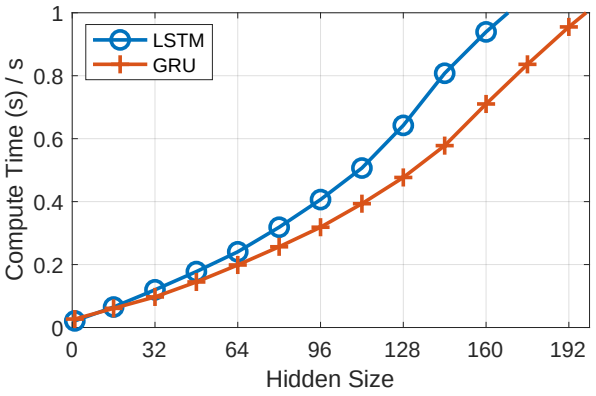


Figure 4: *Compute time of the real-time implementation for processing 1 s of audio at a 44.1-kHz sample rate, using different hidden sizes.*

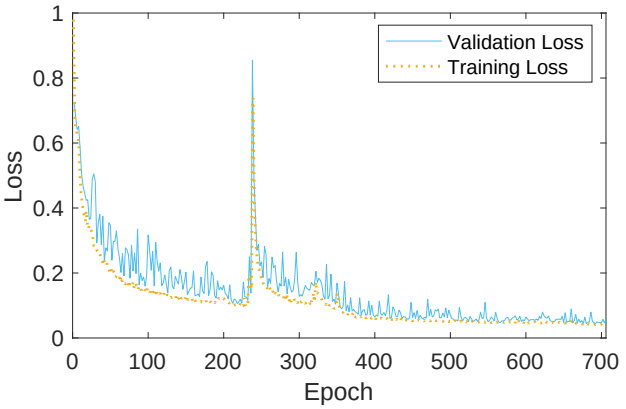


Figure 5: *Validation and Training loss for an RNN model of the Big Muff pedal.*

Model accuracy is evaluated in terms of the ESR loss, computed without pre-emphasis filtering, achieved on the unseen test dataset. The processing speed is reported in terms of compute time required to process one second of audio. A comparison between the RNN and WaveNet models of the HT-1 and the Big Muff is shown in Tables 1 and 2 respectively.

In terms of speed the results show a clear improvement for the RNN in comparison to the WaveNet model. The fastest RNN requires just 0.097 s to process a second of audio. The slowest RNN takes 0.41 s, which is less than the fastest WaveNet which takes 0.53 s to process a second of audio.

In terms of accuracy, the results vary depending on the device being modelled. In the case of the Big Muff pedal, the most accurate RNN model shows a considerable improvement over the most accurate WaveNet model. For the HT-1 amplifier the most accurate WaveNet outperforms the RNN. A comparison of audio processed by the best performing RNN model and the HT-1 amplifier is shown in the time domain in Figure 6 and in the frequency domain in Figure 7. The plots show good agreement between the

Table 1: *Error-to-signal ratio and processing speed for the Wavenet and the proposed GRU/LSTM models of the HT-1 Amplifier. The best results are highlighted.*

Model	Hidden Size	Layers	Number of Parameters	ESR	Time (s) / s of Output
WaveNet	16	10	24065	2.2%	0.53
WaveNet	8	18	11265	1.2%	0.63
WaveNet	16	18	43265	**0.79%**	0.91
GRU	32	1	3393	3.3%	**0.097**
LSTM	64	1	17217	1.8%	0.24
LSTM	96	1	38113	1.1%	0.41

Table 2: *Error-to-signal ratio and processing speed for the Wavenet and proposed LSTM models of the Big Muff pedal. The best results are highlighted.*

Model	Hidden Size	Layers	Number of Parameters	ESR	Time (s) / s of Output
WaveNet	16	10	24065	11%	0.53
WaveNet	8	18	11265	9.9%	0.63
WaveNet	16	18	43265	9.2%	0.91
LSTM	32	1	4513	10%	**0.12**
LSTM	48	1	9841	6.1%	0.18
LSTM	64	1	17217	**4.1%**	0.24

model output and the target device, however in future work listening tests should be conducted to verify the performance of the models.

6.1. Conditioning

The test loss was also computed separately for each of the five conditioning values the RNN models were trained on. For each model, the conditioning value with the greatest test loss was compared to the average test loss over all the conditioning values. The most extreme deviation from the average test loss was 16%, with the average deviation being 8%. This demonstrates that the models achieve a similar level of accuracy for each of the conditioning values.

On the devices modelled, the control knobs are continuous, allowing the user to select any value between the minimum and maximum setting. The models have been trained on data sampled at just five of the possible control knob settings, however the trained model is not limited to just these five settings. In order to test how well the RNNs can extrapolate to conditioning values not seen in training, a model was trained with the center conditioning value removed from the training dataset. The test lost was then computed for the unseen center conditioning value, and compared to the average test loss. For both the HT-1 and Big Muff the test loss for the unseen conditioning value was within 0.2% of the average. Informal listening tests also indicate that the models produces realistic outputs when the conditioned value is set to a value not seen during model training.

Audio examples of the models are available at the accompanying web page [31].

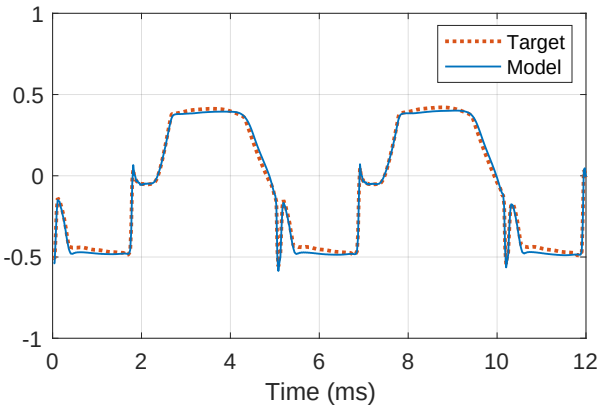


Figure 6: *Waveform of a guitar sound processed through the HT-1 guitar amplifier, and through the most accurate RNN model.*

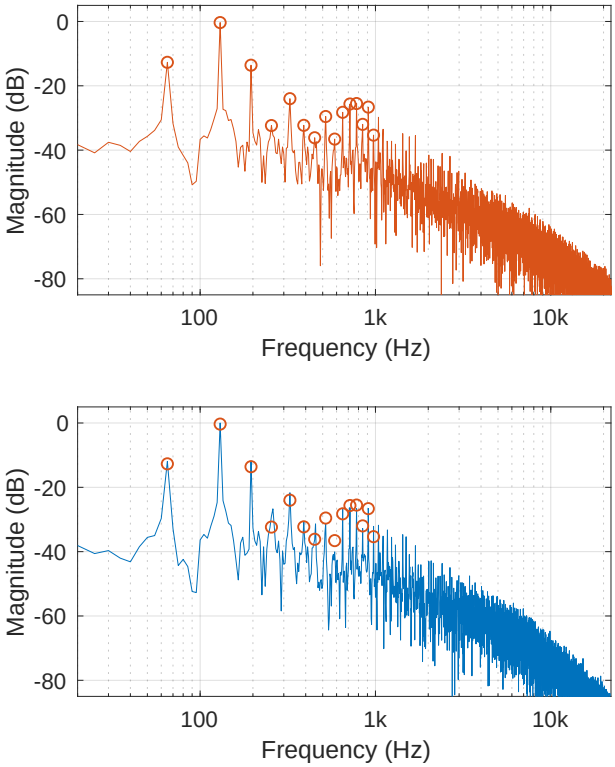


Figure 7: *Spectrum of a guitar sound processed (top) through the HT-1 guitar amplifier, and (bottom) through the most accurate RNN model. The circles indicate the level of the first 15 harmonics in the upper spectrum.*

7. CONCLUSIONS

This work has compared the performance of two types of neural network for the real-time emulation of a distortion pedal and a vacuum tube amplifier. The single layer RNN model presented in this paper requires much less processing power to run in comparison to a previously presented WaveNet-like convolutional neural network, and can be run in real time. The accuracy of the RNN model

was also comparable to or better than the WaveNet, depending on the device being modelled. Whilst listening tests have been carried out previously for the WaveNet-like model [13], future work should include listening tests to validate the proposed RNN model.

Of the two types of the recurrent unit tested, the GRU was shown to run slightly faster than the LSTM, for equivalent hidden sizes. However, we recommend the LSTM network over the GRU, as our tests indicate that for LSTM and GRU networks of roughly equivalent running speed, the LSTM generally achieved higher accuracy.

8. ACKNOWLEDGMENTS

We acknowledge the computational resources provided by the Aalto Science-IT project.

9. REFERENCES

[1] V. Välimäki, S. Bilbao, J. O. Smith, J. S. Abel, J. Pakarinen, and D. Berners, "Virtual analog effects," in *DAFX: Digital Audio Effects*, U. Zölzer, Ed., pp. 473–522. Wiley, Chichester, UK, second edition, 2011.

[2] M. Karjalainen and J. Pakarinen, "Wave digital simulation of a vacuum-tube amplifier," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP-06)*, Toulouse, France, May 2006, vol. 5, pp. 153–156.

[3] J. Pakarinen and D. T. Yeh, "A review of digital techniques for modeling vacuum-tube guitar amplifiers," *Computer Music J.*, vol. 33, no. 2, pp. 85–100, 2009.

[4] F. Santagata, A. Sarti, and S. Tubaro, "Non-linear digital implementation of a parametric analog tube ground cathode amplifier," in *Proc. Int. Conf. Digital Audio Effects (DAFx-07)*, Bordeaux, France, Sept. 2007, pp. 169–172.

[5] R. C. D. Paiva, S. D'Angelo, J. Pakarinen, and V. Välimäki, "Emulation of operational amplifiers and diodes in audio distortion circuits," *IEEE Trans. Circ. Syst. II*, vol. 59, no. 10, pp. 688–692, Oct. 2012.

[6] K. J. Werner, V. Nangia, A. Bernardini, J. O. Smith III, and A. Sarti, "An improved and generalized diode clipper model for wave digital filters," in *Proc. Audio Eng. Soc. 139th Conv.*, New York, NY, Oct. 2015.

[7] R. Kiiski, F. Esqueda, and V. Välimäki, "Time-variant gray-box modeling of a phaser pedal," in *Proc. Int. Conf. Digital Audio Effects (DAFx-16)*, Brno, Czech Republic, Sept. 2016, pp. 31–38.

[8] F. Eichas and U. Zölzer, "Gray-box modeling of guitar amplifiers," *J. Audio Eng. Soc.*, vol. 66, no. 12, pp. 1006–1015, Dec. 2018.

[9] A. Novak, L. Simon, P. Lotton, and J. Gilbert, "Chebyshev model and synchronized swept sine method in nonlinear audio effect modeling," in *Proc. Int. Conf. Digital Audio Effects (DAFx-10)*, Graz, Austria, Sept. 2010, pp. 423–426.

[10] S. Orcioni, A. Terenzi, S. Cecchi, F. Piazza, and A. Carini, "Identification of Volterra models of tube audio devices using multiple-variance method," *J. Audio Eng. Soc.*, vol. 66, no. 10, pp. 823–838, Oct. 2018.

[11] Z. Zhang, E. Olbrych, J. Bruchalski, T. J. McCormick, and D. L. Livingston, "A vacuum-tube guitar amplifier model using long/short-term memory networks," in *Proc. IEEE SoutheastCon*, Saint Petersburg, FL, Apr. 2018.

[12] T. Schmitz and J.-J. Embrechts, "Nonlinear real-time emulation of a tube amplifier with a long short time memory neural-network," in *Proc. Audio Eng. Soc. 144th Conv.*, Milan, Italy, May 2018.

[13] E.-P. Damskägg, L. Juvela, E. Thuillier, and V. Välimäki, "Deep learning for tube amplifier emulation," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP-19)*, Brighton, UK, May 2019, pp. 471–475.

[14] E.-P. Damskägg, L. Juvela, and V. Välimäki, "Real-time modeling of audio distortion circuits with deep learning," in *Proc. Int. Sound and Music Computing Conf. (SMC-19)*, Malaga, Spain, May 2019, pp. 332–339.

[15] The Big Muff Pi Page, "Evolution of the Big Muff Pi circuit," Available online at: http://www.bigmuffpage.com/Big_Muff_Pi_versions_schematics_part1.html, Accessed: 2019-04-05.

[16] K. J. Werner, V. Nangia, J. O. Smith, and J. S. Abel, "Resolving wave digital filters with multiple/multiport nonlinearities," in *Proc. Int. Conf. Digital Audio Effects (DAFX-15)*, Trondheim, Norway, Nov.–Dec. 2015, pp. 387–394.

[17] F. Eichas and U. Zölzer, "Black-box modeling of distortion circuits with block-oriented models," in *Proc. Int. Conf. Digital Audio Effects (DAFx-16)*, Brno, Czech Republic, Sept. 2016, pp. 39–45.

[18] Blackstar Amplification, "HT-1 guitar amplifier – Product page," Available online at: https://www.blackstaramps.com/uk/ranges/ht-1, Accessed: 2019-04-05.

[19] C. Kehling, J. Abeßer, C. Dittmar, and G. Schuller, "Automatic tablature transcription of electric guitar recordings by estimation of score- and instrument-related parameters," in *Proc. Int. Conf. Digital Audio Effects (DAFX-14)*, Erlangen, Germany, Sept. 2014, pp. 219–226.

[20] J. Abeßer, P. Kramer, C. Dittmar, and G. Schuller, "Parametric audio coding of bass guitar recordings using a tuned physical modeling algorithm," in *Proc. Int. Conf. Digital Audio Effects (DAFx-13)*, Maynooth, Ireland, Sept. 2013, pp. 154–159.

[21] J. Pakarinen and M. Karjalainen, "Enhanced wave digital triode model for real-time tube amplifier emulation," *IEEE Trans. Audio Speech Lang. Process.*, vol. 18, no. 4, pp. 738–746, May 2010.

[22] D. T. Yeh, B. Bank, and M. Karjalainen, "Nonlinear modeling of a guitar loudspeaker cabinet," in *Proc. Int. Conf. Digital Audio Effects (DAFx-08)*, Espoo, Finland, Sept. 2008, pp. 89–96.

[23] A. Roginska, A. U. Case, A. Madden, and J. Anderson, "Measuring spectral directivity of an electric guitar amplifier," in *Proc. Audio Eng. Soc. 132nd Conv.*, Budapest, Hungary, Apr. 2012.

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Dec. 1997.

[25] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint*, Dec. 2014, arXiv:1412.3555 [cs.NE].

[26] I. Sutskever, *Training Recurrent Neural Networks*, Ph.D. thesis, University of Toronto, Ontario, Canada, 2013.

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learning Representations (ICLR-15)*, San Diego, CA, May 2015.

[28] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct. 1990.

[29] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, Mar. 1990.

[30] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural Networks: Tricks of the Trade*, pp. 437–478. Springer, 2012.

[31] A. Wright, E.-P. Damskägg, and V. Välimäki, "Real-time black-box modeling with recurrent neural networks," accompanying webpage, available online at: http://research.spa.aalto.fi/publications/papers/dafx19-rnn/, Accessed: 2019-18-06.

NMF TOOLBOX: MUSIC PROCESSING APPLICATIONS OF NONNEGATIVE MATRIX FACTORIZATION

Patricio López-Serrano[1], Christian Dittmar[2], Yiğitcan Özer[2], Meinard Müller[1]

[1]International Audio Laboratories Erlangen* , Erlangen, Germany
[2]Fraunhofer IIS, Erlangen, Germany
{patricio.lopez.serrano, meinard.mueller}@audiolabs-erlangen.de
{christian.dittmar, yigitcan.oezer}@iis.fraunhofer.de

ABSTRACT

Nonnegative matrix factorization (NMF) is a family of methods widely used for information retrieval across domains including text, images, and audio. Within music processing, NMF has been used for tasks such as transcription, source separation, and structure analysis. Prior work has shown that initialization and constrained update rules can drastically improve the chances of NMF converging to a musically meaningful solution. Along these lines we present the NMF toolbox, containing MATLAB and Python implementations of conceptually distinct NMF variants—in particular, this paper gives an overview for two algorithms. The first variant, called nonnegative matrix factor deconvolution (NMFD), extends the original NMF algorithm to the convolutive case, enforcing the temporal order of spectral templates. The second variant, called diagonal NMF, supports the development of sparse diagonal structures in the activation matrix. Our toolbox contains several demo applications and code examples to illustrate its potential and functionality. By providing MATLAB and Python code on a documentation website under a GNU-GPL license, as well as including illustrative examples, our aim is to foster research and education in the field of music processing.

1. INTRODUCTION

The general goal of NMF is to factorize a matrix V with nonnegative entries into two other nonnegative matrices W and H which typically are required to have much lower rank than V. Due to the fact that NMF can learn a semantically meaningful parts-based data representation [1], it has been used extensively in music signal processing and information retrieval, for tasks such as music transcription [2], automatic drum transcription (ADT) [3], drum source separation (DSS) [4], harmonic-percussive source separation (HPSS) [5, 6, 7], in combination with kernel additive modeling (KAM) [8, 9], redrumming [10, 11], sampling detection [12, 13], structure analysis [14, 15, 16], score-informed source separation [17], and key estimation [18], to name a few.

In the case of music tasks, we would like to learn two non-negative matrices W and H that capture *what* happened (i.e., which particular sounds were produced, which drum kit parts were struck, which piano notes were sounded), and *when* each one of these sound events was active in time. Intuitively, we can say that the template (or spectral basis) matrix W contains the "what"—whereas H, called the gain (or activation) matrix, contains the "when". We show this intuition in Figure 1, which illustrates an NMFD model of a short drum signal. The templates are shown on the left as color-coded component spectrograms representing kick drum (KD), snare drum (SD), and hi-hat (HH). The activations are shown as colored curves at the top of the figure. Throughout this paper we present a number of didactic examples, aiming to contribute to music processing education. In contrast to other NMF-related toolboxes [19], we provide an illustrated compendium of musically-motivated applications found throughout the literature. In the spirit of the DAFx book [20], the code examples can be used as a reference implementation in further research, whereas the figures (especially through the color-coding) provide concise illustrations of the principles behind NMF(D), and can be used as learning material.

The remainder of this paper is structured as follows. In Section 2 we introduce the basic theoretical framework and notation for NMF and the variants that we use in this paper. In Section 3 we give an overview of our toolbox code[1], discussing the main functions, parameters, and dependencies. In Sections 4, 5, and 6 we present three application scenarios, illustrated by going through their source code and examining the graphic output from visualization functions. Although our code examples are given as MATLAB listings, we ensured that the naming conventions and usage of our Python implementation are basically the same.

2. NMF AND VARIANTS

In this section we give a brief formal overview of the NMF variants that are provided as code in the toolbox.

2.1. NMF

Here we introduce NMF, closely following [21, Section 8.3] and [1]. NMF is based on iteratively computing a low-rank approximation $\mathrm{U} \in \mathbb{R}_{\geq 0}^{K \times M}$ of the nonnegative matrix (typically a magnitude spectrogram) $\mathrm{V} \in \mathbb{R}_{\geq 0}^{K \times M}$, where $K \in \mathbb{N}$ is the feature dimensionality and $M \in \mathbb{N}$ represents the number of elements or frames along the time axis. Specifically, U is defined as the linear combination of the templates $\mathrm{W} \in \mathbb{R}_{\geq 0}^{K \times R}$ and activations $\mathrm{H} \in \mathbb{R}_{\geq 0}^{R \times M}$ such that $\mathrm{V} \approx \mathrm{U} := \mathrm{W} \cdot \mathrm{H}$. The rank $R \in \mathbb{N}$ of the approximation (i. e., number of components) is an important parameter that needs to be specified beforehand.

* The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and the Fraunhofer-Institut für Integrierte Schaltungen IIS.

[1] https://www.audiolabs-erlangen.de/resources/MIR/NMFtoolbox/

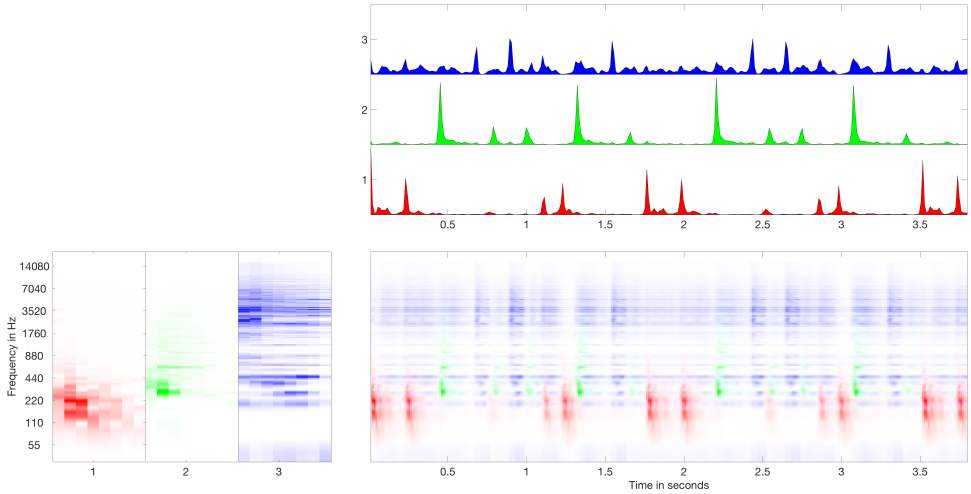


Figure 1: Output visualization generated by code in Listing L.1. NMFD model for drum source separation (KD in red, SD in green, HH in blue). Top: activation matrix H. Bottom left: slices of template tensor P. Bottom right: color-coded approximated target spectrogram U.

NMF typically starts with a suitable initialization of the matrices W and H. For example, both matrices could be populated with non-negative random numbers—however, depending on the task and availability of prior information, it might make more sense to use other initialization strategies. After initialization, both W and H are iteratively updated to approximate V with respect to a cost function $\mathcal{L}$. A standard choice is the generalized Kullback-Leibler Divergence (KLD) [1], given as

$$\mathcal{L} = \mathcal{D}_{\mathrm{KL}}(\mathrm{V} \mid \mathrm{U}) = \sum \left(\mathrm{V} \odot \log\left(\frac{\mathrm{V}}{\mathrm{U}}\right) - \mathrm{V} + \mathrm{U} \right). \quad (1)$$

The symbol $\odot$ denotes element-wise multiplication; the logarithm and division are to be performed element-wise as well. The sum is to be computed over all $K \cdot M$ elements of V. To minimize this cost, an alternating scheme with multiplicative updates is used [1]. The respective update rules are given as

$$\mathrm{W} \leftarrow \mathrm{W} \odot \frac{\frac{\mathrm{V}}{\mathrm{U}} \cdot \mathrm{H}^\top}{\mathrm{J} \cdot \mathrm{H}^\top}, \quad (2)$$

$$\mathrm{H} \leftarrow \mathrm{H} \odot \frac{\mathrm{W}^\top \cdot \frac{\mathrm{V}}{\mathrm{U}}}{\mathrm{W}^\top \cdot \mathrm{J}}, \quad (3)$$

with $\mathrm{U} := \mathrm{W} \cdot \mathrm{H}$, where the symbol $\cdot$ denotes the matrix product. Furthermore, $\mathrm{J} \in \mathbb{R}^{K \times M}$ denotes a matrix of ones. Since this is an alternating update scheme, it should be noted that Eq. (2) uses the latest update of H from the previous iteration. In the same vein, (3) uses the latest update of W. These update rules are typically applied for a limited number of iterations $L \in \mathbb{N}$.

2.2. NMFD

In this section we introduce NMFD, originally proposed in [22]. We follow the notation and formulation found in [23].
NMFD extends NMF by using two-dimensional templates (or *patterns*) so that each of the R templates can be interpreted as a magnitude spectrogram snippet consisting of $T \ll M$ spectral frames. We assume that the magnitude spectrogram V can be modeled using a mixture of R patterns $\mathrm{P}^r \in \mathbb{R}^{K \times T^r}$, $r \in [0 : R-1] := \{0, \ldots, R-1\}$. The parameter $T^r \in \mathbb{N}$ is the number of feature frames or observations for pattern P^r. Although the patterns can have different lengths, without loss of generality, we define their lengths to be the same $T := T^0 = \ldots = T^{R-1}$, which could be achieved by adequately zero-padding shorter patterns until they reach the length of the longest. Based on this assumption, the patterns can be grouped into a pattern tensor $\mathrm{P} \in \mathbb{R}^{K \times R \times T}$. The subdimension (or *slice*) of the tensor which refers to a specific pattern with index r is $\mathrm{P}^r := \mathrm{P}(\cdot,\ r,\ \cdot)$, whereas $\mathrm{P}_t := \mathrm{P}(\cdot,\ \cdot,\ t)$ refers to frame index t simultaneously in all patterns. Thus, the magnitude spectrogram can be modeled as

$$\mathrm{U} := \sum_{t=0}^{T-1} \mathrm{P}_t \cdot \overset{t\rightarrow}{\mathrm{H}}, \quad (4)$$

where $\overset{t\rightarrow}{(\cdot)}$ denotes a frame shift operator [22]. Smaragdis [22] defined the update rules that extend Eqs. 2 and 3 to the convolutive case as follows:

$$\mathrm{P}_t \leftarrow \mathrm{P}_t \odot \frac{\frac{\mathrm{V}}{\mathrm{U}} \cdot \left(\overset{t\rightarrow}{\mathrm{H}}\right)^\top}{\mathrm{J} \cdot \left(\overset{t\rightarrow}{\mathrm{H}}\right)^\top}, \quad (5)$$

$$\mathrm{H} \leftarrow \mathrm{H} \odot \frac{\mathrm{P}_t^\top \cdot \overset{\leftarrow t}{\left[\frac{\mathrm{V}}{\mathrm{U}}\right]}}{\mathrm{P}_t^\top \cdot \mathrm{J}}. \quad (6)$$

As a side note, NMFD can be made to function like a regular NMF (Section 2.1) by using a pattern tensor with dimensions $K \times R \times 1$ (i. e., R patterns with a single frame).

2.3. Diagonal NMF

Diagonal NMF is a variant originally proposed by Driedger et al. in [24] for the task of audio mosaicing. In audio mosaicing, a (timbral) *source* is used to recreate or synthesize the sounds in a *target*. In the original publication, Driedger et al. present an example consisting of buzzing bees as a source and "Let It Be" by the Beatles

as a target. Their objective is to synthesize a signal which sounds as if the bees buzzed at different pitches to the tune of the Beatles' song. In order to maximize the recognizability of the source (buzzing bees), the authors propose using a fixed template matrix as well as an extended set of update rules that support the development of sparse diagonal structures in the activation matrix. The crucial observation is that diagonals activate sequences of frames, helping preserve temporality and recognizability. We present an example of diagonal NMF for mosaicing in Section 6.

3. TOOLBOX

In this section we discuss the overall structure and functionality of our toolbox, summarized in Table 1. The first four rows contain the core functions, implementing variants of NMF. We begin with `NMFD`, a core function used for the examples in Sections 4 and 5 (we briefly discussed the theory in Section 2.2).
The variant used as an application example in Section 6 is `NMFdiag`, described in Section 2.3. It requires more input parameters as fields of the structure `parameter.continuity` and specific information can be found in the respective source code headers, as well as in the original publication [24].
Next in the table we have `convModel` and `shiftOperator`. `convModel` is called from within `NMFD` and `NMFconv`—it performs the core convolution operation between P and H at every iteration. On the other hand, `shiftOperator` is a helper function to perform the frame shifting used in Eqs. 4, 5, and 6, adding boundary checks and zero-padding as necessary. The functions `initActivations` and `initTemplates` are helper functions that help us initialize the basis matrices/tensors and the activations matrices. The user may call these functions explicitly to create and assign parameters, but these functions are also called internally with default settings from within the core functions if the user doesn't pass variables in (see Sections 4 and 5 for more information on the types of initialization that are used for specific circumstances.)
The function `NEMA` is used to introduce exponential decay when initializing certain types of templates and activations with both `init` functions. We also provide utility functions to transform between frequency in Hz and MIDI pitches, which are mainly used to generate log-frequency spectrograms for visualization and to generate harmonic templates. Finally, we use `forwardSTFT` to compute the spectrograms we use with our NMF examples. Some tasks also require resynthesizing time-domain signals (such as DSS in Section 4 and mosaicing in Section 6)—for these cases we include `LSEE_MSTFTM_GriffinLim` [28], `inverseSTFT`, and `alphaWienerFilter` (see [29, 30]).

4. APPLICATION: DRUM SOURCE SEPARATION

We begin with an example from the task of DSS, taken from [26]. Given a recording of mixed drum kit components into one signal, the goal of DSS is to produce individual component signals for each instrument or piece in the drum kit as if it had been recorded in isolation. This technique can be used within recording studio or remixing settings, where it is often desirable to treat individual drum kit components separately.

```
1  inpPath = 'data/';
2  outPath = 'output/';
3  filename = 'Winstons_AmenBreak.wav';
4
5   % 1. load the audio signal
6  [x,fs] = audioread([inpPath filename]);
7  x = mean(x,2);
8
9   % 2. compute STFT
10 % spectral parameters
11 paramSTFT.blockSize = 2048;
12 paramSTFT.hopSize = 512;
13 paramSTFT.winFunc = hann(paramSTFT.blockSize);
14 paramSTFT.reconstMirror = true;
15 paramSTFT.appendFrame = true;
16 paramSTFT.numSamples = length(x);
17
18  % STFT computation
19 [X,A,P] = forwardSTFT(x,paramSTFT);
20
21  % get dimensions and time and freq resolutions
22 [numBins,numFrames] = size(X);
23 deltaT = paramSTFT.hopSize / fs;
24 deltaF = fs / paramSTFT.blockSize;
25
26  % 3. apply NMF variants to STFT magnitude
27 % set common parameters
28 numComp = 3;
29 numIter = 30;
30 numTemplateFrames = 8;
31
32  % generate initial guess for templates
33 paramTemplates.deltaF = deltaF;
34 paramTemplates.numComp = numComp;
35 paramTemplates.numBins = numBins;
36 paramTemplates.numTemplateFrames = numTemplateFrames;
37 initW = initTemplates(paramTemplates,'drums');
38
39  % generate initial activations
40 paramActivations.numComp = numComp;
41 paramActivations.numFrames = numFrames;
42 initH = initActivations(paramActivations,'uniform');
43
44  % NMFD parameters
45 paramNMFD.numComp = numComp;
46 paramNMFD.numFrames = numFrames;
47 paramNMFD.numIter = numIter;
48 paramNMFD.numTemplateFrames = numTemplateFrames;
49 paramNMFD.initW = initW;
50 paramNMFD.initH = initH;
51
52  % NMFD core method
53 [nmfdW, nmfdH, nmfdV, divKL] = NMFD(A, paramNMFD);
54
55  % alpha-Wiener filtering
56 nmfdA = alphaWienerFilter(A,nmfdV,1);
57
58  % visualize
59 paramVis.deltaT = deltaT;
60 paramVis.deltaF = deltaF;
61 paramVis.endeSec = 3.8;
62 paramVis.fontSize = 24;
63 visualizeComponentsNMF(A, nmfdW, nmfdH, nmfdA, paramVis);
64
65 % resynthesize
66 for k = 1:numComp
67   Y = nmfdA{k} .* exp(j * P);
68
69   % re-synthesize, omitting the Griffin Lim iterations
70   y = inverseSTFT(Y, paramSTFT);
71
72   % save result
73   audiowrite([outPath,'Winstons_AmenBreak_NMFD_component_',
          ...
74     num2str(k)],y,fs);
75 end
```

Listing L.1: MATLAB code for drum source separation using NMFD, following [26].

In lines 11–16 we define the parameters for STFT computation: a block size $N = 2048$ samples, a hop size $H = 512$ samples, a Hann window, a flag to discard the mirror spectrum (`reconstMirror = true`), and a flag to indicate that we want to zero-pad the entire signal with half-block lengths at the beginning and end. Now we will discuss the most important part of this example, which is the

Filename	Description and main parameters
`NMFD.m`	Nonnegative Matrix Factor Deconvolution with KLD and fixable components [22]. `V, numComp, numIter, numTemplateFrames, initW, initH, paramConstr, fixH`
`NMF.m`	Nonnegative matrix factorization with KLD as default cost function [21, Section 8.3], [1]. `V, costFunc, numIter, numComp.`
`NMFdiag.m`	Nonnegative matrix factorization with enhanced diagonal continuity constraints [24]. `V, W0, H0, distmeas, numOfIter, fixW, continuity.length, continuity.grid, continuity.sparsen, continuity.polyphony`
`NMFconv.m`	Convolutive NMF with beta-divergence [25, Chapter 3.7]. `V, numComp, numIter, numTemplateFrames, initW, initH, beta, sparsityWeight, uncorrWeight`
`convModel.m`	Convolutive NMF model implementing Eq. (4) from [26]. Note that it can also be used to compute the standard NMF model in case the number of time frames of the templates equals one. `W, H`
`shiftOperator.m`	Shift operator as described in Eq. (5) from [26]. It shifts the columns of a matrix to the left or the right and fills undefined elements with zeros. `A, shiftAmount`
`initActivations.m`	Initialization strategies for NMF activations, including `random` and `uniform`. The `pitched` strategy places gate-like activations at the frames where certain notes are active in the ground truth [27]. The strategy `drums` uses decaying impulses at these positions [26]. `numComp, numFrames, deltaT, pitches, onsets, durations, drums, decay, onsetOffsetTol, tolerance, strategy`
`initTemplates.m`	NMF template initialization strategies, including `random` and `uniform`. The strategy `pitched` uses comb-filter templates [27]. The `drums` strategy uses pre-extracted averaged spectra of typical drum types. `numComp, numBins, numTemplateFrames, pitches, drumTypes, strategy`
`NEMA.m`	Row-wise nonlinear exponential moving average, introducing decaying slopes according to Eq. (3) from [9]. `lambda`
`midi2freq.m, freq2midi.m, logFreqLogMag.m`	Helper functions to convert between MIDI pitches and frequencies in Hz, as well as log-frequency and log-magnitude representations for visualization. `midi, freq, A, deltaF, binsPerOctave, upperFreq, lowerFreq`
`LSEE_MSTFTM_GriffinLim, forwardSTFT.m, inverseSTFT.m`	Reconstruct the time-domain signal by means of the frame-wise inverse FFT and overlap-add method described as least squares error estimation from the modified STFT magnitude (LSEE-MSTFT) in [28]. `blockSize, hopSize, anaWinFunc, synWinFunc, reconstMirror, appendFrame, analyticSig, numSamples`
`alphaWienerFilter.m`	Alpha-related soft masks for extracting sources from mixture. Details in [29] and experiments in [30]. `alpha, binarize`

Table 1: Overview of MATLAB functions, descriptions, and main parameters. The Python version of the toolbox follows the same naming convention as far as possible.

particular setup of the NMF-related variables and parameters. In line 28 we initialize the number of components R to 3, since we know a priori that the drum recording contains the instruments kick drum (KD), snare drum (SD), and hi-hat (HH). We will run NMFD for a total of 30 iterations (of the update rules), $L = 30$, line 29, and each spectral slice P^r will have a length of $T = 8$ time frames, line 30. A unique thing that sets apart this example from the rest is the fact that we initialize the templates with a particular strategy that has proven to be very effective in DSS [26]. In line 37 we use the string parameter `drums` to specify that we want to initialize the templates by seeding them with a single-frame, data-driven statistical mean of many drum sounds of that type, and then applying a short exponential decay to that single frame in order to expand the single frame into multiple frames across the template pattern matrix. This initializes the template tensor to have three patterns, each containing prototypical spectral properties of KD, SD, and HH, respectively. For this particular application, the initial activation matrix `initH` can be initialized using a constant value of `1` throughout the entire matrix, indicated by the parameter `uniform` in line 42. In lines 45–50 all the previously created parameters are assigned to the parameter structure `paramNMFD` which will be passed to `NMFD()`, the main function which we call in line 53. In line 56 we use `alphaWienerFilter()` to compute spectrogram estimates for the components via Wiener filtering (see [29, 30]). We set visualization parameters to `paramVis` in lines 59–62 and call `visualizeComponentsNMF()` in line 63, which produces the output seen in Figure 1. It is important to mention that the visualization function assigns default color schemes depending on the NMF model's rank. For instance, as seen in Figure 1, a model with $R = 3$ components will be colored with with red, green, and blue, respectively. A model with $R = 4$ components (Figure 2) is colored using `hsv(4)`, and models with $R > 4$ (Figure 3) are colored with a grayscale colormap. Finally, we loop over the component spectrograms, resynthesizing each one using `inverseSTFT()` (line 70) and saving the result `y` as a WAV file.

5. APPLICATION: ELECTRONIC MUSIC STRUCTURE

Taken from [23], we show an example of the following task: Given a downmix of an electronic music (EM) track produced with certain loops, together with individual instances of the loops that were used to produce the track, can we use NMFD to learn an activation matrix which tells us when each loop was used in the track? In this example, we want to highlight the fact that after initializing the template tensor pages, each pattern to one instance of one of the loops used, we subsequently disallow updating/learning the templates—also called fixing the templates. Thus, we are only interested in allowing updates to the activation matrix, which will end up telling us when each loop type was used.

```
% initialization
inpPath = 'data/';
outPath = 'output/';
filename = 'LSDDM_EM_track.wav';
filenameEffects = 'LSDDM_EM_Effects.wav';
filenameBass = 'LSDDM_EM_bass.wav';
filenameMelody = 'LSDDM_EM_melody.wav';
filenameDrums = 'LSDDM_EM_drums.wav';

% 1. load the audio signal
[xTr,fs] = audioread([inpPath filename]);

[xEffects, fsEffects] = audioread([inpPath filenameEffects]);

[xBass, fsBass] = audioread([inpPath filenameBass]);
[xMelody, fsMelody] = audioread([inpPath filenameMelody]);
[xDrums, fsDrums] = audioread([inpPath filenameDrums]);
```

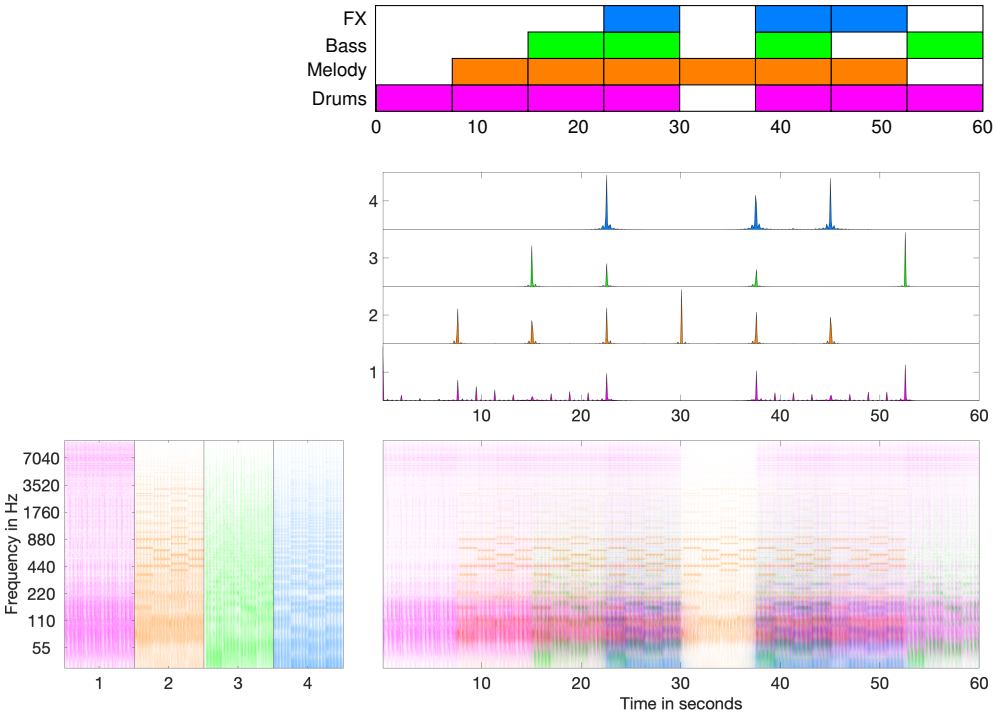


Figure 2: Top: Ground truth loop activations for the track used in Listing L.2. Middle: Activation matrix H with one row per loop/component. Bottom left: Template tensor P with component spectrogram slices for drums (1), melody (2), bass (3), and FX (4). Bottom right: component-colored spectrogram U.

```
17  % make monaural if necessary
18  xTr = mean(xTr, 2);
19  xEffects = mean(xEffects, 2);
20  xBass = mean(xBass, 2);
21  xMelody = mean(xMelody, 2);
22  xDrums = mean(xDrums, 2);
23
24  % 2. compute STFT
25  paramSTFT.blockSize = 4096;
26  paramSTFT.hopSize = 2048;
27  paramSTFT.winFunc = hann(paramSTFT.blockSize);
28  paramSTFT.reconstMirror = true;
29  paramSTFT.appendFrame = true;
30
31  % STFT computation
32  [XTr, ATr, PTr] = forwardSTFT(xTr, paramSTFT);
33
34  % get dimensions and time and freq resolutions
35  [numBinsTr, numFramesTr] = size(XTr);
36  deltaT = paramSTFT.hopSize / fs;
37  deltaF = fs / paramSTFT.blockSize;
38
39  % STFT computation
40  [XEffects, AEffects, PEffects] = forwardSTFT(xEffects,
        paramSTFT);
41  [XBass, ABass, PBass] = forwardSTFT(xBass, paramSTFT);
42  [XMel, AMel, PMel] = forwardSTFT(xMelody, paramSTFT);
43  [XDrums, ADrums, PDrums] = forwardSTFT(xDrums, paramSTFT);
44  [numBinsBass, numFramesBass] = size(XBass);
45
46  % 3. apply NMF variants to STFT magnitude
47  numComp = 4;
48  numIter = 30;
49
50  initW = [];
51  initW{1} = ABass;
52  initW{2} = AMel;
53  initW{3} = ADrums;
54  initW{4} = AEffects;
55  paramNMFD.initW = initW;
56  numTemplateFrames = numFramesBass;
57
58  % generate initial activations
59  paramActivations.numComp = numComp;
60  paramActivations.numFrames = numFramesTr;
61  initH = initActivations(paramActivations, 'uniform');
62  paramNMFD.initH = initH;
63
64  % NMFD parameters
65  paramNMFD.numComp = numComp;
66  paramNMFD.numFrames = numFramesTr;
67  paramNMFD.numIter = numIter;
68  paramNMFD.numTemplateFrames = numTemplateFrames;
69  paramNMFD.numBins = numBinsTr;
70  paramNMFD.fixW = 1;
71
72  % NMFD core method
73  [nmfdW, nmfdH, nmfdV, divKL] = NMFD(ATr, paramNMFD);
74
75  %% visualize
76  paramVis.deltaT = deltaT;
77  paramVis.deltaF = deltaF;
78  paramVis.logComp = 1e5;
79  fh1 = visualizeComponentsNMF(ATr, nmfdW, nmfdH, nmfdV,
        paramVis);
80
81  %% save result
82  saveas(fh1,[outPath,'LSDDM_EM.png']);
```

Listing L.2: Example code for learning loop activation points using NMFD, following [23].

In lines 2–22 we prepare path names and load files for five audio signals. These include the mixed track, as well as the four loops that were used to produce the track: effects, bass, melody, and drums. At the end of this block we convert all signals to mono.
In the next block (lines 25–29), we set the STFT parameters. In lines 32–43 we use the parameter structure `paramSTFT` together

with the respective signals to compute all the necessary spectrograms by calling `forwardSTFT()`.
In the next lines we will be preparing the parameters that are required to run NMFD with the specific conditions for this example. First, in line 47 we set `numComp`, the number of components or patterns R, to 4 (since we know a priori that the track was composed with four loops or patterns). In line 48 we specify 30 as the desired number of iterations for the algorithm. In lines 50–55 we prepare a cell array `initW` containing the magnitude spectrograms for the loops, and set the initial template tensor to the parameter structure `paramNMFD.initW`. As the number of template frames T we use the number of frames of the bass loop, but we could have used any loop's length—we have constructed the example such that all loops have the same number of frames. In lines 59–62 we initialize the activation matrix using the `uniform` strategy (i. e., with a constant value of one for all rows) and append it to the parameter structure. In lines 65–69 we set some basic values to the parameter structure `paramNMFD`. Most importantly, in line 70, we indicate with the flag `fixW = 1` that we want to fix the template tensor. Since we initialized the pattern tensor with the loops we know to be contained in the track, we disallow modifications and only wish to learn the activation matrix. In line 73 we call `NMFD()` with the track magnitude spectrogram and the previously set parameter structure, obtaining four return variables: `nmfdW`, the learned templates (which are not modified during the learning); `nmfdH`, the learned activation matrix, `nmfdV`, a cell array containing the learned component spectrograms; and `divKL`, an array with the KLD at each iteration (to visualize learning error throughout the iterations). In lines 76–78 we set basic visualization parameters to the structure `paramVis`, then visualize the learned NMFD model with `visualizeComponentsNMF()` (line 79), and save the result to disk (line 82).

6. APPLICATION: AUDIO MOSAICING

In this example we will be loading two sounds: a recording of bees buzzing, which will act as the *timbral source* for the mosaicing and a fragment of "Let It Be" by the Beatles, which is the *target* to be synthesized with the sounds of bees buzzing. The main idea is to produce a signal sounding as if buzzing bees were "playing" "Let It Be" by the Beatles by buzzing at different pitches or frequencies.

```
1  % initialization
2  inpPath = 'data/';
3  outPath = 'output/';
4
5  filenameSource = 'Bees_Buzzing.wav';
6  filenameTarget = 'Beatles_LetItBe.wav';
7
8  % 1. load the source and target signal
9  % read signals
10 [xs,fs] = audioread([inpPath filenameSource]);
11 [xt,fs] = audioread([inpPath filenameTarget]);
12 % make monaural if necessary
13 xs = mean(xs,2);
14 xt = mean(xt,2);
15
16 % 2. compute STFT of both signals
17 % spectral parameters
18 paramSTFT.blockSize = 2048;
19 paramSTFT.hopSize = 1024;
20 paramSTFT.winFunc = hann(paramSTFT.blockSize);
21 paramSTFT.reconstMirror = true;
22 paramSTFT.appendFrame = true;
23 paramSTFT.numSamples = length(xt);
24
25 % STFT computation
26 [Xs,As,Ps] = forwardSTFT(xs,paramSTFT);
27 [Xt,At,Pt] = forwardSTFT(xt,paramSTFT);
28
29 % get dimensions and time and freq resolutions
30 [numBins,numTargetFrames] = size(Xt);
31 [numBins,numSourceFrames] = size(Xs);
32 deltaT = paramSTFT.hopSize / fs;
33 deltaF = fs / paramSTFT.blockSize;
34
35 % 3. apply continuity NMF variants to mosaicing pair
36 % initialize activations randomly
37 H0 = rand(numSourceFrames,numTargetFrames);
38
39 % init templates by source frames
40 W0 = bsxfun(@times,As,1./(eps+sum(As)));
41 Xs = bsxfun(@times,Xs,1./(eps+sum(As)));
42
43 % parameters taken from Jonathan Driedger's toolbox
44 paramNMFdiag.fixW = 1;
45 paramNMFdiag.numOfIter = 20;
46 paramNMFdiag.continuity.polyphony = 10;
47 paramNMFdiag.continuity.length = 7;
48 paramNMFdiag.continuity.grid = 5;
49 paramNMFdiag.continuity.sparsen = [1 7];
50
51 % reference implementation by Jonathan Driedger
52 [nmfdiagW, nmfdiagH] = NMFdiag(At, W0, H0, paramNMFdiag);
53
54 % create mosaic, replace magnitude by complex frames
55 contY = Xs*nmfdiagH;
56
57 % visualize
58 paramVis = [];
59 paramVis.deltaF = deltaF;
60 paramVis.deltaT = deltaT;
61 fh1 = visualizeComponentsNMF(At, nmfdiagW, nmfdiagH, [],
       paramVis);
62
63 % save result
64 saveas(fh1,[outPath,'LetItBee_NMFdiag.png']);
65
66 % resynthesize using Griffin_Lim, 50 iterations by default
67 [Xout, Pout, res] = LSEE_MSTFTM_GriffinLim(contY, paramSTFT);
68
69 % save result
70 audiowrite([outPath,'LetItBee_NMFdiag_with_target_',
       filenameTarget],res,fs);
```

Listing L.3: Example of audio mosaicing using NMF model with diagonality-enhanced activation matrix and fixed templates, following [24].

We will now go through the code in Listing L.3. In lines 2–14 we load the audio files for both source and target signals, making them mono for further processing. We then compute the source spectrograms `Xs` (complex-valued), `As` (magnitude), and `Ps` (phase) by calling `forwardSTFT()` in line 26, and the same for the target signal (yielding `Xt`, `At`, and `Pt`, in line 27). In lines 30–33 we obtain the spectrogram dimensions, as well as the time and frequency resolutions under the current settings. Since we want to learn activations that will tell us which source frames to use to synthesize the target, we initialize the activation matrix `H0` with random values in line 37. We then initialize the template matrix `W0` with the source magnitude spectrogram `As`, normalized so that each column has unit sum (line 40). We also normalize the complex-valued spectrogram `Xs` on line 41. The following lines 44–49 define crucial parameters in the structure `paramNMFdiag`. In line 44 we set `fixW = 1`, indicating that the template matrix `W0` should not be updated during the NMF learning (i. e., we do not want to modify the source's timbral characteristics). In line 45 we set the number of iterations to 20. In line 46 we set the degree of polyphony to 10—this means that during learning, for every column in the activation matrix, the 10 highest entries will keep their original magnitude, and the rest will be scaled down. In line 47 we set `continuity.length` to 7, which controls the filter kernel length that will be used to smooth the diagonal lines that we wish

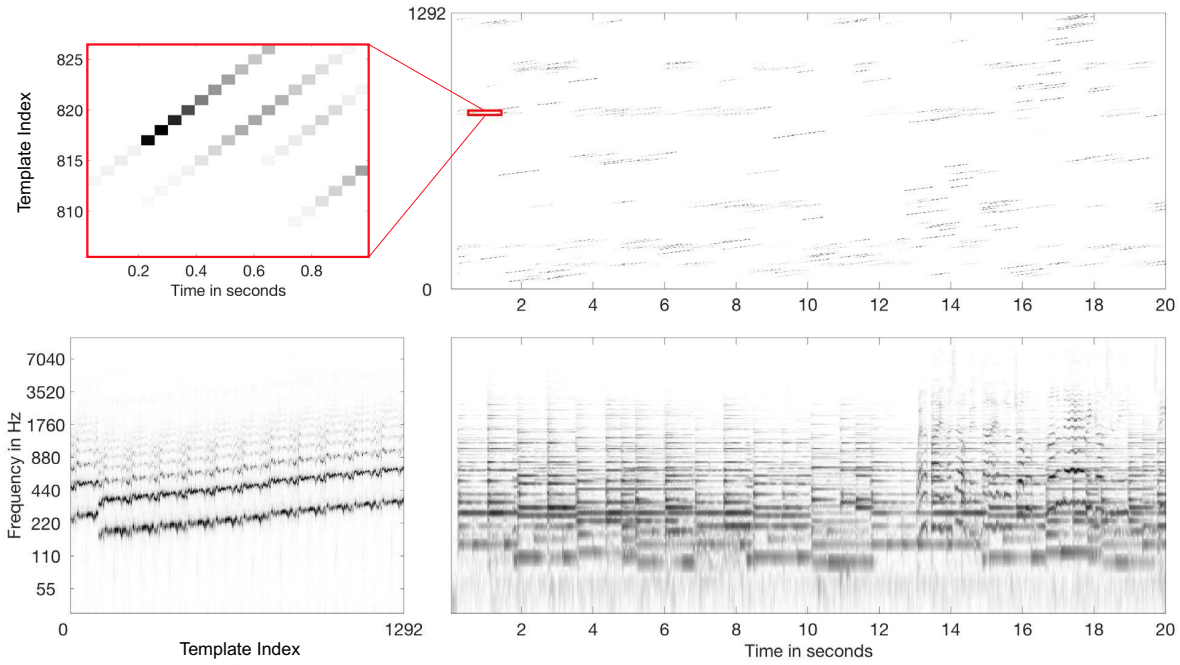


Figure 3: Visual output generated by code in Listing L.3: diagonally enhanced NMF for audio mosaicing. Top right: Diagonally enhanced activation matrix H. Top left: Detail of diagonal structures in subregion of activation matrix (subplot was added manually). Bottom right: Approximated magnitude spectrogram resulting from the diagonal NMF model $\mathrm{U} = \mathrm{W} \cdot \mathrm{H}$. Bottom left: Magnitude spectrogram used as timbral source for mosaicing, used directly as template matrix W. The timbral source consists of bees buzzing sounds, pitched up stepwise throughout an entire octave.

to enhance in the activation matrix. This choice is empirical since it corresponds to perceived sound quality and can be varied according to combinations of source and target signals. In line 48 we set `continuity.grid` to 5, indicating that we want to filter the activation matrix at every fifth iteration. If the user wishes to intervene less in the NMF algorithm updates, a higher number can be set (i. e., filtering will be performed every `continuity.grid` iterations).

We use the field `parameter.sparsen` (line 49) to increase the distance between neighboring diagonal structures. We now call the main function, `NMFdiag()` (line 52), which returns the templates `nmfdiagW` and diagonally enhanced activations `nmfdiagH`. It only remains to create the mosaic by multiplying the learned activations `nmfdiagH` with the previously normalized source complex-valued spectrogram, in line 55. In lines 58–60 we set the visualization parameters as fields of `paramVis`, and call the visualization function in line 61, producing the output seen in Figure 3, and saving it as a `png` file in line 64. To make the mosaicing result audible, we apply `LSEE_MSTFTM_GriffinLim()` to the complex mosaic spectrogram `contY` (line 67) and write the audio file to disk (line 70).

7. SUMMARY

We have presented the NMF Toolbox, an easy-to-use collection of illustrated code examples intended for research and learning of the principles behind NMF, through real-world music processing applications. In particular, the toolbox provides baseline implementations and a small dataset for tasks such as drum source separation, structure analysis of electronic music, and audio mosaicing. Although the toolbox is not optimized for high execution speed and is not comprehensive (considering the large number of existing NMF variants), we hope that it serves an educational purpose and as a starting point for integrating these techniques into existing projects.

8. ACKNOWLEDGMENTS

This work has been supported by the German Research Foundation (DFG MU 2686/10-1).

9. REFERENCES

[1] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Proc. Neural Information Processing Systems (NIPS)*, Denver, Colorado, USA, November 2000, pp. 556–562.

[2] N. Bertin, R. Badeau, and E. Vincent, "Enforcing harmonicity and smoothness in bayesian non-negative matrix factorization applied to polyphonic music transcription," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 18, no. 3, pp. 538–549, 2010.

[3] C.-W. Wu and A. Lerch, "Drum transcription using partially fixed non-negative matrix factorization with template adaptation," in *Proc. Intl. Soc. for Music Information Retrieval Conf. (ISMIR)*, Málaga, Spain, October 2015, pp. 257–263.

[4] M. Kim, J. Yoo, K. Kang, and S. Choi, "Nonnegative matrix partial co-factorization for spectral and temporal drum source separation," *IEEE J. Selected Topics Signal Processing*, vol. 5, no. 6, pp. 1192–1204, 2011.

[5] F. J. Cañadas-Quesada, D. FitzGerald, P. Vera-Candeas, and N. Ruiz-Reyes, "Harmonic-percussive sound separation using rhythmic information from non-negative matrix factorization in single-channel music recordings," in *Proc. Intl. Conf. Digital Audio Effects (DAFx)*, Edinburgh, UK, September 2017, pp. 276–282.

[6] C. Laroche, H. Papadopoulos, M. Kowalski, and G. Richard, "Drum extraction in single channel audio signals using multi-layer non negative matrix factor deconvolution," in *Proc. IEEE Intl. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, New Orleans, Louisiana, USA, March 2017, pp. 46–50.

[7] C. Laroche, M. Kowalski, H. Papadopoulos, and G. Richard, "Hybrid projective nonnegative matrix factorization with drum dictionaries for harmonic/percussive source separation," *IEEE/ACM Trans. Audio, Speech, and Language Processing*, vol. 26, no. 9, pp. 1499–1511, 2018.

[8] D. F. Yela, S. Ewert, D. FitzGerald, and M. B. Sandler, "Interference reduction in music recordings combining kernel additive modelling and non-negative matrix factorization," in *Proc. IEEE Intl. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, New Orleans, Louisiana, USA, March 2017, pp. 51–55.

[9] C. Dittmar, P. López-Serrano, and M. Müller, "Unifying local and global methods for harmonic-percussive source separation," in *Proc. IEEE Intl. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, Calgary, Canada, April 2018, pp. 176–180.

[10] P. López-Serrano, M. E. P. Davies, J. Hockman, C. Dittmar, and M. Müller, "Break-informed audio decomposition for interactive redrumming," in *Late Breaking and Demo Session of the Intl. Soc. for Music Information Retrieval Conf. (ISMIR)*, Paris, France, September 2018.

[11] T. Nakamura, H. Kameoka, K. Yoshii, and M. Goto, "Timbre replacement of harmonic and drum components for music audio signals," in *Proc. IEEE Intl. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, Florence, Italy, May 2014, pp. 7520–7524.

[12] C. Dittmar, K. F. Hildebrand, D. Gärtner, M. Winges, F. Müller, and P. Aichroth, "Audio forensics meets music information retrieval – a toolbox for inspection of music plagiarism," in *Proc. European Signal Processing Conf. (EUSIPCO)*, Bucharest, Romania, August 2012, pp. 1249–1253.

[13] S. Gururani and A. Lerch, "Automatic sample detection in polyphonic music," in *Proc. Intl. Soc. for Music Information Retrieval Conf. (ISMIR)*, Suzhou, China, October 2017, pp. 264–271.

[14] R. J. Weiss and J. P. Bello, "Unsupervised discovery of temporal structure in music," *IEEE J. Selected Topics in Signal Processing*, vol. 5, pp. 1240–1251, 2011.

[15] F. Kaiser and T. Sikora, "Music structure discovery in popular music using non-negative matrix factorization," in *Proc. Intl. Soc. for Music Information Retrieval Conf. (ISMIR)*, Utrecht, The Netherlands, August 2010, pp. 429–434.

[16] P. Seetharaman and B. Pardo, "Simultaneous separation and segmentation in layered music," in *Proc. Intl. Conf. Music Information Retrieval (ISMIR)*, New York City, USA, August 2016, pp. 495–501.

[17] S. Ewert and M. Müller, "Using score-informed constraints for NMF-based source separation," in *Proc. IEEE Intl. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, Kyoto, Japan, March 2012, pp. 129–132.

[18] Ö. Izmirli, "Localized key finding from audio using nonnegative matrix factorization for segmentation," in *Proc. Intl. Soc. for Music Information Retrieval Conf. (ISMIR)*, Vienna, Austria, September 2007, pp. 195–200.

[19] A. Ozerov, E. Vincent, and F. Bimbot, "A general flexible framework for the handling of prior information in audio source separation," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 20, no. 4, pp. 1118–1133, May 2012.

[20] U. Zölzer, *DAFX: Digital Audio Effects*, Wiley Publishing, 2nd edition, 2011.

[21] M. Müller, *Fundamentals of Music Processing*, Springer Verlag, 2015.

[22] P. Smaragdis, "Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs," in *Proc. Intl. Conf. Independent Component Analysis and Blind Signal Separation (ICA)*, Granada, Spain, September 2004, pp. 494–499.

[23] P. López-Serrano, C. Dittmar, J. Driedger, and M. Müller, "Towards modeling and decomposing loop-based electronic music," in *Proc. Intl. Conf. Music Information Retrieval (ISMIR)*, New York City, USA, August 2016, pp. 502–508.

[24] J. Driedger, T. Prätzlich, and M. Müller, "Let It Bee – Towards NMF-inspired audio mosaicing," in *Proc. Intl. Soc. for Music Information Retrieval Conf. (ISMIR)*, Málaga, Spain, October 2015, pp. 350–356.

[25] A. Cichocki, R. Zdunek, A. H. Phan, and S. Amari, *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-Way Data Analysis and Blind Source Separation*, John Wiley and Sons, 2009.

[26] C. Dittmar and M. Müller, "Reverse engineering the Amen break – score-informed separation and restoration applied to drum recordings," *IEEE/ACM Trans. Audio, Speech, and Language Processing*, vol. 24, no. 9, pp. 1531–1543, 2016.

[27] J. Driedger, H. Grohganz, T. Prätzlich, S. Ewert, and M. Müller, "Score-informed audio decomposition and applications," in *Proc. ACM Intl. Conf. Multimedia (ACM-MM)*, Barcelona, Spain, October 2013, pp. 541–544.

[28] D. W. Griffin and J. S. Lim, "Signal estimation from modified short-time Fourier transform," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 32, no. 2, pp. 236–243, 1984.

[29] A. Liutkus and R. Badeau, "Generalized Wiener filtering with fractional power spectrograms," in *Proc. IEEE Intl. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, Brisbane, Australia, April 2015, pp. 266–270.

[30] C. Dittmar, J. Driedger, M. Müller, and J. Paulus, "An experimental approach to generalized Wiener filtering in music source separation," in *Proc. European Signal Processing Conf. (EUSIPCO)*, Budapest, Hungary, August 2016, pp. 1743–1747.

A GENERAL-PURPOSE DEEP LEARNING APPROACH TO MODEL TIME-VARYING AUDIO EFFECTS

Marco A. Martínez Ramírez, Emmanouil Benetos, Joshua D. Reiss

Centre for Digital Music,
Queen Mary University of London
London, United Kingdom
m.a.martinezramirez, emmanouil.benetos, joshua.reiss@qmul.ac.uk

ABSTRACT

Audio processors whose parameters are modified periodically over time are often referred as time-varying or modulation based audio effects. Most existing methods for modeling these type of effect units are often optimized to a very specific circuit and cannot be efficiently generalized to other time-varying effects. Based on convolutional and recurrent neural networks, we propose a deep learning architecture for generic black-box modeling of audio processors with long-term memory. We explore the capabilities of deep neural networks to learn such long temporal dependencies and we show the network modeling various linear and nonlinear, time-varying and time-invariant audio effects. In order to measure the performance of the model, we propose an objective metric based on the psychoacoustics of modulation frequency perception. We also analyze what the model is actually learning and how the given task is accomplished.

1. INTRODUCTION

Modulation based or time-varying audio effects involve audio processors or effect units that include a modulator signal within their analog or digital implementation [1]. These modulator signals are in the low frequency range (usually below 20 Hz). Their waveforms are based on common periodic signals such as sinusoidal, squarewave or sawtooth oscillators and are often referred to as a Low Frequency Oscillator (LFO). The LFO periodically modulates certain parameters of the audio processors to alter the timbre, frequency, loudness or spatialization characteristics of the audio. This differs from time-invariant audio effects which do not change their behavior over time. Based on how the LFO is employed and the underlying signal processing techniques used when designing the effect units, we can classify modulation based audio effects into *time-varying filters* such as phaser or wah-wah; *delay-line based* effects such as flanger or chorus; and *amplitude modulation* effects such as tremolo or ring modulator [2].

The *phaser* effect is a type of time-varying filter implemented through a cascade of notch or all-pass filters. The characteristic sweeping sound of this effect is obtained by modulating the center frequency of the filters, which creates phase cancellations or enhancements when combining the filter's output with the input audio. Similarly, the *wah-wah* is based on a bandpass filter with a variable center frequency, usually controlled by a pedal. If the center frequency is modulated by an LFO or an envelope follower, the effect is commonly called *auto-wah*.

Delay-line based audio effects, as in the case of *flanger* and *chorus*, are based on the modulation of the length of the delay lines. A *flanger* is implemented via a modulated comb filter whose output is mixed with the input audio. Unlike the phaser, the notch and peak frequencies caused by the flanger's sweep comb filter effect are equally spaced in the spectrum, thus causing the known metallic sound associated with this effect. A *chorus* occurs when mixing the input audio with delayed and pitch modulated copies of the original signal. This is similar to various musical sources playing the same instrument but slightly shifted in time. *vibrato* is digitally implemented as a delay-line based audio effect, where pitch shifting is achieved when periodically varying the delay time of the input waveform [3].

Tremolo is an amplitude modulation effect where an LFO is used to directly vary the amplitude of the incoming audio, creating in this way a perceptual temporal fluctuation. A *ring modulator* is also based on amplitude modulation, but the modulation is achieved by having the input audio multiplied by a sinusoidal oscillator with higher carrier frequencies. In the analog domain, this effect is commonly implemented with a diode bridge, which adds a nonlinear behavior and a distinct sound to this effect unit. Another type of modulation based effect that combines amplitude, pitch and spatial modulation is the *Leslie speaker*, which is implemented by a rotating horn and a rotating woofer inside a wooden cabinet. This effect can be interpreted as a combination of *tremolo*, Doppler effect and reverberation [4].

Most of these effects can be implemented directly in the digital domain through the use of digital filters and delay lines. Nevertheless, modeling specific effect units or analog circuits has been heavily researched and remains an active field. This is because hardware effect units are characterized by the nonlinearities introduced by certain circuit components. Musicians often prefer the analog counterparts because the digital implementations may lack this behavior, or because the digital simulations make certain assumptions when modeling specific nonlinearities.

Virtual analog methods for modeling such effect units mainly involve circuit modeling and optimization for specific analog components such as operational amplifiers or transistors. This often requires assumptions or models that are too specific for a certain circuit. Such models are also not easily transferable to different effects units since expert knowledge of the type of circuit being modeled is required, i.e. specific linear and nonlinear components.

Prior to this work, deep learning architectures have not yet been implemented to model time-varying audio effects. Thus, building on [5, 6], we propose a general-purpose deep learning approach to model this type of audio effects. Using convolutional,

recurrent and fully connected layers, we explore how a deep neural network (DNN) can learn the long temporal dependencies which characterizes these effect units as well as the possibilities to match nonlinearities within the audio effects. We include Bidirectional Long Short-Term Memory (Bi-LSTM) neural networks and explore their capabilities when learning time-varying transformations. We explore linear and nonlinear time-varying emulation as a content-based transformation without explicitly obtaining the solution of the time-varying system.

We show the model matching modulation based audio effects such as *chorus*, *flanger*, *phaser*, *tremolo*, *vibrato*, *tremolo-wah*, *ring modulator* and *Leslie speaker*. We investigate the capabilities of the model when adding further nonlinearities to the linear time-varying audio effects. Furthermore, we extend the applications of the model by including nonlinear time-invariant audio effects with long temporal dependencies such as *auto-wah*, *compressor* and *multiband compressor*. Finally, we measure performance of the model using a metric based on the modulation spectrum.

The paper is structured as follows. In Section 2 we present the relevant literature related to virtual analog of modulation based audio effects. Section 3 gives details of our model, the modulation based effect tasks and the proposed evaluation metric. Sections 4 and 5 show the analysis, obtained results, and the respective conclusion.

2. BACKGROUND

2.1. Virtual analog modeling of time-varying audio effects

Virtual analog audio effects aim to simulate an effect unit and recreate the sound of an analog reference circuit. Much of the active research models nonlinear audio processors such as distortion effects, compressors, amplifiers or vacuum tubes [7, 8, 9]. With respect to modeling time-varying audio effects, most of the research has been applied to develop white-box methods, i.e. in order to model the effect unit a complete study of the internal circuit is carried out. These methods use circuit simulation techniques to characterize various analog components such as diodes, transistors, operational amplifiers or integrated circuits.

In [10], *phasers* implemented via Junction Field Effect Transistors (JFET) and Operational Transconductance Amplifiers (OTA) were modeled using circuit simulation techniques that discretize the differential equations that describe these components. Using a similar circuit modeling procedure, delay-line based effects are modeled, such as *flanger* and *chorus* as implemented with Bucket Brigade Delay (BBD) chips. BBD circuits have been widely used in analog delay-line based effect units and several digital emulations have been investigated. [11] emulated BBD devices through circuit analysis and electrical measurements of the linear and nonlinear elements of the integrated circuit. [12] modeled BBDs as delay-lines with fixed length but variable sample rate.

Based on BBD circuitry, a *flanger* effect was modeled in [13] via the nodal DK-method. This is a common method in virtual analog modeling [14] where nonlinear filters are derived from the differential equations that describe an electrical circuit. In [15], a *wah-wah* pedal is implemented using the nodal DK-method and the method is extended to model the temporal fluctuations introduced by the continuous change of the pedal. In [16], the *MXR Phase 90* phaser effect is modeled via a thorough circuit analysis and the DK-method. This effect unit is based on JFETs, and voltage and current measurements were performed to obtain the nonlinear characteristics of the transistors.

Amplitude modulation effects such as an analog *ring modulator* were modeled in [17], where the diode bridge is emulated as a network of static nonlinearities. [18] modeled the rotating horn of the *Leslie speaker* via varying delay-lines, artificial reverberation and physical measurements from the rotating loudspeaker. [19] also modeled the *Leslie speaker* and achieved frequency modulation through time-varying spectral delay filters and amplitude modulation using a modulator signal. In both *Leslie speaker* emulations, various physical characteristics of the effect are not taken into account, such as the frequency-dependent directivity of the loudspeaker and the effect of the wooden cabinet.

In [20], gray-box modeling was proposed for linear time-varying audio effects. This differs from white-box modeling, since the method was based on input-output measurements but the time-varying filters were based on knowledge of analog *phasers*. In this way, *phaser* emulation was achieved by multiple measurements of the impulse response of a cascade of all-pass filters.

Another method to model time-varying audio effects is discretizing electrical circuit elements via Wave Digital Filters (WDF). The Hammond organ vibrato/chorus was modeled using WDFs in [21], and [22] performed circuit modeling through WDFs to emulate modulation based effects that use OTAs.

2.2. End-to-end deep neural networks

End-to-end deep learning is based on the idea that an entire problem can be taken as a single indivisible task which must be learned from input to output. Deep learning architectures using this principle have recently been researched in the music information retrieval field [23, 24, 25], since the amount of required prior knowledge may be reduced and engineering effort minimized by learning directly from raw audio [26]. Recent work also demonstrated the feasibility of these architectures for audio synthesis and audio effects modeling. [27, 28] proposed models that synthesize audio waveforms and [29] obtained a model capable of performing singing voice synthesis.

End-to-end deep neural networks for audio effects modeling were implemented in [5], where Equalization (EQ) matching was achieved with convolutional neural networks (CNN). Also, [6] presented a deep learning architecture for modeling nonlinear processors such as distortion, overdrive and amplifier emulation. The DNN is capable of modeling an arbitrary combination of linear and nonlinear memoryless audio effects, but does not generalize to transformations with long temporal dependencies such as modulation based audio effects.

3. METHODS

3.1. Model

The model is entirely based on the time-domain and operates with raw audio as the input and processed audio as the output. It is divided into three parts: adaptive front-end, latent-space and synthesis back-end. A block diagram can be seen in Fig. 1 and its structure is described in detail in Table 1. We build on the architecture from [6], since we incorporate Bi-LSTMs into the latent-space and we modify the structure of the synthesis back-end in order to allow the model to learn nonlinear time-varying transformations.

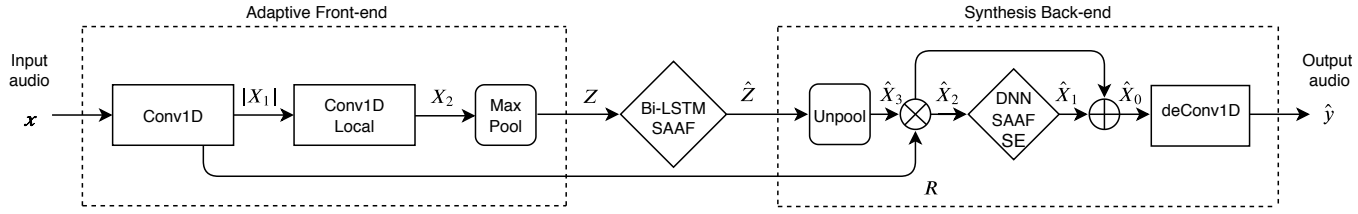


Figure 1: *Block diagram of the proposed model; adaptive front-end, Bi-LSTM and synthesis back-end.*

Table 1: *Detailed architecture of a model with input frame size of* 4096 *samples and* ± 4 *context frames.*

Layer	Output shape	Units	Output
Input	(9, 4096, 1)	.	$\boldsymbol{x}$
Conv1D	(9, 4096, 32)	32(64)	X_1
Residual	(4096, 32)	.	R
Abs	(9, 4096, 32)	.	.
Conv1D-Local	(9, 4096, 32)	32(128)	.
Softplus	(9, 4096, 32)	.	X_2
MaxPooling	(9, 64, 32)	.	Z
Bi-LSTM	(64, 128)	64	.
Bi-LSTM	(64, 64)	32	.
Bi-LSTM	(64, 32)	16	.
SAAF	(64, 32)	25	$\hat{Z}$
Unpooling	(4096, 32)	.	$\hat{X}_3$
Multiply	(4096, 32)	.	$\hat{X}_2$
Dense	(4096, 32)	32	.
Dense	(4096, 16)	16	.
Dense	(4096, 16)	16	.
Dense	(4096, 32)	32	.
SAAF	(4096, 32)	25	.
Abs	(4096, 32)	.	.
Global Average	(1, 32)	.	.
Dense	(1, 512)	512	.
Dense	(1, 32)	32	.
Multiply	(4096, 32)	.	$\hat{X}_1$
Add	(4096, 32)	.	$\hat{X}_0$
deConv1D	(4096, 1)	.	$\hat{y}$

3.2. Adaptive front-end

The front-end performs time-domain convolutions with the incoming audio. It follows a filter bank architecture and is designed to learn a latent representation for each audio effect modeling task. It consists of a convolutional encoder which contains two CNN layers, one pooling layer and one residual connection. This residual connection is used by the back-end to facilitate the synthesis of the waveform based on the specific time-varying transformation.

In order to allow the model to learn long-term memory dependencies, the input consists of the current audio frame $x(t)$ concatenated with the k previous and k subsequent frames. These frames are of size N and sampled with a hop size τ. The input $\boldsymbol{x}$ is described by (1).

$$\boldsymbol{x} = x(t + j\tau), j = -k, ..., k \tag{1}$$

The first convolutional layer has 32 one-dimensional filters of size 64 and is followed by the *absolute value* as nonlinear activation function. The operation performed by the first layer can be described by (2).

$$\boldsymbol{X}_1 = \boldsymbol{x} * \boldsymbol{W}_1 \tag{2}$$

Where $\boldsymbol{X}_1$ is the feature map after the input audio $\boldsymbol{x}$ is convolved with the kernel matrix $\boldsymbol{W}_1$. $\boldsymbol{R}$ is the corresponding row in $\boldsymbol{X}_1$ for the frequency band decomposition of the current input frame $x(t)$. The back-end does not directly receive information from the past and subsequent context frames. The second layer has 32 filters of size 128 and each filter is locally connected. We follow a filter bank architecture since each filter is only applied to its corresponding row in $|\boldsymbol{X}_1|$ and so we significantly decrease the number of trainable parameters. This layer is followed by the *softplus* nonlinearity [30], described by (3).

$$\boldsymbol{X}_2 = \mathit{softplus}(|\boldsymbol{X}_1| * \boldsymbol{W}_2) \tag{3}$$

Where $\boldsymbol{X}_2$ is the second feature map obtained after the local convolution with $\boldsymbol{W}_2$, the kernel matrix of the second layer. The *max-pooling* operation is a moving window of size $N/64$ applied over $\boldsymbol{X}_2$, where the maximum value within each window corresponds to the output.

By using the *absolute value* as activation function of the first layer and by having larger filters $\boldsymbol{W}_2$, we expect the front-end to learn smoother representations of the incoming audio, such as envelopes [24]. All convolutions and pooling operations are time distributed, i.e the same convolution or pooling operation is applied to each of the $2 \cdot k + 1$ input frames.

3.3. Bidirectional LSTMs

The latent-space consists of three Bi-LSTM layers of 64, 32, and 16 units respectively. Bi-LSTMs are a type of recurrent neural network that can access long-term context from both backward and forward directions [31]. Bi-LSTMs are capable of learning long temporal dependencies when processing timeseries where the context of the input is needed [32].

The Bi-LSTMs process the latent-space representation Z. Z is learned by the front-end and contains information regarding the $2 \cdot k + 1$ input frames. These recurrent layers are trained to reduce the dimension of Z, while also learning a nonlinear modulation $\hat{Z}$. This new latent representation is fed into the synthesis back-end in order to reconstruct an audio signal that matches the time-varying task. Each Bi-LSTM has dropout and recurrent dropout rates of 0.1 and the first two layers have the *hyperbolic tangent* as activation function.

The performance of CNNs in regression tasks has improved by using adaptive activation functions [33]. So we add a Smooth Adaptive Activation Function (SAAF) as the nonlinearity for the last layer. SAAFs consist of piecewise second order polynomials which can approximate any continuous function and are regularized under a Lipschitz constant to ensure smoothness. As shown

in [6], SAAFs can be used within deep neural networks to model nonlinearities in audio processing tasks.

3.4. Synthesis back-end

The synthesis back-end accomplishes the reconstruction of the target audio by processing the current input frame $x(t)$ and the nonlinear modulation $\hat{Z}$. The back-end consists of an unpooling layer, a DNN block with SAAF and Squeeze-and-Excitation (SE) [34] layers (DNN-SAAF-SE) and a final CNN layer.

The DNN-SAAF-SE block consists of 4 fully connected (FC) layers of 32, 16, 16 and 32 hidden units respectively. Each FC layer is followed by the *hiperbolic tangent* function except for the last one, which is followed by a SAAF layer. Overall, each SAAF is locally connected and each function consists of 25 intervals between -1 to 1.

The SE blocks explicitly model interdependencies between channels by adaptively scaling the channel-wise information of feature maps [34]. The SE dynamically scales each of the 32 channels and follows the structure from [35]. It consists of a global average pooling operation followed by two FC layers of 512 and 32 hidden units respectively. The FC layers are followed by a rectifier linear unit (*ReLU*) and *sigmoid* activation functions accordingly. Since the feature maps of the model are based on time-domain waveforms, we incorporate an *absolute value* layer before the global average pooling operation.

The back-end matches the time-varying task by the following steps. First, a discrete approximation of $\boldsymbol{Z}$ ($\hat{\boldsymbol{X}}_3$) is obtained by an upsampling operation. Then the feature map $\hat{\boldsymbol{X}}_2$ is the result the element-wise multiplication of the residual connection $\boldsymbol{R}$ and $\hat{\boldsymbol{X}}_3$. This can be seen as a frequency dependent amplitude modulation between the learned modulator $\boldsymbol{Z}$ and the frequency band decomposition $\boldsymbol{R}$.

$$\hat{\boldsymbol{X}}_{\mathbf{2}} = \hat{\boldsymbol{X}}_3 \cdot \boldsymbol{R} \tag{4}$$

The feature map $\hat{\boldsymbol{X}}_1$ is obtained when the nonlinear and channel-wise scaled filters from the DNN-SAAF-SE block are applied to the modulated frequency band decomposition $\hat{\boldsymbol{X}}_2$. Then, $\hat{\boldsymbol{X}}_1$ is added back to $\hat{\boldsymbol{X}}_2$, acting as a nonlinear delay-line.

$$\hat{\boldsymbol{X}}_{\mathbf{0}} = \hat{\boldsymbol{X}}_1 + \hat{\boldsymbol{X}}_2 \tag{5}$$

The last layer corresponds to the deconvolution operation, which can be implemented by transposing the first layer transform. This layer is not trainable since its kernels are transposed versions of $\boldsymbol{W}_1$. In this way, the back-end reconstructs the audio waveform in the same manner that the front-end decomposed it. The complete waveform is synthesized using a *hanning* window and constant overlap-add gain.

All convolutions are along the time dimension and all strides are of unit value. The models have approximately $300k$ trainable parameters, which, within a deep learning context, represents a model that is not very large or difficult to train.

3.5. Training

The training of the model is performed in two steps. The first step is to train only the convolutional layers for an unsupervised learning task, while the second step consists of an end-to-end supervised learning task based on a given time-varying target. During the first step only the weights of *Conv1D* and *Conv1D-Local* are trained and both the raw audio $x(t)$ and wet audio $y(t)$ are used as input and target functions. This means the model is being prepared to reconstruct the input and target data in order to have a better fitting when training for the time-varying task. Only during this step, the unpooling layer of the back-end uses the time positions of the maximum values recorded by the *max-pooling* operation.

Once the model is pretrained, the Bi-LSTM and DNN-SAAF-SE layers are incorporated into the model, and all the weights of the convolutional, recurrent, dense and activation layers are updated. Since small amplitude errors are as important as large ones, the loss function to be minimized is the mean absolute error between the target and output waveforms. We explore input size frames from 1024 to 8192 samples and we always use a hop size of 50%. The batch size consisted of the total number of frames per audio sample.

Adam is used as optimizer and we perform the pretraining for 200 epochs and the supervised training for 500 epochs. In order to speed convergence, during the second training step we start with a learning rate of $5{\cdot}10^{-5}$ and we reduce it by 50% every 150 epochs. We select the model with the lowest error for the validation subset.

3.6. Dataset

Modulation based audio effects such as *chorus*, *flanger*, *phaser*, *tremolo* and *vibrato* were obtained from the *IDMT-SMT-Audio-Effects* dataset [36]. It corresponds to individual 2-second notes and covers the common pitch range of various 6-string electric guitars and 4-string bass guitars.

The recordings include the raw notes and their respective effected versions for 3 different settings for each effect. For our experiments, for each of the above effects, we only use the setting $\#2$ from where we obtained the unprocessed and processed audio for bass guitar. In addition, processing the bass guitar raw audio, we implemented an *auto-wah* with a peak filter whose center frequency ranges from 500 Hz to 3 kHz and modulated by a 5 Hz sinusoidal.

Since the previous audio effects are linear time-varying, we further test the capabilities of the model by adding a nonlinearity to each of these effects. Thus, using the bass guitar wet audio, we applied an *overdrive* (gain$= +10$dB) after each modulation based effect.

We also use virtual analog implementations of a *ring modulator* and a *Leslie speaker* to process the electric guitar raw audio.

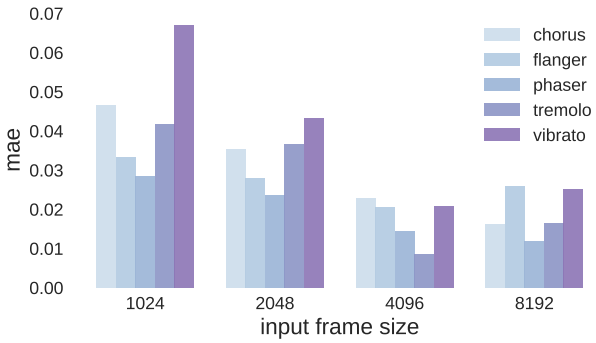


Figure 2: ***mae*** *values for linear time-varying tasks with different input size frames.*

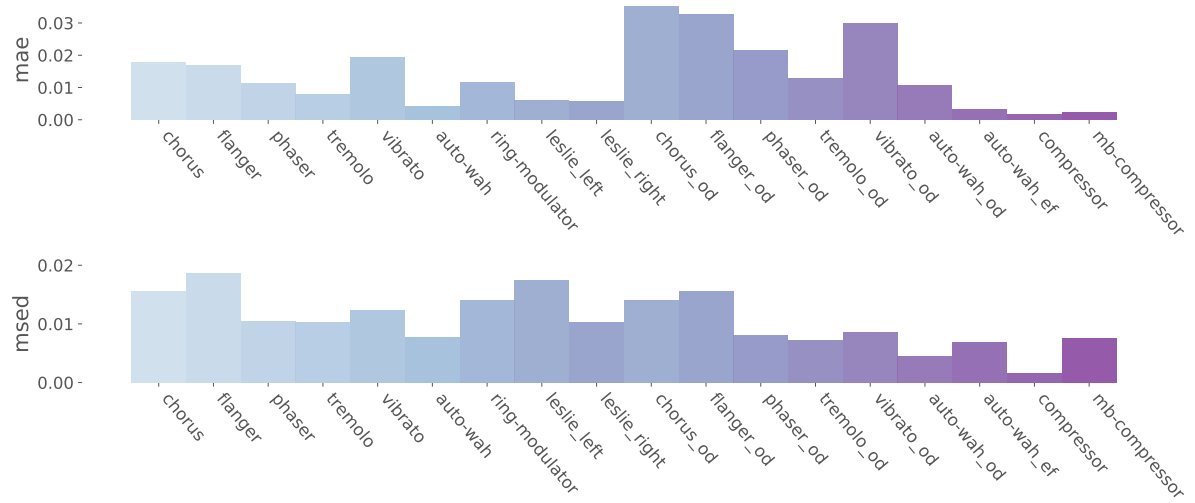


Figure 3: ***mae*** *and* ***msed*** *values with the test dataset for all the time-varying tasks.* od*,* ef *and* mb *mean overdrive, envelope follower and multiband respectively.*

The *ring modulator* implementation[1] is based on [17] and we use a modulator signal of 5 Hz. The *Leslie speaker* implementation[2] is based on [18] and we model each of the stereo channels.

Finally, we also explore the capabilities of the model with non-linear time-invariant audio effects with long temporal dependencies, such as *compressors* and *auto-wah*. We use the *compressor* and *multiband compressor* from *SoX*[3] to process the electric guitar raw audio. The settings of the *compressor* are as follows: attack time 10 ms, release time 100 ms, knee 1 dB, ratio 4:1 and threshold -40 dB. The *multiband compressor* has 2 bands with a crossover frequency of 500 Hz, attack time: 5 ms and 625 μs, decay time: 100 ms and 12.5 ms, knee: 0 dB and 6 dB, ratio: 3:1 and 6:1 and threshold: -30 dB and -60 dB.

Similarly, we use an *auto-wah* implementation[4] with an envelope follower and a peak filter which center frequency modulates between 500 Hz to 3 kHz.

For each time-varying task we use 624 raw and effected notes and both the test and validation samples correspond to 5% of this subset each. The recordings were downsampled to 16 kHz and amplitude normalization was applied with exception to the time-invariant audio effects.

3.7. Evaluation

Two metrics were used when testing the models with the various test subsets. Since the mean absolute error depends on the amplitude of the output and target waveforms, before calculating this error, we normalize the energy of the target and the output and define it as the energy-normalized mean absolute error (*mae*).

We also propose an objective metric which mimics human perception of amplitude and frequency modulation. The modulation spectrum uses time-frequency theory integrated with the psychoacoustics of modulation frequency perception, thus, providing long-term knowledge of temporal fluctuation patterns [37]. We propose the modulation spectrum euclidean distance (*msed*), which is based on the audio features from [38] and [39] and is defined as follows:

- A Gammatone filter bank is applied to the target and output entire waveforms. In total we use 12 filters, with center frequencies spaced logarithmically from 26 Hz to 6950 Hz.
- The envelope of each filter output is calculated via the magnitude of the Hilbert transform and downsampled to 400 Hz.
- A Modulation filter bank is applied to each envelope. In total we use 12 filters, with center frequencies spaced logarithmically from 0.5 Hz to 100 Hz.
- The Fast Fourier Transform (FFT) is calculated for each modulation filter output of each Gammatone filter. The energy is normalized by the DC value and summarized in the following bands: 0.5-4 Hz, 4.5-10 Hz, 10.5-20 Hz and 20.5-100 Hz.
- The *msed* metric is the mean euclidean distance between the energy values at these 4 bands.

4. RESULTS & ANALYSIS

First, we explore the capabilities of Bi-LSTMs to learn long-term temporal dependencies. Fig. 2 shows the mae results of the test dataset for different input frame sizes and various linear time-varying tasks. The most optimal results are with an input size of 4096 samples, since shorter frame sizes represent a higher error and 8192 samples do not represent a significant improvement. Since the average modulation frequency in our tasks is 2 Hz, for each input size we select a k that covers one period of this modulator signal. Thus, for the rest of our experiments, we use an input size of 4096 samples and $k = 4$ for the number of past and subsequent frames.

[1] https://github.com/nrlakin/robot_voice/blob/master/robot.py
[2] https://ccrma.stanford.edu/software/snd/snd/leslie.cms
[3] http://sox.sourceforge.net/
[4] https://github.com/lucieperrotta/ASP

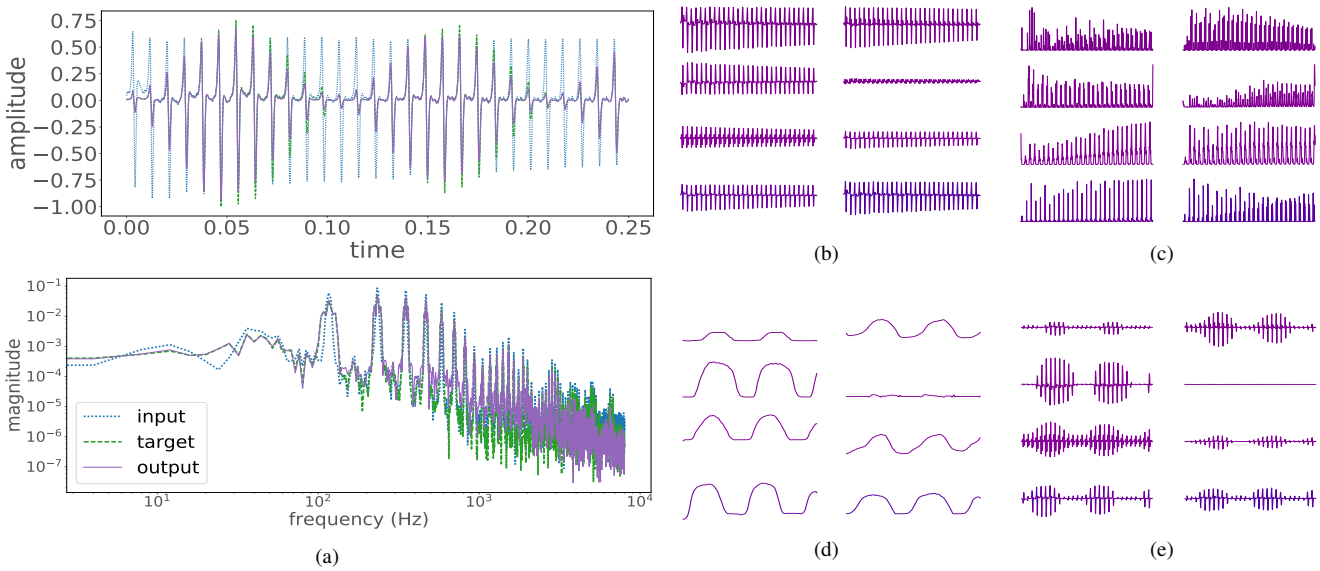


Figure 4: *Various internal plots for the test dataset of the* ***tremolo*** *modeling task. 4a) Input, target and output frames of* 4096 *samples and their respective FFT magnitudes. 4b) For the input frame* $x(t)$*, respective* 8 *rows from* $\boldsymbol{R}$*. 4c) Following the filter bank architecture, respective* 8 *rows from* $\boldsymbol{X}_2$*. 4d) From* $\hat{\boldsymbol{Z}}$*, corresponding* 8 *modulator signals learned by the Bi-LSTM layer. 4e) In the same manner,* 8 *rows from* $\hat{\boldsymbol{X}}_0$*, which is the input to the deconvolution layer prior to obtaining the output frame* $\hat{y}(t)$*. Vertical axes in 4b)-4e) are unitless and horizontal axes correspond to time.*

The training procedures were performed for each type of time-varying and time-invariant audio effect. Then, the models were tested with samples from the test dataset and the audio results are available online[5]. Fig. 3 shows the *mae* and *msed* for all the test subsets. To provide a reference, the mean *mae* and *msed* values between input and target waveforms are 0.15 and 0.11 respectively. It can be seen that the model performed well on each audio effect modeling task. Overall, the model achieved better results with amplitude modulation and time-varying filter audio effects, although delay-line based effects were also successfully modeled.

Fig. 4 visualizes the functioning of the model for the *tremolo* task. It shows how the model processes the input frame $x(t)$ into the different frequency maps $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$, learns a modulator signal $\hat{\boldsymbol{Z}}$, and applies the respective amplitude modulation. This linear time-varying audio effect is easy to interpret. For more complex nonlinear time-varying effects, a more in-depth analysis of the model is required.

For selected linear and nonlinear time-varying tasks, Fig. 5 shows the input, target, and output waveforms together with their respective modulation spectrum. In the time-domain, it is evident that the model is matching the target waveform. From the modulation spectrum it is noticeable that the model introduces different modulation energies into the output which were not present in the input and which closely match those of the respective targets.

The task becomes more challenging when a nonlinearity is added to a linear time-varying transformation. Fig. 5d depicts results for the *phaser-overdrive* task. Given the large overdrive gain the resulting audio has a lower-frequency modulation. It can be seen that the model introduces modulations as low as 0.5 Hz. But the waveform is not as smooth as the target, hence the larger mae values. Although the mae increased, the model does not significantly reduce performance and is able to match the combination of nonlinear and modulation based audio effects.

Much more complicated time-varying tasks, such as the *ring modulator* and *Leslie speaker* virtual analog implementations were also successfully modeled. This represents a significant result, since these implementations include nonlinear modulation; *ring modulator*, or varying delay lines together with artificial reverberation and Doppler effect simulation; and the *Leslie speaker*.

Lastly, the model is also able to perform linear and nonlinear time-invariant modeling. The long temporal dependencies of an envelope driven *auto-wah*, *compressor* and *multiband compressor* are succesfully modeled. Furthermore, in the latter case, the crossover filters are also matched. The *msed* may not be relevant for these effects, but the low *mae* values represent that the model also performs well here.

5. CONCLUSION

In this work, we introduced a general-purpose deep learning architecture for modeling audio effects with long temporal dependencies. Using raw audio and a given time-varying task, we explored the capabilities of end-to-end deep neural networks to learn low-frequency modulations and to process the audio accordingly. The model was able to match linear and nonlinear time-varying audio effects, time-varying virtual analog implementations and time-invariant audio effects with long-term memory.

Other white-box or gray-box modeling methods suitable for these time-varying tasks would require expert knowledge such as specific circuit analysis and discretization techniques. Moreover, these methods can not easily be extended to other time-varying

[5] https://mchijmma.github.io/modeling-time-varying/

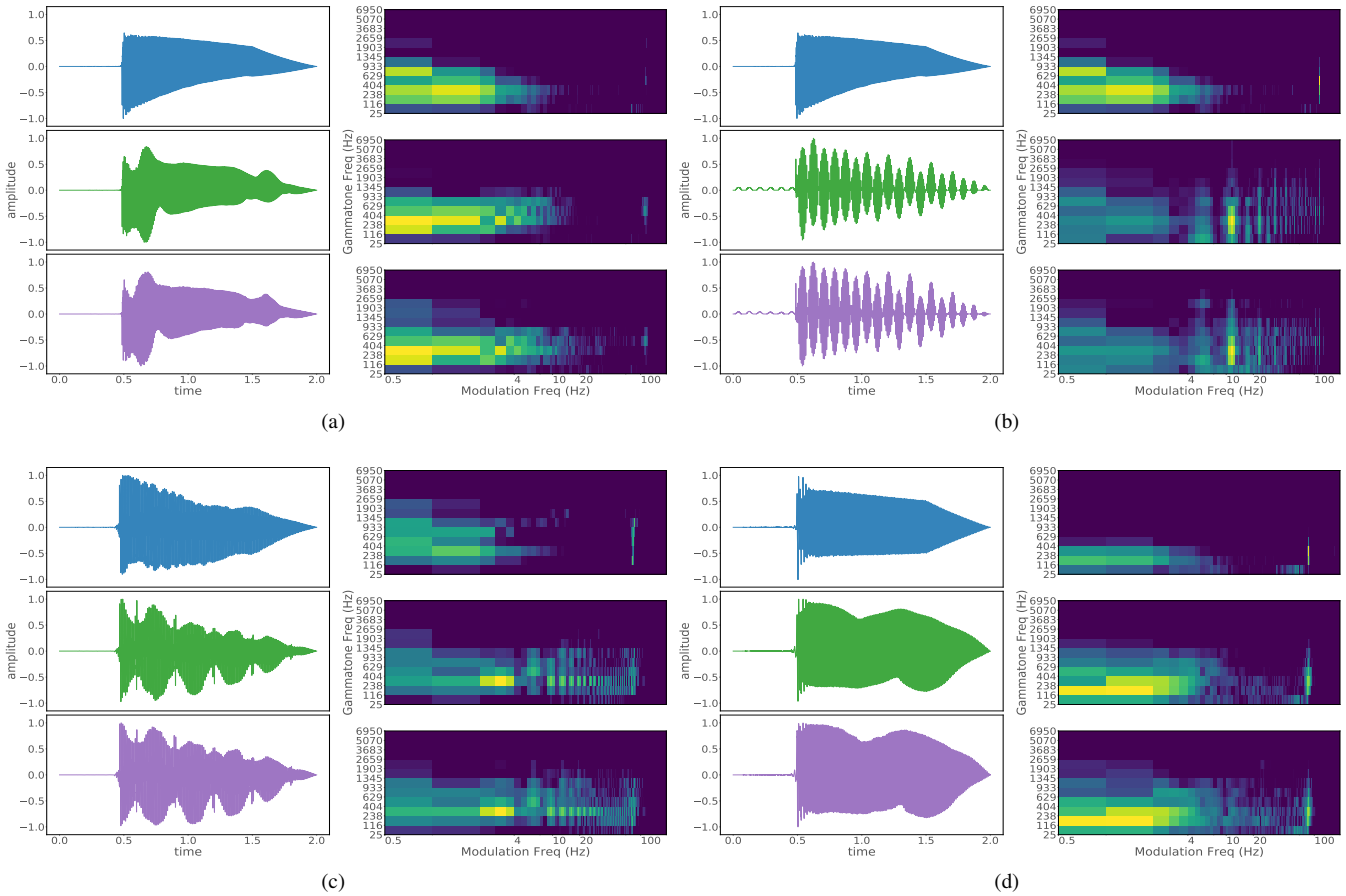


Figure 5: *Results with the test dataset for the following time-varying tasks: 5a)* ***chorus****, 5b)* ***ring modulator****, 5c)* ***Leslie speaker*** *(right channel) and 5d)* ***phaser-overdrive****. For each subfigure and from top to bottom: input, target and output waveforms and respective modulation spectrum plots.*

tasks, and assumptions are often made regarding the nonlinear behavior of certain components. To the best of our knowledge, this work represents the first black-box modeling method for linear and nonlinear, time-varying and time-invariant audio effects. It makes less assumptions about the audio processor target and represents an improvement of the state-of-the-art in audio effects modeling.

Using a small amount of training examples we showed the model matching *chorus*, *flanger*, *phaser*, *tremolo*, *vibrato*, *autowah*, *ring modulator*, *Leslie speaker* and *compressors*. We proposed an objective perceptual metric to measure the performance of the model. The metric is based on the euclidean distance between the frequency bands of interest within the modulation spectrum. We demonstrated that the model processes the input audio by applying different modulations which closely match with those of the time-varying target.

Perceptually, most output waveforms are indistinguishable from their target counterparts, although there are minor discrepancies at the highest frequencies and noise level. This could be improved by using more convolution filters, which means a higher resolution in the filter bank structures [6]. Moreover, as shown in [5], a cost function based on time and frequency can be used to improve this frequency related issue, though listening tests may be required.

The generalization can also be studied more thoroughly, since the model learns to apply the specific transformation to the audio of a specific musical instrument, such as the electric guitar or the bass guitar. In addition, since the model strives to learn long temporal dependencies with shorter input size frames, and also needs past and subsequent frames, more research is needed on how to adapt this architecture to real-time implementations.

Real-time applications would benefit significantly from the exploration of recurrent neural networks to model transformations that involve long-term memory without resorting to large input frame sizes and the need for past and future context frames. Although the model was able to match the artificial reverberation of the *Leslie speaker* implementation, a thorough exploration of reverberation modeling is needed, such as plate, spring or convolution reverberation. In addition, since the model is learning a static representation of the audio effect, ways of devising a parametric model could also be explored. Finally, applications beyond virtual analog can be investigated, for example, in the field of automatic mixing the model could be trained to learn a generalization from mixing practices.

6. ACKNOWLEDGMENTS

The Titan Xp GPU used for this research was donated by the NVIDIA Corporation. EB is supported by a RAEng Research Fellowship (RF/128).

7. REFERENCES

[1] Joshua D Reiss and Andrew McPherson, *Audio effects: theory, implementation and application*, CRC Press, 2014.

[2] Udo Zölzer, *DAFX: digital audio effects*, John Wiley & Sons, 2011.

[3] Julius Orion Smith, *Physical audio signal processing: For virtual musical instruments and audio effects*, W3K Publishing, 2010.

[4] Clifford A Henricksen, "Unearthing the mysteries of the leslie cabinet," *Recording Engineer/Producer Magazine*, 1981.

[5] Marco A. Martínez Ramírez and Joshua D. Reiss, "End-to-end equalization with convolutional neural networks," in *21st International Conference on Digital Audio Effects (DAFx-18)*, 2018.

[6] Marco A. Martínez Ramírez and Joshua D. Reiss, "Modeling of nonlinear audio effects with end-to-end deep neural networks," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2019.

[7] Jyri Pakarinen and David T Yeh, "A review of digital techniques for modeling vacuum-tube guitar amplifiers," *Computer Music Journal*, vol. 33, no. 2, pp. 85–100, 2009.

[8] Dimitrios Giannoulis, Michael Massberg, and Joshua D Reiss, "Digital dynamic range compressor designâa tutorial and analysis," *Journal of the Audio Engineering Society*, vol. 60, no. 6, pp. 399–408, 2012.

[9] David T Yeh, Jonathan S Abel, and Julius O Smith, "Automated physical modeling of nonlinear audio circuits for real-time audio effects part I: Theoretical development," *IEEE transactions on audio, speech, and language processing*, vol. 18, no. 4, pp. 728–737, 2010.

[10] Antti Huovilainen, "Enhanced digital models for analog modulation effects," in *8th International Conference on Digital Audio Effects (DAFx-05)*, 2005.

[11] Colin Raffel and Julius Smith, "Practical modeling of bucket-brigade device circuits," in *13th International Conference on Digital Audio Effects (DAFx-10)*, 2010.

[12] Martin Holters and Julian D Parker, "A combined model for a bucket brigade device and its input and output filters," in *21st International Conference on Digital Audio Effects (DAFx-17)*, 2018.

[13] Jaromír Mačák, "Simulation of analog flanger effect using BBD circuit," in *19th International Conference on Digital Audio Effects (DAFx-16)*, 2016.

[14] David T Yeh, "Automated physical modeling of nonlinear audio circuits for real-time audio effects part II: BJT and vacuum tube examples," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, 2012.

[15] Martin Holters and Udo Zölzer, "Physical modelling of a wah-wah effect pedal as a case study for application of the nodal dk method to circuits with variable parts," in *14th International Conference on Digital Audio Effects (DAFx-11)*, 2011.

[16] Felix Eichas et al., "Physical modeling of the mxr phase 90 guitar effect pedal," in *17th International Conference on Digital Audio Effects (DAFx-14)*, 2014.

[17] Julian Parker, "A simple digital model of the diode-based ring-modulator," in *14th International Conference on Digital Audio Effects (DAFx-11)*, 2011.

[18] Julius Smith et al., "Doppler simulation and the leslie," in *5th International Conference on Digital Audio Effects (DAFx-02)*, 2002.

[19] Jussi Pekonen, Tapani Pihlajamäki, and Vesa Välimäki, "Computationally efficient hammond organ synthesis," in *14th International Conference on Digital Audio Effects (DAFx-11)*, 2011.

[20] Roope Kiiski, Fabián Esqueda, and Vesa Välimäki, "Time-variant gray-box modeling of a phaser pedal," in *19th International Conference on Digital Audio Effects (DAFx-16)*, 2016.

[21] Kurt J Werner, W Ross Dunkel, and François G Germain, "A computational model of the hammond organ vibrato/chorus using wave digital filters," in *19th International Conference on Digital Audio Effects (DAFx-16)*, 2016.

[22] Ólafur Bogason and Kurt James Werner, "Modeling circuits with operational transconductance amplifiers using wave digital filters," in *20th International Conference on Digital Audio Effects (DAFx-17)*, 2017.

[23] Jordi Pons et al., "End-to-end learning for music audio tagging at scale," in *31st Conference on Neural Information Processing Systems (NIPS)*, 2017.

[24] Shrikant Venkataramani, Jonah Casebeer, and Paris Smaragdis, "Adaptive front-ends for end-to-end source separation," in *31st Conference on Neural Information Processing Systems (NIPS)*, 2017.

[25] Daniel Stoller, Sebastian Ewert, and Simon Dixon, "Wave-u-net: A multi-scale neural network for end-to-end audio source separation," in *19th International Society for Music Information Retrieval Conference*, 2018.

[26] Sander Dieleman and Benjamin Schrauwen, "End-to-end learning for music audio," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014.

[27] Soroush Mehri et al., "Samplernn: An unconditional end-to-end neural audio generation model," in *5th International Conference on Learning Representations*. ICLR, 2017.

[28] Jesse Engel et al., "Neural audio synthesis of musical notes with wavenet autoencoders," *34th International Conference on Machine Learning*, 2017.

[29] Merlijn Blaauw and Jordi Bonada, "A neural parametric singing synthesizer," in *Interspeech 2017*.

[30] Xavier Glorot, Antoine Bordes, and Yoshua Bengio, "Deep sparse rectifier neural networks," in *14th International Conference on Artificial Intelligence and Statistics*, 2011.

[31] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, "Speech recognition with deep recurrent neural networks," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013.

[32] Alex Graves and Jürgen Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, 2005.

[33] Le Hou et al., "Convnets with smooth adaptive activation functions for regression," in *Artificial Intelligence and Statistics*, 2017.

[34] Jie Hu, Li Shen, and Gang Sun, "Squeeze-and-excitation networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[35] Taejun Kim, Jongpil Lee, and Juhan Nam, "Sample-level cnn architectures for music auto-tagging using raw waveforms," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2018.

[36] Michael Stein et al., "Automatic detection of audio effects in guitar and bass recordings," in *128th Audio Engineering Society Convention*, 2010.

[37] Somsak Sukittanon, Les E Atlas, and James W Pitton, "Modulation-scale analysis for content identification," *IEEE Transactions on Signal Processing*, vol. 52, 2004.

[38] Josh H McDermott and Eero P Simoncelli, "Sound texture perception via statistics of the auditory periphery: evidence from sound synthesis," *Neuron*, vol. 71, 2011.

[39] Martin McKinney and Jeroen Breebaart, "Features for audio and music classification," in *4th International Society for Music Information Retrieval Conference*, 2003.

VISUALAUDIO-DESIGN – TOWARDS A GRAPHICAL SOUNDDESIGN

Lars Engeln

Chair of MediaDesign
Technische Universität Dresden
Dresden, Germany
lars.engeln@tu-dresden.de

Rainer Groh

Chair of MediaDesign
Technische Universität Dresden
Dresden, Germany
rainer.groh@tu-dresden.de

ABSTRACT

VisualAudio-Design (VAD) is a spectral-node based approach to visually design audio collages and sounds. The spectrogram as a visualization of the frequency-domain can be intuitively manipulated with tools known from image processing. Thereby, a more comprehensible sound design is described to address common abstract interfaces for DSP algorithms that still use direct value inputs, sliders, or knobs. In addition to interaction in the time-domain of audio and conventional analysis and restoration tasks, there are many new possibilities for spectral manipulation of audio material. Here, affine transformations and two-dimensional convolution filters are proposed.

1. INTRODUCTION

After Pierre Schaeffer's first experiments with bouncing records as a representative of the *musique concrète*, the availability of magnetic tapes glued together enabled later composers, such as Karlheinz Stockhausen, John Cage, or Edgard Varèse, to create loops. In the 1970s, spectral music was pioneered as a compositional technique using computer-aided analysis of acoustic music or artificial timbres at IRCAM with the Ensemble *l'Itinéraire* by composers such as Gérard Grisey and Tristan Murail. Murail himself has described spectral music as an aesthetic, not a style - not a set of techniques, but an attitude [1].

Nowadays, spectrograms are frequently used to analyze audio material. After applying audio effects, the spectrogram can illustrate the implications of the manipulation, for instance. However, processing and analysis tools are ordinarily separated. Although, manipulations within the visualization of analysed audio data allow a more comprehensible sound design (cf. [2]).

Popular spectrogram manipulations are time-scaling, transposition, restoration and compression. Apart from that, visual manipulation can also be used for more advanced spectral processing [3]. Especially since small spectral modifications are not perceived as unnatural or synthetic [4].

A spectrogram that is interpreted as a pixel-based representation can be manipulated with image processing to achieve a *Visual-Audio-Design* (VAD). This is an opportunity for a more comprehensible sound design, in contrast to directly editing the parameter of DSP algorithms.

According to Klingbeil [3], the following challenges occur in spectral editing:

- **Time and cost** – The analysis of signals must be computed in the shortest possible time. By efficient implementation, for example the STFT, and an increase of the computational power this goal is reached.
- **Synthesis problem** – Processed signals from the frequency domain must be transformed back for playback.
- **Control problem** – Exciting music is dynamic and consists of many finely tuned frequencies. Processing must therefore be equally finely granular.
- **Compositional problem** – In addition to subsequent editing, there are approaches to compose music directly in the frequency domain.

This creates new demands both on the software and on the composer, who has to deal intuitively with the composition of individual frequencies and their magnitudes.

There are tools and papers concerning generally visual manipulation of the frequency-domain [5, 3] and the timbre design [6]. Our research addresses the Control and Compositional problem. Thus, a workflow for sound *designers* to creatively manipulate and generate sounds is described.

1.1. Coherence of Visual and Auditory Perception

In recent years, studies have shown that the combination of auditory and visual stimuli can change and even improve human perception (see [7, 8]).

In particular, it was shown that multisensory convergence exists in low sensory processing phases [7] and exists for visual-auditive stimuli (see [9, 10]). Convergence does not only occur after extensive processing in unisensory brain regions. This low-level processing of coherent sensor inputs allows the improvement of visual and acoustic perception by simultaneous matching stimuli. Based on the early convergence it can be assumed that acoustic and visual stimuli have a positive effect on each other [11]. For example, multimodal objects were detected faster and more accurately than unimodal objects [12].

Furthermore, the relationship between color and sound was investigated with an empirical approach [13], as existing mappings were often inconsistent and unfounded. The works perform a mapping of tonality, loudness and timbre to hue, saturation and brightness in different constellations. Although, there is a strong correlation between loudness and saturation as well as tonality and brightness (cf. [13]).

2. RELATED WORKS

Spectral editing was often implemented as a kind of painting program (cf. [14]). It started early with *SpecDraw* [2], at which

frequencies were filtered (rejected) with an eraser and rectangular selection and transformation of the frequency-domain. In *AudioSculpt* [5] polygonal and freehand selections provided more freedom for damping, enforcing, and duplicating spectrals, as well as for time-stretching [15]. *TAPESTREA* [16] made it possible for the first time to create sound spaces from different audio sources. *SPEAR* [3] abstracts the frequency-domain and uses a sparse representation of audio. Therefore the manipulation is mediated by a vector visualization (reprasentation of partials with lines) and not by a pixel visualization (sonagram) like the other works. Prehearing is done by STFT, and when exporting files in higher resolutions it is possible to choose between different implementations, such as the McAulay-Quatieri method [17] or oscillator banks. In *MetaSynth* image data is used as input besides the freehand drawing of shapes within the spectrogram [18]. Moreover, filters and transpositions with metaphorical manipulations of the frequency-domain with fluids [19] and with virtual reality [20] are proposed.

All works have a collection of user interface elements and interaction possibilities. Free zoom levels in time and frequency axis allow the user both a quick overview of the entire spectral space and the possibility to edit temporal details. A tool palette usually allows switching between different modes for the selection and modification of magnitudes.

Besides that, with *Sound Mosaics* [21, 22] a system for sound synthesis via influencing graphical variables was created. Thereby, timbre is described by three parameters: sharpness, compactness and roughness (sensory dissonance). Compactness classifies signals on a scale between complex tones and simple noise. The visualization uses color (hue, saturation and brightness), as well as texturial structure. In addition, there is the research field of *soundtextures* (compare [23]), which also describes a graphical synthesis that is sometimes done in the frequency-domain.

Moreover, the *Sonic Visualizer* [24] is a program for analyzing audio signals. The program has an external API and a plugin infrastructure. Thus, for example, psychoacoustic parameters can be calculated and various features like a beat-detection can be extracted. Also, the *ArtemiS SUITE* enables sound and vibration analysis in an industrial context and displays psychoacoustic parameters according to Zwicker and Sottek's hearing model.

3. VISUALAUDIO-DESIGN

Our VisualAudio-Design (VAD) is written in modern C++ and is based on libCinder using FFTW for analysis/resynthesis and openCV for two-dimensional convolutional spectral processing. Therefore, it is cross-platform compileable, but mainly it is developed for Windows.

VAD allows the user to load, analyze and play natural sounds as audio files in a canvas (see Figure 1). Through the analysis (FFT) the overtones (or generally the frequencies involved in the sound) can be visualized.

In addition to natural sounds, images of any type can also be loaded and converted into sound (phase estimation + iFFT). This allows graphical structures to be used as sound material.

The central part is the SpectralCanvas (a sonagram), in which sounds and effects can be freely transformed as nodes in groups and layers.

A sound object can be moved freely in the SpectralCanvas. A horizontal shift in time repositions the sound object and a vertical shift transposes it. Rotation can also be performed. A slight rotation causes a gradual glissando of the entire spectral range of the sound object. With strong rotations (approx. 90°) overtones become transients. Thereby, a sound rich in overtones becomes an impulse-like clicking (see Figure 2). Timestreching can be performed by stretching the node along the time axis. Stretching along the frequency axis contracts or expands the harmonics.

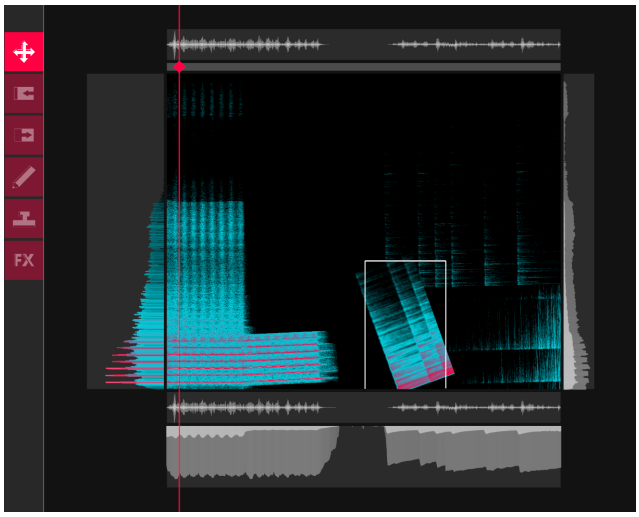


Figure 1: *Workspace of the VisualAudio-Design: the SpectralCanvas with its nodes consisting of sounds and effects (center), surrounded by widgets, and a toolbar (left).*

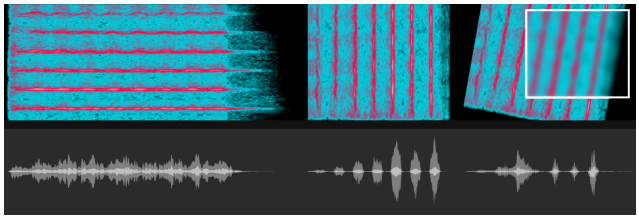


Figure 2: *Sound-nodes can be freely transformed. Thereby, a flute (left), is rotated by 90° (middle) or less (right). Effect-nodes can alter a certain area (see white rectangle (right) with blurring).*

In addition, effects equivalent to image processing can be used to distort the frequency domain to further alter the material. In this way, for example, a blur filter can be used to optically smooth the spectral space and thus audibly noise out the sound. The advantage over conventional audio effects over time is that individual frequency bands or individual overtones can be specifically assigned to an effect.

In this way, the VAD is suitable for creating sound surfaces as collages for creative use. A composer or sound designer is not limited to the mere temporal arrangement of sounds. New ways for musical expression can be found through transposition, rotation, overtone-exact effects, the use of graphical structures, and also by cutting & re-inserting individual frequency ranges/overtones.

3.1. System Design

The system is based on a scenegraph (the SpectralCanvas) of nodes and is highly modularized for extending with new features. At the same time, it is highly parallelized in order to guarantee a smooth

UserExperience with fast calculations in background worker. Basically, there are modules providing tools for the user to create or modify nodes in different ways, and widgets that can visualize additional parameters besides to the sonagram of the SpectralCanvas.

A tool is created as a module (inherits from ModuleBase-class) and is registered at the ModuleManager. This also creates the entry in the GUIManager for the toolbox. The active tool receives all interaction events that are performed with it on the SpectralCanvas. The tool also has direct access to the nodes of the SpectralCanvas and its spectral data. Thus a tool directly manipulates the nodes. A node can own spectral data itself (sound-node) or manipulate underlying spectral data in the hierarchy as effect-node. If a node has been changed, the corresponding layer and all layers above it are updated by the SpectralCanvas and are processed by the SpectralEngine. The SpectralEngine always keeps the time and frequency domain consistent with each other.

Multimodal input (mouse, key and touch at the moment) is preprocessed. Afterwards, the input event is recognized in the InteractionManager as gesture and propagated as interaction event. When starting an interaction, the GUI is handled first. If the GUI is not involved, the event is passed on to the WidgetManager. In this very moment the SpectralCanvas is treated as Widget. If the current interaction takes place on the SpectralCanvas, the event is passed to the active module. However, when interacting on a actual widget, the event is processed in the corresponding widget. If no one handles the event, so the interaction is done somewhere else in the workspace, then standard interactions for zooming and panning to navigate in the workspace are done.

3.1.1. Tools and Interaction

Each module provides a tool for the user, to manipulate the nodes or directly the frequency-domain of the SpectralCanvas. Several modules have been implemented but many more suitable modules are possible. Description of touch gestures according to *GeForMT* [25] are shortend in the following. For instance, a two finger swipe gesture is shortened to $2F_{swipe}$ and a gesture in which one finger holds while another finger performs a circular movement is shortened to $1F_{hold} + 1F_{circle}$.

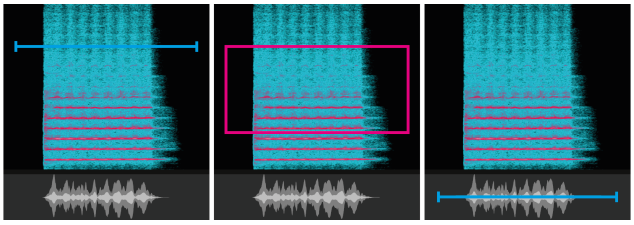


Figure 3: *SaveModule: selecting a time range (left) or an area of spectrals (middle) in the SpectralCanvas, or selecting a time range in the waveform widget aswell (right).*

- **load & save** – Loading of any audio and image files and simultaneous positioning by clicking ($1F_{tap}$) into SpectralCanvas or time-based widgets. The sound is always aligned to the 0Hz line. A time range for saving can be selected as a line in the SpectralCanvas or in the waveform widget. To export the frequency domain, an area above a certain threshold is drawn instead of a line. (see Figure 3)

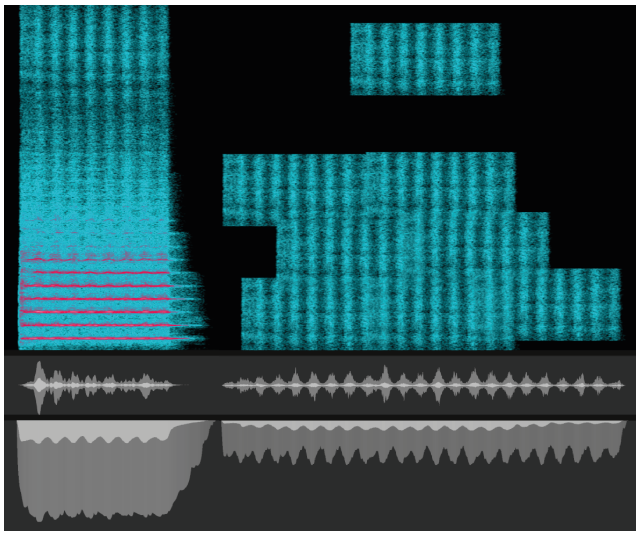


Figure 4: *StampModule: Selecting spectrals (left) and stamping it as new sound-nodes to the SpectralCanvas (right).*

- **transform** – Translation via dragging ($1F_{move}$), rotation & scaling via right-click drag ($1F_{hold} + 1F_{circle}$ & $1F_{hold} + 1F_{split}$), warping via manipulation handles.
- **stamp** – With the stamp-tool an area of spectrals is copied and stamped to a new location within the SpectralCanvas (see Figure 4). Soundtextures can be created.
- **draw** – Lines can be drawn representing sinusoids or transients. Drawing multiple lines via multitouch ($[1F_{move}]^*$) is possible. Therefore, navigational tasks with $3F_{move/sparse}$ are ignored. This module has high potential for drawing fundamental tones with instantaneously generated overtones to create lines with *natural* and *artificial* timbre.
- **effect** – Effect-nodes are treated like sound-nodes. They are created via normal selection.
- **loop** – VAD offers the possibility to create loops. The loops can cover the entire frequency and time range or, like the effects, individual frequency ranges. Individual loops can be started and stopped. In this way, a myriad of spectral synchronous and asynchronous loops can be played.
- **navigational tasks** – Panning via drag on workspace ($1F_{move}$ on workspace, or $3F_{move}$ anywhere) and zooming via scroll ($2F_{split/pinch}$ on workspace, or $3F_{sparse}$ anywhere).

3.1.2. Widgets

The widgets surround the SpectralCanvas (see Figure 5). This allows widgets to have one of the four orientations *(TOP, RIGHT, BOTTOM, LEFT)*. LEFT & RIGHT are frequency-based widgets and TOP & BOTTOM are time-based widgets. The same type of widgets can be arranged in arbitrary ways. In the following table each row can be modelled as a adaptive widget (see Tabel 1). For instance, if the waveform widget is dragged to a LEFT or RIGHT position it transforms into a spectrum. Advanced widgets for displaying and manipulating perceptual parameters such as sharpness, compactness, etc. are planed as future work.

time	frequency
timeline, time axis labeling	frequency axis labeling
waveform	spectrum
spectral power	long-term spectrum
current parameters	parameter distribution

Table 1: Relation of time based and frequency based widgets.

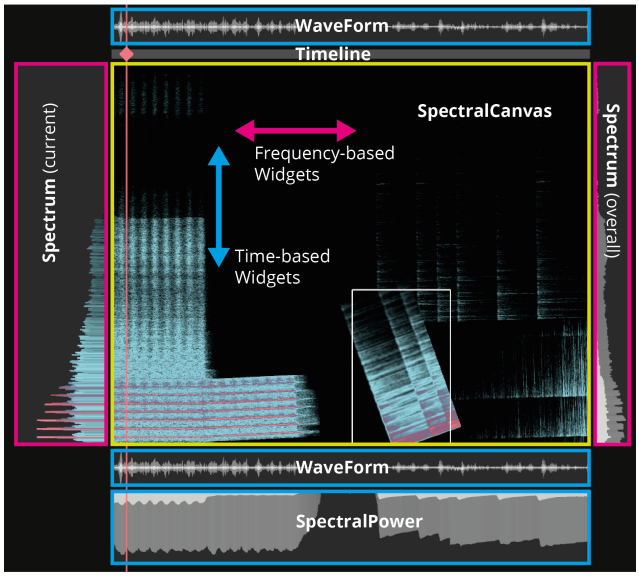


Figure 5: *Widgets are aligned horizontally or vertically to the SpectralCanvas.*

Widgets can bind themselves to events for getting the current play position, current spectrum, the whole frequency-domain, the whole time-domain, and the nodes of the SpectralCanvas. Each event will occur, if the corresponding data has been changed. At the same time, the data in the widgets can also be edited. An example is an equalizer in the spectrum.

The widget for spectral power and long-term spectrum are projections of the frequency-domain on the corresponding axis (see figure 5) and visualizes the minimum, maximum and mean values (min, max, avg).

3.1.3. Spectral Effects

A effect-node receives the underlaying spectral data as an event. Now, convolution kernels are applied. The easiest way is to just apply a kernel for blurring and sharpening, but conditional filter like a gain-threshold (Equation 1) can also be applied (Figure 6).

5x5 filter kernel for gaussian blur:

$$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

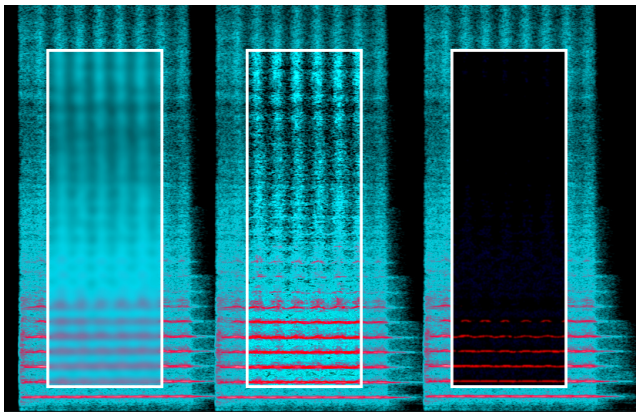


Figure 6: *Examples of effect-nodes: blurring (left), sharpening (unsharp masking) (middle) and gain-threshold (right).*

5x5 filter kernel for unsharp masking:

$$\frac{-1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & -476 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

$$f_{threshold}(spectral) = \begin{cases} spectral, & \text{if } spectral \geq threshold \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

The unsharp masking is thereby derived from the gaussian blur and amplifies the high-frequency components of the neighbouring magnitudes in the two-dimensional space. The gaussian blur creates a more noisy and distant sound. On the other hand, the unsharp masking creates a rough and more direct sound. These kernels are structure-preserving, because harmonics and partials are not displaced. A short table shows the equivalent of time-domain effects for image processing frequency-domain effects (see Table 2). With this transfer, most conventional time-domain effects can be used directly in the VisualAudio-Design in a comprehensible way. Of particular interest are effects from the time-domain and image processing for which no transfer can be found. These are unique features.

audio	image
gate&limiter	tonal correction, black&white reassign
overdrive	tonal correction, white shift
compressor&expander	exposure lights&shadows
simple reverb	directional blur

Table 2: Relation of ordinary audio and image effects.

Additionally, some novel spectral effects are non-structure-preserving effects, like inflate and contract (see Figure 7). Those non-structure-preserving effects deform harmonics and transients, meaning that timbre and envelope are altered.

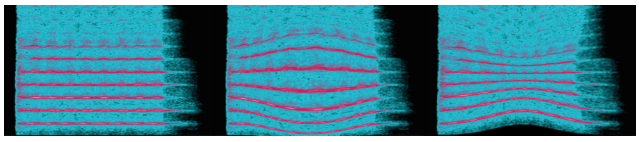


Figure 7: *Examples of non-structure-preserving/structure-invasive effects: none (left), inflating (middle) and contracting (right).*

3.2. Processing

The SpectralEngine has four worker threads for forward transformation, backward transformation, windowed time-data cumulation, and phase-estimation. The SpectralCanvas (scenegraph) has an instance of the SpectralEngine to process the overall results. In addition, every layer and every node have their own SpectralEngine. That offers the opportunity to instantly prehear only the node or layer soloistically. So there are at least $4 * (1 + \sum node + \sum layer)$ threads running, but most of them are idled.

To prehear a layer or the entire project, all spectral data of the nodes are merged. At first, nodes are merged to their layer and then the layers are merged at the same way to one consistent data layer. Merging in fact of the spectral data means, that the maximum value of all magnitudes and the average value of all phases are taken to reduce multiple spectral datasets to one.

To speed up processing, only the changed area of the corresponding and higher layers are recalculated. By transforming a node the changed area has to be processed. The whole changed area consists of the part where the node was before and the part where the node was moved to (see Figure 8). Because of overlapping windows, the influenced time range is *windowSize* many samples larger (Equation 4) than just the time position of the changed spectrums. Especially if an effect-node on top of a changed node is not fully covered in the changed Area ($A_{changed}$), then the half kernel size (see Equation 3) has to be added as well. In fact of multiple layered effect-nodes, for each effect Equation 3 has to be repeated. Only this range will be cumulated to the actual part of the audio buffer. For a smooth prehearing, the playback is double-buffered. While playing buffer A, new data is inserted to buffer B and after that the buffers are swapped.

$$A_{changed} = \sum spectrum_{changed} \tag{2}$$

if effect-node is not totally included in $A_{changed}$, repeat:

$$A_{changed} \mathrel{+}= \begin{cases} \|kernel\|, & \text{if effect is overlapping both sides} \\ \frac{\|kernel\|}{2}, & \text{if effect is overlapping one side} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where: $\|kernel\|$ = size of kernel

$$T_{influenced} = windowSize + hopSize * A_{changed} \tag{4}$$

3.2.1. Phase-estimation

The SpectralEngine decides depending on the strength of the manipulation, whether the phase values can be used further or whether they have to be re-estimated. The phase estimation is iteratively approximated with Griffen&Lim's Algorithm (GLA) [26] on its own thread in the background. After 5-10 iterations the root-mean-square error (RMSE) is already $\lessapprox 0.1$. Therefore,t he data is propagated to the rest of the system for the first time, so that the widgets

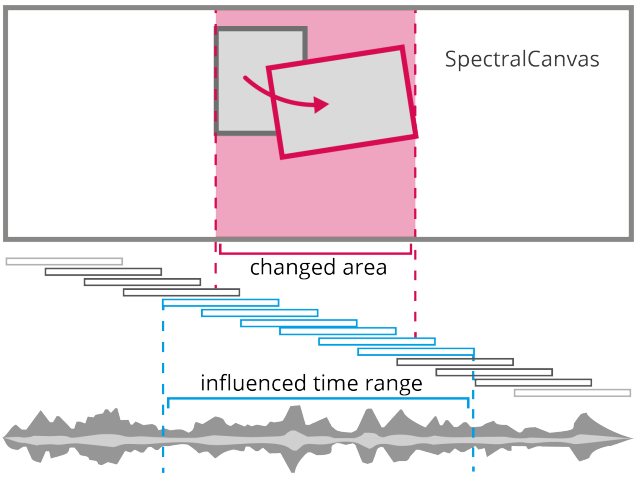


Figure 8: *Illustration of the changed area consisting of the old position (grey) and the new position (red) of a node, which is influencing a certain time range (blue).*

are updated and a first prehearing is possible. In the background, the phases are approximated more and more, so that after 100 iterations (RMSE of some signals can already be $\lessapprox 0.05$) a further propangation takes place. Depending on the desired quality, further iterations are performed.

3.3. Challenges for Visualizations

Objective physical measurements are the basis for analysis, however, psychoacoustics deals with the subjective perception of signals (cf. [27]). Thereby, a *non-linearity* and a *non-orthogonality* occur. Sensory quantities often do not behave linearly to their corresponding acoustic quantities. Moreover, many sensory variables influence each other. For example, both frequency and volume have an influence on the perceived pitch.

Also, visual perception is non-linear and non-orthogonal. For the perception of the areas and volumes, the actual area or volume is underestimated [28, 29], although, for instance, the perception of lengths is directly proportional to the actual length of a line.

Spectrogram visualizations have a high dynamic range. Due to the visualization of a lot of sound nodes, there is a risk that the essential properties of a spectrogram are difficult to capture. Visual recognition is also complicated by noise. In the context of visual or image-based audio processing, however, a fast and simple inspection of sounds spectral space is necessary. One possibility for a improved visual inspection of a spectrogram is an abstraction of the frequency-domain. At the same time, the basic visual properties of the spectrogram, such as its dimensions, transients and harmonics, should be preserved.

When visualizing information, visual differences should be as subtle as possible, while being effective and meaningful [30]. Contrasts are defined as minimal and distinct. The number of distinctions can be increased by minimizing their differences. Using lower contrasts and differences, a potential visual disorder is reduced.

One approach is a sparse vectorized visualization of the overtones as in [31]. Another is the method presented here to extract layers of same decibels and visualize them in analogy to maps (see Figure 9). To achieve that, the magnitudes are quantized using

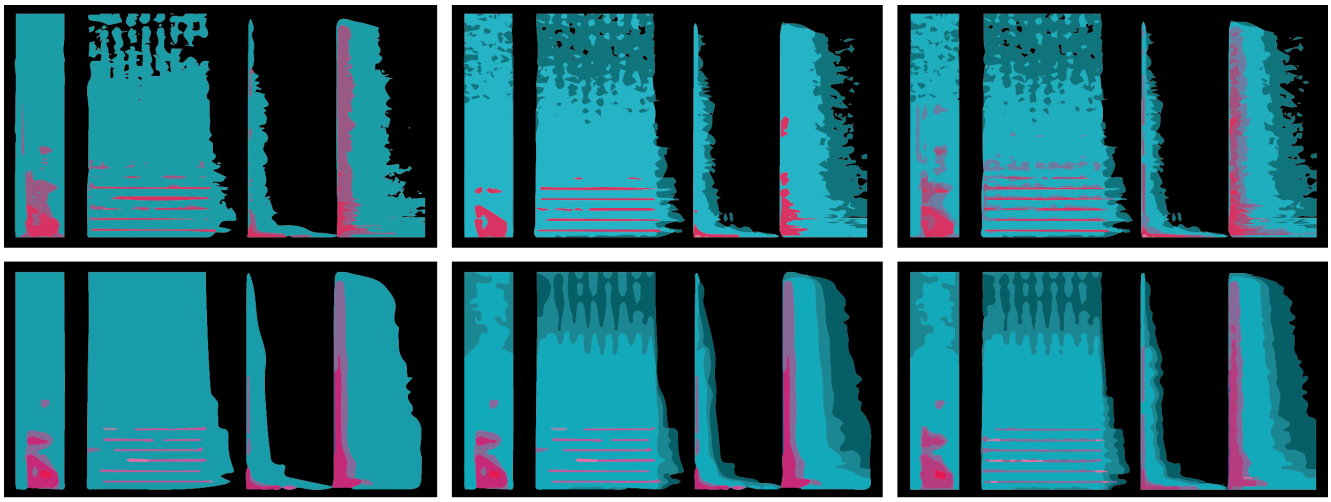


Figure 9: *Abstraction of the frequency-domain with different decibel thresholds: 3 threshold layer (left) and up to 6 (right), with no blurring (upper) and with blurring before thresholding (lower).*

$kmeans$. Here, the x largest clusters of similar magnitudes are determined. However, this means that each sound-node will have layers on different dB levels. Whereby these planes characterize the respective node particularly well. For an improved comparability of sound-nodes a $threshold$ is performed at specific dB level. For this, user-controlled exact levels can be generated. However, important information can be lost during visualization. Therefore, a combination is proposed. The $kmeans$ show the main clusters of a node and via $threshold$ uniform user controlled planes are shown over all nodes. For a smoother result where the plane edges are less noisy a prior blurring is done.

No more than about 20 layers with different color gradations should be used, as otherwise the distinguishability between the color gradations is made more difficult [32]. For the differentiation of two areas the indication of a line is sufficient, if needed at all. The line should have a hue which is in the color scale [32]. The distinguishable brightness values of colors depend on the corresponding hue.

3.4. User-centered Design

VAD highly focuses on UserExperience and ease-of-use. That is why it is designed and developed in a user-centered way. No formal study is carried out, yet. However, qualitative evaluations (expert interviews) with professionals and students from the fields of human-computerinteraction, sound design and music composition are conducted continuously in short design loop intervals.

3.4.1. User study

A brief user study was conducted. 26 subjects (20 male, 6 female) with average age of 26.12 participated. The subjects tested the VisualAudio-Design (VAD) with mouse only and had to perform transpositions of sound-nodes, rotations (arbitrary angles, but always including 90°), and applying effect-nodes (blur, sharpen, threshold) (compare Section 3.1.3). After performing these explicit tasks, they had time to create sound collages on their own.

Then they were asked with a 5-Likert scale whether VAD is creative, the resulting resynthesis is predictable, the interface and visualization is clear (or confusing) and whether it is easy to use. In addition, the user experience was tested with the *short user experience questionaire* (UEQ-S) [33] (with a 7-Likert scale).

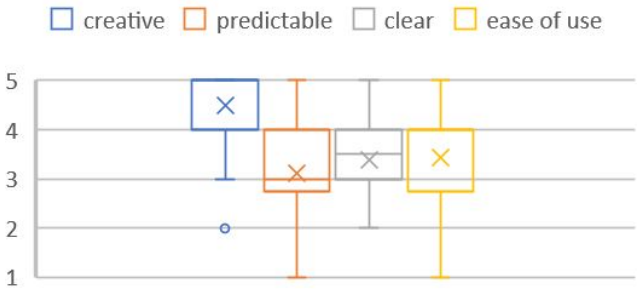


Figure 10: *Results of a brief user study with a 5-Likert scale (the higher the better): uncreative (1) – creative (5), unpredictable – predictable, confusing – clear, complicated – easy to use*

The results are indicating that VAD is highly creative (see Figure 10), while the predictability is moderate. The clarity and the ease of use of VAD is given. The UEQ-S is reflecting these indications. The user experience (hedonic quality) is more than average and the usability (pragmatic quality) is more than moderate (see Figure 11). This is a good starting point, but the interface to effectivly use the node-based VAD has to be improved in future work.

4. FUTURE AND ON-GOING WORK

Users of the qualitative evaluations wanted more intiutive touch gestures than the existing ones in order to operate VAD without GUI as mode switch. To achieve this, a complete GeForMT interpreter has to be implemented. Also gaze and pen should be included in the interaction.

At the moment it is only possible to select magnitudes via a rectangle selection. For this, freehand and polygon selections should be added, as stated in the related works, which follow content based features like harmonic overtones.

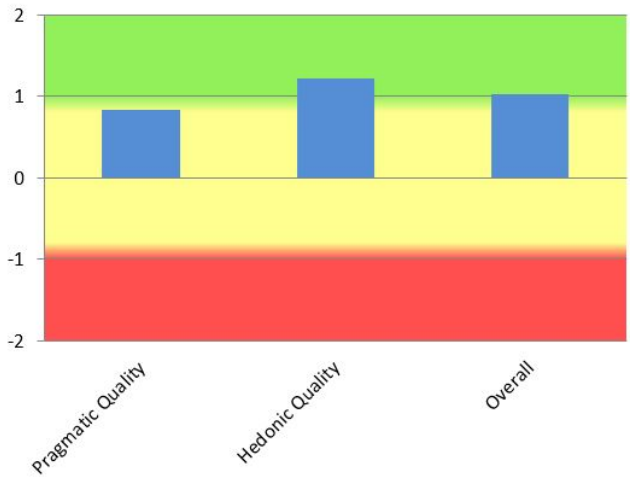


Figure 11: *Results of the UEQ-S test.*

The next stage of development will be formal studies. In particular, the coherence and influence of the visual on the auditory perception will be investigated for achieving a non-invasive *sound*-design.

The draw module has to be completely redesigned in order to paint meaningful timbre directly. In this sense also machine learning supported morphing between timbres should follow.

In addtion, great potential also has the merging of the nodes timbre. In this way, combined sounds are created by aligning the maximum harmonic components and/or the maximum spectral power (transient component) (see Figure 12).

5. CONCLUSIONS

In this paper the VisualAudio-Design was introduced, which is intended for the creative use for sound design and composition. The spectral manipulation is based on nodes within a scenegraph (SpectralCanvas) and its SpectralEngine, a highly multi-threaded processing kernel. In this context, basic tools for manipulating nodes and specialised effect-nodes for convolutions were presented. Additionally, widgets are used for analysis, advanced selection and interaction. As well as an abstraction of the frequency-domain was discussed.

6. ACKNOWLEDGMENTS

Thanks to all students who took part in practical and theoretical courses around VAD, testing it, or served as subjects, and thus contributed to the overall development.

7. REFERENCES

[1] Joshua Fineberg, "Spectral music," *Contemporary Music Review*, vol. 19, no. 2, pp. 1–5, 2000.

[2] G Eckel, "Manipulation of Sound Signals Based on Graphical Representation-A Musical Point of View," in *Proceedings of the International Workshop on Models and Representations of Musical Signals, Capri, Italia*, 1992.

[3] Michael Klingbeil, "Software for spectral Analysis, Editing, and synthesis.," in *International Computer Music Conference (ICMC)*, Barcelona, Spain, 2005.

[4] John M Grey and John W Gordon, "Perceptual effects of spectral modifications on musical timbres," vol. 63, no. 5, pp. 1493–1500.

[5] Niels Bogaards, Axel Roebel, and Xavier Rodet, "Sound analysis and processing with audiosculpt 2," *International Computer Music Conference (ICMC)*, p. 1, Nov 2004.

[6] Allan Seago, "A new interaction strategy for musical timbre design," in *Music and human-computer interaction*, pp. 153–169. Springer, 2013.

[7] Shams Watkins, Ladan Shams, Sachiyo Tanaka, J-D Haynes, and Geraint Rees, "Sound alters activity in human v1 in association with illusory visual perception," *Neuroimage*, vol. 31, no. 3, pp. 1247–1256, 2006.

[8] Gemma A Calvert, Peter C Hansen, Susan D Iversen, and Michael J Brammer, "Detection of audio-visual integration sites in humans by application of electrophysiological criteria to the bold effect," *Neuroimage*, vol. 14, no. 2, pp. 427–438, 2001.

[9] Charles E Schroeder and John J Foxe, "The timing and laminar profile of converging inputs to multisensory areas of the macaque neocortex," *Cognitive Brain Research*, vol. 14, no. 1, pp. 187–198, 2002.

[10] Francesca Frassinetti, Nadia Bolognini, and Elisabetta Làdavas, "Enhancement of visual perception by crossmodal visuo-auditory interaction," *Experimental brain research*, vol. 147, no. 3, pp. 332–343, 2002.

[11] Charles E Schroeder and John Foxe, "Multisensory contributions to low-level, unisensory processing," *Current opinion in neurobiology*, vol. 15, no. 4, pp. 454–458, 2005.

[12] Marie-Helene Giard and Franck Peronnet, "Auditory-visual integration during multimodal object recognition in humans: a behavioral and electrophysiological study," *Journal of cognitive neuroscience*, vol. 11, no. 5, pp. 473–490, 1999.

[13] Kostas Giannakis and Matt Smith, "Imaging soundscapes: Identifying cognitive associations between auditory and visual dimensions," *Musical Imagery*, pp. 161–179, 2001.

[14] Jean-Baptiste Thiebaut, Patrick GT Healey, and Nick Bryan-Kinns, "Drawing electroacoustic music.," in *ICMC*, 2008.

[15] Niels Bogaards and Axel Röbel, "An Interface for Analysis-Driven Sound Processing," in *Audio Engineering Society Convention 119*, 2005.

[16] Ananya Misra, Perry R. Cook, and Ge Wang, "Tapestrea: Sound scene modeling by example," in *ACM SIGGRAPH 2006 Sketches*, New York, NY, USA, 2006, SIGGRAPH '06, ACM.

[17] T Quatieri and Rl McAulay, "Speech transformations based on a sinusoidal representation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 6, pp. 1449–1464, 1986.

[18] E Wenger and E Spiegel, "Metasynth 4.0 user guide and reference," *Redwood City, CA: U&I Software LLC. www. uisoftware. com/MetaSynth*, 2005.

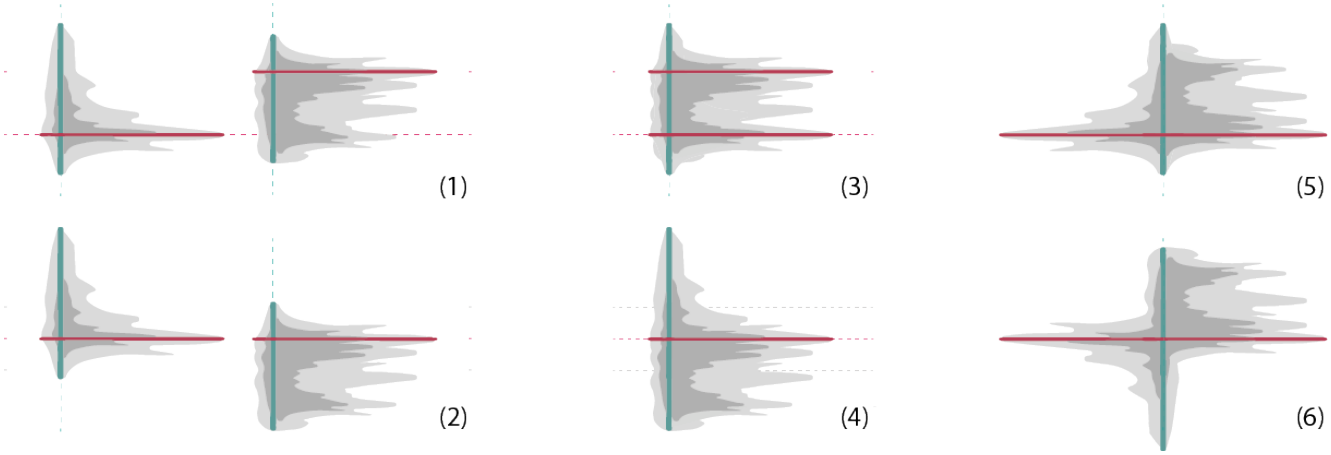


Figure 12: *Alignment of sound-nodes: (1) original position (no alignment), (2) harmonic alignment, (3) temporal alignment, (4) harmonic and temporal, and more complex harmonic & temporal alignments with mirroring of the first node: (5) vertical, (6) vertical and horizontal.*

[19] Lars Engeln and Rainer Groh, "AudioFlux : A Proposal for interactive Visual Audio Manipulation," in *Mensch und Computer 2017 - Workshopband*, M. Burghardt, R. Wimmer, C. Wolff, and C. Womser-Hacker, Eds., Regensburg, Germany, 2017, number September, Gesellschaft für Informatik e.V.

[20] Lars Engeln, Natalie Hube, and Rainer Groh, "Immersive visualaudiodesign: Spectral editing in vr," in *Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion*. ACM, 2018, p. 38.

[21] Konstantinos Giannakis, "Sound mosaics a graphical user interface for sound synthesis based on auditory-visual associations," 2001.

[22] Kostas Giannakis, "A comparative evaluation of auditory-visual mappings for sound visualisation," *Organised Sound*, vol. 11, no. 3, pp. 297–307, 2006.

[23] Diemo Schwarz, "State of the Art in Sound Texture Synthesis," in *Digital Audio Effects (DAFx)*, Paris, France, Sept. 2011, pp. 221–232.

[24] Chris Cannam, Christian Landone, and Mark Sandler, "Sonic visualiser: An open source application for viewing, analysing, and annotating music audio files," in *Proceedings of the 18th ACM international conference on Multimedia*. ACM, 2010, pp. 1467–1468.

[25] Dietrich Kammer, Jan Wojdziak, Mandy Keck, Rainer Groh, and Severin Taranko, "Towards a formalization of multi-touch gestures," in *ACM International Conference on Interactive Tabletops and Surfaces*. ACM, 2010, pp. 49–58.

[26] Nathanaël Perraudin, Peter Balazs, and Peter L Søndergaard, "A fast griffin-lim algorithm," in *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE, 2013, pp. 1–4.

[27] Gareth Loy, *Musimathics: the mathematical foundations of music*, vol. 1, MIT press, 2011.

[28] Stanley S Stevens and Eugene H Galanter, "Ratio scales and category scales for a dozen perceptual continua.," *Journal of experimental psychology*, vol. 54, no. 6, pp. 377, 1957.

[29] James John Flannery, "The relative effectiveness of some common graduated point symbols in the presentation of quantitative data," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 8, no. 2, pp. 96–109, 1971.

[30] Edward R Tufte, Susan R McKay, Wolfgang Christian, and James R Matey, "Visual explanations: images and quantities, evidence and narrative," 1998.

[31] Michael Klingbeil, "SPEAR: Sinusoidal Partial Editing Analysis and Resynthesis," 2012, vol. 12, p. 3.

[32] Edward R Tufte, Nora Hillman Goeler, and Richard Benson, *Envisioning information*, vol. 126, Graphics press Cheshire, CT, 1990.

[33] Martin Schrepp, Andreas Hinderks, and Jörg Thomaschewski, "Design and evaluation of a short version of the user experience questionnaire (ueq-s).," .

SYNTHETIC TRANSAURAL AUDIO RENDERING (STAR): A PERCEPTIVE APPROACH FOR SOUND SPATIALIZATION

Eric Méaux

L3i
University of La Rochelle, France
eric.meaux01@univ-lr.fr

Sylvain Marchand

L3i
University of La Rochelle, France
sylvain.marchand@univ-lr.fr

ABSTRACT

The principles of Synthetic Transaural Audio Rendering (STAR) were first introduced at DAFx-06. This is a perceptive approach for sound spatialization, whereas state-of-the-art methods are rather physical. With our STAR method, we focus neither on the wave field (such as HOA) nor on the sound wave (such as VBAP), but rather on the acoustic paths traveled by the sound to the listener ears. The STAR method consists in canceling the cross-talk signals between two loudspeakers and the ears of the listener (in a transaural way), with acoustic paths not measured but computed by some model (thus synthetic). Our model is based on perceptive cues, used by the human auditory system for sound localization. The aim is to give the listener the sensation of the position of each source, and not to reconstruct the corresponding acoustic wave or field. This should work with various loudspeaker configurations, with a large sweet spot, since the model is neither specialized for a specific configuration nor individualized for a specific listener. Experimental tests have been conducted in 2015 and 2019 with different rooms and audiences, for still, moving, and polyphonic musical sounds. It turns out that the proposed method is competitive with the state-of-the-art ones. However, this is a work in progress and further work is needed to improve the quality.

1. INTRODUCTION

The purpose of sound spatialization (or "3D sound") [1] is to give the listener the sensation that the sound is coming from a certain position in space, or that he/she is surrounded by sounds, etc.

This research area is not new, and several state-of-the-art methods have been proposed, such as Vector Base Amplitude Panning (VBAP) proposed by Pulkki [2], Ambisonics proposed by Gerzon [3] and generalized to higher orders (High Order Ambisonics or HOA) by Daniel [4], or Wave Field Synthesis (WFS) introduced by Berkhout [5]. However, all these methods are based on physics, and tend to reproduce either the sound wave (VBAP) or the sound field at the position of the listener (HOA) or everywhere in space (WFS). Moreover, WFS requires a lot of calibrated loudspeakers, HOA requires also calibration, less speakers but with a sound reproduction localized to a sweet spot. Since we want a system that can be used in practical situations, with a limited number of loudspeakers, in a room where the acoustics cannot be optimized, and computationally efficient, the only option seems to be VBAP, which appears to be a good choice in practice for the musical situations we are interested in [6].

However, together with Mouba we proposed in [7, 8] the principles of a perceptive approach for sound spatialization in practical conditions. We focus neither on the wave field (such as HOA) nor on the sound wave (such as VBAP), but rather on the acoustic paths traveled by the sound to the listener ears. Our method consists in canceling the cross-talk signals between two loudspeakers and the ears of the listener (in a transaural way), with acoustic paths not measured but computed by some model (thus synthetic). Like MPEG Surround [9], our model is based on perceptive cues, used by the human auditory system for sound localization. The aim is to give the listener the sensation of the position of each source, and not to reconstruct the corresponding acoustic wave of field. This should work with various loudspeaker configurations, with a large sweet spot, since the model is neither specialized for a specific configuration nor individualized for a specific listener.

The original method suffered instabilities for some configurations, as well as for extreme frequencies. Since then, the method has been enhanced and we propose now the Synthetic Transaural Audio Rendering (STAR) method.

Experimental tests have been conducted in 2015 and 2019 with different rooms and audiences, for still, moving, and polyphonic sources. The HOA, VBAP, and STAR methods have been compared.

The remainder of this paper is organized as follows. Section 2 introduces the STAR method, Section 3 describes the practical experiments used to evaluate the method, then Section 4 presents the results of experiments conducted in 2015 and 2019, before a conclusion and some perspectives in Section 5.

2. THE STAR METHOD

The principles of Synthetic Transaural Audio Rendering (STAR) we detail here were first introduced in [7, 8].

This method is suitable for spatial audio objects. Each object (or source) consists of a signal to be played at a given position. For now, we focus only on the azimuth θ in the horizontal plane.

For short, the STAR method consists in canceling the cross-talk signals between two loudspeakers and the ears of the listener (in a transaural way), with acoustic paths not measured but computed by some model (thus synthetic). Our model is based on perceptive cues, used by the human auditory system for sound localization. The aim is to give the listener the sensation of the position of each source, and not to reconstruct the corresponding acoustic wave of field. This is indeed a perceptive approach.

In a setup with many speakers such as the one illustrated on Figure 1, we use the classic pair-wise paradigm [10], consisting in choosing for a given source only the two speakers closest to it (in azimuth): one at the left of the source, the other at its right. This is the same choice as in the VBAP method in this two-dimensional

case. Of course, when the source is exactly on one speaker, the source signal is directly played from this speaker and thus the spatialization process is bypassed.

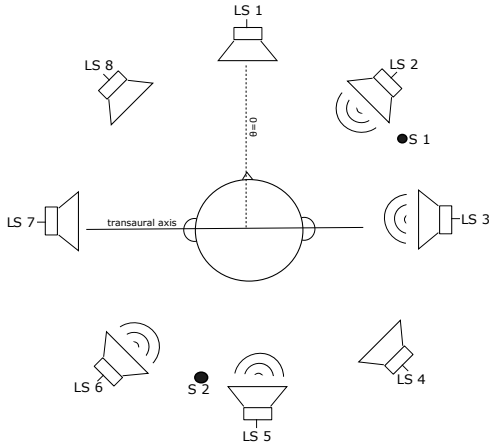


Figure 1: *Octophonic setup, thus with eight loudspeakers (LS), and with two sound sources (S). S1 is located between LS 2 and LS 3, and S2 is between LS 5 and LS 6.*

STAR operates in the spectral domain. Each source signal is passed into the frequency domain with a Short-Time Fourier Transform (STFT), using the Fast Fourier Transform (FFT), processed and distributed among the loudspeakers, then the signal for each loudspeaker is obtained from the spectral domain using the inverse FFT, see Figure 2. Thus, for n sources and m loudspeakers, we compute $n + m$ FFTs in total (i.e. 10 FFTs in the case illustrated in Figure 1). In practice, with use a Hann window and frames of 1024 samples at 44100Hz, with 50% overlap.

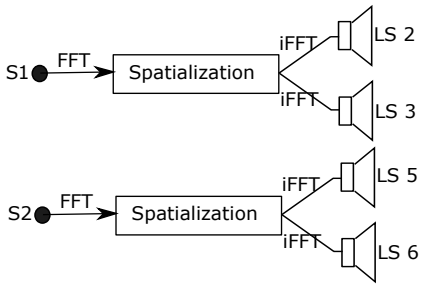


Figure 2: *General principle of STAR spatialization.*

Since low frequencies can cause problems with windowing in the STFT, such as clicks in the case of moving sources (thus changing parameters) when there are not enough periods of the signal within the window w, we filter out the frequencies below 150Hz prior to the spatialization and re-inject them equally in all the loudspeakers afterwards (we could even use a subwoofer, although we preferred not to use this possibility in our experiments). This is not problematic since human beings hardly localize such low frequencies (see [1]).

2.1. Synthetic Paths

With the STAR method, we consider the acoustic paths traveled by the sound to the listener ears. These paths are represented in the spectral domain by their transfer functions, and derived from interaural cues using a model.

The Interaural Time Difference (ITD) corresponds to the travel time difference of a sound between the two ears, while the Interaural Level Difference (ILD) corresponds to the level difference between the two ears.

These interaural cues can be derived from real Head-Related Transfer Functions (HRTFs), which are the spectral versions of the Head-Related Impulse Responses (HRIRs) that can be found, for example, in the CIPIC database [11]. More precisely, we have:

$$\begin{aligned} \mathrm{ILD}_{\mathrm{real}}(f) &= -20\log_{10}(|\mathrm{HRTF}_L(f)/\mathrm{HRTF}_R(f)|) \\ \mathrm{ITD}_{\mathrm{real}}(f) &= -\angle(\mathrm{HRTF}_L(f)/\mathrm{HRTF}_R(f))/(2\pi f) \end{aligned} \quad (1)$$

The HRTFs depend on the subject. Since we are considering only the azimuth, individualization is not really necessary and we could consider average HRTFs among subjects (see Figure 3). However, after Viste [12], in our system we use a model for each

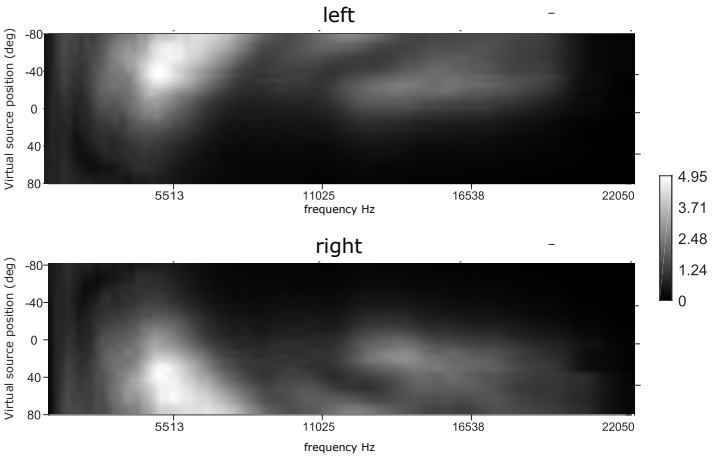


Figure 3: *HRTF magnitude as a function of azimuth and frequency.*

interaural cue. Mouba et al. proposed in [8] the following models:

$$\begin{aligned} \mathrm{ILD}(\theta, f) &= \alpha(f)\sin(\theta) \\ \mathrm{ITD}(\theta, f) &= \beta(f) r \sin(\theta)/c \end{aligned} \quad (2)$$

where α and β are scaling factors obtained from the CIPIC database [11] by matching the model to the data, in the least-square sense (see Figure 4). The overall error for all subjects, azimuths, and frequencies is of 4.29dB for the ILD and 0.052ms for the ITD.

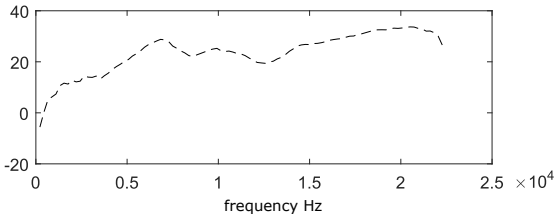
Now that we have synthetic interaural cues (see Figure 5), we can propose synthetic paths respecting these cues, first by computing

$$\begin{aligned} \Delta_a(f) &= \mathrm{ILD}(\theta, f)/20 \\ \Delta_\phi(f) &= \mathrm{ITD}(\theta, f) \cdot 2\pi f \end{aligned} \quad (3)$$

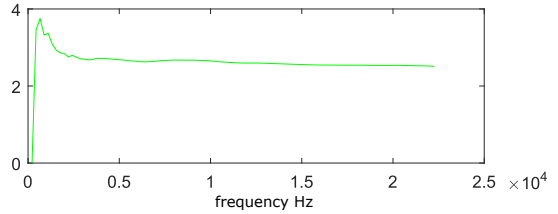
then by using the fact that the left and right HRTFs are roughly symmetric (see Figure 3), we propose

$$\begin{aligned} H_L &= 10^{+\Delta_a(f)/2} \cdot e^{+i\Delta_\phi(f)/2} \\ H_R &= 10^{-\Delta_a(f)/2} \cdot e^{-i\Delta_\phi(f)/2} \end{aligned} \quad (4)$$

where H_L and H_R are the paths going to the left and right ears, respectively.

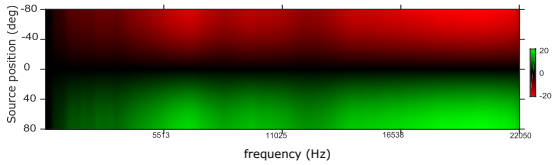


(a) α scaling factor (ILD)

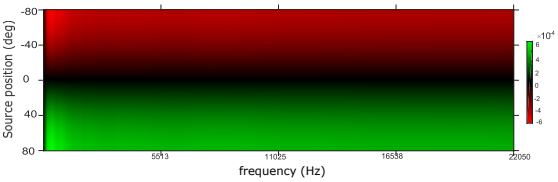


(b) β scaling factor (ITD)

Figure 4: *α and β scaling factors.*



(a) synthetic ILD (dB)



(b) synthetic ITD (s)

Figure 5: *Synthetic ILD (a) and ITD (b).*

2.2. Transaural Principle

The STAR method is largely based on the transaural principle. As shown in Figure 6, we aim at reproducing the paths H_L and H_R between the (virtual) source and the left and right ears of the listener, using the (real) acoustic path between each loudspeaker and each ear (e.g. H_{LR} denoting the path from the left loudspeaker to the right ear).

More precisely, for a given sound s (S being its spectral version), the sounds measured at the left and right should be $H_L \cdot S$ and $H_R \cdot S$, respectively. But to reproduce this virtual source, we use instead two real loudspeakers (the pair at the left and right of the source), thus we must verify the following equation system

$$\begin{aligned} H_L \cdot S &= K_L \cdot H_{LL} \cdot S + K_R \cdot H_{RL} \cdot S \\ H_R \cdot S &= K_L \cdot H_{LR} \cdot S + K_R \cdot H_{RR} \cdot S \end{aligned} \tag{5}$$

where K_L and K_R are some coefficients to be applied to the left and right loudspeakers, respectively. These two coefficients are the solutions of the preceding two-equation system, where S can be simplified.

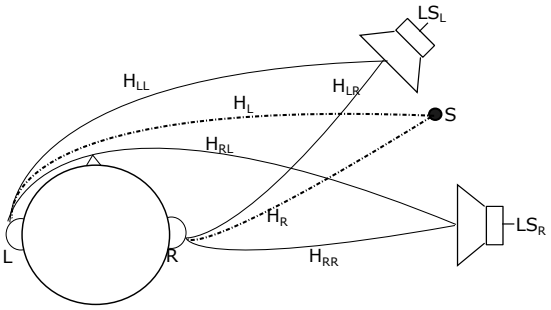


Figure 6: *Transaural principle: 4 real acoustic paths (H_{LL}, H_{RL}, H_{LR}, and H_{RR}) used to reproduce 2 virtual ones (H_L and H_R).*

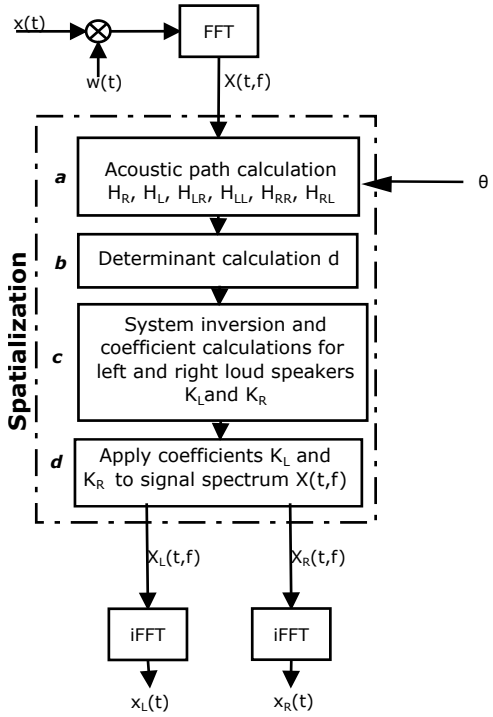


Figure 7: *STAR processing chain.*

Figure 7 then summarizes the whole STAR processing chain.

First (Part a), we start by calculating every acoustic path (as shown on Figure 6), using the procedure described above in Section 2.1, thus with Equation (4).

Second (Part b), we compute the system determinant:

$$d = H_{LR} \cdot H_{RL} - H_{LL} \cdot H_{RR} \tag{6}$$

Third (Part c), it is now possible to invert the system to find the loudspeaker coefficients, and more precisely:

$$\begin{aligned} K_L &= (H_R \cdot H_{RL} - H_L \cdot H_{RR})/d \\ K_R &= (H_L \cdot H_{LR} - H_R \cdot H_{LL})/d \end{aligned} \tag{7}$$

The fourth and last step (Part d) consists in applying the coefficients to the signal spectrum for the left and right speakers of the

speaker pair:

$$X_L(t,f) = K_L(f) \cdot X(t,f)$$
$$X_R(t,f) = K_R(f) \cdot X(t,f) \quad (8)$$

2.3. The Determinant

Of course, things are not so simple in practice. For example, the system determinant d (see Equation (6)) plays an essential role in the STAR method. It shall not approach the 0 value, or else the system is ill-conditioned. As shown in Equation (6), the determinant only depends on the paths of the speakers, thus on the positions of these speakers relatively to the ears of the listener. Of course, a problem would happen if the two speakers were at the same position (which is supposed to be impossible), or very close. Figure 8 shows the determinant norm as a function of the speakers spacing. We see that is necessary to have an angle between the two speakers which is greater than 2 degrees to have a determinant norm greater than 0.01 (the value we chose to guarantee the stability of the system). Hopefully, this will be the case in practice since unlike Wave Field Synthesis (WFS) [5], STAR aims at addressing sparse speaker configurations. In fact, a problem arises with the

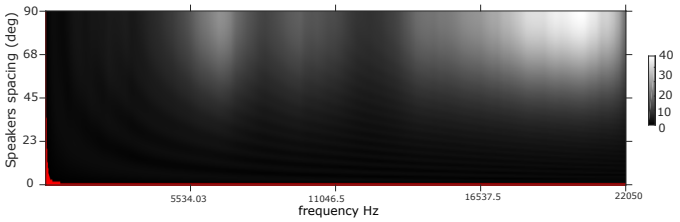


Figure 8: *Determinant norm as a function of frequency and speakers spacing (in red where it is below* 0.01*).*

low frequencies, but hopefully recall that they are filtered out prior to the spatialization.

But if nothing is done, because of the symmetry of our synthetic paths (see Equations (4)), a null determinant problem also appears if the speakers are placed symmetrically with the transaural axis, as in our experimental setup (see Figure 15, LS 2 and 3, or LS 6 and 7). A solution could be to shift the azimuth reference (axis rotation) to break the symmetry of the paths, and move to a problem-free configuration such as the one of Figure 1. This has been done in our first experiments (prior to 2018). For this article, we made a more radical choice: placing the azimuth reference at the center of the loudspeaker pair. This way, the determinant only depends on frequency (and no more on the azimuth). Figure 9 shows the resulting determinant, which is problem-free except for very low frequencies (which are filtered). Studying the influence of this azimuth reference is part of our ongoing research.

2.4. The Coefficients

The calculation of coefficients K_L and K_R described in Equation (7) is the last step of the method.

Even if the determinant of the system is correct, we have to verify that the solutions are also correct. For example, the bigger are the coefficient values, the bigger is the risk to have a saturation on the speakers. Moreover, unlike VBAP or HOA, our coefficients are complex.

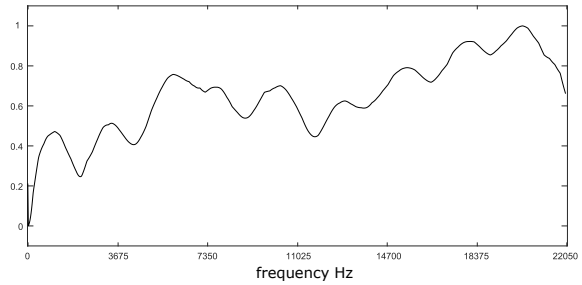


Figure 9: *Determinant norm as a function of the frequency value (here for* $\theta = 0$*).*

Figure 10 shows the modulus of the left and right coefficients depending of the position of the virtual source. The value is mostly between 0 and 1.4, and never exceeds 1.6, thus a risk of saturation exists, for a loud sound source.

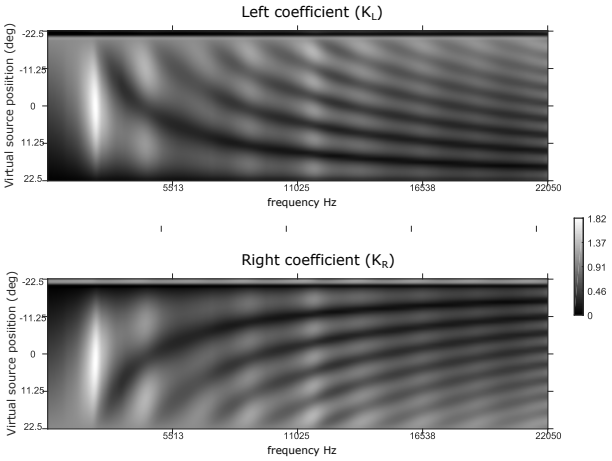


Figure 10: *Magnitude of the* K_L *and* K_R *coefficients as a function of frequency and source azimuth.*

Figure 11 compares the STAR and VBAP coefficients for a virtual source placed at 0 deg, which is in the middle of two speakers in the configuration of our experiment (see Figure 15), the two loudspeakers being then placed relatively at ± 22.5 deg. Although the STAR coefficients are complex, and spectral (i.e. dependent on the frequency), we can see that their modulus stays relatively close to the coefficients of VBAP (which are constant).

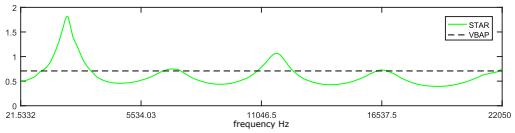


Figure 11: *VBAP (dashed black) and STAR (plain green) coefficients comparison (for* $\theta = 0$*, where the left and right coefficients are the same).*

Figures 11 and 10 also show the sinusoidal and symmetric aspects of the ITD and ILD used to compute the coefficients, in the first step of the method.

3. EXPERIMENTS

The experiments consist of a pre-test followed by three tests. The pre-test is used to identify people who are not sensitive to spatialization. The aim of the three others is to compare the STAR method with two state-of-the-art methods (VBAP and HOA panning) in different situations.

For each of these three tests, a sound example is played using VBAP, HOA, and STAR, together with a spatial anchor consisting of the monophonic version of the sound (with full bandwidth). The sounds are then adjusted to the same volume. The resulting 4 stimuli are randomly attributed a letter (A, B, C, or D), and the sequence (A, B, C, and D) is played and repeated.

The subjects are asked to answer for each method, possibly randomly if they do not know what to answer (for example in the case of the anchor). They also have the possibility to write free comments for each test.

3.1. Preliminary Test

A preliminary test is used for subjects who are neither sound experts nor used to spatial sound. This is a "warm up", so that the subjects listen to the sound system and focus on the spatial aspects. Moreover, this pre-test allows us to identify the subjects who do not pay attention to the spatialization, and thus have to be ruled out from the panel for the results.

For this pre-test, 4 different bird sound examples are played 4 times each, but at different positions. For each example, the first time is the reference, and the subject should retrieve this reference randomly hidden among the 3 other sounds. Thus one position is correct (the one of the reference), and two are far from the one of the reference sound, so that this exercise should be easy for non experts.

3.2. Static Test

In the static test, a musical excerpt is played by a saxophone at a fixed azimuth. The subjects are then asked two things: first, to localize this single source, by placing the letter corresponding to the method on a reference circle (see Figure 12); second, to evaluate the quality of the sound, on a MUSHRA-like [13] notation scale (see Figure 13).

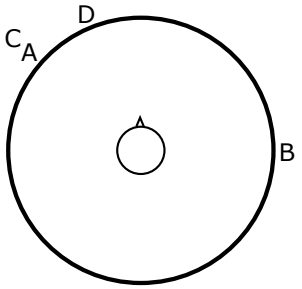


Figure 12: *Reference circle.*

3.3. Dynamic Test

The aim of the dynamic test is to compare the spatialization methods in the case of a moving source. For this purpose, we created a circular trajectory (direct orientation) on a percussive music

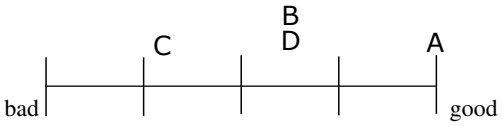


Figure 13: *Notation scale.*

(tablas). Again, the subjects are asked two things: first, to choose the trajectory they find the best among 8 possibilities (see Figure 14); second, again to evaluate the quality of the sound.

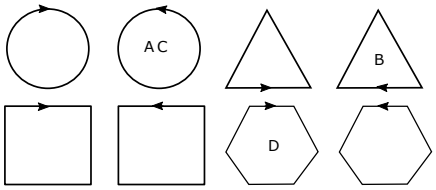


Figure 14: *A choice of* 8 *trajectories, the correct one being the second one (circle in the direct direction).*

3.4. Polyphonic Test

The third test is a polyphonic case. A pop musical song (jazz) is spatialized, with singers and instruments (drums, bass, saxophone, guitar, keyboards) as spatial audio objects, i.e. individual sources distributed in space with positions closest to the choice of the sound engineer for the artistic mix. Moreover, this musical excerpt has the advantage of presenting singing voice with different dynamics, together with various instruments, and ends with a cappella. This time we ask the listeners to evaluate three parameters, all on the notation scale of Figure 13: the quality (like in the two previous tests), the immersion, and the "intelligibility" (or clarity).

4. RESULTS

In 2015, during the Electrocution festival for electroacoustic music in Brest, France, the previous experiments were conducted by a group of Master's students, with an octophonic configuration placed in a quite reverberant concert hall (a former factory made of concrete…), with an audience constituted by composers, sound engineers, and other people with a majority of music professionals used to spatial sound (29 subjects in total). This configuration was exactly the one used for the diffusion of the concerts of the festival.

The conclusion of these experiments showed that STAR had a large sweet spot, was better for the dynamic test (VBAP was the worse, with many hexagons chosen instead of circles, i.e. jumps between loudspeakers), and preferred for the polyphonic test, although the sound timbre has sometimes been described as "nasal quality". Thus, the results were quite promising, although we had to fix this timbre problem, thought to be large coefficient values in the high frequencies (producing a high-pass filter effect).

For the present article, we decided to re-conduct these experiments in La Rochelle, again with an octophonic configuration but in a classroom with moderate reverberation. Figure 15 describes the setup. LS 1 to LS 8 are the active loudspeakers, B are 4 baits (inactive loudspeakers, to artificially increase the complexity of the setup, because it was impossible to hide the other loudspeakers), and 9 seats placed in the middle of this setup (S 1 to S 9).

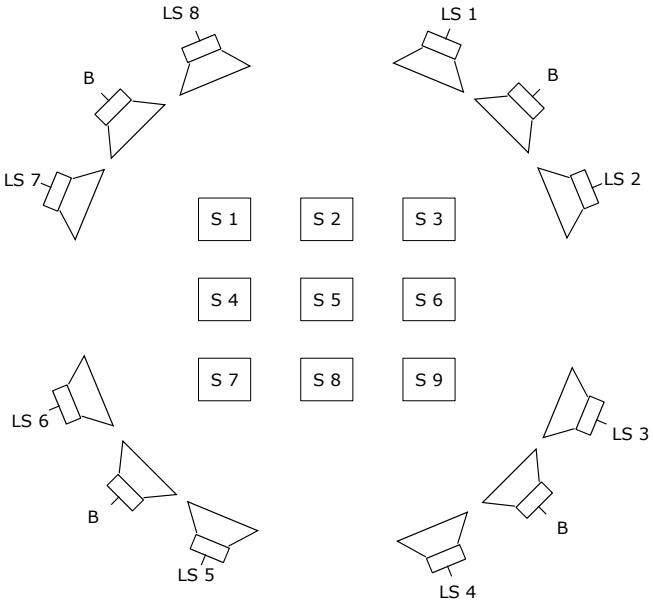


Figure 15: *Experimental setup (2019): 8 active loudspeakers (LS 1 to 8), 4 inactive ones or baits (B), and 9 seats (S 1 to 9).*

The panel consisted of 32 persons (the experiment was run 4 times), almost exclusively composed of amateurs or neophytes on music (only 1 music professional), mainly students and staff of the technical university. After the pre-test only 2 persons were eliminated (people who did not manage to find the reference for at least half of the 4 examples). Thus, the final panel consisted of 30 people (8 women and 22 men), ranging from 17 to 49 years old, with 23 of them below 25 years old.

4.1. Static Test

Figure 16 summarizes the results of the static test, where the listeners were asked to localize a static source, in terms of mean and standard deviation. It appears clearly than the (mono) anchor score is very bad, which is normal. For the three methods the mean is not very far from the real position of the virtual source, but it is clear than VBAP and STAR are better than HOA (which uses all speakers, which can be a drawback for a small room and thus some subjects seating relatively far from the sweet spot or too close to one speaker). The STAR method exhibits the best mean value but a larger standard deviation than VBAP, which seems the best choice. Principal Component Analysis (PCA) showed that the positions of the listeners and their perception of the source position are clearly correlated. More precisely, the listener tends to perceive the sound in the direction of the closest loudspeaker, which is not surprising but problematic.

Regarding the perceived quality, the results (see Figure 17) are quite surprising, because all methods show comparable results, with a mean in the middle and a large standard deviation. This might be a problem with the anchor, which is mono but with full bandwidth, thus with a probably too high quality (for an anchor).

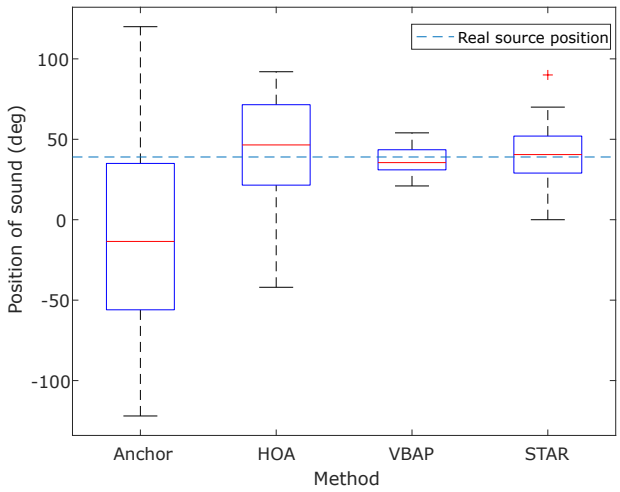


Figure 16: *Static test: perceived source localization. The correct source azimuth is materialized by the dashed line.*

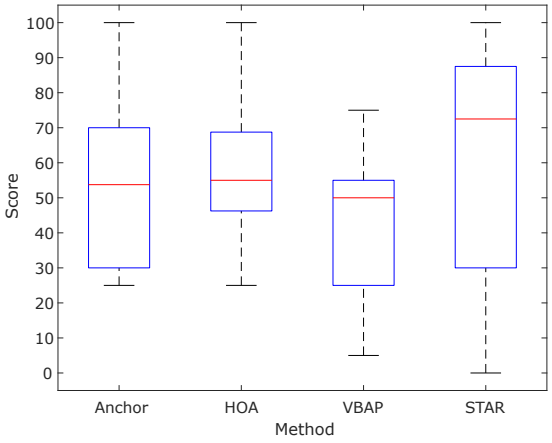


Figure 17: *Static test: perceived sound quality.*

4.2. Dynamic Test

Figure 18 summarizes the results of the dynamic test, where the listeners were asked to recognize the trajectory described in time by a moving source. The anchor exhibits a random behavior, which is normal, since there is no trajectory rendered by this mono version. Then, for all methods, there is some hesitation between the circular (correct one) and hexagonal trajectories. HOA seems to perform best, followed by VBAP then STAR. This looks surprising to us, because for the test conducted in 2015 (see Figure 19) the STAR method was first (and not last…). Apart from the room characteristics and audience qualification, the only change between the 2015 and 2019 tests is the fact that, because we suspected that this was the cause of the "nasal quality", we chose to place the azimuth reference at the center of the loudspeaker pair to improve the system determinant (see Section 2.3). This might be a bad choice.

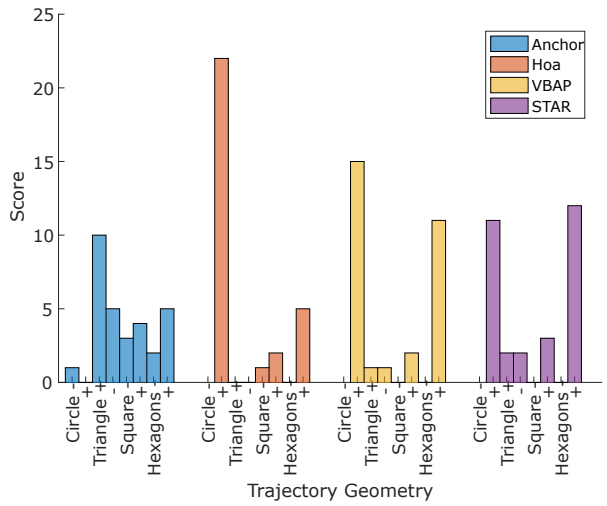


Figure 18: *Dynamic test: perceived source trajectory.*

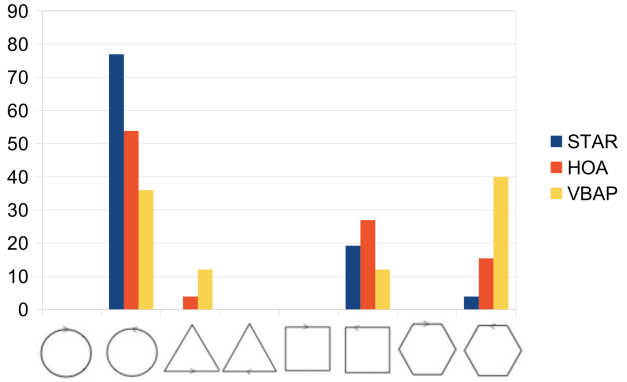


Figure 19: *Dynamic test: perceived source trajectory (values in percents), for the test conducted in 2015.*

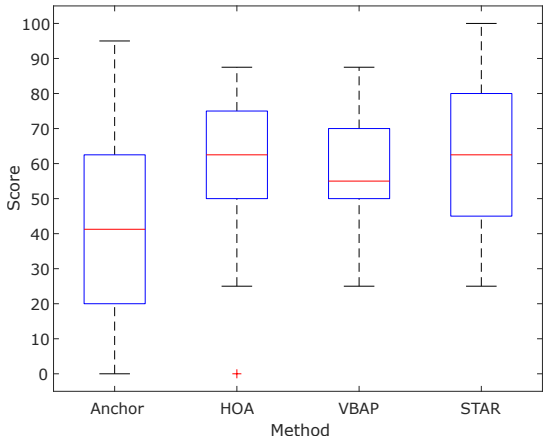


Figure 20: *Dynamic test: perceived sound quality.*

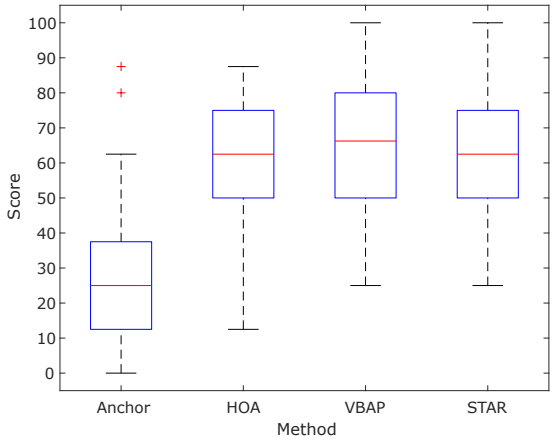


Figure 21: *Polyphonic test: perceived sound quality.*

Regarding the perceived quality, the results (see Figure 20) show that the anchor gets a lower score (which is normal since the anchor does not move), but the score of the three methods are quite similar. Again, this might be a problem of a too high quality anchor for inexperienced listeners.

4.3. Polyphonic Test

Figure 21 shows the perceived quality in the case of the polyphonic test. This time, the anchor is statistically lower, but all the three methods are judged equally good. The results are consistent between 2015 and 2019, even if in 2015 STAR was preferred, but in a non statistically significant way.

In 2019, we asked for subjective immersion and intelligibility (but not in 2015). Figure 22 shows the the perceived immersion is very similar to the perceived quality. Regarding intelligibility (see Figure 23), STAR seems to have some problems, which might be explained by the fact that unlike HOA and VBAP, the STAR coefficients are spectral and complex, thus modify also the phase in a frequency-dependent way, which might help smoothing trajectories but also might modify the timbre of the sound sources.

5. CONCLUSIONS AND FUTURE WORK

In this article we proposed a perceptive approach for sound spatialization. With our STAR method, we focus neither on the wave field (such as HOA) nor on the sound wave (such as VBAP), but rather on the acoustic paths traveled by the sound to the listener ears. The STAR method consists in canceling the cross-talk signals between two loudspeakers and the ears of the listeners (in a transaural way), with acoustic paths not measured but computed by some model (thus synthetic). Our model is based on perceptive cues, used by the human auditory system for sound localization. The aim is to give the listener the sensation of the position of each source, and not to reconstruct the corresponding acoustic wave of field. This should work with various loudspeaker configurations, with a large sweet spot, since the model is neither specialized for a specific configuration nor individualized for a specific listener.

Experimental tests have been conducted in 2015 and 2019 with different rooms and audiences. The positive aspect is that the proposed method is competitive with the state-of-the-art ones. The

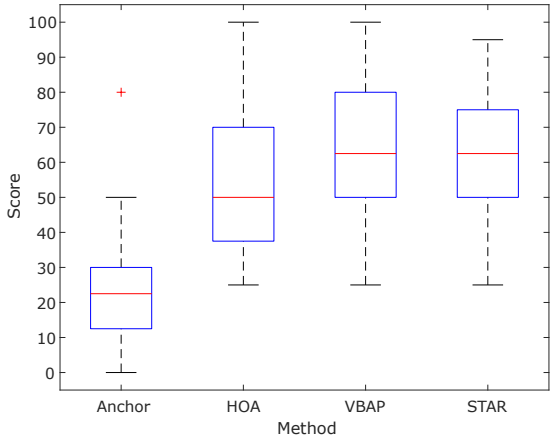


Figure 22: *Polyphonic test: perceived immersion.*

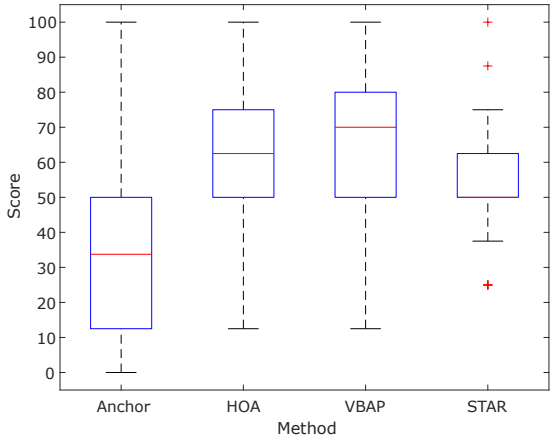


Figure 23: *Polyphonic test: perceived intelligibility.*

negative aspect is that the results are not really consistent between the 2015 and 2019 experiments. One explanation might be the fact that the anchor chosen is only spatial, with mono quality but full bandwidth, which might be too good for the non experts we had in 2019. We plan to re-conduct new tests with a low-pass filtered version, such as in standard MUSHRA tests. Another explanation is that between 2015 and 2019 we chose to place the azimuth reference at the center of the loudspeaker pair to improve the system determinant, because we suspected that this was the cause of the "nasal quality" reported by some listeners. This might be a bad choice, since the performance seems to degrade while the quality is not really improved (although it is quite good).

In the near future we plan to correct these issues, and re-conduct the experiments with a more calibrated loudspeaker configuration (a dome, that should favor HOA), and with expert listeners. Eventually, we will do some A/B testing such as in Marentakis et al. [6]. Finally, we will extend the method to distance and elevation, to generate a full 3D sound.

6. ACKNOWLEDGMENTS

The authors want to thank Aude Besnard, Anaëlle Marsollier, Clément Ruffini, and Aurianne Skybyk, the four Master's student that created the sound signals for the experimental tests and conducted them in 2015 in Brest, as well as Catherine Semal, psychoacoustician, who checked the validity of the test. Finally, thanks go to all the people who participated in the tests in 2015 and 2019.

7. REFERENCES

[1] Jens Blauert, *Spatial Hearing*, MIT Press, Cambridge, Massachusetts, revised edition, 1997, Translation by J. S. Allen.

[2] Ville Pulkki, "Virtual Sound Source Positioning using Vector Base Amplitude Panning," *Journal of the Acoustical Society of America*, vol. 45, no. 6, pp. 456–466, 1997.

[3] Michael A. Gerzon, "Periphony: With-height sound reproduction," *Journal of the Audio Engineering Society*, vol. 21, no. 1, pp. 2–10, 1973.

[4] Jérôme Daniel, *Représentation de champs acoustiques, application à la transmission et à la reproduction de scènes sonores complexes dans un contexte multimédia*, Ph.D. thesis, Université Paris 6, 2001.

[5] A. J. Berkhout, "A Holographic Approach to Acoustic Control," *Journal of the Audio Engineering Society*, vol. 36, pp. 977–995, 1988.

[6] Georgios Marentakis, Franz Zotter, , and Matthias Frank, "Vector-Base and Ambisonic Amplitude Panning: A Comparison Using Pop, Classical, and Contemporary Spatial Music," *Acta Acustica united with Acustica*, vol. 100, no. 5, pp. 945–955, 2014.

[7] Joan Mouba and Sylvain Marchand, "A Source Localization/Separation/Respatialization System Based on Unsupervised Classification of Interaural Cues," in *Proceedings DAFx'06*, Montreal, Quebec, Canada, September 2006, pp. 233–238.

[8] Joan Mouba, Sylvain Marchand, Boris Mansencal, and Jean-Michel Rivet, "RetroSpat: a Perception-Based System for Semi-Automatic Diffusion of Acousmatic Music," in *Proceedings of the Sound and Music Computing (SMC'08) Conference*, Berlin, Germany, July/August 2008, p. 33–40.

[9] Jeroen Breebaart, "Analysis and Synthesis of Binaural Parameters for Efficient 3D Audio Rendering in MPEG Surround," in *IEEE International Conference on Multimedia and Expo (ICME)*, Beijing, China, July 2007.

[10] John M. Chowning, "The Simulation of Moving Sound Sources," *Journal of the Acoustical Society of America*, vol. 19, no. 1, pp. 2–6, 1971.

[11] V. Ralph Algazi, Richard O. Duda, Dennis M. Thompson, and Carlos Avendano, "The CIPIC HRTF Database," in *Proceedings IEEE WASPAA*, New Paltz, New York, 2001, pp. 99–102.

[12] Harald Viste, *Binaural Localization and Separation Techniques*, Ph.D. thesis, École Polytechnique Fédérale de Lausanne, Switzerland, 2004.

[13] "ITU-R BS.1116–3, Methods for the Subjective Assessment of Small Impairments in Audio Systems Including Multichannel Sound Systems," 2015.

TIME SCALE MODIFICATION OF AUDIO USING NON-NEGATIVE MATRIX FACTORIZATION

Gerard Roma
CeReNeM
University of Huddersfield
Huddersfield, UK
g.roma@hud.ac.uk

Owen Green
CeReNeM
University of Huddersfield
Huddersfield, UK
o.green@hud.ac.uk

Pierre Alexandre Tremblay
CeReNeM
University of Huddersfield
Huddersfield, UK
p.a.tremblay@hud.ac.uk

ABSTRACT

This paper introduces an algorithm for time-scale modification of audio signals based on using non-negative matrix factorization. The activation signals attributed to the detected components are used for identifying sound events. The segmentation of these events is used for detecting and preserving transients. In addition, the algorithm introduces the possibility of preserving the envelopes of overlapping sound events while globally modifying the duration of an audio clip.

1. INTRODUCTION

Time-scale modification (TSM) of audio signals is nowadays an essential audio processing tool in music and audio production. TSM became particularly popular in music creation workflows based on the reuse of readily available audio. A common goal in this context is to stretch audio clips, that often contain their own rhythmic micro-structures, so that they will match a given musical context. It can be argued that the key aspect for preserving the structure is the location of sound events in time. Existing TSM algorithms will also change the duration of the sounds themselves, for instance, the duration of a drum hit, which is not necessarily desirable and may sound unnatural. On the other hand, artifacts of TSM algorithms are used as musical features in electronic music experimentation. Different algorithms will produce different kinds of artifacts so they can be creatively abused to produce different sound effects. In the end, a common situation in music production software is to choose between different TSM algorithms that may perform differently for different material.

One difficulty in TSM is the presence of overlapping sound events of different durations and temporal structures. Given recent advances in audio source separation research, we are interested in whether source separation algorithms could help with TSM. Even when separation is not perfect, mixing the estimates of sound sources back together often helps to diminish any artifacts introduced in the separation. This feature can be used to improve TSM by allowing separate scaling of the component sounds of an audio excerpt.

A key algorithm often used for source separation is non-negative matrix factorization (NMF). NMF is an unsupervised method, which has been shown to produce good results for transcription and separation of signals with a clear percussive profile, such as piano sounds [1] or drums [2]. Since it has to learn from the signal (typically the magnitude spectrogram), its effectiveness depends largely on the invariants present in the structure of the data, as the algorithm tries to reconstruct it from a limited set of spectral patterns and their activations. In this sense, while NMF is limited for dealing with polyphonic music, it can often cope well with short sounds such as the musical loops used for some types of rhythmic music. NMF will essentially capture repetitions of spectral patterns, so, for example, in drum patterns it will capture the structure of activations of drum sounds. Similarly, in the case of pitched sounds with fixed spectrum such as piano notes, it can easily capture the structure of tonal melodies.

In this paper we investigate the use of NMF for TSM. By separating different sound events, we expect NMF to make it possible to stretch them separately, allowing the envelope of overlapping events, such as percussive instruments and resonant bodies, to be perceived more naturally. In addition, by introducing a new way of modifying the duration of audio signals, our algorithm introduces a different kind of artifact for materials on which the algorithm fails to produce natural sounding results, which in turn may yield new affordances for sound design and music. Specifically, when NMF fails to identify note-like events, the proposed algorithm may produce rhythmic modifications or even more extreme misplacing of parts of the signal. These artifacts could no doubt be creatively abused by experimenters seeking new sonorities.

The paper is organized as follows. In the next section, we briefly review related work in the field of TSM. In Section 3, we describe our proposed algorithm for TSM using NMF. In Section 4 we discuss some examples that illustrate the potential of the algorithm. Finally, we draw some conclusions and discuss future work.

2. RELATED WORK

Research in TSM was relatively active between the 1980s and early 2000s. Algorithms developed back then, such as time domain overlap and add (OLA), waveform similarity overlap and add (WSOLA) [3], or the phase vocoder [4], remain popular. The latter two algorithms provide relatively good results for music content, but introduce very strong artifacts when stretching transients. Both WSOLA and the phase vocoder have been thus improved with transient detection [5, 6, 7]. Academic research on TSM considerably slowed down afterwards. Perhaps due to the success of TSM for music and audio production in digital audio workstations, industry took the lead. During the last few years, research on TSM has been growing again. This may be partially due to the potential of audio source separation research. To some extent, growing interest in reproducible research in audio signal processing can also be credited for the renewed interest, since algorithms used in com-

mercial products are often not well known outside the companies that make them.

A significant recent contribution was provided in [8] by applying harmonic-percussive source separation (HPSS) and then using OLA for percussive estimates and the phase vocoder for harmonic estimates. The authors also presented the Matlab TSM toolbox, including the classic algorithms and their own, in [9]. The algorithm proposed in [10] applies a similar concept, but unlike [8] it is able to keep transient processing in the frequency domain.

In this paper, we explore the use of another source separation technique, NMF, which allows different treatment of different sources more generally than transient/harmonic separation. The activation curves resulting from the NMF separation are used as a cue for the presence of transients due to specific components. Our approach is inspired by the system presented in [5]. An important detail is that, unlike in the classic phase vocoder, the system in [5] proposed the use of the same hop size for both the analysis and synthesis stages. Using the same hop size allows using the (slower) NMF decomposition as part of the analysis stage, while different scaling factors can be tried in the synthesis stage without the need to analyze again. It also allows tuning all the windowing parameters as suitable for the input material.

3. PROPOSED ALGORITHM

3.1. Overview

We now briefly describe the proposed system. A block diagram is shown in Figure 1. The system is intended for time-scale modification of relatively short (e.g. a few seconds) audio signals. The time domain audio signal is first converted to a spectrogram via the short-time Fourier transform (STFT). The magnitude spectrogram is then decomposed into several components via NMF. As per the NMF framework (described in Section 3.3), each component consists of a basis function and an activation function. Components are then segmented into sound events by analysis of the activation function. The activation function is also used to identify one or more transients within a given sound event. For each of the resulting events, a number of frames are copied verbatim into the synthesized spectrogram. These can be either the detected transients or the whole event, which can be preferred for percussive sounds. For the remaining frames, a new scaling factor is computed in order to respect the scaled duration for the whole event. Time scaling is then applied following the principles of the phase vocoder. The resulting component spectrograms are then mixed and synthesized via inverse STFT.

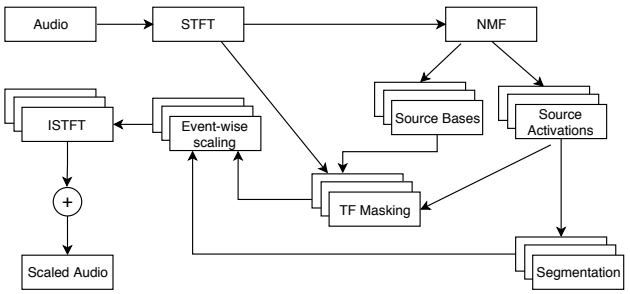


Figure 1: *Block diagram of the proposed system.*

3.2. Source separation

Our system is based on the assumption that the input signal is a mixture of I component signals,

$$x(t) = \sum_{i=1}^{I} x_i(t) \tag{1}$$

Here, the component signal x_i is assumed to have the appropriate gain with respect to the mixture. We want to produce a time-scaled version of x, y, which is analogously a mixture of signals $\sum_i y_i(t)$. Assuming the mixing model holds in the (complex) frequency domain, we estimate the component frequency-domain signals $X_i(k,n)$ from the STFT $X(k,n)$ of x (where k and n are respectively frequency and time indices) via NMF. We then stretch each component X_i into Y_i, and obtain y_i via inverse STFT. The stretched components are then mixed in the time domain.

3.3. NMF decomposition

NMF is typically applied to a magnitude spectrogram. TSM will require using both the magnitude spectrogram, denoted as

$$V = |X| \tag{2}$$

and the phase spectrogram, denoted as

$$\Phi = \angle(X) \tag{3}$$

Under the NMF framework, an approximation of V is obtained as

$$\hat{V} = WH \tag{4}$$

The matrix $W \in \mathbb{R}^{K \times I}$ contains a set of I bases, which typically represent static spectra corresponding to each of the detected sources. The matrix $H \in \mathbb{R}^{I \times N}$ contains a corresponding set of I activations, which represent the temporal envelopes of each component. We can use these activations to find the positions of transients (typically corresponding to note onsets) and general active regions corresponding to each component, thus applying different stretch factors to preserve the structure of sound events. In addition, it is often possible to classify the bases into tonal and percussive sounds [11, 12], which could be used for applying different stretching strategies similar to [8]. However in this paper we just consider the option of preserving the duration of the active region as a user parameter.

For this to work it is of course crucial to obtain a good decomposition that represents the perceived components of the signal. A common strategy is to minimize the divergence

$$D_{KL}(V|WH) = \sum_{kn} d_{KL}(V(k,n)|\sum_{i} W_i(k)H_i(n)) \tag{5}$$

where

$$d_{KL}(x|y) = x \log \frac{x}{y} - x + y. \tag{6}$$

As originally proposed in [13], this can be done via a simple multiplicative update algorithm. This can often produce noisy activation functions, which make the segmentation step more difficult. Some works have proposed constraining the objective function to

produce smooth activations. In this work we use the NMF algorithm presented in [14]. Here, a penalty factor is introduced to promote smoothness of H:

$$S(H) = \frac{1}{2} \sum_i \sum_n (H_i(n) - H_i(n-1))^2. \quad (7)$$

The algorithm then tries to minimize the cost function

$$D_{KL}(V|WH) + \beta S(H), \quad (8)$$

subject to the non-negativity constraints of the NMF framework. Here β is a parameter that can be used to control the smoothness of the resulting activation curves, at the expense of the algorithm having a harder time finding the appropriate components. In our experiments, a value of $\beta = 0.1$ (an order of magnitude higher than in [14]) worked well in most cases.

Another challenge with NMF is in how to select the rank, I, of the decomposition for a particular source. One approach is proposed in [15], where a singular value decomposition (SVD) is performed on V. The SVD of a matrix has the form $Z = U\Sigma V^\intercal$, where the singular values of Z lie along the diagonal of Σ. The NMF rank, I, is then estimated by finding the number of singular values that account for some proportion of the total sum along the diagonal of Σ.

From the NMF decomposition, we obtain a soft mask

$$M_i = \frac{W_i H_i}{\hat{V}} \quad (9)$$

which we can apply to the original magnitude and phase spectrograms to obtain estimates for each component, $\hat{V}_i = M_i \odot V$, $\hat{\Phi}_i = M_i \odot \Phi$ (where $\odot$ denotes the element-wise product).

3.4. Event segmentation

Segmentation is based on the observation that activations tend to loosely follow a binary on-off pattern (Figure 2). We identify sound events when the activation is above a certain threshold defined as $\mu_i + \tau_1 \sigma_i$ (where μ_i and σ_i are respectively the mean and standard deviation of the activation $H_i(n)$, and τ_1 is a paramerter) for more than 3 frames. The end of the event is then adjusted to when the activation crosses a typically lower threshold determined in the same way for a parameter τ_2. We then look for transients within the event by identifying peaks in the first order difference of $H_i(n)$, and pick them in the same way by a third threshold parameter τ_3.

When multiple transients are found within the same active event (e.g. for a rapid succession of percussive or note events with long decay), the event is split so that each transient will always start an event (although in general it is not required that all events start with a transient). A transient is defined to have a fixed number of frames corresponding to 10ms (which can be controlled as a user parameter), depending on the hop size. Finally, the 'silence' in the activation between two events is attached to the preceding event, so that an event is considered to have a transient, an active part and a silence part, where transient and silence may have zero duration. The idea is that—as the signal has a rhythmic structure—we need to proportionally scale the spaces between event onsets, but within an event we may apply different scaling.

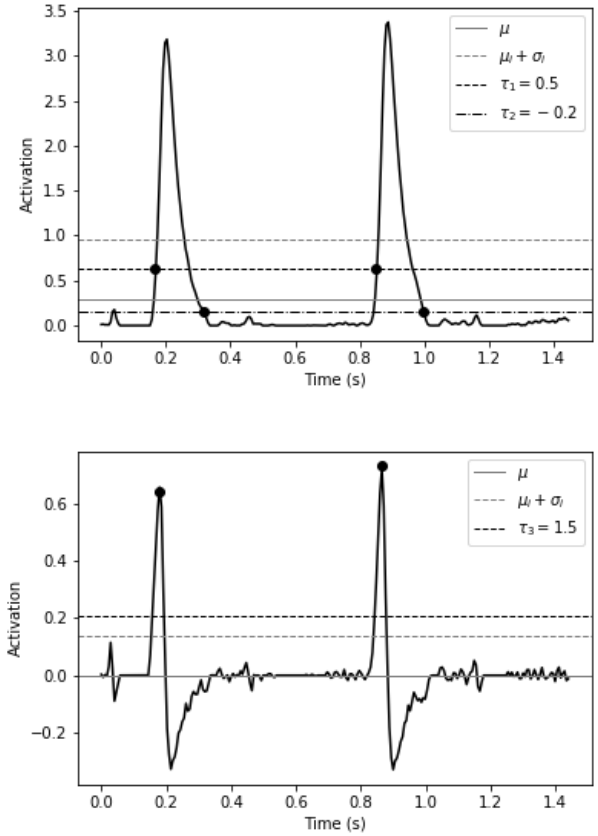


Figure 2: *Event segmentation of NMF activation. Top: activation function. Bottom: first order difference.*

3.5. Scaling and synthesis

The segmented events for each of the NMF components are then scaled according to the desired factor, r. We can take advantage of the component- and event-wise separation in several ways. First, in order to preserve the perceptual quality of transient, the transient part of each event is not scaled. By default, a new scaling factor r_A is computed for the active part. An optional feature, which can be called *envelope preservation* may be used so that the active part is also copied without scaling as if it was a transient. The silence scaling factor r_S has then to be recomputed to keep the whole of the event aligned according to r. When the event has no silence (typically as a result of splitting the event due to multiple transients) envelope preservation is not applied. Outside of transient and potentially percussive events, magnitudes are interpolated from the input $\hat{V}_i$, and phases are propagated according to the phase vocoder strategy, including identity phase locking [16]. Thus, after finding the bin k_p corresponding to the peak in the region of influence of a given magnitude bin k, we compute the phase envelope as

$$\phi_e(k,n) = \Phi(k,n) - \Phi(k_p,n), \quad (10)$$

and the deviation with respect to the bin's frequency as

$$\Delta\phi(k,n) = \Phi(k,n) - \Phi(k,n-1) - \omega(k)R \quad (11)$$

where $w(k)$ is the normalized frequency corresponding to bin k, and R is the hop size. The phase for the scaled signal in non-transient regions is then synthesized as

$$\Phi_{Yi}(k,n) = \Phi_{Yi}(k,n-1) + \omega(k)R + Arg(\Delta\phi(k,n) + \phi_e(k,n)), \quad (12)$$

where $Arg(x)$ is the principal argument function.

The estimates of the scaled spectrogram, $\hat{V}_{Yi}$ and $\hat{\Phi}_{Yi}$, are then composed into $\hat{Y}_i$ as

$$\hat{Y}_i(k,n) = \hat{V}_{Yi}(k,n)\, \mathrm{e}^{\hat{\Phi}_{Yi}(k,n)j}, \quad (13)$$

and synthesized using the inverse STFT as described in Section 3.2.

4. EXAMPLES

We implemented the proposed algorithm in a Python package, which includes a partial port of the Matlab TSM toolbox. The code is available on github[1]. While testing with several excerpts, we found that results are generally better than classic methods such as OLA or WSOLA, or the phase vocoder without transient preservation, and closer to HPSS and state-of-the-art commercial packages. A number of examples can be listened to on the companion web page for this paper[2]. It is possible to obtain good results by automating the choice of the NMF rank as outlined in Section 3.3, which often produces a large number of components. This is an interesting result, considering that the TSM is performed piecewise through potentially several hundreds of events and then reassembled, however it bears a high computational cost. In practice, a better solution is often to manually set a suitable value for the NMF rank. Generally, the algorithm works better for percussive and repetitive material, and suffers with slow frequency or amplitude modulations. With respect to the envelope preservation option, it is generally sensitive to errors in the detection of events, and hence it tends to work best for sounds that are well modeled by NMF, such as percussive loops. In these cases it can produce a more natural sound than most available algorithms. Our approach is generally comparable to using HPSS [8], which is also based on a source separation technique that decomposes the magnitude spectrogram. We now demonstrate the strengths and weaknesses of the NMF decomposition through some examples.

4.1. Glockenspiel

Using NMF tends to produce better attacks for simple percussive loops and melodies. Figure 3 shows the spectrogram of a few notes of the Glockenspiel melody included in the TSM Toolbox, as stretched by both the HPSS and the NMF approaches. It can be seen that using NMF produces sharper transient at note onsets. This is probably partly due to the NMF activations providing a good cue of the locations of transients due to individual notes, but also to the fact that our algorithm stays in the same frame rate, allowing to build a more coherent representation of the note, while the HPSS approach requires going back to the time domain for the percussive part, while staying in the frequency domain for stretching the harmonic part.

[1] https://github.com/flucoma/DAFX-2019

[2] http://www.flucoma.org/DAFX-2019/

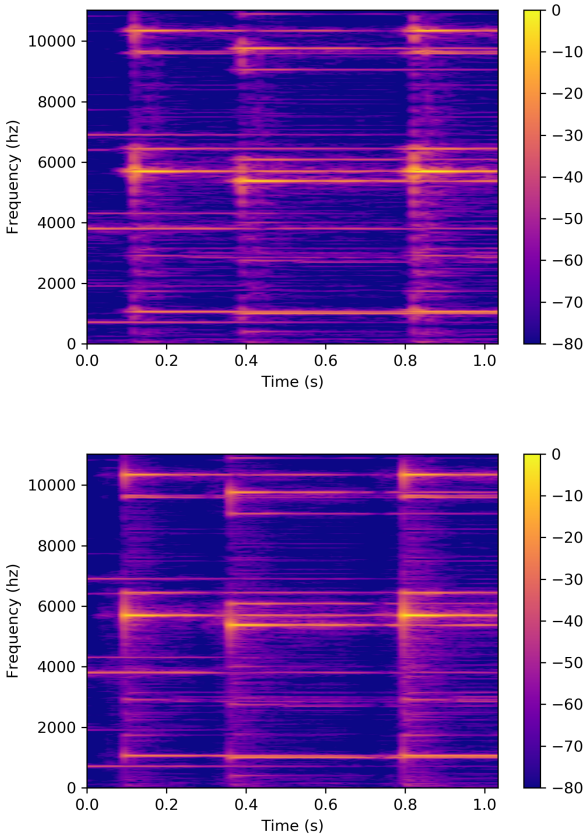


Figure 3: *Excerpt of a glockenspiel melody stretched with a 1.8 factor using HPSS (top) and NMF (bottom)*

4.2. Drum loop

One unique aspect of the proposed algorithm is the ability to modify the duration of the signal while preserving the envelope of percussive sounds. This option may introduce some artifacts if the events are not well detected, but it works well for dry percussive loops. An example is shown in Figure 4, showing the initial beats of a drum loop. Here, we can observe that the NMF approach approximates better the duration of the first two sound events (a bass drum and a hi-hat). Our approach modifies the tempo of the pattern while preserving the natural sound of each beat, while stretching via HPSS also stretches the sound's envelope, which gives it an artificial time profile. The latter is also generally the case in current commercial products.

4.3. Novel artifacts

As mentioned in the introduction, we are also interested in the creative possibilities of the failures of TSM algorithms. In this sense, we hope the proposed algorithm will also contribute new kinds of artifacts that can be used for exploring new musical possibilities. The main user parameters are the NMF rank (I), and the three parameters influencing the event segmentation (Section 3.4). Without the envelope preservation feature, our algorithm can reproduce common artifacts related with the phase vocoder. For

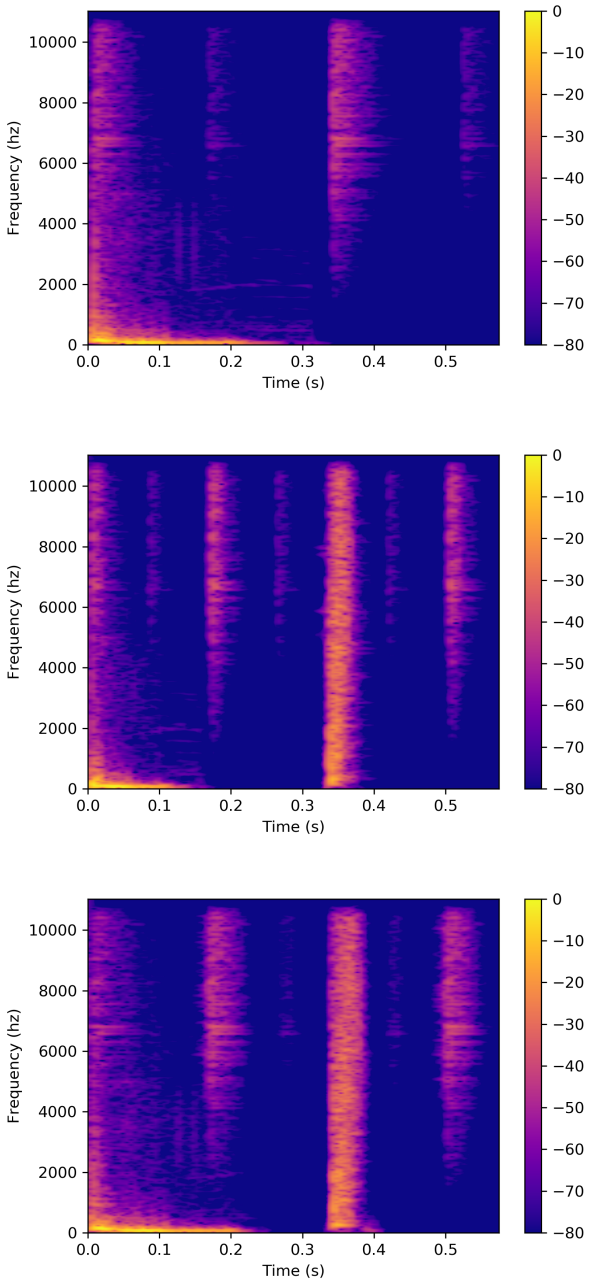


Figure 4: *Excerpt of a drum loop (top:original) stretched with a 0.6 factor using HPSS (middle) and NMF (bottom).*

example, raising the transient detection threshold τ_3 can be used to induce transient smearing, while extreme stretching factors will produce well-known phasing effects. The novel contribution of our algorithm is the possibility to preserve the envelope of a sound event, but when events are not properly detected, this results in misplaced components that produce rhythmic variations and different smearings of time not usually found in phase vocoder. An example (zoomed again for detail) is shown in Figure 5. Here, we intentionally raised the chances of mistakes by raising the rank to 20 (which is more than needed for a drum kit using mainly four sounds) and made it difficult for the algorithm to find the events by playing with parameters τ_1 and τ_2. As a result, part of the rhythm becomes confusing. The main impression is that some of the sounds have been divided and parts of them have been misplaced, creating a new rhythmic effect.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented an algorithm for time scale modification of audio using non-negative matrix factorization. We have presented an implementation and demonstrated several examples. The algorithm has the unique feature of being able to preserve the duration of sound events while modifying the duration of the sequence. This is generally not possible without source separation, unless the signal is purely monophonic, as the envelopes of different events tend to overlap. The NMF framework also helps generally in the identification of transients due to different components in the frequency domain. As future work, we plan to investigate strategies for synchronizing event boundaries across components so that envelope preservation can be used without compromising rhythmic structure. Similarly, classification of NMF bases would allow applying selectively to percussive events. Also, since frequency-domain TSM generally requires careful attention to the phase, we plan to experiment with complex NMF variants.

6. ACKNOWLEDGMENTS

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement n. 725899).

7. REFERENCES

[1] Paris Smaragdis and Judith C Brown, "Non-negative matrix factorization for polyphonic music transcription," in *Proceedings of the 2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2003, pp. 177–180.

[2] Jouni Paulus and Tuomas Virtanen, "Drum transcription with non-negative spectrogram factorisation," in *Proceedings of the 2005 13th European Signal Processing Conference*, 2005, pp. 1–4.

[3] Werner Verhelst and Marc Roelands, "An overlap-add technique based on waveform similarity (wsola) for high quality time-scale modification of speech," in *Proceedings of the 1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 1993, vol. 2, pp. 554–557.

[4] Jean Laroche and Mark Dolson, "Phase-vocoder: About this phasiness business," in *Proceedings of 1997 Workshop on Applications of Signal Processing to Audio and Acoustics*, 1997, pp. 4–pp.

[5] Jordi Bonada, "Automatic technique in frequency domain for near-lossless time-scale modification of audio.," in *Proceedings of the 2000 International Computer Music Conference*. Citeseer, 2000.

[6] Frederik Nagel and Andreas Walther, "A novel transient handling scheme for time stretching algorithms," in *Proceedings of the 127th Audio Engineering Society Convention*, 2009.

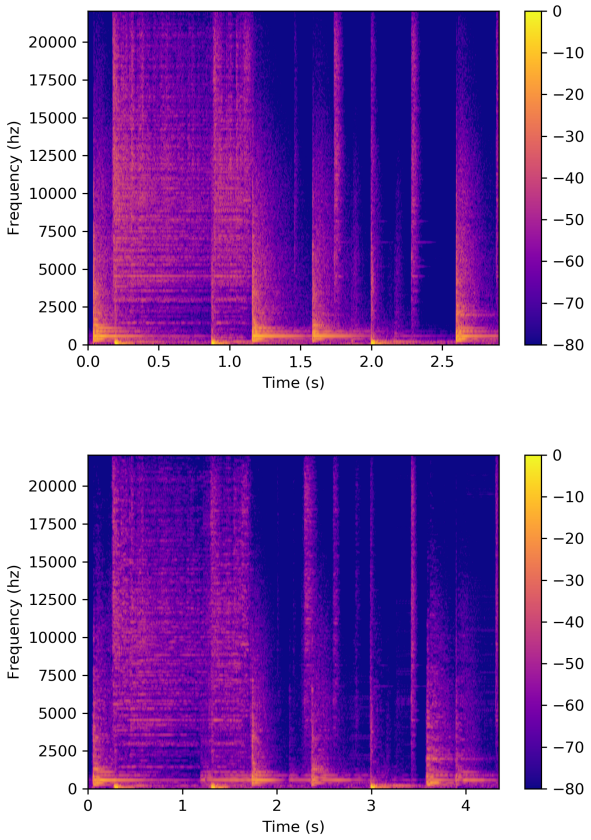


Figure 5: *Excerpt of a drum recording. Top: original. Bottom: stretched using NMF with a 1.5 factor and extreme parameters.*

[7] Shahaf Grofit and Yizhar Lavner, "Time-scale modification of audio signals using enhanced wsola with management of transients," *IEEE transactions on audio, speech, and language processing*, vol. 16, no. 1, pp. 106–115, 2008.

[8] J. Driedger, M. Müller, and S. Ewert, "Improving Time-Scale Modification of Music Signals Using Harmonic-Percussive Separation," *IEEE Signal Processing Letters*, vol. 21, no. 1, pp. 105–109, Jan. 2014.

[9] Jonathan Driedger and Meinard Müller, "TSM Toolbox: MATLAB Implementations of Time-Scale Modification Algorithms," in *Proceedings of the 2014 Conference on Digital Audio Effects (DAFX)*, 2014, p. 8.

[10] Eero-Pekka Damskägg and Vesa Välimäki, "Audio Time Stretching Using Fuzzy Classification of Spectral Bins," *Applied Sciences*, vol. 7, no. 12, pp. 1293, Dec. 2017.

[11] Marko Helen and Tuomas Virtanen, "Separation of drums from polyphonic music using non-negative matrix factorization and support vector machine," in *Proceedings of the 13th European Signal Processing Conference (EUSIPCO)*, 2005, pp. 1–4.

[12] Jordi Janer, Ricard Marxer, and Keita Arimoto, "Combining a harmonic-based nmf decomposition with transient analysis for instantaneous percussion separation," in *Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 281–284.

[13] Daniel D Lee and H Sebastian Seung, "Algorithms for non-negative matrix factorization," in *Advances in neural information processing systems*, 2001, pp. 556–562.

[14] Slim Essid and Cédric Févotte, "Smooth nonnegative matrix factorization for unsupervised audiovisual document structuring," *IEEE Transactions on Multimedia*, vol. 15, no. 2, pp. 415–425, 2013.

[15] Hanli Qiao, "New svd based initialization strategy for non-negative matrix factorization," *Pattern Recognition Letters*, vol. 63, pp. 71–77, 2015.

[16] J. Laroche and M. Dolson, "Improved phase vocoder time-scale modification of audio," *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 3, pp. 323–332, May 1999.

AN FPGA-BASED ACCELERATOR FOR SOUND FIELD RENDERING

Yiyu Tan *
RIKEN Center for Computational Science
Kobe, Hyogo, Japan
tan.yiyu@riken.jp

Toshiyuki Imamura
RIKEN Center for Computational Science
Kobe, Hyogo, Japan
imamura.toshiyuki@riken.jp

ABSTRACT

Finite difference time domain (FDTD) schemes are widely applied to analyse sound propagation, but are computation-intensive and memory-intensive. Current sound field rendering systems with FDTD schemes are mainly based on software simulations on personal computers (PCs) or general-purpose graphic processing units (GPGPUs). In this research, an accelerator is designed and implemented using the field programmable gate array (FPGA) for sound field rendering. Unlike software simulations on PCs and GPGPUs, the FPGA-based sound field rendering system directly implements wave equations by reconfigurable hardware. Furthermore, a sliding window-based data buffering system is adopted to alleviate external memory bandwidth bottlenecks. Compared to the software simulation carried out on a PC with 128 GB DDR4 RAMs and an Intel i7-7820X processor running at 3.6 GHz, the proposed FPGA-based accelerator takes half of the rendering time and doubles the computation throughput even if the clock frequency of the FPGA system is about 267 MHz.

1. INTRODUCTION

Sound field rendering models sound propagation in spatial and time domains, and is fundamental to numerous scientific and engineering applications, which vary widely from interactive applications, such as computer games and virtual reality, to offline applications like architectural design and noise control. Generally, the sound field rendering algorithms are categorized into geometric methods and wave-based methods. The geometric methods make the assumption that surface primitives are much larger than the wavelength of sound. As a result, the low-frequency diffraction effects of sound wave are lost while the computation capability is reduced significantly. Nowadays, the geometric methods are widely applied in interactive applications because of easy implementation and low computation demand. In contrast, the wave-based methods, such as finite difference time domain (FDTD) methods, boundary element and finite element methods, directly solve the acoustic wave equations in either time domain or frequency domain using numerical methods, and they provide highly accurate modelling of all aspects of sound propagation, including full wave diffraction.

Among the wave-based methods, the FDTD method, which numerically solves the wave equation by using a finite number of grids in a discretized space at discrete time steps, has become an essential approach in room acoustic simulation owing to its ease of implementation and parallelization since it was introduced to analyse acoustical behaviour by O. Chiba et al., and D. Botteldooren et al. [1–3]. However, the FDTD method suffers from the numerical dispersion, which is an inherent problem constraining the valid usable bandwidth.

To reduce numerical dispersion in the FDTD method, many works have been done in 3-D scheme. L. Savioja et al., G. R. Campos et al., and D. Murphy et al. proposed alternative digital waveguide mesh topologies [4–6]; K. Kowalczyk and M. Walstijn developed the explicit second-order accurate schemes, including the 27-point compact explicit FDTD scheme [7]. J. Mourik and D. Murphy investigated two-step high-order explicit "large-star" schemes [8]. B. Hamilton and S. Bilbao introduced the fourth-order accurate explicit and implicit FDTD schemes for 2-D and 3-D wave equations [9][10], respectively. They also developed a set of two-step explicit FDTD schemes with high-order accuracy in both spatial and time domains for 3-D room acoustics [11].

On the other hand, numerical dispersion is still challenge in sound field rendering with the FDTD method, and conventional approach to alleviate the effects of numerical dispersion through spatial grid oversampling incurs significant computational cost. This results in the FDTD method scales poorly with the volume of sound spaces and the analysed maximum frequency. Generally, the computing capability of solving wave equations in the FDTD method is increased as the fourth power of frequency [12] and proportionally with the volume of sound spaces. Given the auditory range of humans (20 Hz–20 kHz), simulating sound wave propagation in a space like a concert hall or a cathedral for the maximum simulation frequency of 20 kHz requires petaflops of computing power and terabytes of memory. Only large computer cluster or supercomputer can satisfy such requirements in current computer systems, but they are prohibitive expensive.

In recent years, GPGPUs and FPGAs were applied to accelerate computation in sound field rendering because of their coarse-grain parallelism of thousands of arithmetic units [13-22]. Currently, an FPGA chip has more hardware resources owing to the development of fabrication technology, including thousands of hardened floating-point arithmetic units, large on-chip block memories, millions of reconfigurable logic blocks. Unlike software simulations on PCs and GPGPUs, FPGA-based sound rendering systems directly implement sound wave equations by configurable logic blocks and hardened arithmetic units inside an FPGA. The system data-path and input/output (I/O) interfaces can be customized in accordance to applications, and thousands of arithmetic units are coordinated to work in parallel to improve computation performance. The incident signals and the rendering results can be directly put in and out through the customized I/O interfaces. From the point of view of real-time processing, FPGA provides a promising solution to real-time sound rendering applications. In our previous work, a FPGA-based accelerator was developed for real-time sound field rendering [17-23]. Although the accelerator outperformed PC-based simulations significantly in

* This work was supported by the I-O DATA foundation and JSPS KAKENHI Grant Number JP19K12092.

rendering speed, the rendered sound space is small (32 × 32 × 16 grids) because only small on-chip block memories were applied. In this paper, a FPGA-based accelerator for sound rendering is designed by using high-level synthesis approach, and large external memory is applied to extend the rendering sound volume. The main contributions of this work are shown as follows.

(1) Uniform computing units in rendering algorithm to simplify system design. The explicit FDTD rendering algorithm and its uniform computing format are derived, which makes it easy to design the computing unit in hardware system.
(2) Sliding window-based data buffering to reduce the data access overhead and the requirement of memory bandwidth.
(3) Design and implementation of an FPGA-based accelerator for sound field rendering by using high-level synthesis, including design flow, system architecture, and system implementation.
(4) Evaluation and analysis of system performance based on the prototype machine. The rendering time and computation throughput of the FPGA-based prototype machine are evaluated and compared with those of the software simulation carried out on a PC with 128 GB DDR4 RAMs and an Intel i7-7820X processor running at 3.6 GHz.

The rest of this paper is organized as follows. The rendering algorithm is introduced in Section 2, including the updated equations for general grids and grids on the reflective boundary. In Section 3, the system design and implementation by using the FPGA board DE5a-NET are described, as well as the system architecture and the functions of main components. System performance of the FPGA-based prototype machine is presented in Section 4, followed by the conclusions drawn in Section 5.

2. RENDERING ALGORITHM

The wave equations for 3-D room acoustic simulation is expressed as:

$$(\frac{\partial^2}{\partial t^2} - c^2\nabla^2)p(\mathrm{x},t) = 0 \tag{1}$$

Here, $p(\mathrm{x},t)$ is the sound pressure at time t and position x, $\mathrm{x} = (x,y,z) \in \mathcal{R}^3$ is the spatial position with coordinates being (x,y,z) in a 3-D space, c is the propagation speed of sound in air, the operator $\frac{\partial^2}{\partial t^2}$ denotes the second partial derivative with respect to time, the operator ∇^2 stands for the spatial 3-D Laplacian operator, and $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$. Then, the wave equation (1) can be described by the time domain formulation shown in Equation (2)

$$\frac{\partial^2 p(\mathrm{x},t)}{\partial t^2} = c^2(\frac{\partial^2 p(\mathrm{x},t)}{\partial x^2} + \frac{\partial^2 p(\mathrm{x},t)}{\partial y^2} + \frac{\partial^2 p(\mathrm{x},t)}{\partial z^2}) \tag{2}$$

In order to discretize Equation (2) at time and spatial domains, using $P^n(i,j,k) \cong p(i\Delta x, j\Delta y, k\Delta z, nT)$ as an approximation to $p(\mathrm{x},t)$ at time $t = nT$ and position $\mathrm{x} = (i,j,k)$, where T is the time step, n is the number of time steps, and $\Delta x, \Delta y, \Delta z$ are the Cartesian grid spacing in *x*, *y*, and *z* axes, respectively. Then the temporal and spatial difference operators can be defined as

$$\frac{\partial^2 p(\mathrm{x},t)}{\partial t^2} = \frac{P^{n+1}(i,j,k) - 2P^n(i,j,k) + P^{n-1}(i,j,k)}{T^2}$$

$$\frac{\partial^2 p(\mathrm{x},t)}{\partial x^2} = \frac{P^n(i+1,j,k) - 2P^n(i,j,k) + P^n(i-1,j,k)}{\Delta x^2} \tag{3}$$

$$\frac{\partial^2 p(\mathrm{x},t)}{\partial y^2} = \frac{P^n(i,j+1,k) - 2P^n(i,j,k) + P^n(i,j-1,k)}{\Delta y^2}$$

$$\frac{\partial^2 p(\mathrm{x},t)}{\partial z^2} = \frac{P^n(i,j,k+1) - 2P^n(i,j,k) + P^n(i,j,k-1)}{\Delta z^2}$$

In a cubical grid, letting $\Delta x = \Delta y = \Delta z = \Delta l$ and inserting Equation (3) in Equation (2), Equation (2) is discretized and Equation (4) is yielded.

$$P^{n+1}(i,j,k) = \chi^2[P^n(i+1,j,k) + P^n(i-1,j,k) + P^n(i,j+1,k) + P^n(i,j-1,k) + P^n(i,j,k+1) + P^n(i,j,k-1)] + (2-6\chi^2)P^n(i,j,k) - P^{n-1}(i,j,k) \tag{4}$$

where $\chi = \frac{cT}{\Delta l}$ is the Courant number, and cannot be larger than $\frac{1}{\sqrt{3}}$ because of numerical stability in a 3-D sound space. From Equation (4), to compute the sound pressure of a grid needs three multiplications, six additions, and one subtraction. In order to reduce the multiplication operations, which need more clock cycles and hardware resources, χ is assumed to be $\frac{1}{2}$, and Equation (4) is then rewritten as [18][19][23]

$$P^{n+1}(i,j,k) = \frac{1}{4}[P^n(i+1,j,k) + P^n(i-1,j,k) + P^n(i,j+1,k) + P^n(i,j-1,k) + P^n(i,j,k+1) + P^n(i,j,k-1) + 2P^n(i,j,k)] - P^{n-1}(i,j,k) \tag{5}$$

In Equation (5), two multiplication operations can be replaced by right and left shift operations, which are easily implemented by hardware and consume less clock cycles.

2.1. Reflective boundary

In realistic room acoustics, the boundary conditions should be considered to model the reflection and absorption from walls. In this study, the reflective boundary is concerned. A reflective boundary can be modelled as a locally reacting surface by assuming that wave does not propagate along with the boundary surface, and the acoustical behavior is only affected by the sound pressure and particle velocity perpendicular to the boundary surface. If a sound wave travels in a positive axis (x,y,z) direction, the boundary impedance Z is denoted by the sound pressure and the particle vibration through Equation (6) [24].

$$Z = \frac{P}{U} \tag{6}$$

Here, U is the particle velocity component perpendicular to the boundary, and P is the sound pressure. For a boundary perpendicular to an axis, the momentum conservation equation of wave propagation is

$$\nabla P + \rho\frac{\partial U}{\partial t} = 0 \tag{7}$$

where ρ is the air density. Differentiating both sides of Equation (6) with respect to t and inserting Equation (7), the boundary conditions are obtained in terms of sound pressure [19][23][25].

$$\frac{\partial P}{\partial t} = -c\xi\nabla P \tag{8}$$

where $\xi = \frac{Z}{\rho c}$ is the normalized boundary impedance. For a cubical sound space, boundary grids are classified into interior grids of a boundary, edges, and corners according to their positions. Different formulas are applied to update their sound pressures. For example, for the interior grids of right boundary, wave travels along the positive x axis direction, and Equation (9) is derived by discretizing Equation (8).

$$\frac{P^{n+1}(i,j,k)-P^{n-1}(i,j,k)}{2T} = -c\xi\frac{P^{n}(i+1,j,k)-P^{n}(i-1,j,k)}{2\Delta x} \tag{9}$$

Rearranging the terms in Equation (9) and introducing the parameter χ, Equation (10) is obtained to represent a virtual grid outside the sound space.

$$P^n(i+1,j,k) = P^n(i-1,j,k) + \frac{1}{\chi\xi}[P^{n-1}(i,j,k) - P^{n+1}(i,j,k)] \tag{10}$$

Substituting $P^n(i+1,j,k)$ in Equation (4) with Equation (10), then

$$P^{n+1}(i,j,k) = \Big[\chi^2\big(2P^n(i-1,j,k) + P^n(i,j-1,k) + P^n(i,j+1,k) + P^n(i,j,k-1) + P^n(i,j,k+1)\big) + (2-6\chi^2)P^n(i,j,k) + \Big(\frac{\chi}{\xi}-1\Big)P^{n-1}(i,j,k)\Big] / \Big(\frac{\chi}{\xi}+1\Big) \tag{11}$$

By introducing the reflection factor R as $\frac{(\xi-1)}{(\xi+1)}$ and χ being $\frac{1}{2}$, Equation (11) is changed to Equation (12) [19][23], which is applied to update the sound pressure of the interior grids of right boundary.

$$P^{n+1}(i,j,k) = \frac{1+R}{2(3+R)}[2P^n(i-1,j,k) + P^n(i,j-1,k) + P^n(i,j+1,k) + P^n(i,j,k-1) + P^n(i,j,k+1) + 2P^n(i,j,k)] - \frac{3R+1}{3+R}P^{n-1}(i,j,k) \tag{12}$$

Equation (12) consists of the sum of the sound pressures of a grid and its neighbor grids at the time step n, and its sound pressure at the time step $n-1$. Compared with Equation (5), except for the multiplicands, Equation (12) only replaces the sound pressure of the virtual grid $P^n(i+1,j,k)$ by the sound pressure of the neighbor grid $P^n(i-1,j,k)$ in the summation. Moreover, for the interior grids of other boundaries, the multiplicands are same except for the sound pressure of the substituted virtual grid in the summation. For example, the updated equation for an interior grid of left boundary is obtained by substituting the sound pressure of the virtual grid $P^n(i-1,j,k)$ with the sound pressure of the direct neighbor grid $P^n(i+1,j,k)$ in Equation (12). Hence, the summation is changed as $2\,P^n(i+1,j,k) + P^n(i,j+1,k) + P^n(i,j-1,k) + P^n(i,j,k+1) + P^n(i,j,k-1) + 2\,P^n(i,j,k)$. The similar derivation procedure can be applied to edges and corners by using different boundary conditions.

Equations (5) and (12) show that to compute sound pressure of a grid needs the sound pressures of its own and neighbors at previous time steps. For different types of grids, the updated equations have similar formats except for the multiplicands for the summation and $P^{n-1}(i,j,k)$, respectively. From Equations (5) and (12), a uniform updated Equation (13) can be derived.

$$P^{(n+1)}(i,j,k) = D1 * [P^n(i-1,j,k) + P^n(i+1,j,k) + P^n(i,j-1,k) + P^n(i,j+1,k) + P^n(i,j,k-1) + P^n(i,j,k+1) + 2P^n(i,j,k)] - D2 * P^{(n-1)}(i,j,k) \tag{13}$$

As shown in Table 1, the parameters D1, D2, and items in the summation in Equation (13) are associated with the position of a grid. For grids on boundaries, the sound pressures of the virtual grids are replaced by those of the related direct neighbor grids.

Table 1: *Parameters*

Grid position	D1	D2
General	$\frac{1}{4}$	1
Interior	$\frac{\mathrm{R}+1}{2\ (\mathrm{R}+3)}$	$\frac{3\mathrm{R}+1}{\mathrm{R}+3}$
Edge	$\frac{\mathrm{R}+1}{8}$	R
Corner	$\frac{\mathrm{R}+1}{2\ (5-\mathrm{R})}$	$\frac{5\mathrm{R}-1}{5-\mathrm{R}}$

3. SYSTEM DESIGN AND IMPLEMENTATION

3.1. Design flow

The accelerator is designed using OpenCL, which is a programming language for high-level synthesis of FPGA. As shown in Figure 1, the OpenCL design flow consists of the host and kernel, and the related codes are compiled separately. The accelerator is designed as kernels using OpenCL, which are then compiled to an intermediate representation (LLVM IR), optimized, and converted to the Verilog files by the Intel FPGA SDK for OpenCL. The EDA tool Quartus Prime Pro is called to perform synthesis, placement and routing to generate the FPGA bitstream, which is finally downloaded in the FPGA and executed. The host is developed using C or C++ programming language. It initializes the kernels, maintains the computation flow, and charges data exchange between the host machine and FPGA board. The system drivers and controllers for I/O, such as PCIe bus and DDR memory controllers, are generated and integrated in the system by the Intel FPGA SDK for OpenCL automatically. Therefore, user mainly focuses on designing the kernels. The system design becomes much easier, and the development period is shortened significantly.

3.2. System design

Sound field rendering is memory-intensive. It is impossible to store all data in the on-chip memories inside FPGA as the space volume is increased even if the size of the on-chip memories inside current FPGAs has been increased significantly. Instead, the external large DDR memory is adopted in this research. To improve system performance, the overhead of data access to external memory should be shortened. In the system, a sliding window-based data buffering system is introduced to speed up data access between the rendering engine and the on-board external memory.

In the sliding window-based data buffering, the blocking technique is applied to reduce the buffer size and memory bandwidth demand, in which a large sound space with M × N × K grids is divided into sub-cubes with each having N_x × N_y × N_z grids. The sub-cubes are read into the system along the *Z* direction, and computations are carried out. Inside a sub-cube, sound pressures of grids on two consecutive *x-y* planes are kept by buffers. Therefore, the buffer size is reduced from M × N to N_x × N_y. The data buffering system is implemented by using the high-speed and high-bandwidth on-chip block RAMs inside FPGA. However, the problem of the blocking technique is that the halo exists between two sub-cubes, which will incur additional computations.

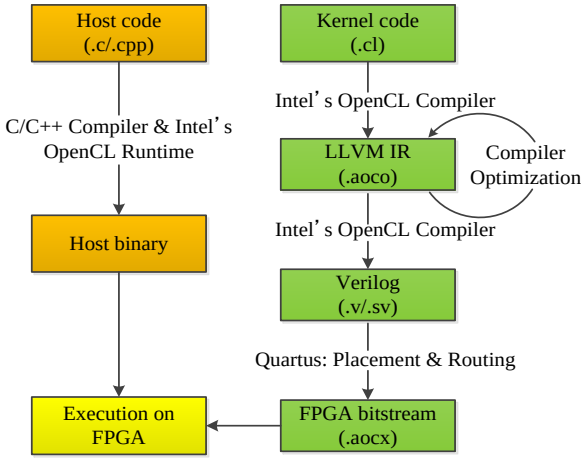


Figure 1: *Design flow*

The system diagram is illustrated in Figure 2, which consists of system controller, three buffers (shift_register_p1, shift_register_p2, and shift_register_posi), computing units, and output controller. The incident data and position flags of grids are firstly written into the on-board DDR memory from the host machine before computation is started. The functions of each module are described as follows.

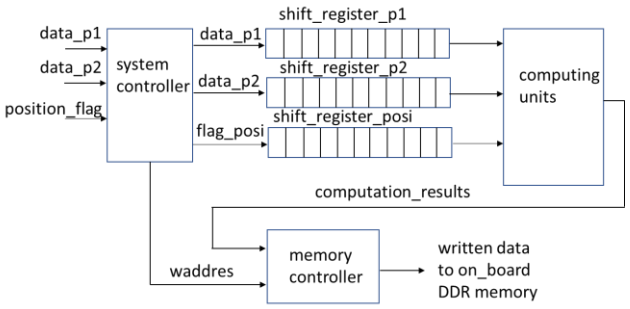


Figure 2: *System diagram*

- system controller. The system controller maintains computation flow and generates control signals according to the computing flow. It reads data and position flags from the on-board DDR memory, and writes them to the buffers shift_register_p1, shift_register_p2, and shift_register_posi, respectively. After computation is completed, it also generates the writing addresses of the grids to store the computation results back to the DDR memory. At each time step, sound pressure of the observation grid is stored in the on-board DDR memory. It is not written back to the host machine until computations at all time steps are finished.
- shift_register_p1, shift_register_p2, and shift_register_posi. The shift_register_p1, shift_register_p2, and shift_register_posi are three buffers to store the data involved in computations and their position flags, respectively. Before computation is started, data in the two continuous *x-y* planes of the sub-cube are streamed into the buffers. The data at the time step *n-1* are stored in the buffer shift_register_p1 while the data at the time step n-2 are kept by the buffer shift_register_p2. And the corresponding position flags of grids are streamed into the shift_register_posi. If the sub-cube contains N_x × N_y × N_z grids and *i* grids are computed concurrently, the depth of the shift_register_p1, shift_register_p2, and shift_register_posi is N_x ∗ N_y + *i*. Along with computation, the three buffers are shifted right by *i* data, and another *i* new data and their position flags are streamed into the buffers at each clock cycle. Such procedure is repeated until sound pressures of all grids inside a sub-cube are computed, and then computation is moved to the next sub-cube. The three buffers are implemented by the high-width and high-speed block memory inside FPGA. In the current design, the sub-cube has 256 × 256 × 256 grids and *i* is 16.
- computing units. The computing units is the arithmetic unit to calculate sound pressures of *i* grids concurrently according to the input sound pressures at previous time steps and location indicators (position_flag). The location indicator is used to select the multiplicands D1 and D2 in Equation (13). As shown in Figure 3, a uniform computing unit is designed based on Equation (13), which consists of a 7-input adder, a subtractor, two multipliers, and four multiplexers [19][23]. In Figure 3, the multipliers are used for boundary grids while they are replaced by the right and left shifters for general grids. Two multiplexers are applied to select the multiplicands D1 and D2 in accordance to the location indicator of a grid. At each clock cycle, sound pressures of *16* grids are computed in parallel.
- memory controller. The memory controller stores the computation results to the external on-board DDR memory.

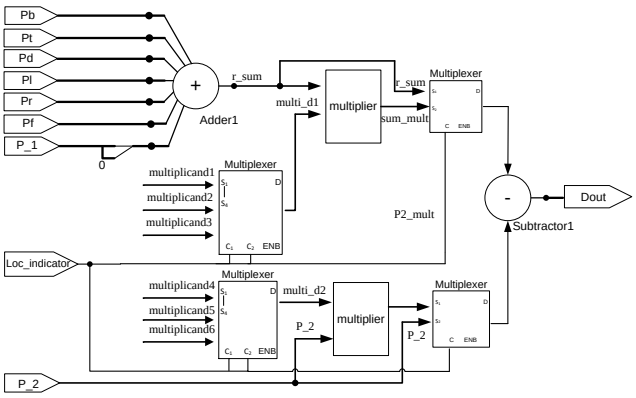


Figure 3: *Computing unit*

3.3. System implementation

The accelerator is implemented by using the FPGA board DE5a-NET from the Terasic Company [26], which contains an Intel Arria 10 FPGA and 8 GB on-board DDR memory arranged in two independent channels. As shown in Figure 4, the incident data and

position information of all grids are firstly written into the on-board memory from the host machine through the PCIe bus. The Rendering Engine reads an incident datum from the memory, calculates the sound pressures of all grids, and stores the computation results back to the DDR memory. Then another incident datum is read into, and computations are repeated. This procedure is iterated until all incident data are read into the rendering engine, and sound pressures of all grids are obtained. Finally, the sound pressure at the observation point will be written back to the host machine. The two independent DDR memory will be updated in turn.

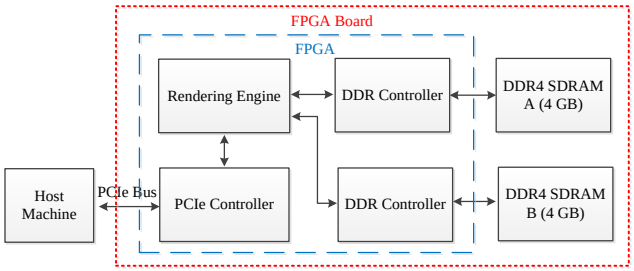


Figure 4: *System diagram*

When the accelerator is implemented by the FPGA board DE5a-net, the hardware resource utilization is shown in Table 2. From Table 2, the current design consumes less half of the available hardware resources inside the FPGA chip Arria 10, and the system performance can be improved further by using more computation kernels to work in parallel.

Table 2: *Hardware resource utilization*

Logic utilization	DSP blocks	RAM blocks	Clock frequency
70701 (17%)	152 (10%)	891 (33%)	267 MHz

4. PERFORMANCE EVALUATION

To estimate the performance of the proposed accelerator, the rendering time in the sound spaces with grids being 128 ×128 × 128, 256 × 256 × 256, 510 × 510 × 510, and 764 × 764 × 764, is measured. The reflection coefficient of boundaries is 0.95, and the time steps are 1000. As a comparison, the same system is developed using C++ programming language, parallelized using OpenMP, and executed on a PC with 128 GB DDR4 memory and an Intel i7-7820X processor, which has eight cores running at 3.6 GHz. The reference C++ codes are compiled by the gcc compiler with the soption -O3 and -fopenmp to use all eight cores in the PC. The simulation and execution environment are shown in Table 3. As shown in Table 3, the memory size of the FPGA-based system, including the external and on-chip memories, is much smaller than that of the PC in software simulation, and the clock frequency of FPGA is about 267 MHz while the PC runs at 3.6 GHz.

4.1. Rendering time

Figure 5 shows the rendering time taken by the software simulations on the PC and the FPGA-based system in the case of different sound space volumes. In Figure 5, the sub-cube is with 128 × 128 × 128 grids in the case of sound space volume being 128 × 128 × 128 grids while it is with 256 × 256 × 256 grids in other cases. The number of grids computed in parallel is 16. As shown in Figure 5, the rendering time taken by the FPGA-based accelerator is almost half of that consumed by the software simulations on PC. In addition, due to the effect of the existing halo, the simulated area becomes a little smaller.

Table 3: *Technology specification*

	FPGA	Software simulation
Model	Arria 10 GX 10AX115N2F45E1SG	i7-7820X
Cores	1518 DSP blocks	8 cores
Clock frequency	About 267 MHz	3.6 GHz
On-chip memory	6.625 MB block RAMs	L1 cache: 256 KB L2 cache: 8 MB L3 cache: 11 MB
External memory	8 GB	128 GB
OS	CentOS 7.0	CentOS 7.0
Programming language	OpenCL	C
Compiler	Intel FPGA SDK for OpenCL 17.1	gcc 4.8.5
Fabrication	20 nm	14 nm

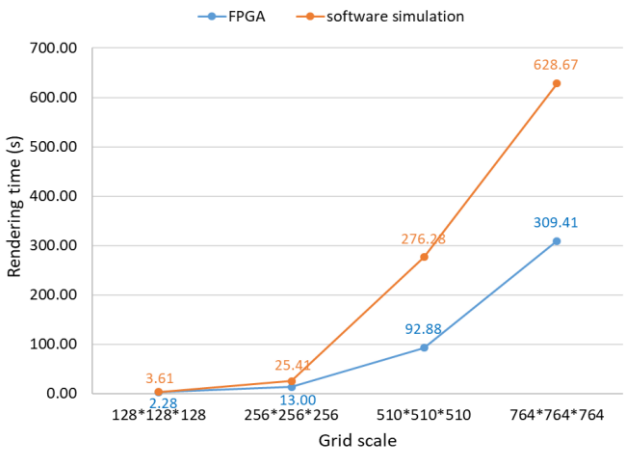


Figure 5: *Rendering time*

4.2. Computation throughput

The computation throughput denotes the updated speed of grids at each time step, and is calculated by using the following formula.

$$D_{throughput} = \frac{N_{grid}}{t_{per\_timestep}} \tag{14}$$

where N_{grid} is the number of grids in a sound space, and $t_{per\_timestep}$ is the rendering time at each time step. Figure 6 presents the computation throughput in the case of different grid scales in the FPGA-based accelerator and the software simulations on the PC. Figure 6 indicates that the proposed accelerator almost doubles the computation throughput of the software simulations, especially in the case of large sound spaces.

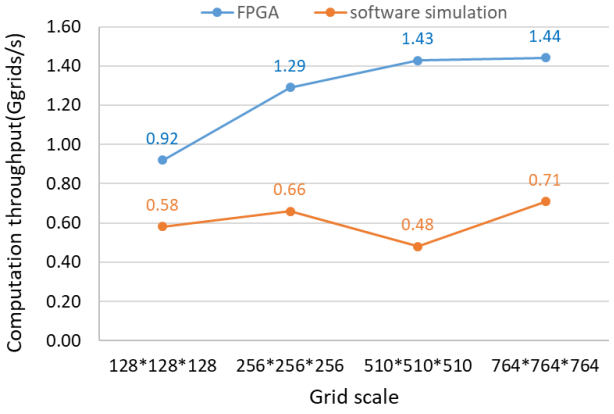


Figure 6: *Computation throughput*

5. CONCLUSIONS

Sound field rendering is computation-intensive and memory-intensive. FPGAs provide an alternative solution to sound field rendering, especially for real-time applications because the I/O interfaces are easily tailored according to applications. In this study, a FPGA-based accelerator for sound field rendering is developed using high-level synthesis in FPGA, in which the sliding window-based data buffering scheme is applied to reduce the demand of memory bandwidth. Although the FPGA-based accelerator runs at 1/13 (0.267/3.6) of clock frequency of the PC in software simulations, and the memory size of the FPGA board is about 1/16 (8/128) of that on the PC, the FPGA-based accelerator doubles the performance of the software simulations carried out on the PC. However, Figure 5 indicates that the rendering time at a time step is still long in the accelerator, which results in low sampling rate at the output rendered results. Hence, the current design is not suitable for real-time applications. From Table 2, we can find that the hardware resource utilization is low, and more computing units may be involved in calculation. Then, more grids may be computed concurrently to shorten the computation time at a time step. The related system is under development.

6. ACKNOWLEDGMENTS

Thanks to the Intel for the donation of the FPGA board DE5a-NET and the related software tools through University Program. This work was supported by the I-O DATA Foundation and JSPS KAKENHI Grant Number JP19K12092.

7. REFERENCES

[1] D. Botteldooren, “Acoustical Finite-difference Time-domain Simulation in a Quasi-Cartesian Grid,” *J. Acoust. Soc. Am.*, vol. 95, pp. 2313-2319, 1994.

[2] D. Botteldooren, “Finite-difference Time-domain Simulation of Low-frequency Room Acoustic Problems,” *J. Acoust. Soc. Am.*, vol. 98, pp. 3302-3308, 1995.

[3] O. Chiba, T. Kashiwa, H. Shimoda, S. Kagami, I. Fukai, “Analysis of Sound Fields in Three Dimensional Space by the Time-dependent Finite-difference Method based on the Leap Frog Algorithm,” *J. Acoust. Soc. Jpn.*, vol. 49, pp. 551-562, 1993.

[4] L. Savioja, and V. Valimaki, “Interpolated Rectangular 3-D Digital Waveguide Mesh Algorithms with Frequency Warping,” *IEEE Trans. Speech Audio Process.*, vol. 11, pp. 783-790, 2003.

[5] G. R. Campos, and D. M. Howard, “On the Computational Efficiency of Different Waveguide Mesh Topologies for Room Acoustic Simulation,” *IEEE Trans. Speech Audio Process.*, vol. 13, pp. 1063-1072, 2005.

[6] D. Murphy, A. Kelloniemi, J. Mullen, and S. Shelley, “Acoustic Modeling Using the Digital Waveguide Mesh,” *IEEE Signal Process. Mag.*, vol. 24, pp. 55-66, 2007.

[7] K. Kowalczyk and M. Walstijn, “Room Acoustics Simulation Using 3-D Compact Explicit FDTD Schemes,” *IEEE Trans. Audio Speech Lang. Process.*, vol. 19, pp. 34-46, 2011.

[8] J. Mourik, and D. Murphy, “Explicit Higher-order FDTD Schemes for 3D Room Acoustic Simulation,” *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 22, pp. 2003-2011, 2014.

[9] B. Hamilton, and S. Bilbao, “Fourth-order and optimised finite difference schemes for the 2-D wave equation,” In *Proc. Digital Audio Effects (DAFx-13)*, Maynooth, Ireland, Sept. 2013, pp. 2-6.

[10] B. Hamilton, S. Bilbao, and C. J. Webb, “Revisiting implicit finite difference schemes for 3D room acoustics simulations on GPU,” In *Proc. Digital Audio Effects (DAFx-14)*, Erlangen, Germany, Sept. 2014, pp. 41-48.

[11] B. Hamilton, and S. Bilbao, “FDTD Methods for 3-D Room Acoustics Simulation with High-order Accuracy in Space and Time,” *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 25, pp. 2112-2124, 2017.

[12] V. Valimaki, J. D. Parker, L. Savioja, J. O. Smith, and J. S. Abel, “Fifty Years of Artificial Reverberation,” *IEEE Trans. Audio Speech Lang. Process.*, vol. 20, no. 5, pp. 1421-1448, 2012.

[13] T. Ishii, T. Tsuchiya, and K. Okubo, “Three-dimensional Sound field Analysis Using Compact Explicit Finite Difference Time Domain Method with Graphics Processing Unit Cluster System,” *Jpn. J. Appl. Phys.*, vol. 52, pp. 07HC11, 2013.

[14] T. Tsuchiya, “Three-dimensional Sound Field Rendering with Digital Boundary Condition Using Graphics Processing Unit,” *Jpn. J. Appl. Phys.*, vol. 49, pp. 07HC10, 2010.

[15] C. Spa, A. Rey, and E. Hernandez, “A GPU Implementation of an Explicit Compact FDTD Algorithm with a Digital Impedance Filter for Room Acoustics Applications,” *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 23, no. 8, pp. 1368–1380, 2015.

[16] M. Tanaka, T. Tsuchiya, and K. Okubo, “Two-dimensional Numerical Analysis of Nonlinear Sound Wave Propagation using Constrained Interpolation Profile Method Including

Nonlinear Effect in Advection Equation," *Jpn. J. Appl. Phys.*, vol. 50, pp. 07HE17, 2011.

[17] Y. Y. Tan, Y. Inoguchi, E. Sugawara, M. Otani, Y. Iwaya, Y. Sato, H. Matsuoka, and T. Tsuchiya, "A Real-time Sound Field Renderer Based on Digital Huygens' Model," *J. Sound Vib.*, vol. 330, pp. 4302–4312, 2011.

[18] Y. Y. Tan, Y. Inoguchi, Y. Sato, M. Otani, Y. Iwaya, H. Matsuoka, and T. Tsuchiya, "A Hardware-oriented Finite-difference Time-domain Algorithm for Sound Field Rendering," *Jpn. J. Appl. Phys.*, vol. 52, pp. 07HC03, 2013.

[19] Y. Y. Tan, Y. Inoguchi, Y. Sato, M. Otani, Y. Iwaya, H. Matsuoka, and T. Tsuchiya, "A Real-time Sound Rendering System Based on the Finite-difference Time-domain Algorithm," *Jpn. J. Appl. Phys.*, vol. 53, pp. 07KC14, 2014.

[20] Y. Y. Tan, Y. Inoguchi, Y. Sato, M. Otani, Y. Iwaya, H. Matsuoka, and T. Tsuchiya, "Analysis of sound field distribution for room acoustics: from the point of view of hardware implementation," In *Proc. Digital Audio Effects (DAFx-12)*, York, UK, Sept. 2012, pp. 93-96.

[21] Y. Y. Tan, Y. Inoguchi, Y. Sato, M. Otani, Y. Iwaya, H. Matsuoka, and T. Tsuchiya, "A FPGA implementation of the two-dimensional digital Huygens' model," In Proc. Field Program. Technol. (FPT 2010), Beijing, China, Dec. 2010, pp. 304-307.

[22] Y. Inoguchi, Y. Y. Tan, Y. Sato, M. Otani, Y. Iwaya, H. Matsuoka, and T. Tsuchiya, "DHM and FDTD based hardware sound field simulation acceleration," In *Proc. Digital Audio Effects (DAFx-11)*, Paris, France, Sept. 2011, pp. 69-72.

[23] Y.Y. Tan, Y. Inoguchi, M. Otani, Y. Iwaya, and T. Tsuchiya, "A Real-Time Sound Field Rendering Processor", *Applied Sciences*, vol. 8, no. 35, 2018.

[24] H. Kuttruff, *Room Acoustics*, Taylor & Francis: New York, NY, USA, 2009.

[25] K. Kowalczyk, and M. Walstijn, "Formulation of Locally Reacting Surfaces in FDTD/K-DWM Modelling of Acoustic Spaces," *Acta Acust. United Acust.*, vol. 94, pp. 891-906, 2008.

[26] https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=231&No=970, Accessed June 18, 2019.

DATA AUGMENTATION FOR INSTRUMENT CLASSIFICATION ROBUST TO AUDIO EFFECTS

António Ramires

Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
antonio.ramires@upf.edu

Xavier Serra

Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
xavier.serra@upf.edu

ABSTRACT

Reusing recorded sounds (sampling) is a key component in Electronic Music Production (EMP), which has been present since its early days and is at the core of genres like hip-hop or jungle. Commercial and non-commercial services allow users to obtain collections of sounds (sample packs) to reuse in their compositions. Automatic classification of one-shot instrumental sounds allows automatically categorising the sounds contained in these collections, allowing easier navigation and better characterisation.

Automatic instrument classification has mostly targeted the classification of unprocessed isolated instrumental sounds or detecting predominant instruments in mixed music tracks. For this classification to be useful in audio databases for EMP, it has to be robust to the audio effects applied to unprocessed sounds.

In this paper we evaluate how a state of the art model trained with a large dataset of one-shot instrumental sounds performs when classifying instruments processed with audio effects. In order to evaluate the robustness of the model, we use data augmentation with audio effects and evaluate how each effect influences the classification accuracy.

1. INTRODUCTION

The repurposing of audio material, also known as sampling, has been a key component in Electronic Music Production (EMP) since its early days and became a practice which had a major influence in a large variety of musical genres. The availability of software such as Digital Audio Workstations, together with the audio sharing possibilities offered with the internet and cloud storage technologies, led to a variety of online audio sharing or sample library platforms. In order to allow for easier sample navigation, commercial databases such as sounds.com[1] or Loopcloud[2] rely on expert annotation to classify and characterise the content they provide. In the case of collaborative databases such as Freesound [1] the navigation and characterisation of the sounds is based on unrestricted textual descriptions and tags of the sounds provided by users. This leads to a search based on noisy labels which different members use to characterise the same type of sounds.

Automatically classifying one-shot instrumental sounds in unstructured large audio databases provides an intuitive way of navigating them, and a better characterisation the sounds contained. For databases where the annotation of the sounds is done manually, it can be a way to simplify the job of the annotator, by providing suggested annotations or, if the system is reliable enough, only presenting sounds with low classification confidence.

The automatic classification of one-shot instrumental sounds remain an open research topic for music information retrieval (MIR). While the research on this field has been mostly performed on clean and unprocessed sounds, the sounds provided by EMP databases may also contain "production-ready" sounds, with audio effects applied on them. Therefore, in order for this automatic classification to be reliable for EMP sample databases, it has to be robust to the types of audio effects applied to these instruments. In our study, we evaluate the robustness of a state of the art automatic classification method for sounds with audio effects, and analyse how data augmentation can be used to improve classification accuracy.

2. RELATED WORK

Automatic instrument classification can be split into two related tasks with a similar goal. The first is the identification of instruments in single instrument recordings (which can be isolated or overlapping notes) while the second is the recognition of the predominant instrument in a mixture of sounds. A thorough description of this task and an overview of the early methodologies used is presented in [2]. These early approaches used two modules for classification, one for extracting and selecting handcrafted features (e.g. Mel Frequency Cepstral Coefficients, spectral centroid, roll-off, and flux) and another for classification (e.g. k-nearest neighbours, support vector machines or hidden Markov models). Datasets used for the evaluation and training of these algorithms included RWC [3] or the University of Iowa Musical Instrument Samples[3]. While these datasets are small (RWC has 50 instruments) they proved to be good for classification using handcrafted features. New datasets such as IRMAS [4] for predominant instrument classification and GoodSounds [5] with single instrument recordings have been created and provided sufficient data for deep learning approaches to be able to surpass more traditional machine learning approaches. A review of the evolution of traditional machine learning and deep learning approaches for instrument classification is presented in [6]. While the performance of traditional machine learning methods rely on developing handcrafted features, deep learning methods learn high-level representations from data using a general-purpose learning procedure, eliminating the need of expert feature extraction [7]. However, the success of these approaches is highly dependent on both the type

[1] https://sounds.com/

[2] https://www.loopcloud.net/

[3] http://theremin.music.uiowa.edu/MIS.html

and amount of data they are provided [8].

Recent work has shown the effectiveness of using Convolutional Neural Networks (CNNs) for instrument classification [9–12]. CNNs can be seen as trainable feature extractors, where kernels (or filters) with trainable parameters are convolved over an input, being able to capture local spatial and temporal characteristics. This architecture has been applied with great success to the detection, segmentation and recognition of objects and regions in images [7]. In the audio domain, when raw audio or spectograms are given, CNNs are able to learn and identify local spectro-temporal patterns relevant to the task to which they are applied. When utilized for MIR tasks, CNNs have outperformed previous state of the art approaches for various tasks [10, 13]. For automatic instrument classification, the state of the art approaches use CNNs trained on different representations of the input, such as raw audio [11], spectograms together with multiresolution recurrence plots [12] and log mel-frequency spectograms [9, 10]. In [9], CNNs were tailored towards learning timbre representations in log mel-frequency spectograms through the use of vertical filters instead of the commonly used square filters. For instrument classification, this approach displays a close to the state of the art [10] accuracy on the IRMAS dataset [4], while reducing the number of trainable parameters by approximately 23 times, on the single-layer proposed model.

Within the context of NSynth [14], a new high-quality dataset of one shot instrumental notes was presented, largely surpassing the size of the previous datasets, containing 305979 musical notes with unique pitch, timbre and envelope. The sounds were collected from 1006 instruments from commercial sample libraries and are annotated based on their source (acoustic, electronic or synthetic), instrument family and sonic qualities. The instrument families used in the annotation are bass, brass, flute, guitar, keyboard, mallet, organ, reed, string, synth lead and vocal. The dataset is available online[4] and provides a good basis for training and evaluating one shot instrumental sound classifiers. This dataset is already split in training, validation and test set, where the instruments present in the training set do not overlap with the ones present in validation and test sets. However, to the best of our knowledge, no methods for instrument classification have so far been evaluated on this dataset.

In order to increase the generalisation of a model further than the data provided to it, one possible approach is to use data augmentation. This approach can be described as applying deformations to a collection of training samples, in a way that the correct labels can still be deduced [15]. In computer vision, transforming images by cropping, rotation, reflection or scaling are commonly used techniques for data augmentation. In the audio domain, an intuitive and practical transformation is applying audio effects to the original training audio files. Transformations such as time-stretching, pitch-shifting, dynamic range compression and adding background noise have been applied with success to environmental sound classification, for overcoming the data scarcity problems [16]. In [17], artificial reverberation was applied to speech recordings, so as to create a speech recognition system robust to reverberant speech. For instrument recognition, the same set of effects used in [16] was applied with success in [15]. We believe that the use of audio effects typically used in EMP such as echo, reverb, chorus, saturation, heavy distortion or flanger can lead to a useful augmentation, as well as to an increase in robustness in instrument classification scenarios where the instrument recordings have these effects applied.

[4] https://magenta.tensorflow.org/datasets/nsynth

3. METHODOLOGY

In our study we will conduct two experiments. First, we will try to understand how augmenting a dataset with specific effects can improve instrument classification and secondly, we will see if this augmentation can improve the robustness of a model to the selected effect.

To investigate this, we process the training, validation and test sets of the NSynth [14] dataset with audio effects. A state of the art deep learning architecture for instrument classification [9] is then trained with the original training set, and appended with each of the augmented datasets for each effect. We use the model trained with the original training set as a baseline and compare how the models trained with augmented versions perform on the original test and on the augmented versions of it for each effect. The code for the experiments and evaluation is available in a public GitHub repository[5].

3.1. Data Augmentation and Pre-Processing

The audio effects for the augmentation were applied directly to the audio files present in the training, validation splits of the NSynth dataset [14]. For the augmentation procedure, we used a pitch-shifting effect present in the LibROSA[6] library and audio effects in the form of VST audio plugins. For the augmentation which used audio plugins, the effects were applied directly to the audio signals using the Mrs. Watson[7] command-line audio plugin host. This command line tool was designed for automating audio processing tasks and allows the loading of an input sound file, processing it using a VST audio effect and saving the processed sound. In order to maintain transparency and reproducibility of this study only VST plugins which are freely distributed online were selected. The parameters used in the augmentation procedure were the ones set in the factory default preset for each audio plugin, except for those whose default preset did not alter significantly the sound.

The audio effects used were the following:

- **Heavy distortion:** A Bitcrusher audio effect which produces distortion through the reduction of the sampling rate and the bit depth of the input sound was used in the training set. The VST plugin used for augmenting the training set was the TAL-Bitcrusher[8]. For the test and validation set, we used Camel Audio's CamelCrusher[9] plugin which provides distortion using tube overdrive emulation combined with a compressor.
- **Saturation:** For this effect, tube saturation and amplifier simulation plugins were used. The audio effect creates harmonics in the signal, replicating the saturation effect from a valve- or vacuum-tube amplifier [18]. For this augmentation we focused on a subtle saturation which did not create noticeable distortion. The plugin used in the training set was the TAL-Tube[8], while for the validation and test set

[5] https://github.com/aframires/instrument-classifier/

[6] https://librosa.github.io/librosa/

[7] https://github.com/teragonaudio/MrsWatson

[8] https://tal-software.com/products/tal-effects

[9] https://www.kvraudio.com/product/camelcrusher-

Shattered Glass Audio's Ace[10] replica of a 1950s all tube amplifier was used.

- **Reverb:** To create a reverberation effect, the TAL-Reverb-4 plugin[11] was used in the test set. This effect replicates the artificial reverb obtained in a plate reverb unit. For the validation and test set we used OrilRiver[12] algorithmic reverb, which models the reverb provided by room acoustics. The default preset for this plugin mimics the reverb present in a small room.
- **Echo:** A delay effect with long decay and with a big delay time (more than 50ms) [18] was used to create an echo effect. We used the TAL-Dub-2[13] VST plugin in the training set and soundhack's ++delay[14] validation and test set. For this last plugin, we adapted the factory default preset, changing the delay time to 181.7 ms and the feedback parameter to 50%, so that the echo effect was more noticeable.
- **Flanger:** For this delay effect, the input audio is summed with a delayed version of it, creating a comb filter effect. The time of the delay is short (less than 15 ms) and is varied with a low frequency oscillator [18, 19]. Flanger effects can also have a feedback parameter, where the output of the delay line is routed back to its input. For the training set, the VST plugin used was the TAL-Flanger[8], while for the test and validation sets we used Blue Cat's Flanger[15], which mimics a vintage flanger effect.
- **Chorus:** The chorus effect simulates the timing and pitch variations present when several individual sounds with similar pitch and timbre play in unison [19]. The implementation of this effect is similar to the flanger. The chorus uses longer delay times (around 30 ms), a larger number of voices (more than one) and normally does not contain the feedback parameter [18, 19]. The VST effect used in the training set was the TAL-Chorus-LX[16] which tries to emulate the chorus module present in the Juno 60 synthesizer. For the test and validation sets, we used Blue Cat's Chorus[17], which replicates a single voice vintage chorus effect.
- **Pitch shifting:** For this effect, the LibROSA Python package for musical and audio analysis was used. This library contains a function which pitch shifts the input audio. As the dataset used contains recordings of the instruments for every note in the chromatic scale in successive octaves, our approach focused on pitch-shifting in steps smaller than one semitone, similarly to what can occur in a detuned instrument. The `bins_per_octave` parameter of the pitch-shifting function was set to $72 = 12 \times 6$ while the `n_steps` parameter was set to a random value between 1 and 5 for each sound. Neither 0 or 6 were selected as possible values as it would be the same as not altering the sound or pitch-shifting it by one semitone. The intention of the random assignment in the `n_steps` is to ensure the size of this augmented dataset is equal to the size of the datasets of other effects.

The audio resulting from this augmentation step can be longer than the original unprocessed audio. In order to keep all examples with the same length, the processed audio files were trimmed, ensuring all audio samples had a fixed duration of 4 s, similar to the sounds presented in the NSynth dataset [14].

The next step in the data processing pipeline is representing each sound in a log-scaled mel-spectogram. First, a 1024-point Short-time Fourier transform (STFT) is calculated on the signal, with a 75% overlap. The magnitude of the STFT result is converted to a mel-spectogram with 80 components, covering a frequency range from 40 Hz to 7600 Hz. Finally, the logarithm of the mel-spectogram is calculated, resulting in a 80×247 log-scaled mel-spectogram for the 4 s sounds sampled at 16 kHz present in the NSynth dataset [14].

3.2. Convolutional Neural Network

The CNN architecture we chose to use in our experiment is the *single-layer* architecture proposed by Pons et al. [9] for the musical instrument classification experiment, which has an implementation available online[18]. This architecture uses vertical convolution filters in order to better model timbral characteristics present in the spectogram, achieving close to state-of-the-art results [10], using a much smaller model (23 times less trainable parameters) and a consequently lower training time.

We chose the *single-layer* architecture presented in this study and adapted it to take an input of size 80×247. This architecture contains a single but wide convolutional layer with different filters with various sizes, to capture the timbral characteristics of the input:

- 128 filters of size 5×1 and 8×1;
- 64 filters of size 5×3 and 80×3;
- 32 filters of size 5×5 and 80×5.

Batch normalisation [20] is used after the convolutional layer and the activation function used is Exponential Linear Unit [21]. Max pooling is applied in the channel dimension for learning pitch invariant representations. Finally, 50% dropout is applied to the output layer, which is a densely connected 11-way layer, with the *softmax* activation function. A graph of the model can be seen in Figure 1. For more information on this architecture and its properties see [9].

3.3. Evaluation

The training of the models used the Adam optimiser [22], with a learning rate of 0.001. In the original paper [9] the authors used Stochastic Gradient Descent (SGD) with a learning rate reduction every 5 epochs. This was shown to provide good accuracy on the IRMAS dataset. However, we chose to use Adam as an optimiser because it does not need significant tuning as SGD. Furthermore, using a variable learning rate dependent on the number of epochs could benefit the larger training datasets as is the case of the ones with augmentation. A batch size of 50 examples was used, as it was the largest batch size able to fit the memory of the available

[10] `http://www.shatteredglassaudio.com/product/103`
[11] `https://tal-software.com/products/tal-reverb-4`
[12] `https://www.kvraudio.com/product/orilriver-by-denis-tihanov`
[13] `https://tal-software.com/products/tal-dub`
[14] `http://www.soundhack.com/freeware/`
[15] `https://www.bluecataudio.com/Products/Product_Flanger/`
[16] `https://tal-software.com/products/tal-chorus-lx`
[17] `https://www.bluecataudio.com/Products/Product_Chorus`
[18] `https://github.com/Veleslavia/EUSIPCO2017`

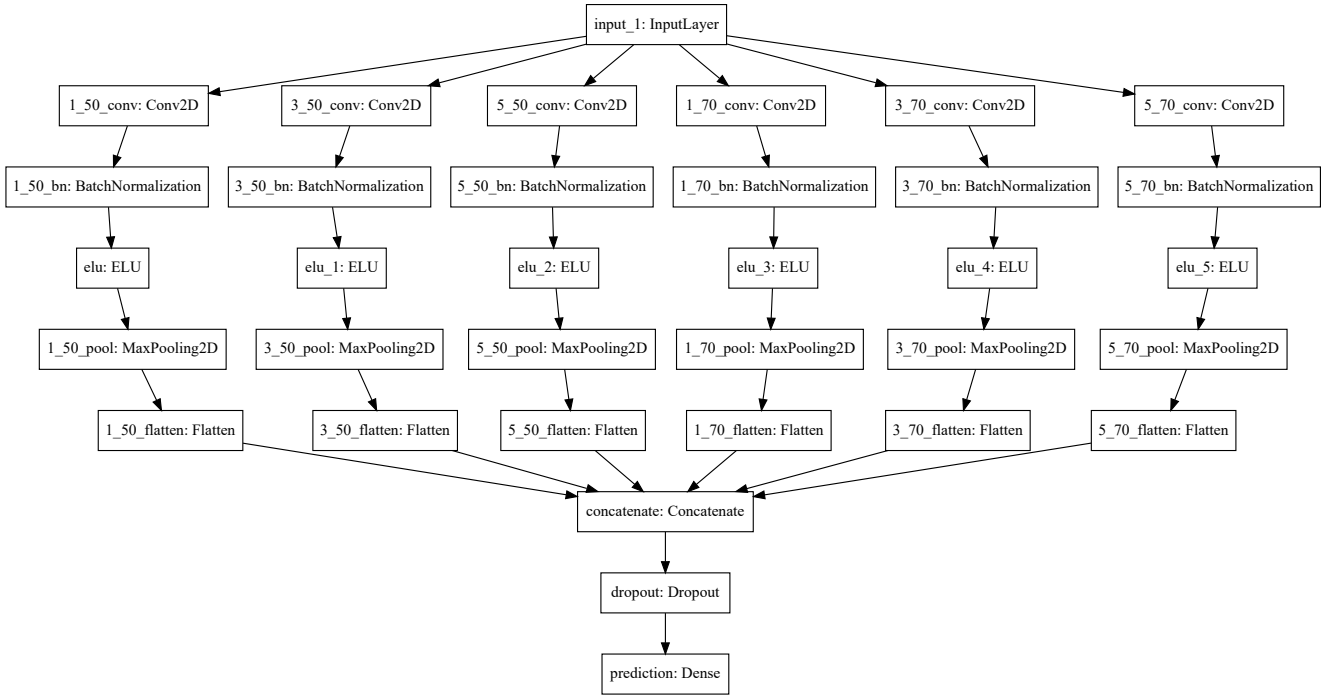


Figure 1: Single-layer CNN architecture proposed in [9]

GPUs. The loss function employed for the training was the categorical cross-entropy, as used in [9], which can be calculated as shown in Equation (1), where N represents the number of observations (examples in the training set) and $p_{model}[y_i \in C_{y_i}]$ is the predicted probability of the i^{th} observation belonging to the correct class C_{y_i}.

$$loss = -\frac{1}{N}\sum_{i=1}^{N} \log p_{model}[y_i \in C_{y_i}] \tag{1}$$

To compare the models trained with the different datasets, we used categorical accuracy as evaluation metric, described in Equation (2). A prediction is considered correct if the index of the output node with highest value is the same as the correct label.

$$\text{Categorical Accuracy} = \text{Correct predictions/N} \tag{2}$$

All the models were trained until the categorical accuracy did not improve in the validation set after 10 epochs and the model which provided the best value for the validation set was evaluated in the test set.

4. RESULTS

Two experiments were conducted in our study. We firstly evaluated how augmenting the training set of NSynth [14] by applying audio effects to the sounds can improve the automatic classification on the instruments of the unmodified test set. In the second experiment we evaluated how robust a state of the art model for instrument classification is when classifying sounds where these audio effects are applied.

The results of the first experiment are presented in Table 1, where the classification accuracy between the models trained with the original NSynth training set augmented with audio effects can be compared to the baseline (unprocessed dataset). We see that the increase in accuracy only occurs for chorus, heavy distortion and flanger effects. The highest classification accuracy was achieved by the dataset augmented with heavy distortion, where an increase of 1% was obtained. However, all the accuracy values are in a small interval (between 0.7334 and 0.7473), which means that the model was not able to learn from the augmented datasets. Future experiments are needed in order to understand why this occurs. In [16], the authors state that the superior performance obtained was due to an augmentation procedure coupled with an increase in the model capacity. Experiments with higher capacity models will be performed to understand if the size of the model used is limiting its performance on learning from the augmented dataset.

Table 1: Classification accuracy on the unprocessed test set.

Test Effect	Train Effect	Accuracy
None	None (baseline)	0.7378
	Heavy distortion	**0.7473**
	Saturation	0.7349
	Reverb	0.7375
	Chorus	**0.7417**
	Echo	0.7336
	Flanger	**0.7412**
	Pitch Shifting	0.7334

In Table 2, we present the accuracy values obtained when evaluating the trained model on test sets processed with effects. The first thing we verify is that the accuracy of the classification greatly decreases for almost all effects, when compared to the un-

Table 2: Classification accuracy on the augmented test set.

Test Effect	Train Effect	Accuracy
Heavy distortion	None	0.3145
	Heavy distortion	**0.3518**
Saturation	None	**0.4836**
	Saturation	0.4607
Reverb	None	**0.3931**
	Reverb	0.3774
Chorus	None	0.6348
	Chorus	**0.6436**
Echo	None	**0.4719**
	Echo	0.4319
Flanger	None	**0.7046**
	Flanger	0.7002
Pitch Shifting	None	**0.6980**
	Pitch Shifting	0.6741

processed sound classification. The model seems to be more robust to the flanger and to the pitch shifting effect, where the difference between the accuracy on the unprocessed test set and on the processed one is smaller than 4%. The effects which caused the biggest drops in accuracy (> 20%) were the heavy distortion, the saturation, the echo and the reverb. When evaluating if training with the augmented datasets increased the robustness of the model, we see that this is only true for the chorus and distortion effect. While for the heavy distortion effect the accuracy when training with the augmented set is improved by a significant value ($\approx 4\%$), the difference in accuracy between training with the augmented and the unprocessed sets are small. Further experiments will be performed to understand the bad generalisation of the model. Besides experimenting with a higher capacity model as previously stated, work will be conducted on further augmenting the datasets. Although the effects applied were the same in the training, validation and test sets, the implementations used were different in the training set. This leads to a different timbre between the sets that the architecture might not be able to generalise to. In future experiments, we will further augment the dataset using a number of different settings for each effect, as well as different combinations of the effects applied.

5. CONCLUSIONS

In this paper we evaluated how a state of the art algorithm for automatic instrument classification performs when classifying the NSynth dataset and how augmenting this dataset with audio effects commonly used in electronic music production influences its accuracy on both the original and processed versions of the audio. We identify that the accuracy of this algorithm is greatly decreased when tested on sounds where audio effects are applied and see that the augmentation can lead to better classification in unprocessed sounds. We note that the accuracy results provided are preliminary, and do not fully exploit the possibilities of using audio effects for data augmentation in automatic instrument classification. We are currently evaluating how a deeper architecture performs on the same task. Further work includes evaluating how using a bigger variety of effects, with different combinations of parameters, further improves the robustness of the classification algorithm.

6. ACKNOWLEDGMENTS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement N° 765068, MIP-Frontiers. We thank Matthew Davies for reviewing a draft of this paper and providing helpful feedback.

7. REFERENCES

[1] Frederic Font, Gerard Roma, and Xavier Serra, "Freesound technical demo," in *ACM International Conference on Multimedia (MM'13)*, Barcelona, Spain, 2013, ACM, pp. 411–412, ACM.

[2] Perfecto Herrera-Boyer, Geoffroy Peeters, and Shlomo Dubnov, "Automatic classification of musical instrument sounds," *Journal of New Music Research*, vol. 32, pp. 3–21, 2003.

[3] Masataka Goto, "RWC music database: Popular, classical, and jazz music databases," in *3rd International Society for Music Information Retrieval Conference (ISMIR)*, 2002, pp. 287–288.

[4] Juan J Bosch, Jordi Janer, Ferdinand Fuhrmann, and Perfecto Herrera, "A comparison of sound segregation techniques for predominant instrument recognition in musical audio signals," in *13th International Society for Music Information Retrieval Conference (ISMIR)*, 2012, pp. 559–564.

[5] Oriol Romani Picas, Hector Parra Rodriguez, Dara Dabiri, Hiroshi Tokuda, Wataru Hariya, Koji Oishi, and Xavier Serra, "A real-time system for measuring sound goodness in instrumental sounds," in *Audio Engineering Society Convention 138*, Warsaw, Poland, 2015, p. 9350.

[6] Venkatesh Shenoy Kadandale, "Musical Instrument Recognition in Multi-Instrument Audio Contexts," MSc thesis, Universitat Pompeu Fabra, Oct. 2018.

[7] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436, 2015.

[8] Luis Perez and Jason Wang, "The effectiveness of data augmentation in image classification using deep learning," *arXiv preprint arXiv:1712.04621*, 2017.

[9] Jordi Pons, Olga Slizovskaia, Rong Gong, Emilia Gómez, and Xavier Serra, "Timbre analysis of music audio signals with convolutional neural networks," in *25th European Signal Processing Conference (EUSIPCO)*. IEEE, 2017, pp. 2744–2748.

[10] Yoonchang Han, Jaehun Kim, Kyogu Lee, Yoonchang Han, Jaehun Kim, and Kyogu Lee, "Deep convolutional neural networks for predominant instrument recognition in polyphonic music," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 25, no. 1, pp. 208–221, Jan. 2017.

[11] Peter Li, Jiyuan Qian, and Tian Wang, "Automatic instrument recognition in polyphonic music using convolutional neural networks," *arXiv preprint arXiv:1511.05520*, 2015.

[12] Taejin Park and Taejin Lee, "Musical instrument sound classification with deep convolutional neural network using feature fusion approach," *arXiv preprint arXiv:1512.07370*, 2015.

[13] Juhan Nam, Keunwoo Choi, Jongpil Lee, Szu-Yu Chou, and Yi-Hsuan Yang, "Deep learning for audio-based music classification and tagging: Teaching computers to distinguish rock from bach," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 41–51, Jan 2019.

[14] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck, and Karen Simonyan, "Neural audio synthesis of musical notes with wavenet autoencoders," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, 2017, pp. 1068–1077.

[15] Brian McFee, Eric J Humphrey, and Juan Pablo Bello, "A software framework for musical data augmentation.," in *16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015, pp. 248–254.

[16] Justin Salamon and Juan Pablo Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.

[17] Tom Ko, Vijayaditya Peddinti, Daniel Povey, Michael L Seltzer, and Sanjeev Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 5220–5224.

[18] Udo Zölzer, *DAFX: Digital Audio Effects*, John Wiley & Sons, 2011.

[19] Joshua D Reiss and Andrew McPherson, *Audio effects: theory, implementation and application*, CRC Press, 2014.

[20] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[21] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[22] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[23] Brian McFee et al., "librosa/librosa: 0.6.3," Feb. 2019.

[24] Robert Ratcliffe, "A proposed typology of sampled material within electronic dance music," *Dancecult: Journal of Electronic Dance Music Culture*, vol. 6, no. 1, pp. 97–122, 2014.

[25] Shruti Sarika Chakraborty and Ranjan Parekh, *Improved Musical Instrument Classification Using Cepstral Coefficients and Neural Networks*, pp. 123–138, Springer Singapore, Singapore, 2018.

[26] Rachel M. Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello, "MedleyDB: A multitrack dataset for annotation-intensive MIR Research," in *the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

ANTIDERIVATIVE ANTIALIASING FOR STATEFUL SYSTEMS

Martin Holters

Department of Signal Processing and Communication
Helmut Schmidt University
Hamburg, Germany
martin.holters@hsu-hh.de

ABSTRACT

Nonlinear systems, like e.g. guitar distortion effects, play an important role in musical signal processing. One major problem encountered in digital nonlinear systems is aliasing distortion. Consequently, various aliasing reduction methods have been proposed in the literature. One of these is based on using the antiderivative of the nonlinearity and has proven effective, but is limited to memoryless systems. In this work, it is extended to a class of stateful systems which includes but is not limited to systems with a single one-port nonlinearity. Two examples from the realm of virtual analog modeling show its applicability to and effectiveness for commonly encountered guitar distortion effect circuits.

1. INTRODUCTION

Nonlinear systems play an important role in musical signal processing. In particular, there are many effects categorized as overdrive, distortion, or fuzz, whose primary objective it is to introduce harmonic distortion to enrich the signal. Usually the nonlinear behavior is in some way combined with (linear) filtering to spectrally shape the output signal or to make the amount of distortion introduced frequency dependent. While many of these systems were originally designed in the analog domain, naturally, there is interest in deriving digital models for them, e.g. [1, 2, 3, 4].

One major problem encountered in digital nonlinear systems, whether designed from scratch or derived by virtual analog modeling, is aliasing distortion. Once the additional harmonics introduced by the nonlinearity exceed the Nyquist frequency, they get folded back to lower frequencies, just as if the corresponding analog signal had been sampled without appropriate band-limiting. Contrary to the desired harmonic distortion, aliasing distortion is usually perceived as unpleasant. Therefore methods to suppress or reduce the aliasing distortion are needed.

The conceptually simplest aliasing reduction method is oversampling. However, if the harmonics decay slowly with frequency, the oversampling factor has to be high, making the approach unattractive due to the rising computational demand. Consequently, various alternatives have been proposed, e.g. [5, 6, 7, 8]. These methods, however, usually come with certain limitations, most commonly the restriction to memoryless systems. In this work, an extension of [7] is presented that loosens the restriction from memoryless systems to a certain class of stateful systems.

2. ANTIDERIVATIVE-BASED ALIASING REDUCTION FOR MEMORYLESS NONLINEAR SYSTEMS

As the proposed method builds upon the approach from [7], we shall briefly summarize the latter. Conceptually, the digital signal is converted to a continuous-time signal using linear interpolation between consecutive samples, the nonlinearity is applied, and the result is lowpass-filtered by integrating over one sampling interval before sampling to obtain the digital output signal. The key insight is that, as there is a linear relationship between time and input signal amplitude (within one sampling interval), one can substitute the integration variable to integrate over amplitude instead of time. Then, by the fundamental theorem of calculus, one only needs to evaluate the antiderivative of the nonlinear mapping function at the input sample amplitudes and does not need to explicitly form the continuous signal. (For a more detailed explanation, the reader is referred to [7].)

The result is that the nonlinear system

$$y(n) = f\big(u(n)\big), \tag{1}$$

where $f(u)$ denotes the nonlinear function, mapping input sample $u(n)$ to output sample $y(n)$, is replaced with

$$\begin{aligned} y(n) &= \tilde{f}\big(u(n), u(n-1)\big) \\ &= \begin{cases} \frac{F(u(n))-F(u(n-1))}{u(n)-u(n-1)} & \text{if } u(n) \not\approx u(n-1) \\ f\big(\frac{1}{2}u(n) + \frac{1}{2}u(n-1)\big) & \text{if } u(n) \approx u(n-1) \end{cases} \end{aligned} \tag{2}$$

where $F(u) = \int f(u)du$ is the antiderivative of $f(u)$ and the $u(n) \approx u(n-1)$ case is treated separately to avoid numerical issues when dividing by $u(n)-u(n-1)$. In addition to reducing aliasing artifacts, the approach introduces a half-sample delay and attenuates high frequencies. This can be readily seen when using the identity function $f(u) = u$ instead of a true nonlinearity. Straight forward calculation yields

$$y(n) = \tfrac{1}{2}u(n) + \tfrac{1}{2}u(n-1) \tag{3}$$

in that case, i.e. a first-order FIR low-pass filter with a group delay of half a sample. The low-pass effect can be countered by a modest amount of oversampling (e.g. by a factor of two) and the delay usually is of no concern.

3. EXTENSION TO STATEFUL SYSTEMS

The half-sample delay introduced by the method of [7] becomes problematic if the nonlinearity is embedded in the feedback loop of a stateful system. As noted in [7], for the particular case of an integrator following the nonlinearity and using trapezoidal rule for

time-discretization, one can simply replace the numerator of the discretized integrator's transfer function with the filter introduced by antialiasing. This fusing of antialiased nonlinearity and integrator then has no additional delay compared to the nonantialiased system, hence can be used inside a feedback system without problems.

Here, we consider systems which do not necessarily have an integrator following the nonlinearity. In particular, we shall consider the general discrete nonlinear state-space system

$$\boldsymbol{x}(n) = \boldsymbol{A}\boldsymbol{x}(n-1) + \boldsymbol{b}u(n) + \boldsymbol{f}_\mathrm{x}(p_\mathrm{x}(n)) \tag{4}$$

$$y(n) = \boldsymbol{c}^T\boldsymbol{x}(n-1) + du(n) + f_\mathrm{y}(p_\mathrm{y}(n)) \tag{5}$$

with

$$p_\mathrm{x}(n) = \boldsymbol{c}_{\mathrm{p_x}}^T\boldsymbol{x}(n-1) + d_{\mathrm{p_x}}u(n) \tag{6}$$

$$p_\mathrm{y}(n) = \boldsymbol{c}_{\mathrm{p_y}}^T\boldsymbol{x}(n-1) + d_{\mathrm{p_y}}u(n), \tag{7}$$

where $\boldsymbol{x}(n)$ is the state vector, $u(n)$ in the input, $y(n)$ is the output, $\boldsymbol{A}$ is the state martix, $\boldsymbol{b}$ is the input matrix, $\boldsymbol{c}^T$ is the output matrix, and d is the feedthrough matrix, where the latter three are reduced to vectors and a scaler, respectively, as we only consider scalar input and output. The nonlinearity of the system is captured in two nonlinear functions, $\boldsymbol{f}_\mathrm{x}$ and f_y, influencing state update and output, respectively. Their arguments $p_\mathrm{x}(n)$ and $p_\mathrm{y}(n)$ are calculated by (6) and (7) similarly to the linear part of the output equation (5). Some remarks are in order:

- While we allow multiple states, collected in the vector $\boldsymbol{x}(n)$, we restrict the presentation to a single input $u(n)$ and a single output $y(n)$, as that is the most common case. Extension to multiple inputs and/or outputs is straight-forward.
- The limitation to scalar-valued $p_\mathrm{x}(n)$ and $p_\mathrm{y}(n)$, however, is necessary, as the method of [7] is restricted to nonlinear functions with scalar argument. Facilitating this is the reason why the linear parts $\boldsymbol{A}\boldsymbol{x}(n-1)+\boldsymbol{b}u(n)$ and $\boldsymbol{c}^T\boldsymbol{x}(n-1)+du(n)$ have not been subsumed in the nonlinear functions in (4) and (5), respectively.
- If the system is obtained in the context of virtual analog modeling, usually the nonlinear functions will only be given implicitly (as the solution of what is sometimes referred to as a delay-free loop), making solving a nonlinear equation necessary. However, they are typically based on a common function, only applying different weighting to its output, i.e. $\boldsymbol{f}_\mathrm{x}(p_\mathrm{x}(n)) = \boldsymbol{W}_\mathrm{x}\boldsymbol{f}(p(n))$ and $f_\mathrm{y}(p_\mathrm{y}(n)) = \boldsymbol{w}_\mathrm{y}^T\boldsymbol{f}(p(n))$ with $p(n) = p_\mathrm{x}(n) = p_\mathrm{y}(n)$. While this redundancy should be kept in mind for optimizing an implementation, we will derive our method for the more general case of two possibly independent nonlinear functions for state update and output.

In a first step, we may consider only applying the aliasing suppression to $f_y(p)$, as it is not part of any feedback loop. We have to be careful though, and may not just replace f_y with $\tilde{f}_y$ in (5), as that would lead to a misalignment in time of the different summed terms. Instead, we have to use

$$y(n) = \frac{1}{2}\boldsymbol{c}^T\big(\boldsymbol{x}(n-1) + \boldsymbol{x}(n-2)\big) + \frac{1}{2}d\big(u(n) + u(n-1)\big) + \tilde{f}_\mathrm{y}(p_\mathrm{y}(n), p_\mathrm{y}(n-1)). \tag{8}$$

However, any aliasing distortion introduced into $\boldsymbol{x}(n)$ by (4) will not undergo any mitigation (except for the lowpass filtering).

Now, if we naively rewrite (4) as we did with (5), we modify our system in an unwanted way as we introduce additional delay in the feedback. But we do that in a very controlled way: The unit delay in the feedback is replaced by a delay of 1.5 samples. This is equivalent to reducing the sampling rate by a factor of 1.5, so we can compensate by designing our system for this reduced sampling rate, arriving at

$$\boldsymbol{x}(n) = \frac{1}{2}\tilde{\boldsymbol{A}}\big(\boldsymbol{x}(n-1) + \boldsymbol{x}(n-2)\big) + \frac{1}{2}\tilde{\boldsymbol{b}}\big(u(n) + u(n-1)\big) + \tilde{f}_\mathrm{x}(p_\mathrm{x}(n), p_\mathrm{x}(n-1)) \tag{9}$$

$$y(n) = \frac{1}{2}\tilde{\boldsymbol{c}}^T\big(\boldsymbol{x}(n-1) + \boldsymbol{x}(n-2)\big) + \frac{1}{2}\tilde{d}\big(u(n) + u(n-1)\big) + \tilde{f}_\mathrm{y}(p_\mathrm{y}(n), p_\mathrm{y}(n-1)). \tag{10}$$

with

$$p_\mathrm{x}(n) = \tilde{\boldsymbol{c}}_{\mathrm{p_x}}^T\boldsymbol{x}(n-1) + \tilde{d}_{\mathrm{p_x}}u(n) \tag{11}$$

$$p_\mathrm{y}(n) = \tilde{\boldsymbol{c}}_{\mathrm{p_y}}^T\boldsymbol{x}(n-1) + \tilde{d}_{\mathrm{p_y}}u(n) \tag{12}$$

where all coefficients are calculated for the reduced sampling rate $\tilde{f}_\mathrm{s} = \frac{2}{3}f_\mathrm{s}$. We can only do this because we do not have a delay-free loop. Or rather, the delay-free loop is hidden inside $f(u)$: Instead of worrying about a nonlinearity within a delay-free loop, we treat the solution of the delay-free loop as the nonlinearity to apply aliasing reduction to. Note that the behavior for frequencies above $\frac{1}{2}\tilde{f}_\mathrm{s} = \frac{1}{3}f_\mathrm{s}$ is ill-defined, but with the mild oversampling suggested by [7] anyway, we do not have to worry about this.

The increased delay is not the only effect of the modification. There is also the low-pass filtering. To study this in more detail, assume $\boldsymbol{f}_\mathrm{x}(p_\mathrm{x})$ and $f_\mathrm{y}(p_\mathrm{y})$ to be linear so that we have a linear system, and let $H(z)$ denote the transfer function obtained from (4)–(7). If we instead use (9)–(12) without adjusting the coefficients, it is straight forward to verify that the resulting transfer function fulfills

$$\tilde{H}(z) = \frac{1}{2}(1+z^{-1})\cdot H\left(\left(\tfrac{1}{2}(z^{-1}+z^{-2})\right)^{-1}\right). \tag{13}$$

We may observe two effects: The well-known filtering with a factor on the outside and the substitution $z \leftarrow \left(\frac{1}{2}(z^{-1}+z^{-2})\right)^{-1}$ in the argument of H. Evaluating the latter for $z = e^{j\omega}$, we note that

$$\left(\tfrac{1}{2}(e^{-j\omega}+e^{-2j\omega})\right)^{-1} = \frac{1}{\cos(\frac{1}{2}\omega)}e^{\frac{3}{2}j\omega} \tag{14}$$

depicted in figure 1. While in the original system $H(z)$ is evaluated on the unit circle $e^{j\omega}$ (shown dotted) to obtain the frequency response, for the modified system, it is evaluated on the trajectory of (14). We notice that, in addition to the frequency scaling by $\frac{3}{2}$, there is an additional scaling away from the unit circle, increasing with frequency. Importantly, as we only evaluate $H(z)$ for z on or outside the unit circle, we preserve stability, i.e. if $H(z)$ is stable, so is $\tilde{H}(z)$. Nevertheless, especially for higher frequencies, this may cause a significant distortion of the frequency response.

An extreme example would be an all-pass filter with high Q-factor, where the transformation might result in the zero moving onto the frequency axis, turning a flat frequency response into one with a deep notch. As the examples will demonstrate, many typical systems are rather well-behaved under the transformation, but one has to be aware of this pitfall.

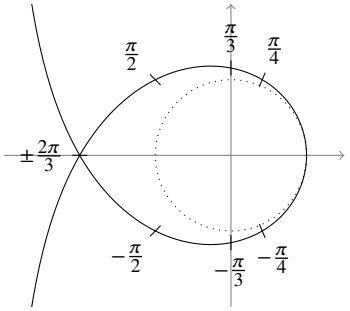


Figure 1: *Trajectory of* $\left(\frac{1}{2}e^{-j\omega} + \frac{1}{2}e^{-2j\omega}\right)^{-1}$ *compared to the unit circle* $e^{j\omega}$ *(dotted)*

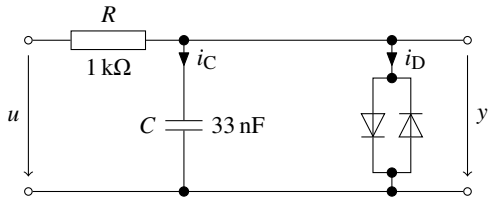


Figure 2: *Schematics of the modeled diode clipper*

4. EXAMPLES

4.1. Diode clipper

As a first example, we consider the diode clipper of figure 2. The circuit is simple enough that we briefly repeat the modeling process here.

From Kirchhoff's and Ohm's laws and $i_C = C\dot{y}$, we immediately obtain

$$y = u - R \cdot (i_C + i_D) = u - RC\dot{y} - Ri_D. \tag{15}$$

Summing over two subsequent sampling instances, we get

$$\begin{aligned} &y(n) + y(n-1) = \\ &u(n) + u(n-1) - RC(\dot{y}(n) + \dot{y}(n-1)) - R(i_D(n) + i_D(n-1)). \end{aligned} \tag{16}$$

We now use the trapezoidal rule to substitute

$$\dot{y}(n) + \dot{y}(n-1) = 2f_s(y(n) - y(n-1)) \tag{17}$$

and obtain

$$\begin{aligned} &y(n) + y(n-1) = \\ &u(n) + u(n-1) - 2RCf_s(y(n) - y(n-1)) - R(i_D(n) + i_D(n-1)). \end{aligned} \tag{18}$$

Collecting all quantities from time step $n-1$ into canonical states

$$x(n-1) = (2RCf_s - 1)y(n-1) + u(n-1) - Ri_D(n-1) \tag{19}$$

allows simplification to

$$y(n) = x(n-1) + u(n) - 2RCf_s y(n) - Ri_D(n). \tag{20}$$

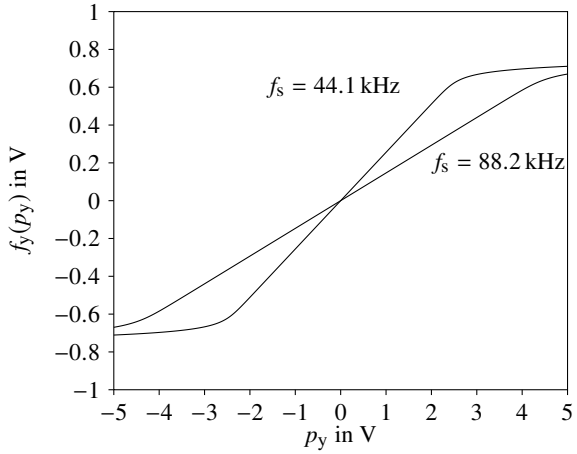


Figure 3: *Nonlinear function* $f_y(p_y)$ *of the diode clipper for two different sampling rates* f_s

Using Shockley's equation for the diodes, we get

$$\begin{aligned} i_D(n) &= I_S \cdot \left(e^{y(n)/v_T} - 1\right) - I_S \cdot \left(e^{-y(n)/v_T} - 1\right) \\ &= 2I_S \sinh\left(y(n)/v_T\right), \end{aligned} \tag{21}$$

where saturation current and temperature voltage have been chosen as $I_S = 1\,\text{fA}$ and $v_T = 25\,\text{mV}$ respectively. Inserting (21) into (20) and introducing

$$p_y(n) = x(n-1) + u(n) \tag{22}$$

then leads to the implicit equation

$$y(n) = p_y(n) - 2RCf_s y(n) - 2RI_S \sinh\left(y(n)/v_T\right) \tag{23}$$

for $y(n)$. Note that we do not treat this as a delay-free loop and apply the antialiasing to the sinh-function. Instead, we let $f_y(p_y(n)) = y(n)$ denote the solution of the implicit equation. The resulting function is depicted in figure 3 (obtained using an iterative solver).

To obtain the state update equation, we rearrange (20) to

$$(2RCf_s - 1)y(n) + u(n) - Ri_D(n) = -x(n-1) + 4RCf_s y(n) \tag{24}$$

and note by comparing with (19) that the left-hand side equals $x(n)$. Thus introducing

$$f_x(p_x(n)) = 4RCf_s f_y(p_y(n)) \tag{25}$$

with $p_x(n) = p_y(n)$, we arrive at

$$x(n) = -x(n-1) + f_x(p_x(n)) \tag{26}$$

$$y(n) = f_y(p_y(n)) \tag{27}$$

of the desired form.

Applying the aliasing mitigation only to the output equation is particularly simple in this case, giving

$$y(n) = \tilde{f_y}(p_y(n), p_y(n-1)) \tag{28}$$

with $\tilde{f_y}$ defined according to (2). The required antiderivative $F_y(p_y)$ of $f_y(p_y)$, depicted in figure 4, has to be determined numerically. For the results below, it has been precomputed at 1024 uniformly

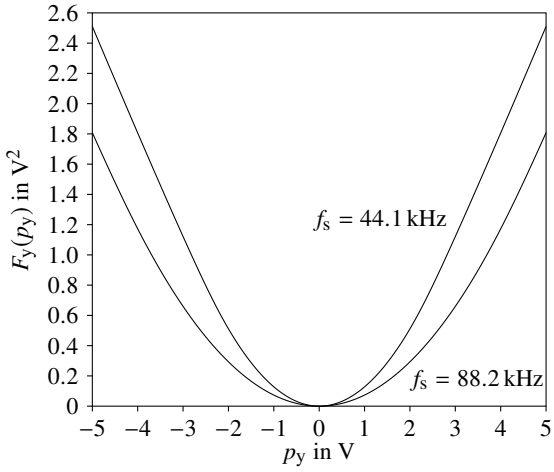


Figure 4: *Antiderivative $F_y(p_y)$ of $f_y(p_y)$ of the diode clipper for two different sampling rates f_s*

distributed points in the relevant range and stored in a table, using cubic interpolation during lookup.

To also apply aliasing mitigation to the state update equation, we have to change it to

$$x(n) = -\tfrac{1}{2}\big(x(n-1) + x(n-2)\big) + \tilde{f}_x(p_x(n), p_x(n-1)) \qquad (29)$$

and substitute $\tilde{f}_s = \frac{2}{3} f_s$ for f_s in (23) and (25). Note that this immediately leads to

$$\tilde{f}_x(p_x(n), p_x(n-1)) = 4RC\tilde{f}_s\tilde{f}_y(p_y(n), p_y(n-1)). \qquad (30)$$

To study the effectiveness of the method, we consider figure 5, where the output spectra for a sinusoidal excitation are depicted for various model configurations. Figures 5(a) and 5(b) give the baseline, the system without any aliasing mitigation at sampling rates $f_s = 44.1$ kHz and $f_s = 88.2$ kHz, respectively. Only applying aliasing mitigation to the output equation according to (28) is of little benefit, as seen when considering figures 5(c) and 5(d) in comparison. We do note, however, the low-pass effect in figure 5(c), where higher harmonics exhibit an attenuation of up to 10 dB.

When also applying the aliasing mitigation to the state update equation according to (29), we observe a significant aliasing reduction in figures 5(e) and 5(f). As explained, the aliasing mitigation should be combined with (modest) oversampling. In this particular case, as verified in figure 5(e), the model is still a relatively good fit even without oversampling, which however must be mainly attributed to lucky coincidence. More relevant is the case of a sampling rate of $f_s = 88.2$ kHz, shown in figure 5(f). Comparing to oversampling to $f_s = 220.5$ kHz without additional aliasing mitigation measures, as shown in figure 5(g), we see that the aliased components at low frequencies, where they are most easily perceived, are at a comparable level.

4.2. Tube screamer-like distortion circuit

As a second example we consider the distortion circuit of figure 6, inspired by the Ibanez Tube Screamer TS-808. We shall not go into details of the modeling procedure (for which we have used ACME.jl[1]), but remark that if one allows the three diodes to be different, one can no longer derive a closed-form expression for their combined behavior. Instead, the nonlinear behavior is determined by a system of three equations. Nevertheless, using the dimensionality reduction approach of [9], the input $p_x(n) = p_y(n)$ to the nonlinearity can be reduced to a scalar value, formed by linear combination of the input and the capacitor states. Hence, the proposed method is applicable.

Figure 7 again shows the output spectra for a sinusoidal excitation. As can be seen in figure 7(a), with plain oversampling to $f_s = 88.2$ kHz, the signal contains strong aliasing components. Applying aliasing mitigation only to the output equation reduces the aliasing distortion to a limited extent, as shown in figure 7(b). In contrast, when also applying aliasing mitigation to the state update equation, the aliasing is significantly reduced, as seen in figure 7(c). Again, the aliasing mitigation is most effective at low frequencies, where it is also perceptually most relevant. As in the diode clipper example, for low frequencies the system with aliasing mitigation at $f_s = 88.2$ kHz performs at least as good as an unmodified system at $f_s = 220.5$ kHz, see figure 7(d).

[1] https://github.com/HSU-ANT/ACME.jl

5. CONCLUSION AND OUTLOOK

The presented approach for aliasing reduction generalizes the approach of [7] to all nonlinear systems that can be cast in a way that the nonlinearity takes a scalar input. This includes, but is not limited to, all models of circuits with a single one-port nonlinear element. If the system contains a delay-free loop, it has to be re-cast such that the nonlinearity is defined as the solution of the delay-free loop. Then, the delay introduced by applying the method of [7] to the nonlinearity can be compensated by adjusting the system's coefficients, even if the nonlinearity is part of a feedback loop.

As is to be expected, the achieved aliasing reduction is comparable to that of [7], allowing to significantly reduce the required oversampling especially for systems which introduce strong distortion, while the additional computational load is modest. Assuming lookup tables are used for $f(u)$ (in general being implicitly defined) and its antiderivative $F(u)$, the main price to pay is in terms of memory used.

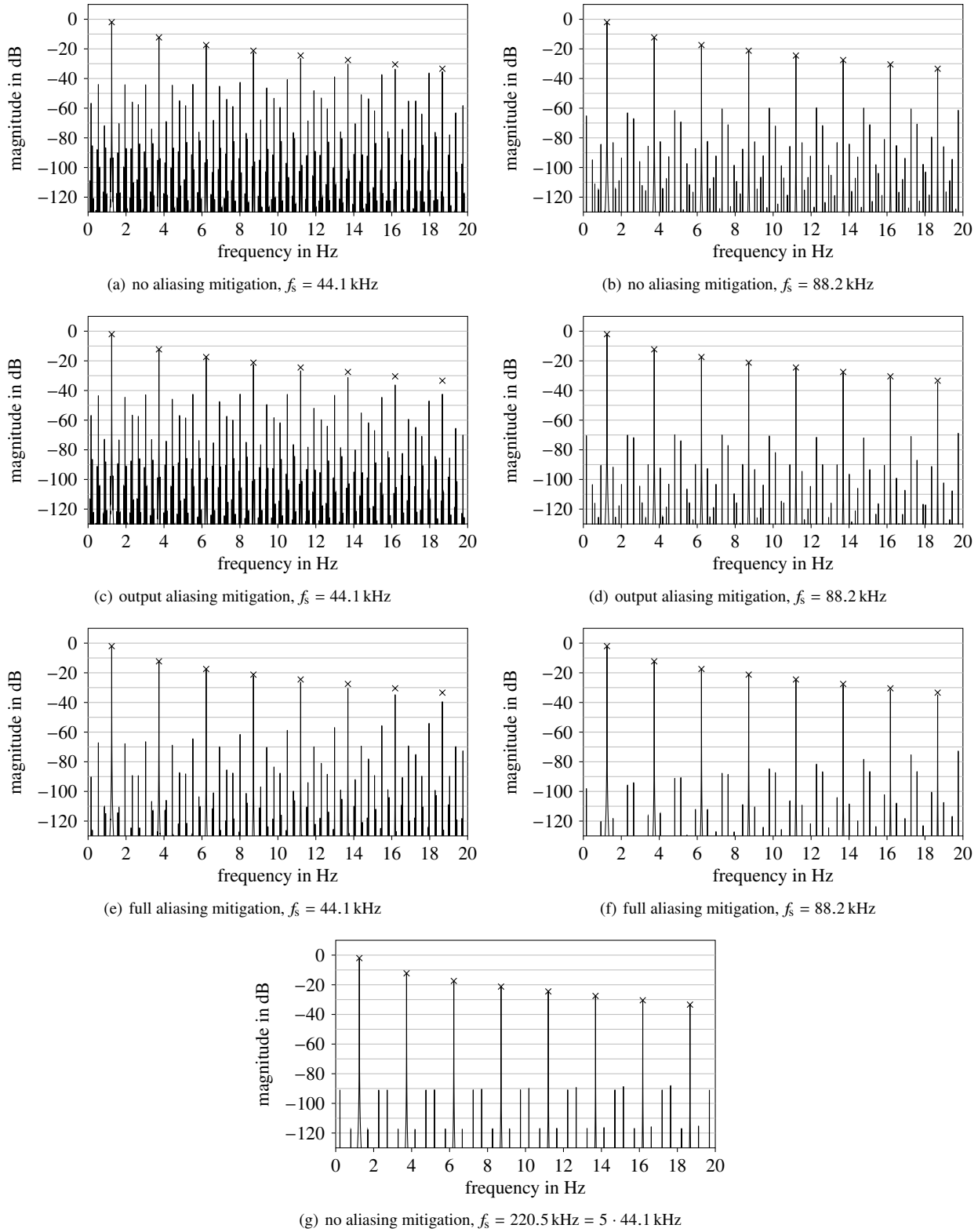
It should be noted that the extensions to higher order antiderivatives as proposed in [10] or [11] should be straight-forward, following the same principle. A more interesting future direction would be to lift the restriction on the nonlinear function to have only scalar input. If the method of [7] (or even the higher order extensions of [10] or [11]) could be generalized to nonlinear functions with multiple inputs, the method proposed in the present paper would immediately generalize to all stateful nonlinear systems.

6. REFERENCES

[1] David T. Yeh, Jonathan S. Abel, and Julius O. Smith III, "Simplified, physically-informed models of distortion and overdrive guitar effects pedals," in *Proc. 10th Int. Conf. on Digital Audio Effects (DAFx-07)*, Bordeaux, France, 2007.

[2] Martin Holters, Kristjan Dempwolf, and Udo Zölzer, "A digital emulation of the Boss SD-1 Super Overdrive pedal based on physical modeling," in *131st AES Convention*, New York, NY, USA, 2011.

[3] W. Ross Dunkel, Maximilian Rest, Kurt James Werner, Michael Jørgen Olsen, and Julius O. Smith III, "The Fender Bassman 5F6-A family of preamplifier circuits — a wave



(a) no aliasing mitigation, f_s = 44.1 kHz

(b) no aliasing mitigation, f_s = 88.2 kHz

(c) output aliasing mitigation, f_s = 44.1 kHz

(d) output aliasing mitigation, f_s = 88.2 kHz

(e) full aliasing mitigation, f_s = 44.1 kHz

(f) full aliasing mitigation, f_s = 88.2 kHz

(g) no aliasing mitigation, f_s = 220.5 kHz = 5 · 44.1 kHz

Figure 5: *Output spectra of various model configurations for the diode clipper when excited with a single sinusoid of* 10 V *amplitude at* 1244.5 Hz. *Crosses mark expected harmonics.*

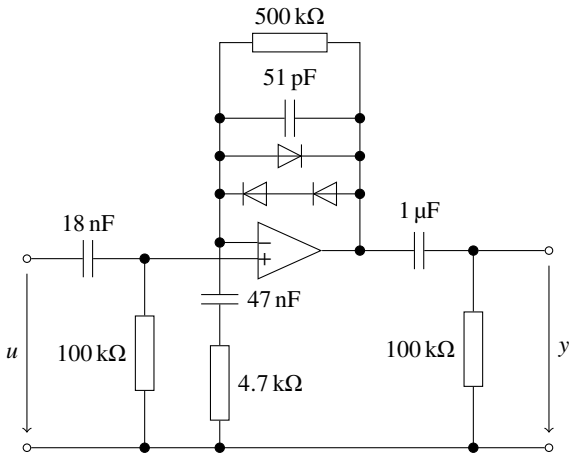


Figure 6: *Tube screamer-like distortion circuit*

digital filter case study," in *Proc. 19th Int. Conf. on Digital Audio Effects (DAFx-16)*, Brno, Czech Republic, 2016, pp. 263–270.

[4] Fabián Esqueda, Henri Pöntynen, Vesa Välimäki, and Julian D. Parker, "Virtual analog Buchla 259 wavefolder," in *Proc. 20th Int. Conf. on Digital Audio Effects (DAFx-17)*, Edinburgh, UK, 2017, pp. 192–199.

[5] Fabián Esqueda and Vesa Välimäki, "Rounding corners with BLAMP," in *Proc. 19th Int. Conf. on Digital Audio Effects (DAFx-16)*, Brno, Czech Republic, 2016, pp. 121–128.

[6] Fabián Esqueda, Vesa Välimäki, and Stefan Bilbao, "Antialiased soft clipping using an integrated bandlimited ramp," in *Proc. 24th European Signal Process. Conf. (EUSIPCO)*, Budapest, Hungary, 2016, pp. 1043–1047.

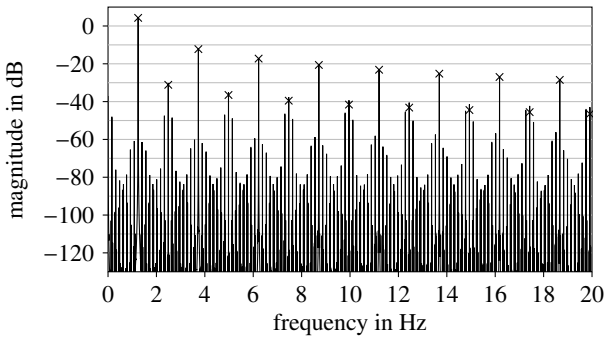
[7] Julian D. Parker, Vadim Zavalishin, and Efflam Le Bivic, "Reducing the aliasing of nonlinear waveshaping using continuous-time convolution," in *Proc. 19th Int. Conf. on Digital Audio Effects (DAFx-16)*, Brno, Czech Republic, 2016, pp. 137–144.

[8] Rémy Muller and Thomas Hélie, "Trajectory anti-aliasing on guaranteed-passive simulation of nonlinear physical systems," in *Proc. 20th Int. Conf. on Digital Audio Effects (DAFx-17)*, Edinburgh, UK, 2017, pp. 87–94.

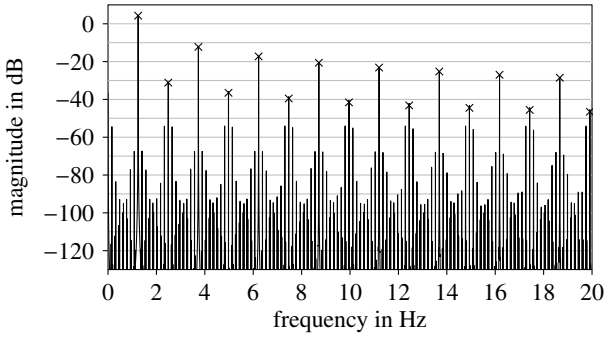
[9] Martin Holters and Udo Zölzer, "A k-d tree based solution cache for the non-linear equation of circuit simulations," in *Proc. 24th European Signal Process. Conf. (EUSIPCO)*, Budapest, Hungary, 2016, pp. 1028–1032.

[10] Stefan Bilbao, Fabián Esqueda, Julian D. Parker, and Vesa Välimäki, "Antiderivative antialiasing for memoryless nonlinearities," *IEEE Signal Process. Lett.*, vol. 24, no. 7, pp. 1049–1053, 2017.

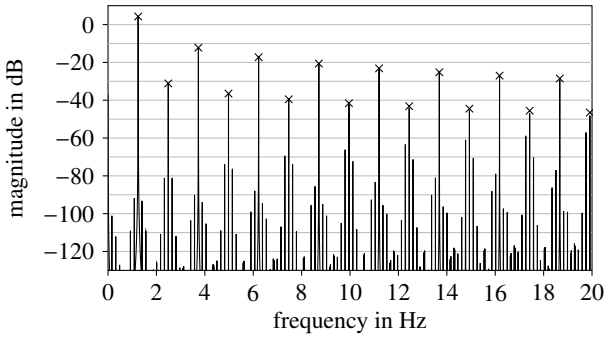
[11] Stefan Bilbao, Fabian Esqueda, and Vesa Välimäki, "Antiderivative antialiasing, lagrange interpolation and spectral flatness," in *2017 IEEE Workshop on Appl. of Signal Process. to Audio and Acoust. (WASPAA)*, New Paltz, NY, USA, 2017, pp. 141–145.



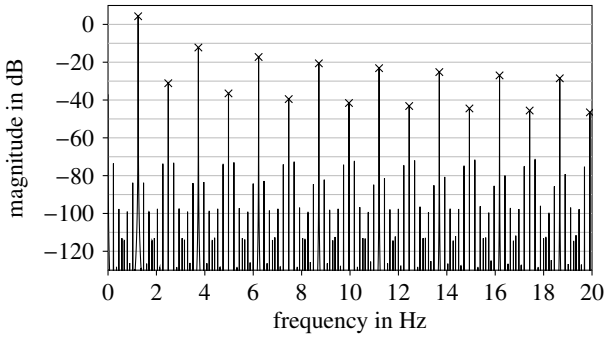
(a) no aliasing mitigation, f_s = 88.2 kHz



(b) output aliasing mitigation, f_s = 88.2 kHz



(c) full aliasing mitigation, f_s = 88.2 kHz



(d) no aliasing mitigation, f_s = 220.5 kHz = 5 · 44.1 kHz

Figure 7: *Output spectra of various model configurations for the tube screamer distortion circuit when excited with a single sinusoid of* 1 V *amplitude at* 1244.5 Hz*. Crosses mark expected harmonics.*

FAST APPROXIMATION OF THE LAMBERT W FUNCTION FOR VIRTUAL ANALOG MODELLING

Stefano D'Angelo
Independent researcher
Agropoli, Italy
s@dangelo.audio

Leonardo Gabrielli
Department of Information Engineering,
Università Politecnica delle Marche
Ancona, Italy
l.gabrielli@univpm.it

Luca Turchet
Department of Information Engineering and Computer Science,
University of Trento
Trento, Italy
luca.turchet@unitn.it

ABSTRACT

When modelling circuits one has often to deal with equations containing both a linear and an exponential part. If only a single exponential term is present or predominant, exact or approximate closed-form solutions can be found in terms of the Lambert W function. In this paper, we propose reformulating such expressions in terms of the Wright Omega function when specific conditions are met that are customary in practical cases of interest. This eliminates the need to compute an exponential term at audio rate. Moreover, we propose simple and real-time suitable approximations of the Omega function. We apply our approach to a static and a dynamic nonlinear system, obtaining digital models that have high accuracy, low computational cost, and are stable in all conditions, making the proposed method suitable for virtual analog modelling of circuits containing semiconductor devices.

1. INTRODUCTION

Typically, circuits are modelled as systems of differential equations which often contain nonlinearities [1, 2]. This is true of both state-space methods [3, 4] and wave digital filters [5, 6, 7]. Because of the exponential relation in the Shockley model of p-n junctions in semiconductors [8], which is also used in the Ebers-Moll model of bipolar junction transistors (BJT) [9], often one has to deal with equations containing both a linear and an exponential part. These equations may be solved numerically by iterative methods. However, these approaches often result in high computational load and can be problematic in terms of stability and/or accuracy [10, 11].

When only a single exponential term is present or predominant, it is possible to utilize the Lambert W function [12, 13, 14] to analytically solve these equations [15, 16, 17, 18, 19, 20, 21]. When applicable, such an approach brings remarkable advantages. In the context of circuit simulation, it is often the case that the solution only involves the main branch of the W function while its argument contains a time-varying exponential term. Various approximations [22, 23, 24, 25, 26] as well as algorithms [27, 28] have been proposed in the literature for the computation of the W function but, to the best of authors' knowledge, evaluation methods have always been preferred, in the context of real-time music DSP, that trade lower accuracy for higher performance [16, 17, 18, 19, 20].

In this paper, we propose to reformulate expressions involving the main branch of the Lambert W function in terms of the Wright Omega function [29] in the context of virtual analog modelling, which eliminates the need to compute the exponential term usually found in the argument. Moreover, we propose simple approximations that can be used when algorithms already available for a precise evaluation of the Omega function (*e.g.*, [30, 31]) are too demanding from a computational standpoint.

The reminder of the paper is structured as follows. Section 2 describes both the Lambert W and Wright Omega functions. Section 3 presents four approximations of the Wright Omega function. Section 4 reports the application of the proposed approach for two circuits, whereas Section 5 concludes the paper.

2. THE LAMBERT W AND WRIGHT OMEGA FUNCTIONS

In this section we introduce very briefly the Lambert W and Wright Omega functions. Since time-domain circuit simulation most often only involves real quantities, we will limit ourselves to the $\mathbb{R} \to \mathbb{R}$ case, that is the argument is real and not less than $-\frac{1}{e}$ for the Lambert W function and just real for the Wright Omega function.

2.1. The Lambert W function

The Lambert W function is a non-injective function, with two branches in the $\mathbb{R} \to \mathbb{R}$ case, which is defined as the inverse function of

$$f(x) = xe^x, \tag{1}$$

where e^x is the exponential function and $x \geq -\frac{1}{e}$. This can be expressed as

$$x = f^{-1}\left(xe^x\right) = W\left(xe^x\right). \tag{2}$$

The defining equation for the W function can be derived by substituting $x_0 = xe^x$ into equation 2,

$$x_0 = W(x_0)e^{W(x_0)}, \tag{3}$$

for any $x_0 \geq -\frac{1}{e}$.

$W(x)$ is two-valued for $x \in \left[-\frac{1}{e}, 0\right)$, therefore we will refer to the main branch ($W(x) \geq -1$) as $W_0(x)$ and to the other branch ($W(x) \leq -1$) as $W_{-1}(x)$.

Now we can express the solution of equation

$$e^{ax+b} = cx + d, \tag{4}$$

as

$$x = -\frac{W\left(-\frac{a}{c}e^{b-a\frac{d}{c}}\right)}{a} - \frac{d}{c}, \tag{5}$$

with a and c not null. In particular, if a and c have the same sign, the argument of $W()$ is negative and there are either two (possibly coincident) or no solutions, otherwise the argument is positive, there is one solution, and $W(x) = W_0(x)$.

In circuit modelling, often a and c have opposite sign and are either constants or control-rate expressions, while b or d contain audio-rate components [17, 18, 19, 20, 21]. Therefore, under the previous formulation, not only one needs to compute $W()$, but also the exponential in its argument. In these cases, it is however still possible to more conveniently reformulate the previous expression as

$$x = -\frac{W_0\left(e^{b-a\frac{d}{c}+\log(-\frac{a}{c})}\right)}{a} - \frac{d}{c}. \tag{6}$$

2.2. The Wright Omega function

The Wright Omega function is defined in terms of the Lambert W function as

$$\omega(x) = W_0(e^x), \tag{7}$$

which can obviously be used to express equation 6 as

$$x = -\frac{\omega\left(b - a\frac{d}{c} + \log(-\frac{a}{c})\right)}{a} - \frac{d}{c}. \tag{8}$$

In practice the logarithm is typically a constant or a control-rate expression and $\omega()$ is the only transcendental function to be computed at audio-rate.

3. APPROXIMATIONS AND COMPUTATION

High-precision algorithms have been proposed to compute $\omega()$ [30, 31], yet in the context of real-time music DSP it is usually preferable to trade some accuracy for higher computational efficiency. Indeed, some approximations have been already proposed in [16, 18], even if $\omega()$ was not explicitly mentioned. In this section we propose four approximations with different degrees of accuracy and complexity. Moreover, since two of these approximations presuppose fast computation of logarithm and exponential functions, we also discuss two commonly used approaches for approximating them with a focus on the problem at hand, in case the standard routines supplied for the target platform are not sufficiently efficient.

3.1. Approximations of $\omega(x)$

Several approximations of increasing computational cost are proposed hereby. The plot of $\omega(x)$, in Figure 1(a), suggests that a first, rough approximation can be

$$\omega(x) \approx \omega_1(x) = \max(0, x). \tag{9}$$

One could use a cubic spline to smooth the function around 0 as

$$\omega(x) \approx \omega_2(x) = \begin{cases} 0 & \text{for } x \leq x_1, \\ \alpha x^3 + \beta x^2 + \gamma x + \zeta & \text{for } x_1 < x < x_2, \\ x & \text{for } x \geq x_2. \end{cases} \tag{10}$$

It is sufficient to set the conditions of continuity C^1 to univocally determine the parameters $\alpha, \beta, \gamma, \zeta$ from x_1 and x_2. Optimzing these last two variables by the least squares method to minimize the absolute error in the range $[-10, 10]$ leads to

$$\begin{aligned} x_1 &= -3.684303659906469, \\ x_2 &= 1.972967391708859, \\ \alpha &= 9.451797158780131 \cdot 10^{-3}, \\ \beta &= 1.126446405111627 \cdot 10^{-1}, \\ \gamma &= 4.451353886588814 \cdot 10^{-1}, \\ \zeta &= 5.836596684310648 \cdot 10^{-1}. \end{aligned}$$

Such an approximation is non optimal for $x \geq x_2$. Since $W(x)e^{W(x)} = x$, then $\omega(x)e^{\omega(x)} = e^x$, or otherwise $\omega(x) = x - \log(\omega(x))$. This last relation can be used as a successive approximation method (*i.e.*, $\omega_n = x - \log(\omega_{n-1})$) for $x \geq 1$ (the same approach is used in [17]). Applying it to improve the accuracy in the range of interest gives

$$\omega(x) \approx \omega_3(x) = \begin{cases} 0 & \text{for } x \leq x_1, \\ \alpha x^3 + \beta x^2 + \gamma x + \zeta & \text{for } x_1 < x < x_2, \\ x - \log(x) & \text{for } x \geq x_2. \end{cases} \tag{11}$$

Again, $\alpha, \beta, \gamma, \zeta$ are univocally determined from x_1 and x_2 if C^1 continuity is imposed. This time we also take into account that fast approximations for $\log(x)$ exist which are exact for $x = 2^y$ with $y \in \mathbb{Z}$, hence we also ensure that x_2 is a power of 2. Reiterating the optimization process, one gets

$$\begin{aligned} x_1 &= -3.341459552768620, \\ x_2 &= 8, \\ \alpha &= -1.314293149877800 \cdot 10^{-3}, \\ \beta &= 4.775931364975583 \cdot 10^{-2}, \\ \gamma &= 3.631952663804445 \cdot 10^{-1}, \\ \zeta &= 6.313183464296682 \cdot 10^{-1}. \end{aligned}$$

To further improve the accuracy of such approximation, a Newton-Raphson iteration can be applied

$$\omega_4(x) = \omega_3(x) - \frac{\omega_3(x) - e^{x-\omega_3(x)}}{\omega_3(x) + 1}, \tag{12}$$

even if an approximation is used for the exponential term.

Figure 1(a) shows a plot of the $\omega(x)$ function along the four proposed approximations, while Figure 1(b) shows the distribution of the absolute errors for each of them.

3.2. Approximation of $\log(x)$

The logarithm function can be efficiently approximated by exploiting the IEEE754 representation of floating point numbers [32]. The memory representation of any such number is functionally equivalent to

$$x = S2^E(1 + M), \tag{13}$$

where S is either -1 or $+1$, $E \in \mathbb{Z}$ is called the exponent, and $M \in [0, 1)$ is the mantissa.

In our case we can safely assume that $S = +1$, and by using basic properties of logarithms

$$\log(x) = \frac{1}{\log_2(e)}\left(E + \log_2(1 + M)\right). \tag{14}$$

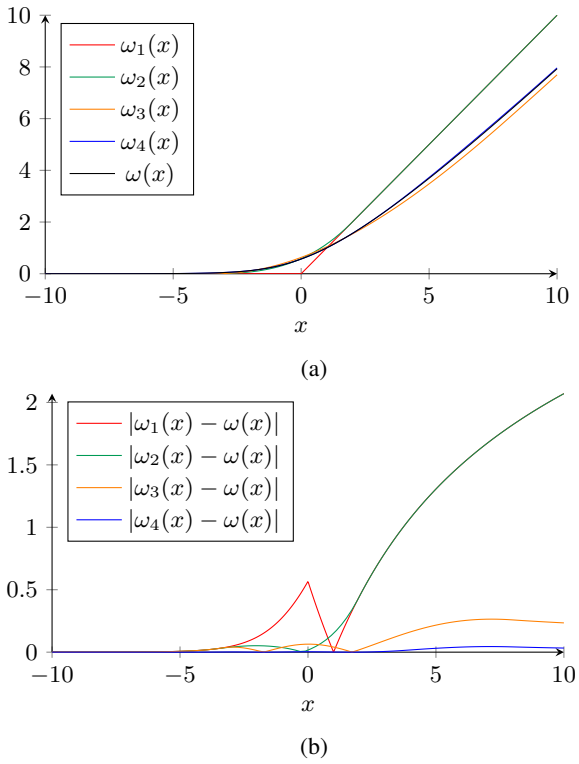


Figure 1: *Approximations of the Omega function (a) and their absolute errors (b).*

E can be easily extracted as an integer from the actual floating point representation of x by means of a bitshift and an integer sum, and can be then converted to floating point representation by routines that are usually hardware-provided. Similarly, $1 + M$ can be quickly obtained from x by setting the exponent to 0. Then, $\frac{1}{\log_2(e)}$ is a constant that should just be precomputed and multiplied for the change of base.

Since $1 \leq 1+M < 2$, we have effectively narrowed the problem to the computation of $\log_2(x)$ in this range. We can finally use a cubic spline and impose C^1 continuity on the extremes, thus obtaining

$$\log_2(x) \approx \alpha x^3 + \beta x^2 + \gamma x + \zeta, \tag{15}$$

with

$$\begin{aligned}\alpha &= 0.1640425613334452,\\ \beta &= -1.098865286222744,\\ \gamma &= 3.148297929334117,\\ \zeta &= -2.213475204444817.\end{aligned}$$

3.3. Approximation of e^x

A similar approach can be employed to approximate the exponential function. First, it is possible to express

$$e^x = 2^{\lfloor y \rfloor} 2^{y - \lfloor y \rfloor} \tag{16}$$

where $y = \frac{1}{\log(2)} x$ can be computed just by a multiplication with a precomputed constant. $2^{\lfloor y \rfloor}$ is also easily obtained in floating point format by converting y to integer format (usually through fast hardware-implemented routines) and ensuring down rounding is used, then using logic and integer operations so that $S = 1$, $E = \lfloor y \rfloor$, and $M = 0$.

Since $0 \leq y - \lfloor y \rfloor < 1$, we have again narrowed the problem to the computation of 2^x in this range. Now $y - \lfloor y \rfloor$ is obtained by converting $\lfloor y \rfloor$ to the floating point format and performing a subtraction, and finally a cubic spline with C^1 continuity on the extremes can be used to get

$$\begin{aligned}2^x &\approx \alpha x^3 + \beta x^2 + \gamma x + \zeta,\\ \alpha &= 0.07944154167983575,\\ \beta &= 0.2274112777602189,\\ \gamma &= 0.6931471805599453,\\ \zeta &= 1.\end{aligned}$$

4. APPLICATIONS

In this section we apply our approach to model two simple circuits, namely a common collector voltage buffer and a dynamic diode clipper, in order to show its suitability for simulating both static and dynamic nonlinear systems.

4.1. Common collector voltage buffer

The common collector configuration is a basic BJT amplifier topology, which is typically used as a voltage buffer. Figure 2 shows the simplest such circuit.

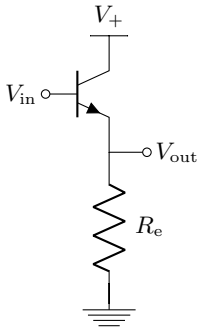


Figure 2: *Diagram of the common collector circuit.*

By examining the emitter node, according to the Ebers-Moll model and Ohm's Law,

$$I_{\text{s}}\left(e^{\frac{V_{\text{in}}-V_{\text{out}}}{V_{\text{T}}}} - e^{\frac{V_{\text{in}}-V_{+}}{V_{\text{T}}}} + \frac{e^{\frac{V_{\text{in}}-V_{\text{out}}}{V_{\text{T}}}} - 1}{\beta_{\text{f}}}\right) = \frac{V_{\text{out}}}{R_{\text{e}}}, \tag{17}$$

where I_{s} is the saturation current, V_{T} is the thermal voltage, and β_{f} is the common-emitter current gain. This can be solved analyt-

ically in terms of $\omega()$ as

$$V_{\text{out}} = V_{\text{T}}\omega\left(\frac{V_{\text{in}} + V_{\text{x}}}{V_{\text{T}}} + k\right) - V_{\text{x}}, \tag{18}$$

$$V_{\text{x}} = I_{\text{s}}R_{\text{e}}\left(e^{\frac{V_{\text{in}} - V_{+}}{V_{\text{T}}}} + \frac{1}{\beta_{\text{f}}}\right), \tag{19}$$

$$k = \log\left(\frac{I_{\text{s}}R_{\text{e}}}{V_{\text{T}}}\left(1 + \frac{1}{\beta_{\text{f}}}\right)\right). \tag{20}$$

Please note that k is constant and that the exponential in Eq. (19) is directly and solely input-dependent, therefore needing to be computed no matter the modelling method adopted. Moreover, the closed-form solution above is exact, therefore the output error will in practice only be affected by approximations of the $\omega()$ and the exponential functions.

Figure 3 shows the output signals obtained by feeding the proposed model implemented using different $\omega(x)$ approximations with a 9 V_{pp} (Volts peak-to-peak) 440 Hz sine with 4.5 V offset. We have set $V_{+} = 9$ V, $V_{\text{T}} = 26$ mV, $I_{\text{s}} = 0.1$ fA, $\beta_{\text{f}} = 100$, $R_{\text{e}} = 1$ kΩ.

4.2. Diode clipper

The diode clipper is a circuit that prevents the output from exceeding a predefined voltage level. Figure 4 shows a passive, dynamic version of the circuit that also incorporates a first-order lowpass filter. The behavior of the circuit can be fully described by [10]

$$\frac{\mathrm{d}V_{\text{out}}}{\mathrm{d}t} = \frac{V_{\text{in}} - V_{\text{out}}}{RC} - 2\frac{I_{\text{s}}}{C}\sinh\left(\frac{V_{\text{out}}}{V_{\text{T}}}\right), \tag{21}$$

where I_{s} is the saturation current and V_{T} is the thermal voltage.

The derivative on the left side of the previous equation can be discretized using any linear 1-step method (*e.g.*, trapezoidal rule) as

$$\mathrm{d}V_{\text{out}}[n] = B_0 V_{\text{out}}[n] + B_1 V_{\text{out}}[n-1] - A_1 \mathrm{d}V_{\text{out}}[n-1], \tag{22}$$

where B_0, B_1, A_1 are the coefficients obtained by applying the chosen discretization method. We will also approximate the behavior of the two antiparallel diodes by assuming that, at any time, the forward current of one is much higher than the reverse current of the other [17, 19, 20], which corresponds to substituting $\sinh(x) \approx \frac{1}{2}\text{sgn}(x)\left(e^{|x|} - 1\right)$. Therefore, we obtain

$$\begin{aligned} &B_0 V_{\text{out}}[n] + B_1 V_{\text{out}}[n-1] - A_1 \mathrm{d}V_{\text{out}}[n-1] = \\ &\frac{V_{\text{in}}[n] - V_{\text{out}}[n]}{RC} - \frac{I_{\text{s}}}{C}\text{sgn}\left(V_{\text{out}}[n]\right)\left(e^{\frac{|V_{\text{out}}[n]|}{V_{\text{T}}}} - 1\right), \end{aligned} \tag{23}$$

which can be analytically solved as

$$V_{\text{out}}[n] = w[n] - V_{\text{T}} r[n]\omega(k_4 r[n] w[n] + k_5), \tag{24}$$

$$w[n] = k_2 q[n] + k_3 r[n], \tag{25}$$

$$r[n] = \text{sign}(q[n]), \tag{26}$$

$$q[n] = k_1 V_{\text{in}}[n] - p[n-1], \tag{27}$$

$$p[n] = k_6 V_{\text{out}}[n] - A_1 p[n-1], \tag{28}$$

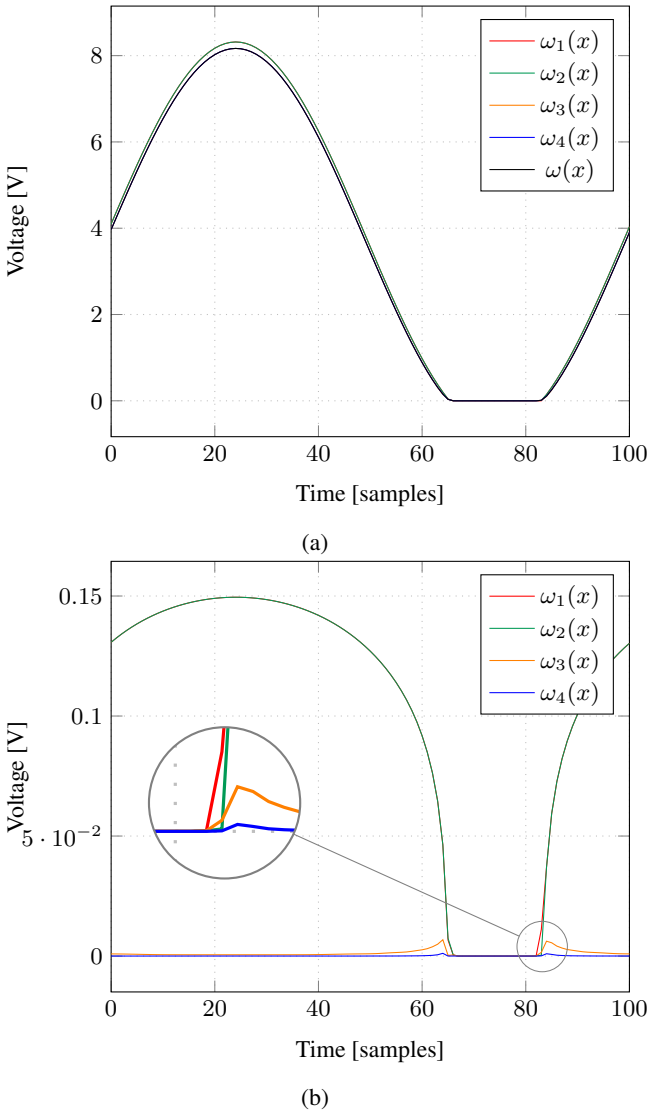


Figure 3: *Time domain waveforms (a) and absolute errors (b) of the output from the proposed common collector voltage buffer model implemented using the $\omega(x)$ approximations described in Section 3.1. The input is a 9 V_{pp} 440 Hz sine with 4.5 V offset sampled at 44.1 kHz.*

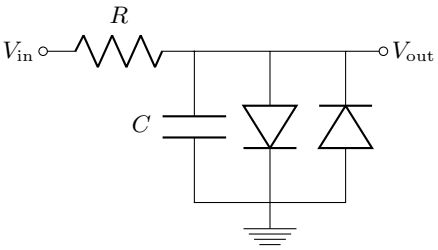


Figure 4: *Diagram of the diode clipper circuit.*

with

$$k_1 = \frac{1}{CR}, \tag{29}$$

$$k_2 = \frac{CR}{B_0 CR + 1}, \tag{30}$$

$$k_3 = \frac{I_\mathrm{s} R}{B_0 CR + 1}, \tag{31}$$

$$k_4 = \frac{1}{V_\mathrm{T}}, \tag{32}$$

$$k_5 = \log\left(\frac{I_\mathrm{s} R}{(B_0 CR + 1) V_\mathrm{T}}\right), \tag{33}$$

$$k_6 = B_1 - A_1 B_0. \tag{34}$$

Figure 5 shows the block diagram of the digital implementation, and Figure 6 shows the output signals obtained by feeding the proposed model implemented using different $\omega(x)$ approximations and running at a sample rate of 44.1 kHz with the sum of two 2 V_{pp} sines of frequencies 110 and 150 Hz. We have set $V_\mathrm{T} = 26$ mV, $I_\mathrm{s} = 0.1$ fA, $R = 2.2$ kΩ, $C = 10$ nF.

Unlike the approaches outlined in [10], our method appears to be stable when using common discretization methods for all inputs of all amplitudes and using any of the approximations of $\omega()$ presented so far. Such a favorable outcome was not unexpected, since the explicit solution of the implicit model, even if approximate and in the discrete-time domain, necessarily relaxes the stiffness constraints of the problem. Indeed, similar results were obtained when modelling related circuits by approximate solutions in [17, 19, 20].

The audio-rate computational load consists of 5 sums, 9 multiplications, 1 sign function, and 1 $\omega()$ evaluation per sample. While we could not directly compare the computational requirements of our algorithm to those presented in [10] for the same circuit, its number of operations is so limited that it can be safely assumed to be suitable for real-time usage on all but the worst performing platforms.

5. CONCLUSIONS

This paper proposed the reformulation of expressions involving the main branch of the Lambert W function in terms of the Wright Omega function in the context of virtual analog modelling. Such an approach has the advantage of eliminating the computation of the exponential term typically found in the argument. Simple approximations of the Omega function have also been proposed that are well suited in real-time contexts where lower accuracy can be traded for lower computational load.

Such an approach was applied to model two example circuits, namely a common collector voltage buffer and a diode clipper, resulting in high quality and high performance digital implementations. In the latter case, the approximate solution of the implicit model in the discrete-time domain also allowed to greatly improve the stability of the algorithm w.r.t. previous models [10].

Implementations of the two applications examples (in MATLAB) and of the approximations of $\omega(x)$, $\log(x)$, and e^x (in C) are available at `http://dangelo.audio/dafx2019-omega.html`.

6. REFERENCES

[1] R. C. D. de Paiva, *Circuit modeling studies related to guitars and audio processing*, Ph.D. thesis, Aalto University, Espoo, Finland, 2013.

[2] S. D'Angelo, *Virtual analog modeling of nonlinear musical circuits*, Ph.D. thesis, Aalto University, Espoo, Finland, 2014.

[3] G. Borin, G. De Poli, and D. Rocchesso, "Elimination of delay-free loops in discrete-time models of nonlinear acoustic systems," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 5, pp. 597–605, 2000.

[4] D. T. Yeh, J. S. Abel, and J. O. Smith, "Automated physical modeling of nonlinear audio circuits for real-time audio effects - Part I: Theoretical development," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 4, pp. 728–737, 2010.

[5] A. Fettweis, "Wave digital filters: Theory and practice," *Proceedings of the IEEE*, vol. 74, no. 2, pp. 270–327, 1986.

[6] S. Petrausch and R. Rabenstein, "Wave digital filters with multiple nonlinearities," in *Proceedings of the European Signal Processing Conference*, Wien, Austria, September 2004, pp. 77–80.

[7] K. J. Werner, V. Nangia, Smith J. O., and J. S. Abel, "Resolving wave digital filters with multiple/multiport nonlinearities," in *Proceedings of the International Conference on Digital Audio Effects*, Trondheim, Norway, November 2015, pp. 387–394.

[8] W. Shockley, "The Theory of p-n Junctions in Semiconductors and p-n Junction Transistors," *Bell System Technical Journal*, vol. 28, no. 3, pp. 435–489, 1949.

[9] J. J. Ebers and J. L. Moll, "Large-signal behavior of junction transistors," *Proceedings of the IRE*, vol. 42, no. 12, pp. 1761–1772, 1954.

[10] D. T. Yeh, J. Abel, and J. O. Smith, "Simulation of the diode limiter in guitar distortion circuits by numerical solution of ordinary differential equations," in *Proceedings of the International Conference on Digital Audio Effects*, Bordeaux, France, September 2007, pp. 197–204.

[11] F. Fontana and M. Civolani, "Modeling of the EMS VCS3 voltage-controlled filter as a nonlinear filter network," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 4, pp. 760–772, 2010.

[12] J. H. Lambert, "Observationes variae in mathesin puram," *Acta Helvetica*, vol. 3, no. 1, pp. 128–168, 1758.

[13] L. Euler, "De serie lambertina plurimisque eius insignibus proprietatibus," *Acta Academiae Scientiarum Imperialis Petropolitanae*, vol. 2, pp. 29–51, 1783.

[14] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth, "On the Lambert W function," *Advances in Computational Mathematics*, vol. 5, no. 1, pp. 329–359, 1996.

[15] T. C. Banwell and A. Jayakumar, "Exact analytical solution for current flow through diode with series resistance," *Electronics letters*, vol. 36, no. 4, pp. 291–292, 2000.

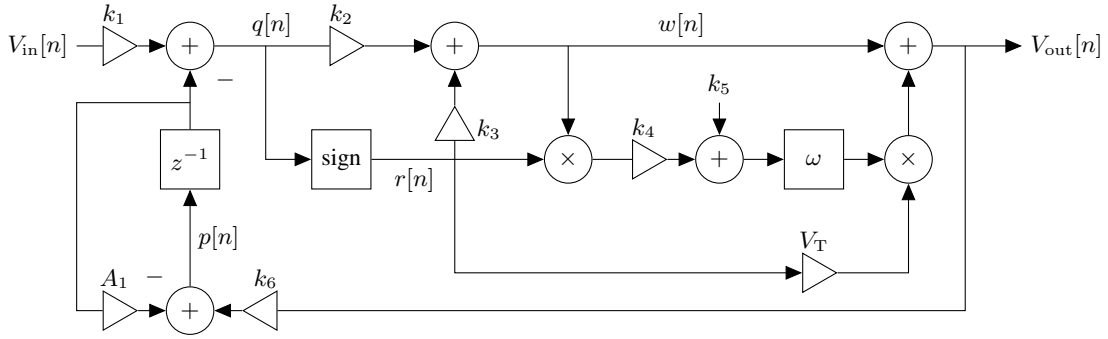


Figure 5: *Digital implementation of the diode clipper circuit.*

[16] J. Parker, "A simple digital model of the diode-based ring-modulator," in *Proceedings of the International Conference on Digital Audio Effects*, Paris, France, September 2011, vol. 14, pp. 163–166.

[17] R. C. D. Paiva, S. D'Angelo, J. Pakarinen, and V. Valimaki, "Emulation of operational amplifiers and diodes in audio distortion circuits," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 59, no. 10, pp. 688–692, 2012.

[18] J. Parker and S. D'Angelo, "A digital model of the Buchla lowpass-gate," in *Proceedings of the International Conference on Digital Audio Effects*, Maynooth, Ireland, September 2013, pp. 278–285.

[19] K. J. Werner, V. Nangia, A. Bernardini, J. O. Smith, and A. Sarti, "An improved and generalized diode clipper model for wave digital filters," in *Audio Engineering Society Convention 139*, New York, USA, October 2015.

[20] A. Bernardini, K. J. Werner, A. Sarti, and J. O. Smith, "Modeling nonlinear wave digital elements using the Lambert function," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 8, pp. 1231–1242, 2016.

[21] F. Esqueda, H. Pöntynen, J. Parker, and S. Bilbao, "Virtual analog models of the Lockhart and Serge wavefolders," *Applied Sciences*, vol. 7, no. 12, 2017.

[22] D. A. Barry, J. Y. Parlange, L. Li, H. Prommer, C. J. Cunningham, and F. Stagnitti, "Analytical approximations for real values of the Lambert W-function," *Mathematics and Computers in Simulation*, vol. 53, no. 1-2, pp. 95–103, 2000.

[23] F. Chapeau-Blondeau and A. Monir, "Numerical evaluation of the Lambert W function and application to generation of generalized Gaussian noise with exponent 1/2," *IEEE Transactions on Signal Processing*, vol. 50, no. 9, pp. 2160–2165, 2002.

[24] D. Veberič, "Having fun with Lambert W(x) function," *arXiv preprint arXiv:1003.1628*, 2010.

[25] D. Veberič, "Lambert W function for applications in physics," *Computer Physics Communications*, vol. 183, no. 12, pp. 2622–2628, 2012.

[26] R. Iacono and J. P. Boyd, "New approximations to the principal real-valued branch of the Lambert W-function," *Advances in Computational Mathematics*, vol. 43, no. 6, pp. 1403–1436, 2017.

[27] F. N. Fritsch, R. E. Shafer, and W. P. Crowley, "Algorithm 443: solution of the transcendental equation $w * e^W = x$," *Communications of the ACM*, vol. 16, pp. 123–124, 1973.

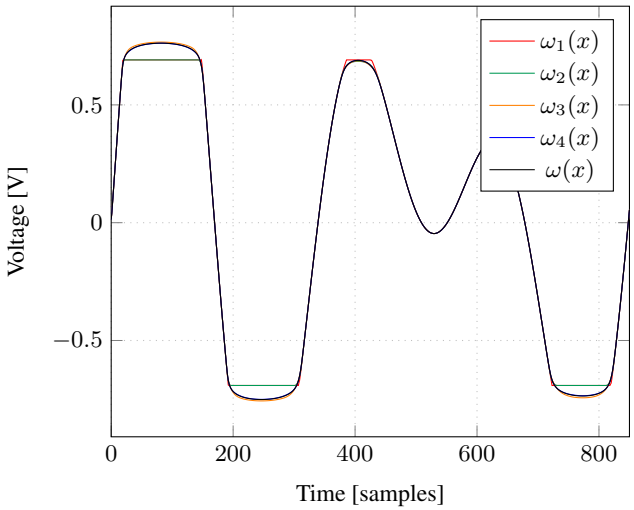
[28] D. A. Barry, S. J. Barry, and P. J. Culligan-Hensley, "Algorithm 743: WAPR – a Fortran routine for calculating real values of the W-function," *ACM Transactions on Mathematical Software*, vol. 21, no. 2, pp. 172–181, 1995.

[29] R. M. Corless and D. J. Jeffrey, "The Wright ω function," in *Artificial intelligence, automated reasoning, and symbolic computation*, pp. 76–89. Springer, 2002.

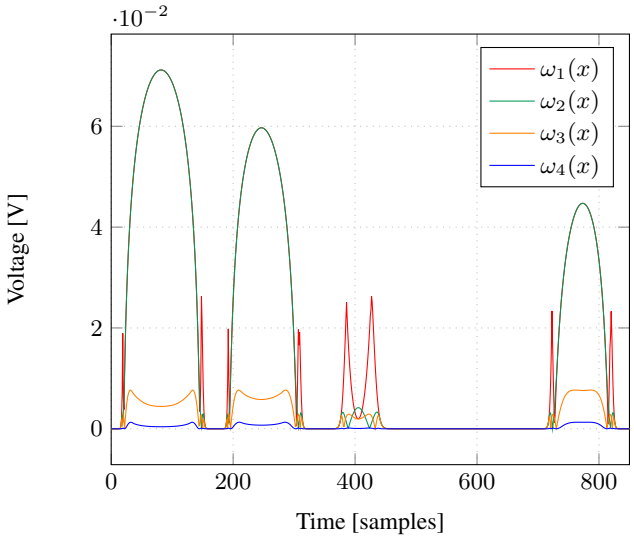
[30] R. M. Corless and D. J. Jeffrey, "Complex Numerical Values of the Wright ω function," 2004, available online at `http://www.orcca.on.ca/TechReports/2004/TR-04-04.html`.

[31] P. W. Lawrence, R. M. Corless, and D. J. Jeffrey, "Algorithm 917: Complex double-precision evaluation of the Wright ω function," *ACM Transactions on Mathematical Software*, vol. 38, no. 3, pp. 20, 2012.

[32] D. Goldberg, "What every computer scientist should know about floating-point arithmetic," *ACM Computing Surveys*, vol. 23, no. 1, pp. 5–48, 1991.



(a)



(b)

Figure 6: *Time domain waveforms (a) and absolute errors (b) of the output from the proposed diode clipper model implemented using the $\omega(x)$ approximations described in Section 3.1. The input is the sum of two 2 V_{pp} sines of frequencies 110 and 150 Hz sampled at 44.1 kHz.*

A MINIMAL PASSIVE MODEL OF THE OPERATIONAL AMPLIFIER : APPLICATION TO SALLEN-KEY ANALOG FILTERS

Rémy Müller

IRCAM-STMS (UMR 9912)
Sorbonne University
Paris, France
remy.muller@ircam.fr

Thomas Hélie *

IRCAM-STMS (UMR 9912)
Sorbonne University
Paris, France
thomas.helie@ircam.fr

ABSTRACT

This papers stems from the fact that, whereas there are passive models of transistors and tubes, a minimal passive model of the operational amplifier does not seem to exist. A new behavioural model is presented that is memoryless, fully described by its interaction ports, with a minimal number of equations, for which a passive power balance can be defined. The proposed model handles saturation, asymmetric power supply, and can be used with non-ideal voltage references. To illustrate the model in audio applications, the non-inverting voltage amplifier and a saturating Sallen-Key lowpass filter are considered.

1. INTRODUCTION

Operational Amplifier (OPA) models can be roughly categorized into a) Controlled Source (CS) models, b) white box macro models and c) Nullor models .

In CS models (see [1]), the power supplies are lumped within the OPA and controlled sources can provide an infinite amount of power. It has the advantage of being simple and hides most of the internal complexity. This is the method of choice used by students to study the functional behaviour of OPA circuits. The main drawback comes from the absence of external supply ports. This results in non passive models, and forbids simulations with non-ideal voltage sources (e.g. in low-budget guitar stomboxes).

White box macro models (see references [2] [3] [4]) use dozens of transistors to accurately reproduce the inner structure and non-ideal characteristics of particular devices. While this is appropriate for offline simulation and circuit design, the main drawback of this approach comes from the high number of (implicit) nonlinear equations which makes it often unsuitable for real-time simulation.

Nullors (see references [5] [6] [7] [8]), are singular two-port elements where the input flow and effort variables are both zero: $e_1 = f_1 = 0$, while the output flow and effort variables e_2, f_2 are unconstrained. One drawback is the lack of flow / effort duality. In addition, similar to CS, Nullors have no explicit power supply ports and thus are not passive devices, inheriting the same drawbacks mentioned above.

For audio applications, dedicated Wave Digital Filters (WDF) models of the OPA for specific circuit topologies have been proposed in [9], more recently, using Modified Nodal Analysis to WDF adaptors, both Nullor and CS general purpose models of the OPA and OTA have been proposed in [10] [11] and Sallen-key filters have been modelled with WDF in [12].

We propose a passive, quasi-ideal, black-box, behavioural model of the OPA, simple enough for realtime simulation, with explicit power supply and modelling nonlinear saturation. In particular, a by-product of this research is to have a model compatible with the port-Hamiltonian formalism [13].

The paper is structured as follows. First a general purpose passive model of the OPA is proposed in section 2, then it is illustrated by treating the non-inverting voltage amplifier circuit in section 3, finally a detailed study and simulation of a saturating Sallen-Key lowpass filter is presented in section 4.

2. OPERATIONAL AMPLIFIER MODEL

The objective of this paper is to find the simplest class of Operational Amplifier models satisfying the following properties:

a) Memoryless: infinite bandwidth, infinite slew rate,

b) Passivity: the power dissipated by the OPA is non-negative (i.e. hidden sources of energy are forbidden),

c) Quasi-ideal behaviour: infinite input impedance, zero output impedance, infinite common-mode rejection ratio,

d) Finite output voltage range and saturation: explicit non-constant power-supply ports,

e) Minimal: behavioural model with a minimum number of equations (i.e. not a white box model containing dozen of transistors).

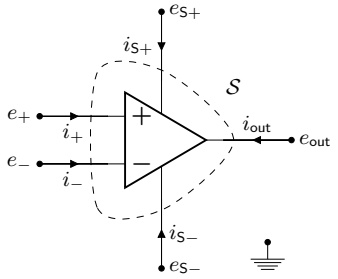


Figure 1: Circuit diagram of an Operational Amplifier (OPA) with currents drawn in receiver convention. The gaussian surface $\mathcal{S}$ enclosing the component is shown in dashed line.

* The author acknowledges the support of the ANR-DFG (French-German) project INFIDHEM ANR-16-CE92-0028.

2.1. Notations

The OPA shown on figure 1 is modelled as a 5-port device with node voltages being measured relatively to the ground, node currents directed toward the element using the receiver convention and pins labelled $\mathcal{P} = \{+, -, \mathsf{S}+, \mathsf{S}-, \mathsf{out}\}$. In this paper, we assume that the ports of the OPA can be partitioned into a voltage-driven set $\mathcal{T}$, and a current-controlled co-set $\mathcal{T}^*$

$$\mathcal{T} := \{+, -, \mathsf{S}+, \mathsf{S}-\}, \quad \mathcal{T}^* := \{\mathsf{out}\}, \quad \mathcal{T} \cup \mathcal{T}^* = \mathcal{P}. \tag{1}$$

The respective inputs and outputs are collected into the vectors

$$\mathbf{u} := [\mathbf{e}_{\mathcal{T}}, \mathbf{i}_{\mathcal{T}^*}]^{\mathsf{T}} = [e_+, e_-, e_{\mathsf{S}+}, e_{\mathsf{S}-}, i_{\mathsf{out}}]^{\mathsf{T}}, \tag{2}$$

$$\mathbf{y} := [\mathbf{i}_{\mathcal{T}}, \mathbf{e}_{\mathcal{T}^*}]^{\mathsf{T}} = [i_+, i_-, i_{\mathsf{S}+}, i_{\mathsf{S}-}, e_{\mathsf{out}}]^{\mathsf{T}}, \tag{3}$$

Finally, the common supply, the differential supply and the differential input voltages are respectively defined by

$$V_{\mathsf{cm}} = \frac{e_{\mathsf{S}+} + e_{\mathsf{S}-}}{2}, \quad V_{\mathsf{dm}} = \frac{e_{\mathsf{S}+} - e_{\mathsf{S}-}}{2}, \quad \epsilon = e_+ - e_-. \tag{4}$$

2.2. Constitutive equations

Since there are 5 ports with dual flow and efforts variables, 5 independent equations are required to specify the device:

1-2) **Non-energetic input ports**: the current entering the pins $\{+, -\}$ is zero (infinite input impedance)

$$i_+ = i_- = 0, \tag{5}$$

3) **Conservation of charge**: Kirchoff Current Law applied over the gaussian surface[1] $\mathcal{S}$ enclosing the AOP implies that the sum of all currents is zero

$$\sum_{\ell \in \mathcal{P}} i_\ell = 0, \tag{6}$$

4) **Passivity**: the power absorbed by the OPA is greater or equal to zero

$$P_{\mathsf{diss}} = \mathbf{y}^{\mathsf{T}}\mathbf{u} = \sum_{\ell \in \mathcal{P}} e_\ell \cdot i_\ell \geq 0, \tag{7}$$

5) **Differential gain and saturation**: the tensions are tied by a continuous relation $e_{\mathsf{out}} = f(e_+, e_-, e_{\mathsf{S}+}, e_{\mathsf{S}-})$ such that

$$\begin{cases} \dfrac{\partial f}{\partial \epsilon} \geq 0, & \text{monotonicity} \\ \max\left(\dfrac{\partial f}{\partial \epsilon}\right) = K, & \text{differential gain} \\ \max(f) = e_{\mathsf{S}+}, \ \epsilon \to +\infty & \text{positive saturation} \\ \min(f) = e_{\mathsf{S}-}, \ \epsilon \to -\infty & \text{negative saturation} \end{cases} \tag{8}$$

This gives 4 equalities and 1 inequality

$$i_+ = 0 \tag{9}$$

$$i_- = 0 \tag{10}$$

$$i_{\mathsf{S}+} + i_{\mathsf{S}-} + i_{\mathsf{out}} = 0 \tag{11}$$

$$P_{\mathsf{diss}} = i_{\mathsf{S}+} \cdot e_{\mathsf{S}+} + i_{\mathsf{S}-} \cdot e_{\mathsf{S}-} + i_{\mathsf{out}} \cdot e_{\mathsf{out}} \geq 0 \tag{12}$$

$$f(e_{\mathsf{S}+}, e_{\mathsf{S}-}, e_+, e_-) - e_{\mathsf{out}} = 0 \tag{13}$$

Since there is an inequality and the relation f is not specified yet, there is an infinite class of models satisfying these equations. A particular instance is chosen as follows.

[1] The Gaussian surface $\mathcal{S}$ is shown on figure 1. For more details see [1].

2.3. Toward a unique model

Substituting (4) into the passivity equation (12), using the conservation of charge (11) and simplifying by i_{out} gives the constraint[2]

$$V_{\mathsf{cm}} + V_{\mathsf{dm}} \left(\frac{i_{\mathsf{S}+} - i_{\mathsf{S}-}}{i_{\mathsf{S}+} + i_{\mathsf{S}-}} \right) = e_{\mathsf{out}} - \frac{P_{\mathsf{diss}}}{i_{\mathsf{out}}}, \quad (i_{\mathsf{out}} \neq 0) \tag{14}$$

which imposes a lot of structure on the form of the output function. In order to specify a unique model, the following choices are made.

2.3.1. Differential input transistor pair

First, motivated by the typical structure of an OPA, composed of a differential pair of transistors, gain stages and a push-pull output (see [14] p.707), the adimensioned modulation factor [3]

$$\rho(\epsilon) := -\frac{i_{\mathsf{S}+}}{i_{\mathsf{out}}} = \frac{\exp(x)}{\exp(x) + \exp(-x)}, \quad x = \frac{K\epsilon}{V_{\mathsf{dm}}}, \tag{15}$$

is introduced and shown on figure 2. According to the conservation of charge (11), this leads to the symmetric current splitting

$$i_{\mathsf{S}+} = -\rho(\epsilon) i_{\mathsf{out}}, \qquad i_{\mathsf{S}-} = -\rho(-\epsilon) i_{\mathsf{out}}. \tag{16}$$

2.3.2. The conservative OPA choice

Second, among all passive OPA models, the conservative ones are chosen, neglecting internal dissipation:

$$P_{\mathsf{diss}} = 0. \tag{17}$$

The power supply ports provide the amount of power necessary to balance the power consumed at the output port. This is an instance of a nonlinear nonenergic n-port [15].

2.3.3. Final model

Substituting (16) and (17) into (14) uniquely defines the output function (a similar result was also derived in [16])

$$e_{\mathsf{out}} = V_{\mathsf{cm}} + V_{\mathsf{dm}} \tanh\left(\frac{K\epsilon}{V_{\mathsf{dm}}}\right). \tag{18}$$

Expressed as a function of $e_{\mathsf{S}+}, e_{\mathsf{S}-}$ this gives

$$e_{\mathsf{out}} = \rho(+\epsilon) e_{\mathsf{S}+} \ + \ \rho(-\epsilon) e_{\mathsf{S}-}. \tag{19}$$

Finally gathering equations (5) (16) (19) in matrix form reveals the modulated hybrid Dirac structure[4] of the conservative OPA model given by the skew-symmetric matrix $\mathbf{J}(\mathbf{u})$:

$$\underbrace{\begin{bmatrix} i_+ \\ i_- \\ i_{\mathsf{S}+} \\ i_{\mathsf{S}-} \\ e_{\mathsf{out}} \end{bmatrix}}_{\mathbf{y}} = \underbrace{\left[\begin{array}{cc|ccc} . & . & . & . & . \\ . & . & . & . & . \\ \hline . & . & . & . & -\rho(+\epsilon) \\ . & . & . & . & -\rho(-\epsilon) \\ . & . & \rho(\epsilon) & \rho(-\epsilon) & . \end{array}\right]}_{\mathbf{J}(\mathbf{u})} \underbrace{\begin{bmatrix} e_+ \\ e_- \\ e_{\mathsf{S}+} \\ e_{\mathsf{S}-} \\ i_{\mathsf{out}} \end{bmatrix}}_{\mathbf{u}}. \tag{20}$$

The singularity of the structure matrix $\mathbf{J}$ encodes the conservation of the so-called Casimir invariants $i_+ = i_- = 0$, in addition to the conservative power-balance

$$P_{\mathsf{diss}} = \mathbf{u}^{\mathsf{T}}\mathbf{y} = \mathbf{u}^{\mathsf{T}}\mathbf{J}(\mathbf{u})\mathbf{u} = 0, \quad (\mathbf{J} = -\mathbf{J}^{\mathsf{T}}). \tag{21}$$

[2] see appendix A for a detailed proof.

[3] Different choices can be made here to adapt to other transistors types.

[4] Please refer to the references [17] [18] [13] for more details on Dirac structures and to [1] for hybrid parameters.

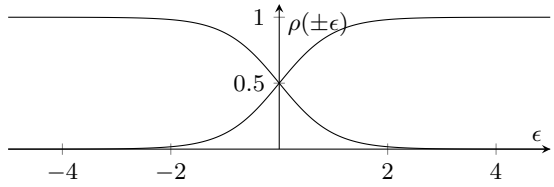
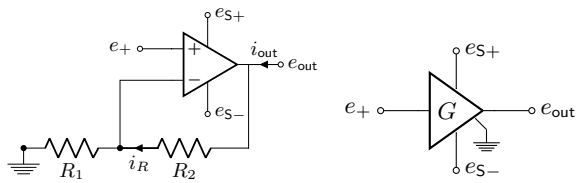


Figure 2: The modulation factor $\rho(\pm\epsilon)$, for $K = 1$, $V_{\mathsf{dm}} = 1$.

3. CASE STUDY

To study the behaviour of the proposed model in practical applications, the case of the voltage amplifier is examined in section 3.1. Then as a pedagogical example, the voltage amplifier is driven by a sinusoidal voltage source and asymmetrically powered by a single capacitor to simulate a discharging battery in section 3.2. The voltage amplifier will be used as a building block of the Sallen-Key lowpass filter shown in section 4.

3.1. The non-inverting voltage amplifier



(a) (b)

Figure 3: a) a non-inverting voltage amplifier circuit with explicit alimentation ports and b) its symbol.

A non-inverting voltage amplifier (figure 3) is achieved by feeding back the output e_{out} to the negative input e_- through a voltage divider

$$\epsilon = e_+ - \frac{e_{\mathsf{out}}}{G}, \qquad G = \frac{R_1 + R_2}{R_1} = 1 + \frac{R_2}{R_1}. \tag{22}$$

The instantaneous feedback makes the circuit act as a proportional corrector with high proportional gain K in order to satisfy the constraint $e_{\mathsf{out}} \approx G e_+$ within the range $e_{\mathsf{out}} \in [e_{\mathsf{S}+}, e_{\mathsf{S}-}]$.

The voltage divider induces an internal current $i_R = e_{\mathsf{out}}/R$, where $R = R_1 + R_2$, and the current splitting (16) becomes

$$i_{\mathsf{S}+} = -\rho(\epsilon)(i_{\mathsf{out}} - i_R), \quad i_{\mathsf{S}-} = -\rho(-\epsilon)(i_{\mathsf{out}} - i_R). \tag{23}$$

This results in the following law for the voltage amplifier

$$\begin{bmatrix} i_+ \\ i_{\mathsf{S}+} \\ i_{\mathsf{S}-} \\ e_{\mathsf{out}} \end{bmatrix} = \left[\begin{array}{c|cc|c} . & . & . & . \\ \hline . & g_+(\epsilon) & g_\pm(\epsilon) & -\rho(\epsilon) \\ . & g_\pm(\epsilon) & g_-(\epsilon) & -\rho(-\epsilon) \\ \hline . & \rho(\epsilon) & \rho(-\epsilon) & . \end{array}\right] \begin{bmatrix} e_+ \\ e_{\mathsf{S}+} \\ e_{\mathsf{S}-} \\ i_{\mathsf{out}} \end{bmatrix}. \tag{24}$$

with conductances

$$g_+(\epsilon) = \frac{\rho(\epsilon)^2}{R}, \; g_-(\epsilon) = \frac{\rho(-\epsilon)^2}{R}, \; g_\pm(\epsilon) = \frac{\rho(\epsilon)\rho(-\epsilon)}{R}. \tag{25}$$

In the following, it is assumed that $R \to \infty$ such that internal losses are negligible. In particular, this is the case of the classical voltage follower circuit for which $R_2 = 0$, and $R_1 = \infty$.

3.1.1. Implicit constraint

The relation (24) is still implicitly defined since ϵ depends on both input and output variables e_+ and e_{out}. To avoid apparent difficulties with discontinuous functions, consider the curve

$$\mathcal{F} = \left\{ (u, y) \in \mathbb{R}^2 \mid F(u, y) = 0 \right\}, \tag{26}$$

specified by the function

$$F(u, y) = V_{\mathsf{cm}} + V_{\mathsf{dm}} \tanh\left(\frac{K}{V_{\mathsf{dm}}} \left(u - \frac{y}{G} \right) \right) - y, \tag{27}$$

and given e_+, look for e_{out} such that $(e_+, e_{\mathsf{out}}) \in \mathcal{F}$.

Since the output function is monotonous with respect to ϵ and bounded in $[e_{\mathsf{S}-}, e_{\mathsf{S}+}]$, a unique solution exists within that range. A global method such as the bisection method is guaranteed to find it, whereas, since K is typically about 10^6, it is very difficult to use either fixed-point or derivative-based methods because of bad numerical conditioning. Numerical simulations are shown on figure 4.

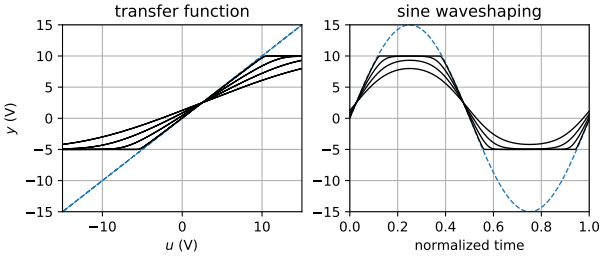


Figure 4: Transfer function of the voltage amplifier for $G = 1$, $K \in \{1, 2, 5, 50\}$, $e_{\mathsf{S}+} = 10\text{V}$, $e_{\mathsf{S}-} = -5\text{V}$. Smaller values than the typical OPA gain $K \approx 10^6$ are used for visualisation purposes.

3.1.2. Explicit representation

Taking the limit when $K \to \infty$ gives an explicit representation of $\mathcal{F}$ as the piecewise continuous curve

$$\mathcal{F}_\infty = \lim_{K\to\infty} \mathcal{F} : \begin{cases} y = e_{\mathsf{S}+}, & Gu > y \\ y = e_{\mathsf{S}-}, & Gu < y \\ y \in [e_{\mathsf{S}-}, e_{\mathsf{S}+}], & y = Gu \end{cases}. \tag{28}$$

One can see on figure 4 that convergence to $\mathcal{F}_\infty$ is very fast even for moderate values of K. This justifies the use of this limit process in following developments.

For $(e_+, e_{\mathsf{out}}) \in \mathcal{F}_\infty$ this gives the explicit form

$$e_{\mathsf{out}} = V_{\mathsf{cm}} + V_{\mathsf{dm}} \operatorname{sat}\left(\frac{G e_+ - V_{\mathsf{cm}}}{V_{\mathsf{dm}}} \right), \tag{29}$$

where

$$\operatorname{sat}(x) = \min(\max(x, -1), 1). \tag{30}$$

Alternatively one can represent this function as

$$e_{\mathsf{out}} = \mu_+(e_+, V_{\mathsf{cm}}, V_{\mathsf{dm}}) e_{\mathsf{S}+} + \mu_-(e_+, V_{\mathsf{cm}}, V_{\mathsf{dm}}) e_{\mathsf{S}-} \tag{31}$$

where the implicit modulation factor $\rho(\pm\epsilon)$ in (24) has been replaced by the explicit one

$$\mu_\pm(e_+, V_{\mathsf{cm}}, V_{\mathsf{dm}}) = \frac{1 \pm \operatorname{sat}(x)}{2}, \quad x = \frac{G e_+ - V_{\mathsf{cm}}}{V_{\mathsf{dm}}}. \tag{32}$$

3.2. A single-rail voltage follower powered by a capacitor

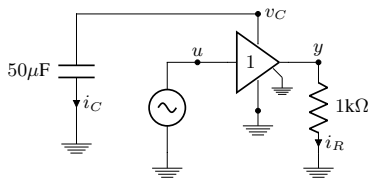


Figure 5: A single-rail voltage amplifier powered by a capacitor.

To illustrate one of the practical interest of having explicit power supply ports, the voltage amplifier is used with the negative supply port grounded, and the positive supply port powered by a capacitor to simulate a discharging battery (figure 5).

Using (20) with $V_{\text{cm}} = V_{\text{dm}} = q/(2C)$, and $i_{\text{out}} = -y/R$, yields the algebro-differential equations

$$\begin{cases} \dot{q} = -\eta(u,q)\dfrac{y}{R}, \\ y = \eta(u,q)\dfrac{q}{C} \end{cases}, \quad \eta(u,q) = \mu_+\left(u, \frac{q}{2C}, \frac{q}{2C}\right). \tag{33}$$

The energy stored in the capacitor is $H(q) = q^2/2C$. Then its differential equation is governed by the monotonous discharge

$$\frac{\mathrm{d}}{\mathrm{d}t}H(q) = \frac{\partial H}{\partial q}\frac{\mathrm{d}q}{\mathrm{d}t} = -\frac{q}{C}\eta(q,u)\frac{y}{R} = -\frac{y^2}{R}. \tag{34}$$

The circuit acts as a half-wave rectifier with a positive clipping threshold governed by the discharge of the capacitor as shown on figure 6.

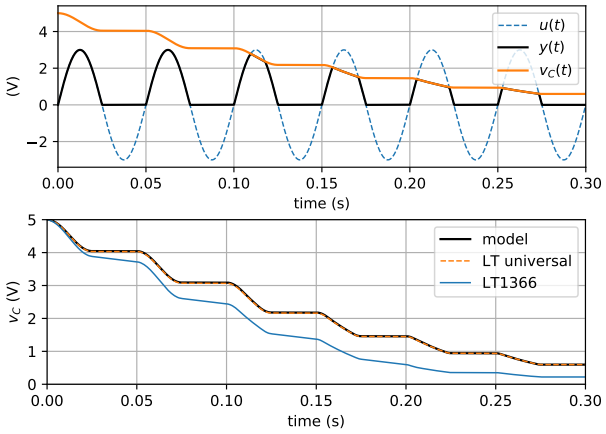


Figure 6: Time domain simulation of the capacitor-powered single rail voltage amplifier with $v_C(0) = 5$V and $|u| = 3V$. Top plot: proposed model. Bottom plot: comparison of discharge rate with LTspice's Universal OPA level.2 and the LT1366 [19].

Remark (Comparison between models)

As expected, with the proposed model, the capacitor does not discharge during negative saturation (energy-preservation), and has a monotonous discharge otherwise. Comparison with LTspice's universal model shows that the two simulations are very close. Finally with the LT1366, the discharge is monotonous and qualitatively similar, but decays faster due to internal dissipation.

4. SALLEN-KEY ANALOG LOWPASS FILTER

The class of Sallen-Key Filters (SKF), introduced in [20], is perhaps one of the most common analog filter design topology. It is used for the realization of analog biquadratic filters, for example in parametric equalisers. It is also the basis of the multimode Steiner filter [21], the Korg MS-20 [22] and the Buchla Lowpass-Gate [23].

A Sallen-Key lowpass filter schematic is shown on figure 8a. The linear regime and its control parameters are studied in 4.1, the circuit is then converted into equations in 4.2. Discretization is performed using the Average Vector Field method in 4.3, finally simulation results are shown in 4.4.

4.1. Linear behaviour and control parameters

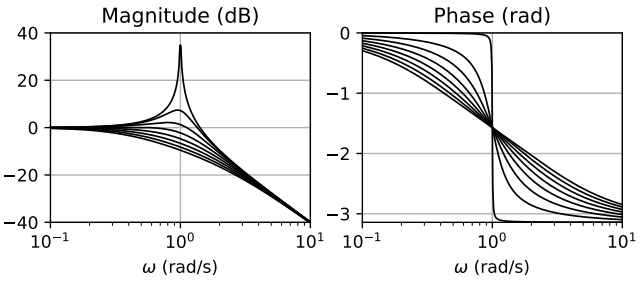


Figure 7: Bode plot of the Sallen-Key filter for $\omega = 1$, $G \in [0, 3]$

It is recalled that the Laplace transfer function (shown on figure 7) of a second order resonant lowpass filters with pulsation ω and quality factor Q is

$$H_{\text{LP}}(s) = \frac{1}{1 + \frac{1}{Q}\left(\frac{s}{\omega}\right) + \left(\frac{s}{\omega}\right)^2}, \tag{35}$$

In the linear regime, the Laplace transfer function of the lowpass Sallen-Key filter is

$$H_{\text{SK}}(s) = \mathcal{L}\left\{\frac{y_{\text{SK}}}{v_{\text{IN}}}\right\} = \frac{1}{1 + a_1 s + a_2 s^2}, \tag{36}$$

where

$$a_1 = \big((1-G)R_1C_1 + (R_1 + R_2)C_2\big), \tag{37}$$

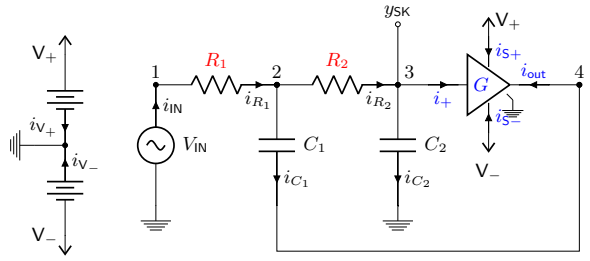
$$a_2 = C_1C_2R_1R_2. \tag{38}$$

Since there are only two target controls (ω, Q), for 5 design parameters (R_1, R_2, C_1, C_2, G), there are many possible design decisions that are often decided according to electronic constraints.

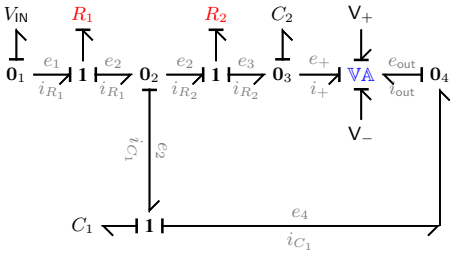
In this paper, the Steiner filter parametrization is used with $R_1 = R_2 = R$, and $C_1 = C_2 = C$ because of its simplicity. The transfer function (36) simplifies to

$$H_{\text{SK}}(s) = \frac{1}{1 + (3-G)\left(\frac{s}{\omega}\right) + \left(\frac{s}{\omega}\right)^2}, \tag{39}$$

with $\omega = 1/(RC)$, and $Q = 1/(3-G)$. In simulations, capacitances are both set to $C = 4.7$nF and the resistors are adjusted to achieve the target cutoff frequencies.



(a) Circuit



(b) Bondgraph

	V_{IN}	v_{C_1}	v_{C_2}	V_+	V_-	i_{R_1}	i_{R_2}	i_+	i_{S+}	i_{S-}	e_{out}
i_{IN}						−1	.	.	.	.	
i_{C_1}						1	−1	.	.	.	
i_{C_2}						.	1	−1	.	.	
i_{V_+}						.	.	.	−1	.	
i_{V_-}						.	.	.	.	−1	
v_{R_1}	1	−1	.	.	.						−1
v_{R_2}	.	1	−1	.	.						1
e_+	.	.	1	.	.						.
e_{S+}	.	.	.	1	.						.
e_{S-}	.	.	.	.	1						.
i_{out}						1	−1	.	.	.	

(c) Skew-symmetric Dirac structure (KCL+KVL)

$$\begin{pmatrix} i_{C_1} \\ i_{C_2} \\ \hline i_{IN} \\ i_{out} \end{pmatrix} = \underbrace{\frac{1}{R}\left(\begin{array}{cc|cc} -2 & 1 & 1 & -2 \\ 1 & -1 & 0 & 1 \\ \hline 1 & 0 & -1 & 1 \\ -2 & 1 & 1 & -2 \end{array}\right)}_{Y} \begin{pmatrix} v_{C_1} \\ v_{C_2} \\ \hline v_{IN} \\ e_{out} \end{pmatrix}$$

(d) Reduced model (admittance form)

Figure 8: a) The original Sallen-Key lowpass filter circuit, b) its corresponding bondgraph (see references [24] [25] [26]) with computational causality assignment. c) the skew-symmetric Dirac structure representing Kirchoff conservation laws. d) the reduced dynamical model.

4.2. Modelling

To model the Sallen-Key filter, the following systematic approach is used:

- **Bondgraph**: The circuit 8a is first converted to an equivalent bondgraph 8b using the rules in [25]. A bond between two ports **A** ⟶ **B** stands for a pair of dual port-variables (e, f). The half-arrow indicates the power sign convention $P = ef \geq 0$. **0** denotes a parallel junction where all bonds share the same voltage, and **1** denotes a serial junction where all bonds share the same current.
- **Causality assignment**: to convert an acausal bidirectional bondgraph to a causal, computable, block-diagram, one needs to partition the flows and efforts into inputs and outputs. The convention uses a vertical stroke **A** ⟼| **B** next to ports that are effort-controlled. *Computational causalities* can be assigned graphically by propagating the following rules: voltage sources and capacitors have an effort-out causality, **0** junctions can only have one input effort, while the dual **1** junctions can only have one output effort.
- **Dirac Structure**: given the causality assignment, shown on 8b, into inputs and outputs, it is now straightforward to fill the Dirac Structure matrix 8c by inspecting circuit 8a and expressing Kirchoff's current and voltage laws.
- **Reduced model**: one can reduce the model by solving trivial equalities like $e_+ = v_{C_2}$, $e_{S+} = V_+$, $e_{S-} = V_-$, treating $V_\pm$ as constants and replacing the linear resistive currents (i_{R_1}, i_{R_2}) by their constitutive laws. This results in the reduced admittance model shown on figure 8d.

4.2.1. *Nonlinear feedback*

To separate the linear and nonlinear feedback, one can write

$$\hat{e}_{out}(v) = Gv - \nabla N(v) \tag{40}$$

where the nonlinear law is

$$\begin{aligned} \nabla N(v) &:= Gv - \hat{e}_{out}(v) \\ &= \min(0, Gv - e_{S-}) + \max(0, Gv - e_{S+}). \end{aligned} \tag{41}$$

and its algebraic potential (figure 9) is given by the line integral

$$\begin{aligned} N(v) &:= \int_0^v \nabla N(s) \cdot \mathrm{d}s \\ &= \frac{\min(0, Gv - e_{S-})^2}{2G} + \frac{\max(0, Gv - e_{S+})^2}{2G}. \end{aligned} \tag{42}$$

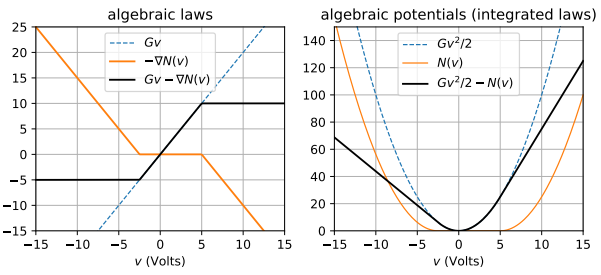


Figure 9: Algebraic feedback laws and their potentials shown for $G = 2$, $e_{S+} = 10\text{V}$, $e_{S-} = -5\text{V}$.

4.2.2. State-space model

Finally replacing the flow and effort variables by their constitutive laws, and only considering the input-state-output, one gets

$$\begin{cases} \dot{\mathbf{x}} = \omega \left[\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} - \mathbf{F}\nabla N(\mathbf{C}\mathbf{x}) \right] \\ \mathbf{y} = \mathbf{C}\mathbf{x} \end{cases}, \tag{43}$$

where $\mathbf{u} = v_{\mathsf{IN}}$, $\mathbf{y} = y_{\mathsf{SK}}$, $\mathbf{x} = [v_{C_1}, v_{C_2}]^{\mathsf{T}}$, $\omega = 1/(RC)$ and

$$\mathbf{A} = \begin{bmatrix} -2 & 1-2G \\ 1 & -1+G \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \tag{44}$$

$$\mathbf{C} = \begin{bmatrix} 0 & 1 \end{bmatrix}, \qquad \mathbf{F} = \begin{bmatrix} -2 \\ 1 \end{bmatrix}. \tag{45}$$

Using the co-energy variables v_{C_1}, v_{C_2} instead of the energy variables q_{C_1}, q_{C_2} is justified here by the fact that the capacitors are linear and time-invariant, i.e. the co-energy $H^*(v) = Cv^2/2$ equals the energy $H(q) = q^2/(2C)$ for the linear law $v = q/C$.

4.3. Discretization using the AVF method

The Average Vector Field (AVF) method is used to discretize (43) because of its structure-preserving properties: it preserves the energy (resp. dissipativity) of conservative (resp. dissipative) systems (see [27]). One can also refer to [28] where it has been shown that the bilinear transform doesn't always guarantee the dissipativity of nonlinear filters (whether time-varying or not).

As an important side-effect, the AVF method can also be interpreted as a first-order instance of anti-derivative antialiasing [29].

4.3.1. The Average Vector Field method

Let $\Omega = [t_0, t_0 + h]$ be a time-step, $\mathbf{x} : \Omega \to \mathbb{R}^n$ a locally affine trajectory parametrized by the normalized variable $\tau \in [0, 1]$

$$\mathbf{x}(t_0 + h\tau) = \mathbf{x}_0 + \tau(\mathbf{x}_1 - \mathbf{x}_0). \tag{46}$$

Introduce the averaging operator $\mathcal{A}$, defined for all functions $f : \mathbb{R}^n \to \mathbb{R}^n$ or operators $f : \mathcal{H} \to \mathcal{H}$, where $\mathcal{H}$ is a functional space from $\Omega \to \mathbb{R}^n$, by

$$(\mathcal{A}\, f)(\mathbf{x}) := \int_0^1 f(\mathbf{x}(t_0 + h\tau))\, \mathrm{d}\tau. \tag{47}$$

For the time derivative and identity operators, one gets

$$\overline{\dot{\mathbf{x}}} := \left(\mathcal{A} \tfrac{\mathrm{d}}{\mathrm{d}t} \right) \mathbf{x} = \frac{\mathbf{x}_1 - \mathbf{x}_0}{h}, \quad \bar{\mathbf{x}} := (\mathcal{A}\mathcal{I})\mathbf{x} = \frac{\mathbf{x}_0 + \mathbf{x}_1}{2}. \tag{48}$$

Using the gradient theorem, this gives the average discrete gradient

$$\begin{aligned} \overline{\nabla} N(v_0, v_1) &:= (\mathcal{A}\nabla N)(v_0 + \tau(v_1 - v_0)) \\ &= \begin{cases} \dfrac{N(v_1) - N(v_0)}{v_1 - v_0} & v_0 \neq v_1 \\ \nabla N(v_0) & v_0 = v_1 \end{cases}. \end{aligned} \tag{49}$$

Computing its derivative with respect to v_1 leads to

$$\frac{\partial \overline{\nabla} N}{\partial v_1}(v_0, v_1) = \begin{cases} \dfrac{\nabla N(v_1) - \overline{\nabla} N(v_0, v_1)}{v_1 - v_0} & v_0 \neq v_1 \\ \dfrac{1}{2}\nabla^2 N(v_0) & v_0 = v_1 \end{cases}. \tag{50}$$

One can refer to [30], where the discrete gradient's derivative is also used for numerical simulation.

4.3.2. Averaged system

Applying the averaging operator $\mathcal{A}$ to (43), leads to the structure-preserving discrete algebraic system

$$\begin{cases} \overline{\dot{\mathbf{x}}} = \omega \left[\mathbf{A}\bar{\mathbf{x}} + \mathbf{B}\bar{\mathbf{u}} - \mathbf{F}\, \overline{\nabla} N(\mathbf{C}\mathbf{x}_0, \mathbf{C}\mathbf{x}_1) \right] \\ \bar{\mathbf{y}} = \mathbf{C}\bar{\mathbf{x}} \end{cases}. \tag{51}$$

Solving the linear part for $\mathbf{x}_1$ gives the discrete state-space update

$$\mathbf{x}_1 = \mathbf{A}_d\mathbf{x}_0 + \mathbf{B}_d\bar{\mathbf{u}} - \mathbf{F}_d\, \overline{\nabla} N(\mathbf{C}\mathbf{x}_0, \mathbf{C}\mathbf{x}_1), \tag{52}$$

with the normalised pulsation $\omega_d = h\omega$ and

$$\begin{aligned} \mathbf{A}_d &= \mathbf{D}^{-1}\left(\mathbf{I} + \frac{\omega_d}{2}\mathbf{A} \right), & \mathbf{B}_d &= \mathbf{D}^{-1}(\omega_d \mathbf{B}), \\ \mathbf{D} &= \left(\mathbf{I} - \frac{\omega_d}{2}\mathbf{A} \right), & \mathbf{F}_d &= \mathbf{D}^{-1}(\omega_d \mathbf{F}). \end{aligned} \tag{53}$$

4.4. Simulation

Simulation results[5] are shown on figures 10 and 11 and exhibit a very close match with offline simulations performed in LTspice. To solve (52), one can either use the simple fixed-point iteration, or Newton's method.

4.4.1. Fixed-point iteration

A simple numerical scheme is to look for the fixed-point $\mathbf{x}_1 = \phi(\mathbf{x}_1)$ of the pre-conditioned fixed-point function

$$\phi(\mathbf{x}_1) := \mathbf{A}_d\mathbf{x}_0 + \mathbf{B}_d\bar{\mathbf{u}} - \mathbf{F}_d\, \overline{\nabla} N(\mathbf{C}\mathbf{x}_0, \mathbf{C}\mathbf{x}_1), \tag{54}$$

with the fixed-point iteration

$$\mathbf{x}_1^{k+1} = \phi\left(\mathbf{x}_1^k \right), \quad \mathbf{x}_1^0 = \mathbf{x}_0. \tag{55}$$

A sufficient convergence condition is detailed in appendix B.

In practice, thanks to the non linear feedback splitting in (40), when the OPA is in the linear regime, $\nabla N = 0$. Then the iteration reduces to an explicit one-step trapezoidal integrator and converges in only one iteration.

4.4.2. Newton iteration

To accelerate convergence, one can use Newton's method [31] as follows: define the auxiliary function

$$\varphi(\mathbf{x}_1) = \mathbf{x}_1 - \phi(\mathbf{x}_1), \tag{56}$$

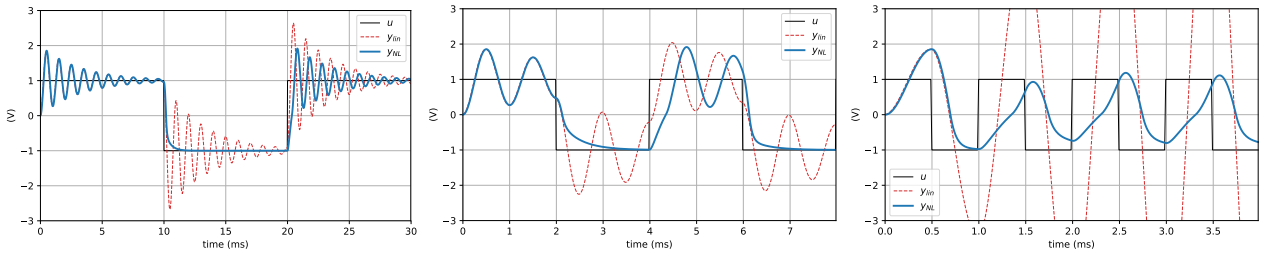
and look for the root $\mathbf{x}_1^*$ such that $\varphi(\mathbf{x}_1^*) = 0$ with the Newton iteration

$$\mathbf{x}_1^{k+1} = \mathbf{x}_1^k - \left(\varphi'(\mathbf{x}_1^k) \right)^{-1} \varphi(\mathbf{x}_1^k), \quad \mathbf{x}_1^0 = \mathbf{x}_0. \tag{57}$$

where the Jacobian of φ is given by

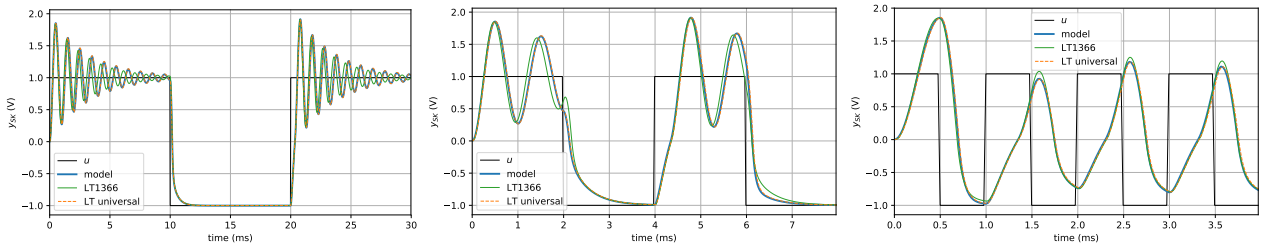
$$\varphi'(\mathbf{x}_1) = \mathbf{I} + \mathbf{F}_d\mathbf{C}\, \frac{\partial \overline{\nabla} N}{\partial v_1}(\mathbf{C}\mathbf{x}_0, \mathbf{C}\mathbf{x}_1). \tag{58}$$

[5] Sound examples and LTspice files are available at the accompanying website: `https://github.com/remymuller/dafx19-opa`.



(a) $f_0 = 50$Hz (b) $f_0 = 250$Hz (c) $f_0 = 1$kHz

Figure 10: SKF filter response to a square wave input with sampling frequency $f_s = 44.1$kHz, $C = 4.7$nF, cutoff $f_c = 1$kHz ($R = 33.8k\Omega$), $Q = 10$, asymmetric saturation $V_+ = 15$V, $V_- = 0$V and different fundamental frequencies. The non linear SKF response is shown in solid blue, with the linear SKF response in dashed red for reference.



(a) $f_0 = 50$Hz (b) $f_0 = 250$Hz (c) $f_0 = 1$kHz

Figure 11: Comparison between the proposed model, LTspice's universal OPA level.2 and the LT1366 opamp. The proposed model output is almost indistinguishable from LTspice's universal model, whereas the tuning of the LT1366 is slightly different because of dissipation.

5. CONCLUSIONS AND PERSPECTIVES

In this paper, a static, passive, black-box model of the operational amplifier with explicit power supply has been examined. It is suitable for the modelling of audio circuits and simple enough for real-time simulation. Furthermore the explicit modelling of external power supply ports allows the use of non-ideal voltage sources.

The choice has been made to ignore internal dissipation to keep the model minimal. However, non-ideal characteristics such as input and output impedance or power supply voltage drop can be achieved by modular composition of the model with other circuit elements. This will be the topic of further research.

The non inverting amplifier is also derived as a dedicated building block. Numerical simulations justify the use of an infinite OPA gain to get an explicit formulation. Having a pre-solved amplifier model also greatly simplifies its use in electronic circuits, avoiding numerical stiffness and high index DAE.

Finally, the amplifier is used for audio simulations to model a saturating Sallen-Key lowpass filter of second order. A reduced state-space model is derived from the circuit schematic, and a structure-preserving discretization is performed using the average vector field method. A comparison with LTspice shows that our results are very close to those of more complex macro models.

The perspectives of this study are a) modelling other non-ideal OPA characteristics such as finite slew-rate and bandwidth, current and voltage offsets, non-zero common-mode input gain. . . b) studying the behaviour of the model in other typical circuits (oscillator, rectifier, comparator) and c) experimental comparison with specific devices such as the common μA741, or TL072 audio OPAs.

6. REFERENCES

[1] L. O. Chua, C. A. Desoer, and E. S. Kuh, *Linear and nonlinear circuits*. McGraw-Hill College, 1987.

[2] G. R. Boyle, D. Pederson, B. Cohn, and J. E. Solomon, "Macromodeling of integrated circuit operational amplifiers," *IEEE Journal of Solid-State Circuits*, vol. 9, no. 6, pp. 353–364, 1974.

[3] B. Carter and T. R. Brown, *Handbook of operational amplifier applications*. Texas Instruments Dallas, Tex, USA, 2001.

[4] M. Alexander and D. F. Bowers, "Spice-compatible op amp macro-models," *Analog Devices, Application Note, AN-138*, 1990.

[5] H. Carlin, "Singular network elements," *IEEE Transactions on circuit theory*, vol. 11, no. 1, pp. 67–72, 1964.

[6] B. Tellegen, "On nullators and norators," *IEEE Transactions on circuit theory*, vol. 13, no. 4, pp. 466–469, 1966.

[7] L. Odess and H. Ur, "Nullor equivalent networks of non-ideal operational amplifiers and voltage-controlled sources," *IEEE Transactions on Circuits and Systems*, vol. 27, no. 3, pp. 231–235, 1980.

[8] G. Martinelli, "On the nullor," *Proceedings of the IEEE*, vol. 53, no. 3, pp. 332–332, 1965.

[9] R. C. Paiva, S. D'Angelo, J. Pakarinen, and V. Valimaki, "Emulation of operational amplifiers and diodes in audio distortion circuits," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 59, no. 10, pp. 688–692, 2012.

[10] K. J. Werner, W. R. Dunkel, M. Rest, M. J. Olsen, and J. O. Smith, "Wave digital filter modeling of circuits with operational amplifiers," in *2016 24th European Signal Processing Conference (EUSIPCO)*, pp. 1033–1037, IEEE, 2016.

[11] Ó. Bogason and K. J. Werner, "Modeling circuits with operational transconductance amplifiers using wave digital filters," in *Proc. 20th Int. Conf. Digital Audio Effects, Edinburgh, UK*, pp. 130–137, 2017.

[12] M. Verasani, A. Bernardini, and A. Sarti, "Modeling sallen-key audio filters in the wave digital domain," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 431–435, IEEE, 2017.

[13] A. Schaft, "Port-hamiltonian systems: an introductory survey," 2006.

[14] A. S. Sedra and K. C. Smith, *Microelectronic circuits*. New York: Oxford University Press, 1998.

[15] J. L. Wyatt and L. Chua, "A theory of nonenergic n-ports," *International Journal of Circuit Theory and Applications*, vol. 5, no. 2, pp. 181–208, 1977.

[16] J. Mačák, *Real-time digital simulation of guitar amplifiers as audio effects*. PhD thesis, Brno University of Technology, Brno, 2012.

[17] T. J. Courant, "Dirac manifolds," *Transactions of the American Mathematical Society*, vol. 319, no. 2, pp. 631–661, 1990.

[18] A. J. Van Der Schaft, *L2-gain and passivity techniques in nonlinear control*, vol. 3. Springer, 2000.

[19] A. Devices, "Lt1366 datasheet." `https://www.analog.com/en/products/lt1366.html`, 2019. Online; accessed: 2019-03-22.

[20] R. P. Sallen and E. L. Key, "A practical method of designing rc active filters," *IRE Transactions on Circuit Theory*, vol. 2, no. 1, pp. 74–85, 1955.

[21] N.Steiner, "Voltage-tunable active filter features, low, high and bandpass modes," in *Electronic design 25, December 6*, 1974.

[22] T. Stinchcombe, "A study of the korg ms10 & ms20 filters." `http://www.timstinchcombe.co.uk/synth/MS20_study.pdf`, 2006. Online; accessed: 2019-03-22.

[23] J. Parker and S. D'Angelo, "A digital model of the buchla lowpass-gate," in *Proc. Int. Conf. Digital Audio Effects (DAFx-13), Maynooth, Ireland*, pp. 278–285, 2013.

[24] H. M. Paynter, *Analysis and design of engineering systems*. MIT press, 1961.

[25] P. C. Breedveld, "A systematic method to derive bond graph models," in *Second European Simulation Congress, Antwerp, Belgium*, 1986.

[26] J. F. Broenink, "Introduction to physical systems modelling with bond graphs," *SiE whitebook on simulation methodologies*, vol. 31, 1999.

[27] E. Celledoni, V. Grimm, R. I. McLachlan, D. McLaren, D. O'Neale, B. Owren, and G. Quispel, "Preserving energy resp. dissipation in numerical pdes using the average vector field method," *Journal of Computational Physics*, vol. 231, no. 20, pp. 6770–6789, 2012.

[28] T. Hélie, "Lyapunov stability analysis of the moog ladder filter and dissipativity aspects in numerical solutions," in *Proceedings of the 14th International Conference on Digital Audio Effects DAFx-11, Paris, France*, pp. 19–23, 2011.

[29] S. Bilbao, F. Esqueda, J. D. Parker, and V. Välimäki, "Antiderivative antialiasing for memoryless nonlinearities," *IEEE Signal Processing Letters*, vol. 24, no. 7, pp. 1049–1053, 2017.

[30] R. Muller and T. Hélie, "Power-balanced modelling of circuits as skew gradient systems," in *21 st International Conference on Digital Audio Effects (DAFx-18)*, 2018.

[31] P. Deuflhard, *Newton methods for nonlinear problems: affine invariance and adaptive algorithms*, vol. 35. Springer Science & Business Media, 2011.

A. STRUCTURE OF THE OUTPUT EQUATION

Using the passivity equation (12), then introducing $V_{\mathsf{cm}}, V_{\mathsf{dm}}$ using (4), factoring $V_{\mathsf{cm}}, V_{\mathsf{dm}}$, finally, for $i_{\mathsf{out}} \neq 0$, dividing by i_{out} and using (11) one gets the general form for the output equation (14).

Proof.

$$\begin{aligned}
& i_{\mathsf{S}+} \cdot e_{\mathsf{S}+} + i_{\mathsf{S}-} \cdot e_{\mathsf{S}-} = -i_{\mathsf{out}} \cdot e_{\mathsf{out}} - P_{\mathsf{diss}} \\
\iff\ & i_{\mathsf{S}+}(V_{\mathsf{cm}} + V_{\mathsf{dm}}) + i_{\mathsf{S}-}(V_{\mathsf{cm}} - V_{\mathsf{dm}}) = -i_{\mathsf{out}} \cdot e_{\mathsf{out}} - P_{\mathsf{diss}} \\
\iff\ & V_{\mathsf{cm}}(i_{\mathsf{S}+} + i_{\mathsf{S}-}) + V_{\mathsf{dm}}(i_{\mathsf{S}+} - i_{\mathsf{S}-}) = -i_{\mathsf{out}} \cdot e_{\mathsf{out}} - P_{\mathsf{diss}} \\
\overset{i_{\mathsf{out}} \neq 0}{\iff}\ & V_{\mathsf{cm}} + V_{\mathsf{dm}} \left(\frac{i_{\mathsf{S}+} - i_{\mathsf{S}-}}{i_{\mathsf{S}+} + i_{\mathsf{S}-}} \right) = e_{\mathsf{out}} - \frac{P_{\mathsf{diss}}}{i_{\mathsf{out}}}.
\end{aligned}$$

□

B. FIXED-POINT CONVERGENCE

According to the Banach fixed-point theorem, existence and uniqueness of the solution are guaranteed if the fixed point (55) is contracting, i.e. there exists a Lipschitz constant $\alpha \in [0, 1)$ such that

$$\left\| \phi(\mathbf{x}_1) - \phi(\mathbf{x}_0) \right\| \leq \alpha \|\mathbf{x}_1 - \mathbf{x}_0\| . \tag{59}$$

A sufficient (but conservative) condition is given by

$$\alpha = 1.162\, G\omega_d < 1. \tag{60}$$

Proof. Using (54), then the derivative of the discrete gradient (50), (bounded by $G/2$), and using the spectral radius of $\mathbf{F}_d\mathbf{C}$, one gets

$$\begin{aligned}
\left\| \phi(\mathbf{x}_1) - \phi(\mathbf{x}_0) \right\|_2 &= \left\| \mathbf{F}_d \left(\overline{\nabla} N(\mathbf{C}\mathbf{x}_0, \mathbf{C}\mathbf{x}_1) - \nabla N(\mathbf{C}\mathbf{x}_0) \right) \right\|_2 \\
&\leq \rho \left(\mathbf{F}_d \frac{\partial \overline{\nabla} N}{\partial v_1} \mathbf{C} \right) \|\mathbf{x}_1 - \mathbf{x}_0\|_2 \\
&\leq \rho(\mathbf{F}_d\mathbf{C}) \sup_{v_1} \left| \frac{\partial \overline{\nabla} N}{\partial v_1}(v_0, v_1) \right| \|\mathbf{x}_1 - \mathbf{x}_0\|_2 \\
&\leq \frac{2\omega_d \sqrt{\omega_d^2 + 8\omega_d + 20}}{\left| \omega_d^2 + 2(3 - G)\omega_d + 4 \right|} \frac{G}{2} \|\mathbf{x}_1 - \mathbf{x}_0\|_2 \\
&\leq 1.162\, G\omega_d \|\mathbf{x}_1 - \mathbf{x}_0\|_2
\end{aligned}$$

where the bound 1.162 is obtained numerically by majorizing over $G \in [0, 3]$ and $\omega_d \geq 0$. □

TOWARDS INVERSE VIRTUAL ANALOG MODELING

Alberto Bernardini

Dipartimento di Elettronica, Informazione e Bioingegneria,
Politecnico di Milano,
Piazza L. Da Vinci 32, 20133 Milano – Italy
alberto.bernardini@polimi.it

Augusto Sarti

Dipartimento di Elettronica, Informazione e Bioingegneria,
Politecnico di Milano,
Piazza L. Da Vinci 32, 20133 Milano – Italy
augusto.sarti@polimi.it

ABSTRACT

Several digital signal processing approaches, generally referred to as Virtual Analog (VA) modeling, are currently under development for the software emulation of analog audio circuitry. The main purpose of VA modeling is to faithfully reproduce the behavior of real-world audio gear, e.g., distortion effects, synthesizers or amplifiers, using efficient algorithms. In this paper, however, we provide a preliminary discussion about how VA modeling can be exploited to infer the input signal of an analog audio system, given the output signal and the parameters of the circuit. In particular, we show how an inversion theorem known in circuit theory, and based on nullors, can be used for this purpose. As recent advances in Wave Digital Filter (WDF) theory allow us to implement circuits with nullors in a systematic fashion, WDFs prove to be useful tools for inverse VA modeling. WDF realizations of a nonlinear audio system and its inverse are presented as an example of application.

1. INTRODUCTION

The term Virtual Analog (VA) modeling generally refers to a variety of approaches for digitally emulating the behavior of linear or nonlinear analog audio circuits [1, 2]. Such approaches can be roughly subdivided in two main classes [3]. The first class is the one of *gray box* modeling methods, which estimate the system starting from input-output measurements and a reference generic model; examples are Volterra-Wiener-Hammerstein models and modifications thereof [4, 5, 6], neural networks [7] and Legendre nonlinear filters [8]. The second class is the one of *white box* modeling techniques, which are based on the solution of the equations characterizing the actual audio circuit to be emulated; examples are state-space methods [9, 10], port-Hamiltonian methods [11] and Wave Digital Filters (WDFs) [12, 13]. Gray box modeling approaches are very general and, usually, once the parameters of the model have been derived, less computationally heavy than white box approaches. On the other hand, white box approaches are generally more accurate and do not require any estimation of the model parameters. The main purpose of VA modeling is to reproduce the output signal of real-world audio gear, e.g., distortion effects, synthesizers or amplifiers, finding an algorithm that ensures a good compromise between high accuracy and light computational weight. In this paper, however, we provide a preliminary discussion about how VA modeling can be exploited to estimate the input signal of an analog audio system, given the output signal and the parameters of the circuit. The scenario of interest is shown in Figure 1; assuming to know the parameters of System 1 and its time-domain output signal $y(t)$, we would like to recover the input signal $u(t)$ computing the output of System 2 given $y(t)$ as input. In the optimal case, System 2 is the exact inverse of System 1 and $\tilde{u}(t) = u(t)$. The capability of recovering the input signal $u(t)$ starting from the processed, eventually distorted, output signal $y(t)$ could be useful in various situations. As an example, effective techniques of the sort would allow us to perform *reamping* of sound signals from electric musical instruments, in order to change the applied effects or re-record with different amplifiers, when the "dry signal" is not available, but the originally used analog audio effect or amplifier is known.

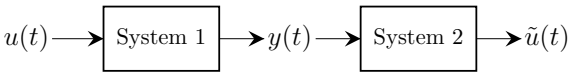


Figure 1: *Inverse system approach. If System* 2 *is the exact inverse of System* 1 *and the two systems share the same initial conditions, the signal* $\tilde{u}(t)$ *is a perfect estimate of the input signal* $u(t)$.

A possible strategy to attack this problem would be to use certain gray box approaches for modeling the system, e.g., Volterra-Wiener models, and then derive the corresponding inverse system, e.g., using functional or operatorial techniques [14, 15, 16]. An alternative idea, discussed in this paper, is to employ white box modeling approaches and resort to a theorem known in circuit theory [17] which, given a system represented as an electrical network, allows us to design its inverse. The theorem is based on the use of a theoretical 2-port circuit element called *nullor* [18]. Since recent theoretical advances in WDF theory allow us to design WDF models of circuits with nullors in a systematic fashion [19, 20], in this paper, we discuss how WDF principles can be exploited to derive digital realization of inverse systems, applying the theorem presented in [17]. Section 2 discusses the main aspects of the nullor-based inversion theorem. Section 3 recalls some recently developed techniques in WDF theory, such as a method for implementing circuits with nullors in the Wave Digital (WD) domain. As an example of application, Section 4 describes WDF realizations of a nonlinear audio clipper and its inverse. Section 5 concludes this paper and proposes possible future developments.

2. INVERSE SYSTEM DESIGN BASED ON NULLORS

2.1. Background on Nullors

A nullor [18], shown in Figure 2, is a theoretical 2-port linear circuit element composed of two other theoretical one-ports, called

nullator and norator. A *nullator* is a circuit element having zero current passing through it and zero voltage across its terminals, while a *norator* is characterized by an arbitrary current passing through it and an arbitrary voltage across its terminals.

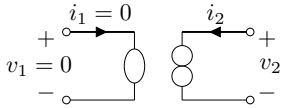


Figure 2: *A nullor composed of a nullator (represented with an ellipse) paired with a norator (represented with two contiguous circles).*

It follows that a nullor is characterized by the constitutive equation

$$\begin{bmatrix} v_1 \\ i_1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} v_2 \\ i_2 \end{bmatrix} \quad , \tag{1}$$

where v_1 is the voltage across the nullator, v_2 is the voltage across the norator, i_1 is the current through the nullator and i_2 is the current through the norator.

Nullators, norators and nullors are often used to build macromodels of more complex multi-ports, such as 2-ports with controlled sources, operational amplifiers (opamps), transconductance amplifiers or transistors. For instance, an ideal opamp can be modeled using a nullor as shown in Figure 3, where v_{1+}, v_{1-} and v_{2+} are the potentials at the two input terminals and the output terminal of the opamp, respectively.

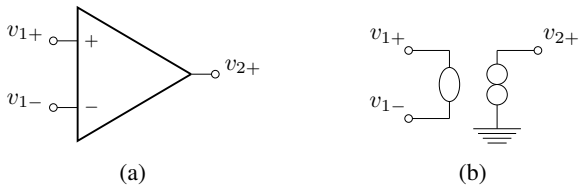


Figure 3: (a) *represents an ideal opamp, while* (b) *is an equivalent nullor-based representation of the same ideal opamp in* (a).

2.2. Inversion Theorem

Techniques for the analog realization of inverse systems have been explored in the mid-nineties [21, 22], especially with the purpose of developing methods for synchronizing chaotic systems [23]. A generalization of the inverse system approach used in [21, 22] was developed in [17], where a proven theorem is exploited for designing the inverse of a system that can be represented as an electrical circuit containing at least one nullor. The theorem presented in [17] is explained in the following.

Let us call the system to be inverted *master* and let us assume that it can be represented as in Figure 4(a); the system is constituted of a generic linear or nonlinear non-autonomous time-invariant network, and at least one nullator-norator pair connected to the network. A single input signal $u(t)$ is injected into the system using a voltage source, while the output signal $y(t)$ is the voltage measured across the norator. Let us then derive the so called *slave* referred to the master in Figure 4(a). The slave is obtained by replacing the input voltage source of the master with the norator and the norator with a voltage-controlled voltage source, whose control signal is $y(t)$; the resulting system is shown in Figure 4(b).

Another possible representation of a master system is reported in Figure 5(a); the system is analogous to the one in Figure 4(a), except that the output signal $y(t)$ is the current through the norator. The corresponding slave is represented in Figure 5(b), where the input voltage source of the master has been replaced with the norator and the norator with a current-controlled current source, whose control signal is $y(t)$.

Given the circuits in Figure 4 and in Figure 5, the following theorem enounced and proved in Section 3 of [17] holds; the theorem is here slightly reworded for the sake of clarity.

Theorem 1. If the two nonlinear dynamical systems in Figure 4 have unique bounded solutions, then, for any pair of signals $u(t)$ and $y(t)$, the slave in Figure 4(b) is the inverse of the master in Figure 4(a). This means that for any input signal $u(t)$ and every initial state vector $\mathbf{x}(0)$ for the master, there is an initial state vector $\tilde{\mathbf{x}}(0)$ for the slave such that $\tilde{u}(t) = u(t)$. The same holds for the master and the slave in Figure 5.

Similar results can be obtained when the input signal $u(t)$ is a current source, but they are not discussed here, since current sources are less common in audio circuits. A very common scenario, instead, is the one in which no nullors are present in the master system to be inverted, as in Figure 6(a), where the output signal $y(t)$ is the difference of potentials at two generic terminals (C and D) of the nonlinear network. It turns out that every circuit can be augmented with a nullor without altering its behavior [17]. In fact, since the series of a nullator and a norator is equivalent to an open circuit, the system in Figure 6(b) is characterized by exactly the same behavior of the one in Figure 6(a). It follows that the same procedure applied to the system in Figure 4(a) to derive its inverse can now be applied to the master in Figure 6(b), obtaining the corresponding slave in Figure 6(c). Hence, according to *Theorem 1*, the system in Figure 6(c) is the inverse of the system in Figure 6(a). Similarly, let us consider the system without nullors in Figure 7(a) whose output signal $y(t)$ is a current. Since the parallel of a nullator and a norator is equivalent to a short circuit, the system in Figure 7(b) is interchangeable with the system in Figure 7(a). It follows that the same procedure applied to the system in Figure 5(a) to derive its inverse can now be applied to the master in Figure 7(b), obtaining the corresponding slave in Figure 7(c). Hence, according to *Theorem 1*, the system in Figure 7(c) is the inverse of the system in Figure 7(a).

3. BRIEF OVERVIEW ON WDF MODELING

3.1. Basic WDF Principles

The WD modeling of a system is based on a port-wise consideration of the reference equivalent circuit and a linear transformation of Kirchhoff port variables into WD variables at each port, as the following [12]

$$a = v + Zi \quad , \quad b = v - Zi \quad , \tag{2}$$

where v is the port voltage, i is the port current, a is the incident wave, b is the reflected wave and Z is a scalar free parameter (port resistance). Variables defined in (2) are the so called *voltage waves* [12], however other kinds of waves with different units of measure [20] or more than one free parameter [24, 25] can be defined. The inverse relation of (2) is

$$v = (a + b)/2 \quad , \quad i = (a - b)/(2Z) \quad . \tag{3}$$

Figure 4: (a) *shows a nonlinear non-autonomous system with at least one nullor (the master). The input signal* $u(t)$ *is a voltage source, while the output signal* $y(t)$ *is the voltage across the norator.* (b) *shows the inverse system (the slave) of the one in* (a). *Letters* A, B, C, D, E *and* F *indicate the same nodes of the nonlinear network both in* (a) *and* (b).

Figure 5: (a) *shows a nonlinear non-autonomous system with at least one nullor (the master). The input signal* $u(t)$ *is a voltage source, while the output signal* $y(t)$ *is the current through the norator.* (b) *shows the inverse system (the slave) of the one in* (a). *Letters* A, B, C, D, E *and* F *indicate the same nodes of the nonlinear network both in* (a) *and* (b).

Substituting (3) in the constitutive equations of circuit elements and then expressing b as a function of a, WD *scattering relations* at each port are found. When modeling most linear one-ports, e.g., resistors, resistive sources, capacitors and inductors, in the WD domain, it is possible to spend the free parameter Z in order to eliminate the instantaneous dependence between a and b; this process is called *adaptation*. In WD structures, the connection networks that connect circuit elements (e.g., in series, in parallel or according to more complex constraints) are implemented using N-port scattering junctions, called *adaptors*. A scattering junction is characterized by scattering matrices that satisfy the following general equation

$$\mathbf{a} = \mathbf{S}\mathbf{b} \ , \tag{4}$$

where $\mathbf{a}$ is the vector of waves reflected from the junction (and incident to the elements or the other adaptors of the WD structure connected to the junction), while $\mathbf{b}$ is the vector of waves incident to the junction (and reflected from the elements or the other adaptors connected to the junction). For a detailed description of basic linear WD elements and adaptors the reader is referred to [12]. Instead, a brief discussion on the modeling of WD structures with nonlinearities, nonreciprocal multi-ports and complex topologies is provided in the following subsection.

3.2. Recent Advances in WDF Theory

Despite the idea of using WD structures for sound synthesis through physical modeling and Virtual Analog modeling dates back to the nineties [1, 26], a considerable effort has been made in the last few years to enlarge the class of audio circuits that can be modeled in the WD domain. In particular, WD models of linear and nonlinear circuit elements, such as operational amplifiers [27, 28, 19, 29, 30], nonlinear transformers [31], diodes [27, 32, 33, 34], vacuum tubes [35, 36] and transistors [37, 38, 39, 40, 41], that were not considered in traditional WDF theory [12] have been discussed. Moreover, novel design strategies for implementing $\mathcal{R}$-type WD adaptors describing arbitrary reciprocal [42] or nonreciprocal [20] connection networks have been developed. Dynamic circuits with up to one nonlinear element can be implemented using WD structures without Delay-Free-Loops (DFLs), i.e., fully *explicit* [43, 26, 34]. Although this desirable property is not maintained when we deal with circuits with multiple nonlinearities, some novel approaches have been recently presented for solving this kind of circuits using the K method [44, 38] or iterative techniques [45, 40, 46, 47, 48].

In the context of this article, the modeling of nullors in the WD domain is of particular interest. Nullators, norators and nullors are singular elements [18]. It follows that they cannot be modeled as other circuit elements in the WD domain using separable blocks characterized by local scattering relations. However, systematic methods developed in [19, 20] can be adopted to model circuits with nullors in the WD domain. Such methods are based on the Modified Nodal Analysis [49] and they allow us to derive WD realizations of nonreciprocal connection networks embedding both topological information and nullors; the result are special $\mathcal{R}$-type WD adaptors which are neither pseudolossless nor reciprocal [20].

In the next Section, as an example, we will show how both an audio clipper with one nonlinearity and its nullor-based inverse can be implemented in the WD domain in an explicit fashion.

4. WDF REALIZATIONS OF A NONLINEAR AUDIO CLIPPER AND ITS INVERSE

4.1. Circuit of the Audio Clipper and Design of its Inverse

The circuit of the nonlinear diode-based audio clipper that we will consider is shown in Figure 8(a). The input signal $u(t)$ is the voltage supplied by the voltage source. The output signal $y(t)$ is the voltage across the resistor with resistance R_{out}. The circuit is composed of: five linear resistors with resistances R_{p}, R_{m}, R_{fl}, R_{fh} and R_{out}; two linear capacitors with capacitances C_{p} and C_{m}; and two nonlinear diodes, D_1 and D_2, in antiparallel. The two diodes are

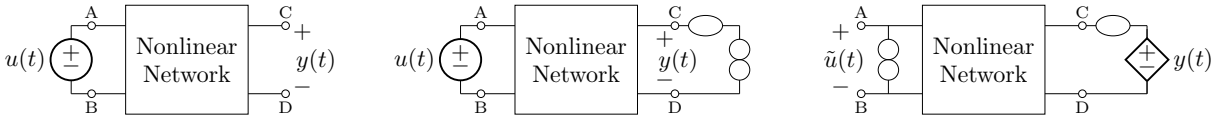


Figure 6: (a) *shows a nonlinear non-autonomous system without nullors (the master). The input signal $u(t)$ is a voltage source and the output signal $y(t)$ is a voltage.* (b) *shows a system equivalent to the one in* (a) *augmented with a nullator-norator pair.* (c) *shows the inverse system (the slave) of the one in* (a).

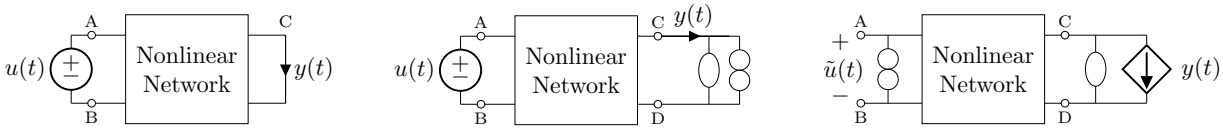


Figure 7: (a) *shows a nonlinear non-autonomous system without nullors (the master). The input signal $u(t)$ is a voltage source, while the output signal $y(t)$ is a current.* (b) *shows a system equivalent to the one in* (a) *augmented with a nullator-norator pair.* (c) *shows the inverse system (the slave) of the one in* (a).

identical and both characterized by the Shockley diode model

$$i_{\mathrm{d}} = I_{\mathrm{s}}(e^{v_{\mathrm{d}}/(\eta V_{\mathrm{th}})} - 1) \ , \tag{5}$$

where i_{d} is the current passing through the diode, v_{d} is the voltage across the diode, I_{s} is the saturation current, η is the ideality factor of the exponential junction and V_{th} is the thermal voltage. A detailed list of the circuit parameters and the corresponding values is reported in Table 1. For the purpose of designing an inverse

Table 1: *Parameters of the Audio Clipper Circuit*.

Param.	Value	Param.	Value	Param.	Value
(ηV_{th})	26 mV	C_{p}	1 pF	R_{p}	39 kΩ
I_{s}	2.52 nA	C_{m}	3.9 nF	R_{m}	22 kΩ
R_{out}	150 kΩ	R_{f1}	50 kΩ	R_{f2}	50 kΩ

audio clipper, given the system in Figure 8(a), we add a nullor to the circuit. In fact, as explained in Subsection 2.2, a circuit can always be augmented with a nullator-norator pair obtaining another circuit equivalent to the original one. Figure 8(b) shows the audio clipper circuit with a series of a nullator and a norator from node C to ground which has exactly the same behavior of the one in Figure 8(a). Starting from the master in Figure 8(b), we get the corresponding slave in Figure 9(a), according to the Theorem in Subsection 2.2. Figure 9(b) shows an alternative representation of the inverse nonlinear audio clipper, where the nullor is replaced by an opamp. It is worth recalling that, if the opamp is characterized by the ideal nullor-based model shown in Figure 3, the circuit in Figure 9(b) is totally equivalent to the circuit in Figure 9(a); if a non-ideal opamp model is considered, instead, the circuit in Figure 9(b) is an approximation of the "exact inverse" in Figure 9(a).

4.2. Design of WD Structures and Simulation Results

A possible WD realization with no DFLs of the nonlinear audio clipper in Figure 8(a) is shown in Figure 10(a). WD blocks (elements or junctions) including a "T-shaped" symbol at one port are adapted at that port. As far as the WD modeling of the circuit elements is concerned, linear elements are all adapted according to traditional WDF principles [12]. In particular, the wave reflected from a resistor with resistance R at the nth port of an adaptor is $b_n = 0$ during the whole simulation, provided that the adaptation condition $Z_n = R$ is satisfied. The reflected wave from a capacitor with capacitance C at sampling step k, instead, is given by $b_n[k] = a_n[k-1]$, provided that the bilinear transform is applied to discretize the time derivative and the adaptation constraint $Z_n = T_s/(2C)$ is set, where $F_{\mathrm{s}} = 1/T_{\mathrm{s}}$ is the sampling frequency. The voltage source is augmented with a negligible small series resistances $R_{\mathrm{e}} = 1\ \mu\Omega$; the reflected wave is computed as $b[k] = u(kT_s)$ with $Z_5 = R_{\mathrm{e}}$. The pair of antiparallel diodes is treated as a one-port. Since it is nonlinear, such a one-port cannot be adapted and, according to [27, 33, 39], its WD mapping can be expressed as

$$b = \mathrm{sgn}(a)F(|a|, Z, I_{\mathrm{s}}, \eta, V_{\mathrm{th}}) \ , \ \text{with} \tag{6}$$

$$F(a, Z, I_{\mathrm{s}}, \eta, V_{\mathrm{th}}) = a + 2ZI_{\mathrm{s}} - 2\eta V_{\mathrm{th}} W\left(\frac{ZI_{\mathrm{s}}}{\eta V_{\mathrm{th}}} e^{\frac{ZI_{\mathrm{s}}+a}{\eta V_{\mathrm{th}}}}\right) \ ,$$

where W is the Lambert function. The connection network of the circuit in Figure 8(a) is reciprocal, hence it is implemented using an interconnection of two reciprocal WD junctions [42]: the parallel adaptor $\mathcal{P}_1$ and the $\mathcal{R}$-type adaptor $\mathcal{R}_1$. The standard 3-port parallel adaptor is realized as discussed in [12], while the 8-port $\mathcal{R}$-type adaptor is characterized by a scattering matrix $\mathbf{S}_{\mathcal{R}_1}$, which can be formed as [42]

$$\mathbf{S}_{\mathcal{R}_1} = 2\mathbf{Q}^T\left(\mathbf{Q}\mathbf{Z}^{-1}\mathbf{Q}^T\right)^{-1}\mathbf{Q}\mathbf{Z}^{-1} - \mathbf{I}_8 \ , \tag{7}$$

where $\mathbf{I}_8$ is the 8×8 identity matrix, $\mathbf{Z} = \mathrm{diag}[Z_1, \ldots, Z_8]$ is the diagonal matrix of port resistances (port numbering follows the convention in Figure 10(a)) and $\mathbf{Q}$ is the fundamental cut-set matrix characterizing the connection network and given by

$$\mathbf{Q} = \begin{bmatrix} -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & -1 & 0 & 0 & 0 & 1 \end{bmatrix} .$$

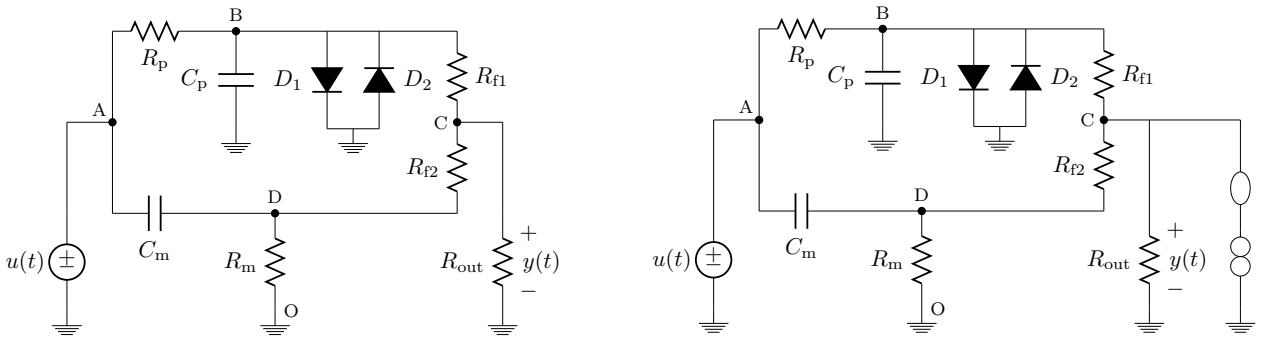


Figure 8: (a) *shows the circuit of the nonlinear audio clipper.* (b) *shows a circuit equivalent to the one in* (a) *augmented with a nullor.*

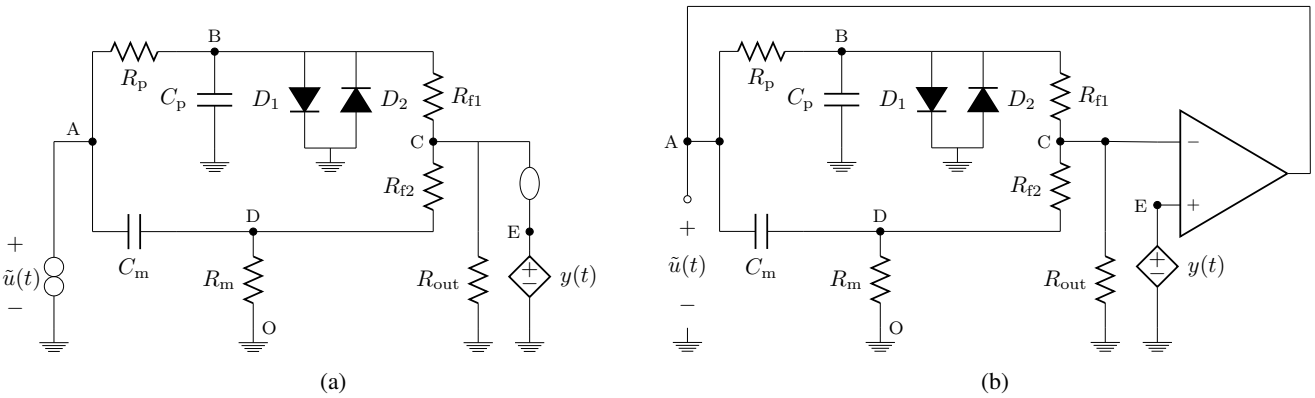


Figure 9: (a) *shows the circuit of the inverse nonlinear audio clipper.* (b) *shows an alternative representation of the inverse nonlinear audio clipper where the nullor is replaced by an opamp. If the opamp is characterized by the ideal nullor-based model in* Figure 3, *the circuit in* (b) *is fully equivalent to the circuit in* (a).

Port 6 of $\mathcal{R}_1$ is made reflection-free by setting the 6th diagonal entry of the scattering matrix equal to zero. The expression of the adaptation constraint on Z_6 is not reported here for the sake of brevity; however, given the circuit parameters in Table 1, we get the constraint $Z_6 = 27110\,\Omega$.

As far as the inverse audio clipper in Figure 9(b) is concerned, a possible WD realization with no DFLs is shown in Figure 10(b). The elements and the 3-port parallel adaptor $\mathcal{P}_2$ are implemented in a similar fashion to the previous case; the $\mathcal{R}$-type adaptor $\mathcal{R}_2$, instead, differs from $\mathcal{R}_1$, as it is a WD realization of a non-reciprocal connection network embedding a nullor. According to [20], the scattering matrix $\mathbf{S}_{\mathcal{R}_2}$ of $\mathcal{R}_2$ is given by

$$\mathbf{S}_{\mathcal{R}_2} = 2\tilde{\mathbf{A}}_\mathrm{p}^T\,[\mathbf{I}_5\ \ \mathbf{0}]\,\tilde{\mathbf{X}}_0^{-1}[\mathbf{I}_5\ \ \mathbf{0}]^T\,\tilde{\mathbf{A}}_\mathrm{p}\mathbf{Z}^{-1} - \mathbf{I}_8\ , \qquad (8)$$

where $\mathbf{I}_5$ is the 5×5 identity matrix, $\mathbf{0}$ is a column vector of five zeros, $\mathbf{Z} = \mathrm{diag}[Z_1, \ldots, Z_8]$ is a diagonal matrix of port resistances (port numbering follows the convention in Figure 10(b)), $\tilde{\mathbf{X}}_0$ is the 6×6 matrix obtained removing the jth row ($1 \leq j \leq 7$) and the jth column of $\mathbf{X}_0$ in Figure 11, while $\tilde{\mathbf{A}}_\mathrm{p}$ is the 5×8 matrix obtained removing the jth column of $\mathbf{A}_\mathrm{p}$ in Figure 11. Port 1 of $\mathcal{R}_2$ is adapted by setting $Z_1 = (Z_3Z_5Z_7)/(Z_3Z_5 + Z_3Z_6 + Z_3Z_7 + Z_3Z_8 + Z_6Z_8)$.

The accuracy of the designed WDFs is verified comparing the output signals with those obtained from a Spice implementation and noting a good matching. One of such tests is reported in Figure 12. WD simulations are performed with a sampling frequency $F_\mathrm{s} = 44100$ Hz. Figures 13, 14 and 15 show some further results of the WD implementations of the master and the slave in Figure 10, when the input signal is a sinusoid, a square wave and a white noise, respectively. Detailed parameters of the input are specified in the captions. We notice that inversion works pretty well for each input signal choice, as $\tilde{u}(t)$ always closely matches $u(t)$. More precisely, let us define the absolute value of the estimation error as $\epsilon(t) = |\tilde{u}(t) - u(t)|$, and then derive its mean $\bar{\epsilon}$ and its maximum value ϵ_max. We get: $\bar{\epsilon} = 1.5 \times 10^{-11}$ V and $\epsilon_\mathrm{max} = 2.5 \times 10^{-11}$ V in the sinusoidal input case; $\bar{\epsilon} = 2.7 \times 10^{-11}$ V and $\epsilon_\mathrm{max} = 1.1 \times 10^{-10}$ V in the square wave case; $\bar{\epsilon} = 1.4 \times 10^{-11}$ V and $\epsilon_\mathrm{max} = 3.6 \times 10^{-11}$ V in the white noise case.

5. CONCLUSIONS AND FUTURE WORK

We discussed a general approach for the design and the WDF realization of the inverse of any input-output audio system that can be described using electrical equivalents. It is worth pointing out that the presented approach works properly only when we deal with invertible nonlinearities and that numerical problems may occur when nonlinear functions to be inverted are very steep. Future research work would be desirable for extending such an approach to

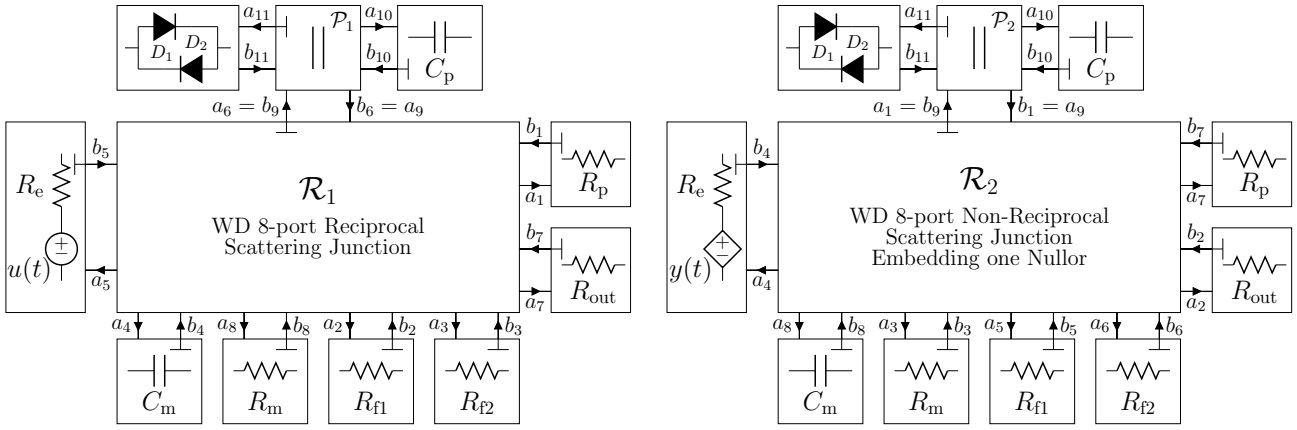


Figure 10: (a) *WDF implementing the nonlinear audio clipper*; (b) *WDF implementing the inverse nonlinear audio clipper*.

$$\mathbf{X}_0 = \begin{bmatrix} G_1+G_2+G_3+G_4 & 0 & -G_1 & -G_2 & -G_3 & -G_4 & 1 \\ 0 & G_7+G_8 & -G_7 & 0 & -G_8 & 0 & -1 \\ -G_1 & -G_7 & G_1+G_5+G_7 & -G_5 & 0 & 0 & 0 \\ -G_2 & 0 & -G_5 & G_2+G_5+G_6 & -G_6 & 0 & 0 \\ -G_3 & -G_8 & 0 & -G_6 & G_3+G_6+G_8 & 0 & 0 \\ -G_4 & 0 & 0 & 0 & 0 & G_4 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 \end{bmatrix} \quad \mathbf{A}_p = \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 11: Matrices needed for deriving the scattering matrix of $\mathcal{R}$-type adaptor $\mathcal{R}_2$.

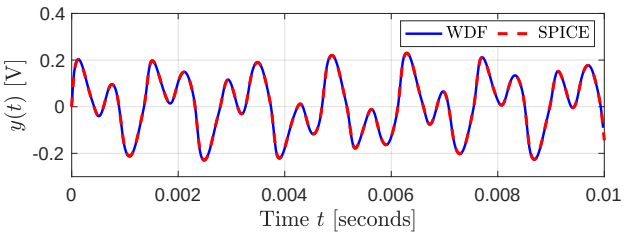


Figure 12: *Comparison between WDF in Figure 10(a) and Spice. Output* $y(t)$ *given input* $u(t) = \sum_{j=1}^{2} g_j \sin(2\pi f_{0j})$ with $g_1 = 0.5$ V, $f_{01} = 650$ Hz, $g_2 = 0.35$ V and $f_{02} = 1450$ Hz.

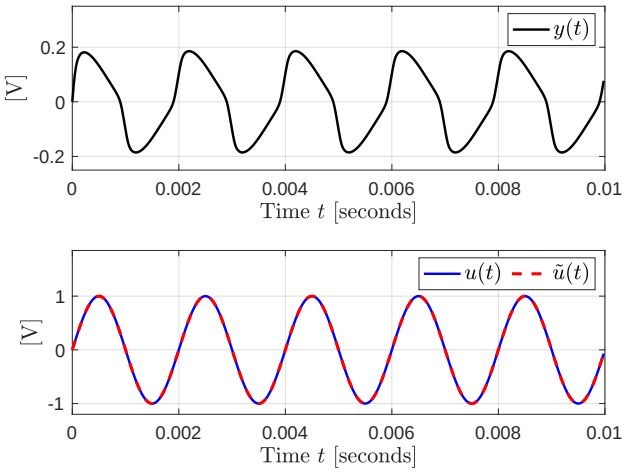


Figure 13: *Results of WDF implementation of master and slave, given a sinusoidal input* $u(t) = \sin(2\pi f_0)$ with $f_0 = 500$ Hz.

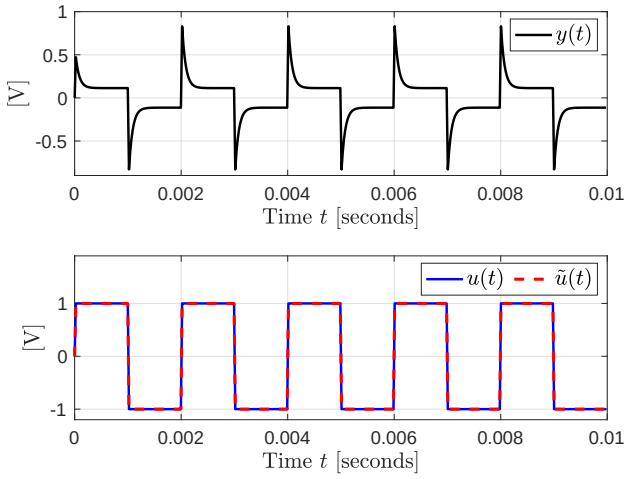


Figure 14: *Results of WDF implementation of master and slave, given a square wave input* $u(t) = \text{sgn}(\sin(2\pi f_0))$ with $f_0 = 500$ Hz.

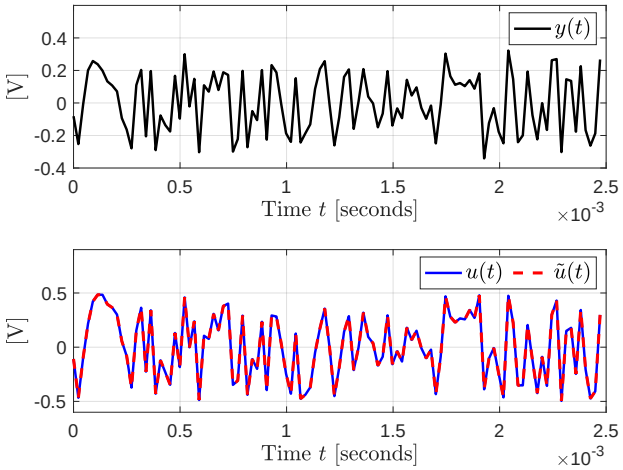


Figure 15: *Results of WDF implementation of master and slave, given a zero-centered white noise with amplitude* 1 *as input.*

MIMO systems and to systems with uncertain parameters.

6. ACKNOWLEDGMENTS

The authors wish to thank Federico Borra and Kurt James Werner for the encouraging discussions about the potential benefits of inverse Virtual Analog modeling in digital audio applications.

7. REFERENCES

[1] J. O. Smith, "Physical modeling synthesis update," *Computer Music Journal*, vol. 20, no. 2, pp. 44–56, 1996.

[2] J. Pakarinen, V. Välimäki, F. Fontana, V. Lazzarini, and J. S. Abel, "Recent advances in real-time musical effects, synthesis, and virtual analog models," *EURASIP J. on Advances in Signal Process.*, vol. 2011, no. 1, pp. 940784, Feb 2011.

[3] Stefano D'Angelo, *Virtual Analog Modeling of Nonlinear Musical Circuits*, Ph.D. Dissertation, Aalto University, Espoo, Finland, Sept. 2014.

[4] T. Helie, "Volterra series and state transformation for real-time simulations of audio circuits including saturations: Application to the moog ladder filter," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 18, no. 4, pp. 747–759, May 2010.

[5] F. Eichas and U. Zölzer, "Gray-box modeling of guitar amplifiers," *J. Audio Eng. Soc*, vol. 66, no. 12, pp. 1006–1015, 2018.

[6] A. Novak, L. Simon, F. Kadlec, and P. Lotton, "Nonlinear system identification using exponential swept-sine signal," *IEEE Trans. Instrum. Meas.*, vol. 59, no. 8, pp. 2220–2229, Aug 2010.

[7] T. Schmitz and J. J. Embrechts, "Nonlinear real-time emulation of a tube amplifier with a long short time memory neural-network," in *Audio Engineering Society Convention 144*, May 2018.

[8] A. Carini, S. Cecchi, M. Gasparini, and G. L. Sicuranza, "Introducing legendre nonlinear filters," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 7939–7943.

[9] David T. Yeh, Jonathan S. Abel, and Julius O. Smith, "Automated physical modeling of nonlinear audio circuits for real-time audio effects - part 1: Theoretical development," *IEEE Trans. Audio, Speech, Language Process.*, vol. 18, pp. 728–737, May 2010.

[10] K. Dempwolf, M. Holters, and U. Zölzer, "Discretization of parametric analog circuits for real-time simulations," in *Proc. 13th Conf. Digital Audio Effects*, Graz, Austria, Sept. 42–49 2010.

[11] Antoine Falaize and Thomas Hélie, "Passive guaranteed simulation of analog audio circuits: A port-hamiltonian approach," *Appl. Sci.*, vol. 6, no. 10, 2016.

[12] Alfred Fettweis, "Wave digital filters: Theory and practice," *Proc. IEEE*, vol. 74, no. 2, pp. 270–327, Feb. 1986.

[13] Giovanni De Sanctis and Augusto Sarti, "Virtual analog modeling in the wave-digital domain," *IEEE Trans. Audio, Speech, Language Process.*, vol. 18, pp. 715–727, May 2010.

[14] M. Schetzen, *The Volterra and Wiener Theories of Nonlinear Systems*, Krieger Publishing Co., Inc., Melbourne, FL, USA, 2006.

[15] A. Sarti and S. Pupolin, "Recursive techniques for the synthesis of a pth-order inverse of a volterra system," *European Transactions on Telecommunications*, vol. 3, no. 4, pp. 315–322, 1992.

[16] A. Carini, G. L. Sicuranza, and V. J. Mathews, "On the inversion of certain nonlinear systems," *IEEE Signal Processing Letters*, vol. 4, no. 12, pp. 334–336, Dec 1997.

[17] A. Leuciuc, "The realization of inverse system for circuits containing nullors with applications in chaos synchronization," *International Journal of Circuit Theory and Applications*, vol. 26, no. 1, pp. 1–12, 1998.

[18] H. J. Carlin, "Singular network elements," *IEEE Trans. Circuit Theory*, vol. 11, no. 1, pp. 67–72, Mar. 1964.

[19] K. J. Werner, W. R. Dunkel, M. Rest, M. J. Olsen, and J. O. Smith III, "Wave digital filter modeling of circuits with operational amplifiers," in *Proc. Eur. Signal Process. Conf.*, Budapest, Hungary, Aug. 2016, pp. 1033–1037.

[20] K. J. Werner, A. Bernardini, J. O. Smith, and A. Sarti, "Modeling circuits with arbitrary topologies and active linear multiports using wave digital filters," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 12, pp. 4233–4246, Dec 2018.

[21] F. Bohme and W. Schwarz, "The chaotizer-dechaotizer-channel," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 43, no. 7, pp. 596–599, July 1996.

[22] U. Feldmann, M. Hasler, and W. Schwarz, "Communication by chaotic signals: the inverse system approach," *International Journal of Circuit Theory and Applications*, vol. 24, no. 5, pp. 551–579, 1996.

[23] Louis M. Pecora and Thomas L. Carroll, "Synchronization of chaotic systems," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 25, no. 9, pp. 0976111–09761112, 2015.

[24] A. Bernardini and A. Sarti, "Biparametric wave digital filters," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 7, pp. 1826–1838, July 2017.

[25] D. Rocchesso and J. O. Smith, "Generalized digital waveguide networks," *IEEE Trans. on Speech and Audio Process.*, vol. 11, no. 3, pp. 242–254, May 2003.

[26] Augusto Sarti and Giovanni De Poli, "Toward nonlinear wave digital filters," *IEEE Trans. Signal Process.*, vol. 47, pp. 1654–1668, June 1999.

[27] Rafael C. D. Paiva, Stefano D'Angelo, Jyri Pakarinen, and Vesa Välimäki, "Emulation of operational amplifiers and diodes in audio distortion circuits," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 59, pp. 688–692, Oct. 2012.

[28] K. J. Werner, J. O. Smith, and J. S. Abel, "Wave digital filters adaptors for arbitrary topologies and multiport linear elements," in *Proc. 18th Conf. Digital Audio Effects*, Trondheim, Norway, Nov. 30 – Dec. 3 2015.

[29] M. Verasani, A. Bernardini, and A. Sarti, "Modeling sallen-key audio filters in the wave digital domain," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017, pp. 431–435.

[30] Ó. Bogason and K. J. Werner, "Modeling circuits with operational transconductance amplifiers using wave digital filters," in *Proc. 20th Int. Conf. Digital Audio Effects*, Edinburgh, UK, Sept. 2017, pp. 130–137.

[31] Rafael Cauduro Dias de Paiva, Jyri Pakarinen, Vesa Välimäki, and Miikka Tikander, "Real-time audio transformer emulation for virtual tube amplifiers," *EURASIP J. on Advances in Signal Process.*, Jan. 2011.

[32] A. Bernardini, K. J. Werner, A. Sarti, and J. O. Smith, "Multi-port nonlinearities in wave digital structures," in *Proc. IEEE Int. Symp. Signals Circuits Syst.*, Iasi, Romania, July 9–10 2015.

[33] K. J. Werner, V. Nangia, A. Bernardini, J. O. Smith III, and A. Sarti, "An improved and generalized diode clipper model for wave digital filters," in *Proc. 139th Conv. Audio Eng. Soc. (AES)*, New York, NY, Oct. 2015, conv. paper #9360.

[34] A. Bernardini and A. Sarti, "Canonical piecewise-linear representation of curves in the wave digital domain," in *Proc. 25th Eur. Signal Proces. Conf. (EUSIPCO)*, Aug. 2017, pp. 1125–1129.

[35] J. Pakarinen and M. Karjalainen, "Enhanced wave digital triode model for real-time tube amplifier emulation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 4, pp. 738–746, May 2010.

[36] Stefano D'Angelo, Jyri Pakarinen, and Vesa Välimäki, "New family of wave-digital triode models," *IEEE Trans. Audio, Speech, Language Process.*, vol. 21, pp. 313–321, Feb. 2013.

[37] A. Bernardini, K. J. Werner, A. Sarti, and J. O. Smith, "Modeling a class of multi-port nonlinearities in wave digital structures," in *Proc. European Signal Process. Conf. (EUSIPCO)*, Nice, France, Aug. 31 – Sept. 4 2015, pp. 669–673.

[38] K. J. Werner, V. Nangia, J. O. Smith, and J. S. Abel, "Resolving wave digital filters with multiple/multiport nonlinearities," in *Proc. 18th Conf. Digital Audio Effects*, Trondheim, Norway, Nov. 30 – Dec. 3 2015.

[39] A. Bernardini, K. J. Werner, A. Sarti, and J. O. Smith III, "Modeling nonlinear wave digital elements using the Lambert function," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 8, pp. 1231–1242, Aug. 2016.

[40] Michael Jørgen Olsen, Kurt James Werner, and J. O. Smith, "Resolving grouped nonlinearities in wave digital filters using iterative techniques," in *Proc. 19th Int. Conf. Digital Audio Effects*, Brno, Czech Republic, September 5–9 2016, pp. 279–286.

[41] D. Hernandez and J. Huang, "Emulation of junction field-effect transistors for real-time audio applications," *IEICE Electronics Express*, May 2016.

[42] A. Bernardini, K. J. Werner, J. O. Smith, and A. Sarti, "Generalized wave digital filter realizations of arbitrary reciprocal connection networks," *IEEE Trans. Circuits and Systems I: Regular Papers*, vol. 66, no. 2, pp. 694–707, Feb 2019.

[43] Klaus Meerkötter and Reinhard Scholz, "Digital simulation of nonlinear circuits by wave digital filter principles," in *IEEE Int. Symp. Circuits Syst.*, June 1989, pp. 720–723.

[44] G. Borin, G. De Poli, and D. Rocchesso, "Elimination of delay-free loops in discrete-time models of nonlinear acoustic systems," *IEEE Trans. Speech Audio Process.*, vol. 8, no. 5, pp. 597–605, Sept. 2000.

[45] Timothy Schwerdtfeger and Anton Kummert, "Newton's method for modularity-preserving multidimensional wave digital filters," in *Proc. IEEE Int. Work. Multidimensional Syst.*, Vila Real, Portugal, Sept. 2015.

[46] Alberto Bernardini, Paolo Maffezzoni, Luca Daniel, and Augusto Sarti, "Wave-based analysis of large nonlinear photovoltaic arrays," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 4, pp. 1363–1376, Apr. 2018.

[47] Alberto Bernardini, Augusto Sarti, Paolo Maffezzoni, and Luca Daniel, "Wave digital-based variability analysis of electrical mismatch in photovoltaic arrays," in *Proc. IEEE Int. Symposium on Circuits and Systems (ISCAS)*, Florence, Italy, May 27–30 2018.

[48] Alberto Bernardini, Kurt James Werner, Paolo Maffezzoni, and Augusto Sarti, "Wave digital modeling of the diode-based ring modulator," in *Proc. 144nd Conv. Audio Eng. Soc.*, Milan, Italy, May 2018, conv. paper #10015.

[49] C. W. Ho, A. E. Ruehli, and P. A. Brennan, "The modified nodal approach to network analysis," *IEEE Trans. Circuits Syst.*, vol. 22, no. 6, pp. 504–509, June 1975.

DIGITAL GREY BOX MODEL OF THE UNI-VIBE EFFECTS PEDAL

Champ Darabundit *

University of Southern California
Los Angeles, CA
darabund@usc.edu

Russell Wedelich

Eventide Inc.
Little Ferry, NJ
rwedelich@eventide.com

Pete Bischoff

Eventide Inc.
Little Ferry, NJ
pbischoff@eventide.com

ABSTRACT

This paper presents a digital grey box model of a late 1960s era Shin-ei Uni-Vibe®[1] analog effects foot pedal. As an early phase shifter, it achieved wide success in popular music as a unique musical effect, noteworthy for its pulsating and throbbing modulation sounds. The Uni-Vibe is an early series all-pass phaser effect, where each first-order section is a discrete component phase splitter (no operational amplifiers). The dynamic sweeping movement of the effect arises from a single LFO-driven incandescent lamp opto-coupled to the light dependent resistors (LDRs) of each stage. The proposed method combines digital circuit models with measured LDR characteristics for the four phase shift stages of an original Uni-Vibe unit, resulting in an efficient emulation that preserves the character of the Uni-Vibe. In modeling this iconic effect, we also aim to offer some historical and technical insight into the exact nature of its unique sound.

1. INTRODUCTION

The Uni-Vibe was a Shin-ei Companion effects box branded as the Univox Uni-Vibe and distributed by the Unicord Corporation in the late 1960s. The Uni-Vibe is thought to be the prominent guitar effect in popular music recordings such as Jimi Hendrix's Woodstock performance in 1969, Robin Trower's 1974 "Bridge of Sighs," and Pink Floyd's 1973 "Breathe." The sound of the Uni-Vibe is characterized by throbbing pulse, "double beat," and a lo-fi sweep [1]. It has a simple user interface with potentiometer controls for Volume and Intensity, a foot pedal for varying rate or speed, and a switch for Chorus or Vibrato Mode.

While the Uni-Vibe was marketed as a simulation of a rotating "Leslie" style speaker cabinet [2], in a more recent interview the inventor, Fumio Saeda, revealed his inspiration drew more from Radio Moscow broadcasts modulated and distorted by the ionosphere as he listened on short wave radio in Japan [3]. In fact, the Uni-Vibe modulation circuit was largely extracted from his first effects box, the Psychedelic Machine, which was a combination of fuzz and modulation (called Mood), both of which were directly inspired by the combination of fuzzy distortions and pitch modulation of the ionosphere distorted broadcasts.

The Uni-Vibe is considered to be a phaser or phase shifter similar to the MXR Phase 90 of the time [4]. A phaser mixes an input signal with the same signal's output from a series chain of all-pass filters to generate a number of notches in the frequency spectra.

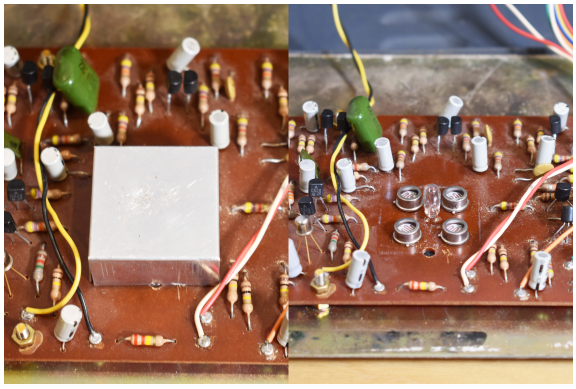


Figure 1: *A closeup of the reflective housing (left) and lamp-LDR assembly (right)*

Each pair of single order all-passes, when mixed with the original signal, creates an instantaneous frequency notch where their 180° phase shifts intersect. For this reason, most phasers match the center frequencies in pairs or use 2nd order filters. Notch locations are modulated by a Low Frequency Oscillator (LFO) sweeping the all-pass center frequencies. As a phaser, the Uni-Vibe is particularly unique in two respects: the instantaneous positions of the notches are determined by the complex interaction between a single incandescent lamp and four adjacent LDRs housed in a roughly cube-shaped reflective chamber (shown in Figure 1), and an almost arbitrary choice of phasing capacitor values. The capacitor values, which determine the all-pass center frequencies, result in staggered or spread notches unlike other phasers which tend to use matched pairs. In addition to a phasing sound this leads to a band limited tremolo-like effect. The exact reasoning for the capacitor selection remains a mystery to Uni-Vibe aficionados [4].

The uniqueness of the Uni-Vibe has thus been the subject of many attempts to commercially clone the original such as ([5] [6] [7]) and [8] by Dunlop Manufacturing who now own the Uni-Vibe trademark, among others. Many of these clones attempt to emulate the Uni-Vibe by recreating the original circuit, or by attempting to replicate a similar lamp and LDR combination.

Previous published work on Uni-Vibe analysis was done in [9] and [10]. Related work in [11] provides a thorough overview for direct digital implementation of generalized analog phasers constructed with operational transconductance amplifiers and field effect transistors. Whereas, the more specific modeling of the MXR Phase 90 pedal from [12] tabulates the main nonlinearity from a JFET used as the notch sweeping variable resistor within a state space discrete model of the circuit. The authors in [13] describe an ad-hoc method for modeling a vactrol, a single element package combining an LED and LDR, in a Buchla low-pass gate. They

* For Eventide Inc.

[1] Uni-Vibe is a registered trademark of Dunlop Manufacturing, Inc.

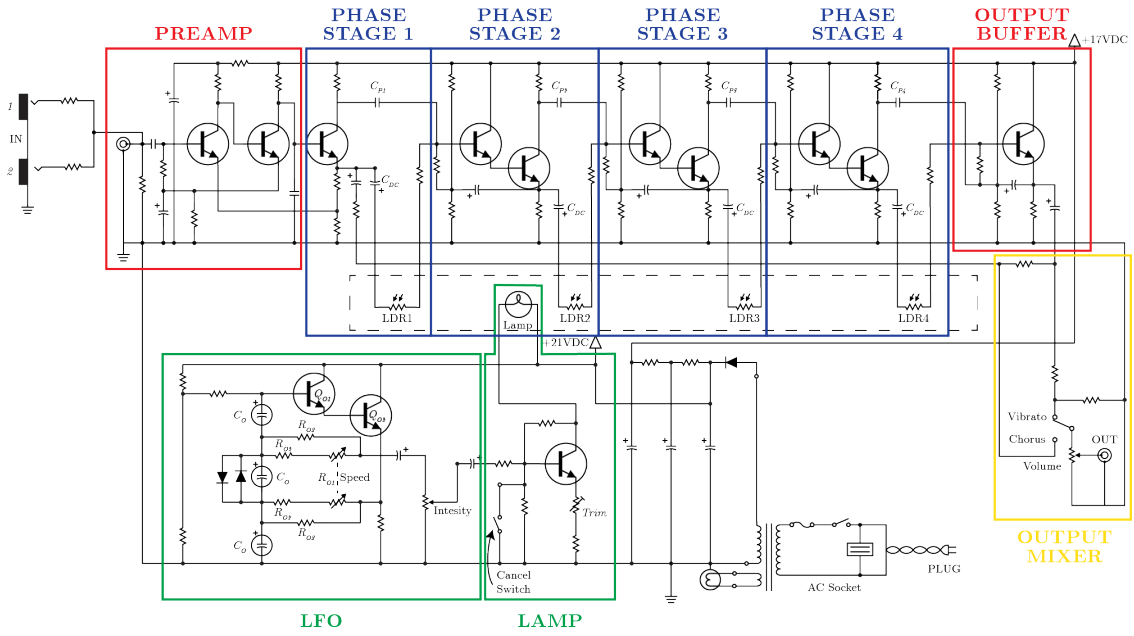


Figure 2: *The complete Uni-Vibe circuit schematic*

reason that realizing full models of photoconductor transients for musical signals is a significant task based on the models in [14].

We propose a digital grey box emulation of the Uni-Vibe, similar to the phaser modeling work done in [15]. The grey box model approach combines physical measurement data (a black box model) and a physically informed model derived from circuit analysis (a white box model). Our primary motivation for this approach, over complete physical modeling of the entire system, arises from the observed complexity in the coupled interaction between the driving LFO circuit, the single incandescent lamp, the four adjacent LDRs, and the reflective housing encasing the lamp and LDRs. We reason that a full model would need to take into account the complex physical relationship of at least the lamp, the LDRs, and the additional reflections and shielding provided by the housing. Since our goal is real-time and efficient implementation of this model, we opted for the black-box approach to this complex system. Consequently, our emulation uses measurement data taken directly from an original Uni-Vibe unit to create three-dimensional wavetables capturing the behavior of each of the LDRs under the influence of the LFO driven incandescent lamp.

While the behavior of the LFO and LDRs is complex, a model of the phasing sections can be derived and discretized via the bilinear transform for the white box portion. Additionally, we will show that the discrete transistor phase splitting does not result in ideal all-passing, and that this does have a perceivable effect in the Uni-Vibe's sound.

This paper will first provide an overview of the Uni-Vibe pedal and circuit in Section 2. In Section 3 we will examine the phasing circuit and the derivation of its continuous-time and discrete-time models, highlighting some of the perceptually relevant aspects yet to be thoroughly covered in previous works. Section 4 will cover the measurement procedure for evaluating the LDRs and the signal analysis done to extract resistance curves from the LDRs. Section 5 will discuss the results and real-time implementation of the preamplifier model of the Uni-Vibe. Section 6 offers conclusions and suggestions for future work.

2. UNI-VIBE CIRCUIT

The Uni-Vibe can be broken down into five basic blocks (as shown in Figure 2): the pre-amplification section, the phasing section, the LFO and lamp section, the output buffer, and the output mixer. The input signal is buffered by the nonlinearity before being passed to the phasing section which is a multi-stage phaser with four phase stages. Each phase stage has its own unique LDR and phasing capacitor, labeled C_P in Figure 2. The LDR and R_6 combined in series forms the resistor-capacitor pair determining the center frequency of the phase-shift contributed by each section.

After passing through the phasing sections, the signal is passed to the output buffer which is a bipolar junction transistor (BJT) buffer, before being passed to the output mixer. Within the output mixer there is the vibrato/chorus switch, along with the volume control knob. The vibrato/chorus switch switches between two resistive networks to determine whether the signal from the output buffer is passed to the output by itself or mixed evenly with a "dry" signal originating from the first phase stage. The volume control is a potentiometer adjusting the amplitude of the output. In comparison to the phasing section and the LFO, the pre-amplification section and the output buffer section do not modulate the audio path in a significant way at low signal levels and their contributions are ignored in our analysis. The effects of these sections at high signal levels due to distortion remain for future work. Volume and the Chorus/Vibrato switch did not need to be extensively modeled as they can be represented by a multiplication and addition, respectively.

2.1. LFO Section

This section contains the most dominant Uni-Vibe controls, Intensity and Speed, found in the LFO section of Figure 2.The Intensity

knob, which goes from a value of one to nine, with additional minimum and maximum settings, scales the LFO voltage across the bulb, and thus determines the range of the notches' frequencies in the phase-shifted signal.The effect of sweeping over a larger frequency at the same LFO speed is what causes notches to sound "deeper." The Speed, controlled via foot-pedal, determines the rate of the LFO. From toe-to-heel the Speed traverses approximately 0 Hz to 7.6 Hz. The 0 Hz Speed originates from a cancel switch located at the heel of the foot pedal that turns off the lamp and prevents the value of the LDRs from changing.

The LFO is a variation on a phase-shift oscillator [4]. A phase-shift oscillator uses regenerative feedback from an RC network from the base of Q_{O1} to the emitter of Q_{O2} to produce a sinusoidal output. The RC network here is the equivalent resistance of the two R_{O1}, R_{O2}, R_{O3} legs along with the three C_O capacitors. The foot pedal controls the speed by varying the value of the coupled potentiometers R_{O1}, which in turn vary the equivalent resistance of the two resistor legs.

Before the LFO signal is passed to the BJT buffer in the lamp section, its amplitude is modulated via the Intensity potentiometer. As the rate of the LFO is dependent on regenerative feedback, coupling the Intensity pot directly to the output of the LFO results in rate drifting proportionally with intensity and vice-versa, resulting in a non-orthogonal relationship between Speed and Intensity. In order to accurately model this parameter interdependency, our black box model utilizes a three-dimensional wavetable, with input phase/sample, speed, and intensity axes.

2.2. Lamp and LDRs

The lamp section of the Uni-Vibe is rather straightforward and consists of a BJT buffer which supplies the LFO signal to the lamp. Within this section is the cancel switch and a trim pot which was not adjusted during our measurements. When the cancel switch is flipped all power to the bulb is cut. The resistance of an LDR is inversely proportional to brightness therefore in complete darkness an LDR is at its maximum resistance. As the center frequency of the phase shifter is inversely proportional to the resistance of the LDR, the cancel switch causes the center frequencies to trend to DC. The cancel switch thus effectively acts as a bypass.

The lamp and the LDRs were specially sourced for the Uni-Vibe. The lamp is a fast-switching incandescent bulb (but not as fast as an LED) and the LDRs are made with cadmium sulfide and are no longer mass produced [3]. The uniqueness and rarity of these components is what, in part, is responsible for the unique tone of the Uni-Vibe.

3. PHASING CIRCUIT ANALYSIS AND MODEL

Figure 3 shows the schematic for phase stages 2, 3, and 4 of the phasing circuit. Each stage consists of a Darlington emitter follower that directly drives a phase inverter whose center frequency is determined by an LDR and phasing capacitor pair, labeled LDR and C_p, respectively. Due to the nature of the Darlington amplifier circuit and C_1 acting as a "bootstrap" capacitor the input impedance of each stage is very high [4]. This allows us to consider each phase stage individually, instead of as a whole. Although phase stage 1 is not driven by a Darlington amplifier, we assume similar behavior as the other phase stages because of the preamplifier section . A similar analysis of this circuit has been done by [9], who modeled the response of each phase stage as

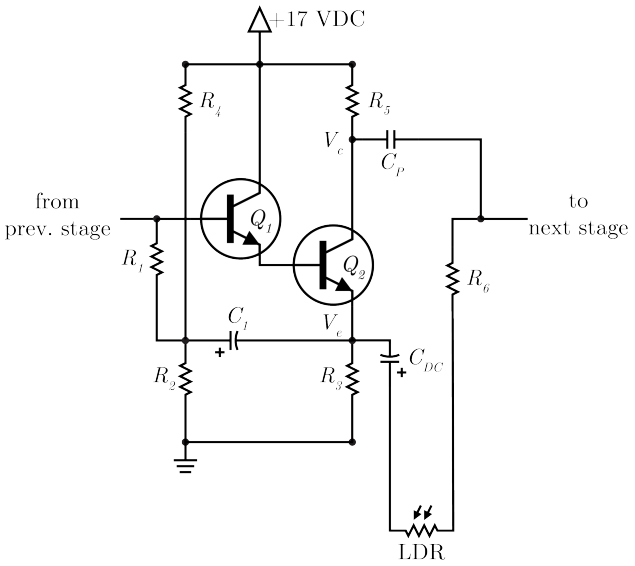


Figure 3: *A single stage of the phasing circuit. The inverting side comes comes off the collector and the non-inverting side comes off the emitter*

an all-pass phaser by representing the Darlington emitter-follower as an phase inverter with gain ≈ 1 feeding an RC bridge. This conclusion assumes, incorrectly, that the amplitude of inverting and non-inverting legs of the transistor originating from the collector and emitter of Q_2 are balanced, and that the block capacitor's (C_{DC}) contribution can be ignored. Through listening, measurement, and simulation of the Uni-Vibe it was determined that these factors could not be ignored and had to be taken into account, thus a corrected analysis of the circuit is presented below.

3.1. Phasing Circuit Analysis

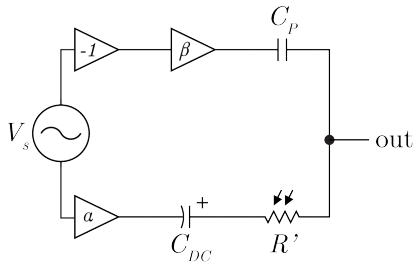


Figure 4: *A block diagram describing the corrected phasing circuit.* $R' = \text{LDR} + R_6$

Figure 4 represents the incoming signal to each stage as V_s, the phase inversion as a gain of -1, and the inverting and non-inverting gains as β and α, respectively. The phasing capacitor and the block capacitor remain unchanged while the value R' represents the series resistance of the LDR and R_6.

The transfer function $H(\omega)$ of the phasing circuit can be found through the superposition of two complex voltage dividers taken from the inverting and non-inverting legs of the circuit. Analyzing the phasing circuit in this manner makes a digital implementation

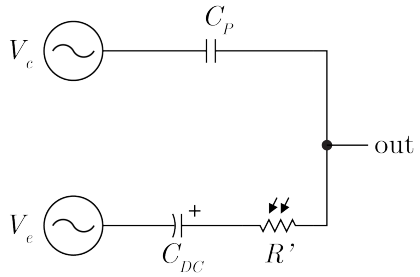


Figure 5: *A further simplification of the phasing block diagram where the V_s, α, and β have been simplified as an inverting and non-inverting source*

of the filter and application of nonlinearity easier as will be shown in Sec. 3.2. This simplification is represented in Figure 5, where:

$$V_e = \alpha V_s \tag{1}$$

and,

$$V_c = -\beta V_s \tag{2}$$

Giving us the complete transfer function:

$$H(\omega) = H_c(\omega) + H_e(\omega) \tag{3}$$

$H_c(\omega)$ and $H_e(\omega)$ are the transfer functions of the inverting and non-inverting legs:

$$H_c(\omega) = \frac{\kappa_c \omega_o + j\omega}{\omega_o + j\omega} \tag{4}$$

$$H_e(\omega) = \frac{\kappa_e \omega_o}{\omega_o + j\omega} \tag{5}$$

Where:

$$\kappa_c = \frac{C_p}{C_p + C_{DC}} \tag{6}$$

$$\kappa_e = \frac{C_{DC}}{C_p + C_{DC}} \tag{7}$$

are constants, and

$$\omega_o = \frac{1}{R'} \frac{C_{DC} + C_p}{C_p C_{DC}} \tag{8}$$

is the center frequency of the filter. If we combine equations (1) (2) (3) (4) (5) we have the transfer function of the entire phasing circuit:

$$H(\omega) = \alpha \left(\frac{\kappa_e \omega_o}{\omega_o + j\omega} \right) - \beta \left(\frac{\kappa_c \omega_o + j\omega}{\omega_o + j\omega} \right) \tag{9}$$

Which has the phase response:

$$\phi(\omega) = \tan^{-1} \left(\frac{-\beta\omega}{\alpha\kappa_e\omega_o - \beta\kappa_c\omega_o} \right) - \tan^{-1} \left(\frac{\omega}{\omega_o} \right) \tag{10}$$

If we let $\beta = 1$, $\alpha = 1$ (i.e. the inverting and non-inverting gains are balanced) and $C_{DC} = \infty$ (which removes the effect of the block-capactior) equation (9) takes the form of a standard first-order all-pass phaser in equation (11), as $\kappa_c = 0$, and $\kappa_e = 1$.

$$H(\omega) = \frac{\omega_o - j\omega}{\omega_o + j\omega} \tag{11}$$

Which matches the analysis done by [9]. If the same equalities are applied to equation (10) we obtain the standard phase response of a first-order all pass:

$$\phi(\omega) = -2\tan^{-1} \left(\frac{\omega}{\omega_o} \right) \tag{12}$$

Since the bootstrap capacitor C_1 allows us to assume high input impedance at each stage the full cascade of the phasing section is:

$$H(\omega) = \prod_{n=1}^{4} H_{c_n}(\omega) + H_{e_n}(\omega) \tag{13}$$

Where,

$$H_{c_n}(\omega) = \frac{\kappa_{c_n}\omega_{o_n} + j\omega}{\omega_{o_n} + j\omega} \tag{14}$$

$$H_{e_n}(\omega) = \frac{\kappa_{e_n}\omega_{o_n}}{\omega_{o_n} + j\omega} \tag{15}$$

and,

$$\kappa_{c_n} = \frac{C_{p_n}}{C_{p_n} + C_{DC}} \tag{16}$$

$$\kappa_{e_n} = \frac{C_{DC}}{C_{p_n} + C_{DC}} \tag{17}$$

$$\omega_{on} = \frac{1}{R'_n} \frac{C_{DC} + C_{p_n}}{C_{p_n} C_{DC}} \tag{18}$$

for each phase-stage ($n = 1, 2, 3, 4$).

Table 1: *Measured capacitance, inverting, and non-inverting gains, and LDR resistances*

	C_p (*F*)	α	β	LDR (Ω)		
				avg (M)	*min. (k)*	*max. (M)*
1	.015μ	1.01	1.11	0.405	12.7	2.79
2	.22μ	.98	1.09	0.233	6.86	2.59
3	470p	.97	1.10	0.29	7.69	3.32
2	.0047μ	.95	1.09	0.240	6.22	4.16

Each stage has its own unique H_{c_n}, H_{e_n}, κ_{c_n}, κ_{e_n}, and ω_{on} as each stage has a unique phasing capacitor, LDR, inverting gain, and non-inverting gain which were found through measurement of each phase stage. Table 1 provides these values. We should note that these values are most likely unique to the particular unit we measured, and we expect these values are varied among differing Uni-Vibe units given the tolerances of the components. Different units most likely have their own particular sound.

The inclusion of the inverting and non-inverting gains as well as the block capacitor in the analysis of the phasing circuit are a sonically relevant addition to the analysis and modeling of the Uni-Vibe. As seen in Figure 6 with R' fixed at LDR1's mean value the block capacitor causes the magnitude of the transfer function to exhibit a high shelf response rather a unity response all-pass filter. This high shelf effect is emphasized as the stages stack up. Additionally dissimilar inverting and non-inverting gains, along with the block capacitor cause a shift in the phase response of the filter as seen in equation (10). In chorus mode the sweeping high shelf causes the low end of the signal to match what the output

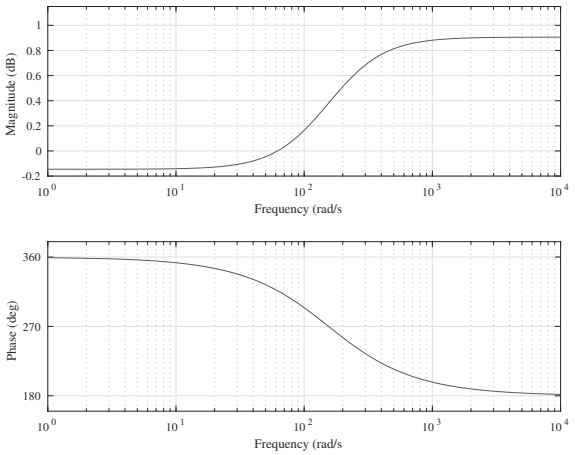


Figure 6: *Transfer function of the first phase stage.* $\alpha = 1.0009$, $\beta = 1.11$, $R' = 409.7\text{k}\Omega$

would be in vibrato mode and is responsible for the Uni-Vibe's characteristic throbbing pulse.

We reason that the selection of the phasing capacitor values is such that only two notches can heard at a time, which creates the Uni-Vibe's characteristic double beat. As the Uni-Vibe stages are not matched pairs like other phasers, without the inclusion of the block capacitor instantaneous notches can "pop" in and out of the frequency spectra.

3.2. Discrete Model

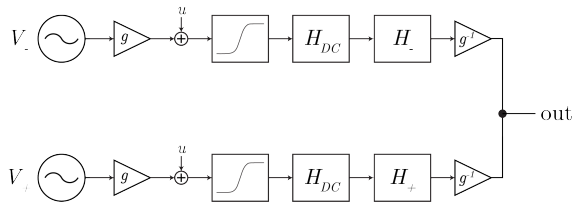


Figure 7: *Block diagram of the applied nonlinearity*

From equations (14) (15) a discrete model of each phase stage circuit was derived using the bilinear transform and pre-warping the filter around the frequency ω_{on} giving us the discrete transfer functions for the inverting and non-inverting legs:

$$H_{c_n}[z] = \frac{(\kappa_{c_n}\tan\omega_{o_n} + 1) + (\kappa_{c_n}\tan\omega_{o_n} - 1)z^{-1}}{(\tan\omega_{o_n} + 1) + (\tan\omega_{o_n} - 1)z^{-1}} \tag{19}$$

$$H_{e_n}[z] = \frac{\kappa_{e_n}(1 + z^{-1})}{(\tan\omega_{o_n} + 1) + (\tan\omega_{o_n} - 1)z^{-1}} \tag{20}$$

in the complete discrete equation:

$$H_n[z] = H_{c_n}[z] + H_{e_n}[z] \tag{21}$$

for each stage $(n = 1, 2, 3, 4)$.

It is important to note that ω_{on} is dependent on the value of the LDR and therefore the bilinear transform and frequency pre-warping must be applied in real-time.

3.2.1. Non-linear elements

It was determined through measurements that the emitter-follower pair in each phase-stage was clipping the input signal asymmetrically. To approximate this non-linear behavior, a tanh() function with a heuristically determined bias and scaling factor is applied in the signal chain before (21). Separating our signal into two signal paths allows us to apply this nonlinearity. A block-diagram of how this nonlinearity is incorporated is shown in Figure 7. Including these non-linearities was necessary in modeling the Uni-Vibe as the clipping added audible harmonics and the biasing adjusted the balance of even and odd harmonics.

The scaling factor g determines the level at which the signal clips by gaining down the input signal. The biasing factor u offsets the signal to determine the extent of the asymmetry in the clipping. The ideal values for g and u were first determined through measurement of the original Uni-Vibe pedal, and then tuned heuristically to match. After the signal is clipped by the tanh() function it is passed through a DC-blocking low-pass filter. The inclusion of this filter is an improvement on the original signal path of the Uni-Vibe. The biasing factor u, which exists in both the Uni-Vibe and in our model, pushes DC impulses through to the output when ω_o approaches DC. These impulses are audible as clicking in the original Uni-Vibe. Our DC-blocking filter removes the DC offset before equation (19) (20) are applied. Before being outputted the signal is gained back up by g^{-1} to return the signal level back to its original value. The gain values are fairly low, so aliasing distortion due to clipping is also fairly minimal, thus in order to keep the model efficient we neglected any oversampling.

4. MEASURING THE UNI-VIBE LDRS

4.1. Physical Measurement

Due to the difficulty in physically modeling the transient resistance of the LDRs in the phase stage, a decision was made to measure the behavior instead. Unlike [10] which measures a Uni-Vibe clone, we directly measure the actual Uni-Vibe unit. Additionally [10] only measures one LDR, while we found it important to measure all four as we could not ensure the LDRs would have the same values based on their make and position respective to the lamp. This decision is supported by Table 1 which demonstrates that each LDR differed from one another. To measure the behavior of a phase stage's LDR the output of the stage was shorted and the input was fed with a 1 kHz sine wave. Voltage measurements were taken across the LDR and R_6 and across R_6 alone, allowing us to derive the transient resistance of the LDR as part of a voltage divider.

$$R_{\text{LDR}} = R_6\frac{V_{\text{LDR}}}{V_6} - R_6 \tag{22}$$

Measurement recordings were taken by playing the 1 kHz input signal while in tandem taking the two aforementioned voltage measurements. These measurements were recorded on a digital audio workstation using the Expert Sleepers ES-8 USB-audio interface which has DC-coupled inputs and outputs [16]. To model the full breadth of the pedal these measurements were repeated at fourteen different LFO frequencies and at eleven different intensities settings for each of the four LDRs. Consequently an additional measurement was taken of the LFO signal driving the lamp, as a method of recording the speed of the LFO in the case of any inconsistencies in the foot pedal settings during measurement.

4.2. Measurement Analysis

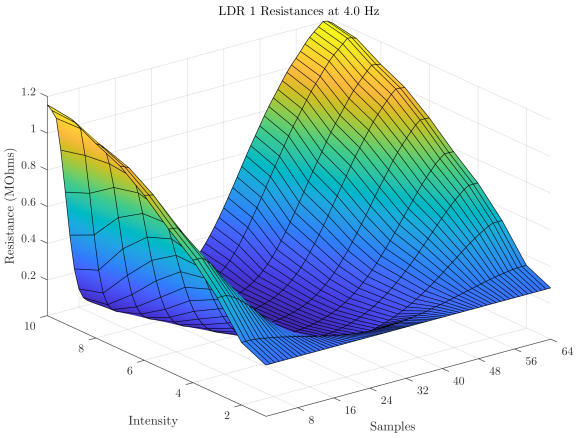


Figure 8: *A complete wavetable for LDR1 at 4 Hz*

The process taken to analyze the measurement for each LDR were as follows. The measurement data was converted to their nominal RMS values based on the specifications of the USB-audio interface. The recordings were then upsampled by four so envelope extraction could be performed using a maximum value filter with a rectangular window with a width of 1 kHz. This allowed us to extract the effect of the circuitry and remove the 1 kHz input signal. Additional filtering was also done to remove power-supply hum. A model of the additive noise in the LDR resistances was found by deriving the transient resistance of an LDR using equation (22) at a minimum Intensity for each Speed. This noise model could then be used in conjunction with a Wiener filter, using Matlab's `wiener` function, to remove any noise in transient resistances at higher Intensity settings.

At this point, single-cycle resistance curves needed to be extracted from the clean resistance signal. To preserve more perceptible information single-cycle curves were extracted by using local maximums to determine the starting and ending peaks for individual periods in the resistance signal. Unfortunately information is lost at the loop point from the start to the end of a cycle. To minimize this local maximums were chosen as the loop point because ω_o values arising from the resistances at local minimums are much more perceptible than ω_o values arising from the local maximums, which approach DC. Each single-cycle period was averaged together to further reduce noise in the resistance signal.

To prepare these resistance curves for implementation in the wavetable each one-period length curve was then downsampled to 64-samples. Then, using the LFO rate data obtained from an FFT of the LFO signal, curves of a similar Intensity were interpolated to be equally spaced across the entire frequency range of the LFO. This was done to facilitate indexing across the lookup table. The first two-samples and last two-samples of each curve were interpolated together to ensure that the loop point was seamless at the expense of data loss. Otherwise, any significant disjoint would be perceived as an audible click. The result of this analysis is a three-dimensional wavetable containing 640 unique LDR resistance curves for each LDR representing every combination of Speed and Intensity settings on the Uni-Vibe, with a sawtooth phase input providing exact sample accurate rate. Figure 8 shows a small sample of this wavetable containing the curves for the first LDR at a speed of 4 Hz. Visual analysis of the resistance curves shows the curves have increasing asymmetry as intensity increases due to the LDRs having different turn-on and turn-off times. The increasing asymmetry is a result of lamp varying its brightness more rapidly as intensity is increased.

5. RESULTS AND EXPERIMENTS

5.1. Real Time Implementation

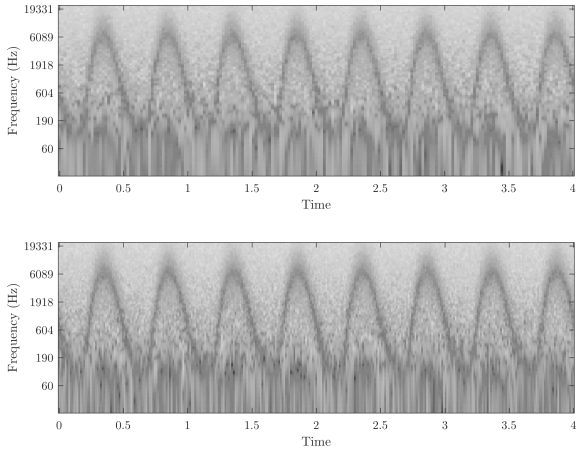


Figure 9: *Noise Spectrograms. Top: Model, Bottom: Uni-Vibe. A second notch can be seen in the lower frequencies under the main notch.*

A real-time implementation of the the Uni-Vibe emulation was implemented in C/C++ both as a VST plugin using the JUCE framework and as a process running in a single core of an embedded Linux ARM Cortex-A9 commercial audio product. The real-time emulation consists of four *univibe_phaser* modules each of which replicates a single stage of the phasing circuit by using the measurement derived three-dimensional wavetable and the discrete-time model of the Uni-Vibe's phase-stage.

Depending on the selection of which stage to emulate the *univibe_phaser* module converts the wavetable of LDR resistances into a wavetable of center frequencies ω_o before implementation using the known values of each stages' C_p and C_{DC}. Using a graphical user interface the desired Intensity and LFO Speed can be set to linearly interpolate a value of ω_o from the wavetable. This value is then passed to an implementation of equation (13).

Figure 9 shows spectrogram output of our real-time implementation and the original Uni-Vibe with white noise as the input source. Upon visual inspection, the model retains the dynamic behavior of the original unit. Intensity and Speed settings were 7 and 2.0 Hz respectively.

5.2. Experimental Methodology

MUSHRA style listening tests were conducted to quantitatively assess the accuracy of our real-time implementation of the Uni-Vibe running in the ARM Cortex-A9 commercial audio product. A second original Shin-ei Uni-Vibe, two analog hardware clones, and two digital emulations in hardware were tested alongside our

implementation for a total of 6 units under test. The original hardware unit measured for our implementation was chosen as the MUSHRA reference.

5.2.1. MUSHRA Test Setup

Reference and MUSHRA anchor recordings were generated as follows: Unaffected electric guitar passages were recorded into a UAD Apollo 8 interface via Direct Injection, or DI. Reference recordings were created by feeding recorded passages through the original Uni-Vibe set at specific Intensity and Speed settings. All recordings were done in Chorus mode with Volume set at 5. Due to the relatively low frequency spectral quality of the electric guitar, 1.5 kHz low-passed versions of the reference recordings were included as anchors instead of the usual 7 kHz and 3.5 kHz low-passed anchors.

For test unit recordings, Intensity settings were visually matched to the reference through printed markings or numerically via graphical displays where applicable. Test unit LFO Speed settings were precisely matched to the reference through waveform inspection of their phase cancellation cycle or numerically via graphical displays where applicable. Trial and error was used to ensure LFO phase alignment between test unit recordings and the reference recording.

5.2.2. Electric Guitar Passage Selection

A total of three electric guitar passages were recorded for the listening tests. Passages were disparately styled and chosen to reflect real-world use cases for the Uni-Vibe in addition to test the full range of Uni-Vibe sounds and settings.

Passage 1 consisted of a 101 BPM Texas Blues styled riff played with hard plectrum attack intended for a Uni-Vibe set to approximate a Leslie in tremolo mode. Reference Intensity and Speed settings were 5 and 4.85 Hz respectively creating a triplet modulation feel.

Passage 2 consisted of a 4 bar phrase using whole note chords followed by an arpeggiated version of the same chord progression played at 110 BPM. Reference Intensity and Speed settings were 7 and 1.89 Hz respectively for a medium depth modulation that followed the quarter note.

Passage 3 consisted of a 62 BPM, two open-chord arpeggiation. Intensity and Speed settings were set at 10 and 0.99 Hz respectively creating a slow and deeply swept modulation. Settings were chosen so participants could best judge LFO modulation contours in the test.

5.3. Experimental Results

Listening tests were administered using webMUSHRA software [17] and a total of 20 participants were included in the test. Figure 10 shows a box plot of the results of listening test 1 where our model scored closest to the reference. Of the analog hardware units tested, scores varied widely with two units averaging below the digital emulations. The second original Uni-Vibe scored substantially lower than the reference which seemed to confirm our initial listening impressions of the high degree of variability between these specific original units. Given the majority of units tested closer to the reference, we suspect the second original Uni-Vibe could be out of factory specification warranting further investigation. Post-test user feedback for listening test 1 revealed a wide range of criteria for judging similarity and included pick

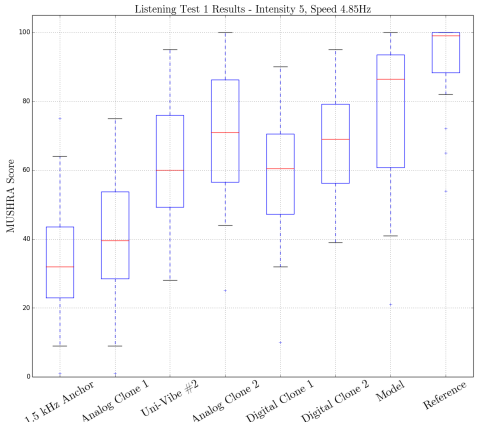


Figure 10: *Listening test 3 results*

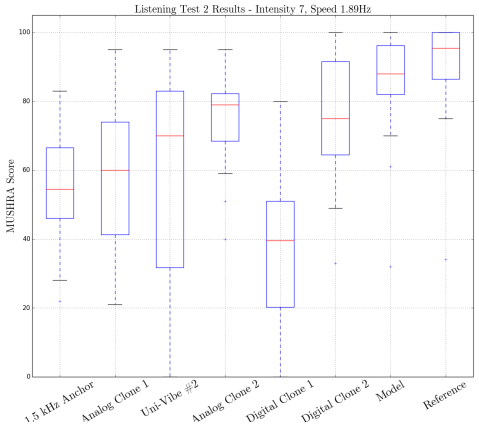


Figure 11: *Listening test 2 results*

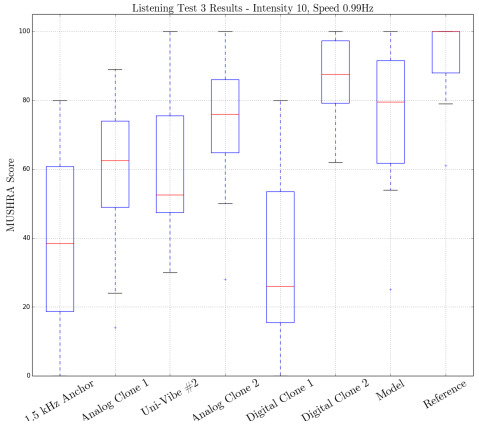


Figure 12: *Listening test 3 results*

attack, tonal balance (timbre), modulation depth, and noise floor. The wide range of judgement criteria may explain the large interquartile sizes and low score outliers in these results.

Figure 11 shows a box plot of the results of listening test 2. These results mirrored listening test 1 however, Digital Clone 1 scored below the anchor. During our own informal comparisons, Digital Clone 1's depth of modulation and overall timbre were markedly different than the reference, both most likely playing large roles when tests were scored by participants. Anchor scores averaged higher than listening test 1 indicating low pass frequency may have been set too high. Post-test user feedback reported listening test 2 as harder to judge than listening test 1 which could explain the average score increase for most units.

Figure 12 shows a box plot of the results from listening test 3. As in test 2, Digital Clone 1 scored lowered than the anchor, most likely for reasons similar to listening test 2. Our model scored second closest to the reference with Digital Clone 2 scoring the highest. Post-test user consensus deemed listening test 3 hardest to score of the three tests. From our own qualitative listening sessions at these settings, our implementation exhibited a slightly deeper and squarer-edged modulation contour than the reference. We believe this is related to our choice of wavetable splice point and interpolation.

6. CONCLUSION

In this work a grey box method was proposed for creation of an accurate Uni-Vibe model. A white-box model of the Uni-Vibe's phasing circuit was created through circuit analysis uncovering aspects of the Uni-Vibe circuit that contribute to its iconic sound. A complementary black-box model was also created through measurement of the Uni-Vibe's LFO-LDR-lamp interaction. A real-time implementation of the grey box model of the Uni-Vibe was implemented on a single core of a Linux ARM Cortex-A9. In comparison to an original unit and other Uni-Vibe clones our implementation was on-average rated closest to the reference in two out of three MUSHRA listening tests, scoring second in the third test.

Further work to improve our model would be deriving and implementing more accurate behavior of the non-linear BJT clipping, as was done by [18], instead of replicating it heuristically. We also plan to further investigate the ideal method of splicing at the wavetable loop point. Conversely, judging from our measurements of the LFO-LDR-lamp interaction, it may be also possible to create a simplified parametric model of that interaction.

7. ACKNOWLEDGMENTS

Special thank you to Chuck Zwicky (www.zmix.net) for making Uni-Vibe recordings of his original unit for the listening test comparisons.

8. REFERENCES

[1] H. Shapiro, M. Heatley, and R. Mayer, *Jimi Hendrix Gear*, Voyageur Press, Minneapolis, MN, USA, 2014.

[2] Unicord Incorporated, *Uni-Vibe Operating Manual*, 1968.

[3] M. Yoshiyuki, "What is Uni-Vibe? pt.1," `https://www.digimart.net/magazine/article/2018052303189.html`, May 5 2018.

[4] R. G. Keen, "The technology of the univibe," `http://www.geofex.com`, accessed March 5, 2019.

[5] MJM Guitar FX, "Sixties Vibe Classic," `http://mjmguitarfx.com/product/sixties-vibe-classic/`, accessed March 5, 2019.

[6] Black Cat Pedals, "Black Cat Vibe," `http://www.blackcatpedals.com/black-cat-vibe/`, accessed March 5, 2019.

[7] Fulltone, "Mini Dejavibe 3 V2," `https://www.fulltone.com/products/mini-dejavibe-3-v2`, accessed March 05, 2019.

[8] Dunlop Manufacturing Inc., "Univibe Chorus/Vibrator," `https://www.jimdunlop.com/product/m68-7-10137-07227-5.do`, accessed March 05, 2019.

[9] C. Hahlweg and H. Rothe, "The unique sound of the Uni-Vibe pedal," in *Proc. SPIE 8487, Novel Optical Systems Design and Optimization XV*, San Diego, California, United States, Oct. 19 2012.

[10] C. Hahlweg and H. Rothe, "The unique sound of the Uni-Vibe pedal: Part ii. transient behaviour," in *Proc. SPIE 8842, Novel Optical Systems Design and Optimization XVI2*, San Diego, California, United States, Sept. 30 2013, pp. 1–8.

[11] A. Huovilainen, "Enhanced digital models for analog modulation effects," in *Proc. Digital Audio Effects (DAFx-05)*, Madrid, Spain, Sept. 20–22, 2005, pp. 155–160.

[12] F. Eichas, M. Fink, M. Holters, and U. Zölzer, "Physical modeling of the MXR phase 90 guitar effect pedal," in *Proc. Digital Audio Effects (DAFx-14)*, Erlangen, Germany, Sept. 1–5, 2014, pp. 153–158.

[13] J. Parker and S. D'Angelo, "A digital model of the buchla lowpass gate," in *Proc. Digital Audio Effects (DAFx-13)*, Maynooth, Ireland, Sept. 2–6, 2013, pp. 278–285.

[14] A. E. Iverson and D. L. Smith, "Mathematical modeling of photoconductor transient response," *IEEE Trans. on Electron Devices*, vol. 34, no. 10, pp. 2098–2107, 1987.

[15] R. Kiiski, F. Esqueda, and V. Välimäki, "Time variant grey-box modeling of a phaser pedal," in *Proc. Digital Audio Effects (DAFx-16)*, Brno, Czech Republic, Sept. 5–9, 2016, pp. 31–38.

[16] Expert Sleepers Ltd., "ES-8 USB Audio Interface," `https://www.expert-sleepers.co.uk/es8.html`, accessed June 19, 2019.

[17] Michael Schoeffler, Sarah Bartoschek, Fabian-Robert Stöter, Marlene Roess, Susanne Westphal, Bernd Edler, and Jürgen Herre, "webMUSHRA: a comprehensive framework for web-based listening tests," *Journal of Open Research Software*, vol. 6, no. 1, 2018.

[18] D. T. Yeh, "Automated physical modeling of nonlinear audio circuits for real-time audio effects - part ii: BJT and vacuum tube examples," *IEEE Trans. on Speech and Audio Processing*, vol. 18, no. 3, pp. 1207–1216, 2011.

SOUND SOURCE SEPARATION IN THE HIGHER ORDER AMBISONICS DOMAIN

Mohammed Hafsati
b<>com, Rennes, France

Nicolas Epain
b<>com, Rennes, France

Rémi Gribonval
Univ Rennes, Inria, CNRS, IRISA, France

Nancy Bertin
Univ Rennes, Inria, CNRS, IRISA, France

ABSTRACT

In this article we investigate how the *local Gaussian model* (LGM) can be applied to separate sound sources in the higher-order ambisonics (HOA) domain. First, we show that in the HOA domain, the mathematical formalism of the local Gaussian model remains the same as in the microphone domain. Second, using an off-the shelf source separation toolbox (FASST) based on the local Gaussian model, we validate the efficiency of the approach in the HOA domain by comparing the performance of toolbox in the HOA domain with its performance in the microphone domain. To do this we discuss and run some simulations to ensure a fair comparison. Third, we check the efficiency of the local Gaussian model compared to other available source separation techniques in the HOA domain. Simulation results show that separating sources in the HOA domain results in a 1 to 12 dB increase in signal-to-distortion ratio, compared to the microphone domain.

Multichannel source separation, local Gaussian model, Wiener filtering, 3D audio, Higher Order Ambisonics (HOA).

1. INTRODUCTION

There is an increasing interest in new, immersive forms of media, such as 360-degree videos and Virtual Reality (VR) experiences. Producing media experiences of this kind requires new techniques and workflows on both the video and audio sides. In particular, the feeling of immersion that is sought after by VR spectators highly depends on the quality of the audio rendering. In order for the experience to be convincing, sounds must be binauralized according to the spectator's location and orientation relative to the different sound sources.

Among the various 3D-audio technologies available, Higher-Order Ambisonics (HOA) [1, 2, 3, 4] has become the *de facto* standard for 360-degree video soundtracks. This is primarily because HOA provides a panoramic representation of the sound field, which can easily be rotated in accordance with the listener's head orientation prior to being played back. Another significant advantage of the HOA representation is that it is straightforward to record HOA sound scenes using relatively compact Spherical Microphone Arrays (SMAs). In contrast to 360-degree videos, however, VR experiences not only allow the spectator to look in any direction, but also to navigate through the virtual environment. This means that the spectator's perspective of the scene may change over time in a manner that cannot be modeled as a simple rotation effect. For instance, the spectator can move toward one of the sound sources, which should translate in this source being louder compared to other sources.

One possible approach to simulate a movement through the scene is to interpolate between several HOA representations corresponding to different points of view [5, 6]. However, in practice this requires to use several SMAs, which may be impossible. Another possibility consists in decomposing one HOA scene into directional components, which typically correspond to sound sources, and changing their directions and gains according to the listener's movements [7, 8, 9]. In addition to being more practical than the interpolation method, this approach was shown to yield a better listening experience [8]. The quality of the navigation effect obtained with the decomposition approach ultimately depends on the accuracy of the sound source separation. In previous work [9], we showed that sound sources could be separated using a simple beamforming technique, which relies solely on spatial information. In the presence of complex sound scenes, however, the quality of the separation could be improved using multi-channel source separation algorithms, such as those based on the so-called *local Gaussian model* [10, 11, 12].

In this article, we investigate the ability of the local Gaussian model to handle the informed source separation problem (knowing directions of arrival) directly in the HOA domain. In Section. 2, we recall the model in the microphone domain and derive its equivalent in the HOA domain. This allows us to perform experiments with an off-the-shelf source separation toolbox, the Flexible Audio Source Separation Toolbox (FASST) [13], presented in Section 3. Experimental results are presented and discussed in Section 5; First, on a small dataset, we investigate the performance of FASST with respect to of the number of channels in the HOA domain and the number of microphones in the microphone domain, and select a number of channels/microphones yielding a fair comparison between the HOA domain and microphone domain. Second, on a large scale dataset, we measure the performance of FASST in both domains, and compare them to each other. Third, we compare the performance of FASST with different types of beamformers. We conclude in Section 6 on the validity of the local Gaussian model and its competitive performance when applied to HOA signals, including in challenging situations such as highly reverberating environments or close source positions.

2. MIXTURE MODELS

2.1. Microphone domain

The source separation problem consists in estimating the contribution $\mathbf{c}_{j,t} \in \mathbb{R}^I$ of each source $j = 1, ..., J$ in each microphone $i = 1, ..., I$ and at each time instant $t = 1, ..., T$. In the absence of noise, the mixture can be written as:

$$\mathbf{x}_t = \sum_{j=1}^{J} \mathbf{c}_{j,t}\,, \tag{1}$$

where $\mathbf{x}_t = [x_{1,t}, \ldots, x_{I,t}]^\mathsf{T} \in \mathbb{R}^I$ are microphone array signals. In a reverberant environment, under the hypothesis of point sources, the source signal $s_{j,t}$ can be related to its contribution $\mathbf{c}_{j,t}$ through:

$$\mathbf{c}_{j,t} = [\alpha_{ij} * s_j]_t\,, \tag{2}$$

where $*$ denotes the convolution product, α_{ij} is the impulse response of the mixing filter between the source j, and the microphone i. Now, under the narrow-band approximation, and assuming the mixing filters are time invariant, the Short-Time Fourier Transform (STFT) of the microphone signals is given by:

$$\mathbf{x}_{f,n} = \sum_{j=1}^{J} \mathbf{A}_{j,f}\mathbf{s}_{j,fn}\,, \tag{3}$$

where f and n denote the frequency bin and time-frame index, respectively. $\mathbf{A}_f \in \mathbb{C}^{I\times J}$ contains the frequency responses $a_{ij,f}$ of the filters $a_{ij}(t)$, and embeds information on the source directions of arrival (DOA).

The source separation problem as defined in Eq.(1) can be addressed using the multichannel Wiener filtering framework, which will be presented with more details in Sec. 3. This framework requires the selection of a distribution model for the variables to estimate. For simplicity we use the local Gaussian model presented in [14]:

$$\forall f \in [1, F]\,, n \in [1, N]\,, \quad \mathbf{c}_{j,f,n} \sim \mathcal{N}_c(0, \mathbf{\Sigma}_{\mathbf{c}_{j,f,n}})\,, \tag{4}$$

where $\mathbf{\Sigma}_{\mathbf{c}_{j,f,n}} = \mathbb{E}\left[\mathbf{c}_{j,f,n}\,\mathbf{c}_{j,f,n}^\mathsf{H}\right]$ is the covariance matrix of the contribution of the j-th source to every microphone at frequency f and time frame n. In line with the literature, this matrix can be further decomposed as the product of a scalar spectral part, $v_{j,f,n}$, with a time-invariant spatial matrix, $\mathbf{R}_{\mathbf{c}_{j,f}}$, as follows: $\mathbf{\Sigma}_{\mathbf{c}_{j,f,n}} = v_{j,f,n}\,\mathbf{R}_{\mathbf{c}_{j,f}}$. Notably, the so-called spatial covariance matrix $\mathbf{R}_{\mathbf{c}_{j,f}}$ respects the relation $\mathbf{R}_{\mathbf{c}_{j,f}} = \mathbf{A}_{j,f}\,\mathbf{A}_{j,f}^\mathsf{H}$ when the assumptions of Eq. (3) hold.

2.2. HOA domain

In the Higher-Order Ambisonic (HOA) framework, the sound field is decomposed over a basis of spherical harmonic functions. The HOA signals, $\mathbf{z}_t$, are typically obtained by applying a set of finite impulse response filters, known as encoding filters, to the signals recorded by a spherical microphone array [15]. Thus, assuming the encoding filters are short enough, the vector of the HOA signal STFTs $\mathbf{z}_{f,n} \in \mathbb{C}^M$ is given by:

$$\mathbf{z}_{f,n} = \mathbf{E}_f\,\mathbf{x}_{f,n}, \tag{5}$$

where $\mathbf{E}_f$ is the matrix of the encoding filter frequency responses. Using Eq. (1), we can now model the HOA mixture as follows:

$$\mathbf{z}_{f,n} = \sum_{j=1}^{J} \mathbf{E}_f\,\mathbf{c}_{j,f,n}, \tag{6}$$

and identify the contribution of the j-th source to the different HOA channels as:

$$\mathbf{b}_{j,f,n} = \mathbf{E}_f\,\mathbf{c}_{j,f,n}. \tag{7}$$

As is the case in the microphone domain, in the ambisonic domain source separation consists in estimating the contribution of every source to every channel $\mathbf{b}_{j,f,n}$, which can be solved using a Wiener filtering approach. To this aim we assume the following local Gaussian model:

$$\mathbf{b}_{j,f,n} \sim \mathcal{N}_c(0, \mathbf{\Sigma}_{\mathbf{b}_{j,f,n}}). \tag{8}$$

Similar to the microphone domain, the covariance $\mathbf{\Sigma}_{\mathbf{b}_{j,f,n}}$ can be further decomposed into a spectral part, $v_{j,f,n}$, and a spatial covariance matrix given by:

$$\mathbf{R}_{\mathbf{b}_{j,f}} = \mathbf{E}_f\,\mathbf{R}_{\mathbf{c}_{j,f}}\,\mathbf{E}_f^\mathsf{H}\,. \tag{9}$$

3. SOURCE SEPARATION WITH FASST

The multi-channel source separation problem can be solved by looking for the filter that minimizes the expected squared error for every source j and every time frequency bin (f, n):

$$\forall j \in [1, J], f \in [1, F] \text{ and } n \in [1, N],$$
$$\mathbf{W}_{j,f,n} = \underset{\mathbf{W}}{\operatorname{argmin}}\, \mathbb{E}\left[\|\mathbf{c}_{j,f,n} - \mathbf{W}_{j,f,n}\,\mathbf{x}_{f,n}\|_2^2\right]\,. \tag{10}$$

The filter $\mathbf{W}_{j,f,n}$ is known as the multichannel Wiener filter and is given by:

$$\mathbf{W}_{j,f,n} = \mathbf{\Sigma}_{(\mathbf{c}_{j,f,n},\,\mathbf{x}_{f,n})}\,\mathbf{\Sigma}^{-1}_{(\mathbf{x}_{f,n},\,\mathbf{x}_{f,n})}\,, \tag{11}$$

where the matrices $\mathbf{\Sigma}_{(\mathbf{x}_{f,n},\,\mathbf{x}_{f,n})}$ and $\mathbf{\Sigma}_{(\mathbf{c}_{j,f,n},\,\mathbf{x}_{f,n})}$, represent the covariance of the mixture $\mathbf{x}_{f,n}$ and the cross-correlation between the vectors $\mathbf{c}_{j,f,n}$ and $\mathbf{x}_{f,n}$, respectively.

From Eq. (4), and assuming the sources are statistically independent, the Wiener filter can be simplified as:

$$\mathbf{W}_{j,f,n} = \mathbf{\Sigma}_{\mathbf{c}_{j,f,n}}\left(\sum_{j'=1}^{J} \mathbf{\Sigma}_{\mathbf{c}_{j',f,n}}\right)^{-1}\,. \tag{12}$$

Thus, the source separation problem reduces to the problem of estimating the covariance matrices $\mathbf{\Sigma}_{\mathbf{c}_{j,f,n}}$ or, equivalently in the HOA domain, $\mathbf{\Sigma}_{\mathbf{b}_{j,f,n}}$. Each source contribution is obtained by applying element-wise its corresponding Wiener filter to the mixture: $\mathbf{c}_{j,fn} = \mathbf{W}_{j,fn}\mathbf{x}_{fn}$, respectively, $\mathbf{b}_{j,f,n} = \mathbf{W}_{j,f,n}\mathbf{z}_{j,f,n}$ in the HOA domain. and finally using overlap-add to reconstruct the time-domain signal.

In this work we use the flexible audio source separation toolbox (FASST) [13, 16], a software toolbox which allows the estimation of these parameters and apply the subsequent Wiener filter. In FASST the parameters are estimated by maximizing the log-likelihood of the observations with an Expectation-Maximization (EM) algorithm, and a multi-channel non negative matrix factorization (NMF) model can be enforced on the source covariances $\Sigma_{c_{j,f,n}}$ [11].

4. EXPERIMENTAL PROTOCOL

4.1. Dataset

In order to evaluate the source separation performance, we built a dataset as follows. First, fifty songs were picked randomly from the Mixing Secret Dataset (MSD100)[1]. In the MSD100 database, each song consists of four sound sources (voice, bass, drums and "others") provided as separate tracks.

In this work, microphone array recordings were then simulated using MCRoomSim [17], a room acoustics simulation software. This software calculates impulse responses modeling the acoustic propagation between acoustic sources and sensors in reverberant environments. A total of 16 simulations were run, corresponding to four rooms and four source configurations. The eight rooms had the same dimensions, 10 m $\times$ 8 m $\times$ 3 m, but different wall absorption coefficients, which resulted in the following reverberation times: 0 s, 0.2 s, 0.4 s and 0.7 s. The four source configurations are illustrated in Fig. 1. In every simulation the microphone array was modeled to match the characteristics of the Eigenmike[2] and was located at the same position in the room. In order to calculate the microphone mixtures, for each song and each of the 32 conditions the separate source tracks were then convolved with the simulated impulse responses and summed with each other.

We then built two different inputs for source separation: a microphone mixture $\mathbf{x}$ obtained using every sensor of the Eigenmike (32 channels), and a fourth order ambisonic mixture $\mathbf{b}$ (25 channels) obtained by applying encoding filters to the microphone mixture $\mathbf{x}$.

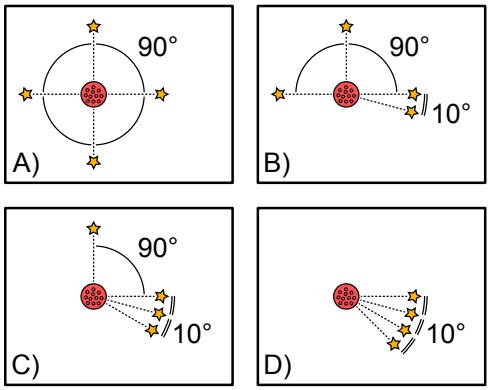


Fig. 1: The four sound source configurations considered in our simulations. Note: stars represent sound source locations. All the sources and the spherical microphone array are at the same height. Across all the examples, the position of sources with similar type was the same.

4.2. Evaluation criteria

In order to validate the adaptability of FASST in the HOA domain, we propose to compare its performance to that obtained by applying FASST in the microphone domain. A fair comparison requires to compute the chosen performance measures in the same domain. However, some information is lost when converting microphone signals to HOA signals, therefore it is impossible to convert the separated HOA signals back to the microphone domain for comparison. To alleviate this issue, instead of computing the evaluation measures in terms of the contribution of each source in each channel/microphone (FASST's outputs), we propose to compute them in terms of sound objects.

We obtain sound objects by applying beamforming to the signals separated by FASST. For simplicity, we use the beamforming technique known as the matched filter for both the microphone and HOA domains. For the j-th sound source, the sound object is calculated by projecting the separated signals onto the steering vector corresponding to a plane wave incoming from the source direction,(θ_j, ϕ_j). In other words, the estimated source object j in the microphone and HOA domains are calculated as follows:

$$\hat{s}^{\text{Mic}}_{j,f,n} = \frac{\mathbf{a}^H_{j,f}}{||\mathbf{a}_{j,f}||^2}\hat{\mathbf{c}}_{j,f,n} \quad (13)$$

$$\hat{s}^{\text{HOA}}_{j,f,n} = \frac{\mathbf{y}^H_{j,f}}{||\mathbf{y}_{j,f}||^2}\hat{\mathbf{b}}_{j,f,n} \quad (14)$$

where $\mathbf{y}_j$ corresponds to the spherical harmonic vector evaluated at the direction of arrival of the source j. We then compare the estimated sound object signals to reference signals, which we define as the sound objects obtained by applying the same beamforming to the actual (i.e. oracle) sound source contributions. In other words for the source j the reference signal in the microphone and HOA domains are given by:

$$s^{\text{Mic}}_{j,f,n} = \frac{\mathbf{a}^H_{j,f}}{||\mathbf{a}_{j,f}||^2}\mathbf{c}_{j,f,n} \quad (15)$$

$$s^{\text{HOA}}_{j,f,n} = \frac{\mathbf{y}^H_{j,f}}{||\mathbf{y}_{j,f}||^2}\mathbf{b}_{j,f,n} \quad (16)$$

Lastly, we assess the source separation performance by comparing the signals given by Eq. (13), and Eq. (14) to the ones given by Eq. (15) and Eq. (16), respectively, using the following performance measures [18]: Signal to Distortion Ratio (SDR), Signal to Artifact Ratio (SAR), and Signal to Interference Ratio (SIR). These measures are then calculated with the BSS-eval toolbox[3] [19].

4.3. Evaluated methods

The first method we examine consists in applying FASST to the HOA mixtures (see Sec. 3.) We compare it to the equivalent microphone-domain mixtures with the same number of channels. We also compare the performance obtained by using FASST in the HOA domain with that of two different beamformers.

The first beamformer has already been introduced in Section. 4.2. It is the matched filter beamformer(PWD), but this time applied directly to the HOA mixture, which is given by:

$$\bar{s}^{\text{HOA}}_{j,f,n} = \frac{\mathbf{y}^H_{j,f}}{||\mathbf{y}_{j,f}||^2}\mathbf{z}_{f,n}. \quad (17)$$

The second beam former, which we refer as the pseudo-inverse beamfomer, consists in multiplying the HOA signals with the pseudo-inverse of the matrix containing the steering vectors for the directions of the sources. The resulting beamformer is a particular

[1] https://sisec.inria.fr/sisec-2015/2015-professionally-produced-music-recordings/

[2] https://mhacoustics.com/products

[3] BSS-eval version 3.0 for Matlab, http://bass-db.gforge.inria.fr/bss_eval/

case of the Linearly-Constrained Minimum-Variance beamformer (LCMV). It is given by:

$$\hat{\bar{s}}^{\mathrm{HOA}}_{j,f,n} = (\mathbf{Y}^H\mathbf{Y})^{-1}\mathbf{Y}^H\mathbf{z}_{f,n}, \quad (18)$$

where the matrix $\mathbf{Y}$ contains the spherical harmonic vectors of the sources direction of arrivals.

4.4. FASST parametrization and initialization

The FASST toolbox requires choosing configuration parameters, as well as providing initial values for the covariance matrices $\mathbf{\Sigma}_{\mathbf{c}_{j,f,n}}$ in Eq. (12). Tab. 1 summarizes the parameters used for all experiments. Further, in order to match the scene configuration, the number of sources was fixed to 4 in the anechoic condition and 5 in reverberant conditions, where we observed that it was beneficial to add a source accounting for diffuse noise or late reverberation.

Transform type	STFT
Sampling frequency	44100 Hz
Window length	69 ms (3072 samples)
NMF rank	16
Stopping criterion	150 iterations

Table 1: FASST parameters

In FASST, covariances are decomposed into a spectral part and a spatial part, and the spectral part further modeled by NMF, as proposed by [16]. For each of the first 4 sources, the spatial covariance was initialized to the rank-1 matrices $\mathbf{R}_f^{\mathrm{HOA}} = \mathbf{y}_j\mathbf{y}_j^{\mathsf{H}}$ and $\mathbf{R}_f^{\mathrm{Mic}} = \mathbf{a}_{j,f}\mathbf{a}_{j,f}^{\mathsf{H}}$ for the HOA and microphone domain, respectively. In the microphone domain, the steering vectors $\mathbf{a}_{j,f}$ were derived from the microphone array characteristics and source locations. In the HOA domain, the steering vectors $\mathbf{y}_j$ were derived as the vector of the first nine spherical harmonic functions evaluated in the source directions. In the reverberant case, the fifth source was assumed to have a full-rank spatial covariance, which was was initialized to the identity matrix. Lastly, regarding the spectral part of the covariance, NMF factors were initialized as random numbers.

5. VALIDATION OF THE APPROACH

As explained before the main goal is to validate experimentally the local Gaussian model assumption for source separation in the HOA domain. To this aim the performances obtained using FASST in the HOA domain are compared to that obtained in the microphone domain. However, we first need to select a number of microphone channels and HOA signals that ensures a fair comparison between the two methods. Thus, we first investigate the influence of the number of microphone channels and HOA signals on the source separation performance for a fraction of the dataset. The results of this study, presented in Sec. 5.1, indicate that it is fair to compare the results obtained with 9 HOA signals (order 2 HOA signals) with that obtained using 9 microphone channels. In Sec. 5.2 we present further experimental results obtained using the selected number of channels for the entire dataset.

5.1. Selection of the number of microphones/channels

The computational cost of FASST depends primarily on the square of the number of channels of the mixture, and considering the size of our dataset, it is important to spare time and resources in the main experiment that will soon be described. A naive approach would be to adopt a lower HOA order $L < 4$, and consider on the one hand HOA mixtures with $M = (L+1)^2$ channels, and on the other one, the same mixtures given by a sub-antenna of the Eigenmike, where the number of the chosen capsules is $I = (L+1)^2$.

However, one could argue that while HOA mixtures are obtained by considering all of the capsules of the Eigenmike, the microphone mixtures are given by only $M = (L+1)^2$ selected microphones, and therefore, the comparison could be considered unfair. To clarify this point, we begin our experiments by measuring the source separation performance in both domains when varying respectively the number of channels and the number of microphones. This preliminary experiment is done on a small proportion of the created dataset (see below).

First, in the microphone domain we considered different sub antennas from the Eigenmike where the capsules were selected in order to be distributed regularly on the sphere. The considered numbers of microphones are $I = 4, 9, 12, 16, 25, 32$ (the numbers $4, 9, 16, 25$ were chosen to match the number of possible channels in the HOA domain, the number 12 is considered because the chosen capsules can be regularly distributed in the best way to cover the sphere). Second, HOA signals make sense if they are grouped by order L, each order L corresponding to a number of channels $M = (L+1)^2$. We have already at our disposal the 4th order signals (25 channels) by encoding the information provided by the 32 capsules of the Eigenmike. In order to have the first, the second, and the third order we have to simply truncate respectively the 25 HOA signals to the first $M = 4, 9, 16$ channels. Considering the selected capsules in the microphone domain and the truncation of the signals in the HOA domain, from our data set we considered randomly 160 mixtures, all the listed time reverberations and sound source configurations were considered.

We applied FASST to the different mixtures, considering the sources DOA known, the initialization and the parametrization of the toolbox are given in Sec. 4.4. The results in terms of SDR, SIR and SAR are given in Fig. 2.

In the microphone domain, we observe that the SIR tends to improve by 0.07 dB in average when increasing the number of microphones, the SAR tends to decrease, when it comes to the SDR we observe that it increases slightly by 0.02 dB in average until 9 microphones and drops after. In the HOA domain, we observe an improvement of all performance measures when increasing the number of channels. We can clearly see that adding more microphones doesn't improve the source separation performance in the microphone domain. As a conclusion it is unnecessary to add more microphones in the microphone domain, and therefore the comparison of FASST's performance between the HOA domain and the microphone domain is fair if the number of channels/microphones is equal to $I = M = 9$.

5.2. Extensive experiments with 9 microphones/channels

In the following the considered number of channels is equal to the considered number of microphones $I = M = 9$. In the microphone domain the selected capsules are given in Table. 2. More

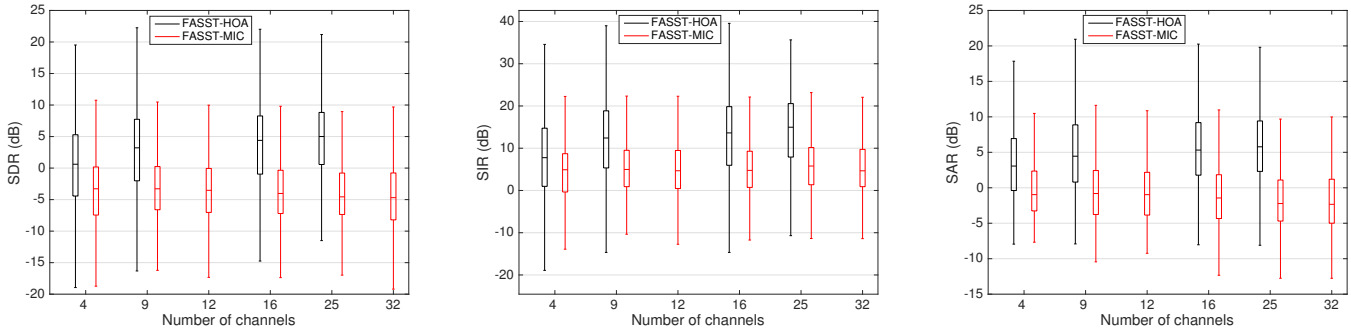


Fig. 2: Comparing FASST performance in regards of number the used microphones/channels.

information about the angular position of the Eigenmike's capsules can be found in [20].

	1	2	3	4	5	6	7	8	9
θ	0	35	-58	-31	0	-58	35	69	-32
ϕ	-32	45	0	90	212	180	135	269	-90

Table 2: Elevation (θ) and azimuth (ϕ), in degrees, of the selected Eigenmike microphone capsules. The radius of the microphone is 4 cm. The origin of space is the center of the Eigenmike.

In the following, we consider the whole dataset described in (Sec. 4.1). The results of the comparison are given in Fig. 3. As expected performance decrease as reverberation and scene complexity increase, regardless of the signal domain. However, in most configurations, separating the sources in the HOA domain resulted in better performance measures compared to the microphone domain. We can clearly see a gain of 7 to 12 dB for the least challenging sound source configuration, and a gain of 1 to 6 dB for the most challenging one. Tab. 3 summarizes the difference in SDR values between the HOA domain and the microphone domain for configurations (A) and (D): the SDR is almost always higher in the HOA domain, regardless of the reverberation or song. As well, the gap between the performance obtained in the two domains reduces as the complexity of the scenario increases, with a more prominent influence of reverberation time. Separating sources in the HOA domain results in a 1 to 12 dB increase in signal-to-distortion ratio, compared to the microphone domain.

	$RT_{60}(s)$	0	0.2	0.4	0.7
A	max	21	14.35	11.9	12
	median	12.43	7.69	7	6.84
	min	4.3	2.52	2.5	2.9
D	max	10.17	6.6	6.7	6.32
	median	6.05	0.83	1.52	2.45
	min	-1.84	-6	-5	-4

Table 3: $\Delta SDR = SDR_{HOA} - SDR_{MIC}$, in dB, for scenarios A and D.

One reason may explain these results. Indeed, in FASST's EM algorithm, the empirical covariance matrix is inverted while estimating the first Wiener filter [13] and the numerical stability of this inversion differ in the two signal domains. We calculated the condition number of the empirical covariance matrix in both domains for a random example picked from the dataset. It appeared that, for frequencies below 2 kHz, the condition number was generally higher in the microphone domain than in the HOA domain, and could be about 1000 times greater for some frequency values. Therefore, the conversion of the microphone signals into HOA signals seems to act as a pre-conditioning for the EM algorithm.

Having established the interest of performing the source separation in the HOA domain with FASST, we now compare it with the reference methods. Results are presented in Fig. 4. FASST clearly outperforms the reference methods. This is because, contrary to the reference methods which are solely based on spatial cues, FASST also exploits spectral cues. This gives FASST an advantage when sources are close to each other and spatial information is more ambiguous. Although this fact has already been observed in microphone domain source separation [21], we confirm it here also on HOA-domain source separation.

Surprisingly, FASST outperforms the PIV method even in anechoic environment where the PIV method could have been expected to give the best results in terms of performance. Indeed, 9 signals should be enough to form a beam toward one source and cancel 3 interfering sources at the same time. This can be explained with the fact that encoded HOA signals don't match perfectly the theoretical signals. This imperfection is mainly caused by the physical limitations of the microphone array. Indeed, the capsules of the Eigenmike are relatively close to each other, which results in spatial aliasing and a loss of lower frequencies [22].

6. CONCLUSION

In this paper we investigated for the first time the ability of the local Gaussian model to handle the source separation problem in the HOA domain. To this aim we have established the model's equations in the HOA domain and run numerical experiments. Our simulation results show that applying a local Gaussian model-based source separation method in the HOA domain typically results in the SDR increasing by 1 to 12 dB, compared to the microphone domain with the same number of microphones/channels $I = M = 9$, including in challenging situations such as reverberant environments and complex source configurations. In future work we will explore using proximity microphones in order to guide the source separation and improve its performance, and finally employ this method to allow navigation through HOA sound scenes.

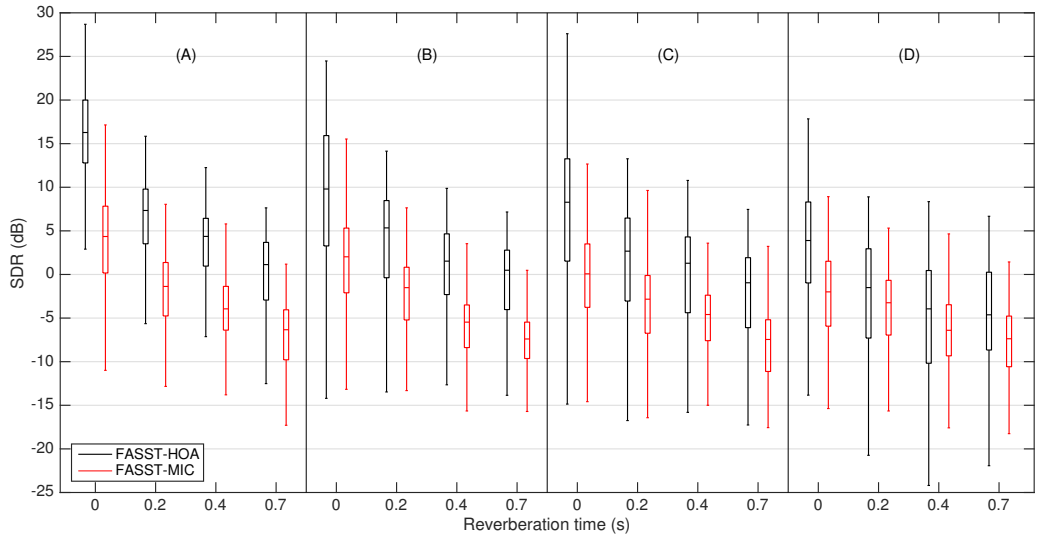


Fig. 3: Comparing FASST's performance in the HOA domain to FASST's performance in the microphone domain, $I = M = 9$

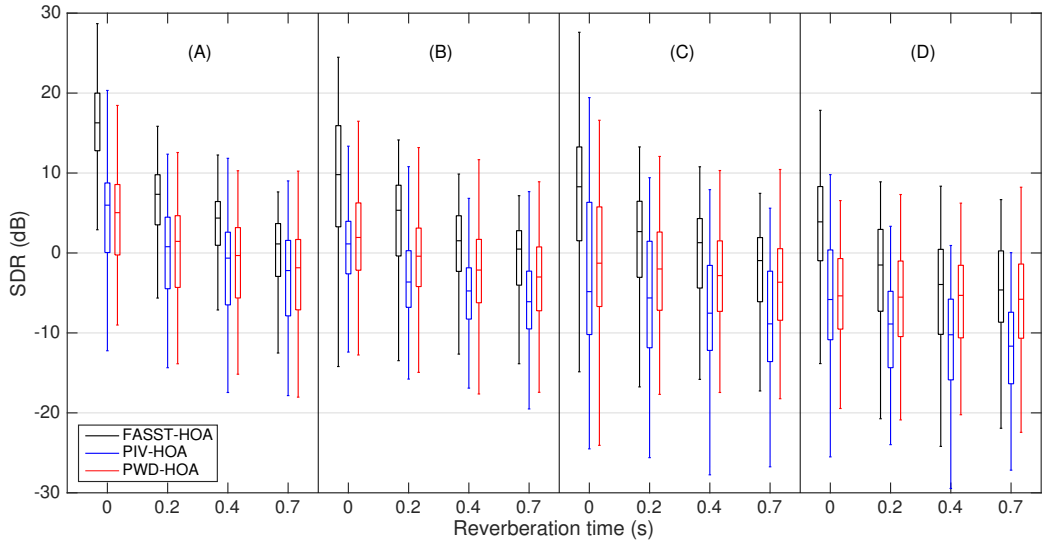


Fig. 4: Comparing FASST to the reference methods in the HOA domain.

7. REFERENCES

[1] Michael A Gerzon, "Periphony: With-height sound reproduction," *Journal of the Audio Engineering Society*, vol. 21, no. 1, pp. 2–10, 1973.

[2] Michael A Gerzon, "Ambisonics in multichannel broadcasting and video," *Journal of the Audio Engineering Society*, vol. 33, no. 11, pp. 859–871, 1985.

[3] Jérôme Daniel, Jean-Bernard Rault, and Jean-Dominique Polack, "Ambisonics encoding of other audio formats for multiple listening conditions," in *Audio Engineering Society Convention 105*. Audio Egineering Society, 1998.

[4] Jérôme Daniel, Sebastien Moreau, and Rozenn Nicol, "Further investigations of High-Order Ambisonics and wavefield synthesis for holophonic sound imaging," in *Audio Engineering Society Convention 114*. Audio Engineering Society, 2003.

[5] Alex Southern, Jeremy Wells, and Damian Murphy, "Rendering walk-through auralisations using wave-based acoustical models," in *Signal Processing Conference, 2009 17th European*. IEEE, 2009, pp. 715–719.

[6] Joseph G Tylka and Edgar Choueiri, "Soundfield navigation using an array of Higher-Order Ambisonics microphones," in *Audio Engineering Society conference: 2016 AES Interna-*

tional Conference on Audio for Virtual and Augmented Reality. Audio Engineering Society, 2016.

[7] Matthias Kronlachner and Franz Zotter, "Spatial transformations for the enhancement of ambisonic recordings," in *International Conference on Spatial Audio*, 2014.

[8] Andrew Allen and W. Bastiaan Kleijn, "Ambisonic soundfield navigation using directional decomposition and path distance estimation," in *International Conference on Spatial Audio*. Graz, 2017.

[9] Mohammed Hafsati, Nicolas Epain, and Jérôme Daniel, "Editing ambisonic sound scenes," in *International Conference on Spatial Audio*. Graz, 2017.

[10] Ngoc QK Duong, Emmanuel Vincent, and Rémi Gribonval, "Under-determined reverberant audio source separation using a full-rank spatial covariance model," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 7, pp. 1830–1840, 2010.

[11] Alexey Ozerov and Cédric Févotte, "Multichannel Nonnegative Matrix Factorization in convolutive mixtures for audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 3, pp. 550–563, 2010.

[12] Emmanuel Vincent, Shoko Araki, and Pau Bofill, "The 2008 Signal separation evaluation campaign: A community-based approach to large-scale evaluation," in *International Conference on Independent Component Analysis and Signal Separation*. Springer, 2009, pp. 734–741.

[13] Yann Salaün, Emmanuel Vincent, Nancy Bertin, Nathan Souviraa-Labastie, Xabier Jaureguiberry, Dung T Tran, and Frédéric Bimbot, "The Flexible Audio Source Separation Toolbox Version 2.0," in *ICASSP*, 2014.

[14] Emmanuel Vincent, Simon Arberet, and Rémi Gribonval, "Underdetermined instantaneous audio source separation via Local Gaussian Modeling," in *International Conference on Independent Component Analysis and Signal Separation*. Springer, 2009, pp. 775–782.

[15] Rozenn Nicol, "Sound spatialization by Higher Order Ambisonics: Encoding and decoding a sound scene in practice from a theoretical point of view," in *International Symposium on Ambisonics and Spherical Acoustics*, 2010.

[16] Alexey Ozerov, Emmanuel Vincent, and Frédéric Bimbot, "A general flexible framework for the handling of prior information in audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 4, pp. 1118–1133, 2012.

[17] Andrew Wabnitz, Nicolas Epain, Craig Jin, and André Van Schaik, "Room acoustics simulation for multichannel microphone arrays," in *Proceedings of the International Symposium on Room Acoustics*. Citeseer, 2010, pp. 1–6.

[18] Emmanuel Vincent, Rémi Gribonval, and Cédric Févotte, "Performance measurement in blind audio source separation," *IEEE transactions on audio, speech, and language processing*, vol. 14, no. 4, pp. 1462–1469, 2006.

[19] Cédric Févotte, Rémi Gribonval, and Emmanuel Vincent, "BSS_EVAL toolbox user guide–revision 2.0," 2005.

[20] MH Acoustics, "EM32 Eigenmike microphone array release notes (v17. 0)," *25 Summit Ave, Summit, NJ 07901, USA*, 2013.

[21] Emmanuel Vincent, *Contributions To Audio Source Separation And Content Description*, Ph.D. thesis, Université Rennes 1, 2012.

[22] Sébastien Moreau, Jérôme Daniel, and Stéphanie Bertet, "3D sound field recording with higher order ambisonics–Objective measurements and validation of a 4th order spherical microphone," in *120th Convention of the AES*, 2006, pp. 20–23.

DIRECTIONAL SOURCE SIMULATION IN FDTD ROOM ACOUSTIC MODELING VIA WEIGHTED FIRST-ORDER REFLECTIONS

Stephen Oxnard

Meridian Audio Ltd.
Huntingdon, UK
stephen.oxnard@meridian.co.uk

ABSTRACT

This paper proposes a means of simulating directional sources in Finite Difference Time Domain (FDTD) based acoustic models for impulse response (IR) calculation which, unlike other currently available methods, is able to accommodate highly irregular directivity patterns. Weighted numerical free-field IRs are applied to the simulation of first-order reflections at the boundaries of a modeled acoustic domain such that the continuing interior wave field is that which would be produced by a directional sound source. Numerical results demonstrate the application of this approach for a 2-Dimensional wave field, implementing both simple and irregular (e.g. highly directional, discontinuous) sound source directivity patterns.

1. INTRODUCTION

Directional sound source modeling is of importance for realistic wave-based room acoustics simulations. Previous work in this area has largely focused on emulating source directivity functions via weighted monopole excitations [1, 2, 3] and spherical harmonic composition [4, 5]. These strategies are capable of simulating analytical directivity functions but are not well-suited to emulating realistic source directivities that include large fluctuations in magnitude over small angles. Some of them (e.g. [3, 4]) provide only first-order spherical harmonic directivity patterns. A method for frequency-dependent directional source modeling based on empirical directivity functions is presented in [6] by which it is shown possible to model smooth variances in source directivity in the far-field. Recent work [7, 8] provides a means of FDTD-based directional sound source modeling that is capable of incorporating source movement and rotation for multipole source directivities. An alternative approach is presented in [9] and is based on the optimization of initial values within the target acoustic field represented by a FDTD scheme. While the strategies of [6, 7, 8, 9] have been shown as capable of representing source directivity patterns that vary slowly with angle (i.e. well-behaved and continuous), it has yet to be demonstrated that such approaches are suitable for representing more challenging patterns such as those which vary rapidly with small changes in angle.

This paper presents a directional source modeling strategy for FDTD acoustics simulations which attains the primary goal of representing highly directional/complex source directivity patterns for static source IR capture. To the author's knowledge, this outcome is unattainable using existing techniques. It is intended that this strategy be extended to 3D and exploited for numerical room impulse response prediction with representation of realistic sound sources.

In brief, this approach begins by pre-calculating the free-field response to a monopole excitation at all boundary locations defined in the target acoustic field. As such, the outgoing direct sound component from a given monopole sound source location is captured in its entirety for each discrete boundary node and its neighbouring nodes in a given FDTD scheme. During simulation of the target acoustic field, pre-calculated responses are weighted and summed into the field at corresponding boundary nodes during run time to cancel a proportion of the outgoing direct sound component produced by a monopole sound source. In doing so, the amplitude of the subsequent first-order reflection may be weighted around the azimuth with respect to the sound source location and directivity pattern. The result is a first-order reflection and subsequent wave field that would be produced in response to a directional impulsive sound source. The following contribution details the formulation of this approach and continues to verify its correctness via a series of test cases.

2. SCHEME FORMULATION

2.1. The FDTD Model

For the purposes of simple and rapid exposition of this directional source model, this derivation is limited to the so-called "leap-frog" explicit FDTD iterative wave equation solution (see e.g. [10]) for a 2D target acoustic field. This solution solves the 2nd-order wave equation, $\frac{\partial^2 p}{\partial t^2} = c^2 \nabla^2 p$, for a scalar acoustic pressure field, p, and sound propagation speed, c, using the iterative update equation:

$$p_{l,m}^{n+1} = \lambda^2 S_{l,m}^n + (2 - 4\lambda^2) p_{l,m}^n - p_{l,m}^{n-1} \tag{1}$$

where $\lambda = \frac{cT}{h}$ for discrete time step, $T = 1/F_s$, spatial sampling instance h and temporal sampling rate Fs. The location of a pressure sample, or 'node', $p_{l,m}^n$, in time and space is given by the integer indexes n, for time nT, and (l, m) for the Cartesian x- and y-coords (lh, mh) in 2D space. Finally, the term $S_{l,m}^n$ refers to the sum of pressure values neighbouring a given node at current time n on a rectilinear grid of nodes defined over an enclosed domain:

$$S_{l,m}^n = p_{l+1,m}^n + p_{l-1,m}^n + p_{l,m+1}^n + p_{l,m-1}^n \tag{2}$$

Following [11], locally reacting surface (LRS) boundary conditions (after [12]) may be applied in such a model at terminating edges and corners of the spatial domain. This facilitates the modeling of absorbing surface characteristics within the acoustic simulation. The so-called "velocity-centered" [13] formulation of the

LRS condition may be written as follows for a right-hand boundary with frequency independent surface impedance ζ:

$$p_{l,m}^{n+1} = \alpha_E \Big(\lambda^2 (p_{l-1,m}^n + p_{l,m+1}^n + p_{l,m-1}^n) + (2-3\lambda^2) p_{l,m}^n - \beta_E p_{l,m}^{n-1} \Big) \qquad (3)$$
$$\alpha_E = \frac{1}{1+\frac{\lambda}{2\zeta}}, \quad \beta_E = 1 - \frac{\lambda}{2\zeta}$$

For a top-right-hand corner boundary node, the LRS condition is given as:

$$p_{l,m}^{n+1} = \alpha_C \Big(\lambda^2 (p_{l-1,m}^n + p_{l,m-1}^n) + (2-3\lambda^2) p_{l,m}^n - \beta_C p_{l,m}^{n-1} \Big); \quad \alpha_C = \frac{1}{1+\frac{\lambda}{\zeta}}, \quad \beta_C = 1 - \frac{\lambda}{\zeta} \qquad (4)$$

Similar expressions may be derived for all other boundary orientations aligned with the rectilinear grid as per [14]. The LRS boundary conditions together with (1) complete the FDTD model used for this study.

2.2. Derivation of Directional Sound Source Solution

The premise of this work is that it is possible to manipulate the first-order reflection exiting a boundary node in response to an omnidirectional impulsive excitation using pre-computed free-field responses such that the reflected wave front, and subsequent wave field, represents that produced by a directional sound source. The required free-field responses are those which preserve continuity of the wave equation at the location of each boundary node for all values of impedance in the range $\zeta = [1, \infty]$. These responses are denoted $E_{l,m}^n$ for edges and $C_{l,m}^n$ for corners. The treatment of right-hand boundaries is the focus of the following derivation, noting that similar free-field response expressions may be reached by the same means for all remaining boundary orientations.

The derivation of $E_{l,m}^n$ is presented here for the case of a right-hand edge boundary with the governing equation (3). Comparing (3) to the discrete wave equation update (1) it is evident the function of responses required to preserve wave motion at the edge boundary, denoted P_E, is:

$$P_E = \Big(1-\alpha_E\Big)\Big(\lambda^2 (p_{l-1,m}^n + p_{l,m+1}^n + p_{l,m-1}^n) + (2-3\lambda^2) p_{l,m}^n - 2p_{l,m}^{n-1}\Big) + \lambda^2 (p_{l+1,m}^n - p_{l,m}^n) \qquad (5)$$

and that by adding P_E to the right hand side of (3), the wave equation results and anechoic conditions are realized at the boundary. However, such a result cannot be computed using responses of the enclosed pressure field p during simulation as these responses would comprise both direct and reflected wave fronts as opposed to the free-field direct sound components as required. Furthermore, the term $p_{l+1,m}^n$ refers to a response that lies outside the simulated domain and is undefined in the numerical solution of the pressure field p as prescribed by equations (1)-(4). Hence, it is proposed that all pressure signals contributing to P_E be computed in the free-field in advance of simulating the problem domain p. Denoting the free-field as $\tilde{p}$, the free-field equivalent of P_E is then the desired signal $E_{l,m}^n$:

$$E_{l,m}^n = \Big(1-\alpha_E\Big)\Big(\lambda^2 (\tilde{p}_{l-1,m}^n + \tilde{p}_{l,m+1}^n + \tilde{p}_{l,m-1}^n) + (2-3\lambda^2) \tilde{p}_{l,m}^n - 2\tilde{p}_{l,m}^{n-1}\Big) + \lambda^2 (\tilde{p}_{l+1,m}^n - \tilde{p}_{l,m}^n) \qquad (6)$$

for a right-hand edge boundary node at location (lh, mh) and time nT.

A similar procedure is followed for the case of a corner boundary node. In this instance, using a top-right-hand boundary node (4), the function required to preserve continuity of the wave equation at the boundary node, P_C, is:

$$P_C = \Big(1-\alpha_C\Big)\Big(\lambda^2 (p_{l-1,m}^n + p_{l,m-1}^n) + (2-2\lambda^2) p_{l,m}^n - 2p_{l,m}^{n-1}\Big) + \lambda^2 (p_{l+1,m}^n + p_{l,m+1}^n - 2p_{l,m}^n) \qquad (7)$$

Again, the required free-field equivalent pressure signals comprising P_C are pre-computed to produce the desired signal $C_{l,m}^n$:

$$C_{l,m}^n = \Big(1-\alpha_C\Big)\Big(\lambda^2 (\tilde{p}_{l-1,m}^n + \tilde{p}_{l,m-1}^n) + (2-2\lambda^2) \tilde{p}_{l,m}^n - 2\tilde{p}_{l,m}^{n-1}\Big) + \lambda^2 (\tilde{p}_{l+1,m}^n + \tilde{p}_{l,m+1}^n - 2\tilde{p}_{l,m}^n) \qquad (8)$$

Having pre-computed the required signals $E_{l,m}^n$ and $C_{l,m}^n$ in the free-field for all edge and corner boundary nodes that facilitate a first-order reflection in the target acoustic field, they may be weighted based on an arbitrary source directivity function, $S(\theta)$, for $\theta \in [0, 2\pi]$ and $S(\theta) \in [-1, 1]$. In order to achieve variable cancellation of the initial sound wave incident on the boundary, appropriate weighting coefficients for $E_{l,m}^n$ and $C_{l,m}^n$ are defined as $A(\theta) = 1 - S(\theta)$. Hence, the final update expression for a right-hand edge boundary node of the target domain p during simulation is:

$$p_{l,m}^{n+1} = \alpha_E \Big(\lambda^2 (p_{l-1,m}^n + p_{l,m+1}^n + p_{l,m-1}^n) + (2-3\lambda^2) p_{l,m}^n - \beta_E p_{l,m}^{n-1} \Big) + A(\theta) E_{l,m}^n \qquad (9)$$
$$\alpha_E = \frac{1}{1+\frac{\lambda}{2\zeta}}, \quad \beta_E = 1 - \frac{\lambda}{2\zeta}$$

where θ is taken to be the angle between the look direction of the sound source and the location of the respective boundary node w.r.t the source location. For a top-right-hand corner boundary node, the following update expression is applied during simulation:

$$p_{l,m}^{n+1} = \alpha_C \Big(\lambda^2 (p_{l-1,m}^n + p_{l,m-1}^n) + (2-3\lambda^2) p_{l,m}^n - \beta_C p_{l,m}^{n-1} \Big) + A(\theta) C_{l,m}^n \qquad (10)$$
$$\alpha_C = \frac{1}{1+\frac{\lambda}{\zeta}}, \quad \beta_C = 1 - \frac{\lambda}{\zeta}$$

The same approach is applied to edge and corner boundary nodes of all other possible orientations.

Examples of the application of the weighted free-field responses include full reflection and anechoic conditions. Full reflection results when $S(\theta) = 1$ and $A(\theta) = 0$ such that the functions $E_{l,m}^n$ and $C_{l,m}^n$ do not contribute to the response at the associated boundary node and, therefore, do not cancel or reduce the reflection. Conversely, for $S(\theta) = 0$ and $A(\theta) = 1$, full cancellation is required and hence the maximum contribution of the functions $E_{l,m}^n$ and $C_{l,m}^n$ are applied to associated boundary nodes thus nullifying the first-order reflection from these nodes yielding anechoic conditions in the subsequent simulated wave field.

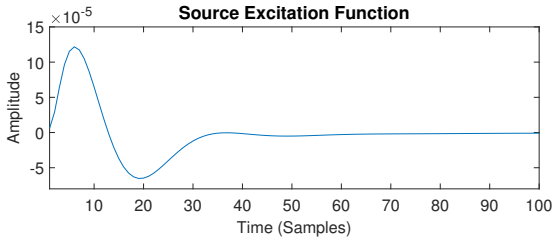


Figure 1: The DC-blocked and low pass filtered Kronecker Delta input excitation function with afterglow removal applied, after [15]. This source function is implemented as a soft-source in all simulations of the target acoustic domain documented in this paper.

2.3. Directional Sound Source IR Simulation

Using the solution described, the initial wavefront incident on each boundary node may be fully or partially cancelled such that a directional source response is imposed on the target acoustic field for all time after the first-order reflections. It is important to note that this approach is valid, in the form described here, for concave domain geometries (see section 5 for more discussion).

Implementing the directional sound source requires functions $E^n_{l,m}$ and $C^n_{l,m}$ to be pre-computed for every boundary node that facilitates a first-order reflection as a consequence of the initial excitation. This pre-computation is conducted using a large leap-frog FDTD scheme that allows capture of node responses for the passage of the direct sound component only. Note that such free-field responses need be only recorded until they reach a negligible amplitude (e.g. -60 dB relative to the peak amplitude value). This condition affects the size of the free-field scheme as it must be large enough to render the free-field IR at all boundary node locations in absence of any unwanted reflections which may arise from domain terminations. Next, the recorded functions $E^n_{l,m}$ and $C^n_{l,m}$ are weighted by $A(\theta)$, with reference to the source location, and applied during simulation as per equations (9) and (10). Note that the free-field responses preserve time alignment with the simulated pressure field p. The resulting IR/s may then be captured as required from any point within the modeled space. These IRs contain the characteristics of the source directivity function $S(\theta)$ for all time after the onset of the first reflection.

A final post-processing step is needed in order to produce an appropriately weighted direct sound component in the resulting IR/s. This is conducted by obtaining the free-field responses at receiver locations during the pre-computation process. These free-field direct sound components may be weighted by $A(\theta)$, where θ is the angle between the receiver locations and the source look direction, and subtracted from the recorded IRs. This completes the directional sound source model for IR capture in FDTD acoustic simulations.

3. EXPERIMENTAL PROCESS

3.1. The FDTD Simulation

The simulated 2D FDTD scheme incorporates an inter-nodal sampling index defined at the Courant limit, yielding the smallest spatial sampling index that preserves numerical stability, $h = \sqrt{2}cT$, for a given $T = \frac{1}{F_s}$ in accordance with von Neumann stability analysis as applied to (1)[16]. For the purposes of this work, a sampling rate of $F_s = 48$ kHz and wave speed $c = 344$ ms^{-1} is applied giving $h = 0.0101$ m. This provides a usable simulation output bandwidth of 0.1 F_s = 4.8 kHz providing a maximum deviation of relative wave phase velocity of ≤ 2 % [17, 18]. The modeled domain is of side length 101 nodes with a central source location given by $(l, m) =$ (51,51) and velocity-centered LRS boundary conditions applied to terminating edges and corners.

The input source excitation is implemented using a soft-source [19] pre-filtered Kronecker delta function. The Kronecker delta function is pre-treated with a DC-blocking filter and a low-pass, 3rd-order Butterworth filter with cut-off frequency f_c = 0.075 F_s to minimize the impact of numerical dispersion effects. Finally, the resulting impulse is treated using afterglow removal to provide a more transient excitation free from afterglow that is intrinsic to the Green's function for the 2D wave equation [15]. Figure 1 depicts the first 100 samples of the resulting input source function. As shown, the filtering applied to the Kronecker delta function produces a function that is smeared in time and maintains a low negative amplitude for time $n \simeq$ [70:100]. These characteristics are important when interpreting results presented in Section 4.

3.2. Implementation of the Directional Sound Source

Having rendered functions $E^n_{l,m}$ and $C^n_{l,m}$ for every corresponding boundary node, they are then weighted appropriately according to $A(\theta) = 1 - S(\theta)$ and summed with the response of the corresponding boundary node during run-time. As the excitation of the FDTD simulation is a pre-filtered Kronecker delta function applied to one central source node, the initial wave propagation will be that of a free-field monopole. The impact of summing $E^n_{l,m}$ and $C^n_{l,m}$ into the corresponding boundary node's response is that the first-order reflection, on exit from the boundary, is weighted as required over the azimuth. A series of five test cases are defined to analyse particular simulation scenarios.

4. VALIDATION OF THE DIRECTIONAL SOUND SOURCE

Due to the fact that this approach to directional sound source simulation relies on weighted first-order reflections within the simulated target acoustic field, it is not possible to take a commonly employed method of analysis such as the measurement of the direct sound around the azimuth using a circular crux of receiver locations (see e.g. [3], [6]). Furthermore, measurement of the reflected wavefront amplitudes around the azimuth immediately after the first-order reflection does not provide a sensible representation of the expected directivity pattern. The reason for this can be explained using Huygen's Principle: an outgoing reflection from a single boundary node will proceed as a half-space monopole and therefore spread and sum with the reflections from adjacent boundary nodes thus producing the reflected wavefront in its totality. As a result, the directivity function will become immediately obscured within the target acoustic domain as directionally weighted first-order reflections spread and sum together. Hence, a series of indirect approaches to validating the proposed directional source method are examined and discussed in the following. All test cases are performed with boundary impedance ζ = 39 giving reflection coefficient R = 0.95 in order to ensure correct operation in conjunction with LRS boundary conditions.

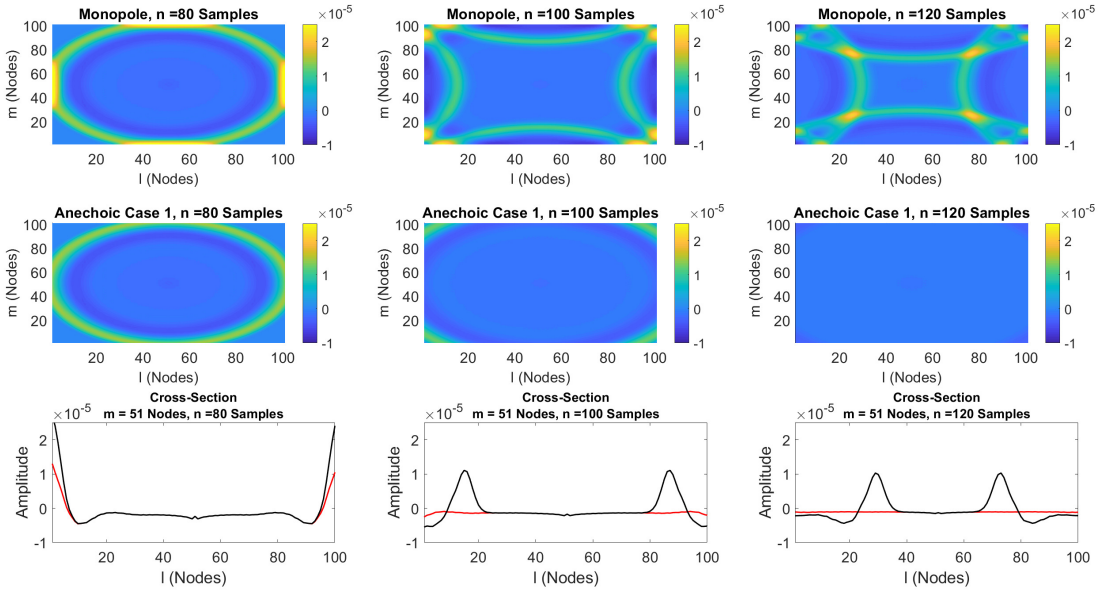


Figure 2: Results of CASE 1. Top row: Monopole sound source excitation with no directional weighting. Middle row: Monopole sound source excitation with full cancellation via free field response functions $E^n_{l,m}$ and $C^n_{l,m}$ applied at all boundary nodes. Bottom row: Cross-sectional amplitude values for the monopole (black) and anechoic (red) simulations. Time sample and node indexes are given in the diagram and all amplitude axes are equal.

CASE 1: Producing Anechoic Boundaries via Full Cancellation of First-Order Reflection

This first test case seeks to verify that the summation of response functions $E^n_{l,m}$ and $C^n_{l,m}$ weighted by $A(\theta) = 1$ for all angles around the azimuth with the simulated target acoustic field p produces anechoic conditions after the passage of the direct sound component. The condition $A(\theta) = 1$ corresponds to a directivity function $S(\theta) = 0$, i.e. the sound source is nullified in all directions due to full cancellation of the first-order reflection at the boundary nodes by $E^n_{l,m}$ and $C^n_{l,m}$. Figure 2 shows the result of implementing this directivity condition in the simulated target acoustic field (middle row) alongside the same target acoustic field excited by the input function only resulting in a monopole response (top row). Time sample instances of 80, 100 and 120 samples are selected to view the progression of the wavefronts throughout the simulated domain. It is clear from the monopole response that strong reflections are present within the domain as time increases. Conversely, in the case where the directivity function is applied, the first-reflection is completely cancelled yielding anechoic conditions as expected. This result is explored further by examining the cross-sectional amplitude values from both simulations across the middle of the domain (l = [1:100], m = 51). It is clear that reflections are preserved in the monopole response simulation (black) and that the same reflections are completely cancelled in the anechoic case (red). The small deviation from zero amplitude observed in the anechoic case cross-section is due to the remaining outgoing samples from the monopole sound source which maintains non-zero amplitude values up to and beyond $n \simeq 100$ samples. This crucial result demonstrates that the proposed directional source strategy may be employed to vary the strength of first-order reflections on a node-by-node basis hence facilitating highly directional source characteristics.

CASE 2: Summing Individual Boundary Node Reflections to Produce a Monopole Response

In order to further demonstrate the ability of the proposed approach to emulate directional sound source characteristics on a node-by-node basis over the boundaries of a modelled domain, the following test case simulates the result of applying $E^n_{l,m}$ to each edge node and $C^n_{l,m}$ to each corner node on an individual basis. As such, each individual response emulates a source directivity function for which $S(\theta) = 1$ ($A(\theta) = 0$) in the direction of a single boundary node and $S(\theta) = 0$ ($A(\theta) = 1$) otherwise. Figure 3 shows two examples of such cases. As shown, the outgoing monopole wavefront progresses to the boundary and is cancelled by all nodes but one located at $[l, m] = [1, 51]$ in one simulation (top row) and $[l, m] = [100, 51]$ in another (middle row). Examining both cases, it is demonstrated that a single half-space monopole reflection spreads from the individual boundary nodes in each simulation. The low amplitude of the reflections is as expected when all other boundary nodes are set to the anechoic condition and thus collectively dissipate a significant amount of energy from the initial source excitation. Referring to the bottom row of Figure 3, the single node reflection from the left boundary is displayed by the cross-section amplitude plots (depicted in black). The reflection passes back into the domain and sums with the continuing wavefront produced by the source. Finally, the amplitude of the reflection from the right boundary (cross-section, depicted in red)

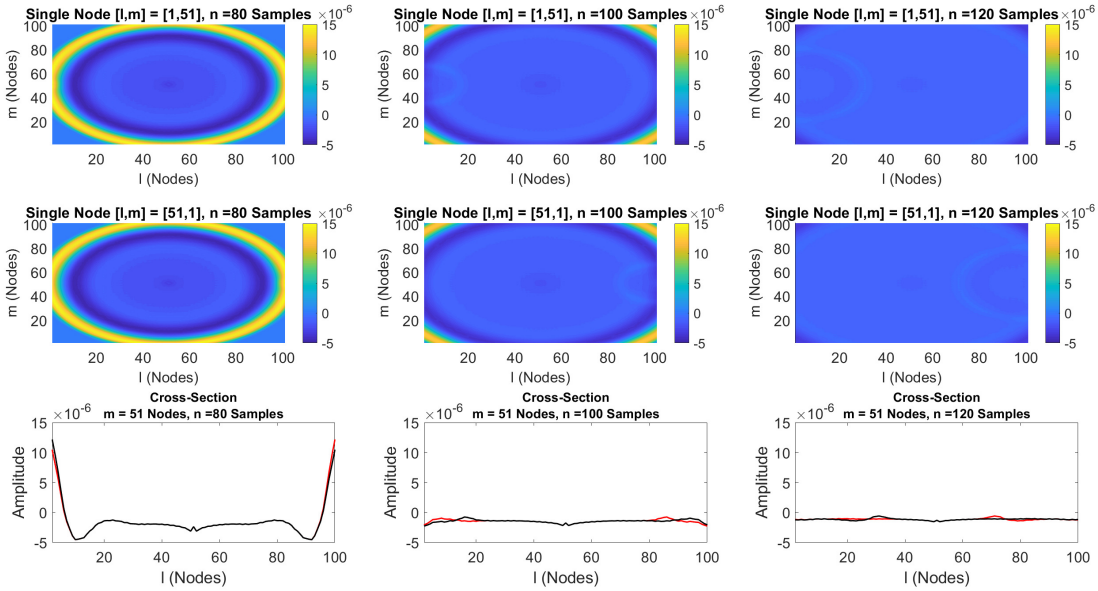


Figure 3: Results of CASE 2. Top row: Individual node reflection at boundary location l, m = [1,51]. Middle row: Individual node reflection at boundary location l, m = [100,51]. In both cases, the free-field response function $E^n_{l,m}$ is applied with $S(\theta) = 1$ ($A(\theta) = 0$). Bottom row: Cross-sectional amplitude values for the top row (black) and middle row (red) simulations. Time sample values and nodes are given in the diagram and all amplitude axes are equal.

is seen to be symmetrical to that of the left boundary reflection w.r.t. the centre of the modeled domain. To proceed, the wave field produced by each individual node reflection is simulated and the result is recorded for all nodes at time n = 120 samples. Recall that each wave field consists of the superposition of the individual node reflection and the continuing source excitation. As such, when all individual wave fields are summed, the source excitation is summed in once for every boundary node. This can be accounted for by subtracting $N - 1$ copies of the anechoic response (see Figure 2) at time n = 120 samples where N is the number of contributing boundary nodes. Note that this subtraction is only required for the purposes of demonstrating this test case. It may be stated that the sum of N individual contributions equates to a monopole when $N - 1$ anechoic responses are subtracted from the result. This outcome is verified by subtracting the summed responses from the monopole response generated in Case 1 at time sample n = 120 (see Figure 4 below).

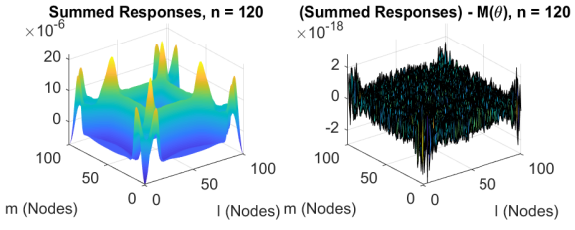


Figure 4: Summed individual boundary node reflections (left) and the difference (right) between the summed responses and the monopole $M(\theta)$. Note, axis scales are not consistent.

As shown in Figure 4, the subtraction of the wave field produced by the treated sum of responses from that of the monopole results in a zero pressure field (the cumulative noise floor of the contributing wave fields). This test case result demonstrates that the superposition of individual node reflections produces a monopole directivity function. Hence, the approach has been shown to be correct and suitable for producing directivity patterns that are highly directional (for example the arc length of one node relative to the sound source location).

CASE 3: Summing Opposing Semi-Circular Directivity Pattern Responses to Produce a Monopole Response

This test case is devised to demonstrate that the directional sound source implementation is capable of reproducing target directivity functions that incorporate discontinuities. In brief, a left-going semi-circular source directivity function is modeled by setting $S(\theta)$ = 1 for all boundary nodes with $0 < l < 51$ and applying $E^n_{l,m}$ and $C^n_{l,m}$ as prescribed in section 2. A further semi-circular source directivity is defined as right-going when $S(\theta)$ = 1 for all boundary nodes with $51 < l \leq 100$. Left- and right-going directivities are denoted $S_L(\theta)$ and $S_R(\theta)$ respectively. The corresponding simulated wave fields are shown in Figure 5 for time sample n = 120. As depicted, the first-order reflected wavefronts for $S_L(\theta)$ and $S_R(\theta)$ are nullified from the right and left boundaries respectively. In addition, it is clear that the reflected wavefronts in the region of l = 51 exhibit diffraction around the discontinuity imposed by the target directivity function. This is to be expected from preserving the continuity of the target pressure field. As a means of verifying the correctness of the simulated wave fields in response to the two tar-

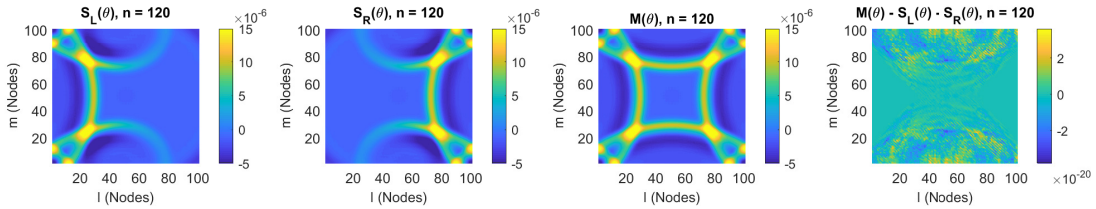


Figure 5: Results of CASE 3. (From left to right) The simulated wave fields produced by weighting boundary first-order reflections by $S_L(\theta)$, $S_R(\theta)$, $M(\theta)$ and $M(\theta) - (S_L(\theta) + S_R(\theta))$ at time sample n = 120. Note, the amplitude axis of the right-most panel is of order 10^{-20}

.

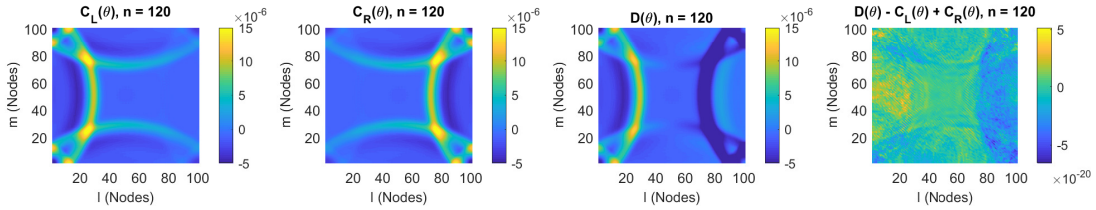


Figure 6: Results of CASE 4. (From left to right) The simulated wave fields produced by weighting boundary first-order reflections by $C_L(\theta)$, $C_R(\theta)$, $D(\theta)$ and the result of combining these responses to generate a zero pressure wave field. Note, the amplitude axis of the right-most panel is of order 10^{-20}

.

get source directivities, their sum is subtracted from a response to a monopole $M(\theta)$. As with CASE 2, it is necessary to account for the continuing sound source present in both semi-circular source simulations. The addition of the two wave fields doubles the contribution of the source. As such, a single anechoic response is subtracted from the sum of the two directional sound source simulations, noting again that this additional process is only necessary for the purposes of verification. The difference between the resulting wave field and the wave field in response to $M(\theta)$ is a zero pressure field (the noise floor of the simulations) as shown in the right most panel of Figure 5. This completes the empirical proof that the directional sound source approach is capable of emulating complex directivity functions.

CASE 4: Summing Opposing Cardioid and Dipole Directivity Responses to Produce a Zero Pressure Field

Up to this point, test case directivities have incorporated only positive values for $S(\theta)$. In this test case, it is demonstrated that the directional sound source implementation is suitable for negative directivity function values. This is achieved by simulating a left-going cardioid response $C_L(\theta) = 0.5 + 0.5\cos(\theta)$, a right-going cardioid response $C_R(\theta) = 0.5 + 0.5\cos(\theta + \pi)$ and a circular bi-directional response $D(\theta) = \cos(\theta)$. Figure 6 displays the resulting wave fields rendered at time sample n = 120. It is straightforward to show that the combination $D_L(\theta) - C_L(\theta) + C_R(\theta)$ = 0. Following on from this, it is assumed that if the rendered responses are combined in this way, a zero pressure field should result. This is indeed the case if a single anechoic response is subtracted to counteract the doubling of the continuing sound source within the modelled domain from the sum of $C_R(\theta)$ and $D(\theta)$. The right-most panel of Figure 6 shows the resulting zero-pressure wave field. As demonstrated here, the directional sound source approach has been shown as applicable to negative directivity function weights.

CASE 5: Summing Random Target and Inverse Target Directivity Responses to Produce a Monopole Response

In this final test case, the directional sound source implementation is applied to reproduce a highly directional, complex directivity function. The method is as follows: a target directivity pattern is generated by assigning a random number in the range [0:1] to $S(\theta)$ (denoted $R(\theta)$) for every contributing boundary node. The correctness of the result is then verified by also simulating the directivity given by 1 - $R(\theta)$ (with some abuse of terminology, referred to here as the "inverse" of $R(\theta)$). The sum of the two resulting wave fields after the first reflection should then be equal to that rendered by the monopole $M(\theta)$. The target directivity functions are shown in Figure 7 below and the results of simulation in Figure 8 overleaf.

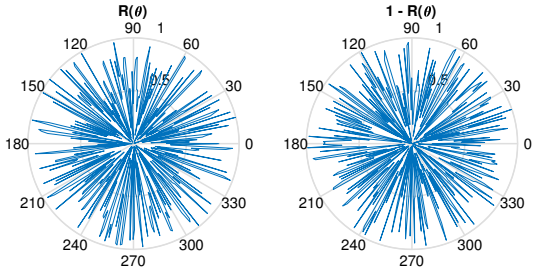


Figure 7: The complex directivity function $R(\theta)$ plotted over the azimuth against degrees alongside its inverse 1-$R(\theta)$.

As shown in Figure 8, the two wave fields rendered in response to the directivity functions $R(\theta)$ and 1-$R(\theta)$ sum together to pro-

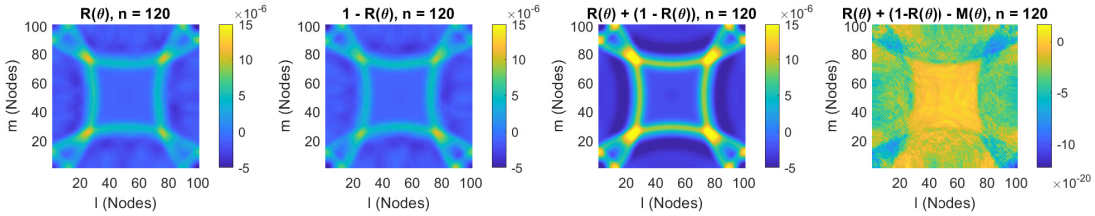


Figure 8: Results of CASE 5. (From left to right) The wave fields rendered in response to directivity functions $R(\theta)$, $1 - R(\theta)$, their sum and the difference between the sum and the monopole $M(\theta)$.

duce a monopole reflection pattern (third panel from the left). As per the previous test cases, the addition of more than one rendered wave field results in an excess of the continuing sound source within the summed pressure field. Hence, in this case a single anechoic response is subtracted from the sum of $R(\theta)$ and $1 - R(\theta)$ prior to rendering the comparison with $M(\theta)$ as shown in the right most panel of Figure 8. Given that a zero pressure field results, it is clear that the sum of the two responses to the complex directional source functions equates that of a monopole. As such, this method has been demonstrated as capable of representing highly complex source directivities.

5. LIMITATIONS OF CURRENT APPROACH

This research has focused primarily on the derivation and proof-of-concept work undertaken to establish the basis for future refinement of the directive source approach. While the approach is shown to deliver encouraging results in a simple concave domain, there are limitations on its usage in acoustic modeling. Firstly, the source strategy is well suited to modeling scenarios in which detailed source directivity functions are sought but the source location and directivity function must remain static for each simulation. Secondly, the approach is limited to application in concave domain geometries for directivities that are consistent across the simulated bandwidth. However, at high frequencies (i.e. those at which diffraction may be neglected) this model may be extended to arbitrary domains through manipulation of transfer functions at boundary nodes that have a direct line of sight to the source location.

6. CONCLUSION

A directional sound source excitation method for IR rendering in FDTD acoustic models has been formulated and shown to provide correct results for a range of test cases and target directivity functions. This approach may be used for static source IR rendering to produce simulations of increased accuracy when complex sound source directivity functions are a primary concern (i.e. room acoustics modeling and virtualization). Future work will focus on improving validation approaches, addressing known limitations and extending the method to 3D acoustics modelling, frequency-dependent source directivity characteristics and multiple simultaneous directional source excitations. Futhermore, the computational overheads of applying the weighting functions during simulation run-time remain to be assessed.

7. ACKNOWLEDGEMENTS

The author wishes to thank the anonymous reviewers for their helpful and detailed input to the preparation of this manuscript and the shaping of future work on this problem.

8. REFERENCES

[1] J. Escolano, J. J. Lopez, and B. Pueo, "Broadband directive sources for acoustic discrete-time simulations," *J. Acoust. Soc. Am.*, vol. 126, no. 6, pp. 2856–2859, December 2009.

[2] J. Escolano, J. J. Lopez, and B. Pueo, "Directive sources in acoustic discrete-time domain simulations based on directivity diagrams," *J. Acoust. Soc. Am.*, vol. 121, pp. EL256–EL262, May 2007.

[3] A. Southern and D. Murphy, "Low complexity directional sound sources for finite difference time domain room acoustic models," in *126th Audio Eng. Soc. Conv.*, May 2009.

[4] F. Georgiou and M. C. J. Hornikx, "Incorporating source directivity in the pseudospectral time-domain method by using spherical harmonics," in *43rd International Congress on Noise Control Engineering*, January 2014.

[5] R. Mehra, L. Antani, and D. Manocha, "Source directivity and spatial audio for interactive wave-based sound propagation," in *20th Int. Conf. on Auditory Display (ICAD)*, June 2014.

[6] H. Hacıhabiboğlu, B. Gunel, and A. M. Kondoz, "Time-domain simulation of directive sources in 3D digital waveguide mesh-based acoustical models," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 16, no. 5, pp. 934–946, July 2008.

[7] S. Bilbao and B. Hamilton, "Directional source modelling in wave-based room acoustics simulation," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, October 2017, pp. 121–125.

[8] S. Bilbao and B. Hamilton, "Directional Sources in Wave-Based Acoustics Simulation," *IEEE/ACM Transations on Audio, Speech and Language Processing*, vol. 27, no. 2, pp. 415–428, February 2019.

[9] D. Takeuchi, K. Yatabe, and Y. Oikawa, "Source directivity approximation for finite-difference time-domain simulation by estimating initial value," *J. Acoust. Soc. Am.*, vol. 145, no. 4, pp. 2638–2649, April 2019.

[10] K. Kowalczyk and M. van Walstijn, "Room acoustics simulation using 3-D compact explicit FDTD schemes," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 19, no. 1, pp. 34–46, April 2011.

[11] K. Kowalczyk and M. van Walstijn, "Formulation of locally reacting surfaces in FDTD/K-DWM modelling of acoustic spaces," *Acta Acustica united with Acustica*, vol. 94, pp. 891–906, September 2008.

[12] H. Kuttruff, *Room Acoustics*, Taylor Francis, London, 5 edition, 2009.

[13] J. Botts and L. Savioja, "Spectral and pseudospectral properties of finite difference models used in audio and room acoustics," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 22, no. 9, pp. 1403–1412, June 2014.

[14] C. J. Webb and S. Bilbao, "Computing room acoustics with CUDA - 3D FDTD schemes with boundary losses and viscosity," in *IEEE Int. Conf. on Acoustics, Speech and Signal Process.*, May 2011, pp. 317–320.

[15] J. Escolano, C. Spa, Adán Garriga, and T. Mateos, "Removal of afterglow effects in 2-D discrete-time room acoustics simulations," *Applied Acoustics*, vol. 74, no. 6, pp. 818–822, June 2013.

[16] S. Bilbao and J. O. Smith, "Finite difference schemes and digital waveguide networks for the wave equation: stability, passivity, and numerical dispersion," *IEEE Trans. Audio and Speech Process.*, vol. 11, no. 3, pp. 255–266, May 2003.

[17] K. Kowalczyk, *Boundary and medium modelling using compact finite difference schemes in simulations of room acoustics for audio and architectural design applications*, Ph.D. thesis, Queen's University Belfast, November 2008.

[18] K. Kowalczyk and M. van Walstijn, "Wideband and isotropic room acoustics simulation using 2-D interpolated FDTD schemes," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 18, no. 1, pp. 78–89, February 2010.

[19] D. Murphy, A. Southern, and L. Savioja, "Source excitation strategies for obtaining impulse responses in finite difference time domain room acoustics simulation," *Appl. Acoustics*, vol. 82, pp. 6–14, August 2014.

FIRST-ORDER AMBISONIC CODING WITH PCA MATRIXING AND QUATERNION-BASED INTERPOLATION

Pierre Mahé
Orange Labs, Lannion, France
L3i, University of La Rochelle, France
pierre.mahe@orange.com

Stéphane Ragot
Orange Labs, Lannion, France
stephane.ragot@orange.com

Sylvain Marchand
L3i, University of La Rochelle, France
sylvain.marchand@univ-lr.fr

ABSTRACT

We present a spatial audio coding method which can extend existing speech/audio codecs, such as EVS or Opus, to represent first-order ambisonic (FOA) signals at low bit rates. The proposed method is based on principal component analysis (PCA) to decorrelate ambisonic components prior to multi-mono coding. The PCA rotation matrices are quantized in the generalized Euler angle domain; they are interpolated in quaternion domain to avoid discontinuities between successive signal blocks. We also describe an adaptive bit allocation algorithm for an optimized multi-mono coding of principal components. A subjective evaluation using the MUSHRA methodology is presented to compare the performance of the proposed method with naive multi-mono coding using a fixed bit allocation. Results show significant quality improvements at bit rates in the range of 52.8 kbit/s (4×13.2) to 97.6 kbit/s (4×24.4) using the EVS codec.

1. INTRODUCTION

Conversational applications such as telephony are typically limited to mono, with no spatial representation of the sound scene. With the emergence of new applications such as virtual reality (VR) and extended reality (XR) and the availability of devices supporting spatial audio capture and playback, there is a need to extend traditional speech/audio codecs to enable immersive communication. There are currently different spatial audio codecs developed for non-conversational applications (streaming, broadcast, etc.), including discrete coding of individual audio channels by MPEG AAC or HE-AAC, MPEG Surround [1], Dolby AC-3 or E-AC-3 [2], and more recently MPEG-H 3D Audio [3], Dolby AC-4 [4] or DTS-UHD [5]. The most recent spatial audio codecs can handle various input formats, such as multichannel audio, object-based audio, ambisonics (also called scene-based audio [3]), and multiple playback formats (e.g. mono, stereo, binaural audio, various multichannel loudspeaker setups). In this work, we investigate how spatial audio can be provided in conversational applications by extending codecs currently used in telephony or voice over IP (VoIP).

We focus in particular on reusing the Enhanced Voice Services (EVS) codec [6] which represents the state-of-art audio quality for mobile telephony applications. The EVS codec supports only mono input and output signals; a naive approach to extend this codec to spatial audio is to code each input channel by a separate instance of the mono codec – this approach is later referred to as *multi-mono coding*. Note that the Opus codec [7] supports mono, stereo, multichannel, and recently it has been extended to code ambisonic signals, using multi-mono coding or fixed channel matrixing followed by multi-stereo coding [8].

Several methods have been proposed to code ambisonic signals. One approach is to use a parametric model of the sound field, with an assumption on the number of audio sources in the scene. For First-Order Ambisonic (FOA) coding, DirAC [9, Chap.5] assumes that there is a single predominant audio source in each time-frequency tile. At the encoder side, a mono downmix signal representing the predominant source is extracted together with directional parameters – in terms of direction of arrival (DoA) and diffuseness – in each time/frequency tile. The mono signal and the parameters are coded and transmitted. At the decoder side, the sound field is reconstructed by panning the predominant source (based on the DoA, typically using VBAP); this is signal is combined with an ambiance using decorrelation and the diffuseness parameters. The DirAC method has been extended to High-Order Ambisonics (HOA) in the so-called HO-DirAC [9, Chap.6] where the sound field is divided into angular sectors. For each angular sector, one source is extracted. More recently, Compass [10] was proposed as a method inspired by DirAC. This method overcomes the limitation of the number of sources. The number of sources is estimated and sources are extracted by Principal Component Analysis (PCA) and the residual signal is downmixed to obtain an ambiance signal.

The Compass method is similar to the MPEG-H 3D Audio codec [3] which supports ambisonic signals as one type of input format. In MPEG-H 3D Audio, the input ambisonic signal is decomposed into a number of component signals; these signals represent a number of predominant sources with an ambiance signal, they are coded using a core codec derived from MPEG USAC [11]. Predominant sources may be extracted using plane wave decomposition or Principal Component Analysis (PCA). When PCA is used – noting that PCA may be equivalently implemented using a Singular Value Decomposition (SVD) – components may change dramatically between consecutive frames causing channel permutations and signal discontinuities [12] for complex sound scenes (with many audio sources, sudden changes, etc.). The MPEG-H 3D audio codec employs channel re-alignment, overlap-add and linear interpolation to mitigate these problems. Improvements to the MPEG-H 3D Audio codec were proposed in [12, 13]; the SVD decomposition is done in the MDCT domain to ensure smooth transitions across frames.

In this work, we wanted to avoid any assumption on the audio scene (e.g. presence of predominant sources, number of sources) based on existing mono and stereo codecs. The most basic approach is multi-mono coding; at low bit rates, due to the corre-

lation of ambisonic components, the signal structure is typically degraded, which creates several spatial artifacts: spatial reduction, blur, phantom source. Multi-stereo coding with fixed matrixing was proposed in [8, 14]. The principle consists in combining the input B-format channels by matrixing, which may be interpreted as beamforming in configurable directions. This approach has the advantage to transform the signal into a format which is more suited to multi-stereo coding, however the fixed matrixing is in general not optimal. An alternative approach proposed in [15] is based on predictive coding of 1st-order FOA components based on the 0th-order component, which is typically efficient when the scene has few sources. In this paper, we propose an extension of multi-mono coding approach to represent FOA signals. We make no assumption on the scene content. The proposed method relies on PCA to obtain an adaptive matrixing of FOA components in time domain. To guarantee signal continuity between components across frames, two mechanisms are implemented: principal component matching (similar to MPEG-H 3D Audio and [12, 13]) and rotation matrix interpolation in quaternion domain. We also describe an adaptive bit allocation algorithm for an optimized multi-mono coding of principal components.

This paper is organized as follows. Section 2 gives a brief overview of ambisonics. Section 3 provides the relevant background on quaternions, the representation of 4D rotation matrices by double quaternions, and the interpolation of rotation matrices in quaternion domain. Section 4 describes in details the proposed coding method. Section 5 presents the results of a subjective quality evaluation comparing the proposed method and naive multi-mono coding, before concluding in Section 6.

2. AMBISONICS

Ambisonics is based on a decomposition of the sound field into an orthogonal basis of spherical harmonics. Initially limited to 1st order by Gerzon [16], the formalism was extended to high orders by Daniel [17]. The sound field may be expressed by the equation [18]:

$$p(t,f,r,\theta,\phi) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} p_{nm}(t,f) Y_{nm}(\theta,\phi) \qquad (1)$$

where $p(t,f,r,\theta,\phi)$ is the sound pressure at time t, frequency f, distance r, azimuth θ and elevation ϕ, $Y_{nm}(.,.)$ is the spherical harmonic function, and $p_{nm}(t,f)$ is the ambisonic coefficients of order n and degree m. In practice, the sound field representation is truncated to a finite order N. The higher N, the better the sound field representation accuracy. The so-called B-format corresponds to the ambisonic components $p_{nm}(t,f)$. For a given order N the number of ambisonic components is $(N+1)^2$ in 3D and $2N+1$ in 2D (in horizontal-only ambisonics). From the B-format, rendering is required to reproduce the sound field to the listener. For loudspeaker configurations the rendering computes the signal played by each loudspeaker. For headphone listening binaural rendering may be performed by decoding over virtual loudspeakers, convolving and combining the resulting feed signals by Head-Related Impulse Responses (HRIRs) [19] which are filters measured for each ear.

3. QUATERNIONS AND 4D ROTATION MATRICES

Quaternions (also called hypercomplex numbers or hamiltonians) were introduced in 1843 by Hamilton [20] to define a vectorial system generalizing complex numbers. A quaternion q is defined as $q = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$, where $(a,b,c,d) \in \mathbb{R}^4$, with the following rules for $\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$: $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$. We can also write $q = a + \mathbf{q}$, where a is the *scalar* (or real) part of q and $\mathbf{q} = b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$ is the *vector* (or imaginary) part of q. Note that $\mathbf{q}$ may be interpreted as a 3D vector by identifying $\mathbf{i}$, $\mathbf{j}$ and $\mathbf{k}$ to three orthogonal Cartesian unit vectors. A quaternion q having zero scalar part is called a *pure quaternion*. We do not review basic operations on quaternions (e.g. addition $q_1 + q_2$, multiplication $q_1 q_2$, dot product $q_1 \cdot q_2$, conjugate $\overline{q}$ and inverse q^{-1}) – see for instance [20, 21, 22] for more details. We recall that the dot product between two quaternions $q_1 = a_1 + b_1\mathbf{i} + c_1\mathbf{j} + d_1\mathbf{k}$ and $q_2 = a_2 + b_2\mathbf{i} + c_2\mathbf{j} + d_2\mathbf{k}$ is $q_1 \cdot q_2 = a_1a_2 + b_1b_2 + c_1c_2 + d_1d_2$ which corresponds to the usual dot product of 4-dimensional vectors, and the norm $|\mathbf{q}|$ of a quaternion is: $|q| = \sqrt{q \cdot q} = \sqrt{a^2 + b^2 + c^2 + d^2}$. When $|q| = 1$, q is said to be a *unit-norm quaternion*.

Quaternions are often used as a representations of 3D rotations. We recall that the set of 3D rotations (called special orthogonal group in dimension 3 or $SO(3)$) can be mapped to the unit sphere in $\mathbb{R}^4$ under a one-to-two mapping [21, 22, 23], namely each 3D rotation matrix maps to two antipodal unit-norm quaternions: q and $-q$, therefore this mapping is not unique. A 3D rotation of angle θ and normalized axis $\mathbf{n}$ (with $|\mathbf{n}| = 1$) is represented by the unit quaternion $q = \cos(\theta/2) + \sin(\theta/2)\mathbf{n}$ (or its opposite $-q$); the rotation of a 3D vector $\mathbf{u}$ to $\mathbf{v}$ can be equivalently obtained by quaternion multiplication: $\mathbf{v} = q\mathbf{u}q^{-1} = q\mathbf{u}\overline{q}$, if $\mathbf{u}$ and $\mathbf{u}$ are interpreted as pure quaternions.

Unlike other representations Euler angles or axis-angle, quaternions are interesting to interpolate rotations. One common method is called spherical linear interpolation (slerp) and consists in the following principle [21]:

$$slerp(q_1, q_2, \gamma) = q_1(q_1^{-1}q_2)^{\gamma} \qquad (2)$$

where q_1 and q_2 are respectively the starting and ending quaternions and $0 \leq \gamma \leq 1$ is the interpolation factor. The slerp interpolation may also be formulated as [21]:

$$slerp(q_1, q_2, \gamma) = \frac{\sin((1-\gamma)\Omega)}{\sin(\Omega)} q_1 + \frac{\sin(\gamma\Omega)}{\sin(\Omega)} q_2 \qquad (3)$$

where $\Omega = \arccos(q_1 \cdot q_2)$ is the angle between the two quaternions q_1 and q_2. This interpolation boils down to interpolating along the grand circle (or geodesics) on the unit 4D sphere with a constant angular speed as a function of γ. Due to the non-unique representation of 3D rotations by antipodal unit-norm quaternions, one has to ensure that the quaternion trajectory follows the shortest path [23] on the unit sphere in $\mathbb{R}^4$ by selecting among the two possible choices $\pm q_2$ for the end point. Alternative interpolation methods, such as normalized linear interpolation (nlerp) or splines [21], are not reviewed here.

In this work, we used the representation of 4D rotation matrices as double quaternions. The multiplication of two quaternion matrices, an anti-quaternion matrix $\mathbf{Q}^{\star}$ and a quaternion matrix $\mathbf{P}$, associated to two quaternions $q = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$ and $p = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$, is defined as [22]:

$$\mathbf{R} = \mathbf{Q}^{\star}.\mathbf{P} = \mathbf{P}.\mathbf{Q}^{\star} \qquad (4)$$

where

$$\mathbf{Q}^{\star} = \begin{pmatrix} a & b & c & d \\ -b & a & -d & c \\ -c & d & a & -b \\ -d & -c & b & a \end{pmatrix} \tag{5}$$

and

$$\mathbf{P} = \begin{pmatrix} w & -x & -y & -z \\ x & w & -z & y \\ y & z & w & -x \\ z & -y & x & w \end{pmatrix} \tag{6}$$

It is possible [22] to show that the product $\mathbf{R}$ has the properties of a rotation matrix ($\mathbf{R}.\mathbf{R}^T = \mathbf{R}^T.\mathbf{R} = \mathbf{I}$ and $\det(\mathbf{R}) = +1$); conversely, given a 4D rotation matrix $\mathbf{R}$, it is possible [22] to factorize this matrix as in Eq. 4. This factorization (sometimes called Cayley's factorization) may be obtained using the method described in [24] (noting that a factor $1/4$ is missing in Eqs. 64 and 65 in [24]); this method is also described in [22] (together with alternative factorization approaches). The factorization relies on an intermediate matrix (called associate matrix in [24] or tetragonal transform in [22]) to determine the two quaternions q and p up to sign (i.e. $-q$ and $-p$ also form a valid solution).

Similar to 3D rotations, two 4D rotation matrices may be interpolated by interpolating separately the associated pairs of quaternions (for instance using slerp). However, it is important to keep the sign consistent between double quaternions when constraining the shortest path.

Note that there are other representations for 4D rotation matrices. In this work, we also used the generalized Euler angles defined in [25]. In general, a n-dimensional rotation matrix is characterized by $n(n-1)/2$ generalized Euler angles and for $n = 4$ we have 6 angles. We refer to [25] for details on the conversion between an n-dimensional rotation matrix and generalized Euler angles.

4. PROPOSED CODING METHOD

We describe in this section the proposed coding method. The input signal is assumed to be a first-order ambisonic signal with the ACN ordering convention (W, Y, Z, X). The $n = 4$ ambisonic components will be labeled respectively with an index $i = 1, \cdots, n$. Each ambisonic component is sampled at 32 kHz. The coding method operates on successive frames of 20 ms. The different coding and decoding steps are described in the high-level diagram shown in Figure 1. We describe below the implementation for each functional blocks.

4.1. High-pass pre-processing (HPF)

The $n = 4$ channels of the input FOA signal are separately pre-processed by a 20 Hz high-pass IIR filter from the EVS codec defined by the transfer function

$$H_{pre}(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{7}$$

where filter coefficients (b_i, a_i) are taken from [26]. This preprocessing is used here to avoid any bias in the subsequent estimation of the covariance matrix.

4.2. Principal Component Analysis (PCA)

In each frame, the sample covariance matrix $\mathbf{C_{XX}}$ is estimated based on the 4-dimensional input $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_n]$:

$$\mathbf{C_{XX}} = \frac{1}{n-1} \sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^T \tag{8}$$

where $\mathbf{x}_i$ is the i-th column of $\mathbf{X}$ and corresponds to the current frame of i-th ambisonic channel. The covariance matrix $\mathbf{C_{XX}}$ is factorized by eigenvalue decomposition as:

$$\mathbf{C_{XX}} = \mathbf{V} \Lambda \mathbf{V}^T \tag{9}$$

where $\mathbf{V}$ is the eigenvector matrix (with eigenvectors as columns) and $\Lambda = diag(\lambda_1, \cdots, \lambda_n)$ is the diagonal matrix whose coefficients are the eigenvalues.

In general the matrix $\mathbf{V}$ is orthogonal and it may be either a rotation matrix ($det(\mathbf{V}) = +1$) or a reflection matrix ($det(\mathbf{V}) = -1$). In this work, we ensured that the eigenvector matrix defines a rotation matrix by inverting the sign of $\mathbf{v}_n$ if $det(\mathbf{V}) = -1$. In the following, the resulting rotation matrix in the current frame of index t will be denoted $\mathbf{V}_t$.

4.3. Re-alignment of eigenvectors

From frame to frame eigenvectors might change significantly. These modifications might create discontinuities in the signal, which can degrade audio quality. To improve signal continuity between successive frames, a signed permutation is applied to the eigenvector matrix $\mathbf{V}_t$ in the current frame of index t based on the eigenvector matrix $\mathbf{V}_{t-1}$ in the previous frame.
The signed permutation is obtained in two steps:

1. A permutation is found by matching eigenvectors in frames t and $t-1$ according to the axes (not directions) of each basis vector. This problem can be seen as an assignment problem where the goal is to find the closest eigenvector in frame t for each eigenvector in the previous frame $t-1$. To solve this assignment problem, we used the Hungarian algorithm to find the optimal solution in a similar way to [12]. After this step, eigenvectors are permuted. This allows to maximize similarity between the two basis.

   The similarity being defined as:

   $$\mathbf{J}_t = tr(|\mathbf{V}_t.\mathbf{V}_{t-1}^T|) \tag{10}$$

   where $tr(|.|)$ is the trace of the matrix $|\mathbf{V}_t.\mathbf{V}_{t-1}^T|$ whose coefficients are the absolute values.

   After applying the optimal permutation to $\mathbf{V}_t$ we obtain a new matrix of eigenvectors $\tilde{\mathbf{V}}_t$.

2. The direction of each eigenvector is determined based on the permuted eigenvector matrix in the current frame $\tilde{\mathbf{V}}_t$ for frame t and the rotation matrix $\mathbf{V}_{t-1}$ in the previous frame $t-1$

   $$\mathbf{\Gamma}_t = \tilde{\mathbf{V}}_t.\mathbf{V}_{t-1}^T \tag{11}$$

   A negative diagonal value in $\mathbf{\Gamma}_t$ indicates a direction inversion between two frames. The sign of the respective columns of $\tilde{\mathbf{V}}_t$ is inverted to compensate for this change of direction.

Note that the resulting matrix $\tilde{\mathbf{V}}_t$ in the current frame will be saved for the next frame processing to become $\mathbf{V}_{t-1}$.

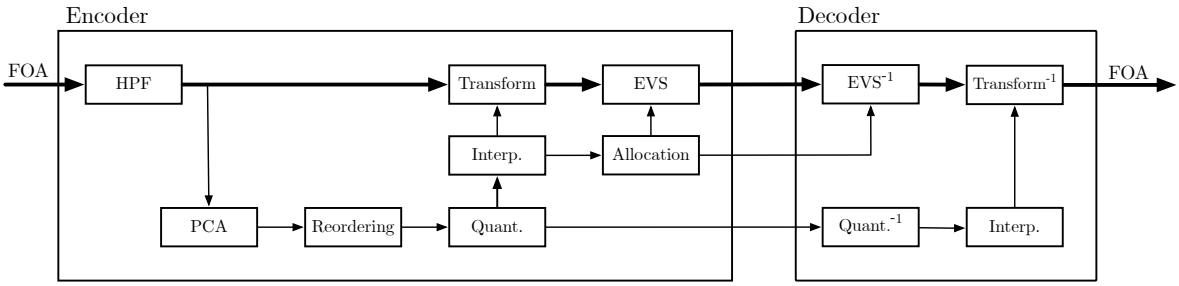


Figure 1: *Overview of proposed coding method.*

4.4. Quantization of the 4D PCA rotation matrix

In general a n-dimensional rotation matrix has $n(n-1)/2$ degrees of freedom. In [27] 2D and 3D rotation matrices were quantized with an angle representation: one angle in 2D, three Euler angles in 3D. In this work we used a similar idea by quantizing 6 generalized Euler angles. The 6 angles were obtained from the 4D rotation matrix [25] and coded by scalar quantization with a budget of respectively 8 and 9 bits for angles defined over a support of length π and 2π, with an overall budget of 51 bits per frame for 6 angles.

4.5. Interpolation of quantized rotation matrices by subframe

The rotation matrices are interpolated by subframes to smooth variations of PCA rotation matrices and avoid signal discontinuities. The rotation matrix representation is not suitable for interpolation. Instead, we converted 4D rotation matrices to double quaternions as explained in Section 3. The current frame of length $L = 640$ samples is divided into K sub-frames. We used $K = 128$ which gives a subframe length of $L/K = 10$ samples (0.3125 ms). For each subframe of index $1 \leq k \leq K$ in the current frame, the left (q_{t-1}, q_t) and right quaternions (p_{t-1}, p_t) are interpolated by the slerp algorithm with an interpolation factor given by $\gamma = k/K$ as defined in Eq. 3. The interpolated double quaternions are converted back to a 4D matrix using Eqs. 4, 5, 6.

4.6. PCA matrixing

The pre-processed FOA signal is transformed into 4 principal components by applyinng the interpolated rotation matrix in each subframe.

4.7. Adaptive bit rate allocation to multi-mono EVS coding

After PCA matrixing the $n = 4$ channels could have been coded using a fixed bit allocation, as in naive multi-mono coding. Depending on the input signals, the signals after PCA matrixing may vary significantly in importance and it was found experimentally that an adaptive bit allocation is necessary to optimize quality. We use a greedy bit allocation algorithm which aims at maximizing the following score:

$$S_(b_1, \cdots, b_n) = \sum_{i=1}^{n} Q(b_i).E_i^{\beta} \tag{12}$$

where b_i and E_i are respectively the bit allocation and the energy of the i^{th} channel in the current frame and $Q(b_i)$ is a quality score reflecting the estimated quality of the codec for a bit rate corresponding to b_i bits per 20 ms frame. This optimization is subject to the constraint $b_1 + \cdots + b_n \leq B$ where B is the budget allocated for multi-mono coding. The term $Q(b_i)$ is defined here in Table 1 to reflect the fact that the EVS codec quality does not increase [28] linearly with increasing bit rate and it was found experimentally the theoretical rate-distortion function that would apply for a usual source model (e.g. Gaussian source) would not be suitable. The scores $Q(b_i)$ correspond here to average MOS (Mean Opinion Score) values found during the performance characterization of the EVS codec by 3GPP [28]. Note that if another core codec than EVS was used, the values $Q(b_i)$ could be adjusted accordingly; for instance, a quality evaluation of Opus can be found in [29]. To guarantee maximum compatibility with the EVS codec, the bit allocation to individual audio channels is restricted to the set of EVS bit rates: 9.6, 13.2, 16.4, 24.4, 32, 48, 64, 96, 128 kbit/s. Note that a minimum bit allocation of 9.6 kbit/s was defined to ensure a super-wideband coded bandwidth. The energy term E_i^{β} is raised to the power β, where $0 \leq \beta \leq 1$ is defined as an extra tuning parameter: when β is close to 1, channels with more energy will dominate the bit allocation and when $\beta = 0$ channels would receive an equal allocation. The value of $\beta = 0.5$ was selected experimentally. The bit allocation $b_1, \cdots, b_n$ selected in the current frame is coded and transmitted to the decoder.

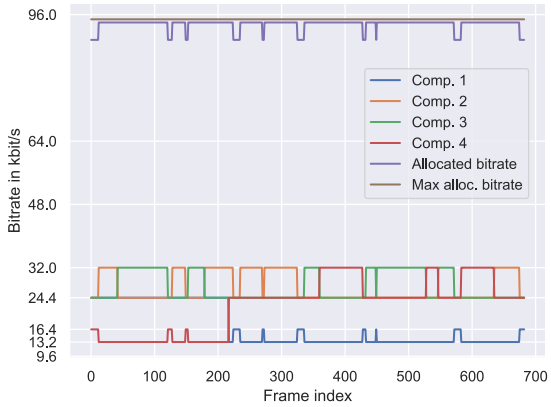


Figure 2: *Bit rate allocation example at* 4×24.4 *kbit/s.*

Figure 2 shows an example of bit rate allocation to components (denoted Comp. 1, 2, 3, 4) in successive 20 ms frames for one test item. The maximum allocated bit rate (94.75 kbit/s) corresponds to the difference between 4×24.4 kbit/s and the bit rate used for side information (2.85 kbit/s). The overall allocated bit

Table 1: *Bit allocation parameters.*

b_i	192	264	328	488	640	960	1280	1920	2560
rate kbit/s	9.6	13.2	16.4	24.4	32	48	64	98	128
$Q(b_i)$	3.62	3.79	4.25	4.60	4.53	4.82	4.83	4.85	4.87

rate (corresponding to $b_1 + b_2 + b_3 + b_4$) does not always use the available budget and in this case padding bits have to be added to the bitstream.

4.8. Multi-mono EVS coding

The bit rate to code the meta-data (rotation matrices, bit allocation) is subtracted to the target bit rate and the remaining bit rate is used for the adaptive bit allocation to multi-mono coding. The transformed FOA channels are coded by separate instances of the EVS codec and the associated bitstreams are transmitted to the decoding part. In this work we used the fixed-point implementation of EVS (V14.2.0) with discontinuous transmission disabled.

A bitstream structure example is shown in Figure 3 for the bit rate of $4 \times 24.4 = 97.6$ kbit/s; the bitstream is divided in several sections: bit allocation (6 bits), quantized generalized Euler angles (51 bits), four coded channels ($b_1 + b_2 + b_3 + b_4$ bits) and padding (when needed).

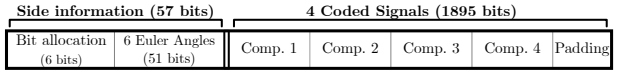


Figure 3: *Bitstream structure example at* $4 \times 24.4 kbit/s$.

4.9. Decoding part

The decoding part is similar to the encoding part. The bit allocation is demultiplexed and the 4 coded channels are decoded by separate instances of the EVS decoder. In addition, the generalized Euler angles are decoded and converted into a 4D rotation matrix. The interpolation in quaternion domain done in each subframe is performed. The signal in the current frame divided in K subframes is then transformed by inverse PCA matrixing to retrieve the reconstructed ambisonic components.

5. EVALUATION

5.1. Test methodology

We conducted subjective tests according to the MUSHRA methodology [30] to compare the perceptual performance of naive multi-mono coding and the proposed coding method. For each item, subjects were asked to evaluate the quality of conditions with an integer grading scale ranging of 0 to 100. This interval is divided in 5 sections of 20 points: bad (0-20) to excellent (80-100).

The test conditions included three specific items: the hidden reference (FOA) and two anchors. Traditional MUSHRA tests for mono signals typically use a low anchor (reference processed by a 3.5kHz low-pass filter) and a medium anchor (reference processed by a 7kHz low-pass filter). For MUSHRA tests with stereo, it is suggested to use "reduced stereo image" as degradations in anchors [30]. There is no clear recommendation on spatial alterations for MUSHRA tests with ambisonic signals. In this work we used anchors with some spatial deformation. We use the following spatial reduction defined by:

$$FOA = \begin{pmatrix} W \\ \alpha X \\ \alpha Y \\ \alpha Z \end{pmatrix}, \qquad \alpha \in [0, 1] \tag{13}$$

with $\alpha = 0.65$ and $\alpha = 0.8$ for the low and medium anchors, respectively.

Table 2: *List of MUSHRA conditions.*

Short name	**Description**
HREF	FOA hidden reference
LOW_ANCHOR	3.5 kHz LP-filtered and spatially-reduced FOA ($\alpha = 0.65$)
MED_ANCHOR	7 kHz LP-filtered and spatially-reduced FOA ($\alpha = 0.8$)
MULTI52	FOA coded by multimono EVS at 4×13.2 kbit/s
MULTI65	FOA coded by multimono EVS at 4×16.4 kbit/s
MULTI97	FOA coded by multimono EVS at 4×24.4 kbit/s
PCA52	FOA coded by proposed method at 52.8 kbit/s
PCA65	FOA coded by proposed method at 65.6 kbit/s
PCA97	FOA coded by proposed method at 97.6 kbit/s

5.2. Experimental setup

The subjects were placed in an audio listening room. The room complied with ITU-R recommendations [31] in terms of acoustics properties, reverberation time and background noise. All subjects conducted the listening test with the same hardware, which included an external sound-card (Focusrite Scarlett 6i6) and high-quality headphones (Sennheiser HD 650) and a test interface running on a MacBook pro computer.

The test items consisted of 10 challenging ambisonic items: 4 voice items, 4 music items and 2 ambient scenes. Six items were real recordings made by ambisonic microphones (EigenMike or SoundField SPS200), others were synthetic items. A description of these items can be found in Appendix 7. Each item was about 10 s long. All FOA items were binauralized with Resonance Audio renderer [32] using generic HRTFs corresponding to the KU100 manikin. Original FOA signals were normalized in loudness using the following procedure: the rerefence FOA signals were binauralized using the Resonance audio renderer; the resulting binaural signal was normalized to -23 dBLUFS according to ITU-R BS.1770 [33], and after verifying that there was no saturation in binaural signals, the obtained scale factor was re-applied to the respective coded FOA signal.
In total 11 listeners participated in the test; all of them are expert or experienced listeners without hearing impairments. Each item was codec at three bit rates for multi-mono coding: 52.8, 65.6, 97.6 kbit/s which corresponds to a fixed allocation of 13.2, 16.4 and 24.4 kbit/s per channel. For the proposed coding method, as explained in Section 4, the bit rate was dynamically distributed between channels; however the target (maximum) bitrate was set to the same bit rate as multi-mono coding for a fair comparison. All test conditions are summarized in Table 2.

5.3. Test results

The subjective evaluation results, including the mean and 95% confidence intervals, are presented in Figure 4. They show that the proposed coding method improves quality when compared with multi-mono coding at the same bit rate. It is particularly noticeable that multi-mono coding at 65.6 kbit/s is equivalent to the proposed coding method at 52.8 kbit/s.

These quality improvements are largely due to the suppression of spatial artifacts. These artifacts are present at every bit rate in multi-mono coding and they can be classified into three categories: diffuse blur, spatial centering, phantom source. These three types of artifacts in multi-mono coding result from the degraded structure between ambisonic components. With the proposed coding method, these artifacts are mostly removed because the structure is less important after PCA matrixing. This explanation was supported by the feedback from some subjects, after they conducted the subjective test.

It is possible to analyze MUSHRA scores for different item categories. Figure 5 shows the scores for recorded and synthetic scenes. As can be seen, the proposed coding method brings significant improvements for synthetic items. The result for each item is the same whatever the position of sources or the number of sources. Two assumptions may explain this result. The first one is that the sources do not interact together, and the PCA can decorrelate each source, consequently the proposed method avoided spatial artifacts. The second one is that spatialization is more pronounced in synthetic items (wider displacement, more localized source). This spatialization puts the emphasis for the listener on spatial artifacts.

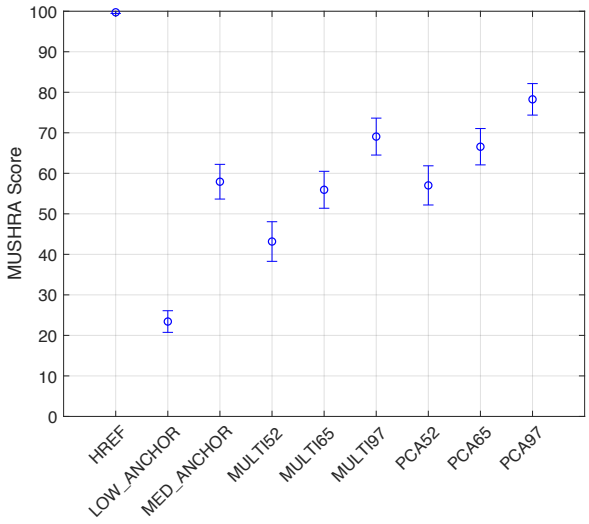


Figure 4: *MUSHRA mean scores with 95% confidence intervals.*

6. CONCLUSION

This paper presented a spatial extension of a mono speech/audio codec. The proposing coding method operates in time domain to avoid extra delay and allow maximum compatibility with existing codecs (e.g. EVS or Opus) which are used as a black box. The ambisonic components are transformed by adaptive matrixing depending on the audio scene.

For each frame, a PCA allows to find a new basis where the components are decorrelated. To guarantee smooth transition between consecutive frames, rotation matrices are interpolated in quaternion domain. Subjective test results show that the PCA matrixing together with the adaptive bit allocation give significant improvements over naive multi-mono coding for bit rates from 4×13.2 to 4×24.4 kbit/s. For future work, it will be interesting to characterize the spatial artifacts for various coding methods and to apply the proposed method to Opus.

7. ACKNOWLEDGMENTS

The authors would like to thank all participants in the audio test. They also thank Jérôme Daniel for helpful discussions on test items and spatial reduction for MUSHRA anchors.

8. REFERENCES

[1] J. Herre et al., "MPEG Surround—The ISO/MPEG Standard for Efficient and Compatible Multichannel Audio Coding," *J. Audio Eng. Soc.*, vol. 56, no. 11, pp. 932–955, 2008.

[2] ATSC Standard, Doc. A/52:2018, "Digital Audio Compression (AC-3, E-AC-3)," January 2018.

[3] J. Herre, J. Hilpert, A. Kuntz, and J. Plogsties, "MPEG-H audio—the new standard for universal spatial/3D audio coding," *J. Audio Eng. Soc.*, vol. 62, no. 12, pp. 821–830, 2015.

[4] ETSI TS 103 190 V1.1.1, "Digital Audio Compression (AC-4) Standard," April 2014.

[5] ETSI TS 103 491 V1.1.1, "DTS-UHD Audio Format; Delivery of Channels, Objects and Ambisonic Sound Fields," April 2017.

[6] S. Bruhn et al., "Standardization of the new 3GPP EVS codec," in *Proc. ICASSP*, 2015, pp. 5703–5707.

[7] J.-M. Valin, K. Vos, and T. Terriberry, "Definition of the Opus Audio Codec," RFC 6716, September 2012.

[8] J. Skoglund, "Ambisonics in an Ogg Opus Container," RFC 8486, October 2018.

[9] V. Pulkki, A. Politis, M.-V. Laitinen, J. Vilkamo, and J. Ahonen, "First-order directional audio coding (DirAC)," in *Parametric Time-Frequency Domain Spatial Audio*, V. Pulkki, S. Delikaris-Manias, and A. Politis, Eds., chapter 5. John Wiley & Sons, 2018.

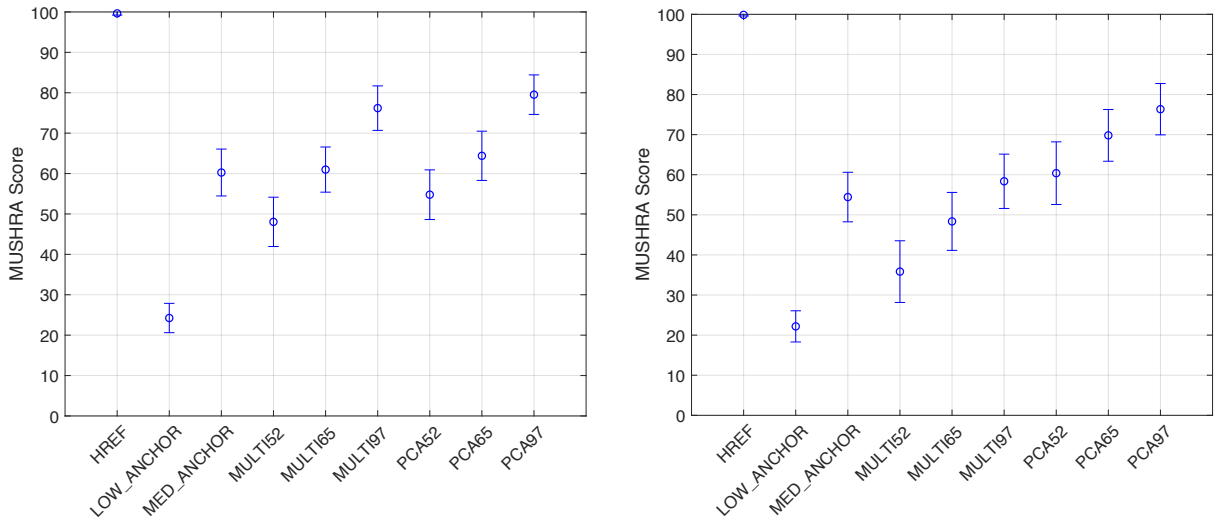
[10] A. Politis, S. Tervo, and V. Pulkki, "Compass: Coding and multidirectional parameterization of ambisonic sound scenes," in *Proc. ICASSP*, 2018, pp. 6802–6806.

[11] M. Neuendorf et al., "The ISO/MPEG unified speech and audio coding standard—consistent high quality for all content types and at all bit rates," *J. Audio Eng. Soc.*, vol. 61, no. 12, pp. 956–977, 2013.

[12] S. Zamani, T. Nanjundaswamy, and K. Rose, "Frequency domain singular value decomposition for efficient spatial audio coding," in *Proc. WASPAA*, 2017, pp. 126–130.

[13] S. Zamani and K. Rose, "Spatial Audio Coding with Backward-Adaptive Singular Value Decomposition," in *145th AES Convention*, 2018.

[14] 3GPP Tdoc S4-171342, "Encoding First-Order Ambisonics with HE-AAC," Source: Dolby.



(a) Recorded scenes

(b) Synthetic scenes

Figure 5: *MUSHRA mean scores with 95% confidence intervals for different types of scenes.*

[15] D. McGrath et al., "Immersive Audio Coding for Virtual Reality Using a Metadata-assisted Extension of the 3GPP EVS Codec," in *Proc. ICASSP*, May 2019.

[16] M.A. Gerzon, "Periphony: With-height sound reproduction," *J. Audio Eng. Soc.*, vol. 21, no. 1, pp. 2–10, 1973.

[17] J. Daniel, *Représentation de champs acoustiques, application à la transmission et à la reproduction de scènes sonores complexes dans un contexte multimédia*, Ph.D. thesis, Université Paris 6, 2000.

[18] B. Rafaely, *Fundamentals of spherical array processing*, Springer, 2015.

[19] H. Møller, M.F. Sørensen, D. Hammershøi, and C.B. Jensen, "Head-related transfer functions of human subjects," *J. Audio Eng. Soc.*, vol. 43, no. 5, pp. 300–321, 1995.

[20] W.R. Hamilton, "On a new species of imaginary quantities connected with a theory of quaternions," *Proc. R. Ir. Acad.*, vol. 2, pp. 424–434, 1843.

[21] A.J. Hanson, *Visualizing Quaternions*, Morgan Kaufmann Publishers, 2006.

[22] P. De Casteljau, *Les quaternions*, Dunod, 1987.

[23] K. Shoemake, "Animating rotation with quaternion curves," *ACM SIGGRAPH Computer Graphics*, vol. 19, no. 3, pp. 245—-254, 1985.

[24] A. Perez-Gracia and F. Thomas, "On Cayley's factorization of 4D rotations and applications," *Advances in Applied Clifford Algebras*, vol. 27, no. 1, pp. 523–538, 2017.

[25] D.K. Hoffman, R.C. Raffenetti, and K. Ruedenberg, "Generalization of Euler Angles to N-Dimensional Orthogonal Matrices," *Journal of Mathematical Physics*, vol. 13, no. 4, pp. 528–533, 1972.

[26] 3GPP TS 26.445, "Codec for Enhanced Voice Services (EVS); Detailed algorithmic description," 2019.

[27] M. Briand, *Études d'algorithmes d'extraction des informations de spatialisation sonore : application aux formats multicanaux*, Ph.D. thesis, INPG Grenoble, 2007.

[28] 3GPP TS 26.952, "Codec for Enhanced Voice Services (EVS); Performance Characterization," 2019.

[29] A. Rämö and H. Toukomaa, "Voice quality characterization of IETF Opus codec," in *Proc. Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[30] ITU-R Rec. BS.1534-3, "Method for the subjective assessment of intermediate quality level of coding systems," 2015.

[31] ITU-R BS.1116-3, "Methods for the subjective assessment of small impairments in audio systems including Multichannel Sound Systems," 2015.

[32] "Resonance audio : Rich, immersive, audio," `https://resonance-audio.github.io/resonance-audio`.

[33] ITU-R Rec. BS.1170-4, "Algorithms to measure audio programme loudness and true-peak audio level," 2015.

[34] ISO/IEC JTC1/SC29/WG11/N13633, "Submission and Evaluation Procedures for 3D Audio," 2013.

[35] "Ambisonia," `www.ambisonia.com`.

[36] E. Roncière, "Compte-rendu de captation (Rapport projet BiLi) – Fiction CNSMDP : 2 femmes pour un fantôme de René de Obaldia," `www.bili-project.org`.

[37] M. Kronlachner, "ambix v0.2.8 – ambisonic plug-in suite," `www.matthiaskronlachner.com/?p=2015`.

APPENDIX: DESCRIPTION OF MUSHRA ITEMS

Synthetic items were generated in Orange Labs, recorded items were captured and mixed by Orange Labs or done jointly with partners (see details below). HOA items were truncated to FOA for testing.

- **Drums (synthetic)** Drum tracks were generated with a MIDI sequencer. HOA mixing, spatialization and near-field simulation were done with an internal Matlab library. This item is a shorterned version of the H_02_Drums1 test item [34] contributed by Orange Labs to MPEG-H standardization.
- **Modern Music (synthetic)** Single source (jazz music) crossing left to right with near-field effect. The mono source is an except of "The Present" in Laurent de Wilde's album "Time for Change". Spatial rendering was done with an internal Matlab library. This item is a shorterned version of the H_03_Modern test item [34] contributed by Orange Labs to MPEG-H standardization.
- **Opera (recorded)** Italian Opera with female singer, harpsichord and strings in concert hall from the Ambisonia database [35]. Players were in front of a SoundField SPS200 microphone.
- **Orchestra (recorded)** Orchestra in concert hall. An EigenMike microphone was placed in middle of the orchestra and spot microphones were also used for ambisonic mixing.
- **Theater (recorded)** Theater play with 3 moving talkers (in French), one in near-field and two in far-field. Recorded with an EigenMike microphone during the Bili Project [36].
- **Kids Playground (synthetic)** Two talkers (female and male in English) mixed with an ambisonic recording of kids playground. The mix was done with Reaper, the mono speech signals were spatialized with the Ambix framework [37], the ambisonic kids playground signal came from the Ambisonia database [35].
- **Little Prince (recorded)** Man reading excerpt of "Le Petit Prince" (in French) at a fixed (left, top) position in a living room. Recorded with an EigenMike microphone.
- **Talks (recorded)** People at different positions talking (in French) in a large room. Recorded with an EigenMike microphone during the Bili Project [36].
- **Nature (synthetic)** Artificial scene with bumblebee sound moving in space, mixed with an ambisonic brook noise and a bird singing at fixed position. The mix was done with Reaper, the mono signals of bumblebee and bird were spatialized with the Ambix framework [37], the bird noise came from the Ambisonia database [35].
- **Applause (recorded)** Applause at the end of a concert. A SoundField SPS200 microphone was placed in the middle of the crowd. The recording came from the Ambisonia database [35].

REAL-TIME PHYSICAL MODELLING FOR ANALOG TAPE MACHINES

Jatin Chowdhury

Center for Computer Research in Music and Acoustics
Stanford University
Palo Alto, CA
jatin@ccrma.stanford.edu

ABSTRACT

For decades, analog magnetic tape recording was the most popular method for recording music, but has been replaced over the past 30 years first by DAT tape, then by DAWs and audio interfaces [1]. Despite being replaced by higher quality technology, many have sought to recreate a "tape" sound through digital effects, despite the distortion, tape “hiss”, and other oddities analog tape produced. The following paper describes the general process of creating a physical model of an analog tape machine starting from basic physical principles, then discusses in-depth a real-time implementation of a physical model of a Sony TC-260 tape machine.

"Whatever you now find weird, ugly, uncomfortable, and nasty about a new medium will surely become its signature. CD distortion, the jitteriness of digital video, the crap sound of 8-bit - all of these will be cherished and emulated as soon as they can be avoided." -Brian Eno [2].

1. INTRODUCTION

While analog magnetic tape recording (see fig. 1) is rarely used in modern recording studios, the sound of analog tape is still often sought after by mixing and mastering engineers. To that end, several prominent audio plugin manufacturers including Waves[1], Universal Audio[2], and U-He[3] have created tape emulating plugins. Unfortunately, the existing literature on analog tape emulation is somewhat lacking. While Arnadottir et at. [3] and Valimaki et al. [4] describe the emulation of tape echo/delay devices, and Valimaki et al [5] describe the emulation of disk-based audio recording media, we were unable to locate any existing research directly discussing digital emulation of the magnetisation process, a gap in research that this publication intends to fill. That said, Kadis [1] and Camras [6] discuss musical use of analog tape recorders in a useful technical manner, and Bertram [7] gives a in-depth technical description of the physical underpinnings of analog magnetic recording; this work intends to build on their foundations. While tape machines also contain electronic circuits that contribute to the machine’s characteristic sound, this publication only considers processes that relate directly to tape magnetisation. For readers wishing to emulate tape machine circuits, a good overview of circuit modelling techniques can be found in [8].

Figure 1: *An analog tape recorder (Sony TC 260)*

2. CONTINUOUS TIME SYSTEM

Audio recorded to and played back from a tape machine can be thought of as going through three distinct processors: the record head, tape magnetisation, and the play head.

2.1. The Record Head

For an instantaneous input current $I(t)$, the magnetic field output of the record head is given as a function of distance along the tape (x), and depth into the tape (y). Using the Karlqvist medium field approximation, we find [7]:

$$H_x(x,y) = \frac{1}{\pi} H_0 \Big(\tan^{-1} \Big(\frac{(g/2)+x}{y} \Big) + \tan^{-1} \Big(\frac{(g/2)-x}{y} \Big) \Big) \quad (1)$$

$$H_y(x,y) = \frac{1}{2\pi} H_0 \ln \Big(\frac{((g/2)-x)^2 + y^2}{((g/2)+x)^2 + y^2} \Big) \quad (2)$$

where H_x and H_y are components of the magnetic field $\vec{H}$, g is the head gap, and H_0 is the deep gap field, given by:

$$H_0 = \frac{NEI}{g} \quad (3)$$

[1] https://www.waves.com/plugins/j37-tape

[2] https://www.uaudio.com/uad-plugins/plug-in-bundles/magnetic-tape-bundle.html

[3] https://u-he.com/products/satin/

where N is the number of turns of wire around the head, and E is the head efficiency which can be calculated by:

$$E = \frac{1}{1 + \frac{lA_g}{\mu_r g}\int_{core}\frac{d\vec{l}}{A(l)}} \tag{4}$$

where A_g is the gap area, μ_r is the core permeability relative to free space (μ_0), and $A(l)$ is the cross-sectional area at a point l along its circumferential length.

2.2. Tape Magnetisation

The magnetisation recorded to tape from a magnetic field can be described using a hysteresis loop, as follows [7]:

$$\vec{M}(x,y) = F_{Loop}(\vec{H}(x,y)) \tag{5}$$

where F_{Loop} is a generalized hysteresis function.

Using the Jiles-Atherton magnetisation model, the following differential equation describes magnetisation in some direction (M) as a function of the magnetic field in that direction (H) [9]:

$$\frac{dM}{dH} = \frac{(1-c)\delta_M(M_{an}-M)}{(1-c)\delta_S k - \alpha(M_{an}-M)} + c\frac{dM_{an}}{dH} \tag{6}$$

where c is the ratio of normal and anhysteric initial susceptibilities, k is a measure of the width of the hysteresis loop, α is a mean field parameter, representing inter-domain coupling, and δ_S and δ_M are given by:

$$\delta_S = \begin{cases} 1 & \text{if H is increasing} \\ -1 & \text{if H is decreasing} \end{cases} \tag{7}$$

$$\delta_M = \begin{cases} 1 & \text{if } \delta_S \text{ and } M_{an}-M \text{ have the same sign} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

M_{an} is the anisotropic magnetisation given by:

$$M_{an} = M_s L\Big(\frac{H+\alpha M}{a}\Big) \tag{9}$$

where M_s is the magnetisation saturation, a characterizes the shape of the anhysteric magnetisation and L is the Langevin function:

$$L(x) = \coth(x) - \frac{1}{x} \tag{10}$$

2.3. Play Head

2.3.1. Ideal Playback Voltage

The ideal playback voltage as a function of tape magnetisation at a point x along the tape is given by [7]:

$$V(x) = NWEv\mu_0 \int_{-\infty}^{\infty} dx' \int_{-\delta/2}^{\delta/2} dy' \vec{h}(x'+x, y') \cdot \frac{\vec{M}(x',y')}{dx} \tag{11}$$

where N is the number of turns of wire, W is the width of the playhead, E is the playhead efficiency, v is the tape speed, δ is the thickness of the tape, and μ_0 is the permeability of free space. Note that $V(x) = V(vt)$ for constant v. $\vec{h}(x,y)$ is defined as:

$$\vec{h}(x,y) \equiv \frac{\vec{H}(x,y)}{NIE} \tag{12}$$

where $\vec{H}(x,y)$ can be calculated by eqs. (1) and (2).

2.3.2. Loss Effects

There are several frequency-dependent loss effects associated with playback, described as follows [1]:

$$V(t) = V_0(t)[e^{-kd}]\Big[\frac{1-e^{-k\delta}}{k\delta}\Big]\Big[\frac{\sin(kg/2)}{kg/2}\Big] \tag{13}$$

for sinusoidal input $V_0(t)$, where k is the wave number, d is the distance between the tape and the playhead, g is the gap width of the play head, and again δ is the thickness of the tape. The wave number is given by:

$$k = \frac{2\pi f}{v} \tag{14}$$

where f is the frequency and v is the tape speed.

3. DIGITIZING THE SYSTEM

3.1. Record Head

For simplicity, let us assume,

$$\vec{H}(x,y,t) = \vec{H}(0,0,t) \tag{15}$$

In this case $H_y \equiv 0$, and $H_x \equiv H_0$. Thus,

$$H(t) = \frac{NEI(t)}{g} \tag{16}$$

or,

$$\hat{H}(n) = \frac{NE\hat{I}(n)}{g} \tag{17}$$

3.2. Hysteresis

Beginning from eq. (6), we can find the derivative of M w.r.t. time, as in [9]:

$$\frac{dM}{dt} = \frac{\frac{(1-c)\delta_M(M_s L(Q)-M)}{(1-c)\delta_S k - \alpha(M_s L(Q)-M)}\dot{H} + c\frac{M_s}{a}\dot{H}L'(Q)}{1 - c\alpha\frac{M_s}{a}L'(Q)} \tag{18}$$

where $Q = \frac{H+\alpha M}{a}$, and

$$L'(x) = \frac{1}{x^2} - \coth^2(x) + 1 \tag{19}$$

Note that eq. (18) can also be written in the general form for non-linear Ordinary Differential Equations:

$$\frac{dM}{dt} = f(t, M, \vec{u}) \tag{20}$$

where $\vec{u} = \begin{bmatrix} H \\ \dot{H} \end{bmatrix}$.

Using the trapezoidal rule for derivative approximation, we find:

$$\dot{\hat{H}}(n) = 2\frac{\hat{H}(n) - \hat{H}(n-1)}{T} - \dot{\hat{H}}(n-1) \tag{21}$$

We can use the Runge-Kutta 4th-order method [8] to find an explicit solution for $\hat{M}(n)$:

$$\begin{aligned} k_1 &= Tf\Big(n-1, \hat{M}(n-1), \hat{\tilde{u}}(n-1)\Big) \\ k_2 &= Tf\Big(n-\frac{1}{2}, \hat{M}(n-1)+\frac{k_1}{2}, \hat{\tilde{u}}\Big(n-\frac{1}{2}\Big)\Big) \\ k_3 &= Tf\Big(n-\frac{1}{2}, \hat{M}(n-1)+\frac{k_2}{2}, \hat{\tilde{u}}\Big(n-\frac{1}{2}\Big)\Big) \\ k_4 &= Tf\Big(n, \hat{M}(n-1)+k_3, \hat{\tilde{u}}(n)\Big) \\ \hat{M}(n) &= \hat{M}(n-1)+\frac{k_1}{6}+\frac{k_2}{3}+\frac{k_3}{3}+\frac{k_4}{6} \end{aligned} \tag{22}$$

We use linear interpolation to find the half-sample values used to calculate k_2 and k_3. Note that many audio DSP systems prefer lower-order implicit methods such as the trapezoidal rule to solve differential equations rather than a higher-order method like the Runge-Kutta method [8]. However in this case, it was found that the lower-order methods quickly became unstable for high-frequency input, particularly when the input is modulated by a bias signal (see Section 4.3).

3.2.1. Numerical Considerations

To account for rounding errors in the Langevin function for values close to zero, we use the following approximation about zero, as in [9]:

$$L(x) = \begin{cases} \coth(x) - \frac{1}{x} & \text{for } |x| > 10^{-4} \\ \frac{x}{3} & \text{otherwise} \end{cases} \tag{23}$$

$$L'(x) = \begin{cases} \frac{1}{x^2} - \coth^2(x) + 1 & \text{for } |x| > 10^{-4} \\ \frac{1}{3} & \text{otherwise} \end{cases} \tag{24}$$

Additionally, $\tanh(x)$, and by extension $\coth(x)$ is a rather computationally expensive operation. With this in mind, for real-time implementation, we approximate $\coth(x)$ as the reciprocal of a Gaussian continued fraction for $\tanh(x)$ [10], namely

$$\tanh(x) = \cfrac{x}{1+\cfrac{x^2}{3+\cfrac{x^2}{5+\cfrac{x^2}{7}}}} \tag{25}$$

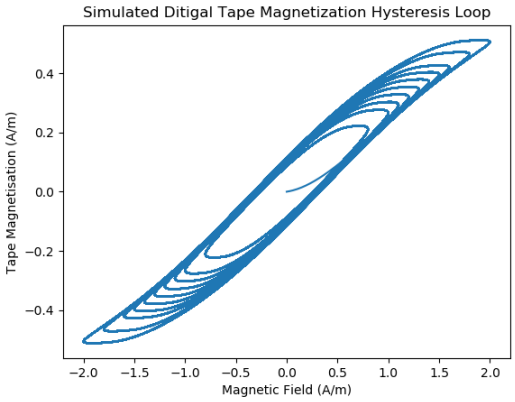


Figure 2: *Digitized Hysteresis Loop Simulation*

3.2.2. Simulation

The digitized hysteresis loop was implemented and tested offline in `Python`, using the constants M_s, a, α, k, and c from [11]. For a sinusoidal input signal with frequency 2kHz, and varying amplitude from 800 - 2000 Amperes per meter, fig. 2 shows the Magnetisation output.

3.3. Play Head

By combining eq. (11) with eqs. (12) and (16), we get:

$$V(t) = NWEv\mu_0 gM(t) \tag{26}$$

or,

$$\hat{V}(n) = NWEv\mu_0 g\hat{M}(n) \tag{27}$$

3.3.1. Loss Effects

In the real-time system, we model the playhead loss effects with an FIR filter, derived by taking the inverse DFT of the loss effects described in eq. (13). It is worth noting that as in eq. (14), the loss effects, and therefore the FIR filter are dependent on the tape speed.

The loss effects filter was implemented and tested offline in `Python` with tape-head spacing of 20 microns, head gap width of 5 microns, tape thickness of 35 microns, and tape speed of 15 ips. The following plot shows the results of the simulation, with a filter order of 100.

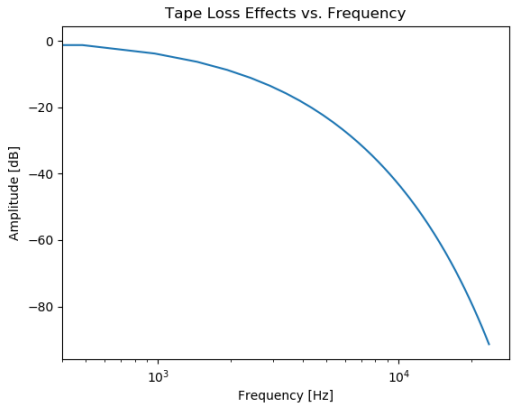


Figure 3: *Frequency Response of Playhead Loss Effects*

4. TAPE AND TAPE MACHINE PARAMETERS

In the following sections, we describe the implementation of a real-time model of a Sony TC-260 tape machine, while attempting to preserve generality so that the process can be repeated for any similar reel-to-reel tape machine.

4.1. Tape Parameters

A typical reel-to-reel tape machine such as the Sony TC-260 uses Ferric Oxide (γF_2O_3) magnetic tape. The following properties of the tape are necessary for the tape hysteresis process eq. (18):

- Magnetic Saturation (M_s): For Ferric Oxide tape the magnetic saturation is $3.5e5$ (A/m) [12].
- Hysteresis Loop Width (k): For soft materials, k can be approximated as the coercivity, H_c [13]. For Ferric Oxide, H_c is approximately 27 kA/m [12].
- Anhysteric magnetisataion (a): Knowing the coercivity and remnance magnetism of Ferric Oxide [12], we can calculate a = 22 kA/m by the method described in [13].
- Ratio of normal and hysteris initial susceptibilities (c): From [13], c = 1.7e-1.
- Mean field parameter (α): From [13], α = 1.6e-3.

4.2. Tape Machine Parameters

4.2.1. Record Head

To determine the magnetic field output of the record head using eq. (17), the following parameters are necessary:

- Input Current ($\hat{I}(n)$): For the Sony TC-260 the input current to the record head is approximately 0.1 mA peak-to-peak [14].
- Gap Width (g): The gap width for recording heads can range from 2.5 to 12 microns [1].
- Turns of wire (N): The number of turns of wire is typically on the order of 100 [7].
- Head Efficiency (E): The head efficiency is typically on the order of 0.1 [7].

These values result in a peak-to-peak magnetic field of approximately 5e5 A/m.

4.2.2. Play Head

Similar to the record head, the following parameters are needed to calculate the output voltage using eqs. (13) and (27) (note that values are only included here if notably different from the record head):

- Gap Width (g): The play head gap width ranges from 1.5 to 6 microns[1].
- Head Width (W): For the Sony TC-260, the play head width is 0.125 inches (note that this is the same as the width of one track on the quarter-inch tape used by the machine) [14].
- Tape Speed (v): The Sony TC-260 can run at 3.75 inches per second (ips), or 7.5 ips [14]. Note that many tape machines can run at 15 or 32 ips [1].
- Tape Thickness (δ): Typical tape that would be used with the TC-260 is on the order of 35 microns thick [14].
- Spacing (d): The spacing between the tape and the play head is highly variable between tape machines. For a typical tape machine spacing can be as high as 20 microns [1].

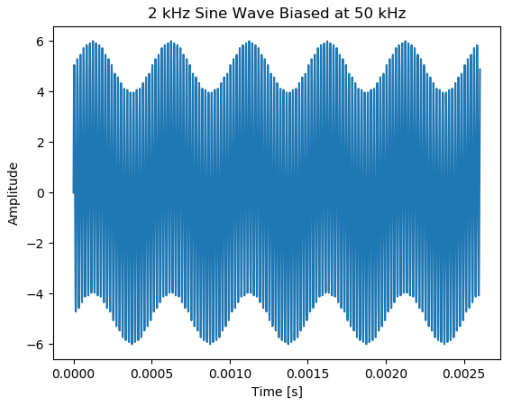


Figure 4: *Example of a biased signal*

4.3. Tape Bias

A typical analog recorder adds a high-frequency "bias" current to the signal to avoid the "deadzone" effect when the input signal crosses zero, as well as to linearize the output. The input current to the record head can be given by [6]:

$$\hat{I}_{head}(n) = \hat{I}_{in}(n) + B\cos(2\pi f_{bias} nT) \tag{28}$$

Where the amplitude of the bias current B is usually about one order of magnitude larger than the input, and the bias frequency f_{bias} is well above the audible range. Figure 4 shows a unit-amplitude, 2 kHz sine wave biased by a 50 kHz sine wave with amplitude 5. To recover the correct output signal, tape machines use a lowpass filter, with a cutoff frequency well below the bias frequency, though still above the audible range [1].

For the Sony TC-260, the bias frequency is 55 kHz, with a gain of 5 relative to the input signal. The lowpass filter used to recover the audible signal has a cutoff at 24 kHz, though note that due to the frequency response of the playhead loss effects, the effects of this filter may be essentially neglible to the real time system. [14]

4.4. Wow and Flutter

Each tape machine has characteristic timing imperfections known as “wow” and/or “flutter.” These imperfections are caused by minor changes in speed from the motors driving the tape reels, and can cause fluctuations in the pitch of the output signal. To characterize these timing imperfections, we use a method similar to [3]: We recorded a pulse train of 1000 pulses through a TC-260, then recorded the pulses back from the tape. Figure 5 shows a section of a superimposed plot of the original pulse train against the pulse train recorded from the tape machine. From this data, we were able to generate a periodic function that accurately models the timing imperfections of the TC-260. The process was performed at both 7.5 ips and 3.75 ips. In the real-time system, the timing imperfection model is used to inform a modulating delay line, to achieve the signature "wow" effect of an analog tape machine.

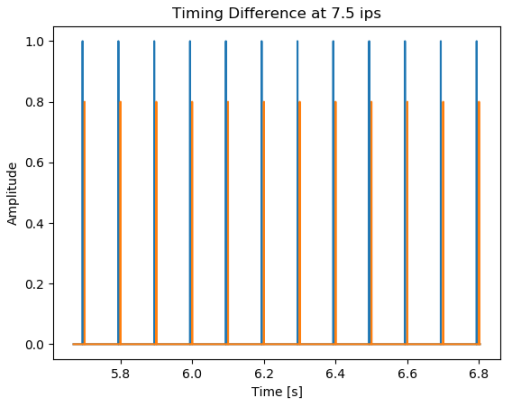


Figure 5: *Input pulse train superimposed with pulse train recorded from TC-260*

Record Head Constants	
Turns of wire (N)	100
Head Efficiency (E)	0.1
Record Head Gap (g)	6 (microns)
Tape Constants	
Magnetic Saturation (M_s)	3.5e5 (A/m)
Hysteresis Loop Width (k)	27 (kA/m)
Anhysteric Magnetisation (a)	22 (kA/m)
Ratio of magnetic susceptibilities (c)	1.7e-1
Play Head Constants	
Play Head Width (W)	0.125 (in)

Figure 6: *Constant values for model implementation*

5. REAL-TIME IMPLEMENTATION

We implemented our physical model of the Sony TC-260 as a VST audio plugin using the JUCE framework. Figure 7 shows the signal flow of the system in detail. We allow the user to control parameters in real-time including the tape speed, bias gain, gap width, tape thickness, tape spacing, and flutter depth.

Bias modulation is implemeted using eq. (28), where bias gain B and bias frequency f_{bias} are controlled by the user.

The record head transfer function calculates the record head magnetic field H from the input current I, and is implemented using eq. (17), with constant values shown in fig. 6.

The hysteresis process calculates the tape magnetisation M from the record head magnetic field H, and is implementated using the Runge-Kutta method described in eq. (22), with constant value defined in fig. 6.

The flutter process is implemented using a modulated delay line as described in section 4.4, with user controlled modulation depth.

The play head transfer calculates the play head voltage V from the tape magnetisation M, using eq. (27) and the loss effects filter described in section 3.3.1. The gap width g, tape speed v, tape thickness δ, and spacing d are controlled by the user, and other constant values are shown in fig. 6.

C/C++ code for the full real-time implementation is open-source and is available on GitHub.[4]

5.1. Oversampling

If no oversampling is used, the system will be unstable for input signal at the Nyquist frequency, due to limitations of the trapezoid rule derivate approximation used in eq. (21). To avoid this instability, early versions of the real-time implementation used a lowpass filter with cutoff frequency just below Nyquist. However, due to aliasing caused by the nonlinearity of the tape hysteresis model, oversampling is necessary to mitigate aliasing artifacts [8]. Further, the system must be able to faithfully recreate not only the frequencies in the audible range but the bias frequency as well. Since the TC-260 uses a bias frequency of 55 kHz [14] and the minimum standard audio sampling rate is 44.1 kHz, a minimum oversampling factor of 3x is required. However, since the biased signal is then fed into the hystersis model, even more oversampling is required to avoid aliasing. With these considerations in mind, our system uses an oversampling factor of 16x.

5.2. Results

In subjective testing, our physical model sounds quite convincing, with warm, tape-like distortion, and realistic sounding flutter. The high-frequency loss and low-frequency "head bump" change correctly at different tape speeds, and are approximately within the frequency response specifications of the TC-260 service manual [14]. When the input to the plugin is silent, the hysteresis processing of the bias signal produces a very accurate "tape hiss" sound. The distortion and frequency response characteristics of our model are subjectively very close when compared to the output of an actual TC-260, though not nearly close enough to "fool" the listener. Additionally, as the bias gain is lowered, the "deadzone" effect appears exactly as expected [6]. The largest difference between the model and the physical machine is the subtle electrical and mechanical noises and dropouts present in the physical machine, presumably caused by the age and wear-and-tear of the machine, which we did not attempt to characterize in our model. Figure 8 shows the results of tests performed on the real-time system, including an example of the "deadzone" effect, and the timing irregularities or "flutter". Figure 9 shows a comparison of hysteresis characteristics between the real-time software model and a physical Sony TC-260 tape machine. Note that some differences between the two hysteresis loops may be due to the circuitry of the tape machine that we did not attempt to model in the real-time system. Audio examples from the real-time system can be found online.[5]

5.3. Evaluation

While there is an audible difference between the real-time software model and a physical Sony TC-260, the most fundamental aspects of the tape machine sound including tape saturation, tape

[4] https://github.com/jatinchowdhury18/AnalogTapeModel

[5] https://ccrma.stanford.edu/~jatin/420/tape/

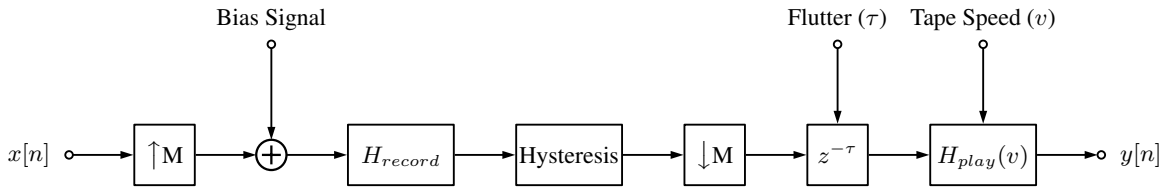
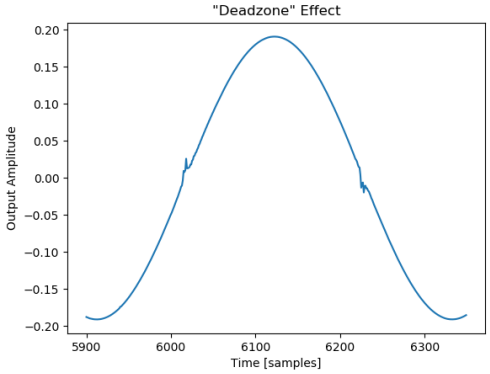


Figure 7: *Flowchart of realtime system: M is the oversampling factor, H_{record} is the transfer function of the record head, and H_{play} is the play head transfer function including loss effects, and a de-biasing filter.*



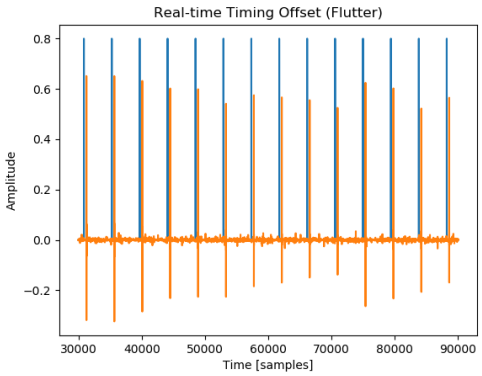
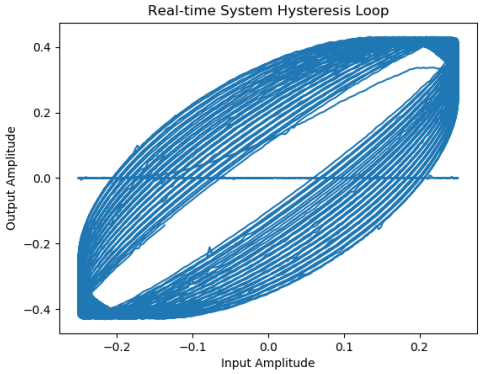


Figure 8: *Testing results for real-time system: sine wave output with no biasing (above), input vs. output pulse train comparison (below).*



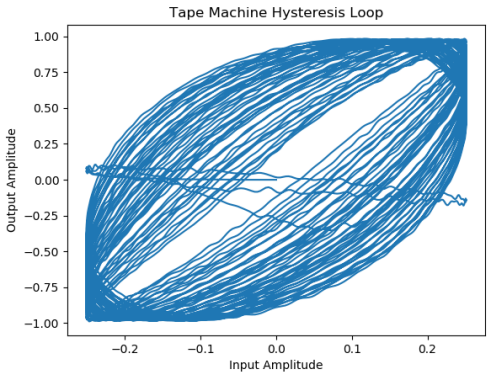


Figure 9: *Test results comparing real-time system to Sony TC-260 physical unit: hysteresis loop for real-time system (above), hysteresis loop for TC-260 (below).*

hiss, flutter/wow, and frequency response, are clearly audible and sound very accurate. The main distinctions between the two systems can be attributed to the tape machine circuitry (in particular the TC-260 contains two shelving filters), as well as mechanical wear of the system, both elements that were not considered in our model.

In our opinion, the strongest proof of the efficacy of our model is that the model responds accurately to the adjustement of model parameters. In particular, the hysteresis process reacted correctly to changes in input gain (saturating for overdriven input, or fading into tape hiss for underdriven input), as well as bias gain (saturation for overbiasing, or "deadzone" effect for underbiasing). Additionally, adjusting the loss effect parameters correctly demonstrated known tape machine phenomena including head "bump" (a resonance at the wavelength of the play head gap width), and spacing loss (filtering due to the spacing between the the play head and tape). The reader is invited to download the plugin (available with the source code) and evaluate the model for themselves. In conclusion, we believe that our model successfully approximates the physical tape recording process, however for those wishing to model a full tape machine, we suggest using this model in combination with a model of the tape machine's circuits.

6. FUTURE IMPROVEMENTS

6.1. Spatial Magnetic Effects

The most obvious improvement to be made for the physical model is the inclusion of spatial effects of the tape. In particular, the approximations made in eq. (15), negate any effects caused by mag-

netisation along the longitudinal length of the tape, and into the depth of the tape. Including spatial effects would involve deriving digital analogues for eqs. (1), (2) and (11), and re-deriving eq. (22) to take an 2-dimensional magnetic field input at every timestep, rather than the zero-dimensional input it currently takes. This change would greatly increase the computational complexity of the system. At an oversampling rate of 16x, using just 100 spatial samples would be 1600x more computationally complex than the current system.

7. ACKNOWLEDGEMENTS

Many thanks to Julius Smith for guidance and support, and thanks to Irene Abosch for kindly donating her Sony TC-260.

8. REFERENCES

[1] Jay Kadis, *The Science of Sound Recording*, Focal Press, Waltham, MA, 1 edition, 2012.

[2] Brian Eno, *A Year with Swollen Appendices*, Faber and Faber, 1996.

[3] Steinunn Arnardottir, Jonathan S. Abel, and Julius O. Smith III, "A digital model of the echoplex tape delay," in *Audio Engineering Society Convention 125*, 10 2008.

[4] Vesa Valimaki, Stefan Bilbao, Julius Smith, J S. Abel, Jyri Pakarinen, and D Berners, *Virtual Analog Effects*, pp. 473 – 522, 03 2011.

[5] Vesa Valimaki, Sira Gonzalez, Ossi Kimmelma, and Jukka Parviainen, "Digital audio antiquing - signal processing methods for imitating the sound quality of historical recordings," *AES: Journal of the Audio Engineering Society*, vol. 56, pp. 115–139, 03 2008.

[6] Marvin Camras, *Magnetic Recording Handbook*, Van Nostrand Reinhold Co., New York, NY, USA, 1987.

[7] H. N. Bertram, *Theory of Magnetic Recording*, 04 1994.

[8] D.T. Yeh, *Digital Implementation of Musical Distortion Circuits by Analysis and Simulation*, Ph.D. thesis, Stanford Univeristy, 06 2009.

[9] Martin Holters and Udo Zolzer, "Circuit simulation with inductors and transformers based on the Jiles-Atherton model of magnetization," 09 2016.

[10] H. S. Wall, *Analytic Theory of Continued Fractions*, Chelsea, New York, 1948.

[11] D. C. Jiles and D. L. Atherton, "Theory of ferromagnetic hysteresis," *Journal of Magnetism and Magnetic Materials*, vol. 61, pp. 48–60, 09 1986.

[12] D. Jiles, *Introduction to Magnetism and Magnetic Materials*, CRC Press, 2015.

[13] D. C. Jiles, J. B. Thoelke, and M. K. Devine, "Numerical determination of hysteresis parameters for the modeling of magnetic properties using the theory of ferromagnetic hysteresis," *IEEE Transactions on Magnetics*, vol. 28, pp. 27–35, 01 1992.

[14] Sony Corporation, *Sony TC-260 Service Manual*, 1965.

IMPROVED REVERBERATION TIME CONTROL FOR FEEDBACK DELAY NETWORKS

Karolina Prawda, Vesa Välimäki [*]

Acoustics Lab, Dept. of Signal Processing and Acoustics
Aalto University
Espoo, Finland
karolina.prawda@aalto.fi

Sebastian J. Schlecht

International Audio Laboratories[†]
Erlangen, Germany

sebastian.schlecht@audiolabs-erlangen.de

ABSTRACT

Artificial reverberation algorithms generally imitate the frequency-dependent decay of sound in a room quite inaccurately. Previous research suggests that a 5% error in the reverberation time (T60) can be audible. In this work, we propose to use an accurate graphic equalizer as the attenuation filter in a Feedback Delay Network reverberator. We use a modified octave graphic equalizer with a cascade structure and insert a high-shelf filter to control the gain at the high end of the audio range. One such equalizer is placed at the end of each delay line of the Feedback Delay Network. The gains of the equalizer are optimized using a new weighting function that acknowledges nonlinear error propagation from filter magnitude response to reverberation time values. Our experiments show that in real-world cases, the target T60 curve can be reproduced in a perceptually accurate manner at standard octave center frequencies. However, for an extreme test case in which the T60 varies dramatically between neighboring octave bands, the error still exceeds the limit of the just noticeable difference but is smaller than that obtained with previous methods. This work leads to more realistic artificial reverberation.

1. INTRODUCTION

Reverberation time is one of the most important parameter used to determine the acoustic quality of physical spaces. Multiple studies have been conducted to evaluate the accuracy of perceiving the changes in the reverberation time for various types of signals. Seraphim [1] determined the just noticeable difference (JND) of the reverberation time to be 5%. However, more recent studies showed that for bandlimited noise the difference is perceivable only when it exceeds 24% of the target value [2], compared to 5% to 7% for impulse signals and 3% to 9% for reverberated speech [3]. The JND of 5% is used in this work to comply with the current ISO standard [4].

Various algorithms are used to produce artificial reverberation, with the Feedback Delay Network (FDN) being currently among the most popular ones [5–7]. The first objective in designing an FDN is to make it lossless. Attenuation filters are introduced to achieve target energy decay. Over time, various types of attenuation filters have been proposed. Initially, a first-order lowpass infinite impulse response (IIR) filter was used because of its low computational cost and ease of design [5, 8]. Later, biquadratic filters were introduced allowing to control the decay time in three independent frequency bands with adjustable crossover frequencies [9]. In [10], a 13th-order filter comprising single bandpass filters as described in [11] and a second-order Butterworth bandpass filter was proposed.

The most advanced method of controlling decay time in artificial reverberation in several frequency bands uses a proportional graphic equalizer [12]. This method was recently improved by Schlecht and Habets, who determined the filter parameters by solving the nonlinear least-squares problem with linear constraints approximating the target reverberation-time response directly [13]. This approach offered very accurate control of decay time and ensured that the FDN remained stable. However, the computation of filter parameters proved to be inefficient, especially in real-time applications [13].

The present work proposes an accurate method to control reverberation time in octave bands utilizing attenuation filters that produce small approximation errors. It is an extension to previous work done by Schlecht and Habets [13]. This paper introduces a novel graphic equalizer (GEQ) with an additional high-shelf filter as an attenuation filter inside the FDN and presents a weighted-gain optimization method that acknowledges nonlinear error propagation from filter magnitude response to reverberation time values. The paper is organized as follows. Section 2 discusses attenuation filters and proposes a new design as well as a weighted-gain optimization method. Section 3 presents case studies in which we test the proposed method and compare the proposed design to other solutions in terms of the approximation error as well as computational cost. Section 4 summarizes the work presented in the paper, gives conclusions about the results, and proposes ideas for future research.

2. ATTENUATION FILTER

An FDN is a comb filter structure with multiple delay lines interconnected by a feedback matrix [5]. When designing FDNs, the first step is to make it lossless, ensuring that the energy will not decay for any possible type of delay [7]. The frequency-dependent reverberation time can then be implemented by inserting an attenuation filter at the beginning or at the end of each delay line. As the filters do not work in relation to one another and are only dependent on their corresponding delay line, they can be analyzed separately. Instead of the FDN, we can analyze the simpler single-delay-line absorptive feedback comb filter, i.e.,

$$H(z) = \frac{1}{1 - A(z)z^{-L}}, \tag{1}$$

[*] This work was supported by the "Nordic Sound and Music Computing Network—NordicSMC", NordForsk project number 86892.

[†] The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institut für Integrierte Schaltungen IIS.

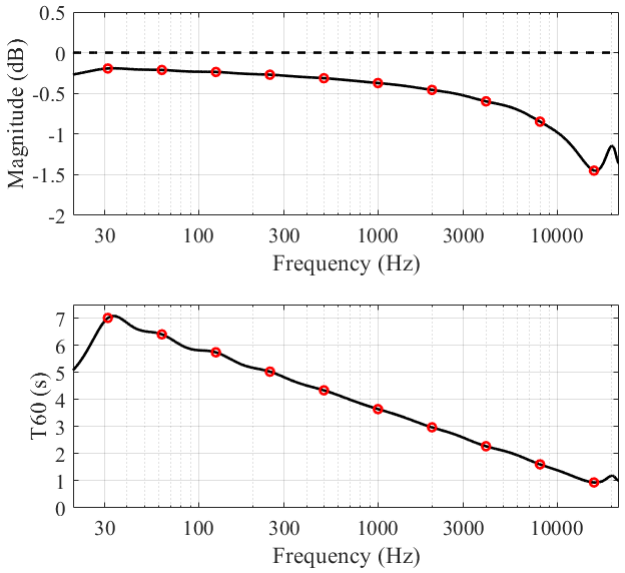


Figure 1: *Relationship between (top) gain-per-sample and (bottom) resulting reverberation time for the delay-line length of* $L = 1000$ *samples. Red markers indicate the octave bands. The horizontal dashed line in the top figure is the unit-gain limit, reaching which would lead to an infinite reverberation time.*

where L is the delay length in samples and $A(z)$ is the transfer function of the attenuation filter. For further analysis of the magnitude in dB, the attenuation filter is given by

$$A_{\text{dB}}(\omega) = 20 \log_{10} |A(e^{j\omega})|, \tag{2}$$

where $\omega = 2\pi f / f_{\text{s}}$ is the normalized frequency, f is the frequency in Hz, and f_{s} is the sampling rate in Hz. Such a filter should be designed to approximate the gain-per-sample necessary to obtain the desired frequency-dependent reverberation time, $T_{60}(\omega)$. This gain in dB is expressed as

$$\gamma_{\text{dB}}(\omega) = \frac{-60}{f_{\text{s}} T_{60}(\omega)}, \tag{3}$$

where $T_{60}(\omega)$ is in seconds. The gain is dependent on the delay-line length, growing proportionally to the number of delay samples L. As a result, longer delay lines decay faster than short ones. To obtain the target gain and, as a consequence, the desired frequency-dependent reverberation time, the following condition should be met:

$$A_{\text{dB}}(\omega) = L \gamma_{\text{dB}}(\omega). \tag{4}$$

Fig. 1 illustrates the relation between the gain-per-sample values of the single-delay-line absorptive feedback comb filter as presented in Eq. (1) with the attenuation filter designed according to Eq. (2-4) and the resulting $T_{60}(\omega)$ values. The delay-line length was set to $L = 1000$ samples and the target reverberation time was set to decrease linearly in octave bands from 7 s at 31.5 Hz to 1 s at 16 kHz.

2.1. Graphic equalizer design

The attenuation filter in the present work is realized with the cascade GEQ, composed of second-order IIR peak-notch filters proposed by Orfanidis [14] and designed using a method proposed by

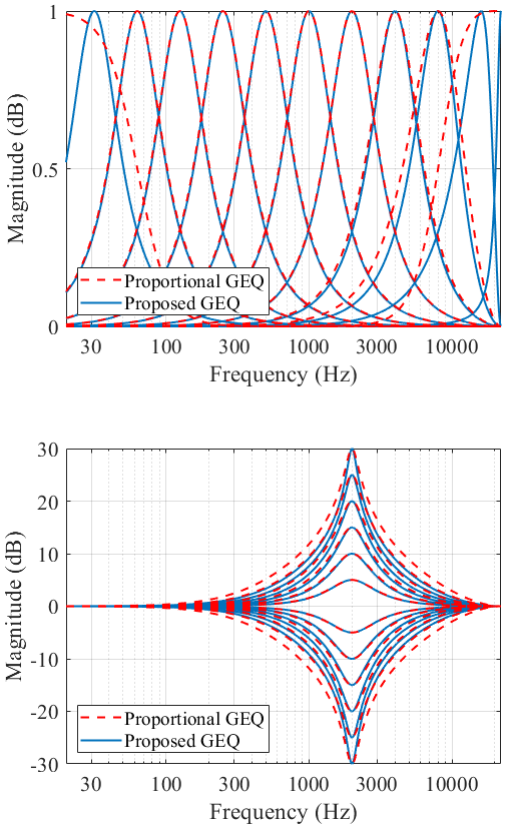


Figure 2: *Comparison of magnitude responses of the proposed GEQ with a high-shelf filter to the proportional graphic equalizer used in [13]. Top: magnitude responses for individual biquadratic filters and a prototype gain of 1 dB for ten frequency bands. Bottom: Single-band proportional gain behavior of the magnitude response.*

Välimäki and Liski [15], where extra frequency points are added and one iteration step is used to obtain a highly accurate magnitude response. The GEQ is also composed of only peak-notch filters, as opposed to the usual approach in which shelf filters are applied to the highest and lowest frequency bands. Using only peak-notch filters improves the symmetry of the magnitude responses of individual filters and the accuracy of the equalizer. This results in the proposed design producing approximation errors of less than ± 1 dB for command gains within a range of -12 to $+12$ dB [15]. The top plot of Fig. 2 depicts the magnitude responses of the individual biquadratic filters of the proposed GEQ, with an additional high-shelf filter as described in Sec. 2.2., compared to the proportional graphic equalizer from [13]. The approach adopted in the present paper displays more symmetrical magnitude responses even for high frequencies. The bottom plot of Fig. 2 presents the magnitude response of peak-notch filters for command gains between -30 and $+30$ dB.

The transfer function of a GEQ with M bands is given by

$$H(e^{j\omega}) = G_0 \prod_{m=1}^{M} H_{\text{m}}(e^{j\omega}), \tag{5}$$

where G_0 is the overall broadband gain factor and $H_{\text{m}}(e^{j\omega})$ are

the frequency responses of equalizing filters ($m = 1, 2, 3..., M$). The corresponding response in dB can be written as

$$H_{\text{dB}}(e^{j\omega}) = g_0 + \sum_{m=1}^{M} H_{\text{dB,m}}(e^{j\omega}). \tag{6}$$

For the accurate approximation of the reverberation time T_{60} over a broad frequency range, the command gains are defined for ten octave bands, having center frequencies ranging from 31.5 Hz to 16 kHz. This results, however, in the magnitude approaching 0 dB quite dramatically outside the considered frequency range. The reverberation time approximated for the octave bands below 31.5 Hz and above 16 kHz can appear to be very long, which may affect whole decay, preventing it from ever reaching -60 dB. To avoid this situation, we propose that the command gains be first shifted up to decrease their distance from zero and after the gain optimization, the entire magnitude response be scaled down by the same amount as for scaling up. This is depicted in the top and middle panes of Fig. 3. This yields the following changes to Eq. (6):

$$\widetilde{H}_{\text{dB}}(e^{j\omega}) = g_0 + \sum_{m=1}^{M} (H_{\text{dB,m}}(e^{j\omega}) - \frac{g_0}{M}). \tag{7}$$

In this way, the rise in magnitude at frequencies below 31.5 Hz and above 16 kHz are less steep. The shifting and scaling value can be set to the median of all command gains, as suggested in [16]. This also smooths the frequency response, causing very little ripple, as seen in the filter response comparison with and without scaling in Fig. 4.

2.2. High-shelf filter

In physical room acoustics, the decay time at high frequencies is usually shorter than at low frequencies, thus making the corresponding command gains considerably lower at high frequencies. Therefore, the operation of scaling and shifting the gains by the median may not be sufficient to prevent the magnitude from quickly approaching large values for frequencies above 16 kHz. For this reason, we use a first-order high-shelf filter implemented as suggested in [17] to equalize the problematic high frequencies. The gain of the shelf filter is set to the gain of the highest considered octave band, and the crossover frequency is set to 20.2 kHz. The latter value was experimentally found to introduce the smallest error in the reverberation time at 16 kHz.

The shelf filter introduces considerable ripple in the equalizer response above the frequency range of interest, but since it occurs at very high frequencies and the resulting reverberation time is much shorter than at lower frequencies, it is assumed to be inaudible. The response of the GEQ with the shelf filter is shown in the bottom pane of Fig. 3.

2.3. Filter-gain optimization

A common flaw in graphic equalizers is the interaction occurring between neighboring peak-notch filters, causing the response of each filter leak to other center frequencies [15–17]. A K-by-N interaction matrix $\mathbf{B}$ that shows this effect and stores the normalized amplitude response in dB of all M filters at K control frequency points is given by:

$$B_{k,m} = H_{\text{dB},m}(e^{j\omega_k})/g_{\text{p},m}, \tag{8}$$

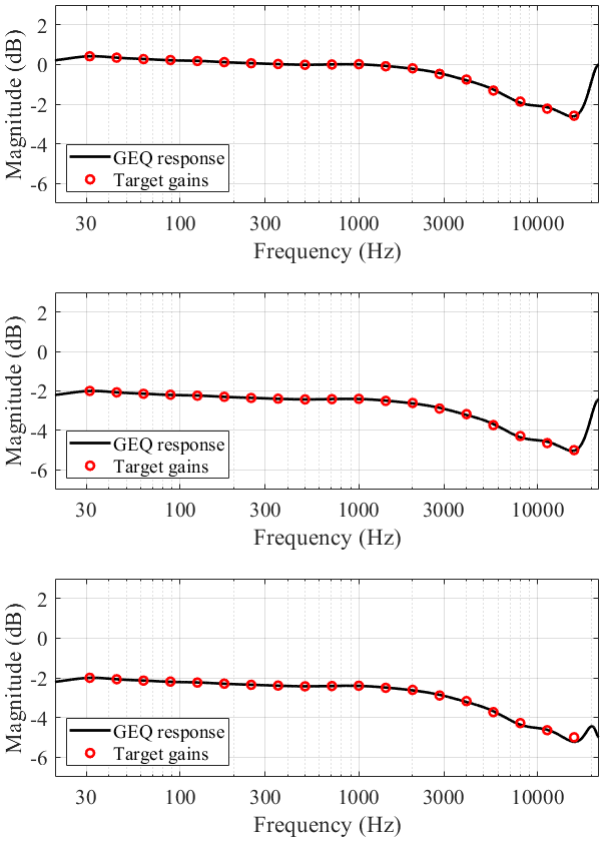


Figure 3: *Stages of obtaining the final frequency response of the GEQ for a delay length of 100 ms. (Top) Gains shifted up by their median value. (Middle) Gains scaled by a constant value. (Bottom) A high-shelf filter inserted to attenuate frequencies above 16 kHz.*

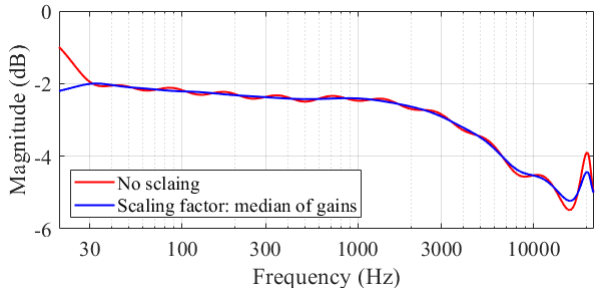


Figure 4: *Frequency response of the GEQ for a delay length of 100 ms, shifted and scaled by the median of gains compared to the response without scaling.*

where $k = 1, 2, ...K$ are control frequency points, $m = 1, 2, ...M$ are filter indices, and $\mathbf{g}_{\text{p}} = \left[g_{\text{p},1}, g_{\text{p},2}, ..., g_{\text{p,m}},\right]^T$, where $(.)^T$ denotes the transpose, is the vector of prototype dB gains common to all equalizing filters. The interaction matrix of the proposed GEQ for $K = 100$ and $N = 11$ is shown in Fig. 5. As a consequence of leakage, the magnitude response of the equalizer depends on the values stored in the interaction matrix. Considering that the GEQ is used as the attenuation filter in the FDN, Eq. (4) can now be

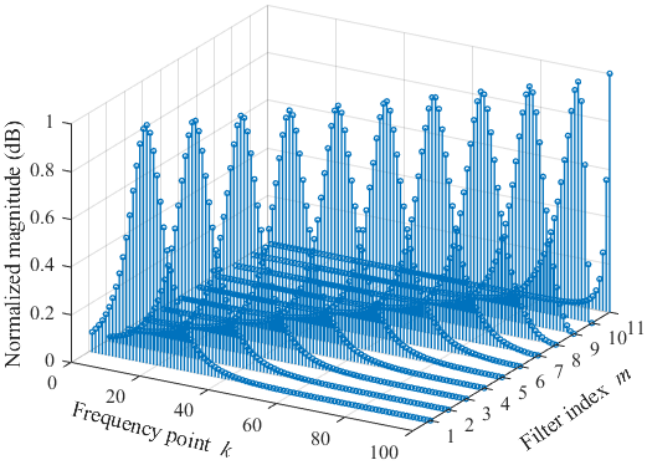


Figure 5: *A* 100*-by-*11 *interaction matrix of the proposed GEQ that stores the normalized amplitude of each filter that leaks to neighboring frequency points.*

expressed as

$$A_{\text{dB}}(\boldsymbol{\omega}) = \mathbf{B}\mathbf{g}_{\text{p}}, \tag{9}$$

where $\boldsymbol{\omega}$ is a $K \times 1$ vector of control frequencies.

2.4. Weighted-gain optimization

The GEQ used in the present work approximates the command gains strictly within 1 dB. However, in some instances of the reverberation time changing dramatically for neighboring octave bands, the differences in the command gains may be too high for the resulting filter magnitude response to follow without producing much error. Therefore, a method for gain optimization is introduced.

The approximation of the target magnitude response can be done on the dB scale by minimizing the error norm based on Eq. (4):

$$\|A_{\text{dB}}(\boldsymbol{\omega}) - L\gamma_{\text{dB}}(\boldsymbol{\omega})\|_2^2. \tag{10}$$

This approach assumes that the error propagates to the resulting reverberation time linearly. In reality, the reverberation time is affected in a nonlinear fashion: a small error in the filter magnitude response, when the attenuation is weak and the gain close to 0 dB, causes much greater changes in the resulting reverberation time than the same error, when the attenuation is strong and the gain much smaller than 0 dB [13].

This problem can be overcome by directly minimizing the squared error in the resulting reverberation time, as suggested earlier in [13]:

$$E = \left\| \frac{1}{A_{\text{dB}}(\boldsymbol{\omega})} - \frac{1}{L\gamma_{\text{dB}}(\boldsymbol{\omega})} \right\|_2^2. \tag{11}$$

Alternatively, we can minimize the relative error between the filter and the target reverberation time:

$$\widetilde{E} = \left\| 1 - \frac{A_{\text{dB}}(\boldsymbol{\omega})}{L\gamma_{\text{dB}}(\boldsymbol{\omega})} \right\|_2^2. \tag{12}$$

When the weighting matrix $\mathbf{W}$ is defined as

$$\mathbf{W} = \text{diag}\left(\frac{1}{L\gamma_{\text{dB}}(\boldsymbol{\omega})} \right), \tag{13}$$

the relative error from Eq. (12) becomes

$$\widetilde{E} = \left\| 1 - \mathbf{W}A_{\text{dB}}(\boldsymbol{\omega}) \right\|_2^2, \tag{14}$$

where the role of the weighting matrix $\mathbf{W}$ is to mimic the nonlinear behavior with a linear approximation in the sense of emphasizing the approximation error occurring close to 0 dB.

Error minimization was also performed by solving the linear problem using the Taylor-series approximation presented in [18]. However, since the method operates on a linear scale, as opposed to the suggested design which operates on the dB scale, no relevant improvement was observed. Therefore the method utilizing the Taylor-series approximation was not implemented further in the present work.

3. EVALUATION

The present work proposes to perform reverberation time approximation using a GEQ with weighted-gain optimization that minimizes the relative error between filter response and target reverberation time values. In order to evaluate that algorithm, two case studies were conducted. The first was aimed to reproduce the reverberation time of Promenadi Hall, a multipurpose hall located in Promenadikeskus in Pori. The second was conducted using a predefined reverberation time that differs considerably between neighboring octave bands to reveal potential weak spots of the algorithm and provide a valid comparison to the previous work. For both cases, the algorithm was tested with three lengths of delay lines: 10 ms, 50 ms, and 100 ms. The performance of the proposed algorithm was compared to the previous method of reverberation-time control in FDN presented in [13]. The computational cost measured in the number of operations per output sample with relation to other graphic equalizers was also examined.

3.1. Promenadi Hall

In the first case, the aim was to approximate the reverberation time of an existing architectural object. The target values were defined for octave bands and are presented in Table 1. The command gains for the GEQ were calculated based on these values. Fig. 6 compares the target magnitude response needed to obtain the desired reverberation time and the response of the GEQ for the delay-line lengths of 10, 50, and 100 ms.

The target magnitude response is followed by the response of the filter very accurately in every octave band it was specified in. The magnitude response below 31.5 Hz approaches the median value for the command gains, which was set as the equalizer's broadband gain. The only visible ripple of 0.02 dB, 0.11 dB and 0.23 dB for delay-line lengths of 10 ms, 50 ms and 100 ms, respectively, occur at very high frequencies, above 16 kHz, and is caused by the high-shelf filter.

The resulting reverberation time was calculated based on the magnitude response of the GEQ by converting it to dB using Eq. (2), and then using the condition from Eq. (4) to obtain the gain-per-sample in dB. The values of $T_{60}(\omega)$ were acquired based on Eq. (3) and are depicted in the top plot of Fig. 7 together with target values from Promenadi Hall. The obtained reverberation follows the behavior of the filter response, approximating the desired values very closely not only in the octave frequencies, but also in the entire frequency range. Although the values were calculated for three different delay-line lengths, the results do not vary visibly from each

Table 1: *Reverberation time values and error percentage for octave frequencies for Promenadi Hall. DL stands for delay length.*

Center frequency	31.5 Hz	63 Hz	125 Hz	250 Hz	500 Hz	1 kHz	2 kHz	4 kHz	8 kHz	16 kHz
Reverberation time	3.00 s	2.80 s	2.68 s	2.55 s	2.47 s	2.50 s	2.30 s	1.89 s	1.40 s	1.20 s
Error for DL of 10 ms	0.12%	0.03%	0.09%	0.05%	0.02%	0.35%	0.79%	1.00%	1.51%	4.43%
Error for DL of 50 ms	0.12%	0.03%	0.09%	0.05%	0.02%	0.35%	0.79%	1.00%	1.53%	4.44%
Error for DL of 100 ms	0.12%	0.03%	0.09%	0.05%	0.02%	0.35%	0.79%	0.99%	1.56%	4.50%

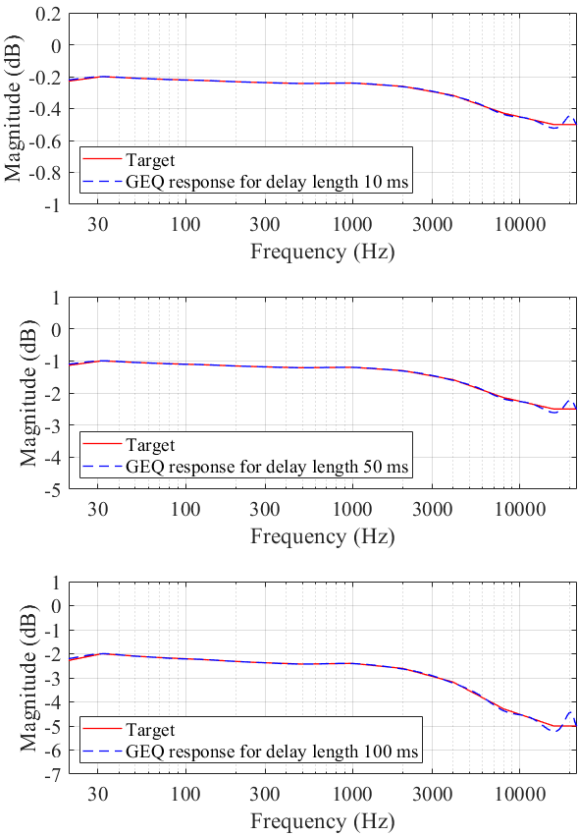


Figure 6: *Target magnitude response and response of the GEQ with first-order high-shelf filter for the case of Promenadi Hall in Pori.*

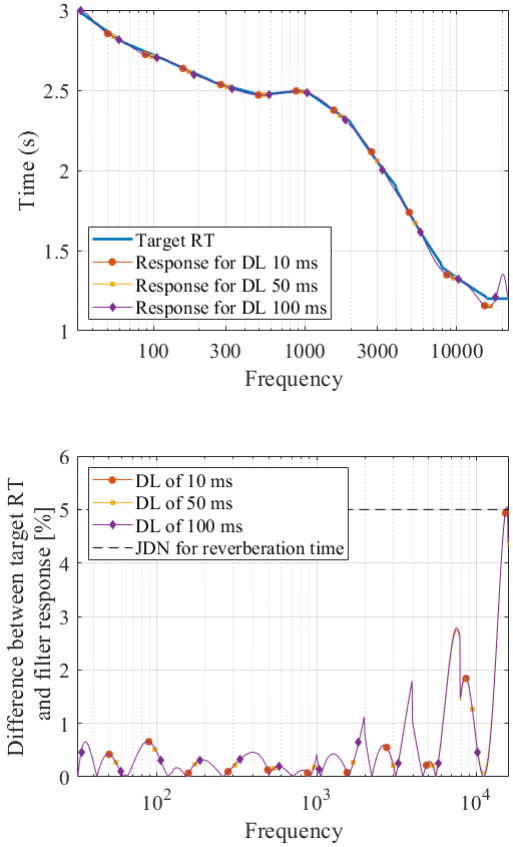


Figure 7: *(Top) Target reverberation time from Promenadi Hall and the values approximated by the GEQ. (Bottom) The difference expressed as percent of obtained reverberation time deviation from target values. RT is reverberation time.*

other, with the biggest difference between them reaching 0.07%. This proves that the proposed method works well regardless of the delay chosen when designing the FDN. This is also confirmed by the error values shown in Table 1, none of which exceed 5%, making the difference unnoticeable. Further evidence for the method's efficiency to accurately approximate reverberation time is in the bottom plot of Fig. 7, which shows the difference between the target and the obtained reverberation time for the whole frequency range.

The results obtained with the GEQ introduce deviations no bigger than 5% from the target value and therefore we refrained from trying to minimize the error.

3.2. Artificial extreme case

The second case was tested with predefined reverberation time values, which were aimed at being similar to those in [13]. Although in [13] the reverberation time values for octave bands were generated randomly, we decided to specify them manually in order to ensure the same tendency, which provides a good reference point for comparison of the two methods and is able to reveal shortcomings of the proposed design. The values of the reverberation time for octave frequencies are shown in Table 2.

The filters' magnitude responses obtained based on the desired decay are presented in Fig. 8. The target response is generally followed accurately. However, large differences in the reverberation between 2 kHz and 4 kHz cause a slight overshoot in magnitude for frequencies between 1 kHz and 2 kHz, which at its highest point

Table 2: *Reverberation time and error on octave frequencies for the artificial extreme case. Errors exceeding JND of 5.0% are highlighted.*

Center frequency	31.5 Hz	63 Hz	125 Hz	250 Hz	500 Hz	1 kHz	2 kHz	4 kHz	8 kHz	16 kHz
Reverberation time	1.00 s	1.00 s	1.00 s	1.00 s	1.00 s	3.00 s	3.00 s	0.25 s	1.00 s	1.00 s
Error for DL of 10 ms	0.12%	0.09%	0.17%	0.07%	3.66%	**12.59%**	**11.47%**	0.38%	2.96%	**5.80%**
Error for DL of 50 ms	0.13%	0.07%	0.22%	0.12%	3.11%	**8.79%**	**15.59%**	2.62%	**5.42%**	4.42%
Errorfor DL of 100 ms	0.15%	0.02%	0.39%	0.50%	1.52%	0.35%	**24.41%**	**6.03%**	**11.12%**	0.77%

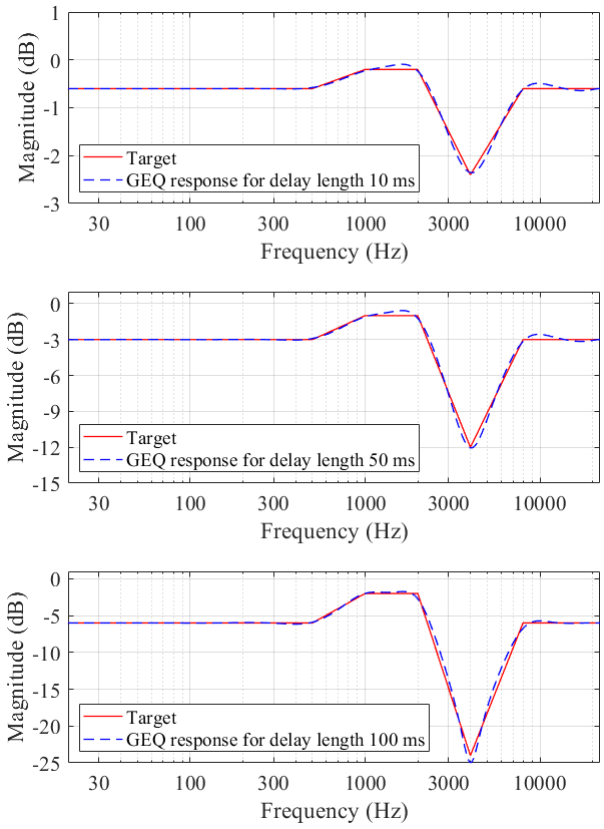


Figure 8: *Target magnitude response and response of the GEQ for three different delay-line lengths for the artificial extreme case.*

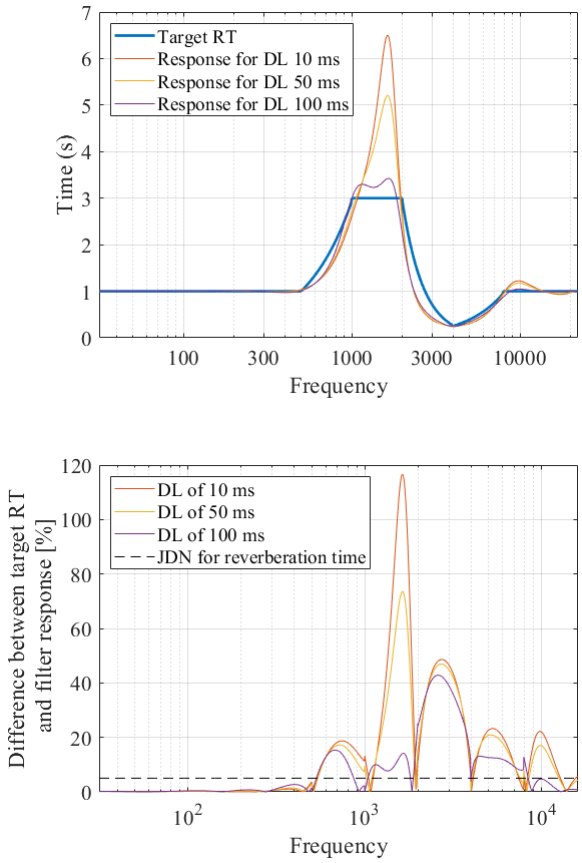


Figure 9: *(Top) Target reverberation time and reverberation time obtained with the GEQ for three different delay-line lengths for the artificial extreme case. (Bottom) The difference expressed as a percent of the obtained reverberation time deviation from target values.*

lies very close to zero. This causes a huge increase in the reverberation time for those frequencies, which is seen in the top plot in Fig. 9. When the difference between the target and the approximated reverberation time is expressed as a percentage into percentage, as shown in the bottom plot in Fig. 9, the 5% JND threshold is exceeded everywhere except for the low frequencies, where the target decay is the same in neighboring octave bands.

The results in Fig. 9 show the effect of the delay-line length on the approximation error. The attenuation for the shortest delay line is the weakest, making all deviations from the target magnitude to cause much error in the resulting reverberation time values. For the longest delay-line length, the overshoots between 1 kHz and 2 kHz, as well as around 8 kHz, are the smallest.

In order to improve the resulting reverberation time, we applied the gain-optimization method proposed in Sec. 2.3 by minimizing the error norm in Eq. (14) and using the weighting matrix in Eq. (13). The magnitude response of the GEQ is presented in Fig. 10. Overshoots between 1 kHz and 2 kHz were decreased at the cost of lower accuracy in approximating the target at 4 kHz. Additionally, some ripple was introduced in the frequency range between 250 Hz and 500 Hz.

The corresponding reverberation time values are shown in Fig. 11. The values that exceed the target the most were successfully reduced. The improvement was made without an unreasonable increase in error in the reverberation time for less problematic frequencies. Fig. 11 shows that the deviation in percent from the target values was the same or less for frequencies over 500 Hz. The error for low frequencies increased slightly, in most octave bands not exceeding the 5% JND.

The error-minimization method worked well for every delay-

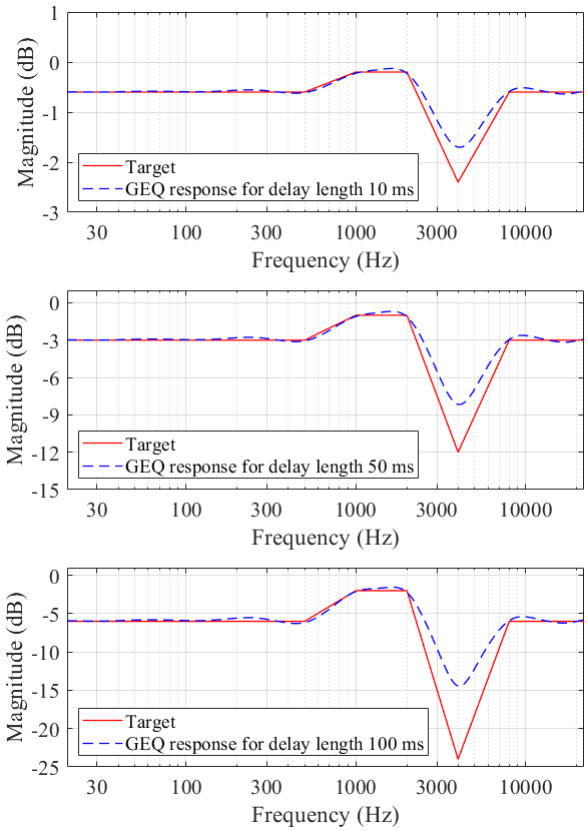


Figure 10: *Target magnitude response and response of the GEQ for three different delay-line lengths for the artificial extreme case. The command gains were weighted according to Eq. (9).*

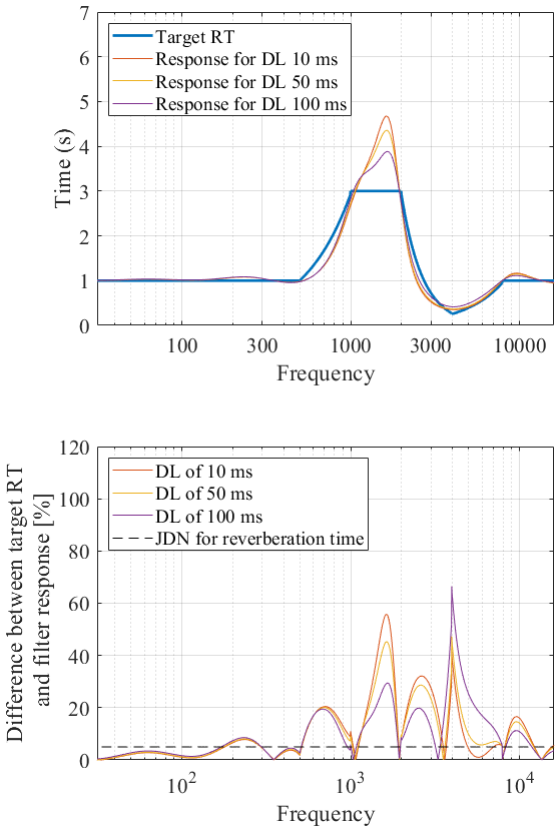


Figure 11: *(Top) Target reverberation time and reverberation time obtained with the GEQ for three different delay-line lengths for the artificial extreme case after gain weighting. (Bottom) Difference expressed as a percentage of the obtained reverberation time deviation from target values. Cf. Fig. 9.*

line length. In the end, all delay lines displayed similar differences from target values, which is a huge improvement from before the optimization.

3.3. Comparison with previous method

The proposed method was compared with the design presented in [13], which solves the nonlinear least-squares problem based on the T_{60} least-squares problem, as given in Eq. (11), with additional constraints on the command gains. The abbreviation TLSCon in Tables 3 and 4 refers to the results obtained with this method. To allow direct comparison, the same reverberation time and delay-line length were chosen. The error percentage in T_{60} for each method are presented in Table 3.

The proposed method produces a smaller approximation error than the solution suggested in [13]. The largest deviation from the target value occurs at 4 kHz, where an attenuation of -60 dB is needed. However, the obtained 62.69% error is a huge improvement compared to the TLSCon solution error of 280% for the same frequency.

3.4. Computational complexity of the proposed design

The GEQ with the first-order high-shelf filter used in the present work was compared in terms of computational cost with three other graphic equalizers: the proportional graphic equalizer (TLSCon) [13,16], the cascaded fourth-order equalizer (EQ4) [19], and the high-precision parallel equalizer (PGE) [17, 20]. The values are given for filter configurations as stated in [15] and are presented together with the number of operations for the proposed design in Table 4.

The design proposed in the present work requires less computation than the cascaded fourth-order equalizer and the high precision parallel equalizer. It needs a few more operations than the proportional graphic equalizer because the first-order high-shelf filter has been inserted to process frequencies above 16 kHz.

4. CONCLUSIONS

The present work investigated the effect of using a cascaded GEQ with a first-order high-shelf filter as the attenuation filter in the FDN. In order to evaluate the performance of the proposed design, two cases, a real-life case of an existing concert hall's reverberation time T_{60} and an artificially created extreme case, were tested. Additionally, weighted-gain optimization was performed to improve the results. The new weighting matrix emphasizes the approximation error occurring close to 0 dB. The gains are then determined by minimizing relative error between the filter response and the target reverberation time.

Table 3: *Error percentage for T_{60} approximated using the previous method (TLSCon) and the proposed method.*

Center frequency	63 Hz	125 Hz	250 Hz	500 Hz	1 kHz	2 kHz	4 kHz	8 kHz	16 kHz
TLSCon [13]	**1.00%**	**0.00%**	**3.00%**	**2.00%**	**9.00%**	**32.00%**	280.00%	31.00%	5.00%
Proposed method	3.39%	1.44%	8.15%	2.54%	16.98%	49.24%	**62.59%**	**20.69%**	**4.98%**

Table 4: *Number of operations per output sample for octave band equalizers.*

Design	ADD	MUL	TOTAL
TLSCon	40	50	90
EQ4	140	150	290
PGE	80	81	161
Proposed	42	52	94

The proposed method was shown to perform an excellent approximation of the real-life reverberation time values, resulting in an error between the target and obtained values that is lower than the JND. When the desired values change dramatically between neighboring frequency bands, the presented algorithm causes greater errors in the reverberation time, which can be then considerably reduced by the weighted-gain optimization. The study showed that the proposed method produces a smaller approximation error in the reverberation time than previous methods, and its computational cost is low or about the same compared to other designs.

The plans for further development of this work include providing subjective evaluation of the results as well as incorporating the proposed attenuation filter and the weighted-gain optimization method in other tools for creating artificial reverberation.

5. REFERENCES

[1] H. P. Seraphim, "Untersuchungen über die Unterschiedsschwelle exponentiellen Abklingens von Rauschbandimpulsen," *Acta Acustica united with Acustica*, vol. 8, no. 4, pp. 280–284, 1958.

[2] M. G. Blevins, A. T. Buck, Z. Peng, and L. M. Wang, "Quantifying the just noticeable difference of reverberation time with band-limited noise centered around 1000 Hz using a transformed up-down adaptive method," in *Proc. Int. Symp. Room Acoustics (ISRA)*, Toronto, Canada, June 9–11, 2013.

[3] M. Karjalainen and H. Järveläinen, "More about this reverberation science: Perceptually good late reverberation," in *Proc. 111th Audio Eng. Soc. Conv.*, New York, USA, Sept. 21–24, 2001.

[4] ISO, "ISO 3382-1, Acoustics – Measurement of room acoustic parameters – Part 1: Performance spaces," Tech. Rep., 2009.

[5] J. M. Jot and A. Chaigne, "Digital delay networks for designing artificial reverberators," in *Proc. 90th Audio Eng. Soc. Convention*, Paris, France, Febr. 19–22, 1991.

[6] V. Välimäki, J. D. Parker, L. Savioja, J. O. Smith, and J. S. Abel, "Fifty years of artificial reverberation," *IEEE Trans. Audio Speech Lang. Process.*, vol. 20, no. 5, pp. 1421–1448, Jul. 2012.

[7] S. J. Schlecht and A. P. Habets, "On lossless Feedback Delay Networks," *IEEE Trans. Signal Process.*, vol. 65, no. 6, pp. 1554–1564, Mar. 2017.

[8] J. Moorer, "About this reverberation business," *Computer Music J.*, vol. 3, no. 2, pp. 13–28, 1979.

[9] J. M. Jot, "Efficient models for reverberation and distance rendering in computer music and virtual audio reality," in *Proc. Int. Computer Music Conf.*, Thessaloniki, Greece, Sept. 1997.

[10] T. Wendt, S. van de Par, and S. D. Ewert, "A computationally-efficient and perceptually-plausible algorithm for binaural room impulse response simulation," *J. Audio Eng. Soc.*, vol. 62, no. 11, pp. 748–766, Nov. 2014.

[11] M. Holters and U. Zölzer, "Parametric high-order shelving filters," in *Proc. 14th European Signal Processing Conference (EUSIPCO)*, Florence, Italy, Sept. 4–8, 2006.

[12] J.-M. Jot, "Proportional parametric equalizers–Application to digital reverberation and environmental audio processing," in *Proc. 139th Audio Eng. Soc. Conv.*, New York, USA, Oct. 29–Nov. 1, 2015.

[13] S. J. Schlecht and A. P. Habets, "Accurate reverberation time control in Feedback Delay Networks," in *Proc. Digital Audio Effects (DAFx-17)*, Edinburgh, UK, Sept. 5–9, 2017, pp. 337–344.

[14] S. J. Orfanidis, *Introduction to Signal Processing*, Rutgers Univ., Piscataway, NJ, USA, 2010.

[15] V. Välimäki and J. Liski, "Accurate cascade graphic equalizer," *IEEE Signal Process. Lett.*, vol. 24, no. 2, pp. 176–180, Feb. 2017.

[16] R. J. Oliver and J. M. Jot, "Efficient multi-band digital audio graphic equalizer with accurate frequency response control," in *Proc. 139th Audio Eng. Soc. Conv.*, New York, USA, Oct. 29–Nov. 24, 2015.

[17] V. Välimäki and J. Reiss, "All about audio equalization: Solutions and frontiers," *Applied Sciences*, vol. 6, no. 5, May 2016.

[18] B. Bank and V. Välimäki, "Robust loss filter design for digital waveguide synthesis of string tones," *IEEE Signal Process. Lett.*, vol. 10, no. 1, pp. 18–20, Jan. 2003.

[19] M. Holters and U. Zölzer, "Graphic equalizer design using higher-order recursive filters," in *Proc. Int. Digital Audio Effects (DAFx-06)*, Montreal, Canada, Sept. 18–20, 2006, pp. 37–40.

[20] J. Rämö, V. Välimäki and B. Bank, "High-precision parallel graphic equalizer," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 22, no. 12, pp. 1894–1904, Dec. 2014.

EXTENSIONS AND APPLICATIONS OF MODAL DISPERSIVE FILTERS

Elliot K. Canfield-Dafilou and Jonathan S. Abel

Center for Computer Research in Music and Acoustics,
Stanford University, Stanford, CA 94305 USA
kermit|abel@ccrma.stanford.edu

ABSTRACT

Dispersive delay and comb filters, implemented as a parallel sum of high-Q mode filters tuned to provide a desired frequency-dependent delay characteristic, have advantages over dispersive filters that are implemented using cascade or frequency-domain architectures. Here we present techniques for designing the modal filter parameters for music and audio applications. Through examples, we show that this parallel structure is conducive to interactive and time-varying modifications, and we introduce extensions to the basic model.

1. INTRODUCTION

Dispersion filters, in which the various frequency components of an input signal are delayed by different amounts, find widespread use in audio processing. Often they are used to emulate the dispersive characteristic of physical systems. For example, [1] proposes a method for designing high-order allpass filters which were applied in [2] to model spring reverberators where low frequencies propagate faster than high frequencies and in [3] to model the dispersion of stiff strings where the high frequencies travel faster.

Other times, dispersive filters are designed to compensate for unwanted frequency-dependent delay. For example, [4–6] propose methods for designing allpass filters to equalize the group delay of elliptic filters. Recently, [7–9] and others have shown applications of dispersive filters for delay compensation between multiple drivers in a loudspeaker.

Rather than compensating for unwanted delay, some situations call for dispersive systems that add frequency-dependent delay with specific characteristics. For example, [10, 11] propose methods for decorrelating audio signals using high-order allpass sytems. Additionally, [12, 13] have proposed using high-order dispersive systems for abstract sound synthesis and processing.

In many applications, the dispersion filters are of very high order, having dozens to hundreds of poles. They are often implemented as high-order difference equations or biquad cascades. In the case of dispersion filters having thousands of poles, DFT-based convolution with the associated impulse response has been used [10]. A drawback to both high-order difference equations and biquad cascades is that they are prone to numerical difficulties. While biquad cascades are robust compared with high-order difference equation implementations, numerical errors accumulate through the cascade. DFT-based convolution techniques can produce high-order systems with large amount of dispersive delay, however they add latency.

Another drawback to both high-order difference equations and biquad cascades is that it is difficult to interactively change the desired group delay, $\tau(\omega)$. First, designing new filter coefficients to produce the new desired dispersive delay may be computationally costly. Second, substituting the new coefficients in the IIR filters is difficult, as the substitution interacts with the filter state, likely producing unwanted artifacts while the change ripples through the system. Third, certain changes, such as increasing or decreasing the number of coils in a simulated spring reverberator or modifying the length of a modeled string, change the number of poles needed to implement the desired dispersion—something that is difficult to do without artifacts. Frequency-domain implementations also present real-time interaction difficulties. Since frequency-domain methods process data in blocks, they produce computational latency precluding sample-by-sample processing and real-time interaction.

We recently proposed a modal approach for designing and implementing dispersive systems [14] that uses the modal architecture described in [15]. Two modal dispersion filters were introduced: a modal comb filter with multiple dispersive arrivals, and a modal delay filter with a single dispersive arrival. As will be shown in this work, the parallel structure is conducive to interactive modification of the dispersive characteristics and avoiding numerical issues associated with other methods. This paper will focus on time-varying dispersive audio effects and other extensions for modal dispersive filters.

2. DISPERSION FILTER DESIGN

We use a modal architecture, as shown in Fig. 1b to implement dispersive delay and comb filters. Its system impulse response, denoted by $h(t)$, is the sum of M parallel resonant filters with mode responses $h_m(t)$, $m = 1, 2, \ldots, M$,

$$h(t) = \sum_{m=1}^{M} h_m(t)\,. \tag{1}$$

The resonant mode responses $h_m(t)$ are complex exponentials, each characterized by a mode frequency ω_m, mode damping α_m, and complex mode amplitude γ_m,

$$h_m(t) = \gamma_m e^{(j\omega_m - \alpha_m)t}\,. \tag{2}$$

The system output $y(t)$ in response to an input $x(t)$ is then seen to be the sum of mode outputs

$$y(t) = \sum_{m=1}^{M} y_m(t), \quad y_m(t) = h_m(t) * x(t)\,, \tag{3}$$

where the mth mode output $y_m(t)$ is the mth mode response convolved with the input.

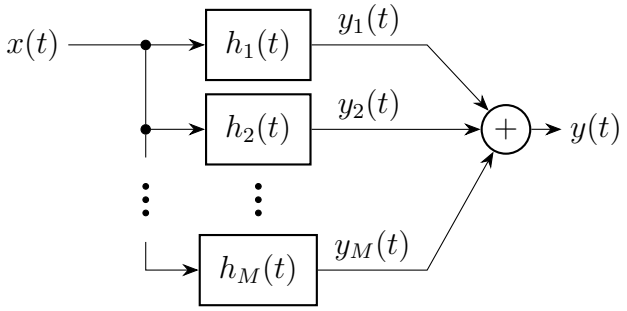


Figure 1: Modal filter architecture consisting of a parallel combination of resonant mode filters.

In the remainder of this section, we will show how to set the modal parameters to implement a desired dispersive filter. The desired delay $\tau(\omega)$ is used to specify the number of modes M and the mode frequencies ω_m. The desired decay time $T_{60}(\omega)$ and desired magnitude equalization $q(\omega)$ are used to fix the mode dampings α_m and mode amplitudes γ_m.

We begin by developing the formulation for dispersive comb and delay filters through Fourier theory.

2.1. Derivahrough Fourier Transform

Consider the M-point Inverse Discrete Fourier Transform

$$x(n) = \frac{1}{M} \sum_{m=-M/2}^{M/2-1} X(m) e^{\frac{j2\pi mn}{M}}, \tag{4}$$

where $X(m)$ represents the coefficients of M basis frequencies, indexed by m, and where n is the discrete time index. In the case where the coefficients $X(m)$ are independent of frequency (e.g., $X(m) = M$),

$$x(t) = \sum_{m=-M/2}^{M/2-1} e^{\frac{j2\pi mf_s}{M}\frac{n}{f_s}} = \sum_{m=-M/2}^{M/2-1} e^{j\omega_m t}. \tag{5}$$

The time domain signal is a band-limited, sampled, periodic sinc function with peaks every M samples as seen in Fig. 2a. We have introduced a sampling rate f_s which allows us to write angular frequency in radians per second, $\omega_m = j2\pi mf_s/M$, and time in seconds, $t = n/f_s$.

If we double M while maintaining the same sampling rate, meaning we double the frequency density of sinusoidal basis functions, the sinc has twice the period in the time domain as seen in Fig. 2b. Following this logic, the delay at each frequency is proportional to the frequency density of sinusoidal basis functions. Instead of a pure delay, a dispersive system can be formed by setting the frequencies of the sinusoidal bases according to the desired frequency-dependent delay $\tau(\omega)$.

To further unite this derivation of dispersive delay filters with our modal implementation, we introduce two more concepts. First, we introduce a damping factor α that causes the signal to subside over time,

$$x(t) = \left(\sum_{m=-M/2}^{M/2-1} e^{j\omega_m t} \right) e^{-\alpha t}, \tag{6}$$

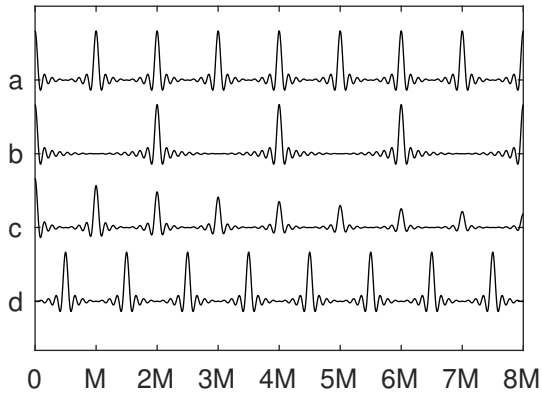


Figure 2: An upsampled sinc function showing the periodicity of the DFT (a). When the length of the DFT is doubled, the delay doubles (b). (c) shows the inclusion of the damping filter and (d) shows the effect of the shift theorem.

as seen in Fig. 2c. This damping can also be frequency-dependent α_m and factored inside the sum,

$$x(t) = \sum_{m=-M/2}^{M/2-1} e^{(j\omega_m - \alpha_m)t}. \tag{7}$$

Additionally, we can use the shift theorem,

$$x(t-\Delta) \longleftrightarrow e^{-j\omega_m \Delta} X(m), \tag{8}$$

to modify the time of the initial delay arrival. For example, we can achieve an odd integer set of arrivals by introducing a phase term of $e^{j\pi m} = (-1)^m$ into (5), as seen in Fig. 2d.

2.2. Dispersive Comb Filter Design

Following [14], the number of modes M is the number of samples of delay, averaged across the band from DC to the Nyquist limit, $f_s/2$,

$$M = \sum_{n=0}^{N} \frac{\tau_n f_s}{2N}, \tag{9}$$

where τ_n, $n = 0, 1, \ldots, N$, represents the desired delay $\tau(\omega)$ evaluated at the N discrete frequencies $\tau(2\pi nf_s/2)$, and where M can be rounded or otherwise adjusted to be an integer.

The mode frequencies ω_m are chosen to be those frequencies at which the cumulative delay $\varphi(\omega_m)$ hits integer multiples of 2π,

$$\varphi(\omega_m) = \int_0^{\omega_m} \tau(\nu) d\nu = 2\pi m. \tag{10}$$

The mode dampings α_m may be set according to a desired 60 dB decay time as a function of frequency $T_{60}(\omega)$,

$$\alpha_m = \frac{\ln(0.001)}{T_{60}(\omega_m)}. \tag{11}$$

Alternatively, α_m may be set to have a 60 dB decay after a given number of arrivals, $N_{60}(\omega)$. We then have

$$\alpha_m = \frac{\ln(0.001)}{(2N_{60}(\omega_m) - 1) \cdot \tau(\omega_m)}, \tag{12}$$

where the factor $2N_{60}(\omega_m) - 1$ is used assuming arrivals at odd integer multiples of the designed arrival time $\tau(\omega_m)$.

Since the energy in a given mode is proportional to its decay time, we set the mode gains according to

$$\gamma_m = \frac{e^{j\theta m}\alpha_m}{\tau(\omega_m)} \cdot q(\omega_m)\,, \quad \theta \in [0, 2\pi)\,, \tag{13}$$

where θ controls the initial time delay according to the shift theorem (8), the denominator $\tau(\omega)$ performs an allpass equalization, and $q(\omega)$ provides the desired frequency equalization. Unless otherwise stated, we use $\theta = \pi$ for odd integer multiples of the designed delay in this paper.

An example dispersive comb filter designed to have regions consisting of both smooth and staircase dispersion is shown in Fig. 3. The first arrival of the desired frequency-dependent spectral delay is shown as a dotted line overlayed on the spectrogram. Note that the impulse response's first arrival closely tracks the target time delay, and the subsequent arrivals are seen to have the anticipated odd integer multiples of the designed dispersive time delay. Fig. 4 shows the same dispersive characteristic as Fig. 3, but designed to decay 60 dB after eight arrivals.

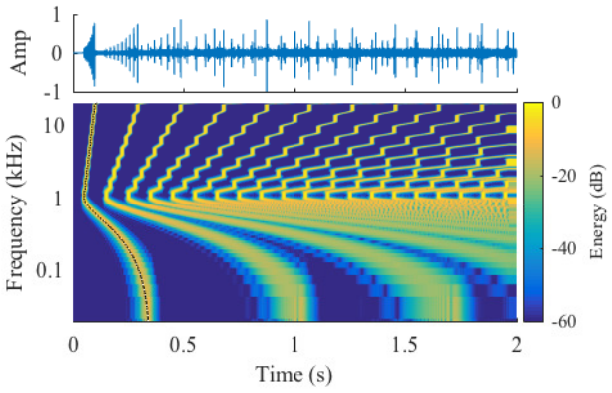


Figure 3: Impulse response (top) and spectrogram (bottom) of an example dispersive comb filter. The desired dispersive delay $\tau(\omega)$ is shown as a dotted line overlaid on the spectrogram.

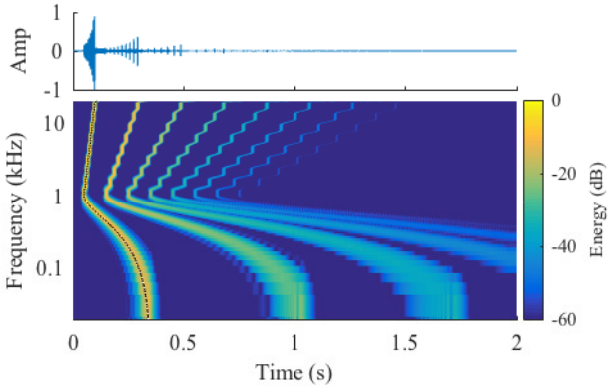


Figure 4: Impulse response (top) and spectrogram (bottom) of the dispersive comb filter from Fig. 3, set to decay 60 dB after eight arrivals.

2.3. Dispersive Delay Filter Design

There are two approaches for converting a dispersive comb filter into a dispersive delay filter that has only one arrival. First, the mode decay rates could be set to achieve a significant amount of attenuation between arrivals. For a system with odd integer multiples of the desired delay, we design the dampings to produce λ dB of decay between successive arrivals,

$$\alpha_m = \frac{\ln(10^{-\lambda/20})}{2\tau(\omega_m)}\,, \tag{14}$$

and scale the mode gains by $\lambda/2$ dB. This means the first dispersive arrival will have a roughly unit level and subsequent arrivals will be attenuated by at least λ dB. For audio applications, λ in the range 60–80 dB would render the unwanted subsequent arrivals inaudible. For instance, an attenuation of $\lambda = 60$ dB was used to design the dispersive delay shown in Fig. 5.

Since the $\lambda/2$ dB gain could create numerical difficulties, an alternative approach is to use a truncated IIR (TIIR) filter [16] to eliminate the unwanted subsequent echos as described in [14].

3. EXTENSIONS

In addition to the dispersive delay and comb filters shown above, the parameters exposed by the modal framework provide a powerful resource additional modification. Throughout this section, we will show the powerful effects of some simple modifications to the modal parameters and show some time-varying examples.[1]

When implementing time-varying filters it is important that the filters remain stable and avoid irritating artifacts that may arise from changing parameters quickly. We use Max Mathews's phasor filter [17] to implement these modal dispersion filters as seen in (3). This filter uses the property that when complex numbers are multiplied together, the magnitude is the product of their magnitudes and the phases sum [18]. With this implementation and

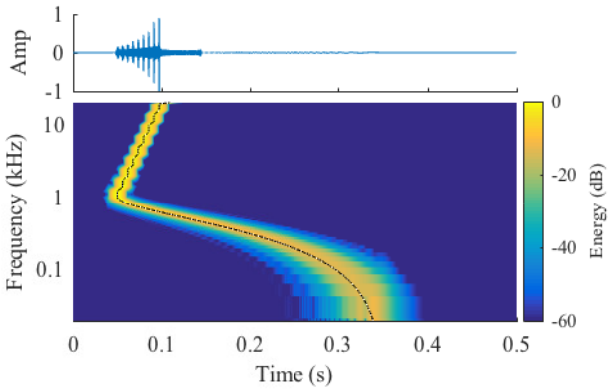


Figure 5: Impulse response (top) and spectrogram (bottom) of a dispersive delay filter constructed using decay time approach and the same designed delay as Figs. 3 and 4.

[1] Audio examples associated with the figures in the paper can be found at `https://ccrma.stanford.edu/~kermit/website/ddf.html`

the parallel structure, the dampings, frequencies, amplitudes, and input level can all be functions of time without introducing transients or other artifacts when the filter state is changed since the state of each filter does not depend on the state of the others.

3.1. Modifying the Number of Necessary Modes

The number of modes necessary to implement a dispersive filter is equivalent to the average delay as described by (9). In certain situations, it could be desirable to implement one of these filters with a specific number of modes (e.g., running with limited computational resources). Modifying the number of modes, however, comes at the expense of distorting the amount of delay at each frequency. A target group delay implemented with a desired number of modes $\tau_M(\omega)$ can be found by normalizing the old group delay by the average delay and scaling the result by the desired number of modes M,

$$\tau_M(\omega) = M \frac{\tau(\omega) + k}{\sum_{n=0}^{N} \frac{(\tau_n + k) f_s}{2N}}, \quad k > -\min_t \tau(\omega), \qquad (15)$$

where τ_n, $n = 0, 1, \ldots, N$, represents the desired delay $\tau(\omega)$ evaluated at the N discrete frequencies $\tau(2\pi n f_s/2)$ and k represents added delay that is independent of frequency. This additional delay controls how the M modes are distributed in frequency. When $k = 0$, the entire group delay curve is scaled by the amount necessary to have M samples of average delay. As k approaches $-\min_t \tau(\omega)$, any constant delay in $\tau(\omega)$ will be eliminated and the frequency regions with the most delay will exaggerated. As k becomes large, the detail of the group delay will be reduced until it is constant across frequency. Fig. 6 shows an example group delay curve warped with different values of k.

3.2. Transitioning Between Delay Characteristics

The amount of delay in a local frequency neighborhood is proportional to the density of modes in that neighborhood.

If we want to transition between two delay characteristics that have the same average delay, $\tau_a(\omega)$ and $\tau_b(\omega)$, we can interpolate between their mode frequencies over time. Because the mode frequencies will move, causing the delay to change, we will observe some Doppler shift during the transition. This may or may not be desirable. An example can be seen in Fig. 7

Another scheme for transitioning between delay characteristics can be accomplished by computing the output of the mode filters of both delay trajectories simultaneously, and crossfading the mode amplitudes. Here there will be no pitch shift, however during the transition, both dispersive characteristics will be audible simultaneously. Fig. 8 shows an example amplitude crossfade using the same dispersion filters and transition time as compared to Fig. 7. Fig. 9 shows a guitar track processed by dispersive comb filters with time-varying mode frequencies.

If the average delay is different (i.e., the number of modes is not the same), or we want to prevent mode frequencies from moving beyond a prescribed amount, we need a scheme for "birth and death" of mode filters [19]. This can be accomplished by using the mode amplitudes to fade "new" modes in and fade out "dead" modes in combination with amplitude and/or frequency morphing.

3.3. Damping Modifications

It is trivial to lengthen or shorten the decay time associated with each mode. The number of echos in each frequency band can be

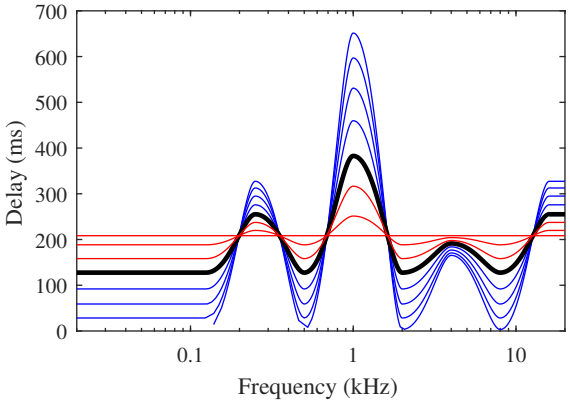


Figure 6: A set of group delays $\tau(\omega)$ that have the same average delay (i.e. the same number of modes) using (15) and different values of k—black: $k = 0$; red: $k > 0$; blue: $k < 0$.

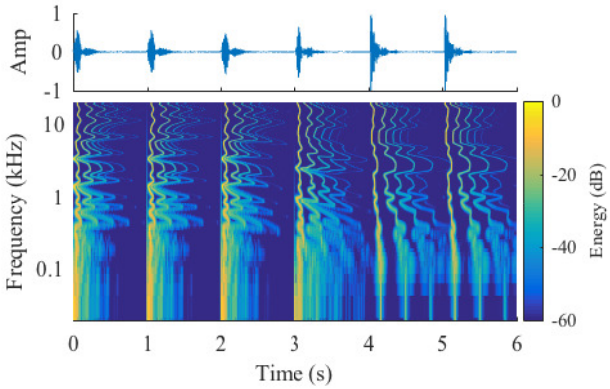


Figure 7: Time domain (top) and spectrogram (bottom) of an impulse train processed by a dispersive comb filter with time-varying mode frequencies. Note that the dispersive characteristic changes with the local mode density and some pitch shifting occurs.

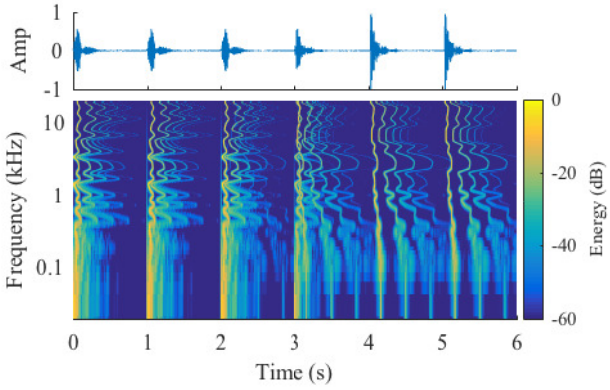


Figure 8: Time domain (top) and spectrogram (bottom) of an impulse train processed by two dispersive comb filters using amplitude modifications to cross-fade between the dispersive characteristics. Note that during the transition, both dispersive characteristics are audible and visible in the spectrogram.

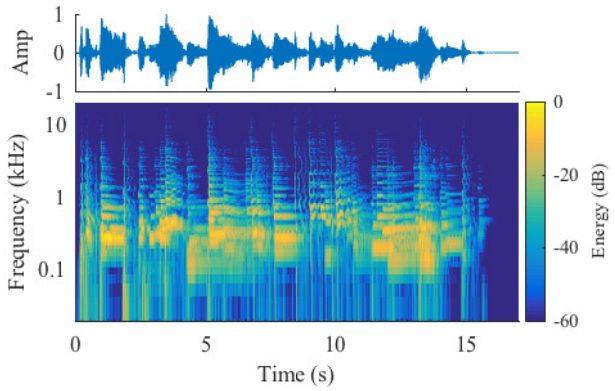


Figure 9: Time domain (top) and spectrogram (bottom) of a guitar track processed with dispersive comb filters with time-varying mode frequencies.

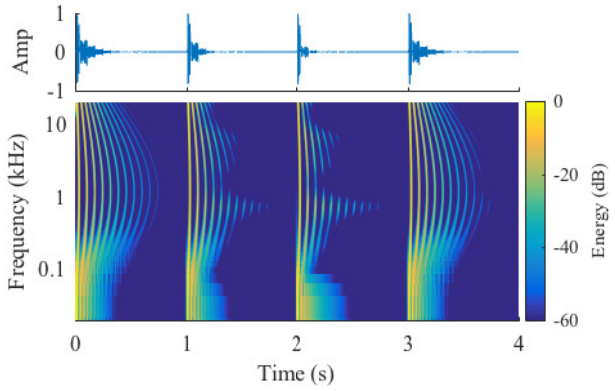


Figure 10: Time domain (top) and spectrogram (bottom) of an impulse train processed by a dispersive comb filter set to have time-varying scaling applied to the damping coefficients.

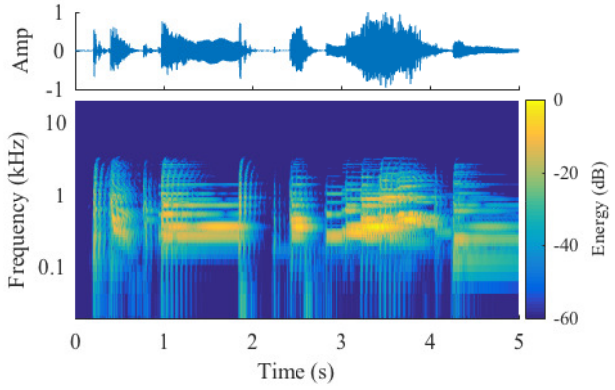


Figure 11: Time domain (top) and spectrogram (bottom) of a guitar track processed by a dispersive comb filter set to have damping coefficients that vary according to the input level of the guitar signal. The dispersion filter models a spring reverb where the reverb lasts longer when the input signal is louder.

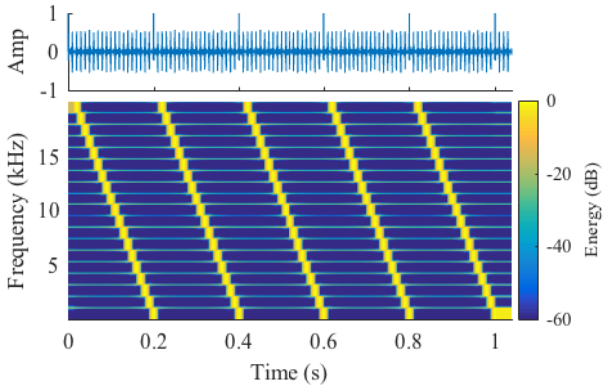


Figure 12: Time domain (top) and spectrogram (bottom) of a dispersive comb filter set to have a constant repeat rate and piecewise linear phase shift as a function of frequency.

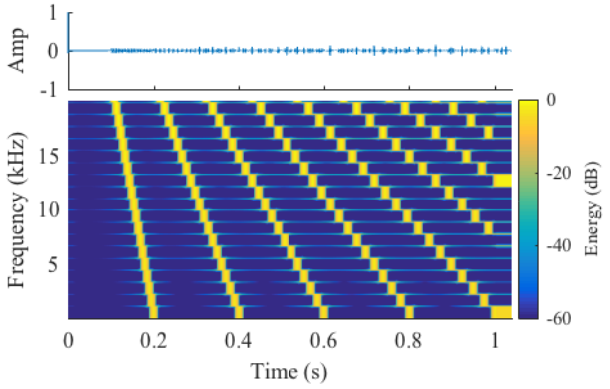


Figure 13: Time domain (top) and spectrogram (bottom) of a dispersive comb filter set to have a piecewise frequency-dependent delay to match the second reflection of Fig. 12. Notice that the lowest frequencies repeat on the same time interval in both figures while short initial delay time of the high frequencies causes a shorter period between subsequent arrivals.

frequency-dependent or be allowed to vary over time. For example, a dispersive comb filter could be designed that decays more quickly in the high frequencies than the low frequencies. Fig. 10 shows an example dispersive filter where the damping factors are frequency-dependent and time-varying.

Alternatively, a level tracker could be employed to modulate the number of echos that appear in the output based on the amplitude of the input signal. Fig. 11 shows a guitar track processed by a dispersive filter that reacts to its tracked level.

3.4. Phase Modifications

At each mode frequency a pulse is observed at repeated intervals that depend on the the desired delay $\tau(\omega)$ and the phase angle θ. We can have multiple frequency-dependent initial time offsets by allowing the phase angle to also be a function of frequency $\theta(\omega)$. As an example, Fig. 12 shows a filter constructed to have piecewise

constant phase on the interval $[0, 2\pi)$ such that all frequencies have the same repeat rate in time but a different initial phase. As a comparison, Fig. 13 models the same second dispersive arrival as frequency-dependent delay rather than with phase modifications. As a result, the periodicity of the subsequent arrivals is frequency-dependent.

If the phase of each mode is randomized, the result is a noisy signal constrained in time by the decay rates. This signal is akin to the description of late-field reverberation synthesis described in [15]. By allowing the phase to vary with time, and smoothly change between a coherent phase (where θ is constant) and a random phase (where each mode output is rotated by an independent, unit magnitude complex number), we can morph between dispersive delay and reverberant effects, such as seen in Fig. 14.

3.5. Echos Subsiding to Noise

Instead of a dispersive comb with clear, decaying echos, it is sometimes desirable to have a comb filter where each subsequent echo is a little more diffuse. After some number of echos, the signal is a noise-like wash where individual reflections are no longer detectable. The idea here is to perturb the frequencies of the modes by an amount small enough to be initially inaudible but cause the succeeding echos to be more spread out in time. The bandwidth of the perturbation is proportional to the number of desired audible echos. To have p distinct arrivals audible above the noise-like wash, the mode frequencies ω_m should be perturbed by noise with standard deviation of $1/p$ of the local frequency difference,

$$\tilde{\omega}_m = \frac{\omega_{m+1} - \omega_{m-1}}{2p} \nu_m + \omega_m, \tag{16}$$

where ν_m is a sample of zero-mean unit-variance noise, e.g. having a Gaussian or triangular distribution. The small perturbations are amplified with each subsequent arrival creating the desired effect. Fig. 15 shows an example dispersion filter, compared to Fig. 16 where the frequencies were perturbed to cause the echos to turn into noise after five reflections. This processing has applications for reverberation type effects.

4. CONCLUSION

Dispersive filters have music and audio applications ranging from physical modeling of dispersive systems to abstract sound synthesis. In this work, we explored extensions and applications of the modal dispersive delay and comb filters introduced in [14].

We began by showing how the modal formulation of these dispersion filters can be interpreted through Fourier theory. We then described how to set the modal parameters to achieve a desired dispersion characteristic based on the modal frequency density. Following that, we showed how the parallel structure of the modal architecture and the numerical properties of the phasor filter make it possible to efficiently and interactively modify the properties of these dispersion filters. Unlike frequency-domain or cascade architectures, it is simple to implement time-varying dispersion effects using the modal approach.

We showed a range of simple modifications for the modal parameters, and backed with examples, demonstrated an assortment of musical uses of dispersive comb and delay filters. Even so, there are certainly many more ways to extend this flexible structure, such as incorporating the pitch, time, and distortion processing described in [20] with the approaches presented here.

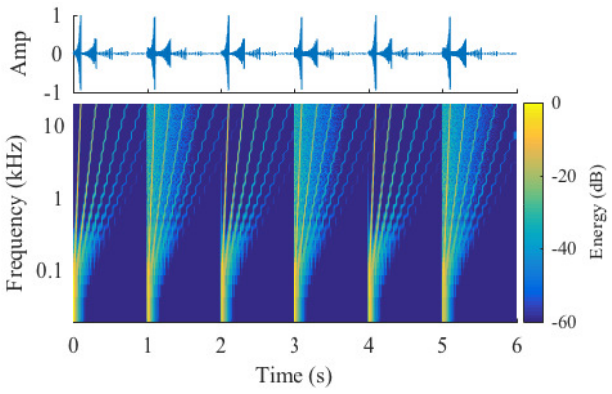


Figure 14: Time domain (top) and spectrogram (bottom) of an impulse train processed by a dispersive comb filter set to have periodic phase synchronization/desynchronization.

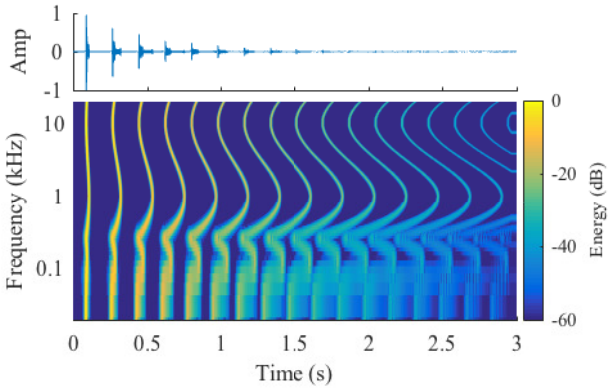


Figure 15: Time domain (top) and spectrogram (bottom) a dispersive comb filter without frequency perturbation.

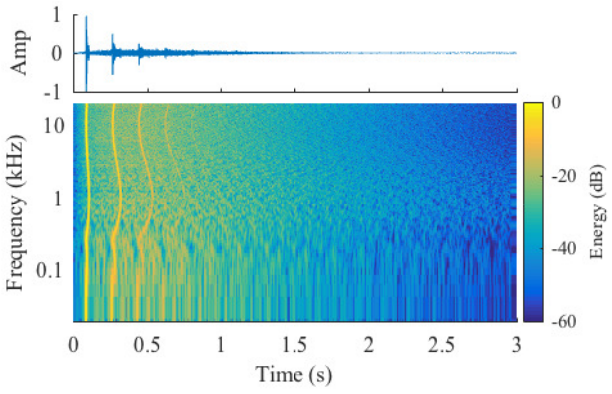


Figure 16: Time domain (top) and spectrogram (bottom) a dispersive comb filter with frequency perturbation designed to transition to noise after 4 arrivals.

5. REFERENCES

[1] Jonathan S. Abel and Julius O. Smith III, "Robust design of very high-order allpass dispersion filters," in *Proceedings of the 9th International Conference on Digital Audio Effects*, 2006, pp. 13–8.

[2] Jonathan S. Abel, David P. Berners, Sean Costello, and Julius O. Smith III, "Spring reverb emulation using dispersive allpass filters in a waveguide structure," in *Proceedings of the 121st Audio Engineering Society Convention*, 2006.

[3] Jonathan S. Abel, Vesa Välimäki, and Julius O. Smith III, "Robust, efficient design of allpass filters for dispersive string sound synthesis," *IEEE Signal Processing Letters*, vol. 17, no. 4, pp. 406–9, 2010.

[4] Zhongqi Jing, "A new method for digital all-pass filter design," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, no. 11, pp. 1557–64, 1987.

[5] Piotr Okoniewski and Jacek Piskorowski, "An analytical approach to the group delay compensation of digital IIR filters," in *Proceedings of the 17th IEEE International Conference on Methods and Models in Automation and Robotics*, 2012, pp. 75–8.

[6] Mauricio F. Quélhas, Antonio Petraglia, and Mariane R. Petraglia, "Efficient group delay equalization of discrete-time IIR filters," in *Proceedings of the 12th IEEE European Signal Processing Conference*, 2004, pp. 125–8.

[7] Aki Mäkivirta, Juho Liski, and Vesa Välimäki, "Effect of delay equalization on loudspeaker responses," in *Proceedings of the 144th Audio Engineering Society Convention*, 2018.

[8] Shintaro Hosoi, Hiroyuki Hamada, and Nobuo Kameyama, "An improvement in sound quality of LFE by flattening group delay," in *Proceedings of the 116th Audio Engineering Society Convention*, 2004.

[9] Stephan Herzog and Marcel Hilsamer, "Low frequency group delay equalization of vented boxes using digital correction filters.," in *Proceedings of the 16th International Conference on Digital Audio Effects*, 2014, pp. 57–64.

[10] Elliot Kermit-Canfield and Jonathan Abel, "Signal decorrelation using perceptually informed allpass filters," in *Proceedings of the 19th International Conference on Digital Audio Effects*, 2016, pp. 225–31.

[11] Elliot K. Canfield-Dafilou and Jonathan S. Abel, "A group delay-based method for signal decorrelation," in *Proceedings of the 144th Audio Engineering Society Convention*, 2018.

[12] Elliot K. Canfield-Dafilou and Jonathan S. Abel, "Group delay-based allpass filters for abstract sound synthesis and audio effects processing," in *Proceedings of the 21st International Conference on Digital Audio Effects*, 2018.

[13] Vesa Välimäki, Jonathan S. Abel, and Julius O. Smith III, "Spectral delay filters," *Journal of the Audio Engineering Society*, vol. 57, no. 7/8, pp. 521–31, 2009.

[14] Jonathan S. Abel and Elliot K. Canfield-Dafilou, "Dispersive delay and comb filters using a modal structure," submitted 2019.

[15] Jonathan S. Abel, Sean Coffin, and Kyle Spratt, "A modal architecture for artificial reverberation with application to room acoustics modeling," in *Proceedings of the 137th Audio Engineering Society Convention*, 2014.

[16] Avery Wang and Julius O. Smith III, "On fast FIR filters implemented as tail-canceling IIR filters," *IEEE Transactions on Signal Processing*, vol. 45, no. 6, pp. 1415–27, 1997.

[17] Max Mathews and Julius O. Smith III, "Methods for synthesizing very high Q parametrically well behaved two pole filters," in *Proceedings of the Stockholm Musical Acoustics Conference*, 2003.

[18] Dana Massie, "Coefficient interpolation for the Max Mathews phasor filter," in *Proceedings of the 133rd Audio Engineering Society Convention*, 2012.

[19] Robert McAulay and Thomas Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 4, pp. 744–54, 1986.

[20] Jonathan S. Abel and Kurt James Werner, "Distortion and pitch processing using a modal reverberator architecture," in *Proceedings of the 17th International Conference on Digital Audio Effects*, 2015.

AUDIO TRANSPORT: A GENERALIZED PORTAMENTO VIA OPTIMAL TRANSPORT

Trevor Henderson

Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA USA
tfh@mit.edu

Justin Solomon

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA USA
jsolomon@mit.edu

ABSTRACT

This paper proposes a new method to interpolate between two audio signals. As an interpolation parameter is changed, the pitches in one signal slide to the pitches in the other, producing a portamento, or musical glide. The assignment of pitches in one sound to pitches in the other is accomplished by solving a 1-dimensional optimal transport problem. In addition, we introduce several techniques that preserve the audio fidelity over this highly nonlinear transformation.

A portamento is a natural way for a musician to transition between notes, but traditionally it has only been possible for instruments with a continuously variable pitch like the human voice or the violin. Audio transport extends the portamento to *any* instrument, even polyphonic ones. Moreover, the effect can be used to transition between different instruments, groups of instruments, or any other pair of audio signals. The audio transport effect operates in real-time; we provide an open-source implementation. In experiments with sinusoidal inputs, the interpolating effect is indistinguishable from ideal sine sweeps. More generally, the effect produces clear, musical results for a wide variety of inputs.

1. INTRODUCTION

A portamento, or musical glide, has been a significant expressive device in music for at least the past 200 years [1, 2]. Short portamenti can connect notes to make a passage sound more fluid, while long portamenti can draw out a transition with anticipation before finally arriving at the destination. The author in [1] claims that "portamento draws on innate emotional responses to human sound, as well as on our earliest memories of secure, loving communication, in order to bring to performances a sense of comfort, sincerity, and deep emotion." Regardless of whether this text describes a universal experience, portamenti have a decidedly unique sound and musical significance.

Due to the nature of the sound, the only instruments that can produce portamenti are instruments that, like the human voice, can vary their pitch continuously. Certain electronic systems described in §1.1 are capable of producing the effect, but they are limited to particular situations (e.g. monophonic glide, offline processing). In this work, we present an audio effect titled, "audio transport," which interpolates between *any* two audio streams in a way that sounds like a portamento, automatically and in real-time.

The audio transport effect relies on solving a 1-dimensional optimal transport problem. The solution to this problem determines how the pitches in one signal will move to pitches in the other. We find that the effect works best on pairs of sounds that do not have sharp attacks or strong tremolo and have comparable brightness.

The paper is organized as follows. §2 gives a brief introduction to the optimal transport problem and its relevance to the rest of the paper. §3 presents the audio transport effect, including a number of techniques necessary to produce artifact-free audio. §4 details our implementation of the audio transport effect and provides perceptual results. Finally, §5 concludes with discussion of potential applications and future work.

1.1. Previous Work

Portamenti have existed in electronic music since its inception. One of the earliest electronic instruments, the theremin, is famed for the sweeping sounds it can produce from its continuous pitch control. Today, a pitch wheel can be found on almost all synthesizers as a way to bend a note's pitch.

In addition to manually-controlled portamenti, many synthesizers have a "glide" parameter which automatically introduces portamenti between sequential notes. Typically this effect is monophonic, but some synthesizers support polyphonic glide using rule based systems [3].

As for sample-based instruments, the pitch of a sample can be changed by varying its playback speed. Alternatively, phase vocoders allow for a sample's pitch to be changed independently of its speed [4, 5]. Both of these methods, like a pitch wheel, can produce a polyphonic portamento but they necessarily move all the pitches in the same direction at the same rate. As such, these techniques can not be used to slide between chords with different harmonies or instruments with different timbre.

Techniques involving phase vocoders [6], modulation vocoders [7], and popular but unpublished commercial products like Melodyne [8] allow for artists to vary pitches within a sample independently, which could conceivably be used to create polyphonic portamenti. This type of pitch manipulation, however, is not suited for real-time use because without manual input, the pitches have no destination.

While not related to portamenti, optimal transport has been applied to audio problems before. The authors in [9] describe how optimal transport can be used to perform spectral unmixing with application to musical transcription. The authors in [10] apply optimal transport to the problem of fundamental pitch estimation. Both of these papers focus on analysis rather than synthesis.

1.2. Contributions

We present audio transport, an audio effect that produces a portamento between arbitrary audio sources. The effect works by interpolating between the spectra of the two input signals according

to an optimal transport map. To our knowledge, this is the first work to apply optimal transport to audio generation and also the first that can achieve this type of portamento effect automatically and in real-time. In addition to the novel application of optimal transport, we present a technique based on time-frequency reassignment [11] that divides the audio spectrum prior to transport and we extend the phase accumulation technique from [5] to prevent phasing between windows.

2. OPTIMAL TRANSPORT OVERVIEW

The optimal transport problem asks how to move probability mass from one configuration to another in a way that minimizes the amount of work (mass times distance) performed on each infinitesimal piece of mass. More formally [12], the problem seeks an optimal plan $\pi^*(x, y)$ that describes how much mass should be transferred from position x to y satisfying:

$$\pi^* = \arg\min_{\pi} \iint_{\mathbb{R}^2} \|x - y\|^p \, d\pi(x, y), \tag{1}$$

subject to nonnegativity as well as conservation of mass for source and target distributions ρ_v and ρ_w:

$$\int_{\mathbb{R}} \pi(x, y)\, dy = \rho_v(x) \quad \text{and} \quad \int_{\mathbb{R}} \pi(x, y)\, dx = \rho_w(y). \tag{2}$$

The p-th root of the optimal value provides an intuitive way to measure the similarity between two distributions known as the p-Wasserstein distance. In the rest of this paper, we will use $p = 2$. The corresponding "least squares" Wasserstein distance satisfies all metric axioms among other attractive properties [13, 12].

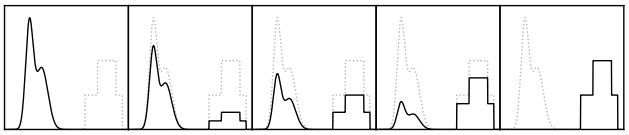
We use the optimal plan to perform *displacement interpolation* between two distributions [14]. This interpolation animates the mass assignment computed in Equation (1) by sliding each particle of mass between its two assignments. In computer graphics, this interpolation technique can be used to naturally transition between histograms, images, or meshes [15, 13, 16, 17, 18].

Consider Figure 1, which demonstrates two different ways to interpolate between distributions. On top, the distributions are interpolated linearly. If we imagine the distributions as audio spectra, then this transformation is simply fading one set of pitches out and another set in. On the bottom, the same distributions are transformed using displacement interpolation. The mass physically slides from one location to another. If these were audio spectra, this sliding would sound like a portamento.

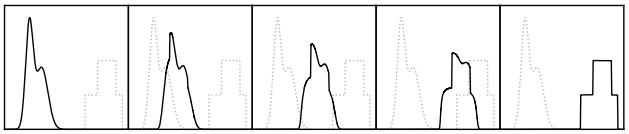
It should be noted that solving the optimal transport problem is known to be computationally challenging for any dimension $d > 1$. Fortunately, solving the problem on the real line can be done in linear time [18].

3. AUDIO TRANSPORT

The audio transport effect works by performing displacement interpolation on input audio spectra, so that pitches in one signal slide to pitches in the other as an interpolation parameter is changed. To modify the spectra over time, the audio transport algorithm follows the phase-vocoder paradigm [4, 5, 6]. In detail, a sliding short-time Fourier transform (STFT) is applied to both input audio streams, producing complex spectra. These spectra are interpolated according to the optimal transport map and fed through an inverse STFT to form the output audio stream.



(a) A linear interpolation or "fade"



(b) Displacement interpolation via optimal transport or a "portamento"

Figure 1: The distribution on the left is transformed into the distribution on the right with two different interpolation methods.

§3.1 describes an efficient way to interpolate between spectra using optimal transport. Alone, this method produces two artifacts which, borrowing from phase-vocoder literature, are known as *vertical incoherence* and *horizontal incoherence* [5]. Solutions to these two phenomena are described in §3.2 and §3.3, respectively.

3.1. Optimal Transport Between Spectra

Consider discrete spectra represented by complex vectors X, Y and corresponding frequency vectors ω^X, ω^Y. Analogously to the continuous optimal transport plan given in Equation (1), we can write the optimal transport plan between these discrete spectra as the plan $\pi^* \in \mathbb{R}^{|X|\times|Y|}$ minimizing:

$$\pi^* = \arg\min_{\pi \geq 0} \sum_{i,j} \left| \omega_i^X - \omega_j^Y \right|^2 \pi_{ij} \tag{3}$$

subject to the conservation of mass constraint

$$\sum_j \pi_{ij} = |X_i| \quad \text{and} \quad \sum_i \pi_{ij} = |Y_j|. \tag{4}$$

This problem assumes that $\sum_i |X_i| = \sum_j |Y_j|$. To treat spectra with different total magnitudes, the plan can be computed on normalized spectra; then, scaling is interpolated linearly over the interpolation.

Once an optimal plan is computed, the spectra can be interpolated with parameter $k \in [0, 1]$ by placing each mass π_{ij}^* at the displaced frequency:

$$(1 - k)\omega_i^X + k\omega_j^Y \tag{5}$$

If multiple masses are placed at the same frequency, they are added together. The phase attributed to the mass is considered in §3.3.

In one dimension, the optimal transport plan is monotone or, in other words, no mass crosses over any other mass [19]. This allows for Equations (3) and (4) to be solved using the greedy strategy presented in Algorithm 1.

The algorithm begins with the initial bins of the two spectra. Since no mass can cross over any other, all of the mass in the smaller bin must be assigned to the larger. With this assignment done, one can imagine virtually removing the smaller bin and shrinking the mass of the larger by the mass assignment. The algorithm then continues inductively on the smaller problem. At every

Algorithm 1 Computing The Optimal Transport Matrix, π^*

$\pi^*_{i,j} \leftarrow 0$
$\rho_X, \rho_Y \leftarrow |X_0|, |Y_0|$ ▷ ρ is the mass left in a bin

loop
  if $\rho_X < \rho_Y$ **then**
    $\pi^*_{ij} \leftarrow \rho_X$ ▷ Assign as much mass as possible
    $i \leftarrow i + 1$ ▷ Refill the emptied bin
    if $i \geq |X|$ **then** break
    $\rho_X \leftarrow |X_i|$
    $\rho_Y \leftarrow \rho_Y - \rho_X$ ▷ Decrease the capacity of the other
  else
    Symmetric to the case above
return π^*

iteration, all of the mass in one bin becomes completely assigned. Therefore, the complexity of the algorithm is $O(|X| + |Y|)$. This runtime is efficient relative to the super-linear runtime of the fast Fourier transform.

3.2. Resolving Vertical Incoherence: Slicing the Spectrogram

One unfortunate effect of using an STFT is the necessary trade-off between time and frequency resolution. As the time resolution increases, the frequency domain becomes "smeared."

The relation between a peak frequency and its smeared components is known to be important for perceptual quality. Treating these independently leads to phasing artifacts within a window known as vertical incoherence [5]. One method to solve this problem in phase vocoder literature is to "lock" regions surrounding a peak frequency so that the relative phase between bins within these regions remains unchanged [20, 5].

If Algorithm 1 were applied directly to audio spectra, it would introduce vertical incoherence by translating smeared components independently. So, applying the locking strategy, we will treat smeared regions as single units with collective magnitude in the transportation map.

It now remains to determine how exactly to choose the boundaries between smeared spectral regions. A common strategy is to use a heuristic to find local peaks and then assign the boundaries to be the midpoints of the peaks. Since displacement interpolation makes extreme changes to the spectra, however, this somewhat naïve plan [5, 6] is not sufficiently robust to produce a clean signal. We propose a more principled segmentation method based on frequency reassignment.

Frequency reassignment uses information in a signal's phase to enhance its frequency resolution. Each spectral component with frequency ω_i is mapped to the reassigned frequency $\hat{\omega}_i$ that better reflects the true energy distribution [11]. Sinusoids that have been smeared across multiple bins become mapped to the same central frequency, which produces the plateaus shown in Figure 2.

With this view, an intuitive way to define sinusoidal regions is by the zero crossings of $\hat{\omega}_i - \omega_i$. Falling crossings indicate the center bin of a region while rising crossings indicate the boundaries. These can be computed at the cost of an additional STFT

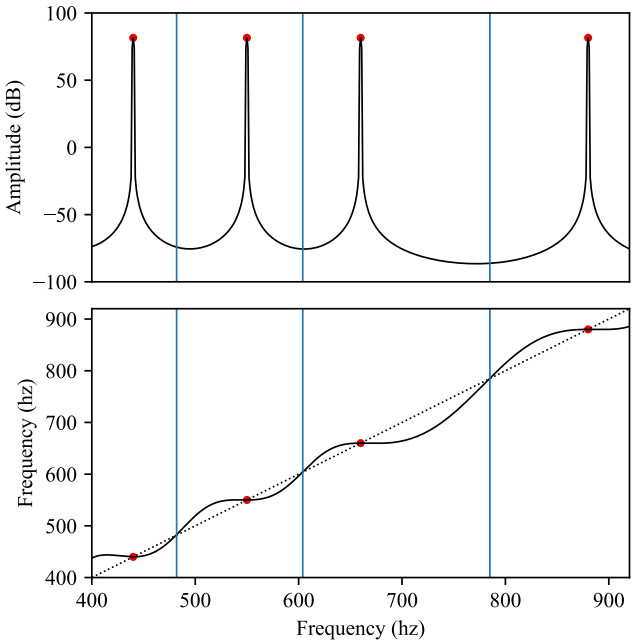


Figure 2: Dividing the spectrum of a sinusoidal A major chord consisting of the notes A_4, $C\sharp_5$, E_5 and A_5. The spectrum is displayed on top. On the bottom, the reassigned frequency $\hat{\omega}$ (solid line) is plotted against the frequency ω (dashed line). The intersections of these lines indicate the boundaries between groups (vertical lines) and their pitch centers (dots).

with the following formula [11]:

$$\hat{\omega}_i - \omega_i = \Im\left\{\frac{X_i^{\mathcal{T}h} \cdot X_i^*}{|X_i^2|}\right\}. \tag{6}$$

$X^{\mathcal{T}h}$ is the STFT computed using a time-weighted analysis window.

3.3. Resolving Horizontal Incoherence: Phase Accumulation

Finally, we reintroduce phase to the spectra. In doing so, we will be concerned with the phase relations between consecutive windows rather than the phase relations within a window. Inter-window phase relations carry information about short-time events like transients and hence ignoring these relations can create a blurry sound in some cases as discussed in § 4.2.

When a particular spectral region is transposed, its phase rotates at a different rate. Thus, applying the phases of consecutive windows in the original signal to the corresponding windows of the transposed signal causes interference known as horizontal incoherence [5].

In phase vocoders, this is resolved by integrating the reassigned frequency over the window difference. In other words, the phase φ_i^t in bin i and window t can be estimated from the phase φ_i^{t-1} in window $t-1$ as follows:

$$\varphi_i^t = \varphi_i^{t-1} + \hat{\omega}_i^{t-1} \cdot \Delta, \tag{7}$$

where Δ is the delay between the windows in seconds. This update is applied to *center* bin of a region as described in §3.2. The other

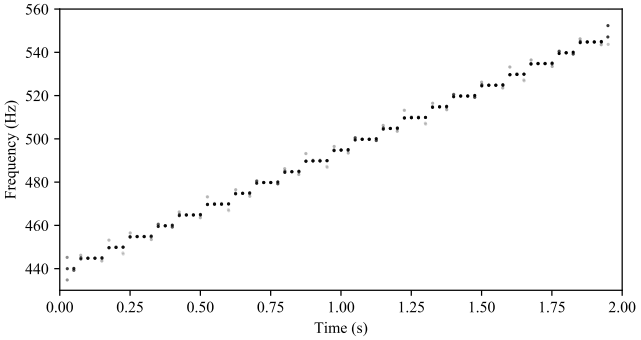


Figure 3: A (reassigned [11]) spectrogram of the audio transport effect being used to interpolate between sinusoids at A_4 and $C\sharp_5$. The "stair-stepping" effect is due to the frequency resolution of $\approx 5\,\mathrm{Hz}$, but this can be reduced arbitrarily by padding each window with additional zeros.

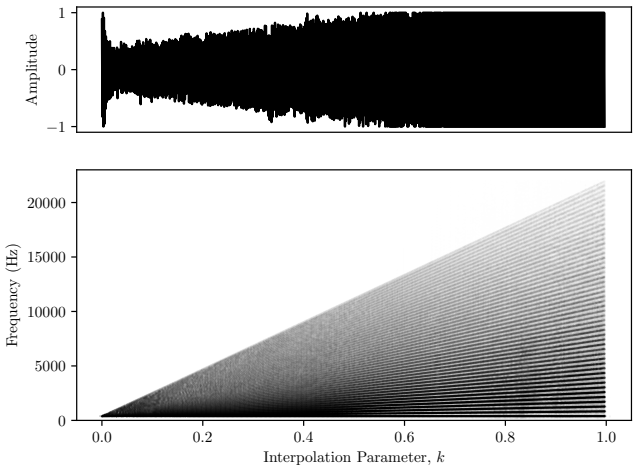


Figure 4: Interpolating between a sine wave ($k = 0$) and a saw wave ($k = 1$) leads to a drop in volume for values of k close to but not equal to zero as shown in the audio signal on top. The spectrogram shows how the sine's single peak splits into many peaks, which interfere when they are close together.

phases in each region are modified accordingly to maintain the same relative phase with respect to the center bin.

Some small modifications must be made to apply Equation (7) to the audio transport effect. First of all, it is possible that many spectral regions overlap on a particular bin, which makes the terms $\hat{\omega}_i^{t-1}$ and φ_i^{t-1} ambiguous. To resolve this we simply choose the frequency and phase of the loudest overlapping region. Additionally, since the audio transport effect consists of rapidly-moving pitches, we can minimize phasing by averaging the current reassigned frequency and the previous reassigned frequency:

$$\varphi_i^t = \varphi_i^{t-1} + \frac{\hat{\omega}_i^t + \hat{\omega}_i^{t-1}}{2} \cdot \Delta. \tag{8}$$

4. RESULTS

We implemented the audio transport effect described in §3 for real-time audio interpolation. We tested our implementation on synthetic sounds described in §4.1 as well as on a variety of complex and natural sounds described in §4.2.

All of our results are performed on $44.1\,\mathrm{kHz}$ audio with a window size of $0.05\,\mathrm{s}$ or 2206 samples. We use a Hann analysis window with 50% overlap and no synthesis window. Additionally, the windows are padded with zeros to increase the frequency resolution of the FFT to $\approx 5\,\mathrm{Hz}$.

Our implementation is open source and available at `https://github.com/sportdeath/audio_transport`.

4.1. Interpolating Sinusoids

We used the audio transport effect to interpolate between single sinusoids. Intuitively, this should sound exactly like a sine sweep between the input pitches. We performed listening experiments for interpolations at a variety of speeds and with inputs spanning the entire perceptive range. The spectrogram of one such interpolation is shown in Figure 3. Almost all of the interpolations were indistinguishable from real sine sweeps. In extreme cases where the interpolations were faster than $2000\,\mathrm{Hz\,s^{-1}}$ we perceived some phase distortion, but these situations would be rare in normal use.

The audio does exhibit "stair-stepping" between frequencies due to the frequency resolution of the FFT and the time resolution of the windows as demonstrated in Figure 3. Due to the small time-frequency resolution of the steps, however, we were unable to perceive them in the listening experiments.

4.2. Interpolating Natural Sounds

We tested the audio transport effect on a variety of sounds, some of which are available at `https://soundcloud.com/audio_transport`. The effect intuitively sounds like a portamento, even when applied to unnatural cases like a piano note gliding into a human voice. It is applicable to monophonic and polyphonic sounds. We even had success using it to transition between entire songs. The audio typically sounds artifact-free for many classes of audio signals, with a few exceptions noted below.

The audio transport effect does not guarantee temporal consistency between transport maps. So interpolating between sounds with dynamic spectra, like a pair of wavering orchestral chords, can produce a fluctuating pitch.

Another artifact that can occur is a sudden drop in volume when the interpolation parameter is close to but not equal to either zero or one as shown in Figure 4. This happens when one single frequency is mapped to a large range of frequencies. As the single frequency separates, its components interfere with each other, reducing the volume. This artifact is most prevalent when interpolating between sounds with vastly different spectral complexity as is the case with pairs of bright and dark sounds.

It is also worth reminding the reader that this method is intended for static sounds and will blur transients, as mentioned in §3.3. This artifact can be subtle, but when we directly compared the output of the audio transport effect with an interpolation factor of 0 to the corresponding input we consistently picked out the original audio when it had sharp transients like hi-hats. We suspect that this could be fixed using phase reinitialization techniques [5], but this exploration is left to future work.

5. CONCLUSION

In this paper, we introduced the *audio transport* effect which can create a portamento-like transition between any two audio signals. The effect produces a novel but intuitive sound and it is controlled by a single interpolation parameter. As a result, it is accessible for musicians to incorporate into both live performances and studio recordings.

In our live experiments, we controlled the interpolation parameter using a MIDI pitch fader, but really it can come from any source. For example, an instrument could be constructed where the velocity of a note controls the interpolation parameter. As much as we have described the effect as a portamento, the input pitches do not need to have a different fundamental. The effect also produces interesting interpolations between signals with the same fundamental pitch but different timbre.

Our work on audio transport suggests several other use cases beyond those explored in our experiments. For example, consider a single audio source that is fed as one input of the audio transport effect, and the output of the effect is fed back into the other input. By keeping the interpolation parameter constant, the pitches in the output should lag behind the input pitches similar to synthesizer "glide." This setup leads to several questions: What would happen if other effects were added to the feedback chain? Is it interesting to use multiple audio transport effects at the same time? The latter may be supported by the notion of *barycenters* in optimal transport [21].

The audio transport effect as described still produces artifacts for certain classes of sounds. Future work to resolve these could investigate ways to sharpen transients, make transport maps temporally consistent, and reduce the effects of energy cancellation. For a wide variety of inputs, however, our effect sounds smooth, musical and inspiring.

Acknowledgments. The authors acknowledge the generous support of Army Research Office grant W911NF-12-R-0011, of Air Force Office of Scientific Research award FA9550-19-1-0319, of National Science Foundation grant IIS-1838071, from an Amazon Research Award, and from the MIT-IBM Watson AI Laboratory. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of these organizations.

6. REFERENCES

[1] Daniel Leech-Wilkinson, "Portamento and musical meaning," *Journal of Musicological Research*, vol. 25, no. 3-4, pp. 233–261, 2006.

[2] Mark Katz, "Portamento and the phonograph effect," *Journal of Musicological Research*, vol. 25, no. 3-4, pp. 211–232, 2006.

[3] Ralph Deutsch and Leslie J Deutsch, "Constant speed polyphonic portamento system," Oct. 19 1982, US Patent 4,354,414.

[4] James L. Flanagan and R.M. Golden, "Phase vocoder," *Bell System Technical Journal*, vol. 45, no. 9, pp. 1493–1509, 1966.

[5] Jean Laroche and Mark Dolson, "Improved phase vocoder time-scale modification of audio," *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 3, pp. 323–332, 1999.

[6] Jean Laroche and Mark Dolson, "New phase-vocoder techniques for pitch-shifting, harmonizing and other exotic effects," in *Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE, 1999, pp. 91–94.

[7] Sascha Disch and Bernd Edler, "An amplitude-and frequency modulation vocoder for audio signal processing," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx)*, 2008.

[8] Peter Neubaecker, "Sound-object oriented analysis and note-object oriented processing of polyphonic sound recordings," 2009, US8022286B2.

[9] Rémi Flamary, Cédric Févotte, Nicolas Courty, and Valentin Emiya, "Optimal spectral transportation with application to music transcription," in *Advances in Neural Information Processing Systems*, 2016, pp. 703–711.

[10] Filip Elvander, Stefan Ingi Adalbjörnsson, Johan Karlsson, and Andreas Jakobsson, "Using optimal transport for estimating inharmonic pitch signals," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 331–335.

[11] Kelly R Fitz and Sean A Fulop, "A unified theory of time-frequency reassignment," *arXiv:0903.3080*, 2009.

[12] Cédric Villani, *Optimal Transport: Old and New*, vol. 338, Springer Science & Business Media, 2008.

[13] Justin Solomon, Fernando De Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas, "Convolutional Wasserstein distances: Efficient optimal transportation on geometric domains," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, pp. 66:1–66:11, 2015.

[14] Robert J McCann, "A convexity principle for interacting gases," *Advances in Mathematics*, vol. 128, no. 1, pp. 153–179, 1997.

[15] Nicolas Bonneel, Michiel Van De Panne, Sylvain Paris, and Wolfgang Heidrich, "Displacement interpolation using Lagrangian mass transport," in *ACM Transactions on Graphics (TOG)*. ACM, 2011, vol. 30, pp. 158:1–158:12.

[16] Hugo Lavenant, Sebastian Claici, Edward Chien, and Justin Solomon, "Dynamical optimal transport on discrete surfaces," *ACM Trans. Graph.*, vol. 37, no. 6, pp. 250:1–250:16, Dec. 2018.

[17] Bruno Lévy and Erica L Schwindt, "Notions of optimal transport theory and how to implement them on a computer," *Computers & Graphics*, vol. 72, pp. 135–148, 2018.

[18] Gabriel Peyré, Marco Cuturi, et al., "Computational optimal transport," *Foundations and Trends in Machine Learning*, vol. 11, no. 5-6, pp. 355–607, 2019.

[19] Filippo Santambrogio, *Optimal Transport for Applied Mathematicians*, vol. 55, Springer, 2015.

[20] Miller Puckette, "Phase-locked vocoder," in *Workshop on Applications of Signal Processing to Audio and Accoustics*. IEEE, 1995, pp. 222–225.

[21] Martial Agueh and Guillaume Carlier, "Barycenters in the Wasserstein space," *SIAM Journal on Mathematical Analysis*, vol. 43, no. 2, pp. 904–924, 2011.

ANALYSIS AND EMULATION OF EARLY DIGITALLY-CONTROLLED OSCILLATORS BASED ON THE WALSH-HADAMARD TRANSFORM

Leonardo Gabrielli
Department of Information Engineering
Università Politecnica delle Marche
Ancona, Italy
l.gabrielli@univpm.it

Stefano D'Angelo
Independent researcher
Agropoli, Italy
s@dangelo.audio

Luca Turchet
Department of Information Engineering
and Computer Science,
University of Trento
Trento - Italy
luca.turchet@unitn.it

ABSTRACT

Early analog synthesizer designs are very popular nowadays, and the discrete-time emulation of voltage-controlled oscillator (VCO) circuits is covered by a large number of virtual analog (VA) textbooks, papers and tutorials. One of the issues of well-known VCOs is their tuning instability and sensitivity to environmental conditions. For this reason, digitally-controlled oscillators were later introduced to provide stable tuning. Up to now, such designs have gained much less attention in the music processing literature. In this paper, we examine one of such designs, which is based on the Walsh-Hadamard transform. The concept was employed in the ARP Pro Soloist and in the Welson Syntex, among others. Some historical background is provided, along with a discussion on the principle, the actual implementation and a band-limited virtual analog derivation.

1. INTRODUCTION

Voltage-controlled oscillators (VCO) are considered among the fundamental building blocks in subtractive synthesis, together with voltage-controlled filters and amplifiers. Before the inception of transistor technology, sound sources in electronic music were very disparate: from heterodyne mixers (e.g., the Theremin), to fixed-frequency electromechanical oscillators (e.g., the Hammond organ), from neon tube oscillators (e.g., the Trautonium) to magnetic tapes. The inception of transistor technology, however, induced early electronic music pioneers to investigate new solutions. Between 1959 and 1960, Harald Bode, a German Engineer that had previously worked for the Cologne studio with Stockhausen, developed a novel concept of modular synthesizer, employing transistor technology [1] and the voltage control paradigm. This system was only meant for sound processing and had no oscillator. Robert Moog later adopted this modular concept to develop what would arguably be the most well-known brand of synthesizers [2]. For his oscillators he considered the 1V per octave paradigm [3], which has later become one of the industry standards. At the beginning of the 1970s many synthesizers were produced, which were based on VCOs.

One of the issues with VCOs is tuning stability. During the 1970s, solutions were proposed for VCOs with better stability, one of which is the shift to Digitally Controlled Oscillators (DCO). Early synthesizer DCO designs were adapted from transistor organs, where usually, a master clock source is divided to obtain the 12 notes of the equal temperament, and from these, a top-octave circuit divides the frequency by multiples of 2 to obtain the lower octaves, all perfectly tuned. These were found on early polyphonic instruments such as string machines and the like.

DCOs became widespread with the growth of the polyphonic synthesizers market, replacing VCOs to reduce pitch drift. With the advent of novel synthesis techniques such as frequency modulation, wavetable, sampling, and physical modelling, the interest in analog VCOs and DCOs was lost. In the 1990s a novel class of digital synthesizers brought back the interest for subtractive synthesis. Research work on *virtual analog* models for oscillators and filter were devised [4, 5], which mostly dealt with alias suppression or faithful recreation of the behavior of analog circuits. From that moment upwards, a great attention has been devoted to such a topic by the research community [6, 7].

Up to now, the literature has dealt mainly with two issues in virtual analog oscillators: aliasing in the generation of geometrical waveforms (e.g., sawtooth or square), as well as analysis and emulation of specific circuits. Since the inception of virtual analog, several techniques have established to generate geometrical waveforms, namely BLIT [4], BLEP [8] (and variations thereof), BLAMP [9], DPW [10] and wavetable synthesis. Other studies addressed the peculiar behavior of analog circuits and their departure from the ideal behavior. This is true for filters, often exhibiting a nonlinear behavior [11, 12], as well as for oscillators departing from the ideal waveform, as it is the case of the Moog sawtooth [13]. Investigating the specificity of existing oscillator designs allows the community to obtain useful information on the timbre of a known instrument, improve its emulation, and verify the applicability of existing aliasing suppression techniques on novel problems.

To the best of authors' knowledge, to date, the virtual analog community has overlooked the study of musical synthesizers' DCOs. DCOs are generally considered less appealing to the musician and the sound designer, because of their supposed precision. For the same reason, they are expected to be of less interest to the researcher as well, as they cause fewer issues in the modelling. Nonetheless, investigating DCO-based synthesizers may bring new insights on the character of these synthesizers, may improve our engineering knowledge, help understand its historical development and revamp some ideas.

This work is concerned with a class of DCO designs based on the Walsh-Hadamard transform. The use of such a transform was appealing for commercial products due to the tuning stability of digital integrated circuits. In the academic literature, the use of this transform for musical purposes was first proposed in a 1973 paper from Bernard Hutchins [14], who described a synthesizer system based on Walsh functions for generating waveforms and envelopes. In his work, Hutchins briefly hinted at its suitability to subtractive synthesis, but focused on additive generation of harmonic and nonharmonic tones. He also acknowledges a colleague,

C. Frederick, for suggesting the very idea of using the Walsh functions for music synthesis. Later works discuss waveform generation, circuit designs, frequency shifting and other purposes of this technique [15, 16].

The remainder of the paper is organized as follows. Section 2 provides a broad definition of DCO and discusses a class of DCOs based on the Walsh-Hadamard transform. Section 3 discusses the peculiar implementation of this method in the Welson Syntex synthesizer and a band-limited virtual analog implementation is proposed in Section 4. Finally, conclusions are drawn in Section 5.

2. DIGITALLY CONTROLLED OSCILLATORS

The definition of a DCO is rather fuzzy. In principle, any oscillator with pitch control acted by digital circuits is a DCO. The term "digital" should not bear confusion with a discrete-time domain oscillator, usually called numerically controlled oscillator (NCO). A NCO is directly implemented in the discrete-time domain, typically to generate a sine wave, and then fed to a Digital to Analog Converter (DAC). The term NCO is generally used in the electronics and telecommunications jargon, but any virtual analog oscillator conforms to that term, since the generation is all numerical.

The term Direct Digital Synthesis is somewhat related to NCO. This technique employs a NCO, often reading an arbitrary waveform from a RAM and generates an analog signal by a DAC. A DCO, instead, is not based on discrete-time algorithms or processing units, but simply works with digital electronics in the continuous-time domain. As an example, whereas a sawtooth VCO accumulates an electric charge into a capacitor and suddenly discharges when a threshold related to an electrical value is reached, a sawtooth DCO discharges at the reaching of a threshold of a digital counter integrated circuit (IC). The sound is, thus, still generated in the continuous-time domain, but the timing is controlled digitally by stable clocks and glue logic. Until digital signal processors, DACs and the production of custom digital VLSI chips became widespread in consumer electronics, the DCO approach was easier and more economical to implement into a synthesizer.

2.1. Walsh-Hadamard DCOs

Synthesizer waveforms based on the Walsh-Hadamard transform can be generated using digital electronics. Although, in principle, other waveforms such as sine waves can be synthesized [14], vintage synthesizers circuits focused on the sawtooth waveform relying on the assumption that it can be decomposed into a sum of square wave signals (with 50% duty cycle) weighted by Walsh-Hadamard transform (WHT) coefficients [17]. This transform reduces a real discrete signal to a weighted sum of orthogonal basis functions. These are the so-called Walsh functions, or Hadamard functions, depending on their ordering. In the following we consider the Walsh ordering.

The $M \times M$ matrix of Walsh functions up to order M is defined as

$$\boldsymbol{W}^{(M)} = \frac{1}{M-1} \cdot (-1)^{\sum_{m=0}^{M} k_m x_{m+1}} \tag{1}$$

where

$$k = \sum_{m=0} k_m 2^m \in \mathbb{N}_0, x = \sum_{m=1} x_m 2^{-m} \tag{2}$$

and both $k_m, x_m \in [0, 1]$.

As an example, the Walsh matrix of order 16 is shown in Figure 1. The WHT of a real-valued row vector $\boldsymbol{x}$ of length M is then

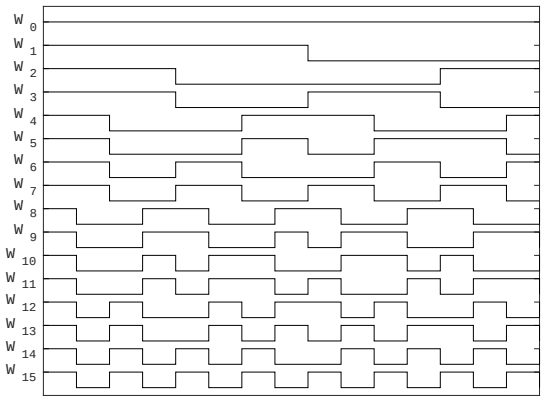


Figure 1: Walsh matrix of order $M = 16$, plotted row-wise. The signals constitute orthogonal bases that are employed in the Walsh transform. The first line corresponds to Walsh function of order 0 (DC), or W_0, while the last has index $M - 1$.

defined as

$$\boldsymbol{X} = \boldsymbol{x} \cdot \boldsymbol{W}^{(M)} \tag{3}$$

The coefficients $\boldsymbol{X}$ correspond to the weights of the orthogonal bases, that allow, thus, resynthesis of the original signal as

$$\hat{\boldsymbol{x}} = \boldsymbol{X} \cdot \boldsymbol{W}^{(M)}. \tag{4}$$

The WHT of a ramp of length M, i.e., a sawtooth of period M, has non-zero coefficients only for odd Walsh functions. For $M = 16$, e.g., only Walsh functions 1, 3, 7 and 15 are non-zero and have coefficients $1, 1/2, 1/4, 1/8$. Digital electronics allows to generate stable square wave signals at a low cost, making this solution viable to generate a sawtooth wave approximation.

3. THE WELSON SYNTEX AND THE ARP PRO SOLOIST

3.1. Historical Background

The Welson Syntex (1976) and the ARP Pro Soloist (1972) were monophonic preset synthesizers of the analog era, similar to the Moog Satellite (1973), the Thomas Synti 1055 and the ARP Soloist (1970). This breed of synthesizers was devised for easy operation on top of other polyphonic instruments such as organs and pianos and generally had limited flexibility. The presets were generally factory hardwired *patches* made of resistor networks that replaced potentiometers to provide fixed values to the synthesizer oscillators, filters, envelope generators, etc. A manual mode was also available where the user could tweak a few parameters regulated by potentiometers on the front panel.

The first successful preset synthesizer was the ARP Pro Soloist (1972), replacing the earlier ARP Soloist that had a limited success, mainly due to tuning stability issues. The ARP Pro Soloist developers devised stable oscillators based on digital electronics. This preset synthesizer had a good success, due to its price, making established brands like Moog eager to add similar products to their product catalogue. The Farfisa Syntorchestra, the Elka Soloist, the Korg 900PS, the Thomas Synti and the Moog Satellite are all similar from a user experience point of view. Many of these were

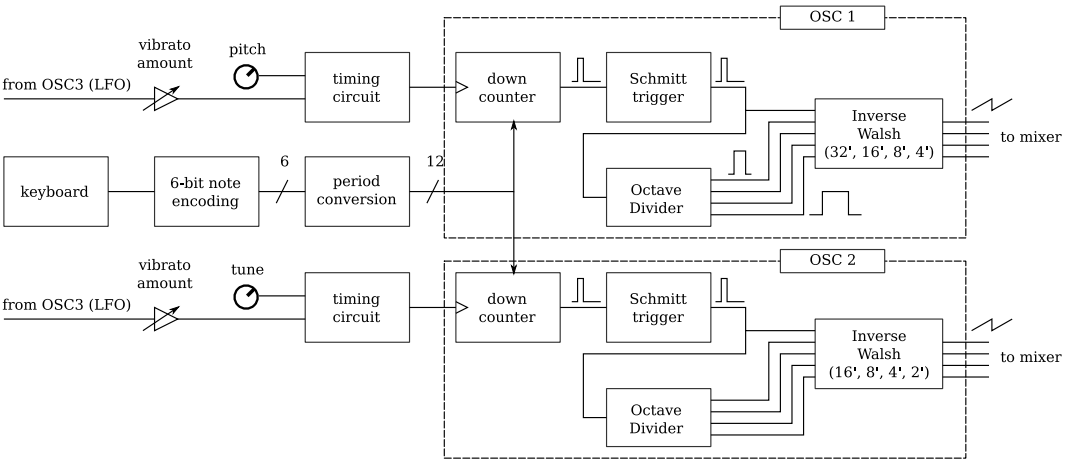


Figure 2: Overview of the Welson Syntex sawtooth generation mechanism.

produced in the Marche region, in Italy. The Farfisa had its main facilities in Camerano, Ancona province, while the Moog Satellite and the Thomas Synti were produced by EME (Elettronica Musicale Europea), a venture of musical instrument manufacturers namely Vox, Thomas and Eko located in Montecassiano, Macerata province. Similarly, the Welson Syntex was produced by Webo Electronics in Passatempo di Osimo, Ancona province. The Italian electronic instruments industry was able to provide know-how, materials and production facilities for these kinds of products.

While the Satellite and the Synti were all-analog, the Pro Soloist featured a digitally controlled oscillator to guarantee tuning stability. This is guaranteed by a resistor ladder DAC that converts ROM-stored values to a control voltage fed to a stable VCO generating a timing pulse train. The circuits, thus, convert a digital value into an analog voltage, and then an analog oscillator feeds a series of digital circuits to obtain several square waves. What is interesting about this oscillator design is that the sawtooth wave is obtained by a weighted sum of square waves, following the WHT DCO concept. This seems to be one of the earliest synthesizers employing the technique, resulting in a very good approximation of a sawtooth wave, as later discussed. Its development may have started a bit earlier than the work in [14], however we have no further information to assess whether the two approaches were developed independently.

The approach was later taken further by the Welson Syntex developers, that greatly reduced the complexity of the circuitry, only employing logic circuits between the keyboard and the oscillator. They also reduced the complexity of the oscillator, obtaining a far from perfect sawtooth wave but with a distinct character that is worth investigating.

3.2. Welson Syntex Synthesizer Architecture

The Syntex was developed in the years 1974–1976 by Mr. Menchinelli and Mr. Elio Bellagamba who were employees at Welson. Its sound engine follows the traditional VCO-VCF-VCA approach. It has two oscillators, differently from all other preset synthesizers mentioned above and a 4-pole transistor ladder filter with automatic keyboard tracking. It features a third oscillator, which actually is a LFO, and a noise generator. A touch of craze was introduced by a "Random Music" button, denoted by an atom icon, that generates random pitches at the rate of the LFO.

The use of digitally controlled oscillators modifies substantially the design of the keyboard and pitch control circuitry with respect to other synthesizers with a VCO. In vintage instruments, the keyboard is usually a resistor ladder that provides a linear voltage change that is later processed, depending on the oscillator. A voltage-controlled oscillator following the 1V/octave paradigm, e.g., rises its pitch exponentially by one octave in response to an increase of 1V to the input. The Syntex, instead, relies on timing and counter circuits without any analog voltage processing stage. An overview is provided in Figure 2. The keyboard is fed to a series of logic IC and gates that generates a 6-bit code. This is further processed by additional glue logic to obtain a 12-bit binary word that is the period duration in clock cycles.

3.3. Oscillator Design

The binary value is loaded into a chain of three 74191 synchronous counters ICs, hardwired to count downwards and arranged to act as a 12-bit counter. At the reaching of zero the binary value is loaded again and the counting starts again. The reset output of the last counter in the chain is also fed to a Schmitt trigger to generate a pulse Q. This pulse has a short active time, $50\mu s$ and a period 8 times shorter than the 8' output tone. The pulse is fed to a 7493 binary counter, with 4 outputs, that acts as an octave divider. Four octave signals are generated, Q_A, Q_B, Q_C, Q_D, yielding a total of five octaves including Q, although the signals Q_A, Q_B, Q_C, Q_D are 50% duty cycle square waves. The five signals are summed together with different ratios to obtain four footage output. These are all available and are blended together by front panel potentiometers. Oscillator 1 produces 32', 16', 8' and 4' tones, while Oscillator 2 produces 16', 8', 4', and 2' [1] In the following analysis we refer to the notation of Oscillator 1, although similar considerations apply to Oscillator 2.

The weighted sum is performed by means of an inverting summing operational amplifier for each footage output as shown in Figure 3. From the resistor values we can obtain the weights for

[1] The timing pulse Q of Oscillator 2 is run at twice the frequency of that of Oscillator 1. The presence of a tuning potentiometer with a wide range for Oscillator 2, however, allows to tune Oscillator 2 in unison to Oscillator 1.

each of the footages outputs:

$$S_{32} = -(Q_{16} + \frac{5}{11}Q_8 + \frac{10}{39}Q_4 + \frac{5}{41}Q_2) \quad (5)$$

$$S_{16} = -(Q_8 + \frac{5}{11}Q_4 + \frac{10}{39}Q_2 + \frac{5}{28}Q) \quad (6)$$

$$S_8 = -(Q_4 + \frac{5}{11}Q_2 + \frac{10}{27}Q) \quad (7)$$

$$S_4 = -(\frac{6}{5}Q_2 + \frac{4}{5}Q) \quad (8)$$

In the Syntex oscillators, four square waves are available, corresponding to Walsh functions of order 1, 3, 7 and 15. Signal Q, in general, is not orthogonal to the others, given the fact that its duty cycle varies with the note pitch. It cannot, therefore, be explained in the light of the WHT transform. The use of Q has been explained by one of its developers, Mr. Elio Bellagamba, as a trade-off between costs and benefits to add a fifth octave wave without resorting to additional components. The musical experts in the company approved this as the produced sound was more aggressive, Mr. Bellagamba recalls.

Regarding the weights of the WHT that are implemented in the Syntex, the chosen discrete resistor values depart from the theoretical values. Signal S_{32} is composed of all square wave signals, thus, it could approximate the sawtooth using a WHT of order 16 if the following resistor values would be employed:

$$S_{32} = -(Q_{16} + \frac{1}{2}Q_8 + \frac{1}{4}Q_4 + \frac{1}{8}Q_2). \quad (9)$$

However, weights do differ in the actual implementation, as seen in Eq. 5. This choice was motivated by the higher cost of precise resistors. Furthermore, the other footages deviate from the ideal WHT formulation as signal Q is not orthogonal to the others. In this case resistor values were agreed with the musical experts. The result of the four outputs are shown in Figure 4, and compared to the measured waveforms.

3.4. Comparison with the ARP Pro Soloist

The ARP Pro Soloist generates the approximated sawtooth waveform by summing 6 square waves generated from a top octave, according to the following:

$$S_{APS} = \sum_{o=1}^{6} \frac{P_o}{2^o}, \quad (10)$$

where P_1 is the square wave with fundamental frequency pitch. These are the weights as required per the WHT to approximate a sawtooth signal. The simplified diagram in Figure 6 shows how the sawtooth wave was obtained.

By employing 6 Walsh functions, the approximation of the sawtooth wave is very good, as shown in Figure 7. The only departure from the ideal sawtooth is the lack of every 64th harmonic, which can be considered negligible, especially for tones over E4 where the 64th harmonic is over the human hearing range. The difference between the ARP Pro Soloist tone and the S_{32} signal from the Welson Syntex is still hardly noticeable. Signals S_{16}, S_8 and S_4, however, depart significantly from the ideal sawtooth tone, making the Welson Syntex oscillator much more interesting to study and model with known virtual analog techniques.

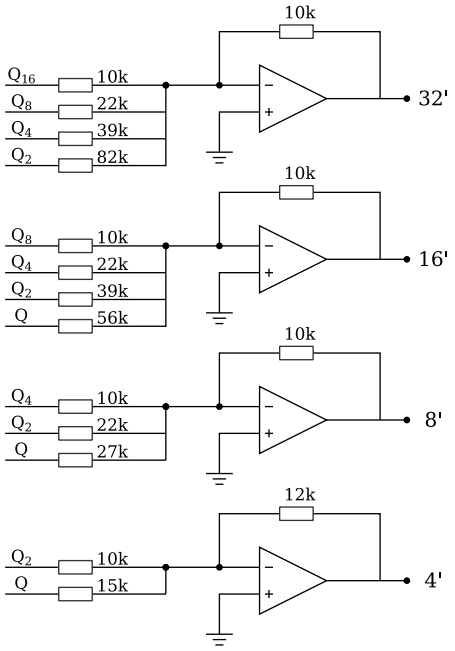


Figure 3: Summing different octave square waves to approximate a sawtooth wave in Welson Syntex Oscillator 1. The lower octave signals Q_2, Q_4, Q_8, Q_{16} are obtained from the top octave signal Q by a binary counter IC (7493) acting as frequency divider, so that Q_2 is half the frequency, Q_4 has one fourth of the frequency and so forth. Please note that the Oscillator 2 shares the same circuit, but the footage outputs are named $16'$ to $2'$ because the timing circuits run at twice the frequency of Oscillator 1, with typical values of the pitch and tuning potentiometers.

4. VIRTUAL ANALOG EMULATION

From the virtual analog side, the generation of the Welson Syntex sawtooth wave is not trivial as the discontinuities can be source of aliasing. In general, there are at least three different strategies to generate the signals seen above:

A: generate the Q_i signals and sum them according to Eqns. 5–8;

B: directly generate a staircase saw;

C: filtering a sawtooth with a comb filter[2].

We shall analyze each of them, their computational cost and their drawbacks. As far as alias suppression is concerned, we shall take the BLEP technique [8] as reference, truncated to 4 samples (2 backward, 2 forward).

Option A is very straightforward in principle. As a drawback, BLEP, or similar alias suppression techniques, need to be applied separately to all five signals. On the other hand, after each of the five signals is generated, all four output footages are obtained with little extra cost. In this case the computational cost per period is $2 + 4 + 8 + 16 + 32$ BLEP, plus the generation of the waves and the weighted sums (10 mul + 9 sums).

Option B allows two solutions: the wave can be directly synthesized by knowing at what point the steps happen and applying

[2]This is not valid for signals S_{16}–S_4 due to the presence of the non-orthogonal signal Q.

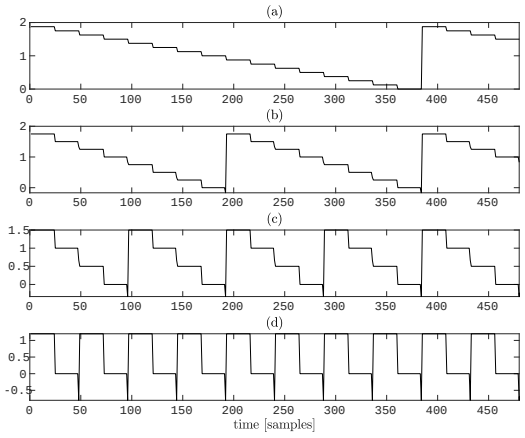


Figure 4: Simulation of the four sawtooth waves from the Welson oscillator: 32' (a) to 4' (d). The duty cycle of the Q signal has been set to 1%.

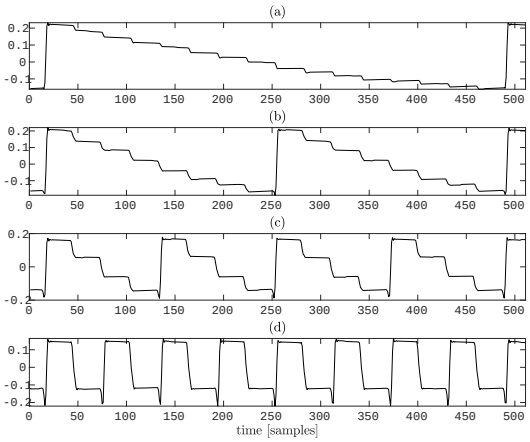


Figure 5: The four sawtooth waveforms sampled from the instrument output with the VCF completely open: 32' (a) to 4' (d). The effect of the sound card DC blocking stage affects the waveform shape by tilting the steps, while some light lowpass filtering due to parasitic components in the analog path slightly smooths the waveforms. This is not seen using an oscilloscope directly at the output of the components.

BLEP at all these discontinuities or by noting that the wave results from the sum of two sawtooth waves, one of larger period and one of shorter period. In other words, a staircase sawtooth S_K with K steps can be generated by the following

$$S_K[n] = S_L[n] - \frac{1}{K} S_H[n] \tag{11}$$

where

$$S_L[n] = 2(n\frac{f_0}{F_s} \mod 1) - 1 \tag{12}$$

$$S_H[n] = 2(n\frac{Kf_0}{F_s} \mod 1) - 1 \tag{13}$$

This requires K+1 BLEP per period, proving very inexpensive if only one of the footage outputs is to be generated. If all outputs need be generated, sawtooth waveforms of period T, $T/2$, $T/4$ and $T/8$ and $T/16$ are generated, costing $1 + 2 + 4 + 8 + 16$ BLEP per period. Finally the Q signal can be obtained by subtracting another sawtooth of period $T/16$, with a phase shift, in order to obtain the $50\mu s$ active time. The overall count is 47 BLEP, less than option A. Figure 8 shows a 2 kHz S_4 signal generated according to option B with and without BLEP.

Option C results from the observation that a sawtooth ramp with K stairs has a null in the spectrum every N harmonics, thus it can be shown that applying a comb filter designed to suppress these harmonics results exactly in a staircase sawtooth. A generalized recursive comb filter with both feedback and feedforward delay lines may be required to filter out the harmonics without affecting the rest of the content. Such a filter is characterized by the following difference equation:

$$y_n = b_0 x_n + b_L x_{n-L} + a_L y_{n-L} \tag{14}$$

where the length of the delay lines is L samples.

The computational cost of this filter is 2 sum and 3 mul per sample and some pointer arithmetics to update the delay lines, which should be added to the cost of 1 BLEP per period to generate the alias-suppressed sawtooth, interpolation of the comb for

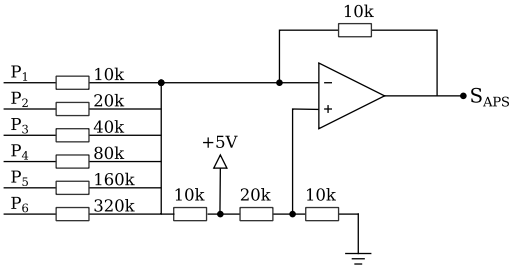


Figure 6: Sawtooth waveform generation in the ARP Pro Soloist. The input square wave signals $P_1, ...P_6$ are generated by a top octave divider. P_1 is the lowest generated octave, corresponding to the note fundamental.

precise tuning and, possibly the emulation of the Q pulse which should be added to emulate signals S_{16}, S_8, S_4. Overall, this solution is very inexpensive, despite the need to design the comb filter coefficients and to allocate memory for storing the delay lines values. The design of the comb filter may be problematic if, as it is the case with synthesizers, the pitch of the oscillator is modulated. For this reason, options A and B may still be preferable.

5. CONCLUSIONS

This work described the use of the Walsh-Hadamard transform for sawtooth signals generation and its application in early electronics synthesizers. Two of such oscillator designs, taken from historical synthesizers, have been discussed, and their differences are outlined. These oscillators are classified as DCOs. Therefore, the emulation of their waveforms is not demanding in terms of computational cost or circuit analysis. However, aliasing represents an issue. Several options for virtual analog emulation are described in the paper to obtain very similar results and their computational

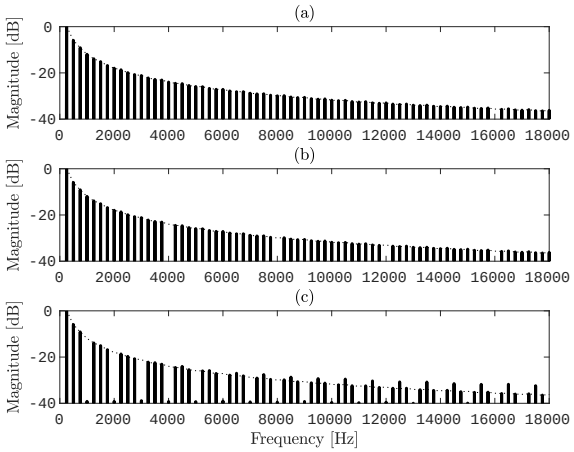


Figure 7: Comparison of ARP Pro Soloist and Welson Syntex sawtooth spectra for tones pitched at 250 Hz. In (a) the ARP Pro Soloist spectrum (solid line) is compared to the ideal sawtooth frequency envelope (dotted line) showing perfect matching (at least up to the 64th harmonic, at 16 kHz). In (b) the S_{32} Welson Syntex spectrum is shown to be identical to (a) except for the lack of each 16th harmonic. In (c) the S_8 signal shows a larger departure from the ideal sawtooth spectral envelope, with every 4th harmonic attenuated. These are not totally canceled due to signal Q not being a 50% duty cycle square wave.

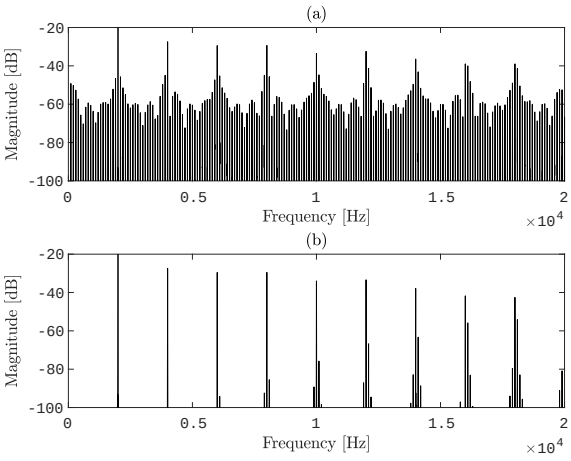


Figure 8: Magnitude spectrum of a 2 kHz S_4 signal generated at 44100 Hz without aliasing suppression (a) and with 4-samples BLEP (b) according to emulation option B.

cost is reported.

Several other synthesizers implemented the WHT for sawtooth generation, making the results of this work useful for the emulation of other historical synthesizers. An example of these is the Korg Poly800, a polyphonic synthesizer from the 1980s. Overall, we argue that DCO designs deserve more interest from the research community both for emulation goals and for preserving good engineering practices. The authors hope that this work could inspire others to study the solutions produced by past engineers, as these represent a rich heritage that may be valuable to progress the state-of-the-art.

6. ACKNOWLEDGMENTS

The authors would like to acknowledge the Acusmatiq-MATME association and Dr. Paolo Bragaglia for their valuable work in preserving the Italian synthesizer heritage and providing a Welson Syntex for circuit analysis. Many thanks to Mr. Elio Bellagamba for providing anecdotes regarding the development of the Syntex. We would like to acknowledge the reviewers for their detailed comments and for pointing out useful historical information.

7. REFERENCES

[1] Harald Bode, "A new tool for the exploration of unknown electronic music instrument performances," *Journal of the Audio Engineering Society*, vol. 9, no. 4, pp. 264–266, 1961.

[2] Harald Bode, "History of electronic sound modification," *Journal of the Audio Engineering Society*, vol. 32, no. 10, pp. 730–739, 1984.

[3] Robert A. Moog, "Voltage-controlled electronic music modules," in *Journal of the Audio Engineering Society*. Audio Engineering Society, 1965, vol. 13.

[4] Timothy Stilson and Julius O Smith, "Alias-free digital synthesis of classic analog waveforms.," in *International Computer Music Conference*, 1996.

[5] Timothy Stilson and Julius O Smith, "Analyzing the Moog VCF with considerations for digital implementation," in *International Computer Music Conference*, 1996.

[6] Vesa Välimäki and Antti Huovilainen, "Antialiasing oscillators in subtractive synthesis," *IEEE Signal Processing Magazine*, vol. 24, no. 2, pp. 116–125, 2007.

[7] Jyri Pakarinen, Vesa Välimäki, Federico Fontana, Victor Lazzarini, and Jonathan S Abel, "Recent advances in real-time musical effects, synthesis, and virtual analog models," *EURASIP Journal on Advances in Signal Processing*, vol. 2011, no. 1, 2011.

[8] Eli Brandt, "Hard sync without aliasing," in *International Computer Music Conference*, 2001.

[9] Fabián Esqueda, Vesa Välimäki, and Stefan Bilbao, "Rounding corners with blamp," in *Proc. Int. Conf. Digital Audio Effects (DAFx-16), Brno, Czech Republic*, 2016, pp. 121–128.

[10] Vesa Välimäki, Juhan Nam, Julius O Smith, and Jonathan S Abel, "Alias-suppressed oscillators based on differentiated polynomial waveforms," *IEEE Transactions on audio, speech, and language processing*, vol. 18, no. 4, pp. 786–798, 2010.

[11] Stefano D'Angelo and Vesa Välimäki, "Generalized Moog ladder filter: Part II–explicit nonlinear model through a novel delay-free loop implementation method," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 12, pp. 1873–1883, 2014.

[12] Federico Fontana and Marco Civolani, "Modeling of the EMS VCS3 voltage-controlled filter as a nonlinear filter network," *IEEE transactions on audio, speech, and language processing*, vol. 18, no. 4, pp. 760–772, 2010.

[13] Jussi Pekonen, Victor Lazzarini, Joseph Timoney, Jari Kleimola, and Vesa Välimäki, "Discrete-time modelling of the moog sawtooth oscillator waveform," *EURASIP Journal on Advances in Signal Processing*, vol. 2011, 2011.

[14] Bernard A. Hutchins, Jr., "Experimental electronic music devices employing walsh functions," *J. Audio Eng. Soc*, vol. 21, no. 8, pp. 640–645, 1973.

[15] E. Insam, "Walsh functions in waveform synthesizers," *J. Audio Eng. Soc*, vol. 22, no. 6, pp. 422–425, 1974.

[16] Bernard A. Hutchins, Jr., "Application of a real-time hadamard transform network to sound synthesis," *J. Audio Eng. Soc*, vol. 23, no. 7, pp. 558–562, 1975.

[17] K.G. Beauchamp, *Applications of Walsh and Related Functions - With an Introduction to Sequency Theory*, Academic Press, 1984.

[18] Vesa Välimäki and Antti Huovilainen, "Oscillator and filter algorithms for virtual analog synthesis," *Computer Music Journal*, vol. 30, no. 2, pp. 19–31, 2006.

[19] Jussi Pekonen and Vesa Välimäki, "Filter-based alias reduction for digital classical waveform synthesis," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2008, pp. 133–136.

NEURAL THIRD-OCTAVE GRAPHIC EQUALIZER

*Jussi Rämö and Vesa Välimäki**

Acoustics Lab, Dept. of Signal Processing and Acoustics
Aalto University
Espoo, Finland
jussi.ramo@aalto.fi

ABSTRACT

This paper proposes to speed up the design of a third-order graphic equalizer by training a neural network to imitate its gain optimization. Instead of using the neural network to learn to design the graphic equalizer by optimizing its magnitude response, we present the network only with example command gains and the corresponding optimized gains, which are obtained with a previously proposed least-squares-based method. We presented this idea recently for the octave graphic equalizer with 10 band filters and extend it here to the third-octave case. Instead of a network with a single hidden layer, which we previously used, this task appears to require two hidden layers. This paper shows that good results can be reached with a neural network having 62 and 31 units in the first and the second hidden layer, respectively. After the training, the resulting network can quickly and accurately design a third-order graphic equalizer with a maximum error of 1.2 dB. The computing of the filter gains is over 350 times faster with the neural network than with the original optimization method. The method is easy to apply, and may thus lead to widespread use of accurate digital graphic equalizers.

1. INTRODUCTION

The design of a graphic equalizer (GEQ) has advanced considerably in the past few years [1, 2]. Much research has been conducted to improve the design of both the cascade [3–8] and the parallel GEQs [9–13]. Currently it is possible to design either a cascade [2, 7] or a parallel GEQ [11–13] to have a maximum error of 1 dB, which is often considered sufficient for hi-fi audio. However, the design still requires optimization, which includes matrix operations, when the command gains are changed. This means that the accurate design of a GEQ needs large computational resources, if the parameters need to be updated quickly, such as in low-latency real-time applications.

We have recently proposed the idea of simplifying the calculation of filter gain optimization in a cascade graphic equalizer using a neural network [14], instead of the previous heavier method, which requires the calculation of DFT and matrix inversions. The training of the neural network becomes easy, when the network is presented with the pairs of command gains and the corresponding optimized gains obtained with an accurate design method. Then the task of the neural network is to imitate the nonlinear mapping, which the optimization method uses. This is simpler than using the neural network to learn to design the graphic equalizer by optimizing its magnitude response. It is also a different approach than the teaching of an equalizer using a neural network directly from an audio signal [15]. The training using the gain pairs was applied first to the cascade octave GEQ using a conventional perceptron with a single hidden layer [14].

The neural network introduces an error, when it approximates the nonlinear mapping. In [14] it was shown that a perceptron having twice as many hidden layer cells as input parameters was large enough for good approximation. The number of input parameters was 10 in the case of an octave GEQ, so 20 hidden layer cells were needed [14]. The approximation error can be kept smaller than 0.085 dB, which is sufficient for a maximum error of 0.7 dB for the GEQ itself [14].

In this paper, we apply the same idea to the design of a very common large GEQ, which has third-octave-octave bands. The third-octave GEQ has 31 bands to control the signal gain on narrow bands over the whole audio frequency range from 20 Hz to 20,000 Hz. This paper shows that the complexity of the problem is much larger than in the case of the octave GEQ, which has only 10 bands, and, consequently, a neural network with a single large hidden layer may not learn the mapping sufficiently accurately. We thus test a larger network structure having two hidden layers. It seems necessary that one of the hidden layers should contain twice as many nodes as the input layer.

The rest of this paper is organized as follows. Section 2 briefly recapitulates the design of a cascade third-octave GEQ, which will be approximated with the neural net. Section 3 explains the structure and training of the neural network. Section 4 presents validation and results of this work. Section 5 concludes this paper.

2. THIRD-OCTAVE GRAPHIC EQ DESIGN

An accurate design for a third-octave cascade graphic EQ (ACGE3) was proposed at the DAFx-17 conference [2]. The method is an extension of the corresponding accurate GEQ design for the octave case with ten bands [7]. Both designs take the user-set command gain values as inputs and then optimize the filter gains by evaluating the interaction between different band filters, which are second-order IIR filters. Each band filter is designed as a specific parametric equalizer, which is controllable at its own center frequency and at the center frequencies of its neighboring bands by defining the bandwidth in an unusual manner. This parametric equalizer is a modification of the design proposed by Orfanidis in his textbook [16].

The transfer function of the second-order band filter with user-

* This research is related to the "Nordic Sound and Music Computing Network—NordicSMC", NordForsk project number 86892.

Table 1: *Center frequencies f_c and bandwiths B for third-octave bands m.*

m	f_c (Hz)	B (Hz)	m	f_c (Hz)	B (Hz)	m	f_c (Hz)	B (Hz)	m	f_c (Hz)	B (Hz)
1	19.69	9.178	**9**	125.0	58.28	**17**	793.7	370.0	**25**	5040	2350
2	24.80	11.56	**10**	157.5	73.43	**18**	1000	466.2	**26**	6350	2846*
3	31.25	14.57	**11**	198.4	92.51	**19**	1260	587.4	**27**	8000	3502*
4	39.37	18.36	**12**	250.0	116.6	**20**	1587	740.1	**28**	10080	4253*
5	49.61	23.13	**13**	315.0	146.9	**21**	2000	932.4	**29**	12700	5038*
6	62.50	29.14	**14**	396.9	185.0	**22**	2520	1175	**30**	16000	5689*
7	78.75	36.71	**15**	500.0	233.1	**23**	3175	1480	**31**	20160	5570*
8	99.21	46.25	**16**	630.0	293.7	**24**	4000	1865			

* Manually adjusted bandwidths due to warping close to the Nyquist frequency.

set linear gain G_m is [2]

$$H_m(z) = b_{0,m}\frac{1 + b_{1,m}z^{-1} + b_{2,m}z^{-2}}{1 + a_{1,m}z^{-1} + a_{2,m}z^{-2}}, \tag{1}$$

where

$$\begin{aligned} b_{0,m} &= \frac{1+\beta_m}{1+G_m\beta_m}, \\ b_{1,m} &= -2\frac{\cos(\omega_{c,m})}{1+G_m\beta_m}, \qquad a_{1,m} = -2\frac{\cos(\omega_{c,m})}{1+\beta_m}, \\ b_{2,m} &= \frac{1-G_m\beta_m}{1+G_m\beta_m}, \qquad a_{2,m} = \frac{1-\beta_m}{1+\beta_m}, \end{aligned} \tag{2}$$

where

$$\beta_m = \begin{cases} \sqrt{\dfrac{|G_{B,m}^2 - 1|}{|G_m^2 - G_{B,m}^2|}}\tan\left(\dfrac{B_m}{2}\right), & \text{when } G_m \neq 1, \\ \tan\left(\dfrac{B_m}{2}\right), & \text{when } G_m = 1, \end{cases} \tag{3}$$

$$g_{B,m} = cg_m, \quad \text{where } c = 0.4, \tag{4}$$

$$\omega_{c,m} = 2\pi f_{c,m}/f_s, \tag{5}$$

with $g_{B,m} = 20\log(G_{B,m})$ and $g_m = 20\log(G_m)$. The sampling rate f_s used throughout this work is 44.1 kHz. Table 1 shows the center frequencies $f_{c,m}$ and bandwidths B_m of the third-octave bands used in this work.

One such second-order IIR filter is used per band, see Fig. 1(a), and all the 31 filters are cascaded to form the overall transfer function of the GEQ:

$$H(z) = \prod_{m=1}^{31} H_m(z), \tag{6}$$

as illustrated in Fig. 1(b). The gain factor G_0 in front of the graphic equalizer in Fig. 1(b) is the product of the scaling coefficients $b_{0,m}$ of the band filters:

$$G_0 = \prod_{m=1}^{31} b_{0,m}. \tag{7}$$

This way the multiplier related to the scaling factor $b_{0,m}$ can be removed from each band filter section, as can be seen in Fig. 1(a), which saves $M - 1$ multiplications in total [13].

2.1. Least Squares Optimization of Filter Gains

The optimal filter gains for the cascade graphic equalizer are solved using the least-squares method with the help of an interaction matrix [7]. The magnitude response of each equalizer filter with an example gain (17 dB is used in this work) is evaluated at the third-octave center frequencies and at their geometric means. These data are used to form the interaction matrix $\mathbf{B_0}$, which represents the leakage caused by each band filter to the other frequency points. Each row of the interaction matrix contains the normalized magnitude response of the m^{th} band filter sampled at the 61 prescribed frequencies. Because of the normalization, the value of the interaction matrix at the center frequency of the filter itself is always 1.0, since the magnitude response is divided by the filter gain. Furthermore, an additional iteration is used, which calculates another interaction matrix based on the filter gains obtained as the first LS solution. The second interaction matrix is used for further optimization [7]. This iteration round helps to restrict the approximation error in the magnitude response to be less than ± 1 dB, which was the design goal during the development of ACGE3 [2]. The

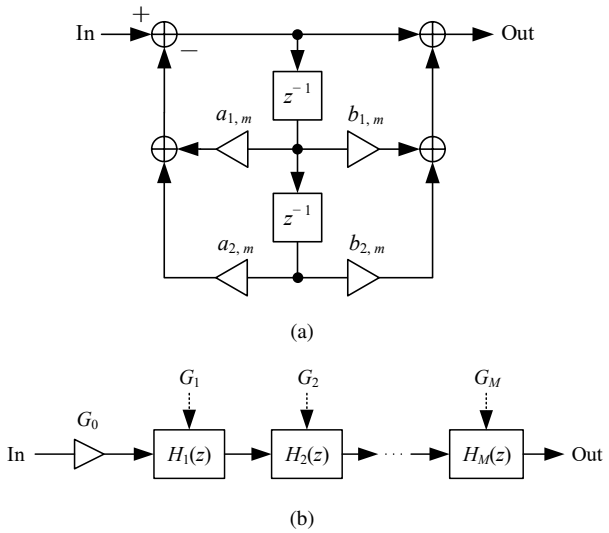


Figure 1: *(a) The second-order IIR filter structure of each band filter $H_m(z)$, and (b) the graphic equalizer structure containing a series of such filters and showing the filter gain controls, G_m. In the third-octave design, the number of filter sections is $M = 31$.*

matrix inversions cause this optimization method to be computationally costly.

3. NEURAL NETWORK

3.1. Training Data

The training data for the feedforward neural net is created using the ACGE3 design [2], which was reviewed in Sec. 2. With that design it is possible to create a huge number of input-output gain pairs, where the input values are the user-set command gains between −12 dB and 12 dB and the outputs are the optimized filter gains used in the underlying filter design, see Sec. 2.1.

For this work we created 1500 input-output pairs with random input gains using the ACGE3 algorithm. Six special gain configurations, known to be hard for GEQs, were included in the training data. They were two constants cases with all gains set to +12 dB and all gains at −12 dB, and two zigzag cases [2], as well as two hard configurations which are special zigzag settings[1].

3.2. Network Structure and Training

By definition, the third-octave EQ has 31 frequency bands, meaning it has 31 user-adjustable command gains. Thus the neural network has 31 nodes in its input layer, one for each band's gain setting. The ACGE3 design is implemented using one second-order IIR filter ber band, resulting in 31 optimized gain values for the EQ filters. Thus, the size of the output layer is also set to 31.

After initial training tests of the neural network it was decided that the network structure should be in the form of 31-J-K-31, i.e., it should have two hidden layers of size J and K. After training several different prototype neural networks we settled on the layer sizes of $J = 62$ and $K = 31$. Based on our previous experiments with the octave GEQ it is beneficial to have the size of the first hidden layer twice the size of the input layer [14]. Figure 2 shows the structure of the neural network, where g_1, g_2, ..., g_{31} are the user-set command gains in dB and $g_{\text{opt},1}$, $g_{\text{opt},2}$, ..., $g_{\text{opt},31}$ are the optimized filter gains in dB.

The neural network was trained using Matlab's `fitnet` function, which is a function-fitting neural network that is able to form a generalization of the input-output relation of the training data. Thus, after the network is trained, it is possible to use it to generate outputs for inputs that were not in the training dataset. The training algorithm was selected to be `trainbr`, a Bayesian regularization backpropagation algorithm [17]. It updates the weight and bias values according to the Levenberg-Marquardt (LM) optimization [18, Ch. 12]. The LM algorithm provides a desirable compromise between speed and guaranteed convergence of steepest descent [18], while the Bayesian regularization also ensures that the resulting network generalizes well by minimizing a combination of the squared errors and the network weights [17].

The training dataset was split into two sets, a training set (70% of the whole dataset) and test data (the remaining 30%). The test data is not used in the training per se, it is only used to monitor the performance of the model to unseen data during the training. The stopping conditions were set so that the training would continue until it is converged. With Bayesian regularization, a good indication of convergence is when the LM μ parameter reaches a high value (Matlab's default is 10^{10}). However, after 15,000 epochs the training was stopped, before reaching the maximum μ. One epoch takes approximately a minute calculate, when using 12 parallel CPUs, so it is quite time consuming to train the neural net. Thus, increasing the training time could still improve the accuracy of the proposed neural net.

[1] $\mathbf{g}_{\text{special}} = [12\ -12\ -12\ 12\ -12\ -12\ -12\ 12\ -12\ -12\ 12\ -12\ -12\ 12\ -12\ -12\ 12\ -12\ -12\ 12\ -12\ -12\ -12\ 12\ -12\ -12\ 12\ -12\ -12\ 12\ -12\ -12\ -12\ 12]^T$ and its opposite $-\mathbf{g}_{\text{special}}$.

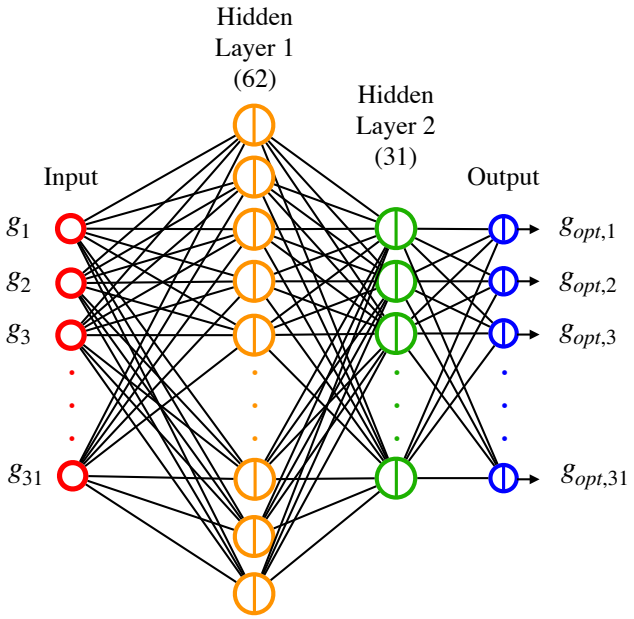


Figure 2: *Neural network presented in this work has 31 inputs and outputs, and two hidden layers.*

3.3. Final Neural Network

Figure 2 shows the resulting neural network while Fig. 3 depicts individual neurons in the hidden layers and the output layer. In Fig. 3, the leftmost neuron is the j^{th} neuron of hidden layer 1. Its inputs are the scaled user-set command gains g'_1, g'_2, ..., g'_{31}, since the neural network assumes that the input data has values between −1 and 1. Matlab does the scaling automatically during training using `mapminmax` function. The j^{th} neuron uses the weights $w^1_{j,1}$, $w^1_{j,2}$, ..., $w^1_{j,31}$ to scale the inputs, sums them and adds the bias value θ^1_j to the sum, and then uses the nonlinear sigmoid function σ to calculate the output o^1_j of for the neuron:

$$o^1_j = \sigma\left(\sum_{m=1}^{M=31} w^1_{j,m} g'_m + \theta^1_j\right), \tag{8}$$

where σ is equivalent to $\tanh(x) = 2/(1 + e^{-2x}) - 1$.

The output of a neuron in the second hidden layer is calculated in similar manner as in Eq. (8), but now the inputs are the outputs from every neuron in hidden layer 1. The output of the k^{th} neuron of hidden layer 2 is calculated as

$$o^2_k = \sigma\left(\sum_{j=1}^{J=62} w^2_{k,j} o^1_j + \theta^1_k\right), \tag{9}$$

and finally the m^{th} neuron in the output layer outputs the optimized

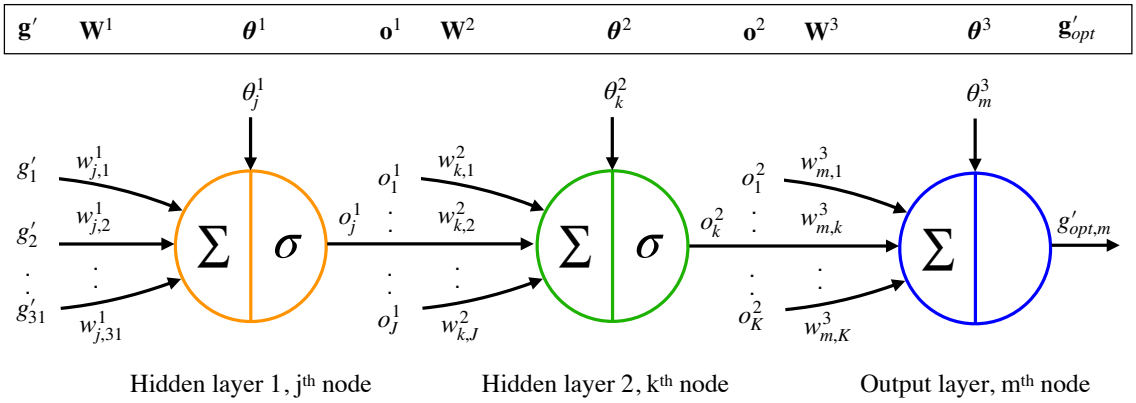


Figure 3: *Structure of individual neurons in the neural net. Cf. Fig. 2.*

gain for the m^{th} filter by calculating

$$g'_{\text{opt},m} = \sum_{k=1}^{K=31} w^3_{m,k} o^2_k + \theta^3_m. \tag{10}$$

Equations (8)–(10), which are used for running the neural network, can be written in matrix form as

$$\mathbf{g}' = 2 \cdot \frac{\mathbf{g} - \mathbf{x}_{\text{min}}}{\mathbf{x}_{\text{max}} - \mathbf{x}_{\text{min}}} - 1, \tag{11}$$

$$\mathbf{o^1} = \tanh(\mathbf{W}^1 \mathbf{g}' + \boldsymbol{\theta}^1), \tag{12}$$

$$\mathbf{o^2} = \tanh(\mathbf{W}^2 \mathbf{o}^1 + \boldsymbol{\theta}^2), \tag{13}$$

$$\mathbf{g'_{opt}} = \mathbf{W}^3 \mathbf{o}^2 + \boldsymbol{\theta}^3, \tag{14}$$

$$\mathbf{g_{opt}} = (\mathbf{t}_{\text{max}} - \mathbf{t}_{\text{min}}) \frac{\mathbf{g'_{opt}} + 1}{2} + \mathbf{t}_{\text{min}}, \tag{15}$$

where all the vectors and matrices correspond to those shown in the top part of Fig. 3. That is, Eq. (11) maps the user-set dB-gain values $\mathbf{g} \in [-12\ 12]$ to $\mathbf{g}' \in [-1\ 1]$, where all $x_{\text{min},m} = -12$ and $x_{\text{max},m} = 12$. Eq. (12) calculates the outputs $\mathbf{o}^1$ of hidden layer 1 based on $\mathbf{g}'$ by using weights $\mathbf{W}^1$, bias values $\boldsymbol{\theta}^1$, and the nonlinear transfer function $\tanh()$. Similarly, Eq. (13) uses all of the outputs $\mathbf{o}^1$ of hidden layer 1 to calculate the outputs of hidden layer 2 using a different set of weights $\mathbf{W}^2$ and bias values $\boldsymbol{\theta}^2$, including the nonlinear sigmoid function. The output layer takes the outputs $\mathbf{o}^2$ of hidden layer 2 as its inputs and weights them with $\mathbf{W}^3$ and adds the bias values defined in $\boldsymbol{\theta}^3$. Note that the output layer has no nonlinearity in it. Finally, the output layer of the neural network outputs the optimized gain vector $\mathbf{g}'_{\text{opt}}$ that have values between $[-1\ 1]$, which are then mapped to dB values based on the maximum and minimum values found in the training data targets, $\mathbf{t}_{\text{max}}$ and $\mathbf{t}_{\text{min}}$, respectively.

With these three weight matrixes, three bias vectors, and four output/input extreme values, it is possible to run the neural network for any arbitrary user command gain configurations (between -12 and 12 dB). We will provide all of the needed parameters to run the model.

4. RESULTS AND VALIDATION

In order to validate the actual performance and accuracy of the proposed third-octave neural GEQ (NGEQ3), we need to compare it against ACGE3, which was used to train the network. In order to do this, a validation dataset of 10,000 random command gain settings was created.

4.1. Computational Performance

The main purpose of substituting the ACGE3 filter optimization with a neural network is to computationally simplify the procedure so that Fourier transforms and matrix inversions are not needed. Although the designing and training of neural networks may take some time, running a trained neural network is often computationally quite straightforward. The neural network proposed in this work has 4929 parameters, consisting of the weights and biases, however, the main computation consists of only three matrix multiplications and additions, and two `tanh` calculations for vectors of sizes 62 and 31, see Eqs. (12)–(14).

To evaluate the computational time of the filter optimization, the validation dataset of 10,000 input command gains were optimized and the averages of the optimization times were recorded. The results are shown in Table 2. As can be seen, the proposed NGEQ3 optimization (13 μs) is much faster than that of the original ACGE3 (4661 μs). The ACGE3 optimization is heavier than the proposed NGEQ3 optimization, since it requires the calculation and inversion of the interaction matrix, during the iteration round, and several matrix multiplications. The interaction matrix is constructed by using the discrete-time Fourier transform which is used to evaluate the magnitude response of the band filters at 61 frequency points, consisting of the 31 third-octave center frequencies and their midpoints. The matrix inversion requires the computing of the Penrose-Moore pseudoinverse for the resulting 61-by-31 interaction matrix, which involves a matrix inversion and three matrix multiplications [7].

Table 2: *Comparison of computing times of the third-octave ACGE3 and proposed NGEQ3 methods, average of 10,000 trials. The fastest case in each column is highlighted.*

	Gain optimization	Coefficient update	Total
ACGE3 (DAFx-17)	4661 μs	57 μs	4718 μs
NGEQ3 (proposed)	**13 μs**	57 μs	**70 μs**

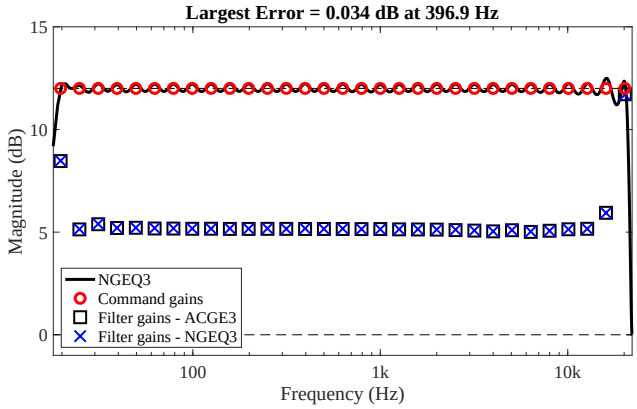


Figure 4: *Comparison of ACGE3 and NGEQ3 filter optimization, when all command gains are set to 12 dB.*

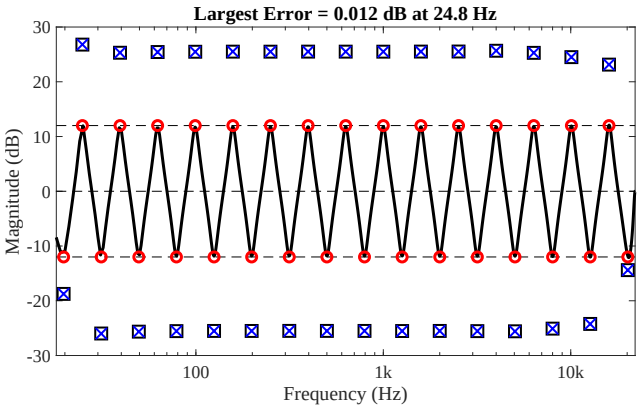


Figure 5: *Alternating ±12 zigzag command gain settings. See the legend in Fig. 4.*

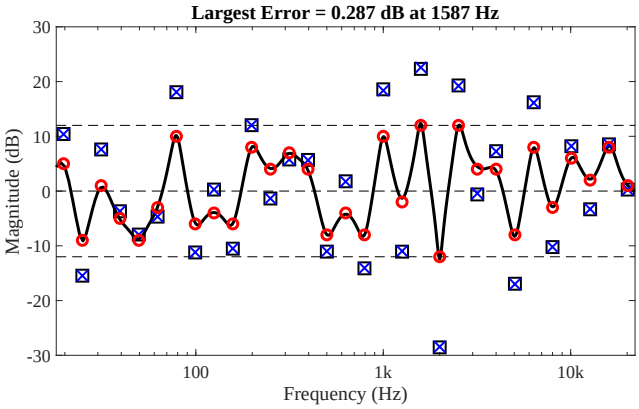


Figure 6: *Worst case scenario based on the validation dataset of 10,000 gain configurations. See the legend in Fig. 4.*

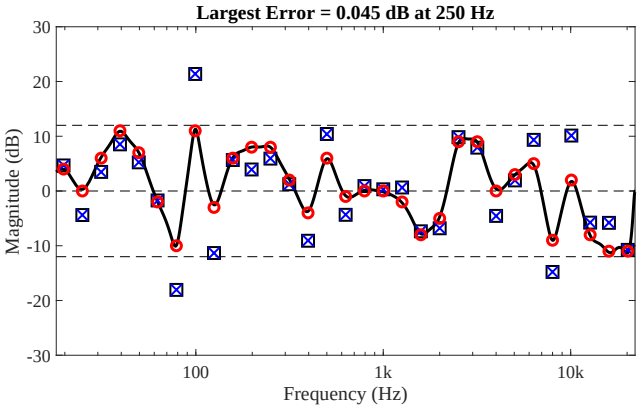


Figure 7: *Random command gain settings illustrating the mean accuracy of NGEQ3. See the legend in Fig. 4.*

Furthermore, the calculation of the filter coefficients takes approximately 57 μs, which is the same for both methods, meaning that the NGEQ3 gain optimization is even faster than the actual filter design.

4.2. Accuracy

While getting the implementation of the filter gain optimization faster can be essential to certain applications, the proposed method needs to be accurate in order to be useful. Figures 4 and 5 show magnitude responses of two example runs of the proposed neural network. Both cases are known to be challenging for a GEQ, and thus, both of these example cases were also included in the training dataset. Figure 4 shows a gain setting where all command gains are set to +12 dB, while Fig. 5 shows a gain setting with alternating commands at ±12 dB. In both figures, red circles (○) are the user-set command gains, black squares (□) are the ACGE3 optimized filter gains, blue crosses (×) are the optimized filter gains by the proposed NGEQ3, and the black line plots the magnitude response of the whole NGEQ3. Thus, in ideal case the crosses should lie inside the squares (⊠). Furthermore, the horizontal dashed lines plot the zero line, as well as the used maximum and minimum values ±12 of the command gains.

These two examples clearly illustrate the importance of filter gain optimization, since it is evident that the optimized filter gains (□ and ×) can be totally different than the actual user-set command gains (○). In Fig. 4, where all the gains are set to +12 dB, the optimized gains are considably smaller that the command gains, so that the final response settles at 12 dB. On the other hand, in the zigzag case in Fig. 5, the optimized gains are more than twice the value of the user-set command gains.

The accuracy of the proposed NGEQ3 was evaluated using the same validation dataset as above. The proper error evaluation is to compare NGEQ3 to ACGE3, since that is how the neural network was trained. That is, a perfect neural net with zero error would produce identical responses (and errors) with ACGE3. However,

Table 3: *Magnitude-response errors in dB at command point frequencies for 10,000 random gain settings.*

	ACGE		**Commands**	
	Max	Mean Max	Max	Mean Max
ACGE3 (DAFx-17)	–	–	1.1	0.53
NGEQ3 (proposed)	0.28	0.07	1.2	0.53

the absolute errors in respect to the user-set command gains are what eventually matters to the end user.

Table 3 shows the accuracy validation results. Each row in the table compares the absolute error, calculated at the defined command gain frequencies where the target can be specified, to ACGE3 and to the actual user-set command gains values. The largest error in NGEQ3 with respect to ACGE3 was 0.28 dB[2]. This case is plotted in Fig. 6, where the largest error occurs at 1587 Hz. However, the magnitude response still goes through the command gain setting (○), so there is no visible error for the end user.

Finally, Fig. 7 shows a random gain setting (not included in the validation or training datasets) to illustrate how small the error typically is. As can be seen in Table 3, the mean value of all the maximum errors (mean max) over the 10,000 sample validation dataset, when compared to ACGE3, was 0.07 dB, which is incredibly small. Furthermore, the last two columns of the table show the maximum and average of all of the maximum errors calculated against the user-set command gains. As can be seen, the overall maximum errors of ACGE3 and the proposed NGEQ3 are almost the same and close to 1 dB, whereas the mean max of the validation dataset is the same for both methods, approximately 0.5 dB.

5. CONCLUSIONS

This paper proposed to simplify the calculation of the gain optimization of a third-octave graphic equalizers using a neural network. This became possible after our team recently proposed an accurate graphic equalizer design method, which optimizes filter gains based on user-defined command gains. The filter gains are determined using a least-squares technique with one iteration and then, as all parameters are known, the IIR filter coefficients are computed using closed-form formulas. Thus, the main complication in the design has been the filter gain optimization.

In this work, the command gain-filter gain vector pairs obtained with the accurate design method are used as training data for a multilayer neural network. After the training, the LS optimization can be replaced with the neural network. The computing of the filter gains is over 350 times faster with the neural network than with the original LS method. The filter coefficients are finally computed using traditional closed-form formulas, which now takes more time than the gain optimization. The proposed method turns accurate graphic equalizer design easy and fast. The associated Matlab code is available online at `http://research.spa.aalto.fi/publications/papers/dafx19-ngeq/`.

While in this work the neural network was trained by using the input-output gain pairs from a previously known optimization algorithm, in the future, it could be interesting to explore the possibilities to train a neural network with a novel cost function based on the actual gains of a GEQ.

6. REFERENCES

[1] V. Välimäki and J. D. Reiss, "All about audio equalization: Solutions and frontiers," *Appl. Sci.*, vol. 6, no. 129/5, pp. 1–46, May 2016.

[2] J. Liski and V. Välimäki, "The quest for the best graphic equalizer," in *Proc. Int. Conf. Digital Audio Effects (DAFx-17)*, Edinburgh, UK, Sep. 2017, pp. 95–102.

[3] J. S. Abel and D. P. Berners, "Filter design using second-order peaking and shelving sections," in *Proc. Int. Computer Music Conf.*, Miami, FL, USA, Nov. 2004.

[4] M. Holters and U. Zölzer, "Graphic equalizer design using higher-order recursive filters," in *Proc. Int. Conf. Digital Audio Effects (DAFx-06)*, Montreal, Canada, Sep. 2006, pp. 37–40.

[5] J. Rämö and V. Välimäki, "Optimizing a high-order graphic equalizer for audio processing," *IEEE Signal Process. Lett.*, vol. 21, no. 3, pp. 301–305, Mar. 2014.

[6] R. J. Oliver and J.-M. Jot, "Efficient multi-band digital audio graphic equalizer with accurate frequency response control," in *Proc. Audio Eng. Soc. 139th Conv.*, New York, NY, USA, Oct. 2015.

[7] V. Välimäki and J. Liski, "Accurate cascade graphic equalizer," *IEEE Signal Process. Lett.*, vol. 24, no. 2, pp. 176–180, Feb. 2017.

[8] S. Prince and K. R. S. Kumar, "A novel Nth-order IIR filter-based graphic equalizer optimized through genetic algorithm for computing filter order," *Soft Comput.*, vol. 23, no. 8, pp. 2683–2691, Apr. 2019.

[9] S. Tassart, "Graphical equalization using interpolated filter banks," *J. Audio Eng. Soc.*, vol. 61, no. 5, pp. 263–279, May 2013.

[10] Z. Chen, G. S. Geng, F. L. Yin, and J. Hao, "A pre-distortion based design method for digital audio graphic equalizer," *Digital Signal Process.*, vol. 25, pp. 296–302, Feb. 2014.

[11] J. Rämö, V. Välimäki, and B. Bank, "High-precision parallel graphic equalizer," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 22, no. 12, pp. 1894–1904, Dec. 2014.

[12] B. Bank, J. A. Belloch, and V. Välimäki, "Efficient design of a parallel graphic equalizer," *J. Audio Eng. Soc.*, vol. 65, no. 10, pp. 817–825, Oct. 2017.

[13] J. Liski, B. Bank, J. O. Smith, and V. Välimäki, "Converting series biquad filters into delayed parallel form: Application to graphic equalizers," *IEEE Trans. Signal Process.*, vol. 67, no. 14, pp. 3785–3795, Jul. 2019.

[14] V. Välimäki and J. Rämö, "Neural graphic equalizer," *IEEE/ACM Trans. Audio Speech Lang. Process.*, submitted for publication in Jan. 2019.

[15] M. A. Martínez Ramírez and J. D. Reiss, "End-to-end equalization with convolutional neural networks," in *Proc. 21st Int. Conf. Digital Audio Effects (DAFx-18)*, Aveiro, Portugal, Sep. 2018, pp. 296–303.

[16] S. J. Orfanidis, *Introduction to Signal Processing*, Prentice-Hall, Upper Saddle River, NJ, 1996.

[17] F. D. Foresee and M. T. Hagan, "Gauss-Newton approximation to Bayesian learning," in *Proc. IEEE Int. Conf. Neural Networks (ICNN'97)*, Houston, TX, USA, Jun. 1997, pp. 1930–1935.

[18] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. De Jesús, *Neural Network Design*, Second edition, 2014. [E-Book] Available: http://hagan.okstate.edu/nnd.html.

[2] When the neural network was trained using a single layer with 62 nodes, the maximum error was 0.52 dB, which was considered to be too large for our purposes.

POTENTIOMETER LAW MODELLING AND IDENTIFICATION FOR APPLICATION IN PHYSICS-BASED VIRTUAL ANALOGUE CIRCUITS

Ben Holmes and Maarten van Walstijn

Sonic Arts Research Centre,
School of Electronics, Electrical Engineering, and Computer Science,
Queen's University Belfast,
Belfast, UK
{ bholmes02@qub.ac.uk, m.vanwalstijn@qub.ac.uk }

ABSTRACT

Physical circuit models have an inherent ability to simulate the behaviour of user controls as exhibited by, for example, potentiometers. Working to accurately model the user interface of musical circuits, this work provides potentiometer 'laws' that fit to the underlying characteristics of linear and logarithmic potentiometers. A strategy of identifying these characteristics is presented, exclusively using input/output measurements and as such avoiding device disassembly. By breaking down the identification problem into one dimensional, search spaces characteristics are successfully identified. The proposed strategy is exemplified through a case study on the tone stack of the Big Muff Pi.

1. INTRODUCTION

Virtual Analogue (VA) modelling is largely motivated by the heritage aim of recreating analogue effects in functional digital form. A core component of an analogue effect is its user interface, i.e. the controls available to the musician to design and fine-tune the timbral qualities of the effect. To create a complete VA model of a device, its user interface must therefore also be carefully recreated.

A ubiquitous element of the user interface is found in potentiometers, which are present in countless guitar effects, synthesisers, etc. These devices map a change in rotation (or other movement) to a change in a specified phenomenon. Often, available schematics omit potentiometer laws, requiring a method of determining them from the circuit. Further, ideal laws that are commonly used may not truly reflect the behaviour of real potentiometers.

The aim of this paper is to investigate, identify, and model such mappings with a view of incorporating the resulting potentiometer laws into VA circuit models. Physical modelling is a good match for the overall simulation in this case, as it preserves the circuit topology, meaning that potentiometer changes result in local rather than global system changes. This does not hold for black- and grey-box models [1, 2], which focus on deriving a model for a single setting. A solution would require interpolation across a large number of coefficient data sets to facilitate such potentiometer control, equivalent to the strategy used to model a systems' response to changes in input signal amplitude [3].

The main advantage of black box models is that they are derived entirely from input/output (I/O) measurements, preventing the need to disassemble a device and so avoiding any risk of damage to the device under test. Previously it was found that for physical models, values of the components in a circuit can be identified using only I/O measurements and positioning potentiometers at the extreme ends of their travel [4]. The objective of this work is to complete a VA circuit model by identifying the underlying characteristic of a potentiometer, again only using I/O measurements. A key aspect of this challenge is to determine suitable fitting functions.

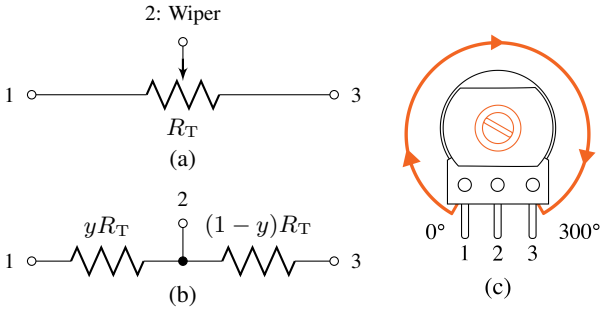


Figure 1: (a) *Annotated potentiometer schematic symbol.* (b) *Separation of the potentiometer symbol into the inter-terminal resistances as used to model its behaviour.* (c) *Rotary potentiometer diagram with labelled terminals, and the rotating shaft highlighted and range of rotation indicated, the shaft shown at 0 °.*

The rest of this paper is organised as follows: §2 investigates a variety of potentiometer laws and fits them to characteristics both from literature and also measurements of individual devices. §3 presents the identification strategy used to estimate potentiometer characterstics from I/O measurements, utilising the Big Muff Pi tone stack as a case study. §4 then presents the results of the characteristic identification from real I/O measurements, and finally §5 concludes the research and notes lines of future research. Companion material including MATLAB code and data sets are available online.[1]

2. MODELLING POTENTIOMETER CHARACTERISTICS

Potentiometers are a common component in musical circuits, used to provide a direct user interface. Some of the most common applications are for 'volume', 'tone', 'gain', etc. Illustrated in Figure 1, the potentiometer implements control over such quantities/phenomena by changing two resistances between three terminals relative to the position of its wiper which is actuated by the user.

The focus of this work is on how those resistances change with respect to a change in position of the wiper, referred to as its 'law'.

[1] https://bholmesqub.github.io/DAFx19/

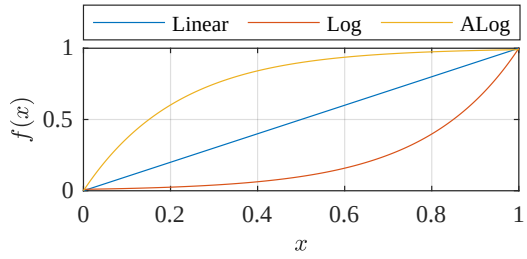


Figure 2: *Linear, logarithmic, and anti-logarithmic potentiometer tapers,* $\Delta_{\mathrm{dB}} = 40\,\mathrm{dB}$.

This law effectively maps the user's control to a change in the behaviour of the circuit, and is an essential feature of the user interface. Within the field of audio the most commonly encountered laws are linear and logarithmic, and will be the focus of the presented modelling and identification. Out of these two, it could be said that the logarithmic law is the most widely used, applied to map a linear control to logarithmic quantities such as loudness and frequency for application to volume controls and filter circuits.

2.1. Ideal laws

This section addresses 'ideal' potentiometer laws that are suitable for musical expression, but may not correspond exactly with how a physical potentiometer behaves electronically. An ideal potentiometer law is defined having a maximum total resistance R_{T} which is the sum of resistance between terminals 1 and 3 as illustrated in Figure 1. Each terminal and wiper are assumed to be perfect conductors i.e. have no resistance. Wiper position is defined here by the variable x which notes the rotation between terminals, and can be normalised such that $0 \leq x \leq 1$ which maps to a degree of total rotation usually between 0 and 300 degrees. The potentiometer law is defined as a function of the rotation, $y = f(x)$ which dictates the proportion of R_{T} that each resistor represents: $R_{1,2} = yR_{\mathrm{T}}$ and $R_{2,3} = (1-y)R_{\mathrm{T}}$, where subscript indicates terminal index. To reverse the orientation of the potentiometer the resistors change position between terminals, modelled as $y = 1 - f(x)$.

Ideal laws are shown in Figure 2, including an antilogarithmic law in addition to the previously mentioned linear and logarithmic laws. An ideal linear law can be defined as $f_{\mathrm{lin}}(x) = x$. A decibel ranged logarithmic function can be found by placing x in the exponent,

$$f_{\mathrm{log}}(x) = 10^{\frac{\Delta_{\mathrm{dB}}}{20}(x-1)}, \tag{1}$$

where Δ_{dB} is the desired range in decibels. This effectively maps linear rotation to a logarithmic law, a mapping which is used in other contexts for example in envelope design [5]. This law never reaches 0, which corresponds to $-\inf$ dB. Should a zero-value be desired the law can be translated and scaled, though it would then no longer be truly logarithmic.

The exponent operation can be computationally expensive depending on the system, and often approximations are offered such as a power law, e.g. $f(x) = x^4$ in [5]. Modern audio plug-in frameworks typically offer a variety of options to suit the needs of the developer [6].

An anti-logarithmic law is found by reflecting the logarithmic curve around $x = 0.5$ and $y = 0.5$, i.e. $f_{\mathrm{Alog}}(x) = \hat{f}_{\mathrm{log}}(x)$ where $\hat{f}(x) = 1 - f(1-x)$.

2.2. Laws from specifications

Though the ideal laws discussed in §2.1 may provide suitable control in software, these laws are unlikely to be encountered in real devices. In this section specifications are first investigated from a modelling perspective to determine which functions are suitable for capturing the behaviour of a real potentiometer law. Two functions are proposed to model the characteristics encountered in real devices, a $\tanh$ based function for broadly capturing multiple laws with a single function, and a piecewise function that aims to match the manufactured composition of the studied devices. The piecewise functions are then used to model measurements from real potentiometers that will then be used in the case study of the Big Muff Pi tone stack in §3.

To utilise data of potentiometer mapping characteristics shown in figures in the literature, online software for extracting data from images was used [7].

2.2.1. *An analytic multi-law function*

An authoritative source on potentiometers, 'The Potentiometer Handbook' [8] provides a reference for commonly occurring laws as well as the underlying manufacturing techniques behind them. The text refers to the Military Specification MIL-R-94B (now at revision G though characteristics are unaltered [9]), reproduced here in Figure 3. Each of these characteristics deviates from the ideal law, shown with gentle transitions towards the extreme ends of the functions. For the laws shown in Figure 3, a suitable non-piecewise function is found in $\tanh()$, parameterised with 4 free parameters,

$$f_{\tanh}(x) = t_1 \tanh(t_2 x + t_3) + t_4. \tag{2}$$

By introducing lower and higher limits on the potentiometer function y_{l} and y_{h}, this function can be constrained by requiring $f_{\tanh}(0) = y_{\mathrm{l}}$ and $f_{\tanh}(1) = y_{\mathrm{h}}$ resulting in

$$t_4 = y_{\mathrm{l}} - t_1 \tanh(t_3), \quad t_1 = \frac{y_{\mathrm{h}}}{\tanh(t_2 + t_3) - \tanh(t_3)}, \tag{3}$$

leaving free parameters t_2 and t_3. For a full-range law $y_{\mathrm{l}} = 0$ and $y_{\mathrm{h}} = 1$, though due to inherent terminal and wiper resistances these values will never be found from measurements of a real device.

The solid lines in Figure 3 show the fit of the $\tanh$ law to those from [8]. The $\tanh$ function with 2 free parameters in (2 - 3) were fit using the `fit` function from MATLAB 2018a's Curve Fitting toolbox, with initial parameters $t_2 = 1$ and $t_3 = -0.5$. For the anti-log characteristic the function was defined as $f^{\mathrm{Alog}}_{\tanh}(x) = \hat{f}_{\tanh}(x)$. The resultant function coefficients are shown in Table 1.

A maximum error of 4% is found in Figure 3 between the specified characteristic and fitted function for the logarithmic law, though most error falls between $\pm 2\%$. The larger error observed in the log law can be attributed to it reaching the y limits prior to the corresponding x limits. To mitigate this error, a piecewise function could be employed with the $\tanh$ function only fitting the central region, and transitioning to a different sub-function in the regions where the most error is encountered. This would however reduce the simplicity of the implementation, and other sub-functions may offer improved fit to real potentiometer device laws. The $\tanh$ function maintains applicability without additional sub-functions, offering a single function that can model each of the examined characteristics and providing this flexibility with only 2 parameters.

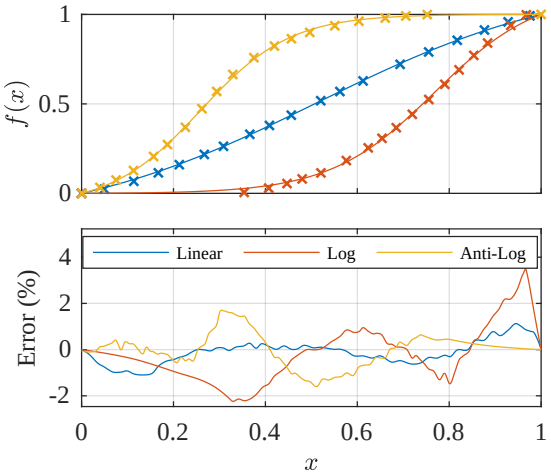


Figure 3: *Specified linear, logarithmic, and anti-logarithmic potentiometer tapers. The specified characteristic is marked by* ×*, and the fit using a* tanh *function by the solid lines. Error is shown in the lower plot.*

Table 1: *Fit parameters of the general* tanh *function to the potentiometer laws specified in [8]. Limits are given as* $y_\mathrm{l} = 0$ *and* $y_\mathrm{h} = 1$.

t	Linear	Log	ALog
t_1	0.701	0.566	0.536
t_2	1.790	4.400	5.113
t_3	−0.919	−3.380	−3.787
t_4	0.508	0.564	0.535

2.2.2. *Piecewise linear/cubic functions*

The specific potentiometers investigated within this work are of the brand Alpha, a popular brand in the building of guitar effects-pedals. Specifications for the potentiometer laws are found on their website [10]. Again the tanh function could be employed to model both linear and logarithmic laws for these specifications, however after an initial study the error peaked at 8% at the same region as observed in Figure 3, though a figure of this result is omitted for brevity. To reduce this error a piece-wise function would need to be used to capture the end-regions where the function gradient is zero. In the case that a piecewise function is required, it is a logical step to test which functions can provide the optimal fit.

The logarithmic characteristics from [10] are reproduced in Figure 4. Each law is specified by the percentage of total resistance between terminals 1 and 2 at 50 % rotation. Upon inspection of this set of characteristics there appear distinct sections with constant gradients, joined via smooth transitions. This property can be exploited through the use of a piecewise function containing linear sub-functions. Defining a linear sub-function with local lower and higher limits – $y_\mathrm{ll} = \bar{f}_\mathrm{lin}(x_\mathrm{ll})$ and $y_\mathrm{lh} = \bar{f}_\mathrm{lin}(x_\mathrm{lh})$ – results in the expression

$$\bar{f}_\mathrm{lin}(x) = \frac{y_\mathrm{lh} - y_\mathrm{ll}}{x_\mathrm{lh} - x_\mathrm{ll}}(x - x_\mathrm{ll}) + y_\mathrm{ll}, \quad \bar{f}'_\mathrm{lin}(x) = \frac{y_\mathrm{lh} - y_\mathrm{ll}}{x_\mathrm{lh} - x_\mathrm{ll}}. \quad (4)$$

The derivative of the linear sub-function is important here as there are no apparent jumps in gradient in any of the characteristics, and so the gradient of the sub-functions must match at the transitional values of the piecewise function. To achieve a match in derivatives at transitional points, 4 free parameters would be required, two for each transitional point, matching value and derivative. One such sub-function that offers this is found in a general cubic polynomial, expressed as

$$\bar{f}_\mathrm{cub}(x) = c_4 x^3 + c_3 x^2 + c_2 x + c_1, \quad (5)$$

where $c_1 - c_4$ are the cubic coefficients used to fit the sub-function $\bar{f}_\mathrm{cub}(x)$. To find values for $c_1 - c_4$ the following set of equations must be solved,

$$\bar{f}_\mathrm{cub}(x_\mathrm{ll}) = c_4 x_\mathrm{ll}^3 + c_3 x_\mathrm{ll}^2 + c_2 x_\mathrm{ll} + c_1, \quad (6)$$

$$\bar{f}_\mathrm{cub}(x_\mathrm{lh}) = c_4 x_\mathrm{lh}^3 + c_3 x_\mathrm{lh}^2 + c_2 x_\mathrm{lh} + c_1, \quad (7)$$

$$\bar{f}'_\mathrm{cub}(x_\mathrm{ll}) = 3c_4 x_\mathrm{ll}^2 + 2c_3 x_\mathrm{ll} + c_2, \quad (8)$$

$$\bar{f}'_\mathrm{cub}(x_\mathrm{lh}) = 3c_4 x_\mathrm{lh}^2 + 2c_3 x_\mathrm{lh} + c_2. \quad (9)$$

An explicit solution of this set of equations is available on the supporting online content, but is omitted here for brevity.

The choice of the linear-cubic piecewise function aims to build-in the specified behaviour of the manufacturer's characteristics. By matching the underlying structure of potentiometer characteristics, the ability to produce an accurate law from a reduced/incomplete set of measurements is improved, without needing a higher number of measurements to interpolate between.

From the specified or measured potentiometer characteristics a set of x values can be found at which the function transitions from linear to cubic (either through direct visual inspection or inspection of the gradient of the law). At these points the corresponding y values can be found by interpolating between the available data points of the characteristic, yielding a set of points that make up the piecewise function transitions.

In Figure 4, logarithmic characteristics from [10] are shown with their piecewise fit. The transitional points are approximately equal between each function, meaning that only one set of x values was required to fit the full set of characteristics. These values of x and the corresponding y values are shown in Table 2 where they are marked with their respective sub-function. A total of 7 piecewise sections were needed to match the characteristic specified by the manufacturer. The solid line in Figure 4 shows the fit of the piecewise function to the specification, the error between the two shown in the plot beneath. A peak error of 2% is found for the 15A law, with most error for each law falling within ±1%.

Figure 5 shows the characteristics as given for 'linear' laws from [10]. A similar level of accuracy can be achieved using only 5 piecewise sections as opposed to the 7 used for the logarithmic characteristics. The piecewise transitional points for each of these laws varies significantly, necessitating an individual search for the characteristic x values at which the transitions occur, found using an optimisation algorithm.

From MATLAB's Optimisation toolbox the Nelder-Mead algorithm was selected for an easy-to-implement, derivative-free algorithm to minimise the error of the function to the characteristic by changing transitional values of x [11]. Four x variables were gathered into $\boldsymbol{\theta}_x$ to use as parameters, excluding those at 0 and 1. The corresponding estimated values of $\hat{f}(x, \boldsymbol{\theta}_x)$ were then found through interpolating between the available values of y, and a complete piecewise function was assembled. The employed objective function,

$$\xi_x(\boldsymbol{\theta}_x) = \frac{1}{\eta}\sum_{n=0}^{N_\mathrm{s}} \left(y_n - \hat{f}(x_n, \boldsymbol{\theta}_x)\right)^2, \quad \eta = \sum_{n=0}^{N_\mathrm{s}} y_n^2, \quad (10)$$

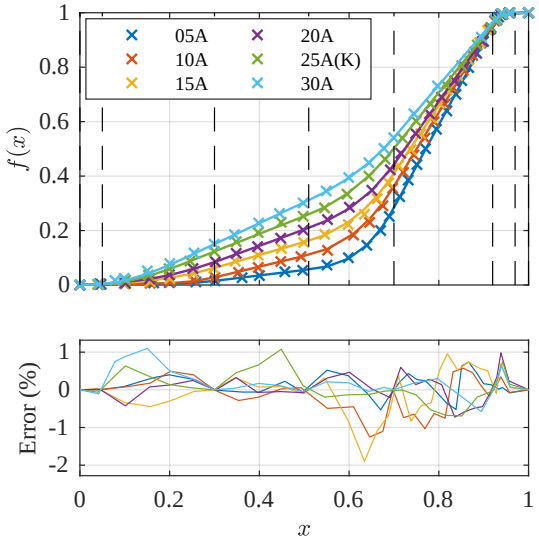


Figure 4: *Logarithmic Alpha characteristics (× marks), and fit using linear-cubic piecewise functions (solid lines). Characteristics are noted by the manufacturer's code which refer to the resistance percentage at 50% rotation. Error is shown in the lower plot. Vertical dashed lines mark the points at which the piecewise function sections change.*

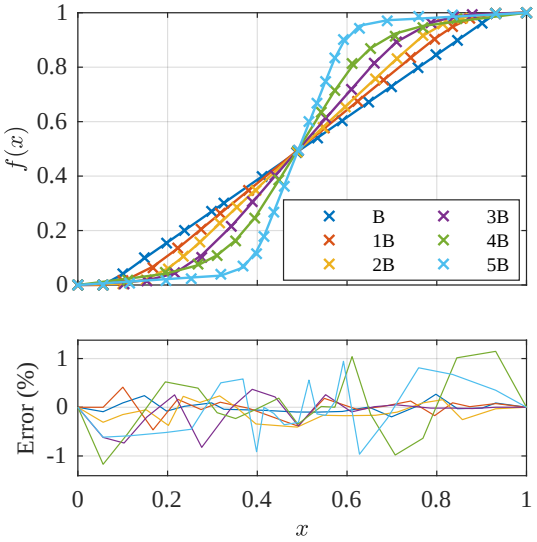


Figure 5: *Linear Alpha characteristics from (× marks), and fit using linear-cubic piecewise functions (solid lines). Characteristics are noted by the manufacturer's code. Error is shown in the lower plot. Transitions are omitted for this figure as they each fall at different values.*

enumerates the sum-squared error between the measured points of $f(x_n) = y_n$ and those found using the piecewise function with estimated transitional points $\hat{f}(x_n, \boldsymbol{\theta}_x)$. A normalisation factor η is applied to ensure that between data sets with different numbers of elements, the enumerated error would be comparable, allowing comparison between the results of the optimisation using different data sets.

Constraints are applied directly to the objective function, returning $\xi_x = 10^3$ if the constraints are not satisfied. These constraints prevent x values from exceeding domain limitations, i.e. $0 \leq \theta_x \leq 1$, and that they are incremental in value, i.e. for the jth element $\theta_x^j < \theta_x^{j+1}$.

Convergence tolerances were set at a change in value beneath 10^{-8} for both $\boldsymbol{\theta}_x$ and $\xi_x(\boldsymbol{\theta}_x)$. The resulting optimised x and corresponding y values are omitted here for brevity but can be found on the companion webpage. The solid line in Figure 5 represents the fitted piecewise function for each law, with the error between fit and specification shown in the lower plot. Peak error is approximately 1.2% with most contained within $\pm 1\%$.

With suitable functions that match the specified potentiometer laws to within 2% error, the following step is to apply this function to model the characteristic of real potentiometer devices.

2.3. Measured potentiometer characteristic

Several potentiometers were purchased from a local distributor of components for the DIY construction of guitar effects pedals. This source was selected to ensure that the potentiometers would be intended for the use in guitar effects pedals, and that they could be used to determined which laws from the presented sets in Figures 4 and 5 are used in potentiometers popular among effects-builders. The purchased potentiometers were specified to have $R_\mathrm{T} = 100\,\mathrm{k}\Omega$ ($\pm 20\%$) as this is the value of the potentiometer used in the Big Muff Pi tone stack, further discussed in §3.

Table 2: *Optimised transition points of the general cubic-linear piecewise function, fit to the logarithmic potentiometer characteristic specified in [10]. Values are rounded to 3 decimal places. 'Law' column is offset to indicate which law corresponds to each set of end-points.*

Law	x	y					
		05A	10A	15A	20A	25KA	30A
	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Lin.	0.050	0.003	0.003	0.003	0.004	0.002	0.004
Cub.	0.300	0.015	0.028	0.063	0.084	0.123	0.151
Lin.	0.510	0.057	0.111	0.162	0.210	0.259	0.311
Cub.	0.700	0.284	0.363	0.410	0.443	0.501	0.542
Lin.	0.920	0.954	0.959	0.958	0.952	0.954	0.965
Cub.	0.970	0.999	0.999	1.000	1.000	0.999	1.000
Lin.	1.000	1.000	1.000	1.000	1.000	1.000	1.000

To measure the potentiometer characteristic, a measurement jig was designed with markers at 15 degree angles around a central hole where the potentiometer was fixed. The markers were placed using computer aided design/manufacturing, the jig made from FR-4 with plated copper as used in circuit-board manufacturing techniques.

Direct measurements presented here are only cursory to provide an approximate law with which to compare to those found through the following identification from I/O measurements. The potentiometer was rotated by hand to line the knob indicator to each marker, while the resistance between adjacent terminals was measured continuously with an LCR meter. Measuring the resistance at 15 degree increments over a total of 300 degrees of travel yielded 21 measurements. The obtained characteristics can be considered as suitably representative test data even in the presence of possible measurement errors, including human error when performing the manual wiper rotation and heating from continuous

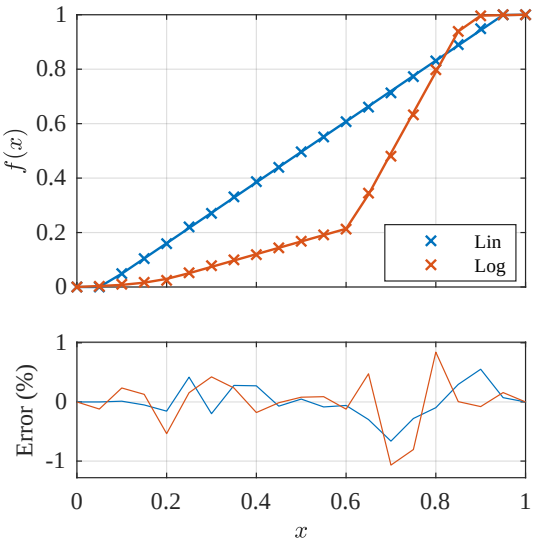


Figure 6: *Measured (× marks) and fit linear-cubic (solid lines) potentiometer laws of 1 linear and 1 logarithmic potentiometer. Error is shown in the lower plot. The laws of the two potentiometers fit with that of 'B' from Figure 5 and '15A' from Figure 4.*

Table 3: *Optimised transition points of the general cubic-linear piecewise function, fit to measurements taken of a linear and logarithmic potentiometer. Values are rounded to 3 decimal places. 'Law' column is offset to indicate which law corresponds to each set of end-points.*

	Linear		Log	
Law	x	y	x	y
Lin.	0.000	0.000	0.000	0.000
Cub.	0.050	0.000	0.071	0.005
Lin.	0.093	0.041	0.239	0.045
Cub.	0.951	1.000	0.603	0.215
Lin.	0.951	1.000	0.659	0.368
Cub.	1.000	1.000	0.850	0.939
Lin.	-	-	0.908	0.997
	-	-	1.000	1.000

driving from the LCR meter. Other physical parameters exist for the potentiometer in addition to its law, such as terminal resistance, but these were found to be of a magnitude that was impossible to measure with available equipment, and therefore likely insignificant to the identification.

Figure 6 shows the measured characteristics of both a linear and a logarithmic potentiometer, and the fit of the piecewise linear-cubic function to the measurements. A notable deviation was found from the logarithmic tapers of Figure 4, with the transition to the zero-gradient section at the maximum of the function occurring at a lower value of x. Therefore the optimisation approach used to fit the linear characteristics of Figure 5 was applied, i.e. the x positions found by optimising them and finding the corresponding values of y through interpolation of the measurement. The result is a good fit to the measurements with a peak error just over 1%. Final transitional values of the piecewise function are shown in Table 3.

3. IDENTIFYING POTENTIOMETER CHARACTERISTICS FROM I/O MEASUREMENTS

Having fit laws to characteristics obtained from both linear and logarithmic potentiometers, sufficient information is available to validate results from the law identification strategy exclusively using input/output measurements, presented in this section. The case study chosen to demonstrate this strategy is the tone stack from the Big Muff Pi, informed from the description in [12], with the schematic shown in Figure 7. In simple terms, the potentiometer in the Big Muff Pi tone stack blends between a low pass formed between R_1 and C_2, and a high pass filter formed by C_1 and R_2. Both linear and logarithmic potentiometers will be used to demonstrate the capability of the identification strategy to succeed independent of potentiometer law.

The tone stack was assembled on a breadboard to enable direct measurement of each component prior to the measurement of the circuit's transfer function. Use of a breadboard also facilitates the measuring of both linear and logarithmic potentiometers to demonstrate the identification procedure for the two most common laws. Specified and directly measured component values can be found in Table 4.

Only one potentiometer is present in the circuit. The presented method should allow for independent identification of multiple potentiometers by setting all but the potentiometer of interest at a known position, at 0 or 300 degrees rotation, resulting in $N \times M$ independent identifications where N is the number of positions per potentiometer and M the number of potentiometers. This property cannot be demonstrated using this circuit, but can be inferred as only one variable, a single potentiometer position, would be changed in between each measurement.

To begin identifying the potentiometer laws from input/output measurements, one must start with estimates of the component values, obtained from schematics or other identification strategies. Should the component values be highly accurate, the input/output measurements should theoretically be matched when the potentiometer is at either extreme of its rotational travel. At these points the values of y are assumed known. The accuracy of the fit achieved at these limits will provide some indication of how accurately the law can be retrieved.

The selection of a linear case study provides several desirable traits: the circuit model can be represented using a transfer function which requires only a few data points to capture a large frequency range, and issues of nonlinear behaviour such as aliasing are avoided. The tone stack is modelled as a transfer function derived using Modified Nodal Analysis [13]. The resultant function is of the form

$$H(j\omega) = \frac{b_3(j\omega)^3 + b_2(j\omega)^2 + b_1 j\omega}{a_4(j\omega)^4 + a_3(j\omega)^3 + a_2(j\omega)^2 + a_1(j\omega)^1 + a_0}. \tag{11}$$

Complete coefficients are omitted due to their complexity, but MATLAB code for calculating the transfer function is available on the companion webpage. Figure 8 shows the simulated amplitude responses of the Big Muff Pi's tone stack with the potentiometer at each end of its travel: presuming these responses to be accurate the challenge then lies in measuring intermediate responses and identifying the corresponding values of $f(x)$.

Table 4: *Component values of the Big Muff Pi tone stack: specified values and the values directly measured from the circuit using an LCR meter.*

	Unit	Specified	Measured
R_1	kΩ	39.000	39.080
R_2	kΩ	22.000	21.950
R_T	kΩ	100.000	94.940 (lin) / 98.140 (log)
R_o	kΩ	100.000	99.978
C_1	nF	4.700	4.698
C_2	nF	10.000	9.470
C_3	nF	100.000	98.010
C_d	pF	100.000	33.227

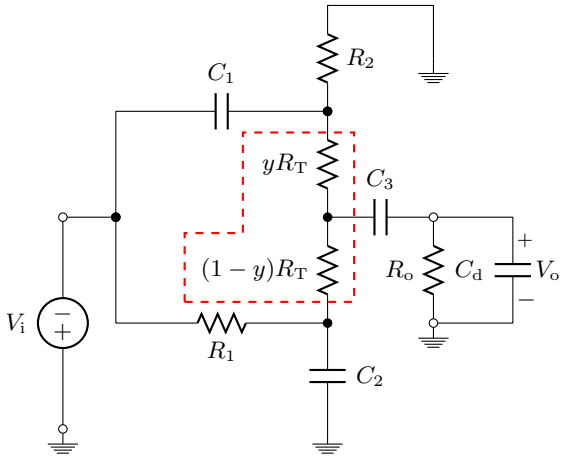


Figure 7: *Schematic of the Big Muff Pi tone stack with potentiometer marked by red line. C_d is the input capacitance of the measurement equipment used.*

3.1. Identification strategy

The identification strategy proposed in this work identifies a potentiometer's characteristic from input/output measurements of a circuit. A series of optimisations are performed, only operating on a single value of x at a time. Values of $y = f(x)$ are estimated at each point, thus identifying the potentiometer characteristic. To perform such optimisations, an objective function is required which compares the measurements of the circuit to the equivalent data from the model, enumerating the error between circuit and model.

Considering the linear case exhibited by the Big Muff Pi tone stack, the input/output measurements can be condensed into the form of a transfer function. Due to the limitations of the measurement equipment (further discussed in Section 3.2) only the amplitude response of the transfer function is used, with phase information discarded. From this information the objective function was constructed, for an estimated value of $y = f(x)$,

$$\xi_\mathrm{io}(y) = \frac{1}{\sum_{n=0}^{N_\mathrm{s}-1}|H(j\omega_n)|^2}\sum_{n=0}^{N_\mathrm{s}-1}\left(|H(j\omega_n)| - |\hat{H}(j\omega_n, y)|\right)^2, \tag{12}$$

where the operator $|\cdot|$ indicates the magnitude of a complex value. Frequencies of the transfer function are specified using ω_n, where n indicates index of the discrete frequency selected to be included in the measurement data.

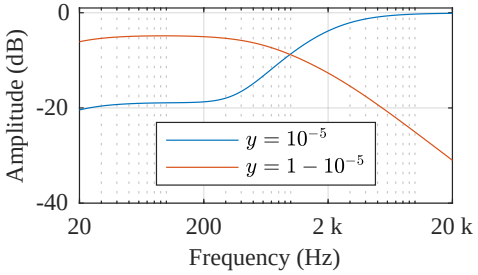


Figure 8: *Amplitude responses of the Big Muff Pi tone stack with the potentiometer at extreme ends of its rotational travel.*

The optimisation algorithm chosen to minimise the value of $\xi_\mathrm{io}(y)$ was that of `fminsearch` from MATLAB's optimisation toolbox, specifically the Nelder-Mead simplex algorithm. Convergence conditions were again set to be a change in y or ξ_io less than 1×10^{-8}. Initial experiments using simulated I/O measurements demonstrated successful identification of the potentiometer characteristics to within $10^{-5}\%$ of the accurate value.

3.2. Measurement procedure

Among various possible valid measurement approaches to find the transfer function of the Big Muff Pi tone stack, a multi-sine excitation signal was chosen, expressed as

$$V_\mathrm{i} = V_\mathrm{p}\sum_{d=d_\mathrm{l}}^{d_\mathrm{u}} A_d \cos(2\pi d f_0 n T + \phi_d), \quad n = 0, \ldots, N_\mathrm{s}. \tag{13}$$

The integer values of d are limited to contain sinusoidal components at multiples of $f_0 = f_\mathrm{s}/N_\mathrm{s}$ between the lower and upper limits d_l and d_u. Phase terms ϕ_d are specified as in [14], i.e.

$$\phi_d = -2\pi\sum_{l=1}^{d-1}(d-l)A_d, \quad d = d_\mathrm{l}, d_\mathrm{l}+1, ..., d_\mathrm{u}. \tag{14}$$

which requires that $1 = \sum_{d=d_\mathrm{l}}^{d_\mathrm{u}} A_d$. Individual amplitude components allow a frequency-domain weighting to be applied which can be used to maximise the signal-to-noise ratio, but in this case were set to be $A_\mathrm{d} = 1/(d_\mathrm{u} - d_\mathrm{l})$.

Finally the value of V_p is selected such that the peak voltage of the resultant signal is normalised to a chosen peak voltage typically as defined by the measurement equipment.

To produce a transfer function from the measured output of the circuit, the input is deconvolved from the output signal by performing an element-wise division in the frequency domain.

An excitation signal was designed with frequencies between 1 Hz − 20.1 kHz, i.e. $f_0 = 1$, $d_\mathrm{l} = 1$ and $d_\mathrm{h} = 20100$. The measured frequency range was measured outside of the anticipated required range in case this information was important to the identification, though the range is later restricted to 20 Hz – 20 kHz. The limitations of the analogue input was 2 V which was used to find the value of V_p. The signal was repeated 60 times and averaged to one period to reduce stochastic noise.

The measurement equipment used is a National Instruments myDAQ. Previous experiments with Data Acquisition devices and identification have shown that any errors in the phase response can severely effect accuracy in the identification procedure [4]. Upon

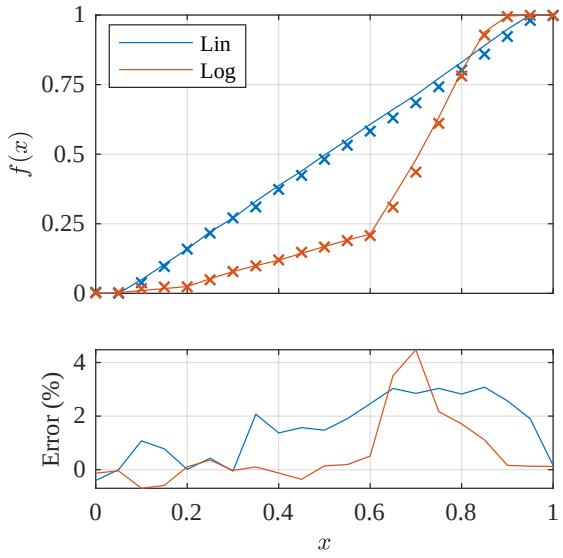


Figure 9: *Potentiometer laws as directly measured (solid line) and estimated from I/O measurements (× marks), and error between the two sets presented beneath.*

detection of phase error by testing the unloaded I/O response, it was decided that the phase response should be omitted from the identification data.

A notable capacitance was measured across the input to the myDAQ, noted as C_d in Table 4. This capacitance was included in the circuit model to incorporate any effect it may have on the measured transfer functions. The measured value of C_d was found by driving a series impedance of $2\,\mathrm{M\Omega}$ with a multi-sine signal, and measuring the cutoff frequency of resulting RC low pass filter at approximately $2.395\,\mathrm{kHz}$.

Measurements were taken from 21 positions along the rotational axis of both the linear and logarithmic potentiometers (15 degree rotations from 0-300 degrees). The amplitude response of each of the measured transfer functions was then used to estimate the value of y at each position. To minimise computational expense during optimisation, the transfer function data was downsampled from 20.1×10^3 data points to 512 points spaced logarithmically between 20 Hz and 20 kHz, each point rounded to the nearest integer such that it corresponds to a measured value, not requiring interpolation. Removing duplicate entries that occur at low frequencies results in a unique set of 469 amplitudes/frequencies.

4. RESULTS

Optimisation was applied as described in Section 3.1, using the Nelder-Mead algorithm and the objective function described in (12). Convergence was achieved successfully from each optimisation.

The identified potentiometer laws for both linear and logarithmic potentiometers are shown in Figure 9, with the solid line representing the directly measured law and the × marking the values estimated using the identification strategy. Error between the sets peak at approximately 4.5 % for the logarithmic characteristic and 3 % for the linear characteristic.

Illustrated for both potentiometer laws in Figure 10 are the measured and identified amplitude responses of the Big Muff Pi tone stack. This serves to provide one way of attributing error but also as a source of validation. Inspecting the x values of high error in Figure 9 it is clear that there is not an anomalous amount of error in the fit to the amplitude response at corresponding values. Consistent accuracy from identified amplitude responses but increasing error in the identified potentiometer characteristic points towards the error being introduced by device heating/human error during measurement.

As a form of validation, the fit to measurements demonstrated in Figure 10 is accurate to within 1 dB across all measurements, indicating that the tone stack filtering effect is captured over the the full wiper range.

5. CONCLUSION

This work has focused on the modelling and identification of potentiometer laws without device disassembly. Two functions were proposed to model potentiometer characteristics as found in specifications and from measurements, a $\tanh()$ based function and a piecewise linear-cubic function. The latter was used to model the laws of Alpha potentiometers as given by both specifications and measurements within 2% error.

Identification of the potentiometer characteristics from I/O measurements of the device was tested on a linear subcircuit of the Big Muff Pi: the tone stack. Identification exclusively using the amplitude response of the subcircuit was shown to be possible, with the results of the identification retrieving both linear and logarithmic potentiometer characteristics to within 4.5% error.

It is likely that some of this 4.5% error has been introduced by device heating and/or human error. Increasing the precision and repeatability of the measurements could be achieved by the design of an automated mechatronic system that would simultaneously rotate the potentiometer wiper and perform measurements. One such solution might involve e.g. a stepper motor to control the potentiometer and a pulsed measurement system to control the variation in resistance caused by changes in temperature.

For those seeking a fast method of determining the orientation and approximate law of a potentiometer, a complete set of 21 points is not required. Assuming $f(0)$ and $f(1)$ are known, from e.g. parameter estimation as in [4], only one additional measurement at $f(0.5)$ is sufficient to differentiate between linear and logarithmic/anti-logarithmic. A further measurement at e.g. $f(0.25)$ or $f(0.75)$ would then enable differentiation between orientations of the potentiometer and also logarithmic and anti-logarithmic laws.

The anticipated application of this identification strategy is for complete, nonlinear audio circuits. Elements that may cause issues in the identification include nonlinear behaviour, and effects for which potentiometers control behaviour that is difficult to measure, e.g. LFO rate in a phasor effect. The low-data point amplitude response facilitates fast identification, which has significant benefits with regard to rapid design and refinement of the identification process. Time-based and/or nonlinear effects would prevent this selection and therefore demand a search for a suitable, similarly efficient, objective function. For example, a gain control may use the Total Harmonic Distortion of the output waveform when driven by a sinusoid, or for a delay length control, the time between repeats when driven by a pulse-type signal. Each control type requires individual attention, but due to the monotonic nature of the studied potentiometer laws, and as each potentiometer can be identified independently, each position should yield a unique measured value, so long as the objective function is well-chosen.

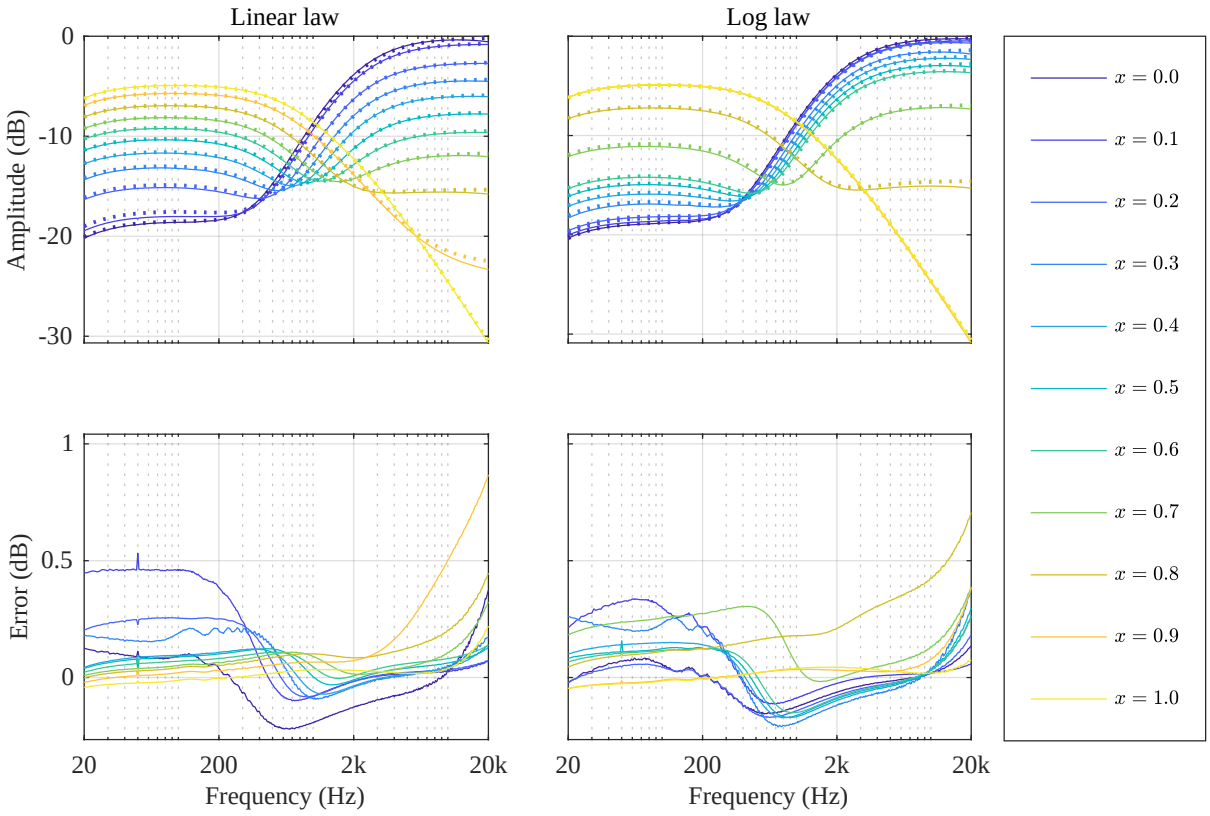


Figure 10: *Estimated (dotted) and measured (solid) amplitude responses of the Big Muff tone stack, with error in decibels shown below. Only 11 of the 21 measured amplitude responses are shown to improve clarity of the figure.*

6. REFERENCES

[1] Felix Eichas and Udo Zölzer, "Gray-Box Modeling of Guitar Amplifiers," *Journal of the Audio Engineering Society*, vol. 66, no. 12, pp. 1006–1015, Dec. 2018.

[2] Antonin Novak, Pierrick Lotton, and Laurent Simon, "Synchronized Swept-Sine: Theory, Application, and Implementation," *Journal of the Audio Engineering Society*, vol. 63, no. 10, pp. 786–798, Nov. 2015.

[3] Lamberto Tronchin and Vanna Lisa Coli, "Further Investigations in the Emulation of Nonlinear Systems with Volterra Series," *Journal of the Audio Engineering Society*, vol. 63, no. 9, pp. 671–683, Oct. 2015.

[4] Ben Holmes, *Guitar Effects-Pedal Emulation and Identification*, Ph.D. thesis, Queen's University Belfast, June 2019.

[5] Miller Puckette, *The Theory and Technique of Electronic Music*, World Scientific Publishing Co. Pte. Ltd., 2007.

[6] Oliver Larkin, Alex Harker, and Jari Kleimola, "iPlug 2: Desktop Plug-in Framework Meets Web Audio Modules," in *Proceedings of the 4th Web Audio Conference*, Sept. 2018.

[7] Ankit Rohatgi, "WebPlotDigitizer," https://automeris.io/WebPlotDigitizer/, Apr. 2019.

[8] Carl David Todd, *The Potentiometer Handbook: Users' Guide to Cost-Effective Applications*, McGraw-Hill, New York, 1975.

[9] U.S.A. Department of Defense, "Military specification MIL-PRF-94G: Resistors, variable, composition general specification for," Tech. Rep., Aug. 2011.

[10] Alpha Products, "Taper Curves for our Potentiometers," http://www.alphapotentiometers.net/html/taper_curves.html, 2014.

[11] Jeffrey C. Lagarias, James A. Reeds, Margaret H. Wright, and Paul E. Wright, "Convergence properties of the Nelder–Mead simplex method in low dimensions," *SIAM Journal on Optimization*, vol. 9, no. 1, pp. 112–147, 1998.

[12] Electrosmash, "Big Muff Pi Analysis," https://www.electrosmash.com/big-muff-pi-analysis.

[13] Chung-Wen Ho, Albert E. Ruehli, and Pierce A. Brennan, "The Modified Nodal Approach to Network Analysis," *IEEE Transactions on Circuits and Systems*, vol. 22, no. 6, pp. 504–509, 1975.

[14] Manfred Schroeder, "Synthesis of low-peak-factor signals and binary sequences with low autocorrelation (Corresp.)," *Information Theory, IEEE transactions on*, vol. 16, no. 1, pp. 85–89, 1970.

DRUM TRANSLATION FOR TIMBRAL AND RHYTHMIC TRANSFORMATION

Maciek Tomczak, Jake Drysdale and Jason Hockman

Digital Media Technology Lab (DMT Lab)
Birmingham City University
Birmingham, United Kingdom
{maciek.tomczak@bcu.ac.uk, jake.drysdale@bcu.ac.uk, jason.hockman@bcu.ac.uk}

ABSTRACT

Many recent approaches to creative transformations of musical audio have been motivated by the success of raw audio generation models such as WaveNet, in which audio samples are modeled by generative neural networks. This paper describes a generative audio synthesis model for multi-drum translation based on a WaveNet denosing autoencoder architecture. The timbre of an arbitrary source audio input is transformed to sound as if it were played by various percussive instruments while preserving its rhythmic structure. Two evaluations of the transformations are conducted based on the capacity of the model to preserve the rhythmic patterns of the input and the audio quality as it relates to timbre of the target drum domain. The first evaluation measures the rhythmic similarities between the source audio and the corresponding drum translations, and the second provides a numerical analysis of the quality of the synthesised audio. Additionally, a semi- and fully-automatic audio effect has been proposed, in which the user may assist the system by manually labelling source audio segments or use a state-of-the-art automatic drum transcription system prior to drum translation.

1. INTRODUCTION

The creative transformation addressed in this paper is generative audio synthesis of percussive instruments, which involves mapping (or *translation*) of musical audio to drum sounds achieved with artificial neural networks. Flexible digital audio effects that utilise machine learning techniques would benefit musicians and music producers by generating audio controllable by a target rhythm, melody or style that may be used directly in the music production process. Among such tools, autoregressive (AR) models for raw audio generation such as WaveNet [1] have inspired several systems that utilise the power of generative neural networks for musical audio synthesis. Audio synthesis in these models is achieved by learning an AR distribution that predicts the next audio sample from the previous samples in its receptive field using a series of dilated convolutions. The majority of these systems have been developed to address pitched instruments [1, 2], while no such systems have been focused on the generation of percussive instruments and rhythmic aspects of such transformations.

1.1. Background

In the field of digital audio effects, rhythmic and timbral transformations have initially been addressed through signal processing architectures [3, 4, 5, 6]. At present there are only a handful of systems that approach this type of transformation using artificial neural networks in areas such as audio style transfer [7] and AR generative audio synthesis [1, 8, 9].

WaveNet is a generative audio synthesis model that was developed for tasks related to speech synthesis [10, 11]. To date, a relatively small number of systems have experimented with it for synthesis of musical audio. Engel et al. [2] proposed a WaveNet autoencoder that learns codes that meaningfully represent the space of musical instruments with the ability to model long temporal dependencies. This work led to the NSynth system [2], a neural synthesiser capable of generating new sound embeddings learned from a large dataset of musical notes. Dieleman et al. [12] adapted the WaveNet architecture for the unconditional generation of piano music that exhibits stylistic consistency at longer timescales across tens of seconds. In [13], the authors combined audio and symbolic models and use a long short-term memory recurrent neural network (RNN) to learn melodic structures of different styles of music, which are used as conditioning input to a WaveNet-based instrument melody generator. Other AR models include RNN-based architectures such as: VRNN [14], SampleRNN [15] and WaveRNN [16]. Alternatively, the WaveNet architecture has been used in the context of musical timbre transfer. Huang et al. [17] adapted an image-based style transfer method [18] for translation of an image from one domain to another using a conditional WaveNet synthesiser within the TimbreTron model. Kim et al. [8] proposed a music synthesis system with timbre control that learns to generate spectrograms from symbolic music representations and instrument embeddings, and generates raw audio with a WaveNet vocoder.

Mor et al. [9, 19] introduced a novel system for timbre and style translations that combined the WaveNet autoencoder with unsupervised adversarial training. This architecture differed from other generative audio synthesis models in that it could convert the timbre of one instrument to that of another while preserving the melody and rhythm of the input. The WaveNet autoencoder architecture of Mor et al. [19] has been adopted, with key differences made in training strategies and the use of a simplified architecture, specialised for rhythmic and timbral transformations of percussion instruments.

1.2. Motivation

In this paper, a system that adapts the music translation approach for timbre transfer is proposed, with the aim of encoding the rhythmic structure of an arbitrary audio input as a combination of different percussion instruments from the common drum kit. In this

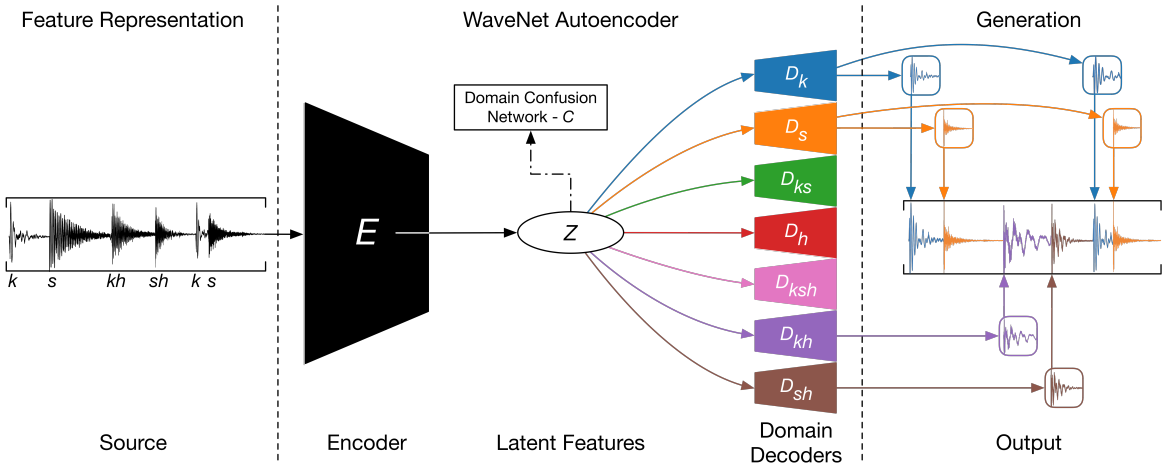


Figure 1: *Drum translation overview in three stages. Source audio is transformed to output through a single shared autoencoder of domain p specialised on domain decoders D_p, where p represents: kick (k), snare (s), kick and snare (ks), hi-hat (h), kick, snare and hi-hat (ksh), kick and hi-hat (kh) or snare and hi-hat (sh). Colours illustrate pathways between source and corresponding D_p trained to synthesise the target instrument (e.g., orange decoder D_s synthesises snare drums). Solid lines represent information flow during synthesis and dashed-dotted line represents information flow to a domain confusion network present only during training.*

transformation, the timbre of an input is transformed such that it sounds as if it were played on a different drum. To that end, a denoising WaveNet autoencoder architecture is modified and specialised for drum translation by utilising an unsupervised training strategy of a multi-domain latent space that is trained end-to-end on combinations of drum samples. In this architecture, a single encoder of [19] encodes a shared latent space for multiple decoders to use during training and audio generation. The size of the architecture is adjusted to learn short-term sounds of drum samples, while maintaining encodings for different drum instruments. The rhythmic accuracy of this model has been explored as well as a variety of creative percussive transformations in a simplified task of *drum-to-drum* translation akin to the task of redrumming or drum replacement [20]. The aims of the system are to facilitate the creation of new drum arrangements from arbitrary audio inputs provided by untrained musicians and to uncover the musical relationships between audio recordings that might otherwise have never been heard.

The remainder of this paper is structured as follows: Section 2 outlines our proposed method for drum translation. Section 3 presents experiments undertaken to assess the rhythmic accuracy and the quality of the translated audio. Section 4 provides experiment results with a discussion on audio degradation caused by the system for different drum domains. Conclusions and suggestions for future work are presented in Section 5.

2. METHOD

This approach to drum translation concerns the task of synthesising source audio to corresponding drum sounds. The system is inspired by architecture of [19], in which music signals can be translated across instruments and styles. This paper contributes to this kind of transformation by simplifying the translation network and proposing a new training strategy specialised towards percussion instruments.

Figure 1 provides an overview of the proposed drum translation system, which is comprised of three stages: (1) Feature representation; (2) WaveNet autoencoder; and (3) Generation. At the core of the system is a WaveNet autoencoder network with a shared encoder and a disentangled latent space, distributed across each drum domain decoder D_p, where p represents a percussion domain for P total number of domains. In total, there are seven percussion domains ($P = 7$) defined as kick (`k`), snare (`s`), hi-hat (`h`) instruments as well as their combinations such as, kick and hi-hat (`kh`). During training, multiple source-target p pathways (one per drum domain illustrated by different colours in Figure 1) are encoded by a domain-independent encoder E. The input to the neural network is an audio segment X_p of length T samples ($T = 6000$) representing a waveform of one of the seven drum domains. Each segment is distorted by random pitch modulation to prevent the network from memorising the input signal and provide a semantic encoding.

To improve the generalisability of a single encoder during training and to increase the size of the training data, a pitch augmentation approach of [19] is implemented. In popular music production, pitch shifting of individual drum samples is a common processing technique that is used either on all drums or on a subset of drum samples that are layered underneath other sounds to create richer timbres. Instead of augmenting only parts of the input data as in [19], pitch is modulated across the whole length of each audio segment X_p by a random value between ± 3 semi-tones with LibROSA [21]. The final representation of each augmented percussion segment used for training is $X_p = \{x_{p,1}, ..., x_{p,T}\}$.

The input audio goes through the Wavenet encoder E, a fully convolutional network, and outputs latent space Z that is downsampled with 8-bit μ-law encoding [1]. The latent space is then used to condition a domain confusion network [22] responsible for providing an adversarial signal to the encoder during training. The latent signal is then temporally upsampled to the original audio rate and is used to condition a WaveNet decoder D_p. Each decoder uses a softmax activation to output the probability of the next time step. Once training is finished the embeddings of all

drum domains in the shared latent space can be used to transform source audio from any arbitrary audio domain.

2.1. WaveNet Autoencoder

A similar dilated convolutional WaveNet encoder architecture as [2] is adopted and a WaveNet decoder from [19] to model percussion sounds in the time domain. Dilated convolutions are convolutions with holes that greatly increase the receptive field while significantly reducing the model parameters and computational cost. In addition to dilated convolutions, WaveNet incorporates a residual learning framework [23] that reduces training time and avoids the vanishing gradient problem linked with the training of very deep neural networks.

The shared encoder E has 18 layers with two blocks of nine residual-layers [23] and a maximum dilation of 512 samples. As in [19], a residual-layer structure with a ReLU non-linearity is used, a non-causal dilated convolution with an increasing kernel size, a second ReLU, and a 1x1 convolution (i.e., a time-distributed fully connected layer) followed by a residual summation of the activations before the first ReLU. Unless specified otherwise, 64 channels are used in all hidden layers of the autoencoder architecture. After two blocks, the encoding goes through a 1x1 layer and an average pooling with a kernel size of υ samples ($\upsilon = 400$).

The WaveNet domain decoder D_p is conditioned with a temporally upsampled version of the latent encoding Z^{up} obtained with nearest neighbour interpolation. The conditioning signal adds parameters to the probability distribution so that it depends on variables that describe the audio to be generated instead of only using the previously generated samples. Without conditioners, WaveNet has been shown to mix sequences of speech by repeated phoneme shifting between voices of all speakers used in training of the model. As in [19], the conditioning goes through a different 1x1 layer for each decoder D_p to ensure that the latent space is domain independent. This reduces source-target pathway memorisation, which is also aided by pitch augmentation. To ensure that only previous samples are used in the generation of the new ones, decoders D_p use dilated causal convolutions together with additional non-linear operations to enable them to learn input audio representations that cannot be captured with just linear operations. Each D_p has two blocks of nine residual-layers, where each layer contains a causal dilated convolution with an increasing kernel size, a gated hyperbolic tangent activation [1] (the main source of non-linearity), a 1x1 convolution followed by the residual summation of the layer input, and a 1x1 convolution layer for skip connections. Encoding Z^{up} is used to condition each residual-layer during training. The skip connections are summed with a ReLU non-linearity activation and passed through a 1x1 convolution layer before a softmax activation layer.

2.2. Domain Confusion Network

In order to introduce an adversarial signal to the autoencoder and ensure that the encoding is not domain-specific, a domain confusion network C is implemented following [19]. The network predicts the percussion domain label of the input data based on the latent vectors Z. It uses a single gradient reversal layer defined in [22] and three 1D-convolution layers. The gradient reversal layer (GRL) reverses the gradient by multiplying it with a negative scalar λ ($\lambda = 0.01$). The GRL ensures that the feature distributions over the P drum domains are made similar (i.e., as difficult as possible to recognise for the domain classifier C), thus resulting in the domain-independent features. The three 1D-convolutional layers all include ELU non-linearities [24] with 128 channels in all hidden layers. After three layers the output is passed through a tanh and a 1x1 convolution layer to project the vectors to P total number of domains.

2.3. μ-law Quantisation

WaveNet predicts a non-normalised probability distribution from the residual-layers and transforms it into a proper probability distribution by using a softmax function. The authors of the original WaveNet [1] show that softmax distribution tends to be more flexible than other mixture models and can more easily model arbitrary distributions as it makes no assumptions about their shape.

All audio files processed by the model use a sampling rate of 22.05 kHz and are stored as 16-bit integers. To model all possible values per time step, a softmax layer would need to output 2^{16} probabilities. To moderate this high bit-depth resolution of the input audio, a μ-law algorithm [25] is implemented and quantises the data to 2^8 quantisation levels:

$$f(x_p) = sign(x_p) * \frac{ln(1+\mu|x_p|)}{ln(1+\mu)}, \tag{1}$$

where μ = 255 and $-1 < x_p < 1$. This quantises the high resolution input to 256 possible values causing a loss in audio quality [1], however it makes the model feasible to train.

2.4. Model Details

Two losses are minimised during training with regards to an input sample x_p at time step t from augmented segment X_p: (1) domain confusion loss $\mathcal{L}_{dc}$,

$$\mathcal{L}_{dc} = \sum_{p} \sum_{x_p} \ell_{ce}(C(E(x_p)), p), \tag{2}$$

which applies cross entropy loss ℓ_{ce} to each element of the output Z and the corresponding percussion label p, and (2) autoencoder loss $\mathcal{L}_{ac}$,

$$\mathcal{L}_{ac} = \sum_{p} \sum_{x_p} \ell_{ce}(D_p(E(x_p)), x_p). \tag{3}$$

The decoder D_p is an AR model conditioned on the output of the shared encoder E. Final loss $\mathcal{L}$ is defined as:

$$\mathcal{L} = \mathcal{L}_{ac} - \lambda \mathcal{L}_{dc}, \tag{4}$$

where λ is a scaling factor for $\mathcal{L}_{dc}$ described in Section 2.2.

An Adam optimiser with the initial learning rate of 0.001, and a decay factor of 0.98 is used. The model is trained for 10 epochs and 50,000 iterations in total, where each iteration takes a random mini-batch of 8 randomly pitch shifted X_p segments. All weights in the network are initialised using Xavier initialisation [26].

The system consists of a naïve WaveNet architecture [27], where $O(2^L)$ is the overall computation time for a single output with L total number of layers of the WaveNet autoencoder outlined in Section 2.1. The system is implemented using the Tensorflow Python library[1] and was trained on an NVIDIA Tesla M40 computing processor.

[1] `https://www.tensorflow.org/`

2.5. Generation

During the transformation of an unaugmented audio sample y from any source domain, the autoencoder of domain p with its corresponding D_p is used to output the new sample $\hat{y}$ through:

$$\hat{y} = D_p(E(y)). \tag{5}$$

3. EXPERIMENTS

Two individual experiments are conducted to evaluate the rhythmic modification characteristics of the system as well as the translation quality through numerical analysis. In the first experiment (Section 3.2), the rhythmic similarity between source and drum translated audio pairs are evaluated by measuring the cosine similarity between rhythmic envelopes. For the second experiment (Section 3.3) a numerical analysis of audio degradation by [8] is presented measuring the Pearson correlation between the translated and source audio.

3.1. Training Data

For all experiments, the system is trained using raw audio waveforms as input features. Kick, snare and hi-hat samples used in the creation of different domains p for training are selected from a variety of different sample libraries included in the Ableton Live 10 Suite software. Drum samples are reduced to mono 16-bit WAV files and downsampled from 44.1 kHz to 22.05 kHz. To ensure each domain is accurately represented by the model, silence is removed from the start of each audio recording. The additional domains, which represent when two or more percussion instruments are played simultaneously (e.g., kick drum and hi-hat together), are artificially synthesised by overlaying randomly selected percussion samples.

In total, there are 1000 drum recordings for each domain resulting in a total dataset size of 7000. The mean duration of the drum recordings is 4862 samples (i.e., 0.22s). The resulting segments are normalised and zero-padded to a constant length T. Audio samples longer than T are trimmed, with a linear fade applied to the last 1000 samples. To mitigate the low resolution at ranges near ± 1 that results from the μ-law encoding stage, the amplitudes of all audio segments are randomly scaled between 0.5 and 0.6.

3.2. Experiment 1: Rhythmic Similarity

The purpose of the first experiment is to evaluate the capacity of the proposed system for preservation of rhythmic patterns during transformations. A variety of different drum loops are used as the source for this experiment. The events in each drum loop are manually labelled, then translated into an output drum loop where domains correspond to the source. The cosine similarity is measured between the rhythmic envelopes of source and output transformation pairs.

In order to extract the rhythmic envelope R from each file, the short-time Fourier transform of each audio file is computed using an n-length Hanning window ($n = 2048$) with a hop size of $\frac{n}{4}$. The standard spectral difference envelopes are then calculated as the sum of the first-order difference between each spectrogram (e.g., [28]). The resulting envelopes are then normalised between 0 and 1. Following the approach described in [7], the cosine similarity Φ between rhythmic envelopes of inputs and their transformations is calculated as follows:

$$\Phi_{\alpha,\beta} = \frac{R_\alpha \cdot R_\beta}{\|R_\alpha\|\|R_\beta\|}, \tag{6}$$

where α and β represent source input and drum translated output respectively.

This experiment is conducted using 20 drum loops selected from the Apple Logic Pro sample library, resulting in 20 transformations to be evaluated. The drum loops are chosen to reflect a variety of different drum patterns and styles, with multiple domains reflected in each loop. The drum loops have a mean duration of 3.5s and tempo ranges from 100 beats per minute (BPM) to 170 BPM. All loops are in the mono WAV format and are resampled to 22.05 kHz with 16-bit resolution.

3.3. Experiment 2: Translation Quality

In the second experiment, the timbral differences linked to the μ-law quantisation and the limited WaveNet model capacity are analysed. As an objective measure of audio quality, following [8], the Pearson correlations between the source and the drum translated audio are plotted. The correlations are visualised over a logarithmically scaled range of frequencies between 32–10548 Hz and calculated using 101-bin log-magnitude constant-Q transforms (CQT) with 12 bins per octave starting from C1 (≈32.70Hz) and hop size of 512 using LibROSA [21]. For the purpose of this experiment, seven audio tracks are created using drum samples selected from the Logic Pro sample library. Each track contains ten drum samples with different timbres from the corresponding domains p (e.g., ten kick drum samples). All ten samples are translated into their corresponding domains (e.g., ten kick samples transformed into ten kick samples), resulting in a total of 70 separate drum translations.

4. RESULTS AND DISCUSSION

4.1. Rhythmic Similarity

The mean cosine similarity score across the 20 transformations in the rhythmic similarity experiment is 0.75. This indicates that in most cases, the proposed system is capable of preserving rhythmic structure when translating from source to target domain. The highest rhythmic similarity score is 0.96, which suggests that for this transformation the majority of the target domains were successfully translated resulting in a rhythmic envelope that is almost identical to the input. Transformations receiving low similarity scores failed to translate parts of the input resulting in dissimilar rhythmic envelopes. The lowest performing transformation has a rhythmic similarity score of 0.48 due to a frequent failure in properly translating domain `k`. The possible reasons for different artifacts and behaviours of the synthesised audio are discussed using two translation outputs presented in Figure 2.

To better understand the issues present in the translated outputs, two examples of source and output waveforms have been plotted in Figure 2. The cosine similarity of the translation A is 0.79 indicating some differences in their rhythmic similarities. In the first beat of the output example A, it can be seen that the waveform of the source kick drum was unnaturally smeared, whereas the source kick drum A from beat 3 was not translated at all. Both, the smearing and the missing event are examples of the system failing to correctly translate to domain `k` that is a cause of multiple failed transformations in Experiment 1. In example B, all drum

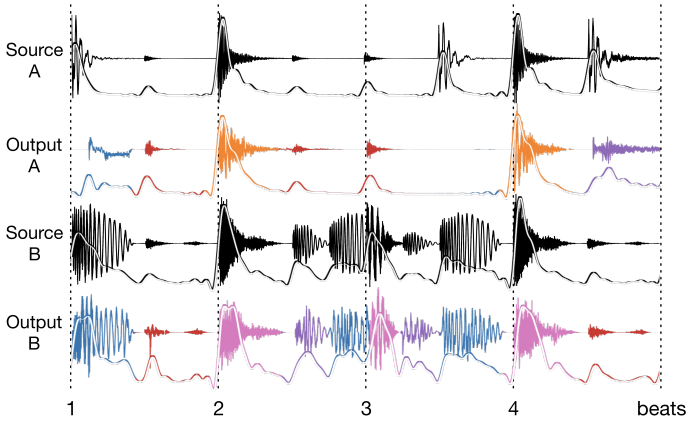


Figure 2: *Example translations generated from two sources (A and B), with spectral difference functions as solid lines over each waveform. Output colours correspond to target drum domains (e.g., blue represents kick drum translations).*

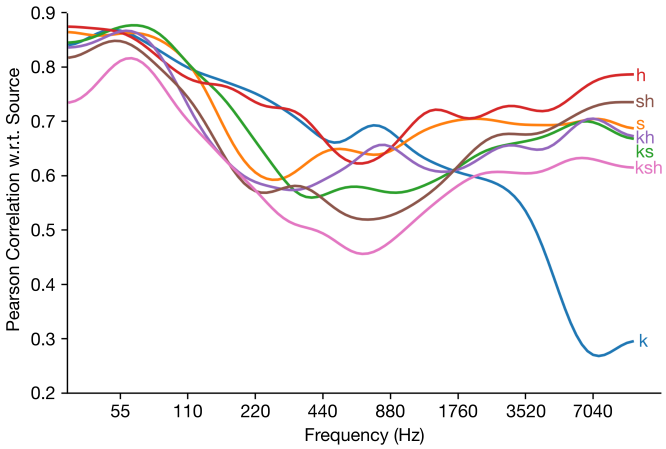


Figure 3: *Smoothed mean Pearson correlations between the translated and source audio for all drum domains.*

domains were effectively translated resulting in a high rhythmic similarity score of 0.96. Is is likely that the timbral characteristics of source B contributed to a more suitable translation output that shared, to a certain extent, some of the latent representations of different drum domains. For instance, the output kick drum translation from source B in beat 1 exhibits a noisy attack but then effectively transforms the expected low frequencies, while still notably changing the characteristics of the source thus creating a new kick sound.

4.2. Translation Quality

Figure 3 presents the results using evaluation methodology described in Section 3.3. For each domain, the mean Pearson correlation is taken for ten different source and output log-magnitude CQT spectrograms. The correlation values across all frequencies are smoothed using a median filter. High Pearson correlations indicate that translation quality for a particular domain is well preserved, whereas lower correlations indicate larger quality degradation from the source.

For all domains, the results indicate that frequency information below 100 Hz is well preserved. Across all domains there is a significant drop in correlation for low-mid frequencies between 220–1760 Hz. In comparison to other instruments, domain `k` (blue curve) maintains higher correlation for these low-mid frequencies however, there is a significant roll-off for high frequencies above 1760 Hz due to noise introduced by the model. Domain `ksh` (pink curve) represents when a kick, snare and hi-hat are all played simultaneously and has the lowest Pearson correlation across all frequency bands demonstrating that this domain was most difficult to reconstruct accurately due to complexity introduced by the three instruments.

Drum translations used in Experiment 1 as well as other examples can be found on the supporting website for this project.[2] As can be heard from many of the drum translations, the proposed system is capable of generating samples indicative of the intended target domains; however, a considerable amount of noise is produced in the transformations. It is assumed that the audio quality of the system is currently restricted by the limited amount of data on which it has been trained; by using more varied training data that has not been synthesised artificially (e.g., audio samples corresponding to real life audio loops or drum recordings), it is expected that the system would be capable of producing more accurate transformations. In addition to this, the limited size of the model restricts the number of possible transformations.

[2] `https://maciek-tomczak.github.io/maciek.github.io/Drum-Translation-for-Timbral-and-Rhythmic-Transformation`

4.3. Automatic and Semi-automatic Drum Translation

An automatic and semi-automatic extension to drum translation has been proposed, in which a user can choose to automatically label different drums in the audio input with ADTLib [29] or manually annotate the input prior to translating it into various target drum domains. In this transformation, the length of segments attributed to the labelled domain corresponds to detected inter-onset-intervals. A novice or expert user can use such transformation in a music composition scenario. Once all onsets are processed the system translates annotated source audio segments into specified drum domains.

5. CONCLUSIONS

A drum translation technique that explores the rhythmic and timbral capabilities of generative audio synthesis with WaveNet autoencoders has been presented. In this transformation, an input file is transformed so that it sounds as if it were performed by different drum instruments. Two experiments were conducted to assess the rhythmic accuracy and overall audio quality of the transformations with respect to different drum instruments. The experimental results demonstrate that the system produces rhythmically accurate transformations, while there exist significant aspects of the proposed transformation that contribute to the creation of audio artifacts and added noisiness that could be mitigated with a network architecture of larger capacity. The memory requirements of the current model limit the number of available residual channels and the number of layers which can be overcome by architectures such as Parallel WaveNet [30]. Another possible avenue of future

work would explore a broader range of signals outside the tested datasets as well as other applications of diverse adversarial losses to audio to improve translation audio quality.

6. REFERENCES

[1] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu, "WaveNet: A generative model for raw audio," *Computing Research Repository, abs/1609.03499*, 2016.

[2] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck, and Karen Simonyan, "Neural audio synthesis of musical notes with WaveNet autoencoders," in *Proceedings of the International Conference on Machine Learning*, pp. 1068–1077, 2017.

[3] Emmanuel Ravelli, Juan P. Bello, and Mark Sandler, "Automatic rhythm modification of drum loops," *IEEE Signal Processing Letters*, vol. 14, no. 4, pp. 228–231, 2007.

[4] Jason Hockman, Juan P. Bello, Matthew E. P. Davies, and Mark D. Plumbley, "Automated rhythmic transformation of musical audio," in *Proceedings of the International Conference on Digital Audio Effects*, pp. 177–180, 2008.

[5] Matthew E. P. Davies, Philippe Hamel, Kazutomo Yoshii, and Masataka Goto, "AutoMashUpper: Automatic creation of multi-song music mashups," *in IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 12, pp. 1726–1737, 2014.

[6] Shun Sawada, Satoru Fukayama, Masataka Goto, and Keiji Hirata, "TransDrums: A drum pattern transfer system preserving global pattern structure," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 391–395, 2019.

[7] Maciek Tomczak, Carl Southall, and Jason Hockman, "Audio style transfer with rhythmic constraints," *in Proceedings of the International Conference on Digital Audio Effects*, pp. 45–50, 2018.

[8] Jong W. Kim, Rachel Bittner, Aparna Kumar, and Juan P. Bello, "Neural music synthesis for flexible timbre control," *Computing Research Repository, abs/1811.00223*, 2018.

[9] Noam Mor, Lior Wolf, Adam Polyak, and Yaniv Taigman, "A universal music translation network," *Computing Research Repository, abs/1805.07848*, 2018.

[10] Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, R. J. Skerrv-Ryan, et al., "Natural TTS synthesis by conditioning WaveNet on mel spectrogram predictions," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pp. 4779–4783, 2018.

[11] Andrew Gibiansky, Sercan Arik, Gregory Diamos, John Miller, Kainan Peng, Wei Ping, Jonathan Raiman, and Yanqi Zhou, "Deep voice 2: Multi-speaker neural text-to-speech," in *Proceedings of the Neural Information Processing Systems*, pp. 2962–2970, 2017.

[12] Sander Dieleman, Aäron van den Oord, and Karen Simonyan, "The challenge of realistic music generation: Modelling raw audio at scale," in *Proceedings of the Neural Information Processing Systems*, pp. 8000–8010, 2018.

[13] Rachel Manzelli, Vijay Thakkar, Ali Siahkamari, and Brian Kulis, "Conditioning deep generative raw audio models for structured automatic music," *Computing Research Repository, abs/1806.09905*, 2018.

[14] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C. Courville, and Yoshua Bengio, "A recurrent latent variable model for sequential data," in *Proceedings of the Neural Information Processing Systems*, pp. 2980–2988, 2015.

[15] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio, "SampleRNN: An unconditional end-to-end neural audio generation model," *Computing Research Repository, abs/1612.07837*, 2016.

[16] Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aäron van den Oord, Sander Dieleman, and Koray Kavukcuoglu, "Efficient neural audio synthesis," *Computing Research Repository, abs/1802.08435*, 2018.

[17] Sicong Huang, Qiyang Li, Cem Anil, Xuchan Bao, Sageev Oore, and Roger B. Grosse, "TimbreTron: A WaveNet (CycleGAN (CQT (Audio))) pipeline for musical timbre transfer," *Computing Research Repository, abs/1811.09620*, 2018.

[18] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the International Conference on Computer Vision*, pp. 2223–2232, 2017.

[19] Noam Mor, Lior Wolf, Adam Polyak, and Yaniv Taigman, "A universal music translation network," in *Proceedings of the International Conference on Learning Representations*, 2019.

[20] Patricio López-Serrano, Matthew E. P. Davies, and Jason Hockman, "Break-informed audio decomposition for interactive redrumming," *Late-Breaking Demo Session Abstract in the International Society for Music Information Retrieval*, 2018.

[21] Brian McFee, Colin Raffel, Dawen Liang, Daniel P. W. Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the Python in Science Conference*, pp. 18–25, 2015.

[22] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky, "Domain-adversarial training of neural networks," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[24] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," *Computing Research Repository, abs/1511.07289*, 2015.

[25] ITU-T. Recommendation G. 711, "Pulse code modulation (PCM) of voice frequencies," 1988.

[26] Xavier Glorot and Yoshua Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pp. 249–256, 2010.

[27] Tom Le Paine, Pooya Khorrami, Shiyu Chang, Yang Zhang, Prajit Ramachandran, Mark A. Hasegawa-Johnson, and Thomas S. Huang, "Fast WaveNet generation algorithm," *Computing Research Repository, abs/1611.09482*, 2016.

[28] Simon Dixon, "Onset detection revisited," in *Proceedings of the International Conference on Digital Audio Effects*, pp. 133–137, 2006.

[29] Carl Southall, Ryan Stables, and Jason Hockman, "Automatic drum transcription using bi-directional recurrent neural networks.," in *Proceedings of the International Society of Music Information Retrieval Conference*, pp. 591–597, 2016.

[30] Aäron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis Cobo, Florian Stimberg, et al., "Parallel WaveNet: Fast high-fidelity speech synthesis," *Computing Research Repository, abs/1711.10433*, 2017.

ON THE IMPACT OF GROUND SOUND

Ante Qu and Doug L. James

Computer Science,
Stanford University
Stanford, CA 94305
[antequ|djames]@cs.stanford.edu

ABSTRACT

Rigid-body impact sound synthesis methods often omit the ground sound. In this paper we analyze an idealized ground-sound model based on an elastodynamic halfspace, and use it to identify scenarios wherein ground sound is perceptually relevant versus when it is masked by the impacting object's modal sound or transient acceleration noise. Our analytical model gives a smooth, closed-form expression for ground surface acceleration, which we can then use in the Rayleigh integral or in an "acoustic shader" for a finite-difference time-domain wave simulation. We find that when modal sound is inaudible, ground sound is audible in scenarios where a dense object impacts a soft ground and scenarios where the impact point has a low elevation angle to the listening point.

1. INTRODUCTION

Many sound synthesis examples in computer animation and virtual environments contain moving objects that impact the ground or other large flat surfaces. The ground affects the sound in two ways: 1) as a passive scatterer: sound waves in the room are reflected off the ground, and 2) as an emitter: the surface of the ground vibrates due to impact events, and thus emits sound. Typical approaches incorporate the passive scattering and reflection depending on context and methodology; however, very few physics-based approaches consider the acoustic emissions of the ground itself. In this paper we model the ground as an idealized elastodynamic halfspace, and analyze its sound emission during an object-ground impact. Its relative importance is assessed in various object-ground impact scenarios, and is found to vary greatly.

Ground emission and scattering have been explored in many works over the decades. One line of works [1, 2] fit data-driven models to synthesize footstep sounds. Works on fracture and micro-collisions [3, 4] treat the ground and table as a large modal vibration source; of these, one paper [3] models the modes from a 9 m $\times$ 9 m $\times$ 0.9 m concrete slab; these dimensions directly affect modal resonant frequencies. The modal method also requires heavy precomputation resources and storage because large objects have many vibration modes within audible frequencies. Furthermore, the above methods [3, 4] compute propagation with only one object at a time and omit repeated object-ground reflections.

After an object-ground collision, we may hear three types of sounds: (1) the object emits ringing sound from on its resonant modes, (2) the object emits a transient acceleration noise upon impact, and (3) the ground emits a transient sound upon impact. While many previous papers [3, 4, 5] model the first two in their sound synthesis, they omit the third type of sound. Most recently, [5] models the collisions of many floor materials; however, it incorporates the floor properties only in the excitation force profile and models sound just from the object's surface.

One argument for omitting the ground sound is that object sounds are often louder, especially for larger objects, and can mask quieter ground contributions. Nevertheless, depending on the flooring materials, contact parameters, and listening angle, the ground can sometimes be a more efficient sound source than a small object. The interference from the object's reflection can also change its waveform and make it distinct from the ground sound. Our ground vibration model allows us to quantify the intensity of the ground sound, albeit for a simplified elastic halfspace ground model.

Another work [6] develops a discretized modal model to collide vibrating strings with solid obstacles. We aim not to model ringing sound but rather to introduce a closed-form formula for the transient surface vibrations due to a single impulse.

Finite-difference, time-domain (FDTD), wave-based sound synthesis methods [7, 8, 9] naturally handle scattering with static or moving objects. Recently, [9] enabled scattering with moving objects by rasterizing their boundaries during each timestep. This method abstracts away object sources using an "acoustic shader" interface; the simulation queries the object shader for the surface vibrations and uses them to drive sound waves that propagate to the listener. However, no method is proposed to evaluate ground vibrations in an acoustic shader. We implement a ground shader in this work. Our work focuses on the topic of sound emission rather than reflections; it is orthogonal to room acoustics models that simulate room impulse responses and modes.

This surface vibration problem has been studied in seismology literature as Lamb's problem [10], and its ideal solution is well-known with a closed form. However, the ideal solution to an instantaneous load contains singularities at wavefronts that are difficult to evaluate numerically. To smooth the singularities, we derive a closed-form temporal regularization of the solution to Lamb's problem that removes the singularities at the three wavefronts, similar to how [11] regularizes the singularity in an infinite elastic medium for animation effects. This closed-form expression makes it easy to model ground sound without simulation.

We consider the following problem: Given a simple solid object, such as a ball, colliding with the ground (modeled as an elastic half-space) how do we estimate the sound emitted by the surface vibrations of the ground? Our contributions are

1. an estimate of the material properties and object sizes where the ground sound is not masked by the object sound,
2. an interactive method to synthesize ground sound (no precomputation is required), and
3. an "acoustic shader" for finite-difference time-domain simulations that directly evaluates the regularized solution.

2. GROUND SOUND MODEL: BACKGROUND

We model the transient ground sound by first modeling the ground surface vibration, and then using this motion to drive sound propagation into the air. For the former, we derive a closed-form model of the ground vibration to minimize computation while preserving accuracy. Our propagation model is one-way coupled because air pressure oscillations are not powerful enough to affect the ground.

In particular, we use Lamb's problem [10] and its solutions to model the floor surface vibrations from an impact, and we describe them in Sections 2.2 and 2.3. We regularize the model in Section 3 to eliminate undesired singularities, and then we model the sound propagation in Section 4.

2.1. Lamb's problem

We present Lamb's problem here, which involves applying an instantaneous normal point load to an elastic halfspace. We present it with a load rather than an impulse in order to simplify the mathematical representation of the solution. In later sections we will derive and use a closed-form representation of the surface acceleration in response to a specific impulse profile.

Consider a linear isotropic elastic half-space with Poisson's ratio ν and stiffness (shear modulus) μ, as shown in Figure 1. We consider the elastic half-space to be on the bottom (z negative), and free space to be above it, with the boundary being the horizontal $z = 0$ plane. Starting at time $t = 0$, a normal point force of magnitude 1 is applied and held ("push") at the origin $(0, 0, 0)$. The input force profile on the $z = 0$ plane is therefore

$$f(x, y, t) = \delta(x, y)\, \theta(t)\, \hat{\mathbf{z}}, \tag{1}$$

where δ, θ are the Dirac delta and Heaviside theta functions.

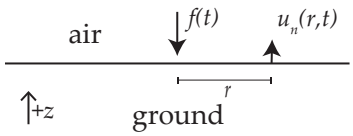


Figure 1: **Notation for Lamb's problem:** $f(t)$ is the ground excitation force, and $u_n(r, t)$ is the vertical displacement response. Note that while our diagram shows $f(t)$ in its usual downward direction ($-z$), we define $f(t)$ in (1) to point in the $+z$ direction.

The linear partial differential equations and boundary conditions can be found, for example, in equations 4 and 1 (respectively) of [12]; we present their closed-form solution in the next section.

2.2. Solution to Lamb's problem

Pekeris [12] first solved Lamb's problem in 1955 for $\nu = 1/4$. Others [13] later solved it for generic ν. We present the solution for generic ν from [14]. Some relevant notation is the following:

$$c_p = \text{speed of compression (P)-waves in the medium,}$$
$$c_s = \text{speed of shear (S)-waves in the medium,}$$
$$a = \frac{c_s}{c_p} = \sqrt{\frac{1-2\nu}{2-2\nu}}, \quad r = \sqrt{x^2 + y^2}.$$

Define $\kappa_1^2, \kappa_2^2, \kappa_3^2$ as the complex roots to the Rayleigh equation:

$$16(1-a^2)\kappa^6 - 8(3-2a^2)\kappa^4 + 8\kappa^2 - 1 = 0. \tag{2}$$

This equation admits three real solutions when $\nu < 0.2631$; otherwise, it has one real root and two complex conjugates. Let κ_1^2 be the largest real root, and define $\gamma = \kappa_1$. Treat these roots as mathematical tools to help express the result with no direct physical meaning (except that γ is the ratio of the S- and R- wave speeds).

Define the following set of coefficients:

$$A_j = \frac{(\kappa_j^2 - \frac{1}{2})^2 \sqrt{a^2 - \kappa_j^2}}{(\kappa_j^2 - \kappa_i^2)(\kappa_j^2 - \kappa_k^2)}, i \neq j \neq k$$

While the response contains both horizontal and vertical displacement, only the vertical motion produces sound. The final vertical displacement response $u_n(r, t)$ is the following:

$$u_n(r,t) = \frac{1-\nu}{2\pi\mu r} \begin{cases} 0 & \tau \leq a, \\ \frac{1}{2}\left(1 - \sum_{j=1}^{3} \frac{A_j}{\sqrt{\tau^2 - \kappa_j^2}}\right), & a < \tau < 1, \\ 1 - \frac{A_1}{\sqrt{\tau^2 - \gamma^2}}, & 1 \leq \tau < \gamma, \\ 1 & \tau \geq \gamma, \end{cases} \tag{3}$$

$$\tau = \frac{c_s t}{r}. \tag{4}$$

This solution applies for all ν, from 0 to 0.5 (see [14]). The piecewise boundaries correspond to the three wavefronts: the pressure P-wave arrives first, when $\tau = a$, travelling at speed c_p. The shear S-wave arrives when $\tau = 1$, travelling at speed c_s. Finally, the Rayleigh R-wave arrives when $\tau = \gamma$, travelling the slowest at speed $c_r = c_s/\gamma$. See the blue line in Figure 2 for an illustration.

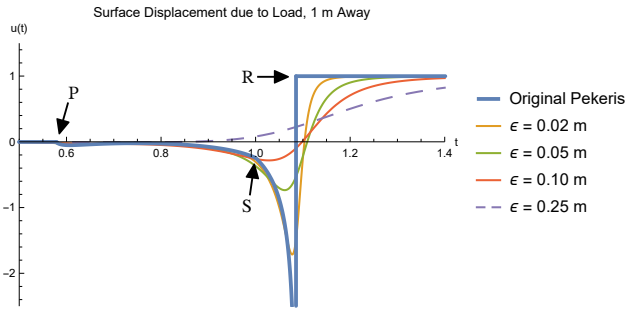


Figure 2: **Elastic wavefronts in time:** (Blue:) Scaled displacement response in the Pekeris solution, at 1 m away. The three wavefronts (P-, S-, R-) are labeled. (Other colors:) Our temporal regularization, described in Section 3. The horizontal axis is time in seconds; the vertical axis is scaled normal displacement.

Note: It is often convenient to flip the signs of a^2, γ^2, and τ^2 in the square roots of both the numerator and denominator in the terms containing A_1, so that the inside of the square root is real.

2.3. Singularities

In order to radiate sound waves we need to evaluate the acceleration in the impulse response of Lamb's problem. Unfortunately, the push-like load's displacement response, $u_n(r, t)$, already contains four singularity locations, which means that at each singularity it will be difficult to numerically approximate surface motion.

- One singularity occurs at all positive t at the origin, where $r = 0$. This singularity occurs due to the spatial δ load location, and it has asymptotic behavior $1/r$.

- One singularity occurs at each of the wavefronts—one each at the P- ($\tau = a$), the S- ($\tau = 1$), and the R- ($\tau = \gamma$) wave fronts. The first two wavefronts have continuous but not differentiable singularities. The third wavefront is discontinuous with asymptotic behavior $(\gamma - \tau)^{-1/2}$.

Our goal is to design a regularizing function in time or in space, as a smooth approximation of a delta impulse, to act as the initial force. We then convolve our function with the u_n solution to get a closed-form response that removes the singularities.

We consider the physical parameters of our problem in choosing temporal versus spatial regularization. We would like the regularization parameter to directly match the contact timescale and area. Typical contact radii are much smaller than the contact timescale multiplied by any of the three wave speeds; see Table 1 for one example. Therefore the spatial contact area smooths the resulting wave by very little compared to the temporal smoothing. Temporal regularization thus gives us a more accurate response than spatial.

3. TEMPORAL REGULARIZATION OF THE GROUND VIBRATION MODEL

Consider a function $f_\epsilon(t)$ that approximates $\delta(t)$ on a smoothing timescale ϵ. Since the elastic wave equation is linear and u_n is the response to a Heaviside θ load, we can get the vertical displacement response to the force, $f_\epsilon * \theta$ (which is an approximate θ), by computing the convolution $u_\epsilon = f_\epsilon * u_n$, or

$$u_\epsilon(r,t) = \int_{-\infty}^{\infty} f_\epsilon(t-t')u_n(r,t')dt'. \tag{5}$$

The above gives us the displacement response to a "push load" (c.f. [11]). We want an impulse response corresponding to a $f_\epsilon(t)$ force profile. Since δ is the derivative of θ and f_ϵ can be written $f_\epsilon * \delta$, we can subsequently compute the displacement response w_ϵ to an f_ϵ impulse force by taking a time derivative of u_ϵ, and likewise the acceleration a_ϵ by taking more derivatives:

$$w_\epsilon(r,t) = \frac{\partial u_\epsilon}{\partial t}, \tag{6}$$

$$a_\epsilon(r,t) = \frac{\partial^3 u_\epsilon}{\partial t^3}. \tag{7}$$

We use the regularization function f_ϵ defined by

$$g_\epsilon(t) = \frac{c_s\epsilon}{\pi(c_s^2t^2+\epsilon^2)}; \tag{8}$$

$$f_\epsilon(t) = 2g_\epsilon(t) - g_{2\epsilon}(t). \tag{9}$$

We chose this function for several reasons. Firstly, it approximates a $\delta(t)$ function as $\epsilon \to 0$: for all ϵ, the total impulse applied is 1, and as ϵ gets smaller, a larger proportion of the impulse is applied over a smaller amount of time ($\int_{|t|<\sqrt{\epsilon}} f_\epsilon(t) \to 1$ as $\epsilon \to 0$); see Figure 3 for an illustration. Secondly, it is a smooth approximation of a Hertzian half-sine contact acceleration profile, with timescale $4\epsilon/c_s$ (see Section 4.1). Thirdly, while g_ϵ is only second-order ($g_\epsilon(t) = O(t^{-2})$ as $t \to \infty$), we can form linear combinations of g_ϵ with varying ϵ to achieve higher-order falloff, just like the multiscale extrapolation in [15]; in this case, our f_ϵ achieves fourth-order falloff ($O(t^{-4})$).

The final reason is that we can analytically derive the closed-form expression for $u_\epsilon(r,t)$ that is provided in (28) of the appendix. Our regularization eliminates the three singularities at the wavefronts and leaves an integrable, fixed $1/r$ singularity at the origin. In the supplemental material[1] we show that there are no branch cut crossings (a common type of numerical artifact in complex functions) when $\nu \leq 0.2631$. We still observe branch cut issues when $\nu > 0.2631$, which is when κ_2^2, κ_3^2 become complex. We recommend using a piecewise polynomial regularization function (see Conclusion Section 6.1) to deal with the branch cuts.

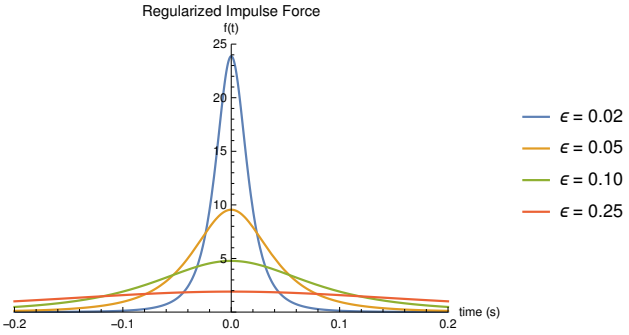


Figure 3: **Smoothed delta function used as the impulse force profile** $f_\epsilon(t)$. Here ϵ is in meters, the horizontal axis is time in seconds, and the vertical axis is scaled force.

4. SOUND SYNTHESIS

4.1. Impulse profile approximation

Similar to [5, 16], we model the acceleration $a(t)$ using the Hertz contact model. To avoid a discontinuous jerk we approximate the half-sine force with our fourth-order temporal kernel, with ϵ/c_s set to one-fourth the contact timescale t_c:

$$f(t) \approx Jf_\epsilon(t), \tag{10}$$

$$4\epsilon = c_s t_c = 2.87c_s\left(\frac{m^2}{a_0E^{*2}v_n}\right)^{1/5}, \tag{11}$$

where a_0, m, E^*, J, v_n are the object's local radius of curvature, mass, effective stiffness, impulse, and normal impact velocity.

4.2. Direct sound synthesis via Rayleigh integration

Assuming no scattering or absorption from nearby objects, the Rayleigh integral [17] says the sound pressure at a point $(\mathbf{r}, z)$ due to the plane vibration source is equal to

$$p(\mathbf{r},z,t) = \rho_0\int_{\mathbb{R}^2}\frac{a_\epsilon(\mathbf{r}', t-R'/c_0)}{2\pi R'}d\mathbf{r}', \tag{12}$$

where $R' = \sqrt{|\mathbf{r}-\mathbf{r}'|^2+z^2}$, ρ_0 is air density, and c_0 is the speed of sound in air.

We evaluate this integral numerically in Wolfram Mathematica. We found that the singularity at the origin ($r = 0$), mentioned in Section 2.3, does not cause issues: to check, we experimented with modified versions of u_ϵ where in each version we subtract out a ramp $R(r)$ of radius H times the singularity and add back in a ramp $CR(r)$ scaled to have the same average value (from analytically integrating about the origin), and we found that numerically the results were identical to those from the unmodified u_ϵ. We tested radii of $H = 0.01$ m, 0.02 m, and 0.10 m.

4.3. Floor sound shader for FDTD acoustic wavesolvers

We implemented our floor acceleration model in a general-purpose wavesolver [9] that incorporates the scattering of nearby objects. It solves the acoustic wave equation with Neumann boundary conditions

$$\frac{\partial^2 p(\mathbf{x},t)}{c_0^2 \partial t^2} = \nabla^2 p(\mathbf{x},t) + \frac{\alpha}{c_0}\nabla^2 \frac{\partial}{\partial t} p(\mathbf{x},t), \qquad x \in \Omega; \quad (13)$$

$$\partial_n p(\mathbf{x},t) = -\rho_0 a_n(\mathbf{x},t), \qquad x \in \partial\Omega, \quad (14)$$

by discretizing a region of space onto a rectangular grid and timestepping it with finite differences (see [9] for details); here Ω is the air region, $\partial\Omega$ is the boundary with objects, the subscript n indicates the normal direction, and we set the air viscosity damping coefficient $\alpha = 2\text{E-}6$ m. The wavesolver samples the boundary normal acceleration $a_n(\mathbf{x},t)$ through acoustic shaders.

We implemented the floor acceleration model as an "acoustic shader" which evaluates the regularized acceleration $a_f(r,t)$ due to each contact impulse, where r is the distance, projected onto the ground plane, between the shader's sample point $\mathbf{x}$ and the floor impact location. Since there is theoretically an object in contact at the contact point and therefore no adjacent fluid cells, we do not evaluate an acceleration there; therefore the singularity at the contact point ($r = 0$) does not cause a problem.

For consistency, we modified the acceleration shader in [9] to use the same smooth force profile and impulse evaluation constraints as our ground shader. This also corrects for any amplitude or spectral mismatches between acceleration noise and ground sound.

5. RESULTS

Sound samples for our results are available online.[1]

5.1. Model Validation

The push-like volume displacement D is given by

$$D(t) = \int_{\mathbb{R}^2} u_\epsilon(\mathbf{r},t)\, d\mathbf{r}. \quad (15)$$

We evaluate this on a scenario with a small stainless steel ball dropped onto a medium density fiberboard ground and make sure that the volume displacement is consistent with the unregularized Pekeris solution. Relevant parameters are given in Table 1.

We examined the response to a push load with our temporal regularization. Figure 2 plots the vertical displacement at a point 1 m away, and Figure 4 plots the total volume displacement. The curves converge to the Pekeris solution as ϵ decreases, and asymptotically, each $D(t)$ converges to the correct value as $t \to \infty$.

We also examined the volume displacement, the volume flux, and the momentum flux in response to an impulse. These are each defined as integrating w_ϵ, dw_ϵ/dt, and a_ϵ over the $\mathbb{R}^2$ plane. As expected, their curves look like the derivatives of those in Figure 4.

5.2. Sound Synthesis Results

5.2.1. FDTD Synthesis Examples

We added our ground surface acceleration shader to the time domain simulation system from [9]. We also use the modal shader and the acceleration noise shader, which synthesize impact sound for objects. We show a few notable examples in Figures 5, 6, and 7. In each example the modal sound is almost inaudible.

Figure 5 shows 13 steel balls with a 2 cm diameter hitting a concrete ground from various heights between 3 cm and 23 cm above ground, and Figure 6 shows these balls hitting a soil ground. Each ball has no audible ringing modes. In both examples the sound from the acceleration noise and the ground have similar frequency spectra. The concrete ground smooths the total sound of the steel ball collision; however, the short duration of the transient sound makes it difficult to discern the sound spectrum. On the other hand, the soil greatly amplifies the total sound from the steel ball collision. Since the ball-soil collision has a longer timescale than the ball-concrete collision, we can hear that the soil sound has a slightly different shape than the ball sound, making the ground relevant.

Figure 7 shows a spherical granite rock with a 30 cm diameter dropped from a height of 25 cm above ground (centroid at 40 cm). The only audible ringing modes are at much higher frequencies than the contact timescale, hence they were soft, with a peak am-

[1] http://graphics.stanford.edu/papers/ground/

Parameter	Value
Ball Material	Stainless Steel (see Table 2)
Ground Material	Wood (see Table 2)
Ball Diameter ($2a_0$)	2 cm
Drop Distance	15 cm
Restitution Coefficient (κ)	0.5
Impact Location	(0, 0, 0) m
Listening Location ($\mathbf{R}$)	(0, 0, 0.2) m
c_s	2422 m/s
Contact Time (t_c)	1.633E-4 s
Contact Radius (r_c)	6.316E-4 m
$\epsilon = c_s t_c/4$	9.888E-2 m

Table 1: **"Ball Drop" Simulation Parameters:** Scenario information for the validation, the steel ball, wood ground example in Figure 8, and the comparisons in Table 3. The lowest frequency nontorsional vibration mode for the steel ball is at 131 kHz, so we omit modal sound. Note that ϵ is much larger than the contact radius r_c, implying that temporal regularization has a much larger smoothing effect than spatial. These parameters are used in the rest of the results unless stated otherwise.

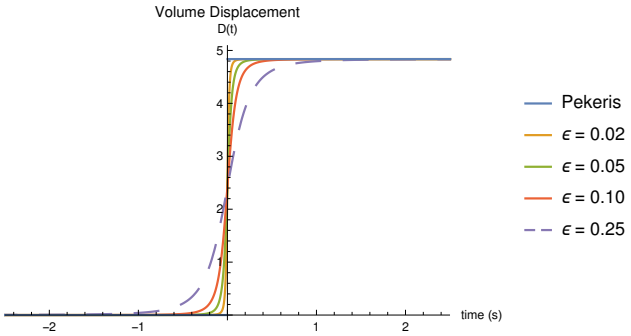


Figure 4: **Volume displacement,** $D(t)$**:** Here ϵ is in meters, and the vertical axis is volume displacement scaled by the same factor as in Figure 2. The modified temporal regularization with a smoothed origin proposed in Section 4.2 has a volume displacement plot that looks identical.

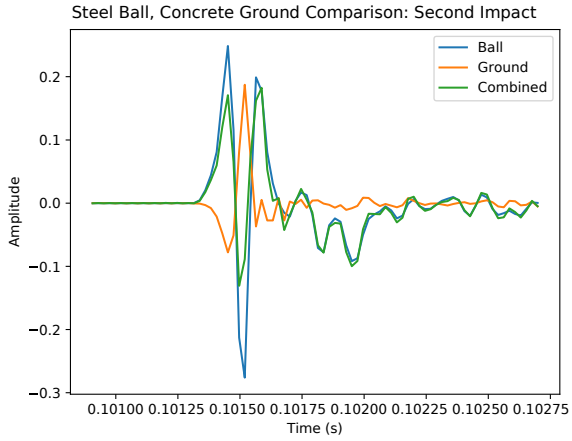


Figure 5: An example with 13 balls dropped from various heights onto a concrete ground, simulated with our wavesolver. See the supplemental material for the sound. Each sound (ball, ground, combined) is normalized to 10 Pa. The listening point is at (0.20, 0.12, 0.16) m, with the z coordinate specifying the height.

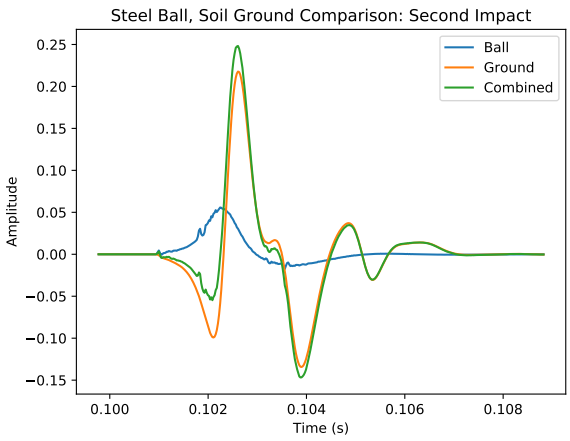


Figure 6: An example with 13 balls dropped from various heights onto soil ground, simulated with our wavesolver. See the supplemental material for the sound. Each sound (ball, ground, combined) is normalized to 4.5 Pa. The listening point is at (0.20, 0.12, 0.16) m.

plitude of 0.106 Pa. In comparison, the acceleration noise was at a peak amplitude of 1.76 Pa, and the ground contributed a noticeable rumble peaking at 17.2 Pa.

5.2.2. *Ball ground impact comparisons*

Similar to prior work [16], we can use a closed-form expression to model the sound from a small ball. We treat it as a compact translating sphere, which forms an acoustic dipole source. The far-field acoustic pressure depends on the jerk, with a $1/r$ falloff. The nearfield pressure depends on the acceleration with a $1/r^2$ falloff. The final expression, according to Eq (6.20) in [18], is

$$p(\mathbf{r},t) = \frac{\rho_0 a_0^3 \cos(\theta)}{2}\left(-\frac{a(t-\frac{r-a_0}{c_0})}{r^2} + \frac{\frac{da}{dt}(t-\frac{r-a_0}{c_0})}{c_0 r}\right) \tag{16}$$

where $a(t)$ is the acceleration of the ball at time t and θ is the angle between the acceleration and $\mathbf{r}$. We assume perfect reflection and model it by adding the reflection image source of this ball, reflecting the dipole direction and position over the y axis. The total is a longitudinal quadrupole source for hard reflective grounds, and a dipole source for absorptive grounds.

We model the acceleration with the same fourth-order temporal force as that used for the ground in section 4.1.

$$a(t) = -f(t)/m. \tag{17}$$

We simply use $(1+\kappa)mv_n$ as the impulse, where κ is the coefficient of restitution of the collision.

Figure 8 illustrates an ideal 2 cm steel ball, wood ground impact, with their respective amplitudes. We verified the amplitudes from our wavesolver against these amplitudes. For harder ground materials such as concrete, or lighter object materials such as ceramic, wood, or dice, the ground sound would be much softer compared to the ball sound. The next section generalizes this observation.

Material Property Reference

Material	E (Pa)	ν	ρ (kg m^{-3})
Stainless Steel	1.965E+11	0.27	7955
Ceramics	7.2E+10	0.19	2700
Granite	5.07E+10	0.28	2670
Concrete	1.85E+10	0.20	2250
Wood	1.1E+10	0.25	750
Plastic (ABS)	1.4E+9	0.35	1070
Soil	4.0E+7	0.25	1350
Paraffin Wax	5.57E+7	0.37	786

Table 2: **Material parameters used for common materials:** The Young's modulus is E, Poisson's ratio is ν, and density is ρ. We used medium density fiberboard for wood.

5.3. Impact Sound Parameter Dependence

Let us describe the impact scenario with the parameters (t_c, a_0, v_n, κ, E_f, ν_f, c_s, ρ_b, R, θ), where the subscript f indicates ground, b indicates ball, and (R,θ) indicate the listening point distance and elevation angle. We hereby fix all parameters to their Table 1 values and vary just one or two of them at a time.

ρ_b, E_f: By algebra, the ground sound amplitude is proportional to ρ_b/E_f, while the ball sound stays constant. Table 2 lists these properties for common materials, and Table 3 lists the intensity ratio for each material pair.

ν_f: We found that changing the ground Poisson's ratio does not significantly affect either sound amplitude.

t_c: Figure 9 discusses the dependence on contact timescale for one example. In the far field ($R \gg c_0 t_c$) both the ground and the ball sound intensity have similar power law dependence.

θ: Figure 11 shows the dependence on listening point angle from the plane. As the listening point gets closer to the plane, the ball sound gets softer at a faster rate than the ground sound.

c_s: The ball sound does not depend on c_s, the speed of shear waves in the ground, and Figure 10 discusses the ground sound dependence on c_s. The ground amplitude increases linearly in proportion to c_s until a threshold $c_k \approx A\sqrt{c_0 R/t_c}$ determined by the

Relative Intensities (dB) of Ground Sound Compared to Ball Sound

ball \ ground	Steel	Ceramics	Granite	Concrete	Wood	Plastic	Soil	Wax
Steel	-30.25	-21.30	-18.94	-11.83	-6.12	4.15	19.06	19.58
Ceramics	-39.63	-30.69	-28.33	-21.22	-15.51	-5.23	9.68	10.19
Granite	-39.73	-30.78	-28.43	-21.32	-15.60	-5.33	9.58	10.10
Concrete	-41.21	-32.27	-29.91	-22.80	-17.09	-6.81	8.09	8.61
Wood	-50.76	-41.81	-39.46	-32.34	-26.63	-16.36	-1.45	-0.93
Plastic	-47.67	-38.73	-36.37	-29.26	-23.55	-13.27	1.64	2.15
Soil	-45.65	-36.71	-34.35	-27.24	-21.53	-11.25	3.65	4.17
Wax	-50.35	-41.41	-39.05	-31.94	-26.22	-15.95	-1.04	-0.53

Table 3: **Theoretical relative intensity (dB) of ground to ball sound**, for the scenario in Table 1. Ball materials are listed on the left, ground on the top. Positive values indicate the ground was louder than the ball. Impact timescale was kept constant at 1.63E-4 s and Poisson's ratio at 0.25, as neither significantly affect relative amplitude. Scenarios with louder ground sound ($\geq$ 0 dB) are highlighted in teal, and scenarios where the ground sound can be audible (above the most sensitive JND level of -13 dB [19]) are highlighted in light orange. Note that our overhead listening point is near the maximum relative loudness for the ball, whereas low listening angles tend to receive more ground sound (Figure 11 expands on this relationship).

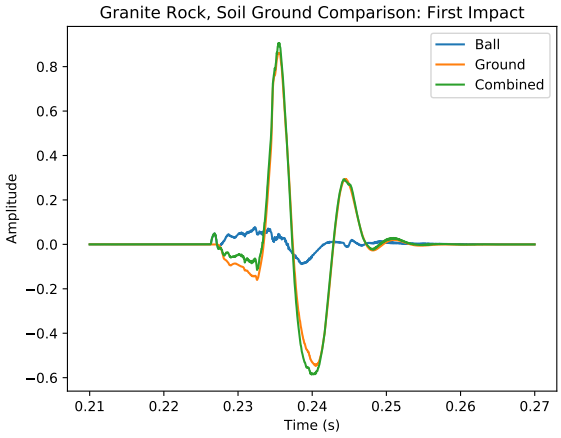


Figure 7: An example with a 30 cm spherical granite rock dropped from 25 cm above ground onto soil, simulated with our wave-solver. See the supplemental material for the sound. Each sound (rock, ground, combined) is normalized to 20 Pa. For the almost-silent modal component, we used the modal shader used in [9] with Rayleigh damping parameters $\alpha = 6$, $\beta =$ 1E-7, in SI units. The listening point is at (0.45, 0.27, 0.48) m.

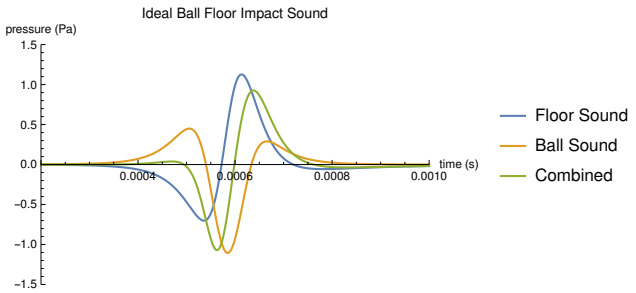


Figure 8: Ideal unobstructed sound for a 2 cm steel ball dropped from 15 cm impacting a wood ground with restitution coefficient 0.5. The listening point is 20 cm directly above the impact point. The quadrupole shape of the ball sound is different from the ground sound, but at high frequencies the frequency content sounds similar and it is hard to tell perceptually. The ground sound adds a significant amount of amplitude to the combined sound, and the combined sound seems to be higher pitched than either sound.

contact timescale t_c and the listening point distance R.

a_0, v_n, κ, R: In the far field, they affect both sounds equally.

5.4. Discussion

We found that in most everyday scenarios with rigid objects and listening points with high elevation angle, the ground sound would be masked by the object sound: the amplitude of the ball sound is louder, the frequency content is similar, and the contact timescale is often too short to hear the distinct waveforms. In these scenarios, namely the unhighlighted cells in Table 3, we can omit the ground sound.

If the object is dense and the ground has a low shear modulus, then the ground sound can be as loud or louder than the object's acceleration noise. Furthermore, the contact timescale can be slow enough for us to hear the difference between the object and ground sounds. In a few examples we examined, such as steel or granite objects hitting wood, concrete, and soil, the modal ringing sound for the object is too soft, but for larger, less round, and softer objects, the modal ringing sound can dominate the total power output.

6. CONCLUSION AND FUTURE WORK

We regularized the solution to Lamb's problem to give us a closed-form expression for ground surface acceleration. For impacts from small balls, we used a Rayleigh integral to compute ground sound amplitudes and compared them with object acceleration noise. Furthermore, we implemented an acoustic shader in an FDTD wave-solver to synthesize sound from generic object impacts with the ground, combining modal sound, acceleration noise, and ground sound. We found that the ground sound is more important when the listening point is at a low angle, when the ground has a low shear modulus, or when the object has a high density. Furthermore, ground noise (similar to acceleration noise) is important only for objects where modal ringing noise, which is louder in larger objects, was not audible. In the absence of modal sound, the relative importance of ground sound was not affected by object size in "ball

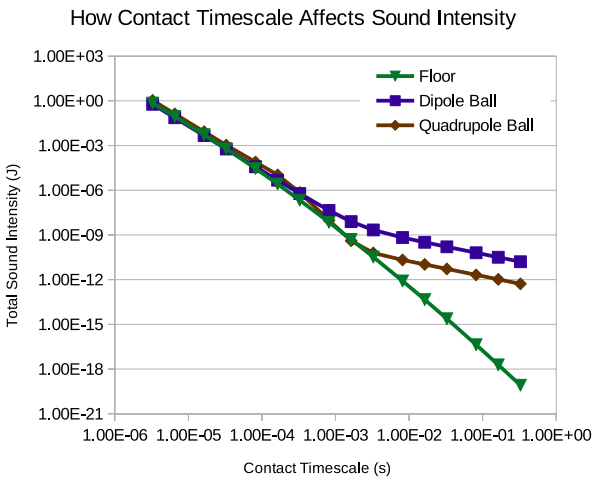


Figure 9: **Sound dependence on contact timescales** measured overhead at $z = 20$ cm. For low timescales, the ball and ground both have a τ^{-3} dependence; however, at high timescales, the near-field term of the ball sound dominates, and its power falls off as τ^{-3}

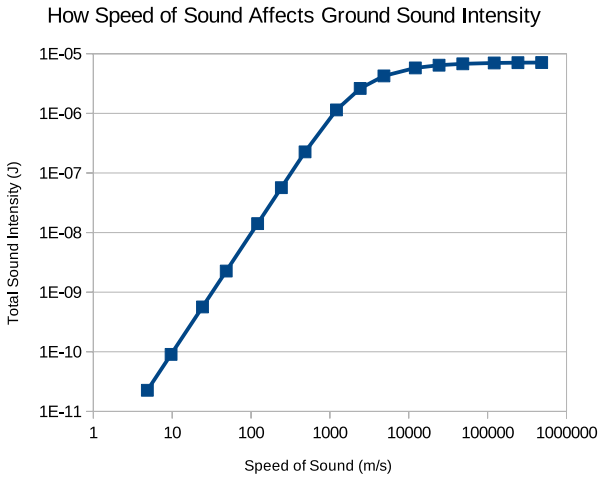


Figure 10: **Ground sound dependence on** c_s measured overhead at $z = 20$ cm. At low c_s, the ground sound intensity is proportional to c_s^2, and at high c_s, it is constant. The knee cutoff, c_k, is about 2576 m/s. By testing a few more parameters, we experimentally determined that $c_k \approx A\sqrt{c_0 R/t_c}$, where c_0 is the air speed of sound, R is the listening point distance, t_c is the contact timescale, and A is a dimensionless constant between 3 and 4.

drop" tests, notwithstanding changes in contact duration.

6.1. Limitations and future work

Our work has several limitations that motivate future work:

1. *Stable numerical evaluation for general ν:* Our model crosses a discontinuous branch cut when evaluated for high $\nu \geq 0.2631$. We were unable to express the regularized response u_ϵ in a form that eliminates this branch cut. We explored an alternate regularization using a piecewise polynomial $f_\epsilon = (1 - (t/\epsilon)^2)^n$ for $|t| < \epsilon$, and this gives an expression that does not have the branch jump. However, we used $n = 4$ to get a continuous acceleration, and the degree-8 polynomial produces a result that suffers from catastrophic cancellation when ϵ is small. Future work should ensure stable numerical evaluation for all ν values.

2. *Finite-depth ground and realistic flooring:* Our ground sound model applies well for ground that is homogeneous for a very deep layer, greater than approximately 50 m deep. For shallower ground layers, the reflections between the layer boundaries form resonance modes that our model does not capture. Furthermore, when an object is dropped onto a hard floor in a building, we hear the vibrational response of the building. Future work could model the responses of more realistic building and flooring structures.

3. *Tangential frictional loads:* We only modeled the vertical response to a vertical load. Future work can regularize the closed-form solutions for a vertical response to a tangential load, such as incurred by contact friction.

4. *No closed-form sound:* We provided an expression for surface acceleration but not the sound. Future work could derive a model for the final sound based on listening position.

7. ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their feedback. We acknowledge support from the National Science Foundation (NSF) under grant DGE-1656518, the Toyota Research Institute (TRI), and Google Cloud Platform compute resources. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF, the TRI, any other Toyota entity, or others.

8. REFERENCES

[1] Perry R Cook, "Modeling Bill's gait: Analysis and parametric synthesis of walking sounds," in *AES 22nd Conf: Virtual, Synthetic, and Entertainment Audio*. Audio Eng Soc, 2002.

[2] Luca Turchet, Stefania Serafin, Smilen Dimitrov, and Rolf Nordahl, "Physically based sound synthesis and control of footsteps sounds," *Proc. of the 13th Int. Conf. on Digital Audio Effects (DAFx-10)*, 2010.

[3] Changxi Zheng and Doug L James, "Rigid-body fracture sound with precomputed soundbanks," in *ACM Transactions on Graphics (TOG)*. ACM, 2010, vol. 29, p. 69.

[4] Changxi Zheng and Doug L James, "Toward high-quality modal contact sound," in *TOG*. ACM, 2011, vol. 30, p. 38.

[5] Sota Nishiguchi and Katunobu Itou, "Modeling and rendering for virtual dropping sound based on physical model of rigid body," *Proc. of the 21st Int. Conf. on Digital Audio Effects (DAFx-18)*, 2018.

[6] Clara Issanchou, Stefan Bilbao, Jean-Loic Le Carrou, Cyril Touzé, and Olivier Doaré, "A modal-based approach to the nonlinear vibration of strings against a unilateral obstacle: Simulations and experiments in the pointwise case," *Journal of Sound and Vibration*, vol. 393, pp. 229–251, 2017.

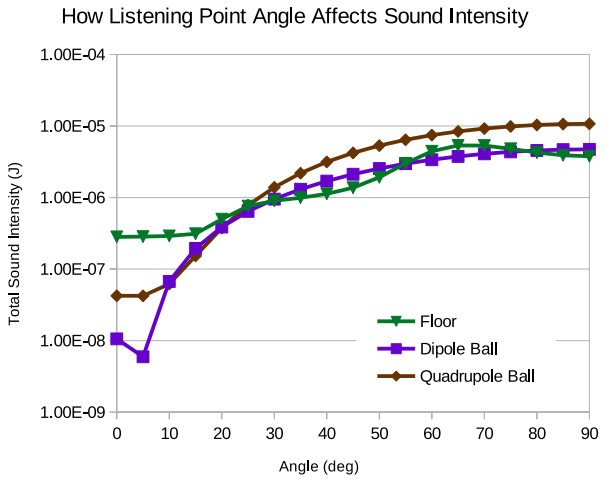


Figure 11: **Angular Dependence of Sound Intensity:** The listening point is 20 cm away, with 90°being directly overhead, and 0°in the plane. The dipole ball model has a minimum at 5° because its center is 1 cm above the ground. *Observe that the ground sound is significantly louder than the ball sound at low angles.*

[7] Nikunj Raghuvanshi, Rahul Narain, and Ming C Lin, "Efficient and accurate sound propagation using adaptive rectangular decomposition," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 5, pp. 789–801, 2009.

[8] Stefan Bilbao, Alberto Torin, and Vasileios Chatziioannou, "Numerical modeling of collisions in musical instruments," *Acta Acustica united with Acustica*, vol. 101, pp. 155–, 01 2015.

[9] Jui-Hsien Wang, Ante Qu, Timothy R Langlois, and Doug L James, "Toward wave-based sound synthesis for computer animation," *ACM Trans Graph*, vol. 37, no. 4, pp. 109, 2018.

[10] Horace Lamb, "I. on the propagation of tremors over the surface of an elastic solid," *Phil Trans of the Royal Society A*, vol. 203, no. 359-371, pp. 1–42, 1904.

[11] Fernando De Goes and Doug L James, "Dynamic Kelvinlets: Secondary motions based on fundamental solutions of elastodynamics," *ACM TOG*, vol. 37, no. 4, pp. 81, 2018.

[12] CL Pekeris, "The seismic surface pulse," *Proceedings of the national academy of sciences of the United States of America*, vol. 41, no. 7, pp. 469, 1955.

[13] Harold M Mooney, "Some numerical solutions for Lamb's problem," *Bulletin of the Seismological Society of America*, vol. 64, no. 2, pp. 473–491, 1974.

[14] Eduardo Kausel, "Lamb's problem at its simplest," *Proc of the Royal Society A*, vol. 469, no. 2149, pp. 20120462, 2013.

[15] Fernando De Goes and Doug L James, "Regularized Kelvinlets: Sculpting brushes based on fundamental solutions of elasticity," *ACM Trans Graph*, vol. 36, no. 4, pp. 40, 2017.

[16] Jeffrey N Chadwick, Changxi Zheng, and Doug L James, "Precomputed acceleration noise for improved rigid-body sound," *ACM Trans Graph*, vol. 31, no. 4, pp. 103, 2012.

[17] David T Blackstock, *Fundamentals of physical acoustics*, John Wiley and Sons, New York, 2001.

[18] SW Rienstra and A Hirschberg, "An introduction to acoustics," *Eindhoven University of Technology*, 2004.

[19] Marshall Long, "3 - human perception and reaction to sound," in *Architectural Acoustics (Second Edition)*, pp. 81 – 127. Academic Press, Boston, 2014.

A. DERIVATION OF REGULARIZED RESPONSE

In this derivation, we let $t' = c_s t$, and we note that at the end, we need to scale by the right power of c_s.

$$g_\epsilon(t') = \frac{\epsilon/\pi}{t'^2 + \epsilon^2} \tag{18}$$

We want to find the convolution $k'_\epsilon = g_\epsilon(t') * u_n(r, t')$. This represents the displacement response to an arctan load, which approximates the Heaviside theta load. Define $\mathcal{U}, \mathcal{W}, \mathcal{V}$, as the following

$$\mathcal{U}_\epsilon(t', \sigma) = \frac{1}{r}\int_\sigma^\infty g_\epsilon(t' - s)ds; \tag{19}$$

$$\mathcal{V}_\epsilon(t', s, \alpha) = \int \frac{g_\epsilon(t' - s)}{\sqrt{s^2 - \alpha^2}}ds; \tag{20}$$

$$\mathcal{W}_\epsilon(t', s, \alpha) = \int \frac{g_\epsilon(t' - s)}{\sqrt{\alpha^2 - s^2}}ds. \tag{21}$$

Integrating,

$$\mathcal{U}_\epsilon(t', \sigma) = \frac{1}{2r} + \frac{1}{\pi r}\arctan\left(\frac{t' - \sigma}{\epsilon}\right); \tag{22}$$

$$Z_\epsilon(t', \alpha) = \sqrt{\alpha^2 + (\epsilon - it')^2}; \tag{23}$$

$$\mathcal{V}_\epsilon(t', s, \alpha) = \mathrm{Re}\left(\frac{1}{\pi Z_\epsilon(t', \alpha)}\left(-\log(\epsilon - i(t' - s)) + \log(\alpha^2 - (t' + i\epsilon)s - iZ_\epsilon(t', \alpha)\sqrt{s^2 - \alpha^2})\right)\right); \tag{24}$$

$$\mathcal{W}_\epsilon(t', s, \alpha) = \mathrm{Im}\left(\frac{-1}{\pi Z_\epsilon(t', \alpha)}\left(-\log(\epsilon - i(t' - s)) + \log(\alpha^2 - (t' + i\epsilon)s + Z_\epsilon(t', \alpha)\sqrt{\alpha^2 - s^2})\right)\right). \tag{25}$$

Check the Mathematica notebook on the website[1] for verification.

Plugging in the integration limits, the convolution k' is

$$k'_\epsilon(r, t') = \frac{1 - \nu}{4\pi\mu}\left(\mathcal{U}_\epsilon(t', ar) + \mathcal{U}_\epsilon(t', r) + 2\mathcal{W}_\epsilon(t', \gamma r, \gamma r) - \mathcal{W}_\epsilon(t', r, \gamma r) - \mathcal{W}_\epsilon(t', ar, \gamma r) + \sum_{j=2}^{3}(\mathcal{V}_\epsilon(t', r, \kappa_j r) - \mathcal{V}_\epsilon(t', ar, \kappa_j r))\right). \tag{26}$$

Our final expression, in terms of the original t, is

$$k_\epsilon(r, t) = k'_\epsilon(r, c_s t), \tag{27}$$

that is, there is no missing c_s scale factor because the extra c_s from the convolution is cancelled by the missing c_s from normalizing g_ϵ. For fourth-order, we simply take

$$u_\epsilon(r, t) = 2k_\epsilon(r, t) - k_{2\epsilon}(r, t). \tag{28}$$

In the supplemental material[1] we show that when $\nu \in [0, 0.2631)$, this solution does not cross any branch cuts as we vary (r, t).

THE SHAPE OF REMIXXXES TO COME: AUDIO TEXTURE SYNTHESIS WITH TIME-FREQUENCY SCATTERING

Vincent Lostanlen

Music and Audio Research Lab
Steinhardt School of Culture, Education, and Human Development
New York University
New York, NY, USA
vincent.lostanlen@nyu.edu

Florian Hecker

Edinburgh College of Art
College of Arts, Humanities, and Social Sciences
University of Edinburgh
Edinburgh, UK
florian.hecker@ed.ac.uk

ABSTRACT

This article explains how to apply time–frequency scattering, a convolutional operator extracting modulations in the time–frequency domain at different rates and scales, to the re-synthesis and manipulation of audio textures. After implementing phase retrieval in the scattering network by gradient backpropagation, we introduce *scale-rate DAFx*, a class of audio transformations expressed in the domain of time–frequency scattering coefficients. One example of scale-rate DAFx is chirp rate inversion, which causes each sonic event to be locally reversed in time while leaving the arrow of time globally unchanged. Over the past two years, our work has led to the creation of four electroacoustic pieces: *FAVN*; *Modulator (Scattering Transform)*; *Experimental Palimpsest*; *Inspection (Maida Vale Project)* and *Inspection II*; as well as *XAllegroX (Hecker Scattering.m Sequence)*, a remix of Lorenzo Senni's *XAllegroX*, released by Warp Records on a vinyl entitled *The Shape of RemiXXXes to Come*.

1. INTRODUCTION

Several composers have pointed out the lack of a satisfying trade-off between interpretability and flexibility in the parametrization of sound transformations [1, 2, 3]. For example, the constant-Q wavelet transform (CQT) of an audio signal provides an intuitive display of its short-term energy distribution in time and frequency [4], but does not give explicit control over its intermittent perceptual features, such as roughness or vibrato. On the other hand, a deep convolutional generative model such as WaveNet [5] encompasses a rich diversity of timbre; but, because the mutual dependencies between the dimensions of its latent space are unspecified, music composition with autoencoders in the waveform domain is hampered by a long preliminary phase of trials and errors in the search for the intended effect.

Scattering transforms are a class of multivariable signal representations at the crossroads between wavelets and deep convolutional networks [6]. In this paper, we demonstrate that one such instance of scattering transform, namely time–frequency scattering [7], can be a relevant tool for composers of electroacoustic music, as it strikes a satisfying compromise between interpretability and flexibility. We describe the scattering-based DAFx underlying the synthesis of five electroacoustic pieces by Florian Hecker: *FAVN* (2016); *Modulator (Scattering Transform)* (2016-2017); *Experimental Palimpsest* (2016); *Inspection (Maida Vale Project)*(2016) and *Inspection II* (2017); as well as *XAllegroX (Hecker Scattering.m Sequence)*, a remix of Lorenzo Senni's *XAllegroX*, released by Warp Records on a vinyl entitled *The Shape of RemiXXXes to Come* (2017). In addition, we demonstrate the result of this algorithm in the companion website of this paper, which contains short audio examples as well as links to full-length computer-generated sound pieces.

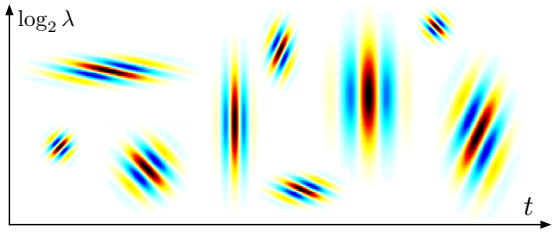


Figure 1: Interference pattern between wavelets $\psi_\alpha(t)$ and $\psi_\beta(\log_2 \lambda)$ in the time–frequency domain $(t, \log_2 \lambda)$ for different combinations of amplitude modulation rate α and frequency modulation scale β. Darker shades of red (resp. blue) indicate higher positive (resp. lower negative) values of the real part. See Section 2 for details.

Section 2 defines time–frequency scattering. Section 3 presents a gradient backpropagation method for sound synthesis from time–frequency scattering coefficients. Section 4 introduces "scale-rate DAFx", a new class of DAFx which operates in the domain of spectrotemporal modulations, and describes the implementation of chirp reversal as a proof of concept.

2. TIME–FREQUENCY SCATTERING

In this section, we define the time–frequency scattering transform as a function of four variables — time t, frequency λ, amplitude modulation rate α, and frequency modulation scale β — which we connect to spectrotemporal receptive fields (STRF) in auditory neurophysiology [8]. We refer to [9] for an in-depth mathematical introduction to time–frequency scattering.

This work is supported by the ERC InvariantClass grant 320959. The source code to reproduce experiments and figures is made freely available at: https://github.com/lostanlen/scattering.m

Companion website: https://lostanlen.com/pubs/dafx2019

2.1. Spectrotemporal receptive fields

Time–frequency scattering results from the cascade of two stages: a constant-Q wavelet transform (CQT) and the extraction of spectrotemporal modulations with wavelets in time and log-frequency. First, we define Morlet wavelets of center frequency $\lambda > 0$ and quality factor Q as

$$\boldsymbol{\psi}_\lambda(t) = \lambda \exp\left(-\frac{\lambda^2 t^2}{2Q^2}\right) \times (\exp(2\pi \mathrm{i} \lambda t) - \kappa), \qquad (1)$$

where the corrective term κ ensures that each $\boldsymbol{\psi}_\lambda(t)$ has one vanishing moment, i.e. a null average. In the sequel, we set $Q = 12$ to match twelve-tone equal temperament. Within a discrete setting, acoustic frequencies λ are typically of the form $2^{n/Q}$ where n is integer, thus covering the hearing range. For $\boldsymbol{x}(t)$ a finite-energy signal, we define the CQT of $\boldsymbol{x}$ as the matrix

$$\mathbf{U}_1\boldsymbol{x}(t, \lambda) = |\boldsymbol{x} * \boldsymbol{\psi}_\lambda|\,(t), \qquad (2)$$

that is, stacked convolutions with all wavelets $\boldsymbol{\psi}_\lambda(t)$ followed by the complex modulus nonlinearity.

Secondly, we define Morlet wavelets of respective center frequencies $\alpha > 0$ and $\beta \in \mathbb{R}$ with quality factor $Q = 1$. With a slight abuse of notation, we denote these wavelets by $\boldsymbol{\psi}_\alpha(t)$ and $\boldsymbol{\psi}_\beta(\log \lambda)$ even though they do not necessarily have the same shape as the wavelets $\boldsymbol{\psi}_\lambda(t)$ of Equation 2. Frequencies α, hereafter called amplitude modulation *rates*, are measured in Hertz (Hz) and discretized as 2^n with integer n. Frequencies β, hereafter called frequency modulation *scales*, are measured in cycles per octave (c/o) and discretized as $\pm 2^n$ with integer n. The edge case $\beta = 0$ corresponds to $\boldsymbol{\psi}_\beta(\log \lambda)$ being a Gaussian low-pass filter $\boldsymbol{\phi}_F(\log \lambda)$ of bandwidth F^{-1}. These modulation scales β play the same role as the *quefrencies* in a mel-frequency cepstrum [7].

We define the spectrotemporal receptive field (STRF) of $\boldsymbol{x}$ as the fourth-order tensor

$$\begin{aligned}\mathbf{U}_2\boldsymbol{x}(t, \lambda, \alpha, \beta) &= \left|\mathbf{U}_1\boldsymbol{x} \overset{t}{*} \boldsymbol{\psi}_\alpha \overset{\log_2 \lambda}{*} \boldsymbol{\psi}_\beta\right|(t, \lambda) \\ &= \left| \iint \mathbf{U}_1\boldsymbol{x}(\tau, s) \boldsymbol{\psi}_\alpha(t - \tau) \boldsymbol{\psi}_\beta(\log_2 \lambda - s)\, \mathrm{d}\tau\, \mathrm{d}s \right|, \qquad (3)\end{aligned}$$

that is, stacked convolutions in time and log-frequency with all wavelets $\boldsymbol{\psi}_\alpha(t)$ and $\boldsymbol{\psi}_\beta(\log_2 \lambda)$ followed by the complex modulus nonlinearity [10]. Figure 1 shows the interference pattern of the product $\boldsymbol{\psi}_\alpha(t - \tau)\boldsymbol{\psi}_\beta(\log_2 \lambda - s)$ for different combinations of time t, frequency λ, rate α, and scale β. We denote the multiindices (λ, α, β) resulting from such combinations as scattering *paths* [11]. We refer to [12] for an introduction to STRFs in the interdisciplinary context of music cognition and music information retrieval (MIR), and to [13] for an experimental benchmark in automatic speech recognition.

2.2. Invariance to translation

Because it is a convolutional operator in the time–frequency domain, the STRF is equivariant to temporal translation $t \mapsto t + \tau$ as well as frequency transposition $\lambda \mapsto 2^s\lambda$. In audio classification, it is useful to guarantee invariance to temporal translation up to some time lag T [11]. To this aim, we define time–frequency scattering as the result of a local averaging of both $\mathbf{U_1}\boldsymbol{x}(t, \lambda)$ and $\mathbf{U}_2\boldsymbol{x}(t, \lambda, \alpha, \beta)$ by a Gaussian low-pass filter $\boldsymbol{\phi}_T$ of cutoff frequency equal to T^{-1}, yielding

$$\mathbf{S}_1\boldsymbol{x}(t, \lambda) = \left(\mathbf{U}_1\boldsymbol{x} \overset{t}{*} \boldsymbol{\phi}_T\right)(t, \lambda) \quad \text{and} \qquad (4)$$

$$\mathbf{S}_2\boldsymbol{x}(t, \lambda, \alpha, \beta) = \left(\mathbf{U}_2\boldsymbol{x} \overset{t}{*} \boldsymbol{\phi}_T\right)(t, \lambda, \alpha, \beta) \qquad (5)$$

respectively. In practice, for purposes of signal classification, T is of the order of $50\,\mathrm{ms}$ in speech; of $500\,\mathrm{ms}$ in instrumental music; and of $5\,\mathrm{s}$ in ecoacoustics [14]. The delay of a real-time implementation of time–frequency scattering is of the order of T.

2.3. Energy conservation

We restrict the set of modulation rates α in $\mathbf{U_2}\boldsymbol{x}$ to values above T^{-1}, so that the power spectra of the low-pass filter $\boldsymbol{\phi}_T(t)$ and all wavelets $\boldsymbol{\psi}_\alpha(t)$ cover uniformly the Fourier domain [15, Chapter 4]: at every frequency ω, we have

$$\left|\widehat{\boldsymbol{\phi}}_T(\omega)\right|^2 + \frac{1}{2} \sum_{\alpha > T^{-1}} \left(\left|\widehat{\boldsymbol{\psi}}_\alpha(\omega)\right|^2 + \left|\widehat{\boldsymbol{\psi}}_\alpha(-\omega)\right|^2\right) \lessapprox 1, \qquad (6)$$

where the notation $A \lessapprox B$ indicates that there exists some $\varepsilon \ll B$ such that $B - \varepsilon < A < B$. In the Fourier domain associated to $\log_2 \lambda$, one has $\sum_\beta |\widehat{\boldsymbol{\psi}}_\beta(\omega)|^2 \lessapprox 1$ for all ω. Therefore, applying Parseval's theorem on all three wavelet filterbanks (respectively indexed by λ, α, and β) yields $\|\mathbf{S}_1\boldsymbol{x}\|_2^2 + \|\mathbf{U}_2\boldsymbol{x}\|_2^2 \lessapprox \|\mathbf{U}_1\boldsymbol{x}\|_2^2$. Figure 2 illustrates the design of these filterbanks in the Fourier domain.

The spectrotemporal modulations in music — e.g. tremolo, vibrato, and dissonance — are captured and demodulated by the second layer of a scattering network [16]. Consequently, each scattering path (λ, α, β) in $\mathbf{U}_2\boldsymbol{x}(t, \lambda, \alpha, \beta)$ yields a time series whose variations are slower than in the first layer $\mathbf{U}_1\boldsymbol{x}(t, \lambda)$; typically at rates of $1\,\mathrm{Hz}$ or lower. By setting T to 1 second or less, we may safely assume that the cutoff frequency of the low-pass filter $\boldsymbol{\phi}_T(t)$ in Equation 5 is high enough to retain all the energy in $\mathbf{U}_2\boldsymbol{x}(t, \lambda, \alpha, \beta)$. This assumption writes as $\|\mathbf{S}_2\boldsymbol{x}\| \lessapprox \|\mathbf{U}_2\boldsymbol{x}\|$ and is justified by the theorem of exponential decay of scattering coefficients [17]. Let $\mathbf{S}$ be the operator resulting from the concatenation of first-order scattering $\mathbf{S_1}$ with second-order scattering $\mathbf{S_2}$. In the absence of any DC bias in $\boldsymbol{x}(t)$, we conclude with the energy conservation identity

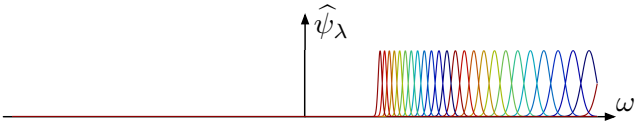
$$\|\mathbf{S}\boldsymbol{x}\|_2^2 = \|\mathbf{S}_1\boldsymbol{x}\|_2^2 + \|\mathbf{S}_2\boldsymbol{x}\|_2^2 \lessapprox \|\mathbf{U}_1\boldsymbol{x}\|_2^2 \lessapprox \|\boldsymbol{x}\|_2^2. \qquad (7)$$

3. AUDIO TEXTURE SYNTHESIS

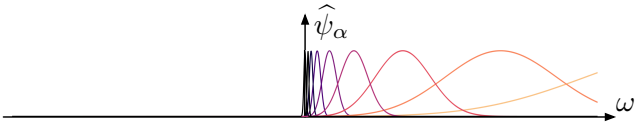
In this section, we describe how to pseudo-invert time–frequency scattering, that is, to generate a waveform whose scattering coefficients match the scattering coefficients of some other, pre-recorded waveform.

3.1. From phase retrieval to texture synthesis

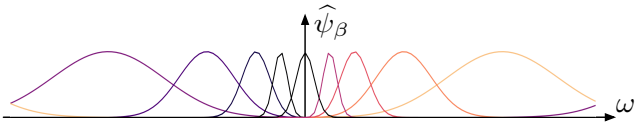
Although the invertibility of the convolutional operator involved in the constant-Q transform is guaranteed by wavelet frame theory [15, Chapter 5], the complex modulus nonlinearity in Equation 2 raises a fundamental question: is it always possible to recover $\boldsymbol{x}$, up to a constant and therefore imperceptible phase shift, from the magnitudes of its wavelet coefficients $\mathbf{U}_1\boldsymbol{x}(t, \lambda)$? This question has



(a) Wavelets $\widehat{\psi}_\lambda$ of frequencies λ and quality factor $Q = 12$.



(b) Wavelets $\widehat{\psi}_\alpha$ of rates α and quality factor $Q = 1$.



(c) Wavelets $\widehat{\psi}_\beta$ of scales β and quality factor $Q = 1$.

Figure 2: Filterbanks of Morlet wavelets in the Fourier domain: (a) for CQT; (b) for STRF, temporal dimension; (c) for STRF, log-frequential dimension. See Section 2 for details.

recently been answered in the affirmative, suggesting that a redundant CQT should be preferred over critically sampled short-term Fourier transforms (STFT) when attempting to sonify spectrograms in the absence of any prior information on the phase of the target waveform [18].

By recursion over layers in the composition of wavelet modulus operators, the invertibility of wavelet modulus implies the invertibility of scattering transforms of infinite depth [19], up to a constant time shift of at most T. However, the time–frequency scattering network presented here has a finite number of layers (i.e., most usually two layers) and is therefore not exactly invertible. Because of the theorem of exponential decay of scattering coefficients [17], the residual energy that is present in deeper layers can be neglected on the condition that T is small enough. In the rest of this section, we set T to $186\,\mathrm{ms}$, which corresponds to 8192 samples at a sample rate of $44.1\,\mathrm{kHz}$. Since the unit circle is not a convex subset of $\mathbf{R}^2$, the optimization problem $\boldsymbol{y}^*(t) = \arg\min_{\boldsymbol{y}} \mathbf{E}(\boldsymbol{y})$ where $\mathbf{E}(\boldsymbol{y}) = \frac{1}{2}\|\mathbf{S}\boldsymbol{x} - \mathbf{S}\boldsymbol{y}\|$ is nonconvex; therefore, its loss surface $\mathbf{E}$ may have local minimizers in addition to the global minimizers of the form $\boldsymbol{y}^*_\tau(t) = \boldsymbol{x}(t-\tau)$ with $|\tau| < T$. As a consequence, we formulate the problem of audio texture synthesis in loose terms of perceptual similarity. We refer to [20] for a literature review on texture synthesis, and to [21] for a discussion of quantitative evaluation procedures.

Starting from a colored Gaussian noise $\boldsymbol{y}_0(t)$ whose power spectral density matches $\mathbf{S}_1\boldsymbol{x}(t,\lambda)$, we refine it by additive updates of the form $\boldsymbol{y}_{n+1}(t) = \boldsymbol{y}_n(t) + \boldsymbol{u}_n(t)$, where the term $\boldsymbol{u}_n(t)$ is defined recursively as $\boldsymbol{u}_n(t) = m \times \boldsymbol{u}_n(t) + \mu_n \boldsymbol{\nabla}\mathbf{E}(\boldsymbol{y}_n)(t)$. In subsequent experiments, the momentum term is fixed at $m = 0.9$ while the learning rate is initialized at $\mu_0 = 0.1$ and modified at every step according to a "bold driver" heuristic [22].

3.2. Gradient backpropagation in a scattering network

Like deep neural networks, scattering networks consist of the composition of linear operators (wavelet transforms) and pointwise nonlinearities (complex modulus). Consequently, the gradient $\mathbf{E}(\boldsymbol{y}_n)$ can be obtained by composing the Hermitian adjoints of these operator in the reverse order as in the direct scattering transform — a method known as backpropagation [23].

First, we backpropagate the gradient of Euclidean loss for second-order scattering coefficients:

$$\boldsymbol{\nabla}\mathbf{U_2}\boldsymbol{y}(t,\lambda,\alpha,\beta) = \Big((\mathbf{S_2}\boldsymbol{x} - \mathbf{S_2}\boldsymbol{y}) \overset{t}{*} \boldsymbol{\phi}\Big)(t,\lambda,\alpha,\beta). \quad (8)$$

Secondly, we backpropagate the second layer onto the first:

$$\boldsymbol{\nabla}\mathbf{U_1}\boldsymbol{y}(t,\lambda) = \Big((\mathbf{S_1}\boldsymbol{x} - \mathbf{S_1}\boldsymbol{y}) \overset{t}{*} \boldsymbol{\phi}\Big)(t,\lambda,\alpha,\beta) + \sum_{\alpha,\beta} \Re\left(\left[\frac{(\mathbf{U_1}\boldsymbol{y} \overset{t}{*} \bar{\boldsymbol{\psi}}_\alpha \overset{\log\lambda}{*} \bar{\boldsymbol{\psi}}_\beta)}{|\mathbf{U_1}\boldsymbol{y} \overset{t}{*} \bar{\boldsymbol{\psi}}_\alpha \overset{\log\lambda}{*} \bar{\boldsymbol{\psi}}_\beta|} \times \boldsymbol{\nabla}\mathbf{U_2}\boldsymbol{y}\right] \overset{t}{*} \boldsymbol{\psi}_\alpha \overset{\log\lambda}{*} \boldsymbol{\psi}_\beta\right)(t,\lambda,\alpha,\beta), \quad (9)$$

where the symbol $\Re(z)$ denotes the real part of the complex number z. Lastly, we backpropagate the first layer into the waveform domain:

$$\boldsymbol{\nabla}\mathbf{E}(\boldsymbol{y})(t) = \sum_\lambda \Re\left(\left[\frac{\boldsymbol{y} \overset{t}{*} \boldsymbol{\psi}_\lambda}{|\boldsymbol{y} \overset{t}{*} \boldsymbol{\psi}_\lambda|} \times \boldsymbol{\nabla}\mathbf{U_1}\boldsymbol{y}\right] \overset{t}{*} \boldsymbol{\psi}_\lambda\right)(t) \quad (10)$$

Time–frequency scattering bears a strong resemblance with the set of spectrotemporal summary statistics developed by [24] to model the perception of auditory textures in the central auditory cortex. A qualitative benchmark has shown that time–frequency scattering is advantaged if $\boldsymbol{x}(t)$ contains asymmetric patterns (e.g. chirps), but that the two representations perform comparably otherwise [14]. Nevertheless, time–frequency scattering is considerably faster: the numerical optimizations of wavelet transforms and the recursive structure of backpropagation allows time–frequency scattering (both forward and backward) to be on par with real time on a personal computer, i.e. about 20 times faster than the other implementation. Therefore, an audio snippet of a few seconds can be fully re-synthesized in less than a minute, which makes it relatively convenient for composing sound serendipitously.

3.3. Creation: *FAVN* (2016) and other pluriphonic pieces

FAVN is an electroacoustic piece that evokes issues surrounding late 19th-century psychophysics as well as Debussy's *Prélude à l'après-midi d'un faune* (1894), which itself is a musical adaption of Stéphane Mallarmé's *L'après-midi d'un faune* (1876). To create *FAVN*, we began by composing 47 blocks of sound of duration equal to 21 seconds, and spatialized across three audio channels. These blocks are not directly created with time–frequency scattering; rather, they originate from the tools of the electroacoustic studio, such as oscillators and modulators. After digitizing these blocks, we analyze them and re-synthesize them by means of time–frequency scattering. We follow the algorithm described above: once initialized with an instance of Gaussian noise, the reconstruction is iteratively updated by gradient descent with a bold driver learning rate policy. We stop the algorithm after 50 iterations.

During the concert, the performer begins by playing the first iteration of the first block, and progressively moves forward in the reproduction of the piece, both in terms of compositional time (blocks) and computational time (iterations). The Alte Oper Frankfurt, Frankfurt am Main, Germany premiered *FAVN* on October

5th, 2016. The piece was presented again at *Geometry of Now* in Moscow in February 2017, and became a two-month exhibition at the Kunsthalle in Wien in November 2017, with a dedicated retrospective catalogue [25].

In the liner notes of *FAVN*, librettist Robin Mackay elucidates the crucial role of analysis-synthesis in the encoding of musical timbre:

> The analysis of timbre — a catch-all term referring to those aspects of the *thisness* of a sound that escape rudimentary parameters such a pitch and duration — is an active field of research today, with multiple methods proposed for classification and comparison. In *FAVN*, Hecker effectively reverses these analytical strategies devised for timbral description, using them to synthesize new sonic elements. In the first movement, a scattering transform with wavelets is used to produce an almost featureless ground from which an identifiable signal emerges as the texture is iteratively reprocessed to approximate its timbre. [Wavelets] correspond to nothing that can be heard in isolation, becoming perceptible only when assembled en masse. [26]

We refer to [27] for further discussions on the musical implications of time–frequency scattering, and to [28, 29] on the musical aesthetic of Florian Hecker. Since the creation of *FAVN*, we have used time–frequency scattering to create four new pieces.

Modulator (Scattering Transform) is a remix of Hecker's electronic piece *Modulator* (2012), obtained by retaining the 50^{th} iteration of the gradient descent algorithm. Editions Mego has released this remix in a stereo-cassette format. In addition, we presented an extended version of the piece in a 14-channel format at the exhibition "Florian Hecker - Formulations" at the Museum für Moderne Kunst Frankfurt, Frankfurt am Main, Germany from November 2016 to February 2017. In this multichannel version, 14 loudspeakers in the same room play a different iteration number of the reconstruction algorithm.

Experimental Palimpsest is an eight-channel variation upon *Palimpsest* (2004), Hecker's collaboration with the Japanese artist Yasunao Tone, obtained by the same procedure. This piece was premiered at the Lausanne Underground Film Festival, Lausanne, Switzerland, in October 2016.

Inspection (Maida Vale Project) is a seven-channel piece for synthetic voice and computer-generated sound, performed live at BBC's Maida Vale studios in London and has been broadcast on BBC Radio 3 in December 2016, marking the BBC's first ever live binaural broadcast. An extended version, *Inspection II*, will be released as a CD by Editions Mego, Vienna and Urbanomic, Falmouth, UK in Fall 2019.

3.4. *XAllegroX (scattering.m sequence)*

Lastly, *XAllegroX (scattering.m sequence)* is the remix of a dance music track by Lorenzo Senni, entitled *XAllegroX* and originally released by Warp Record in Senni's *The Shape of Trance to Come* LP (WAP406). Like in Hecker's experimental pieces, we remixed *XAllegroX* by, first, isolating a few one-bar loops from the original audio material, and secondly, by reconstructing them from their scattering coefficients. While, at the first iteration, the loop sounds hazy and static — or, a electronic musicians would call it, *droney* — it regains some of its original rhythmic content in subsequent iterations, thus producing a feeling of sonic rise that is fitting to the typical structuration of dance music. The peculiarity of this *scattering.m sequence* remix is that the musical "rise" is not produced over the well-known sonic attributes of frequency and amplitude (as is usually the case in electronic dance music), but on a relatively novel, joint parameter of texture; that is, a notion of sonic complexity which consists of the organization of frequencies and amplitude through time. Therefore, the development of gradient backpropagation for time–frequency scattering over new avenues for musical creation with digital audio effects: in addition to remixing amplitude (by EQ-ing) and frequency (by phase vocoder), it is now possible to *remix texture itself*, independently of amplitude and frequency. Along the same lines of amplitude modulation (AM) and frequency modulation (FM), we propose to call this new musical transformation a meta-modulation (MM), because it operates over spectrotemporal modulations rather than on the direct acoustic content. Future work will strive to further the understanding of MM, both from mathematical and compositional standpoints.

In July 2018, Warp released *XAllegroX (scattering.m sequence)* as part of a remix 12" named *The Shape of RemiXXXes to Come* (WAP425), hence the title of the present paper. This remix has been pressed on vinyl and made available on all major digital music platforms. The remix was aired on the Camarilha dos Quatro weekly podcast. Mary Anne Hobbs, an English DJ and music journalist, shared another of the album songs on her BBC show "6 Music Recommends". FACT listed the record as one of the must-haves of the month.

4. SCALE-RATE DIGITAL AUDIO EFFECTS

In this section, we introduce an algorithm to manipulate the finest time scales of spectrotemporal modulations (from $10\,\text{ms}$ to $1\,\text{s}$) while preserving both the temporal envelope and spectral envelope at a coarser scale (beyond $1\,\text{s}$). As an example, we implement chirp rate reversal, a new digital audio effect that flips the pitch contour of every note in a melody, without need for tracking partials. This concept will be featured throughout in the pluriphonic sound piece *Syn 21845* (2019), a sequel to Hecker's *Statistique Synthétique aux épaules de cascades* (2019).

4.1. Mid-level time scales in music perception

The invention of digital audio technologies allowed composers to apply so-called *intimate transformations* [30] to music signals, affecting certain time scales of sound perception while preserving others. The most prominent of such transformations is perhaps the phase vocoder [31], which transposes melodies and/or warps them in time independently. By setting T to $50\,\text{ms}$, a wavelet-based phase vocoder disentangles frequencies belonging to the hearing range (above $20\,\text{Hz}$) from modulation rates that are afferent to the perception of musical time (below $20\,\text{Hz}$) [32]. Frequency transposition is then formulated in $\mathbf{S}_1\boldsymbol{x}$ as a translation in $\log_2 \lambda$ whereas time stretching is formulated as a homothety in t.

In its simplest flavor, the phase vocoder suffers from artifacts near transient regions: because all time scales beyond T are warped in the same fashion, slowing down the tempo of a melody comes at the cost of a smeared temporal profile for each note. This well-known issue, which motivated the development of specific methods for transient detection and preservation [33], illustrates the importance of mid-level time scales in music perception, longer than a physical pseudo-period yet shorter than the time span between adjacent onsets [34].

The situation is different in a time–frequency scattering network: the amplitude modulations caused by sound transients are encoded in the scale-rate plane (α, β) of spectrotemporal receptive fields [7]. Therefore, time–frequency scattering appears as a convenient framework to address the preservation of such mid-level time scales in conjunction with a change in rhythmic parameters (meter and tempo); or, conversely, changes in articulation in conjunction with a preservation of the sequentiality in musical events.

4.2. General formulation

We propose to call *scale-rate DAFx* the class of audio transformations whose control parameters are foremostly expressed in the domain $(t, \lambda, \alpha, \beta)$ of time–frequency scattering coefficients, and subsequently backscattered to the time domain by solving an optimization problem of the form

$$\boldsymbol{y}^* = \arg\min_{y} \left\| f(\mathbf{S})\boldsymbol{x} - \mathbf{S}\boldsymbol{y} \right\|_2^2, \tag{11}$$

where the functional $f(\mathbf{S}) = (f_1(\mathbf{S}_1), f_2(\mathbf{S}_2))$ is defined by the composer. Compared to Section 3.1, the loss function in the equation above is not only nonconvex, but also devoid of a trivial global minimizer. Indeed, if the image of the reproducing kernel Hilbert space (RKHS) associated to Equation 3 by the complex modulus operator and low-pass filter $\boldsymbol{\phi}_T(t)$ (Equation 5) does not contain the function $f(\mathbf{S}\boldsymbol{x})$, then there is no constant-Q transform $\mathbf{U}_1^*(t, \log_2 \lambda)$ whose smoothed STRF is $f_2(\mathbf{S}_2)$; and *a fortiori* no waveform $\boldsymbol{y}^*(t)$ such that $\mathbf{S}\boldsymbol{y} = f(\mathbf{S})\boldsymbol{x}$. In order to allow for more flexibility in the set of valid choices of f, we replace the definition of $\mathbf{S}_2\boldsymbol{x}$ in Equation 5 by

$$\mathbf{S}_2\boldsymbol{x}(t, \lambda, \alpha, \beta) = \big(\mathbf{U}_2\boldsymbol{x} \overset{t}{*} \boldsymbol{\phi}_T \overset{\log_2 \lambda}{*} \boldsymbol{\phi}_F\big)(t, \lambda, \alpha, \beta), \tag{12}$$

that is, a blurring over both time and frequency dimensions; and likewise for $\mathbf{S}_1\boldsymbol{x}$. This new definition guarantees that $\mathbf{S}\boldsymbol{x}$ is invariant to frequency transposition up to intervals of size F (expressed in octaves), a property that is often desirable in audio classification [35]. Transposition-sensitive scattering (Equations 4 and 5) are a particular case of transposition-invariant scattering (equation above) at the $F \to 0$ limit, i.e. the Gaussian $\boldsymbol{\phi}_F$ becoming a Dirac delta distribution.

A thorough survey of scale-rate DAFx is beyond the scope of this article; in the sequel, we merely give some preliminary insights regarding their capabilities and limitations as well as a proof of concept. With $Q \gg 12$ wavelets per octave in the constant-Q transform and F of the order of one semitone, scale-rate DAFx would fall within the well-studied application domain of vibrato transformations [36]: a translation of the variable $\log_2 \alpha$ (resp. $\log_2 |\beta|$) would cause a multiplicative change in vibrato rate (resp. depth). Perhaps more interestingly, with $Q \ll 12$ and F of the order of an octave, scale-rate DAFx address the lesser-studied problem of roughness transformations in complex sounds: since the scattering transform captures pairwise interferences between pure tones within an interval of Q^{-1} octaves or less [16], a translation of the variable $(\log_2 \lambda + \log_2 \alpha)$ would transpose the sound while preserving its roughness, whereas a translation of the variable $(\log_2 \lambda - \log_2 \alpha)$ would affect roughness while preserving the spectral centroid. We believe that the capabilities of such transformations, both from computational and musical perspectives, are deserving of further inquiry.

4.3. Example: controlling the axis of time with chirp inversion

Because both Morlet wavelets $\boldsymbol{\psi}_\alpha(t)$ and $\boldsymbol{\psi}_\beta(\log_2 \lambda)$ have a symmetric profile, we have the following identity between Kronecker tensor products:

$$\boldsymbol{\psi}_\alpha \otimes \boldsymbol{\psi}_{-\beta} = \overline{\boldsymbol{\psi}_{-\alpha} \otimes \boldsymbol{\psi}_\beta}. \tag{13}$$

Since the constant-Q transform modulus $\mathbf{U}_1\boldsymbol{x}$ is real-valued, the above implies that $\mathbf{S}_2\boldsymbol{x}(t, \lambda, \alpha, -\beta) = \mathbf{S}_2\boldsymbol{x}(t, \lambda, -\alpha, \beta)$. In other words, flipping the sign of the modulation scale β is equivalent to reversing the time axis in the wavelet $\boldsymbol{\psi}_\alpha$; or, again equivalently, to reversing the time axis in the constant-Q transform $\mathbf{U}_1\boldsymbol{x}$ around the center of symmetry t before analyzing it with $\boldsymbol{\psi}_\alpha$ and $\boldsymbol{\psi}_\beta$. From these observations, we define a chirp inversion functional $f(\mathbf{S}) = (f_1(\mathbf{S}_1), f_2(\mathbf{S}_2))$ where $f_1(\mathbf{S}_1) = \mathbf{S_1}$ and f_2 is parameterized as

$$\begin{aligned} f_2 : \mathbf{S}_2(t, \lambda, \alpha, \beta) \longmapsto \quad & \frac{1 + \boldsymbol{\sigma}(t)}{2} \times \mathbf{S}_2(t, \lambda, \alpha, \beta) \\ & + \frac{1 - \boldsymbol{\sigma}(t)}{2} \times \mathbf{S}_2(t, \lambda, \alpha, -\beta), \end{aligned} \tag{14}$$

with $\boldsymbol{\sigma}(t)$ a slowly varying function at the typical time scale T. Observe that setting $\boldsymbol{\sigma}(t) = 1$ leaves $\mathbf{S}$ unchanged; that $\boldsymbol{\sigma}(t) = -1$ resembles short-time time reversal (STTR) of $\boldsymbol{x}(t)$ with half-overlapping windows of duration T [37]; and that $\boldsymbol{\sigma}(t) = 0$ produces a re-synthesized sound that is stationary, yet not necessarily Gaussian, up to the time scale T. It thus appears that the parameter $\boldsymbol{\sigma}(t)$ in Equation 14 is amenable to an "axis of time" knob that can be varied continuously through time within the range $[-1; 1]$.

As a proof of concept, Figure 3a shows the constant-Q transform of a repetitive sequence of synthetic chirps with varying amplitudes, frequential extents, and orientations; as well as its transformation by the functional f described above, with

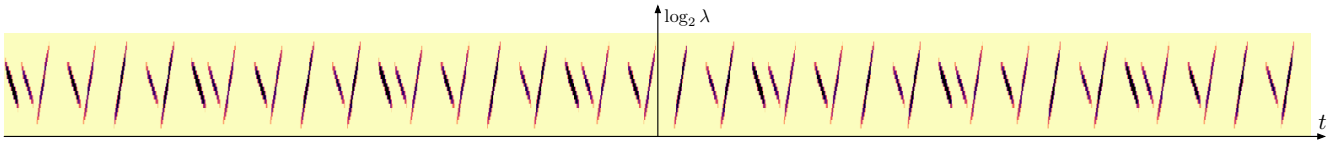
$$\boldsymbol{\sigma}(t) = \frac{1 - \exp\left(\frac{t}{\tau}\right)}{1 + \exp\left(\frac{t}{\tau}\right)} \tag{15}$$

the sigmoid function with a time constant $\tau \gg T$. The frequency transposition invariance F is set to 1 octave. We observe that, while the metrical structure of the original excerpt is recognizable at all times, the pitch contour of every musical event is identical to the original for $t \ll -\tau$ and inverted with respect to the original for $t \gg \tau$. For $|t| < \tau$, there is a progressive metamorphosis between the "forward time" and "backward time" regimes. The effect obtained in Figure 3b, although relatively simple to express in the scale-rate domain, would be difficult to implement in the time–frequency domain.

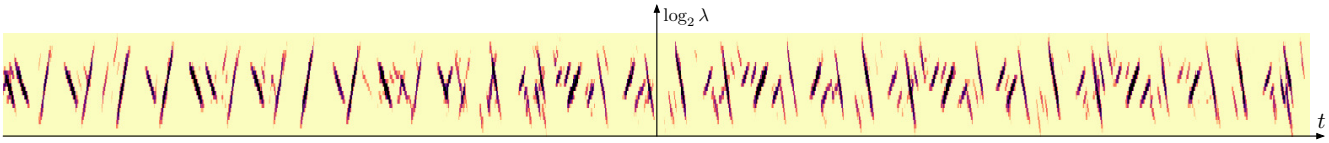
4.4. Towards digital audio effects on the pitch spiral

One evident drawback of scale-rate DAFx is the need to manually adjust the frequency transposition invariance F according to the analysis-synthesis task at hand. Forgoing this calibration step would certainly streamline the creative process. Furthermore, setting F to any value above 1 octave does not only affect spectrotemporal modulations but also the spectral envelope of $\mathbf{U}_1\boldsymbol{x}(t, \lambda)$. In the context of speech transformations, this undesirable phenomenon has led to the development of specific improvements to the phase vocoder [31].

With the aim of addressing both of these shortcomings, we suggest replacing the resort to STRF in Equation 3 $\mathbf{U}_2\boldsymbol{x}(t, \lambda)$ by



(a) An interpretation of Steve Reich's *Clapping Music* (1972) with synthetic chirps of varying rates, scales, and amplitudes.



(b) Re-synthesis after chirp rate inversion. The arrow of time goes forward for $t < 0$ and locally backward for $t > 0$.

Figure 3: An example of chirp rate inversion with time–frequency scattering. Top: original audio material. Bottom: computer-generated output after 100 iterations of gradient descent on time–frequency scattering coefficients. The chirp inversion functional follows a sigmoidal dynamic, as described in Equations 14 and 15. See Section 4 for details.

spiral scattering [38], a convolutional operator cascading wavelet transforms along time, along log-frequencies, and across neighboring octaves. Denoting by $\lfloor \log_2 \lambda \rfloor$ the octave index associated to the frequency λ – that is, the integer part of its binary logarithm – the spiral scattering transform of $\boldsymbol{x}(t)$ writes as

$$\mathbf{U}_2\boldsymbol{x}(t,\lambda,\alpha,\beta,\gamma) = \left|\mathbf{U}_1\boldsymbol{x} \overset{t}{*} \boldsymbol{\psi}_\alpha \overset{\log_2 \lambda}{*} \boldsymbol{\psi}_\beta \overset{\lfloor \log_2 \lambda \rfloor}{*} \boldsymbol{\psi}_\gamma\right|(t,\lambda) \quad (16)$$

where $\boldsymbol{\psi}_\gamma$ is a Morlet wavelet of quality factor $Q = 1$ and center frequency γ, with $|\gamma| < \frac{1}{2}$ measured in cycles per octave. Since spiral scattering disentangles temporal modulations of the non-stationary source-filter model [14], it is conceivable that source modulations and filter modulations could be manipulated independently in the space of spiral scattering coefficients. In particular, the aforementioned effect of "chirp rate reversal" could be generalized to the modulations of the source-filter model. For nonstationary harmonic tones, this would result in a reversal of melodic profile with preservation of the formantic profile, or vice versa. Although the present article does not give a demonstration of such effects, it is worth remarking that their future implementation in the *scattering.m* library would rely on the same principles as the gradient backpropagation of time–frequency scattering coefficients.

The procedure of rolling up the log-frequency axis into a spiral which makes a full turn at every octave, thus aligning power-of-two harmonics onto the same radius, is central to the construction of auditory paradoxes of pitch perception [39] and has recently been applied to musical instrument classification [14] and real-time pitch tuning [40]; yet, to the best of our knowledge, never as a mid-level representation for DAFx. As such, the theoretical framework between scale-rate DAFx and spiral DAFx lies at the interaction between two concurrent approaches in the DAFx community: sinusoidal modeling [33] and texture modeling via neural networks [41]. The former is more physically interpretable requires no training data, yet makes strong assumptions on the detectability of partials in the input spectrum; conversely, the latter is bereft of partial tracking, yet requires a training set and allows for less post hoc manipulations. The long-term goal of scale-rate DAFx is to borrow from both of these successful approaches, and ultimately achieve a satisfying compromise between interpretability and flexbility in texture synthesis.

5. CONCLUSION

The past decade has witnessed a breakthrough of deep convolutional architectures for signal classification, with some noteworthy applications in speech, music, and ecoacoustics. Yet there is, to this day, virtually no adoption of any recent deep learning system by electroacoustic music composers. This is due to several shortcomings of deep learning in its current state, among which: its high computational cost [42]; the need for a large dataset of musical samples, often supplemented with univocal human annotation [5]; the difficulty of synthesizing audio without artifacts [43]; and a certain opacity in the structure of the learned representation [44].

In this article, we have argued that time–frequency scattering – a deep convolutional operator involing little or no learning – is adequate for several use cases of contemporary music creation. We have supported our argumentation by three mathematical properties, which are rarely guaranteed in deep learning: energy conservation (Section 2); well-conditioned adjoint operators in closed form (Section 3); and psychophysiological interpretation in terms of modulation rates and scales (Section 4)..

All of the numerical applications presented here might, after enough effort, be implemented without resorting to time–frequency scattering at all. Yet, one noteworthy trait of time–frequency scattering resides in its versatility: it connects various topics of DAFx that are seemingly disparate, such as coding [4], texture synthesis [20], adaptive transformations [33], and similarity retrieval [45]. The guiding thread between these topics is that – adopting the categories of Iannis Xenakis [46] – time–frequency scattering extracts musical information at the *meso-scale* of musical motifs, allowing to put it in relation with both the *micro-scale* of musical timbre and the *macro-scale* of musical structures [28].

From a compositional perspective, the appeal of time–frequency scattering stems from the possibility to generate sound with a holistic approach, devoid of intermediate procedures for parametric estimation; yet leaving some room to serendipity and surprise in the listening experience. The best evidence of this fruitful trade-off between flexibility and interpretability is found in the breadth of computer music pieces that have resulted from the ongoing interaction between the two authors of this paper. The earliest musical creation that was based on time–frequency scattering (e.g. *FAVN* in 2016) was conceived as an operatic experience with pluriphonic spatialization at the Alte Oper Frankfurt, Frankfurt am Main,

Germany. In contrast, *Modulator (Scattering Transform)* (2017) is a stereo-cassette mix for Editions Mego; *Inspection (Maida Vale Project)* (2017) is a live radio performance at the BBC; and lastly, *XAllegroX (Hecker Scattering.m Sequence)* (2018) is the remix of a dance music track for Warp Records.

More than a fortuitous affinity of personal initiatives, the research-creation agenda that is outlined in the present paper wishes to espouse the noble tradition of *musical research* [47], i.e. a kind of creative process in which the conventional division of labor between scientists and artists is tentatively called into question. Here, for the sake of crafting new music, a signal processing researcher (VL) becomes *de facto* a computer music designer, while a composer (FH) takes on a role akin to a principal investigator [48].

6. ACKNOWLEDGMENTS

This work is supported by the ERC InvariantClass grant 320959 and NSF awards 1633259 and 1633206. The two authors wish to thank Bob Sturm for putting them in contact with each other; Lorenzo Senni for accepting that his record title, *The Shape of RemiXXXes to Come*, is being reused as the title of the present article; and the anonymous reviewers of both DAFX-18 and DAFX-19 for their helpful comments.

7. REFERENCES

[1] H Kaper and S Tipei, "Formalizing the concept of sound," in *Proc. ICMC*, 1999.

[2] J.-C Risset, "Fifty years of digital sound for music," in *Proc. SMC*, 2007.

[3] C.-E Cella, "Machine listening intelligence," in *Proc. Int. Workshop on Deep Learning for Music*, 2017.

[4] G. A Velasco, N Holighaus, M Dörfler, and T Grill, "Constructing an invertible constant-Q transform with non-stationary Gabor frames," in *Proc. DAFx*, 2011.

[5] J Engel, C Resnick, A Roberts, S Dieleman, D Eck, K Simonyan, and M Norouzi, "Neural audio synthesis of musical notes with WaveNet autoencoders," in *Proc. ICML*, 2017.

[6] S Mallat, "Understanding deep convolutional networks," *Phil. Trans. R. Soc. A*, vol. 374, no. 2065, 2016.

[7] J Andén, V Lostanlen, and S Mallat, "Joint time-frequency scattering for audio classification," in *Proc. MLSP*. IEEE, 2015, pp. 1–6.

[8] K Patil, D Pressnitzer, S Shamma, and M Elhilali, "Music in our ears: the biological bases of musical timbre perception," *PLoS computational biology*, vol. 8, no. 11, 2012.

[9] J Andén, V Lostanlen, and S Mallat, "Ieee trans. sig. proc.," *IEEE Transactions on Signal Processing*, vol. 67, no. 14, pp. 3704–3718, July 2019.

[10] T Lindeberg and A Friberg, "Idealized computational models for auditory receptive fields," *PLoS one*, vol. 10, no. 3, 2015.

[11] S Mallat, "Group invariant scattering," *Comm. Pure Appl. Math.*, vol. 65, no. 10, pp. 1331–1398, 2012.

[12] K Siedenburg, I Fujinaga, and S McAdams, "A comparison of approaches to timbre descriptors in music information retrieval and music psychology," *J. New Music Research*, vol. 45, no. 1, pp. 27–41, 2016.

[13] M. R Schädler, B. T Meyer, and B Kollmeier, "Spectro-temporal modulation subspace-spanning filter bank features for robust automatic speech recognition," *J. Acoust. Soc. of Am.*, vol. 131, no. 5, pp. 4134–4151, 2012.

[14] V Lostanlen, *Convolutional operators in the time-frequency domain*, Ph.D. thesis, École normale supérieure, 2017.

[15] S Mallat, *A wavelet tour of signal processing: the sparse way*, Academic press, 2008.

[16] J Andén and S Mallat, "Scattering representation of modulated sounds," in *Proc. DAFx*, 2012.

[17] I Waldspurger, "Exponential decay of scattering coefficients," in *Proc. IEEE SampTA*, 2017.

[18] I Waldspurger, "Phase retrieval for wavelet transforms," *IEEE Trans. Inf. Theory*, vol. 63, no. 5, pp. 2993–3009, 2017.

[19] I Waldspurger, *Wavelet transform modulus: phase retrieval and scattering*, Ph.D. thesis, École normale supérieure, 2015.

[20] D Schwarz, "State of the art in sound texture synthesis," in *Proc. DAFx*, 2011.

[21] D Schwarz, A Röbel, C Yeh, and A Laburthe, "Concatenative sound texture synthesis methods and evaluation," in *Proc. DAFx*, 2016.

[22] I Sutskever, J Martens, G Dahl, and G Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. ICML*, 2013, pp. 1139–1147.

[23] J Bruna and S Mallat, "Audio texture synthesis with scattering moments," *arXiv preprint arXiv:1311.0407*, 2013.

[24] J. H McDermott and E. P Simoncelli, "Sound texture perception via statistics of the auditory periphery: evidence from sound synthesis," *Neuron*, vol. 71, no. 5, pp. 926–940, 2011.

[25] V. J. M Nicolaus Schafhausen, Ed., *Florian Hecker: Halluzination, Perspektive, Synthese*, Sternberg Press, Berlin, 2019.

[26] R Mackay, Program notes to *FAVN*'s premiere. Alte Oper, Frankfurt, October 5th, 2016.

[27] V Lostanlen, "On time-frequency scattering and computer music," in *Florian Hecker: Halluzination, Perspektive, Synthese*, N Schafhausen and V. J Müller, Eds. Sternberg Press, Berlin, 2019.

[28] F Hecker and R Mackay, "Sound out of line: In conversation with Florian Hecker," *Urbanomic*, , no. 3, 2009.

[29] F Hecker, Ed., *Chimerizations*, Primary Information, New York, 2013.

[30] J.-C Risset, "Exploration of timbre by analysis and synthesis," in *The Psychology of Music, 2nd Ed.*, D Deutsch, Ed., chapter 5, pp. 113–169. Elsevier, 1999.

[31] A Röbel, "A shape-invariant phase vocoder for speech transformation," in *Proc. DAFx*, 2010.

[32] R Kronland-Martinet, "The wavelet transform for analysis, synthesis, and processing of speech and music sounds," *Comp. Mus. J.*, vol. 12, no. 4, pp. 11–20, 1988.

[33] A Röbel, "A new approach to transient processing in the phase vocoder," in *Proc. DAFX*, 2003.

[34] P Leveau, E Vincent, G Richard, and L Daudet, "Instrument-specific harmonic atoms for mid-level music representation," *IEEE Trans. Audio Speech Lang. Proc.*, vol. 16, no. 1, pp. 116–128, 2008.

[35] J Andén and S Mallat, "Deep scattering spectrum," *IEEE Trans. Sig. Proc.*, vol. 62, no. 16, pp. 4114–4128, 2014.

[36] A Röbel, S Maller, and J Contreras, "Transforming vibrato extent in monophonic sounds," in *Proc. DAFx 2011*, 2011.

[37] H.-S Kim and J. O. I Smith, "Short-time time-reversal on audio signals," in *Proc. DAFx*, 2014.

[38] V Lostanlen and S Mallat, "Wavelet scattering on the pitch spiral," in *Proc. DAFx*, 2015.

[39] D Deutsch, K Dooley, and T Henthorn, "Pitch circularity from tones comprising full harmonic series," *J. Acoust. Soc. Am.*, vol. 124, no. 1, pp. 589–597, 2008.

[40] T Hélie and C Picasso, "The Snail: a real-time software application to visualize sounds," in *Proc. DAFx*, 2017.

[41] H Caracalla, "Synthèse de textures sonores à partir de statistiques temps-fréquence," M.S. thesis, Ircam, 2016.

[42] M Kim and P Smaragdis, "Bitwise neural networks," in *Proc. ICML*, 2015.

[43] C Donahue, J Macaulay, and M Puckette, "Synthesizing audio with GANs," in *Proc. ICLR, workshop track*, 2018.

[44] G Lample, N Zeghidour, N Usunier, A Bordes, L Denoyer, et al., "Fader networks: Manipulating images by sliding attributes," in *Proc. NIPS*, 2017.

[45] T Pohle, E Pampalk, and G Widmer, "Generating similarity-based playlists using traveling salesman algorithms," in *Proc. DAFx*, 2005.

[46] I Xenakis, "Concerning time," *Perspectives of New Music*, vol. 1, no. 27, pp. 84–92, 1989.

[47] J.-C Risset, "Le compositeur et ses machines : de la recherche musicale," *Esprit*, vol. 3, no. 99, pp. 59–76, 1985.

[48] A Cont and A Gerzso, "The C in IRCAM: Coordinating musical research at IRCAM," in *Proc. ICMC*, 2010.

[49] R Mackay, Ed., *Florian Hecker: Formulations*, Koenig Books, London, 2016.

[50] V Lostanlen, "Scattering.m: a matlab toolbox for wavelet scattering," June 2019.

IMPROVED CARILLON SYNTHESIS

Mark Rau, Orchisama Das, Elliot K. Canfield-Dafilou *

Center for Computer Research in Music and Acoustics,
Stanford University, Stanford, CA 94305 USA
[mrau|orchi|kermit]@ccrma.stanford.edu

ABSTRACT

An improved and expanded method for carillon bell synthesis is proposed. Measurements of a carillon bell and its clapper were made to serve as the basis for an efficient synthesis framework. Mode frequencies, damping, and amplitudes are used to form a modal model fit to measurements. A parameterized clapper interaction model is proposed to drive the bell model, reproducing variation of timbre as the bell is played in different dynamic ranges. Reverberation of the belfry was measured from several listener perspectives and an efficient modal reverberation architecture is shown to model the sound of the bell from locations inside and outside the belfry.

1. INTRODUCTION

Musical acousticians and composers have long held interest in carillons and other bells. Rossing and others seek to understand the physics of how bells vibrate [1, 2]. Others, such as Lehr, work for bell foundries and are interested in improving their casting and tuning techniques [3, 4]. Recently, some researchers have modeled bells using finite element analysis [5]. Romantic composers have often evoked and imitated the sound of bells in their music. Twentieth century works by John Chowning, Jean-Claude Risset, and Jonathan Harvey have all prominently feature synthesized bell timbres. More recent "Hack the Bells" initiatives such as [6] have lead to an increased interest in music for carillon and live electronics.

In contrast to [7], where the authors' goal was to provide a simple model for carillon bell synthesis suitable for processing by composers of electroacoustic music, in this paper we provide a more sophisticated model. Like in previous work, a modal architecture is presented where a bell is modeled as a sum of exponentially decaying sinusoids. One of the issues of [7] was that the authors were limited to a single recording of each bell of the carillon they were modeling. This means they were unable to model the spectral differences between quiet and loud bell hits and were restricted to the perspective of the single microphone.

For this work, we made a comprehensive set of measurements of the bell we model with the idea that the results can be applied for modeling other bells. These measurements include multiple microphone locations, laser Doppler vibrometer (LDV) measurements from the impact position, and accelerometer measurements of the clapper's movement. We additionally made impulse response measurements of the belfry. With these measurements, we perform better when estimating the modal parameters, incorporate a driving function that adequately controls the spectral changes associated with quiet and loud bell strikes, and demonstrate an efficient modal reverberation algorithm allowing the possibility to control and modify the position of the listener.

A carillon is a musical instrument consisting of multiple cup-shaped cast bronze bells. The bells are stationary and struck on the inside by clappers. The bell measured and modeled in this study comes from the Stanford Carillon located in the tower of the Hoover Institution. This carillon consists of thirty-five bells originally cast by Michiels in Tournai, Belgium for the 1939 New York World's Fair [8]. In 2002, The Dutch foundry Royal Eijsbouts recast eleven of these bells and added an additional thirteen bells, bringing the total to forty-eight bells. Additionally, they upgraded several other components of the instrument such as the keyboard and the hanging mechanism. The instrument is equally tempered.

We begin by outlining the measurements of the bell and belfry in §2. Then in §3 we describe the modal synthesis model and how we estimate the various parameters. We discuss results in §4. Finally, §5 offers some concluding remarks.

2. CARILLON BELL MEASUREMENTS

For this study, we measured a large carillon bell tuned to C3, having a fundamental frequency of 129 Hz. An accelerometer was used to measure the keyboard driven clapper interaction while a laser Doppler vibrometer and various microphones were used to measure the resulting bell vibrations. Figure 1 shows the locations of the microphones, LDV, accelerometer, and loudspeaker in relation to the tower, belfry, and bell.

2.1. Clapper Interaction

An accelerometer was placed on the back of the clapper to measure its acceleration as it strikes the bell. It was in line with the clapper's primary direction of motion, orthogonal to the shell of the bell. It was assumed that the arc of the clapper path was negligible and only in one spacial direction. A laser Doppler vibrometer was used to measure the surface velocity at a point on the outside of the bell corresponding to the location of the clapper hit. The measurement locations were chosen to be at an approximation of the driving point, taking physical considerations into account.

2.2. Near Field Radiation

In addition to the contact measurements, several measurement-quality omnidirectional pressure microphones were used to record the sound radiated by the bells at various locations. One microphone was placed in the belfry about 1 m away from the carillon bell to capture the close field radiation. A second microphone was

* All authors should be considered co-first author of this paper.

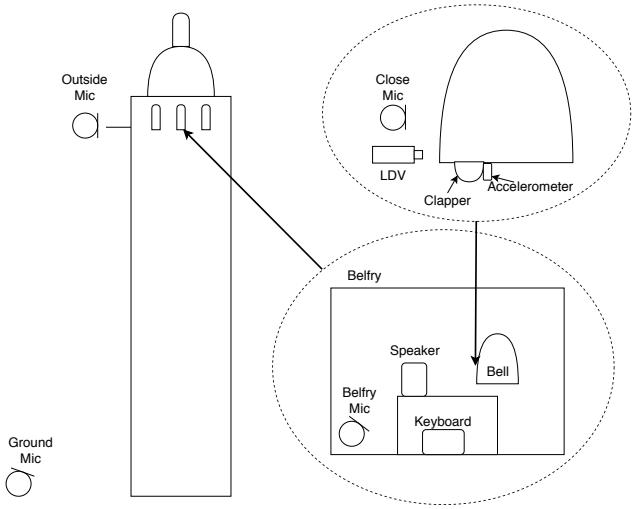


Figure 1: Measurement setup.

placed in the belfry below the bells, in a location similar to where a person may stand in the tower while the carillon is played. A third microphone was suspended on a pole 3 m out of a window in the belfry. This captures the on axis portion of the signal that would pass through the windows of the carillon tower. While this measurement is not from a listener's perspective, it is from an accessible location partway between the belfry and the ground and provides a different prospective of the bell sound.

2.3. Far Field Radiation

In an attempt to measure impulse responses of the bell tower to common listening positions, inexpensive portable recorders were placed at four locations outside of the tower. One recorder was located 25 m from the base of the tower, and the other three were approximately 150, 600, and 1000 m away, at ground level. Two-minute-long exponential sinusoidal sweeps covering the audible frequency range from 20–20000 Hz were played from a loudspeaker in the tower and recorded using all near field and far field microphones. Several of the far field recorders were corrupted by noise caused by other sound sources in the area and the large distance from the tower.

2.4. Bell Coupling

To test if the bells are acoustically coupled, an LDV was used to measure the surface vibration of one bell while other bells were played. Two pairs of bells were tested—one pair was adjacent to each other and the other tonally separated by an octave. In both cases, no significant coupling was measured, so this was not considered further in the model.

3. MODAL CARILLON BELL MODEL

The sound of a bell can be described as an inharmonic series of partials. Like in [7], we use a modal model to represent each bell as a sum of exponentially decaying sinusoids,

$$g(t) = \sum_{m=1}^{M} \alpha_m e^{j\omega_m t} e^{-t/\tau_m}\,, \tag{1}$$

where α_m is the complex amplitude, ω_m the frequency, and τ_m the decay rate for each mode m.

Throughout this section, we will describe our methods for estimating the modal parameters for synthesizing carillon bell sounds. We extend the modal analysis proposed in [7] using simple and robust methods rather than complex high resolution methods such as the one described in [9].

3.1. Estimating Partial Frequencies

First, the clapper impact is deconvolved from the time domain bell measurements. Then, all ten measurements at different loudness levels are averaged in the time domain for further analysis. This improves the signal to noise ration (SNR) as the spectral peaks are common in all measurements. Additionally, nulls in the spectrum of the clapper are dependent on the loudness of the strike while the mode parameters of the bell itself should be independent of loudness. By averaging recordings at several loudness levels, we reduce the bias in the peak picking and amplitude fitting that would over-fit to a single bell loudness.

Similar to [7], the deconvolved bell signal is high-passed filtered half an octave below the hum tone before its Fourier Transform is taken. This reduces the likelihood of picking spurious peaks in the lower frequencies that are simply background noise.

The method for peak picking proposed in [7] is not able to detect close-frequency beating partials (doublets) and it misses a number of high frequency partials which decay very quickly, resulting in inaccurate reconstruction of the bell attack. The following subsections explain the updated method for overcoming these shortcomings.

3.1.1. Resolving doublets

In the previous method, peaks that were very close to each other were discarded. For resolving doublets, this constraint is removed. Even with a larger FFT size, resolving doublets can be tricky. The Hann window used in [7] has a side lobe height of -31.5 dB. This poses the danger of partial side lobes being detected as peaks. To fix this, we use the Hann-Poisson window (2) instead, which is essentially a Hann window multiplied with an exponentially decaying Poisson window [10]. The advantage of this window is that for $\gamma \geq 2$, the side-lobes are smoothed. For high γ, this window has no side-lobes. With the Hann-Poisson window, the peaks detected are guaranteed to be partials and not their side-lobes.

$$w(n) = \frac{1}{2}\left[1 + \cos\left(\frac{2\pi n}{M-1}\right)\right] \exp\left(\frac{-2\gamma|n|}{M-1}\right) \tag{2}$$

3.1.2. Adaptive threshold for peak picking

In the previous method, the threshold for peak picking was kept constant. However, as can be seen from Fig. 2, the spectral envelope shape is not flat, but resembles a low-pass filter. This means that a constant threshold will not be able to detect higher partials, which fall below it. A smarter decision is to pick a threshold that follows the spectral envelope. To do so, first we use a median filter with a filter order of 100 to smooth the spectrum and estimate the spectral envelope. We then fit a straight line to the spectral envelope, and add a constant value to it to get the new threshold. Flattening the spectrum with a "pre-emphasis" filter would have also worked.

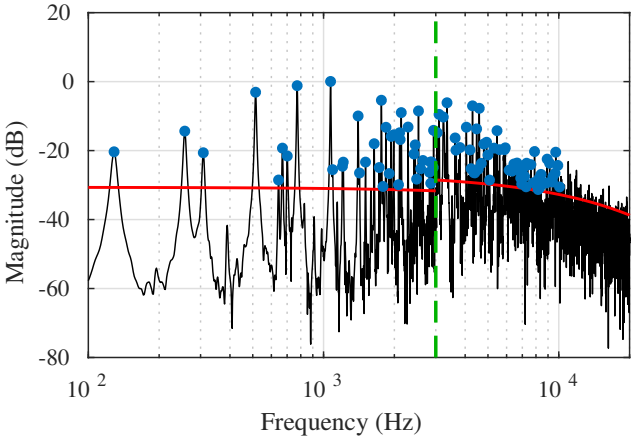


Figure 2: Mode frequency estimates.

3.1.3. Two FFT sizes

We pick two different FFT sizes for low and high frequency partials, with a transition frequency at 3 kHz. A larger FFT size of 2^{15} for detecting lower partials ensures that beating partials are resolved. For higher frequency partials, resolution is not an issue. Lower partials have a small beat frequency, however the beat frequency increases for higher partials, which is why resolving high frequency doublets is possible with a smaller FFT size. Instead, too large an FFT size for higher partials results in poor signal to noise ratio in estimation of decay rates. So, we choose an FFT size of 2^{13} for higher partials. For lower partials, we use a Hann-Possion window with $\gamma_l = 7$ and for higher partials, $\gamma_h = 3$. We can get away with less smoothing of side lobes for higher partials because we ensure that two nearby candidate peaks are not detected as two separate partials. This ensures no side lobes are incorrectly labeled as partial peaks. The peaks picked with two different FFT sizes are shown in Fig. 2. We do not detect any peaks above 10 kHz because the accelerometer is band-limited and there is a high noise floor so we are not confident in the measurements in this region. Instead, we statistically generate higher frequency modes as described in §3.4.

3.2. Estimating Partial Decay Rates

As in [7], we use each frequency found in §3.1 as the center frequency for a fourth-order Butterworth band-pass filter. We find the energy envelope for each partial by averaging the band-pass filtered signals using a 100 ms RMS procedure. However, in [7], the decay rate of each partial was estimated by performing a linear fit to the amplitude envelope using least squares. The region over which the linear fit was performed was selected manually. Since many more partials are detected in our updated method, this procedure is inefficient and time consuming. Instead, we use the method in [11] to automatically estimate the decay rates. This algorithm is based on nonlinear optimization of a model for exponential decay plus stationary noise floor. It works well, even for beating partials that are coupled. For higher modes that decay quickly, we use a weighting function that fits an exponential decay only over the first second of the energy envelope and discards the rest, so that high frequency noise does not perturb the calculation of decay rates. This method was rejected in [7] because the SNR of the single-bell recordings often challenged the algorithm. Here, our SNR is much higher so the algorithm performs better. A disadvantage of this method is that it solves a large optimization problem, and is therefore, quite slow.

3.3. Estimating Partial Amplitudes

Once we have estimated the frequency and decay rate of each mode, we estimate the initial amplitude of each partial required to reconstruct the original bell recording. To do this, we form a matrix where each column holds each partial independently as in

$$\mathbf{M} = \begin{bmatrix} 1 & \dots & 1 \\ e^{(j\omega_1 - 1/\tau_1)} & \dots & e^{(j\omega_M - 1/\tau_M)} \\ \vdots & \dots & \vdots \\ e^{(j\omega_1 - 1/\tau_1)T} & \dots & e^{(j\omega_M - 1/\tau_M)T} \end{bmatrix}, \tag{3}$$

where $\boldsymbol{\omega}_m$ are the frequencies, $\boldsymbol{\tau}_m$ the decay rates, and T is the length of the time vector. We use least squares to find the complex amplitudes

$$\boldsymbol{\alpha} = (\mathbf{M}^\intercal \mathbf{M})^{-1} \mathbf{M}^\intercal \boldsymbol{g}\,, \tag{4}$$

where $\boldsymbol{g}$ is the original bell recording and $\boldsymbol{\alpha}$ is the vector of complex amplitudes.

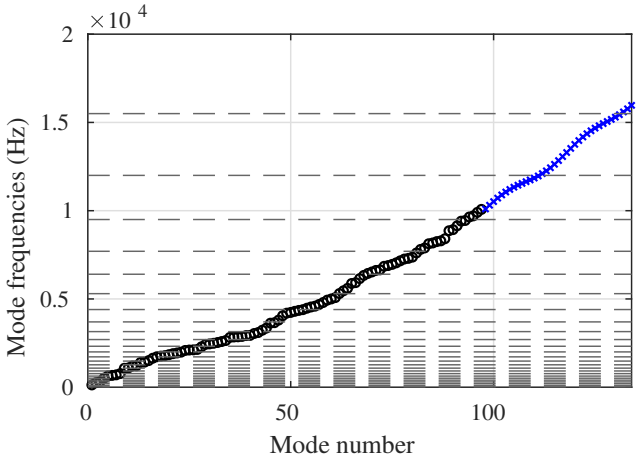
3.4. Statistically Generating High Frequency Modes

While the low frequency modes are louder and decay more slowly than the high frequency modes, the high modes play a role in developing the transient attack sound. It is difficult to measure and estimate modal parameters for the high modes for a variety of reasons. The signal energy at high frequencies is much lower than the low frequencies and the noise floor becomes a significant impediment. Additionally, the measurements from the accelerometer are band-limited, making it impossible to accurately estimate the high frequency modes.

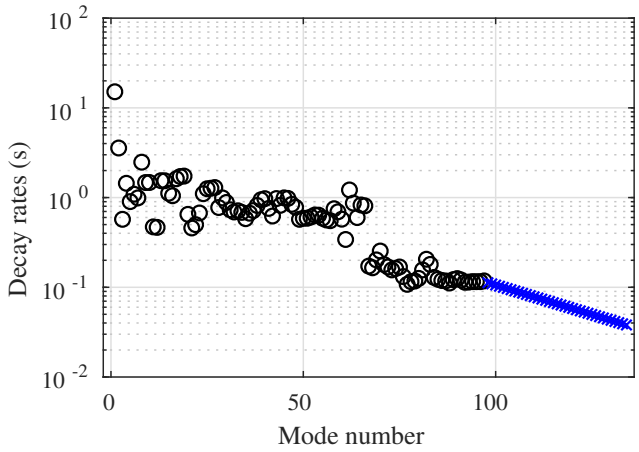
Since the resonant structure of the bell is inharmonic and human hearing has low acuity in high frequencies, it is possible to synthesize artificial high frequency modes. The idea here is to synthesize enough high frequency modes to produce the transient sound without increasing the computational cost too much. To do this, we generate a set of modal frequencies above the frequency we stop fitting modal parameters, i.e. 10 kHz. Figure 3a shows the frequency of estimated modes in black circles, along with the cut-off frequencies of the Bark critical bands of hearing in grey [12]. We can see that mode density increases with critical band number. We fit a quadratic function along with some sinusoidal modulation to the existing mode frequencies and generate new mode frequencies in the range of 10–16 kHz (shown as blue crosses). For the decay rates and amplitudes, we extrapolate from the lower modes. We fit a decaying exponential to the existing decay rates to generate new data points (Fig. 3b). For the amplitudes, we sample from a Gaussian distribution with a mean given by that of the estimated mode amplitudes from §3.3 and a variance of 10^{-5} (Fig. 3c).

3.5. Modeling the Bell-Clapper Interaction

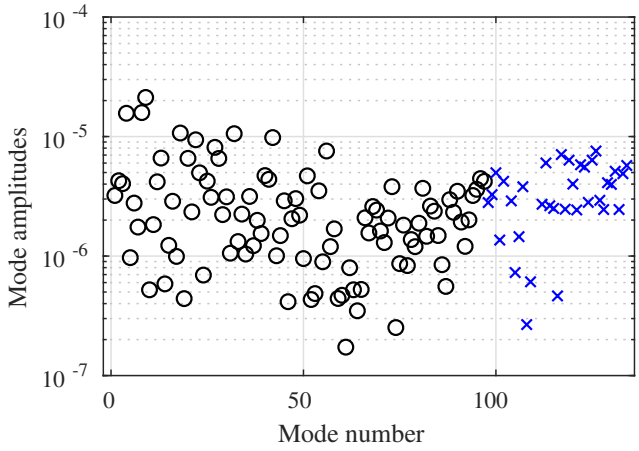
Previous literature suggests that when the carillon is performed at higher dynamic ranges, the high harmonics become more prominent in the timbre. It is suggested that this change in timbre is due to the contact time of the clapper interaction, with a louder hit



(a) Extrapolated mode frequencies (Bark band cutoff frequencies in grey).



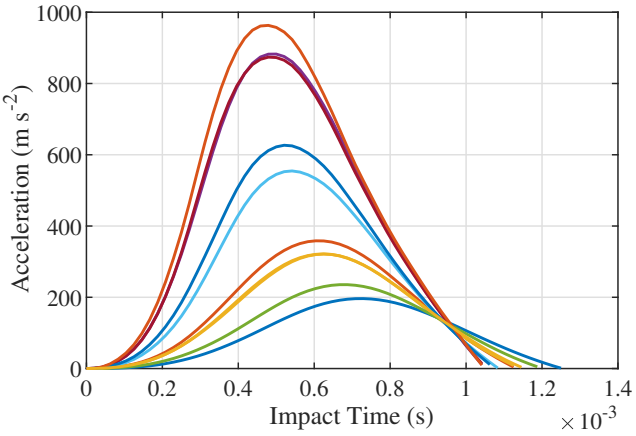
(b) Extrapolated mode decay rates.



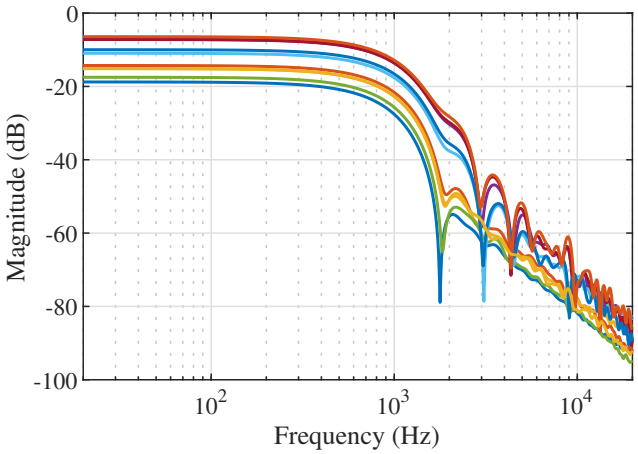
(c) Extrapolated mode amplitudes.

Figure 3: Extrapolated mode parameters (in blue crosses).

having a shorter contact time [13, 14]. To confirm this effect, the clapper acceleration of the initial impact was recorded during nine different dynamic levels. It was assumed that the acceleration of



(a) Time domain.



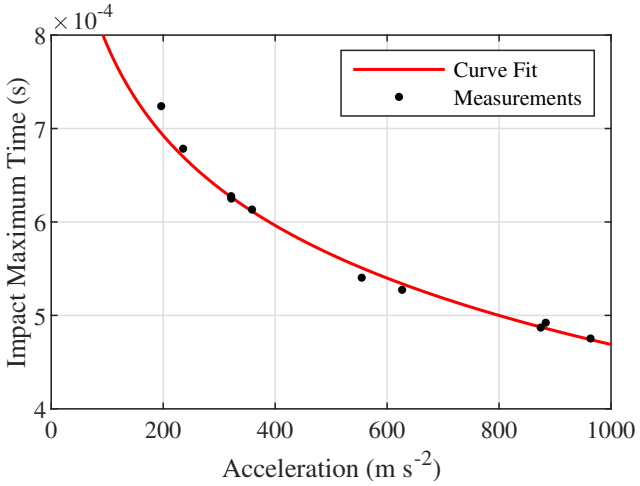
(b) Frequency domain.

Figure 4: Measured impact acceleration of the clapper striking the bell.

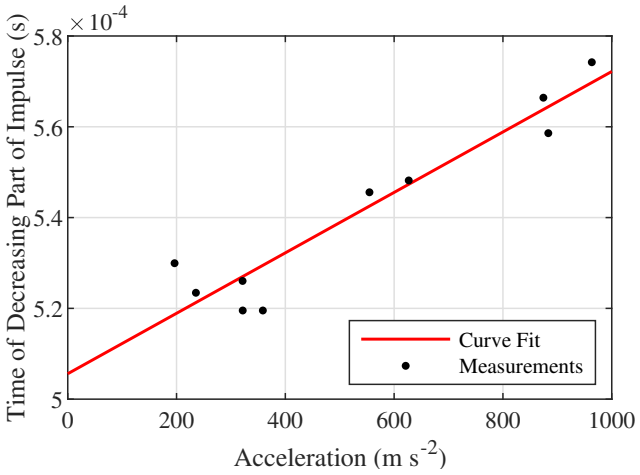
the clapper during the time when it was in contact with the bell is proportional to the force exerted by the clapper, so the acceleration can be used as an input signal to drive the carillon model. The measured acceleration during impact of nine hits as well as the corresponding frequency response of the impulses are shown in Fig. 4.

As confirmed by our measurements, when the carillon is performed at a louder dynamic level, the impact time is shorter and the pulse shape changes in an asymmetrical manner. The frequency response shows that the locations of the nulls change, and the louder hits boost frequencies in the range of 1.5–3 kHz in relation to the quiet hits.

To create a musically usable synthesis model, it is desirable to have as few variable parameters as possible. In this case, it would be ideal to have one parameter for strike amplitude which can drive the carillon model. There have been multiple solutions proposed to model the impact ranging from a half-cycle sine wave proposed by Rayleigh [15], or a Gaussian [5], to numerical solutions [5, 16, 17]. However, the half-cycle sine wave and Gaussian solutions are oversimplified and a numerical solution is not practical for real-time synthesis, so a compromise was made to generate

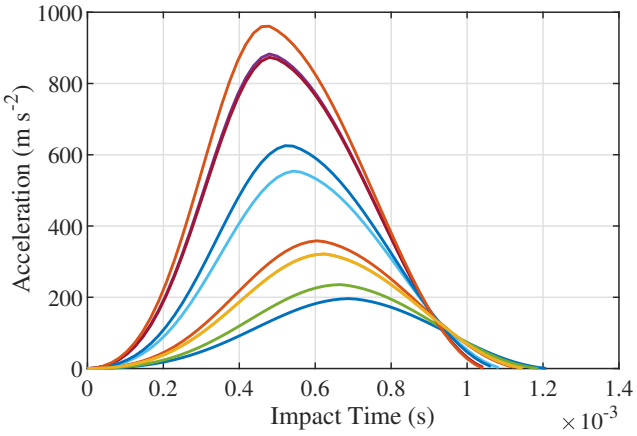


(a) Impact peak time as a function of impact acceleration.

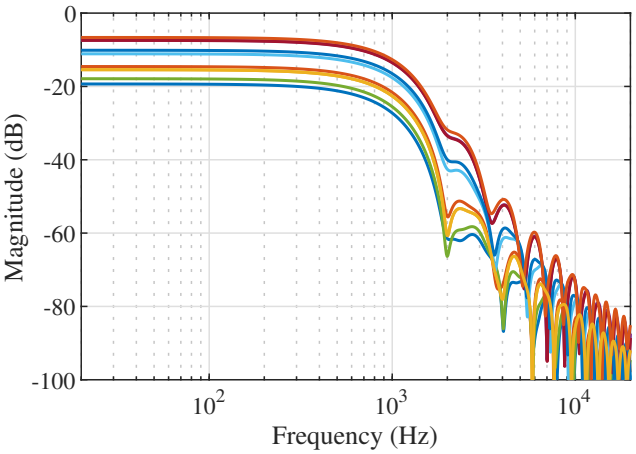


(b) Time of the decreasing portion of the impact as a function of impact acceleration.

Figure 5: Data used to fit curves for parameterized clapper impact model.



(a) Time domain.



(b) Frequency domain.

Figure 6: Synthesized impact acceleration of the clapper striking the bell.

impact signals which are created to fit the data and are parameterized by the impact peak acceleration.

The two defining characteristics of the impacts were chosen to be the time of the acceleration peak and the length of the decreasing portion of the acceleration. Figure 5a shows the time of the acceleration peak plotted against the peak acceleration, while Fig. 5b shows the time taken for the acceleration to decrease to zero after the peak acceleration, against the peak acceleration.

The impact peak time was fit using the logarithmic function:

$$p(a_p) = \frac{-\ln\left(\frac{a_p}{29170(ms^{-2})}\right)}{7193(s^{-1})}, \tag{5}$$

where a_p is the peak amplitude, and ln is the natural logarithm. The time taken for the impact acceleration to decrease was fit using the linear function:

$$d(a_p) = 6.658 \times 10^{-8}(m^{-1}s^3) \times a_p + 5.056 \times 10^{-4}(s), \tag{6}$$

where a_p is the peak amplitude. The curve fits are shown in Fig. 5a and 5b.

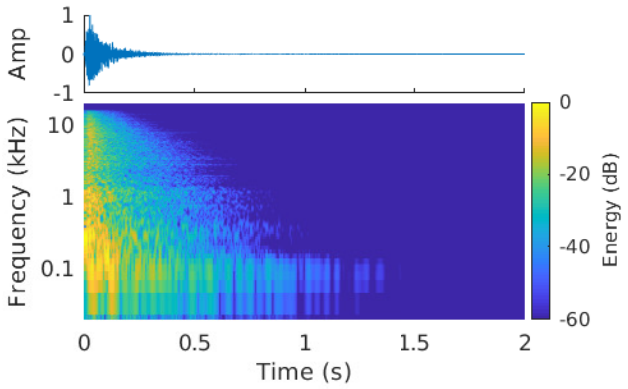
Synthesized clapper impulses were created by combing the left and right portions of two window functions. The left portion of the impulse is the left half of a Blackman window,

$$w_l(n) = 0.42 - 0.5\cos\left(\frac{2\pi n}{N}\right) + 0.08\cos\left(\frac{4\pi n}{N}\right), \tag{7}$$

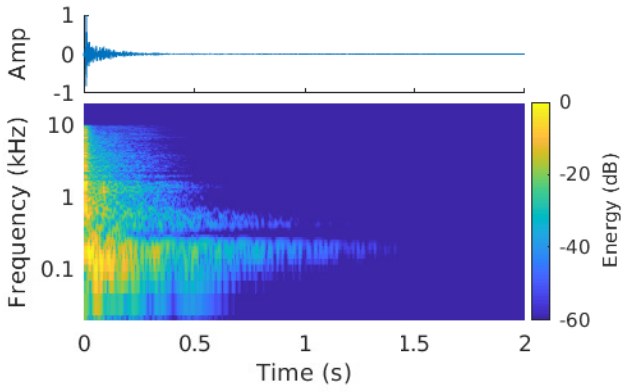
for $0 \leq n \leq N-1$, where $N = 2f_s \times p(a_p)$, and f_s is the sample rate. The right portion of the impulse is the right half of a Bartlett-Hann window,

$$w_r(m) = 0.62 - 0.48\left|\frac{m}{M} - 0.5\right| - 0.38\cos\left(\frac{2\pi m}{M}\right), \tag{8}$$

for $0 \leq m \leq M-1$, where $M = 2f_s \times d(a_p)$ [10]. These windows are parameterized by the peak amplitude of the desired hit (a_p), and created to be of length $2p(a_p)$ and $2d(a_p)$ seconds respectively for the Blackman an Bartlett-Hann windows. Figure 5 shows time and frequency domain plots of synthesized impact signals having the same peak acceleration as the measured impacts shown in Fig. 4.



(a) Belfry impulse response.



(b) Ground impulse response.

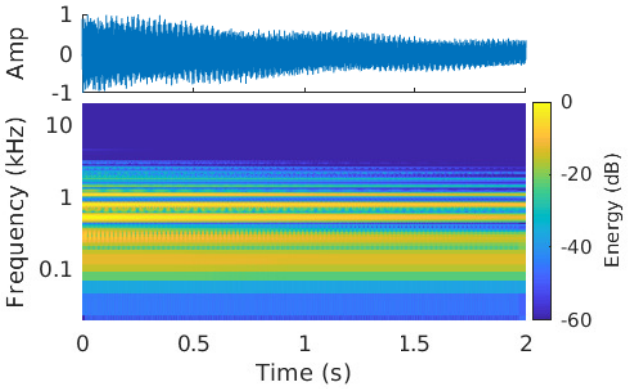
Figure 7: Impulse responses taken from Hoover tower.

3.6. Efficient Belfry Reverberation

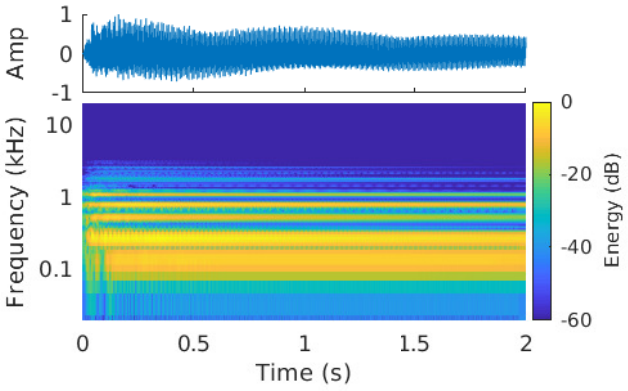
The belfry where the bells are housed can have a large impact on the sound of a carillon in addition to affecting how the sound radiates from the tower. Figure 7 shows impulse responses of the reverberation of Hoover tower measured 3 m outside the belfry and from the bottom of the tower. The response from the bottom of the tower contains much less high-frequency energy and has a prominent slapback echo from a nearby building. Needless to say, the bells sound very different from inside the tower.

The sound of a carillon bell ringing in a belfry can be represented as the convolution of the bell with the reverberation of the belfry. Since we are using a modal bell model, we can also implement the reverberation with a modal architecture [18]. The computation time is independent of the fact that this architecture is a combination of series and parallel components. For each time step, the number of operations scales linearly with the total number of modes. That being said, we can implement an even more efficient reverberation algorithm by taking advantage of the modal architecture.

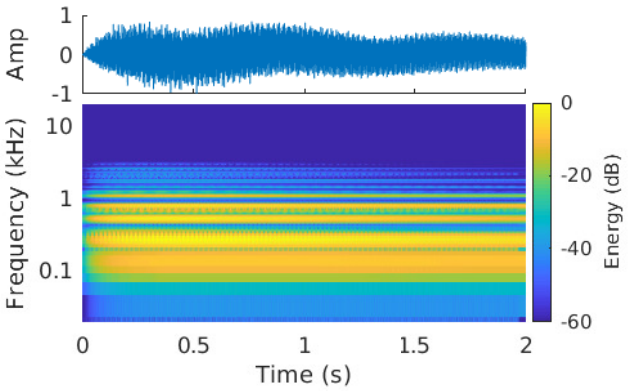
Most of the modes of the carillon bell are orthogonal to most of the modes of the reverb. If we assume that the majority of the energy of each bell mode only drives the reverb modes that are close in frequency, we find an efficient way to implement the reverbera-



(a) Synthesized bell.



(b) Synthesized bell with belfry convolution reverb.



(c) Synthesized bell with efficient modal reverb.

Figure 8: Comparison of a synthesized bell without reverb (8a), with convolution reverb (8b), and with an efficient modified modal reverb (8c).

tion. Instead of processing each mode of the bell through the full complement of reverberation modes, we can pass each bell mode through the reverberation mode nearest in frequency. Furthermore, we fold the reverberation model into the bell model by replacing the decaying exponential amplitude envelopes of the bell modes and reverberation modes by the convolution of the envelopes. Now (1) can be replaced by

$$g(t) = \sum_{m=1}^{M} \alpha_m \beta_m e^{j\omega_m t} \frac{\tau_m \zeta_m \left(e^{-t/\tau_m} - e^{-t/\zeta_m} \right)}{\tau_m - \zeta_m}, \quad (9)$$

where β_m is the amplitude and ζ_m the decay rate of the belfry reverberation nearest the m^{th} mode where the frequency term is shared for the bell and reverb. Figure 8 shows a synthesized bell without reverb and the same bell resynthesized with the belfry reverberation implemented with convolution reverberation and with the efficient scheme shown in (9).

4. RESULTS AND DISCUSSION

Sound examples of the original measurements and resynthesized bells with and without reverb can be found at `https://ccrma.stanford.edu/~kermit/website/morebells.html`. The modal bell driven with synthesized impacts to emulate different loudness levels is shown in Fig. 9. As the impact force on the bell increases, higher modes become more perceptually prominent and ring longer. There is also more energy imparted into all modes, as is intuitive and clear from the figure. A comparison of a measured and modeled bell strike can be seen in Fig. 10. Note that the resynthesized bell has significantly less noise than the recording but they otherwise sound similar.

The clapper interaction model is simple and controlled by one parameter, providing a reasonable approximation of the spectral variation one would expect when playing a carillon at varying dynamic ranges. However, the model is an approximation based on measurements and not derived analytically from the physics governing contact dynamics of the interaction. This leads to the high frequency nulls not appearing at the exact location of the measured impulses. A more accurate model based on the physics would be a large improvement, but it must be easily parameterizable and require low computation to be useful in a performance context.

The efficient reverberation algorithm presented in §3.6 does not sound exactly the same as the convolution reverb. This is due to the fact that some energy spreads to modes at other frequencies. One solution to this issue would be to compute a small subset of the reverberation modes for each bell mode. At the expense of a little more computation, this will more accurately implement the reverberation. Another possibility would be to implement a hybrid reverberation algorithm where the efficient modal implementation is used in combination with a short FIR filter containing the a windowed version of the few milliseconds of the belfry reverberation with the shared mode frequencies removed. This would implement the mode coupling while remaining computationally efficient.

5. CONCLUSION AND FUTURE WORK

In this paper, we have proposed an accurate modal model of a carillon bell capable of being driven by different impact functions which have been derived from physical measurements of the clapper striking the bell. Artificial high frequency modes have been

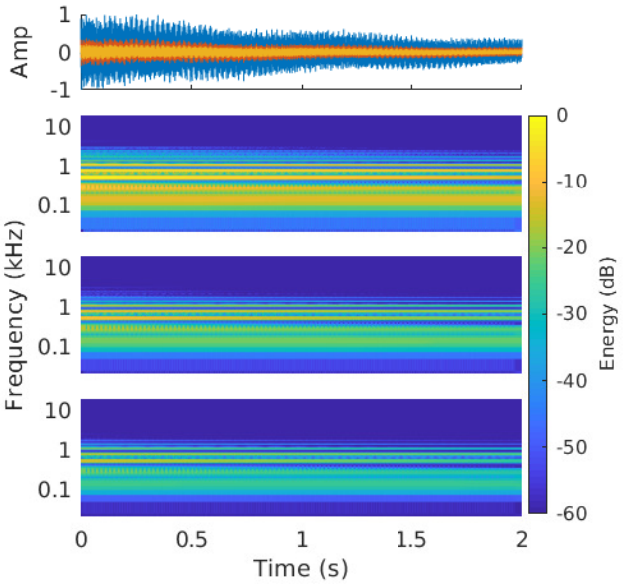


Figure 9: Synthesized bell at different dynamic levels: ***ff*** (top), ***mf*** (middle), and ***p*** (bottom).

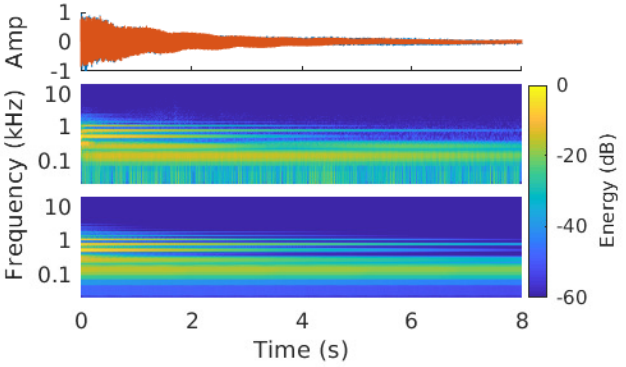


Figure 10: Measured bell (top) and modeled bell (bottom).

generated using data extrapolation for more accurate reconstruction of the bell transient. We also measured the belfry impulse responses and proposed an efficient implementation of the belfry reverberation with a modal architecture. This model is more accurate than previous attempts, can be run real-time, and takes musical dynamics into consideration. We hope that this added flexibility in modeling bells will aid composers to conceive pieces with novel and interesting uses of the carillon bell aided with live electronics.

A method for modeling bells has been presented, but has only been tested on one bell. Future work may include further validation on multiple bells. Of particular interest is testing how the clapper interaction model will translate to smaller bells. We attempted to measure a smaller bell as well, but the accelerometer clipped due the higher acceleration of the less massive clapper. We hope

to continue working on carillon models to increase their accuracy and expand the flexibility for composers.

6. ACKNOWLEDGMENTS

The authors thank Stanford University Carillonneur Timothy Zerlang and the Hoover Institution for providing access to the carillon and Hoover Tower.

7. REFERENCES

[1] Thomas Rossing and Robert Perrin, "Vibrations of bells," *Applied Acoustics*, vol. 20, pp. 41–70, 1987.

[2] Thomas Rossing, Ed., *The Acoustics of Bells*, Van Nostrand Reinhold, 1984.

[3] Andre Lehr, "Designing chimes and carillons in history," *Acta Acustica united with Acustica*, vol. 83, no. 2, pp. 320–36, 1997.

[4] Andre Lehr, "The removal of warbles or beats in the sound of a bell," *Acta Acustica united with Acustica*, vol. 86, no. 3, pp. 550–6, 2000.

[5] Bjrn-Andre Lau, Peter Wriggers, Albrecht Schneider, and Rolf Bader, "Finite-element transient calculation of a bell struck by a clapper," in *Concepts, Experiments, and Fieldwork: Studies in Systematic Musicology and Ethnomusicology*, pp. 137–56. Peter Lang, 2010.

[6] "University of Michigan hack the bells," `https://gobluebells.wordpress.com/2016/09/21/hackthebells/`, 2016.

[7] Elliot Kermit Canfield-Dafilou and Kurt James Werner, "Modal audio effects: A carillon case study," in *Proceedings of the International Conference on Digital Audio Effects*, 2017.

[8] Elena S. Danielson, *For peace alone do I ring: The history of the Hoover Tower carillon and its restoration*, Hoover Institution, 2002.

[9] Corey Kereliuk, Woody Herman, Russel Wedelich, and Daniel J. Gillespie, "Modal analysis of room impulse responses using subband esprit," in *Proceedings of the International Conference on Digital Audio Effects*, 2018.

[10] Julius O. Smith, *Spectral Audio Signal Processing*, W3K publishing, 2011.

[11] Poju Antsalo, Aki Makivirta, Vesa Valimaki, Timo Peltonen, and Matti Karjalainen, "Estimation of modal decay parameters from noisy response measurements," in *Proceedings of the 110th Audio Engineering Society Convention*, 2001.

[12] Eberhard Zwicker, "Subdivision of the audible frequency range into critical bands (frequenzgruppen)," *The Journal of the Acoustical Society of America*, vol. 33, no. 2, pp. 248, 1961.

[13] Neville H. Fletcher, William T. McGee, and Alex Z. Tarnopolsky, "Bell clapper impact dynamics and the voicing of a carillon," *The Journal of the Acoustical Society of America*, vol. 111, no. 3, pp. 1437–44, 2002.

[14] Jim Woodhouse, James C. Rene, Catherine S. Hall, Luke T. W. Smith, Frank H. King, and John W. McClenahan, "The dynamics of a ringing church bell," *Advances in Acoustics and Vibration*, vol. 2012, 2012.

[15] John William Strutt and Baron Rayleigh, *The theory of sound*, Dover, 1945.

[16] Piotr Brzeski, Tomasz Kapitaniak, and Przemysław Perlikowski, "Experimental verification of a hybrid dynamical model of the church bell," *International Journal of Impact Engineering*, vol. 80, pp. 177–84, 2015.

[17] Jernej Klemenc, Andreas Rupp, and Matija Fajdiga, "Dynamics of a clapper-to-bell impact," *International Journal of Impact Engineering*, vol. 44, pp. 29–39, 2012.

[18] Jonathan S. Abel, Sean Coffin, and Kyle Spratt, "A modal architecture for artificial reverberation with application to room acoustics modeling," in *Proceedings of the 137th Convention of the Audio Engineering Society*, 2014.

REAL-TIME MODAL SYNTHESIS OF CRASH CYMBALS WITH NONLINEAR APPROXIMATIONS, USING A GPU

Travis Skare

CCRMA
Stanford University
Stanford, CA
travissk@ccrma.stanford.edu

Jonathan Abel

CCRMA
Stanford University
Stanford, CA
abel@ccrma.stanford.edu

ABSTRACT

We apply modal synthesis to create a virtual collection of crash cymbals. Synthesizing each cymbal may require enough modes to stress a modern CPU, so a full drum set would certainly not be tractable in real-time. To work around this, we create a GPU-accelerated modal filterbank, with each individual set piece allocated over two thousand modes. This takes only a fraction of available GPU floating-point throughput.

With CPU resources freed up, we explore methods to model the different instrument response in the linear/harmonic and nonlinear/inharmonic regions that occur as more energy is present in a cymbal: a simple approach, yet one that preserves the parallelism of the problem, uses multisampling, and a more physically-based approach approximates modal coupling.

1. INTRODUCTION

Modal synthesis is an effective way of capturing a variety of physical sounds. Its parameters are intuitive and have a space-efficient representation. Parameters may be computed by solving the equations governing the system, analytically from a series of system measurements or instrument recordings, or by combining or morphing between other instruments' coefficients.

Increasing the number of modes N of our synthesizer is important to broaden the types of sounds we may represent. A cowbell may be synthesized with a few dozen modes, but qualitatively, a cymbal wash benefits from hundreds or even thousands of modes. Compare the whole-sound spectra of a cowbell against that of a cymbal in Figure 1. To see behavior over time, the spectrogram of a Sabian 16" HHX Evolution Crash, a relatively "dark" or "complex" crash, is shown in Figure 2. We note evidence of highly nonlinear effects, such as higher frequencies emerging in the 100ms-300ms range. These cannot be simulated directly with our basic linear modal synthesis method, but we will explore methods that attempt to obtain such sounds by extending our linear synthesis system with approximations.

In addition to having a complex sound with many modes, thin shells exhibit highly nonlinear behavior, with response often divided into three regimes: linear, inharmonic, and chaotic. This makes creating accurate physical models difficult[1, 2, 3], though the sounds and control space are rich.

Ducceschi et al.[4] solve the von Kármán equations that govern the underlying nonlinear plate physics of plates and cymbals in terms of the system's modes. They obtain values for coupling coefficients offline, yet efficiently. In [5] the authors extend this work and apply a simpler, though less inherently stable, Störmer-Verlet method for time integration to obtain synthesis results running fairly quickly–only 8x slower than real-time on a CPU. Nguyen et al. [6] use a similar time stepping method to cymbal synthesis, paying particular attention to specific cymbal geometry variations that is relatively unique in the literature–cymbals are not plates of uniform thickness, but vary from bell to inner bow to edge, and the authors demonstrate this is an important property to model when considering cymbals over gongs.

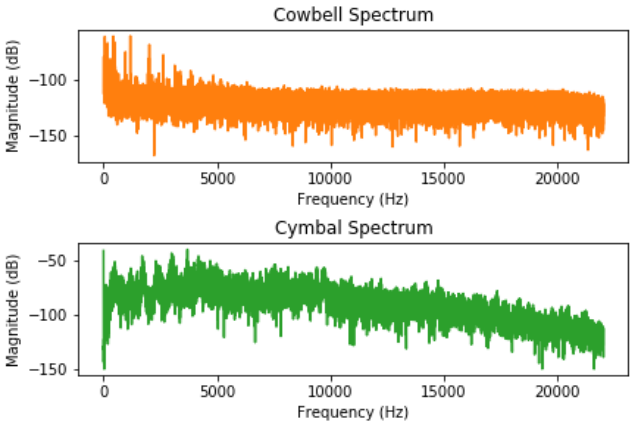


Figure 1: *Top: Cowbell spectrum. Bottom: Sabian HHX Crash Cymbal Spectrum. Note the latter has a significantly higher density of modes, while the former can be better represented by a smaller number of local maxima.*

Our end goal is to run simulation of several cymbals in realtime with enough processing power left over for the rest of a drum set and the other instruments. GPU acceleration is attractive for this application–using the *single instruction, multiple thread* model of execution, modern graphics processors excel at executing one piece of code simultaneously across tens, hundreds, or even thousands of threads, each performing the same operations on different data. This is a more parallel version of vector instructions, and while it is not applicable to every application (handling one input through a series of connected effects plugins, for example), there are some audio applications where it excels. In [4] for example, Ducceschi et al. reduced computation time from 90 minutes per second of audio on a desktop CPU down to 55 seconds per sound. And we again emphasize their work is physically accurate while our work here will aim for simpler approximations for

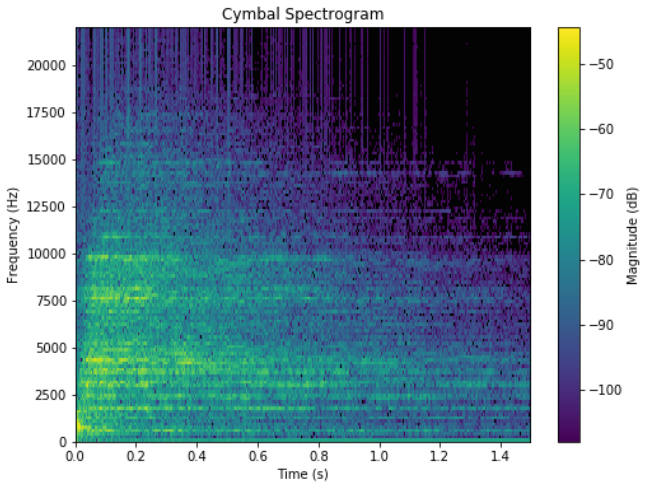


Figure 2: *Sabian HHX Evolution Crash Spectrogram.*

cymbal-like sounds.

The literature has several projects involving GPU-Accelerated additive synthesis or massive parallel filtering:

Savioja et al.[7] give an overview of potential audio tasks that may be accelerated via GPGPU programming at audio rate and reasonable buffer sizes for real-time performance. For example, FFTs running on a GPU were able to be eight times as long as their CPU-implementation counterparts. GPU-accelerated FIR filters were able to be 130x as long as the baseline versions. In [8], the authors synthesized 1.9 million sinusoids in real-time, a 1300x speedup over a serial lookup table computation on one CPU. This was on a GPU that is six generations behind ours and three major GeForce architecture revisions behind the card we are using, and we note that our card is itself a generation behind state of the art.

Trebien et al.[9] use modal synthesis to synthesize physically-accurate collision sounds between objects of different materials – balls rolling down a ramp, for example. They introduce a transform of the system's IIR filters to be able to compute several samples in parallel, turning the operation into linear convolutions that are well-suited for a GPU. More recently, Cirio et al.[10] tackle this problem with specific attention on cymbals and gongs, and in particular attempt to model the chaotic and wave-turbulent effects. This work leverages parallelism between frequency bands as one strategy to parallelize, and as they simulate nonlinear effects in a more physically-accurate fashion than we will, they obtain rich sounds with only 100 high-frequency modes. Simulation costs for one cymbal are still 43x slower than real-time, but that is a 70x speedup over their target algorithm, making the approach tractable for generation audio to graphics offline. Chadwick et al.[11] also apply modal strategies, including simulated coupling, plus introduce far-field acoustic transforms, for such "virtual foley" work. Collisions of thin shells including water bottles and crash cymbals sound realistic. Simulation and precomputation are expensive, but the runtime acoustic transfer map step runs relatively faster, at 16x slower than real-time.

Belloch et al. [12] apply a transform on IIR filters to make them more suitable for the GPU. At 44.1kHz they run over 1,200 256th-order IIR filters simultaneously, with latency of less than a millisecond. Subsequently, Belloch et. al apply massively parallel filtering[13] to Wave Field synthesis on a 96-speaker array[14], running nearly ten thousand fractional-delay room filters with thousands of taps. Several dozen sources were able to be placed into the field.

Bilbao and Webb[15] describe a GPU-accelerated model of a set of timpani, simulating both the drums as well as the space outside. They obtain speedups of 8x to 30x over CPU implementations, and the drum update runs in near-real-time (2 milliseconds per sample), with the bottleneck being a linear drum membrane update.

With advances in machine learning and deep learning fields, some groups are using GPUs to train models that output audio during inference, an interesting approach different from physical modeling. The convolutional WaveRNN by Kalchbrenner, Elsen et al.[16] can run inference (generation) even on mobile CPUs.

The physical modeling applications mentioned above often exploit parallelism across time – different GPU threads are simultaneously working on $x(t_n)$ and $x(t_{n+k})$ for small k. This is often the result of a clever transform, for example moving IIR filters to parallel form. In our application, we'd like to retain the option to modulate system parameters based on a signal we are fed sample-by-sample, and unfortunately cannot apply such transforms. In [17] we demonstrated our modal synthesis filters can be run at audio rates on modern GPUs in a serial fashion–that is, GPU core speed and FPU throughput have advanced sufficiently in recent years such that developers do not need to parallelize across time for certain audio applications (of course, if you can, it still unlocks a great deal of parallelism). Floating-point throughput on modern high-end consumer hardware is enough to run over a million such filters.

Since that work on building blocks and benchmarks, a real-time GPU filterbank was developed, which we present here and then leverage toward cymbal synthesis.

2. BUILDING BLOCKS: VERY HIGH-Q PHASOR FILTERS

We use a modal filterbank for synthesis. This consists of N resonant filters. We make the assumption that all the filters are uniform in construction and vary in parameters, though a GPU could run multiple styles of filters in parallel, either through conditional execution or simultaneous kernel execution.

Modal synthesis may be performed using the developer's favorite resonant filter, or via adding sinusoids. In practice, rapidly changing the coefficients on e.g. Direct-Form II filters may result in audible artifacts. Adding sinusoids is efficient, especially if we're simply performing lookups into a table that's in processor cache, but we have a preference for another approach if computational resources are available.

In [18] Max Mathews and Julius Smith proposed a filter that is very-high-Q, numerically stable, and artifact-free, based on properties of complex multiplication. The parameters map one-to-one with the physical properties of our system which makes it a great choice for modal synthesis and modal reverberators such as in [19]. The recursive update equation we need to implement is:

$$y_m(t) = \gamma_m x(t) + e^{(j\omega_m - \alpha_m)} y_m(t-1) \quad (1)$$

where:
$x()$ is an input or excitation signal.
ω_m is mode m's frequency.
γ_m is a per-mode complex input amplitude gain.

α_m is a per-mode damping factor.

In terms of implementation details, the state we store for each mode is limited to the prior output $y_m(t-1)$, plus having the input parameters available (α_m, γ_m, and ω_m). This fact will be relevant when we walk through the GPU code.

This filter has some nice properties, such as that it preserves phase across restrikes if parameters are updated on zero-crossings. We next present how we may run hundreds of thousands of them in parallel.

3. GPU FILTERBANK IMPLEMENTATION

We present a system that has three major pieces across two processes. A system diagram is in Figure 3.

- The **CPU Client** is a JUCE C++ plugin that accepts MIDI events, translates them into modes, and populates data structures to facilitate communication to the GPU (modal parameters) and from the GPU (audio data to copy to a DAW audio callback buffer). For a set of cymbals, we may copy in a set of modes that are mapped to a particular MIDI note representing that instrument selection. For a tonal instrument application, modes would be scaled to start on the correct fundamental frequency.
- The **GPU Process** contains code to allocate memory on the graphics card and on the host PC. It copies relevant data to the GPU and manages kernel execution.
- The **GPU kernel** is the code that runs on the GPU and represents the core of the synthesis.

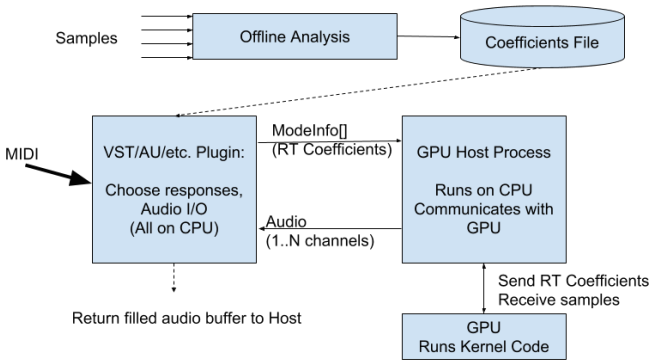


Figure 3: *Overall System Diagram. Top: offline components analyze modes from samples. Bottom: real-time system synthesizes using the proposed parallel architecture.*

For the sake of discussion, we aim to keep the GPU kernel code as simple, short, and approachable as possible. Specifically, it will take as input our modal filter parameters $\omega_N, \gamma_N, \alpha_N$ and input signal $x()$, and simply synthesize the resulting audio output. Later versions may perform parameter interpolation, modal coupling, etc. or post-process audio.

Keeping the GPU process as simple as possible allows much of the complexity to be moved to an application that calls the GPU code, enables easy extension to multiple client applications, enables rapid development, and allows for code reuse across projects as the GPU process will happily run in the background indefinitely while only taking a few MB of RAM.

Communication between host and GPU processes was accomplished using shared memory, and signaling between processes was done using semaphores. The shared memory is simply a chunk of memory mapped into each of the two CPU processes. It is a series of `ModeInfo` structures as defined in Listing 1 followed by a chunk of memory to allow transmitting audio data back to the JUCE plugin so it can in turn be sent back to the DAW or the next plugin in the chain for more processing. This was chosen for simplicity and speed; the code should compile without any external libraries beyond CUDA. Other projects may wish to replace the cross-process communication with, for example, an OSC client and run the host application on a tablet.

```
struct ModeInfo {
  bool enabled;  // is this filter on?
  bool reset;    // should we zero state?

  // filter parameters
  float amp_real;  // Re{alpha}
  float amp_imag;  // Im{alpha}
  float damp;      // gamma
  float freq;      // omega

  // currently-unused optimizations
  bool amp_changed;  // did alpha change?
  bool freq_changed; // did omega change?
};
```

Listing 1: ModeInfo structure used in shared memory block.

Most development and demos were run at 256 samples at 44.1kHz (5.6ms); a buffer of 128 samples was tested successfully.

Our graphics card is an NVIDIA GeForce 1080Ti; it should run on some older cards where core frequency and floating-point throughput are sufficient. It is also expected to run with even better performance on the newer RTX series cards. We have not attempted a port to OpenCL to run on AMD cards, but with a high enough core speed and floating-point throughput it should be straightforward. For those interested in CUDA GPU programming, a variety of books are available, and the NVIDIA developer documentation[20] likely contains all that is required to get started.

3.1. CPU vs GPU limits

In synthetic benchmark settings in [17], where the entire system is concentrated on running these filters, we found a GPU could run over 3.5 million phasor filters at audio rate, or 1.8 million allowing for continuous modulation of filter parameters (requiring additional multiplications). Porting this test to run on an Intel i7 4770k CPU core showed the CPU could run just over 3,000 phasor filters in real-time on one thread, allowing for per-sample parameter modulation. A synthetic benchmark constructed in FAUST using alternate building blocks–second-order resonant filters–could run 4,600 filters on one CPU thread.

In terms of real-world use cases, a "cymbal-verb" modal reverberator plugin was developed with such a phasor filterbank at its core; when running inside inside a DAW, at 128 samples, pressured by audio callbacks, with a few other audio tracks plus one instance of the modal filter effect plugin, overruns happened between 1600 and 1900 modes, with significant variance. A 2012 MacBook Pro, with a laptop Intel i7 3820QM, suffered dropouts when synthesizing 1000-1100 modes. The GPU process running on the desktop alongside the DAW was able to successfully synthesize 100,000 modes, above which we are blocked on a need to optimize the communication between the processes. We note in

this case we are utilizing the GPU, while the DAW does not leverage that resource. Our GPU resource closest to bottlenecking in this trial is the 32-bit floating-point units, which debugging tools estimated at 5.5% utilization.

We note our benchmark parts in this system are slightly mismatched: the CPU is a mid-to-high-end 2013 consumer part while the GPU is the flagship model from its series from 2017, and cost over twice as much. We also note that the majority of audio plugins will have a natural affinity for running on a CPU, but highly parallel tasks may take advantage from a GPU especially if it would otherwise sit idle.

3.2. GPU Kernel Code

The code is cross-platform; Linux and Windows used different APIs for shared memory and semaphores, so we simply guard platform-specific code with e.g. `#ifdef WINDOWS`. The GPU kernel code itself is completely platform-agnostic, however there are two important limitations as of this writing: first, JUCE support for VST3 under Linux is under development. Second, CUDA drivers are not available for MacOS 10.14+. We benchmarked with the JUCE plugin wrapper, and ran integration tests with a DAW on Windows.

GPU Kernel code is presented in Listing 3, and is available online on the CCRMA GitLab[1]. It is commented at the block level in the listing; the code walk in section 3.4 communicates a higher-level explanation.

3.3. GPU Programming Considerations

A few notes are provided for readers unfamiliar with GPGPU programming to get up to speed for this application. Anyone with GPGPU experience may wish to skip to the next section.

The code provided here will be executed in parallel by many threads. Specifically, we may have 10 instruments at 2,000 modes per instrument, allocate one thread each, and run 20,000 threads in parallel. In practice, the GPU will run this work in batches[2], but we expect thousands of threads to be active at any time, versus our CPU which has four physical cores. Each of the GPU threads is less general-purpose than a CPU thread. There is currently no out-of-order execution or branch prediction, and arithmetic APIs are less rich, though still very capable for many applications.

Bundles of 32 GPU threads are called a *warp*, execute in lockstep, and have some memory shared between them for fast communication. It is possible to send data across warps, but if we have subtasks that can be executed with 32 or fewer threads (even in multiple steps), it might be worth trying to keep communicating threads inside the same warp.

Memory is allocated on the device (GPU) using a special API call `cudaMalloc` that looks similar to `malloc`. It returns a *device pointer*; GPU code can read and write from this location but CPU code cannot. Sharing data between the CPU and GPU generally involves copying it to and from, though some API calls exist to simplify this or even hide latency. Our host code uses the original `cudaMalloc` for simplicity and greatest compatibility.

GPU programming is often an interesting puzzle of how to best utilize the resources of the device. For example, we have a very small number of extremely fast registers, a few kilobytes per thread of fast memory, and access to slow, but plentiful RAM (11GB on our card). Different cards may execute different numbers of single- and double-precision floating point arithmetic; in particular consumer cards tend not to have strong FP64 performance.

NVIDIA provides documentation for more details, and also provides development, debugging, and profiling tools for Eclipse or Visual Studio to help maximize performance.

3.4. Code Walk

First, we remember the GPU kernel code is launched, executes, and returns to the host, and will be rescheduled again for the next audio buffer where it is launched and returns. It may or may not execute on the same *streaming multiprocessor* as it did the previous iteration. Perhaps some cards may zero memory to prevent data leaks. And for high numbers of modes we can be sure that this assumption can't hold, because we have more threads than resources available. We can't depend on our state to be kept for us.

Our first task, therefore, is to decide how to save and restore the state of the system across executions. Recall that from the description of these filters in Section 2 that the filter's state is limited to the prior complex output, so our state is a mere 64 bits per filter: two 32-bit single-precision floats to make up a complex number. We store these sequentially in global memory, $\Re_0\Im_0\Re_1\Im_1 ... \Re_{N-1}\Im_{N-1}$.

The first few lines instruct each thread to determine which mode $m_i \in [0..N]$ it is responsible for computing. `blockIdx` and similar are local variables provided by CUDA so each thread may orient itself in the world in terms of the parallel problem the developer has written.

The next lines load state in from global memory. If we had no prior state this will be garbage, so there is a `bool reset` flag that allows for easily reseting the filter state to zero; useful for this case but also useful if we are for example switching between presets in a plugin.

Note that accessing global memory tends to be slow; in the digital waveguide acceleration portion of [17] we found we would be limited in applications if we had to access main memory to read and write each sample at audio rate speeds–but here we just need to load the state once.

Next, all the input for this mode m_i is available as the ith `ModeInfo` structure. We use this data to perhaps update the inner exponential term for the filter. Versions of the code where γ_m or α_m could be modulated on a per-sample basis would require moving this computation inside the loop.

Next, we loop to compute our audio data sample-by-sample. The complex math is provided by the CUDA library, with the exception of `cexpf` which needed to be written; it's provided in Listing 2, or in the GitLab repository.

```
__device__ __forceinline__
cuFloatComplex custom_cexpf(
    cuFloatComplex z) {
  cuComplex res;
  float t = expf(z.x);
  sincosf(z.y, &res.y, &res.x);
  res.x *= t;
```

[1] https://cm-gitlab.stanford.edu/travissk/dafx19-gpu-kernel

[2] Our request to execute 20,000 threads exceeds the 3,584 CUDA cores supported on our GPU. In such cases, the GPU will schedule warps on streaming multiprocessors as they become available, so our work is actually run in smaller parallel batches rather than *fully* parallel. Furthermore, this code will bottleneck on the availability of floating-point units and threads will compete for that resource as well. However, from the calling code's perspective the kernel executes as one unit of work.

```
  res.y *= t;
  return res;
}
```

Listing 2: cexpf implementation

We sum the newly-computed sample across all threads in a warp and write this to our output buffer; some applications may also wish to sum across warps especially if using hundreds of thousands of modes.

Our computation is done; there's nothing to clean up but we do need to write our new system state back to device memory so it can be read by the next kernel execution. Then, the function returns, our host code stops waiting, it can sum audio data generated by each warp, copy it back to shared memory, and notify the plugin in the DAW process that computation has finished.

Next, we consider how we may use all these high-Q filters.

4. EXTENDING MODAL SYNTHESIS

4.1. Baseline: Implementing Linear Models

Using the GPU filterbank, linear modal synthesis is straightforward.

First, we compute modal frequencies and decay rates offline and on the CPU. An offline analysis script trimmed sounds to several seconds, then computed DFTs of a second or more, and used these to compute modal coefficients and decay rates. This step is not time-sensitive, but did complete faster than real-time. This analysis step may also be performed at runtime, for example importing a user's samples, substituting an approximation algorithm if speed is of the essence.

The modal coefficients are stored in a file on disk. A directory with recordings captured from over a dozen cymbals is loaded into memory of the JUCE plugin on launch. The user may assign different cymbals to different MIDI notes; this is done by copying mode data from the appropriate file into `ModeInfo` structures in shared memory; the data files are small and memory-mapped so this operation is fast.

The plugin's process method is called for each audio buffer. If it detects MIDI notes that are assigned instruments, we feed an excitation signal based on the input velocity into the instruments' input signal shared memory buffer[3].

Then, the GPU processes enough samples to fill our audio buffer. Sample data is copied from the GPU's onboard memory to the RAM inside the GPU process, to shared RAM, which the plugin will read and copy back out to audio buffers.

Qualitatively, this approach works very well for bar percussion, cowbells, etc. where we have clear, exponentially decaying modes. With enough modes allocated, cymbal tails sound somewhat realistic. While the cymbal attack is instantaneous, and clearly lacks the interesting "bloom" as we move from linear to nonlinear regimes, a high density of modes bring about qualitative time-varying behavior from beat frequencies and adjacent frequencies having different decay rates. Noting the attack is limited by a linear modal synthesis model, we explore a couple ways to introduce nonlinear approximations.

[3]as in the GPU code description, we could alternatively just send a note-played signal and leave the GPU to handle articulations

4.2. Multi-sampling

Sample libraries commonly include multiple velocities for a given sound to capture tonal variation when an instrument is played at varying intensities. Taking inspiration from this, we capture several recordings for cymbals struck at different velocities.

The simplest use for the recordings at V velocity levels is to compute all the modes as before for each level, and set our filterbank to accept the union of all modes in all files:

$[m_{0,v_0}..m_{N-1,v_0}; ...; m_{0,v_{V-1}}..m_{N-1,v_{V-1}}]$

Then, as the player strikes the virtual cymbal we excite one set of V modes (or an interpolation). A few variations exist:

- Per-note velocity lookup: The control application sends a signal such as note-on velocity, which we use to pick the mode data closest to that velocity level. This data is used for the every sample of that strike contributing to $x()$.
- Per-sample velocity lookup: Similar to the above, but we continuously vary the mode parameters excited on a sample-by-sample basis: the early part of an excitation contributes from low-velocity V, and the maximum amplitude portion contributes from higher-velocity V captures (thus, the γ in $\gamma_m x(t)$ depends on $x(t)$ rather than being static).
- System-energy lookup: the amount of energy in the system governs which response we use regardless of immediate input level. This may be the most physically-accurate option.

The output of a simulation using the per-sample velocity switching approach is shown in Figure 4; the model in this case is driven by an increasing noise signal which introduces modes sampled from higher regimes over time. Of course, we are still synthesizing using a linear filterbank and not truly modeling interactions between modes at this stage, but this modification results in a more playable instrument for little overhead, so long as we stay with the maximum N allowed by our system.

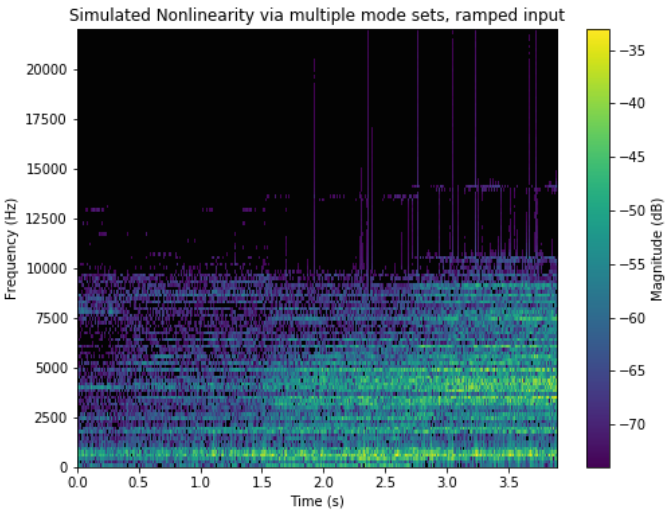


Figure 4: *Response using switched velocity samples and ramped input.*

This is similar to cross-fading samples, with the improvement that re-strikes will re-excite the system versus playing a duplicate sample. However, we immediately recognize it is an inaccurate

representation of the physical system: we would have modes excited that do not exist simultaneously in practice. As cymbals vibrate and bend, some modes will come in and out of existence. Experimentally, we saw that depending on the cymbal, roughly 60%-80% of modes were shared between any set of 3-5 files.

However, qualitatively this is still an improvement, and we get more accurate responses playing at low versus high velocities on a drum pad.

It is likely even better results could be obtained by continuously exciting the cymbal, by driving it mechanically or even with continuous strikes with mallets at a certain velocity. This allows for capturing the response to a narrow range of energy; if we take the DFT of a several-seconds long recording of a high-velocity cymbal strike, we will incorporate low-, medium-, and high-energy regimes as the cymbal sound blooms and decays.

4.2.1. Multi-sampling for Performance Characteristics

One final note on this approach - while this paper focuses on crash cymbals, a variation of the multisampling-based approach seems to work well for modeling ride cymbals: we capture a few velocity hits for each of bow, bell, and edge strikes, and use those to excite certain modes in one shared modal system. Modifying playing position between the bell and bow allows energy to build up and brings out a sound that is difficult to capture with pure sampling.

4.3. Frequency Shifts

An extension to this method is to capture the frequency shifts by pairing modes across velocity levels that are at nearby frequencies: $\omega_{m,v1}$ and $\omega_{m',v2}$ under some threshold frequency shift. At performance time, we not only choose which modes are excited, but are able to change their frequencies as well.

As an offline computation we have many options as far as clustering and nearest-neighbor algorithms; if we ever needed to perform it real-time, a tracking algorithm such as that included in PARSHL [21] or linear programming [22] are worth considering, if they cope with spectrally dense content.

4.4. Implementing Modal Coupling

Finally, we introduce modal coupling to approximate a phenomena in the nonlinear regimes: identify the set of frequencies that only appear in high-velocity captures, and couple each one of these to one or more modes that is present at lower velocities by $\psi_{m,n}$, representing a coupling from mode m to n. Modes may feed multiple other modes, for example both $\psi_{m,n1}$ and $\psi_{m,n2}$ may exist to establish a one-to-many relationship. At performance time, with some system energy envelope, we establish an energy transfer between modes $\propto k\psi_{m,n}$. This is one area where the high-Q filters we've chosen excel; we simply immediately scale down the previous complex output value y for one mode, and inject that energy into another mode (this can be a new term adjacent to the input response term $\gamma_m x(t)$). When the update is performed at a zero-crossing this preserves phase; qualitatively in terms of sound, waiting for a zero-crossing does not seem critical here.

In practice, the prior sections on multi-sampling and frequency gating may capture some of this behavior. However, this approach has the opportunity to introduce real-time performance control over how much coupling is present in the system – k in the above expression. It may easily be suppressed, exaggerated, made frequency-dependent, etc.

While this evolves our synthesis beyond a simple modal filterbank excitation, it is still related to the linear modal model, and is not a true physical simulation of the chaotic regime of cymbals. Nevertheless it is a computationally efficient approximation that starts to bridge the gap between our simple modal responses and a real-time playable cymbal synthesizer.

GPU programming details have been absent from previous sections, as the choice of modes is up to the plugin running in the DAW process and we can treat the GPU as a black box. It is relevant here, however, as modes directly influence each other and must communicate, and the GPU kernel would need to perform the trading of energy between modes.

The fact that threads run in groups of 32 has some relevance. If we can keep groups of modes that couple in clusters of 32, then programming becomes much easier since threads inside a warp have access to a block of shared memory that may be used for computation. It is not a showstopper if we instead must communicate using global memory, just less optimal.

We tried a few sub-approaches as to how to choose modes: modes could pair with the closest (in frequency) unused higher-regime mode, modes could look for modes close to an octave up, fifths, etc., or we could simply pair modes randomly. Qualitatively, these resulted in similar effects, with more difference when using smaller N.

A sound example for modal coupling is in Figure 5. We drive a virtual cymbal mechanically with constant noise, and k is set fairly high so modes are introduced quickly once crossing a manually-specified energy threshold between regions.

As a comparison with the real cymbal spectrograms, we see modes emerging over time, an improvement over our first step of pure linear modal synthesis, and have model controls over when and how quickly the modes marked as high-energy-regime emerge, which adds an interesting performance dimension. Work remains, however, to automatically model the attack for multiple stick hit velocities; our examples here all involved some manual iteration.

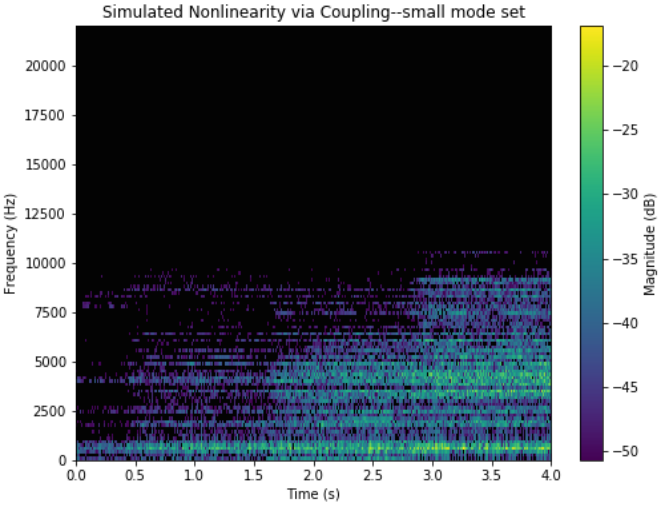


Figure 5: *Response using simulated modal coupling; coupled sources/targets only*

```
__global__ void filterbankKernel(
    float* yprev,        // Previous state
    const ModeInfo* mi,  // Mode frequencies and dampings.
    const float* input,  // Input signal.
    float* output) {     // Buffer we write output samples to.
  // Initialization - each thread figures out its
  // place in the world (i.e., it's the ith of N threads).
  int i = threadIdx.x + blockIdx.x * blockDim.x;
  int which_warp = (int)(i / 32);
  bool is_first_thread_in_warp = (i % 32) == 0;

  // Load prior state from global memory.
  cuComplex y;
  y.x = yprev[2 * i];
  y.y = yprev[2 * i + 1];
  // If prior state should be discarded, zero it.
  if (mi[i].reset) {
      y.x = 0.0f;
      y.y = 0.0f;
  }

  // Introduce variables to make expressions more readable later.
  cuComplex input_amp;
  input_amp.x = mi[i].amp_real;
  input_amp.y = mi[i].amp_real;
  cuComplex input_complex;
  cuComplex exp_term;
  int which_drum = (int)(i / MODES_PER_DRUM);
  const float *input_base = input + (BUFFERSIZE*which_drum);

  // Regenerate the exponential term at the start of each buffer.
  // This is moved inside the loop in case of parameter interpolation,
  // modifying params on zero-crossings, or similar cases.
  cuComplex e_term_tmp;
  e_term_tmp.x = -mi[i].damp;
  e_term_tmp.y = mi[i].freq;
  exp_term = custom_cexpf(e_term_tmp);

  // Main loop - run enough cycles to generate the whole buffer.
  for (int samp = 0; samp < BUFFERSIZE; samp++) {
    y = cuCmulf(exp_term, y);
    // We always compute input for now, but could skip this multiply
    // if we know there's no input.
    input_complex.x = input_base[samp];
    input_complex.y = 0.0f;
    y = cuCaddf(y, cuCmulf(input_complex, input_amp));

    // Merge audio across all threads in the warp (32 threads),
    // and store the sample in our output buffer.
    // This sums in parallel, and completes in log_2(32) = 5 steps.
    // We use the real part of the complex sample as output audio.
    float merge_output = y.x;
    for (int offset = 16; offset > 0; offset /= 2)
        merge_output += __shfl_down_sync(0xffffffff, merge_output, offset);
    if (is_first_thread_in_warp) {
        output[which_warp*BUFFERSIZE + samp] = merge_output;
    }
  }
  // Save state back to shared memory, so we may restore it
  // on the next kernel launch.
  yprev[2 * i] = y.x;
  yprev[2 * i + 1] = y.y;
}
```

Listing 3: *CUDA Kerne* *vailable in Section 3.4.*

5. ACKNOWLEDGMENTS

Thanks to the conference organizers, and to the anonymous reviewers for suggestions, comments, and time.

6. CONCLUSIONS

This work presented a GPU modal filterbank system architecture, with a walkthrough of CUDA code that implements the GPU kernel. This enabled acceleration of our modal synthesis building blocks; we were resource-constrained with a single cymbal on the CPU but can run an entire ensemble of cymbals in real-time on the GPU with room to spare. Next, a few extensions to basic linear modal synthesis were presented toward bringing modal cymbal synthesis closer to real models which are highly nonlinear. These extensions, while approximations still based on a linear synthesis system, improve playing dynamics and allow introduction of new performance controls.

7. REFERENCES

[1] Antoine Chaigne, Cyril Touzé, and Olivier Thomas, "Nonlinear vibrations and chaos in gongs and cymbals," *Acoustical science and technology*, vol. 26, no. 5, pp. 403–409, 2005.

[2] KA Legge and NH Fletcher, "Nonlinearity, chaos, and the sound of shallow gongs," *The Journal of the Acoustical Society of America*, vol. 86, no. 6, pp. 2439–2443, 1989.

[3] Stefan Bilbao, "The changing picture of nonlinearity in musical instruments: Modeling and simulation," in *Proc. Int. Symp. Musical Acoustics*, 2014.

[4] M Ducceschi and Cyril Touzé, "Modal approach for nonlinear vibrations of damped impacted plates: Application to sound synthesis of gongs and cymbals," *Journal of Sound and Vibration*, vol. 344, pp. 313–331, 2015.

[5] Michele Ducceschi and Cyril Touzé, "Simulations of nonlinear plate dynamics: an accurate and efficient modal algorithm," in *18th International Conference on Digital Audio Effects (DAFx-15)*, 2015.

[6] Quoc Bao Nguyen and Cyril Touzé, "Nonlinear vibrations of thin plates with variable thickness: Application to sound synthesis of cymbals," *The Journal of the Acoustical Society of America*, vol. 145, no. 2, pp. 977–988, 2019.

[7] Lauri Savioja, Vesa Välimäki, and Julius O Smith, "Audio signal processing using graphics processing units," *Journal of the Audio Engineering Society*, vol. 59, no. 1/2, pp. 3–19, 2011.

[8] Lauri Savioja, Vesa Valimaki, and Julius O. Smith III, "Real-time additive synthesis with one million sinusoids using a gpu," *128th Audio Engineering Society Convention 2010*, vol. 1, 5 2010.

[9] Fernando Trebien and Manuel Oliveira, "Realistic real-time sound re-synthesis and processing for interactive virtual worlds," *The Visual Computer*, vol. 25, pp. 469–477, 5 2009.

[10] Gabriel Cirio, Ante Qu, George Drettakis, Eitan Grinspun, and Changxi Zheng, "Multi-scale simulation of nonlinear thin-shell sound with wave turbulence," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 110, 2018.

[11] Jeffrey N Chadwick, Steven S An, and Doug L James, "Harmonic shells: a practical nonlinear sound model for near-rigid thin shells.," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 119–1, 2009.

[12] Jose Belloch, Balazs Bank, Lauri Savioja, Alberto Gonzalez, and Vesa Valimaki, "Multi-channel iir filtering of audio signals using a gpu," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 05 2014, pp. 6692–6696.

[13] Belloch Rodríguez, *Performance Improvement of Multichannel Audio by Graphics Processing Units*, Ph.D. thesis, 2014.

[14] Jose A Belloch, Alberto Gonzalez, Enrique S Quintana-Orti, Miguel Ferrer, and Vesa Välimäki, "Gpu-based dynamic wave field synthesis using fractional delay filters and room compensation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 2, pp. 435–447, 2017.

[15] Stefan Bilbao and Craig J Webb, "Physical modeling of timpani drums in 3d on gpgpus," *Journal of the Audio Engineering Society*, vol. 61, no. 10, pp. 737–748, 2013.

[16] Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron van den Oord, Sander Dieleman, and Koray Kavukcuoglu, "Efficient neural audio synthesis," *arXiv preprint arXiv:1802.08435*, 2018.

[17] Travis Skare and Jonathan Abel, "Gpu-accelerated modal processors and digital waveguides," in *Linux Audio Conference 2019 (LAC-19)*.

[18] Max Mathews and Julius O. Smith III, "Methods for synthesizing very high q parametrically well behaved two pole filters," in *Proceedings of the Stockholm Musical Acoustics Conference (SMAC 2003)(Stockholm), Royal Swedish Academy of Music (August 2003)*, 2003.

[19] Jonathan S. Abel, Sean Coffin, and Kyle Spratt, "A modal architecture for artificial reverberation with application to room acoustics modeling," in *Audio Engineering Society Convention 137*, Oct 2014.

[20] NVIDIA Corporation, "NVIDIA CUDA toolkit documentation," `https://docs.nvidia.com/cuda/`, [Online].

[21] Julius O Smith and Xavier Serra, "Parshl: An analysis/synthesis program for non-harmonic sounds based on a sinusoidal representation," in *Proceedings of the 1987 International Computer Music Conference, ICMC; 1987 Aug 23-26; Champaign/Urbana, Illinois.[Michigan]: Michigan Publishing; 1987. p. 290-7.* International Computer Music Conference, 1987.

[22] Julian Neri and Philippe Depalle, "Fast partial tracking of audio with real-time capability through linear programming," in *21st International Conference on Digital Audio Effects (DAFx-18)*, 2018.

Author Index