

Department of AI Music and Music Information Technology
Central Conservatory of Music
Beijing, China
25–27 October 2025

Tenth International Conference on Technologies
for Music Notation and Representation

Edited by Zhang Yuan



TENOR BEIJING 2025

INTERNATIONAL CONFERENCE
ON TECHNOLOGIES FOR MUSIC
NOTATION & REPRESENTATION





The 10th International Conference on Technologies for Music Notation and Representation (TENOR 2025)
Beijing, China · October 25–27, 2025

Edited by Zhang Yuan

Published by
Digital Score Lab, Central Conservatory of Music, Beijing
ISBN: 978-0-6481592-1-6

©2025 Central Conservatory of Music. All rights reserved.
These proceedings are published for academic and research purposes.

License
These proceedings are licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0).

Conference website
https://zhuanti.ccom.edu.cn/tenorbjen/index.htm
https://tenor-conference.org

Disclaimer
The views expressed in individual papers are solely those of the authors and do not necessarily represent the position of the Central Conservatory of Music or the TENOR 2025 Organising Committee. This volume is published as conference proceedings and does not constitute a commercial publication within China.

PREFACE

The conference team at CCOM – Central Conservatory of Music (Department of AI Music and Music Information Technology, Central Conservatory of Music) – is pleased to present the proceedings of the 10th TENOR Conference 2025, hosted at the Central Conservatory of Music in Beijing from October 25 to 27, 2025. From a broad and diverse international response, the program of TENOR 2025 features 32 scholarly papers, 16 works presented across two concerts, 3 workshops and 1 commissioned project, bringing together works selected through the open call, contributions curated by DigiScore's global partner institutions, and a commissioned work by the Digital Score Lab of the Central Conservatory of Music, reflecting the richness and vitality of current research and creative practice in music notation and representation.

The call for submissions proposed a focus on AI and Neuroscience, highlighting the evolving role of Artificial Intelligence in music and score creation practices. The keynote presentation by Craig Vear, Digital Musicking – Transformational Encounters with the Digital Score, explored the changing role of notation in musical meaning-making. Vear emphasized how contemporary approaches to score creation increasingly embrace color, shapes and symbols, as well as gamified, networked, generative, agential and intelligent systems. The papers in this volume reflect both the perspectives of composers experimenting with notation in technologically mediated formats, and the research of computer scientists developing new approaches to music representation.

Great effort was made to provide the best possible conditions for the presentation of sonic works, reflecting China's decades of sustained development in electroacoustic music research and performance. A particular highlight of the concert program was its distinctive focus on Chinese traditional instruments, realized through the collaboration between international artists and CCOM performers. We especially acknowledge and thank the students and faculty from the Department of Chinese Traditional Instruments for their invaluable contribution, which vividly demonstrated the creative potential of intercultural and technologically mediated musical exchange.

We thank all the researchers and artists who responded to our call, all the authors represented in these proceedings, our collaborative partners from universities across China, the TENOR Steering Committee, and especially our colleagues at CCOM whose dedication made TENOR Beijing 2025 possible. We offer our special thanks to Zhang Yuan, the Conference Organizing Chair, for her outstanding leadership and for serving as the editor of these proceedings.

Li Xiaobing
Honorary Chair

Ken Fields
Conference Chair

ORGANIZING COMMITTEE

Professor Kenneth Fields, Conference President
University of Chinese Academy of Sciences

Professor Li Xiaobing, Conference President (Honorary)
Central Conservatory of Music

Zhang Yuan (Nancy), Conference Organizing Chair
Central Conservatory of Music

Kenneth Fields, Paper Review Chair
University of Chinese Academy of Sciences

Zhang Xinran (Sam), Paper Review Co-Chair
Central Conservatory of Music

Li Yunpeng (Raymond), Sonic Work Chair
Wuhan Conservatory of Music

Zhao Yixuan (Latoro), Sonic Work Co-Chair
Central Conservatory of Music

Wang Xueying (Olivia), Media Tech Chair
Central Conservatory of Music

Yu Haibo (Laurel), Coordination & Volunteer Chair
Central Conservatory of Music

STEERING COMMITTEE

Cat Hope (General Chair)	Monash University, Melbourne - Australia
Craig Vear	Nottingham University - UK
Georg Hajdu	Hochschule für Musik und Theater Hamburg - Germany
Rama Gottfried	ZHdK, Zurich - Switzerland
Sventlana Maraš	University of Applied Sciences and Arts Northwestern - Switzerland
Zhang Yuan	CCOM, Beijing - China

PAPER REVIEWERS

We express our sincere thanks to the following scholars and artists who generously contributed their time and expertise to the blind peer-review process for the scholarly papers of TENOR 2025.

Ali Balighi	Texas Tech University
Andrea Agostini	Conservatoire of Turin
Annie Yen-Ling Liu	Peking University
Benjamin Holzapfel	Hamburg Musik gGmbH
Blake Stevens	Peking University
Cat Hope	Monash University
Craig Vear	University of Nottingham
David Kim-Boyle	The University of Sydney
Dong Zhou	Leuphana University
Emmanouil Karystinaios	Johannes Kepler University
Federico Visi	Luleå University of Technology
Fernando Egido	individual composer
Georg Hajdu	Hochschule fur Musik und Theater Hamburg
Haotian Zhou	Central Conservatory of Music
Ilana Shapiro	University of California, San Diego
James Harkins	Xinghai Conservatory
James Saunders	Bath Spa University
Jean-Louis Giavitto	STMS, CNRS - IRCAM, Sorbonne University
Jiayue Wu	University of Colorado
Josephine Calandra	IReMus
Karim Haddad	IRCAM
Ken Fields	Chinese Academy of Sciences
Marc Battier	IREMUS
Mathias Josefson	MathiasJosefson
Matteo Bizzarri	Scuola Normale Superiore
Nicola Privato	University of Iceland
Paul Turowski	University of Liverpool
Pavlos Antoniadis	University of Ioannina; IRCAM
Phillip Kirlin	Rhodes College
Pierre-Luc Lecours	University de Montréal
Qiaoxi Zhang	Queen Mary University of London
Sanjay Majumder	Style tree
Stephen Pope	FASTLab and HeavenEverywhereMedia
Takuto Fukuda	University of Pécs
Veronika Reutz Drobnic	HfM Karlsruhe
Xinran Zhang	Central Conservatory of Music
Xiran Sun	Victoria University of Wellington
Yuan Zhang	Central Conservatory of Music
Yunpeng Li	Wuhan Conservatory Of Music

TABLE OF CONTENTS

005 **Analysing the Digiscore Corpus: The Pacmman Framework**
Craig Vear

022 **Syntactic Complexity of Music with Lambek Calculus**
Matteo Bizzarri, Satoshi Tojo

032 **The Role of Artificial Intelligence in Electroacoustic Score Generation: A Review**
Ali Balighi, Hideki Isoda, Ph.D.

042 **A Data Model and a TScore Representation for the TABstaff+ Notations for Ableton Pus**
Markus Lepper, Baltasar Tranc´on y Widemann

055 **Visualizing Vibrato: a Library of Wiggles to Broaden Technical and Cultural Horizons**
Barnaby Brown, Austin Oting Har

062 **Engravinggnn: a Hybrid Graph Neural Network for End-to-End Piano Score Engraving**
Emmanouil Karystinaios, Francesco Foscarin, Gerhard Widmer

068 **Mykdf — Der Kunst Der Fuge Own Mixdown, Realized with Tscore and Sig/Adlib**
Markus Lepper, Baltasar Tranc´on y Widemann

079 **Vippsa: A Web-Based Tool for Video Score Synchronization and Annotation**
Xavier Davenport, Craig Vear, Christoph Guttandin

084 **Artificial Authenticity - Real Authority: 你們是蟲子] (You are bugs), an AI-assisted playbite on Luigi Nono's ...*sofferte onde serene*... for piano and tape**
Pavlos Antoniadis

094 **Detecting Notational Errors in Digital Music Scores**
Léo Géré, Nicolas Audebert, Florent Jacquemard

104 **Moz'Lib & Pwformax: Rewiring Cac Heritage**
Julien Vincenot, Örjan Sandred, Juan S. Vassallo

106 **Sonogram-Based Digital Scores and Flipped Learning: Breaking Down Hierarchies in Chinese Electroacoustic Music Education**
Ruibo Zhang (Mungo), Mingyue Guan, Kaiying Dong

123 **Chordsnest: Developing a Multiphonics Palette for Maxscore**
Aigerim Seilova, James Tsz-Him Cheung, Georg Hajdu

129 **Modeling of Cognitive Classification and Segmentation Operations During Music Listening with Morfos Music Analysis Software**
Jos´ephine Calandra[abc], Jean-Marc Chouvel[a], Myriam Desainte-Catherine[c]

141 **Interpreting Chord Distances of Tonal Pitch Space: a Correlation Study with Corpus-Based Chord Progression Frequencies**
Naohiko Yamaguchi, Tetsuro Kitahara

148 **Modeling Ragtime Structure and Style From Rhythmic Notation Alone**
Phillip B. Kirlin

158 **Cognitively-Aware Symbolic Notation Via Simulation: a Software-Only Framework for Adaptive Music Representation**
Sanjay Majumder, Neal Anderson

169 **AI Reads Fluxus: Text Scores, Language Games, and The Art of Being Literal**
Nicola Privato, Thor Magnusson

177 **Graphic Notation as Generative System: Visual Logic in Composition and Improvisation**
Ming Yang

185 **Livelily: The Evolution of a Live Scoring / Live Sequencing Environment**
Alexandros Drymonitis

193 **Declarative Music Composition with Event Graph Transformations**
Taosheng Qiu, Ryutaro Ichise

201 **Composing and Designing Web-Based Digital Scores: Practices for Networked Music Performance**
Xiao Xiang

206 **AI Sentience is just Around the Corner: Solaris 2, a Creative-AI Score and Punk Rock Band with Human in the Loop**
Fabrizio Augusto Poltronieri, Craig Vear

214 **OscillaScore: A Modular Platform for Graphic Notation in Networked Music Performance**
Dr. Rob Canning

222 **Interpretation and Learning of Modular Synthesis: Three Studies**
Pierre-Luc Lecours, Maryse Gagnon-Legault

231 **Gesture as Score: A Hybrid Notation System for Guzheng and Electroacoustic Music**
Bihe Wen

236 **Practice Strategies for Polyphonic Music: Exploring the Role of Gesture Controllers in Enhancing Technical Stability and Optimizing Mapping Methods**
Yu-Hsin Chang

241 **Just Intonation in Bach**
Andrea Agostini, Daniele Ghisi

253 **Spar-Rt: Evolving Representations of Musical Timbre, Pitches, and Tuning in Real-Time Using Symmetric Projection Attractor Reconstruction**
Wentao Hao, Merlin Chao, Mateusz Soli ´ nski, Elaine Chew

261 **Notation for Organ Music: Balancing Prescriptive and Descriptive Models for an Instrument Without Standardisation to Unlock Compositional and Performative Potential**
DDr. Martin Ritter, Dr. Alyssa Aska, Mag. Pablo Mari˜na

268 **From Shape to Music: Contour-Conditioned Symbolic Music Generation**
Qiaoxi Zhang, Mathieu Barthet, Anna Xambó Sedó

276 **Chinese Characters as Digital Scores: A Prototype for Explainable, Semantics-Driven Composition**
Zhang Yuan, Zhang Xinran

ANALYSING THE DIGISCORE CORPUS: THE PACMMAN FRAMEWORK

Craig Vear
University of Nottingham, Music Department & School of Computer Science
craig.vear@nottingham.ac.uk

ABSTRACT

This research reconceives digital scores through a multidimensional sensory lens: the PACMMAN framework. In doing so, digital scores are positioned as interactive, embodied, and rich medium. The study explores the digital score as a dynamic, co-creative system that reshapes musical meaning through real-time interaction, phenomenological engagement, and multisensory integration. By synthesizing insights from experimental performance contexts, it reveals how digital scores function as responsive, algorithmic entities that foster collaborative meaning-making between musicians and technological systems.

Key themes include the score's transformation into an interactive, sentient collaborator; its phenomenological presence through gestural and spatialized interfaces; and its capacity for multisensory expression, blending visuals, haptics, and sound. The composer's role shifts from authoritative author to system designer, structuring emergent, improvisatory frameworks that balance algorithmic processes with performer agency. Interdisciplinary integration dissolves boundaries between notation, movement, and multimedia, while socio-political and neurodiverse perspectives highlight the score's potential for inclusivity and cultural resonance.

Ultimately, this research proposes an ecological model of musicking, where performers, algorithms, and environments co-create within an entangled network. The digital score becomes an active participant in a fluid, process-oriented practice, challenging hierarchical structures and redefining musical presence as a collective, exploratory act. This PACMMAN framework not only reimagines music scores future but also expands the very conception of digital musicking as a living, evolving dialogue between human and non-human agents.

1. INTRODUCTION

The Digital Score is a five-year European Research Council-funded project (2021–2026) exploring the transformation of musical notation through digital and computational technologies. Led by Principal Investigator Prof. Craig Vear (University of Nottingham), the project involves four international partners: Prof. Cat Hope (Australia), Prof. Sandeep Bhagwati (Canada), Prof. Kenneth Fields (USA), and Prof. Li Xiaobing with support from Dr. Zhang Yuan (China).

During a recent project meeting, a key question emerged: *What have we learned about digital scores?*. This paper responds directly to that question by presenting findings from a longitudinal investigation into creativity and meaning-making through the DigiScore corpus, a collection of digital score case studies generated and studies through the project. While the project has also generated insights into digital musicianship and AI's role in new forms of musicking, this paper focuses on the analysis and evaluation of the corpus.

The paper is structured in three parts: first, an overview of the project's aims; second, a detailed account of the methodology, including data capture techniques and the PACMMAN evaluation framework; and third, a discussion on the long-term impact of engaging with digital scores.

1.1 Defining a Digital Score - the "message"

Vear [1] [2] defines a digital score as *a communications interface for musical ideas between musicians that utilizes the creative potential of digital technology*. While digital scores encompass diverse approaches and technologies, they share a common purpose: *to communicate musical ideas through technological mediation*.

The Digital Score project (hereafter DigiScore) specifically investigated *how* digital scores communicate musical ideas, *what* mediates this communication, and what musical materials could serve as meta-materials for future digital scores. Taking a phenomenological approach, the project examined these questions from within the creative act of music-making itself. This represents a deliberate shift away from external perspectives that treat music as either textual artifact or performance object. While such perspectives remain valuable, they fail to account for the full spectrum of experiential and meaning-making processes in musical practice.

A core principle of DigiScore is that the technologies constituting a digital score fundamentally shape the communication of musical ideas. While notational language remains important, the project particularly emphasizes how scores extend "tendrils of affordance" [1], drawing musicians into their worlds through networks of connections and possibilities. This perspective aligns with Marshall McLuhan's foundational media theory concept that "the medium is the message" [3]. McLuhan argued that communication media exert greater influence on human perception and behaviour than their specific content - a television program affects us differently than a printed article

Copyright: ©2025 Craig Vear et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

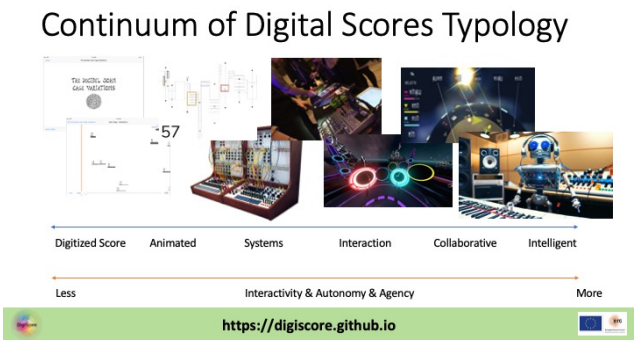


Figure 1. Continuum of digital score types

not just through its content, but through its very form as a medium.

Digital scores, as technological media, similarly transform musical meaning-making in ways that transcend their notational content. The project seeks to understand these medium-specific effects, which often operate at subconscious levels, to reveal how digital technologies fundamentally reshape musical communication and creativity.

1.2 Continuum of digital score types

When examining digital scores, there is a risk of focusing too narrowly on either the technology or the notational content in isolation, thereby overlooking the deeper connections and meaning-making processes that underlie their surface appearance or semantic language. Vear contends that we should instead assess digital scores through their dynamic networks of connections - how these evolve during both the creation and realization of the work.

This connective perspective reveals digital scores as existing along a continuum, from basic implementations (such as digital screens displaying static or animated notation) to complex systems incorporating AI and intelligent agents. Crucially, at each point along this spectrum, the technological medium fundamentally transforms what can be communicated and how it facilitates creative engagement. These shifting relationships reshape musical creativity itself, ultimately exerting a profound influence on musical ideas and practices.

The continuum of digital score types (see Figure 1) can be classified into three distinct sections based on step-changes in their relationship with musicking: Referential Screen, Interactive Systems, and Co-operative Code.

Part A - Referential Screen includes three types:Augmented Page: Enhances the musician's relationship with the score through screen-based technology, displaying static images of the printed page. Technological Conductor: Uses fixed media elements to guide the musician through a structured, linear timeline of sonic and visual elements. Collaborating Score: Integrates real-time manipulation of sound elements, actively participating in the creation of the music alongside the performer. These digital scores journey alongside the musician, providing direction and influence, but without deep interaction or agency compared to the following types.

Part B - Interactive Systems involves three types that engage more dynamically in the creative process: 4. Animated Score: A visual-based system where the design and signals are dynamic, evolving with the music. 5. System-as-Score: Utilizes hardware and electronics to create tactile environments for interaction. 6. Creative System: Combines pre-defined audio, dynamic visuals, and sound processing, reacting in real-time during performance. These systems are spontaneous and generative, with the full creative potential realized only during performance.

Part C - Co-operative Code consists of four types: 7. Performative Code and Hacked Bodies: Involves physical movements and data streams, with the machine responding interactively. 8. Gesamtkomposition: Coordinates multiple media streams in real-time with autonomous, generative behaviour. 9. Networked Ensembles – Connected Score: Links performers in a network, organizing and distributing compositional materials. 10. Living Score: Uses intelligent computation to actively co-create within the musicking process. These scores are co-operative, with clear shared tasks between the musician and the digital score, offering a creative presence and autonomy in the performance.

With this typology in mind, it should be clear that different types of technologies will mould the way we think, perceive, and interact with the music ideas that are embedded within digital scores. The next section defines the methodology for generating data about such relationships.

2. METHOD

The investigation unfolded through three phases designed to examine digital score practices. The first phase focused on capturing the complete creative lifecycle through multi-layered data collection [4]. During the creation stage, we gathered musicians' formal artistic proposals and intention statements as reflective documents, while simultaneously collecting real-time creative process documentation through journals and blogs. The complete digital score artifacts - including all code, media assets, and technical documentation - were preserved as primary research objects. For the performance/ realization stage, we employed video documentation alongside immediate stimulated recall interviews to capture initial reactions, followed by more reflective semi-structured interviews conducted within days of performances. Audience surveys provided external perspectives, while legacy questionnaires administered four weeks later revealed lasting impacts on both performers and creators.

The second phase involved pattern analysis as the dataset grew. We employed grounded theory methodologies, with systematic coding verification using NVivo software and AI summarization tools. Through iterative examination of case studies, distinct patterns began to emerge. This analytical process developed Vear's PACMMAN framework (see below), which synthesized these emergent understandings into a coherent structure for assessing digital score practices.

Currently in the final analytical phase, Vear is conducting a comprehensive evaluation of the complete dataset cor-

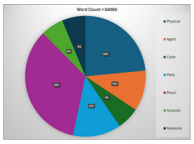


Figure 2. Corpus Survey word count.

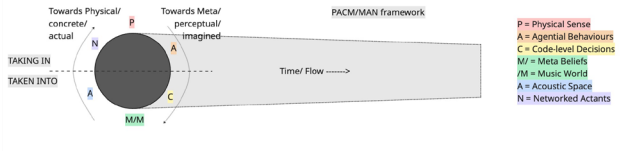


Figure 3. The PACMMAN concept

pus - for this paper it amounts to 60 detailed case studies derived from 35 distinct digital scores (composers and performers forming different case studies). This involves aligning key phrases and concepts with the PACMMAN framework's categorical structure, followed by both quantitative and qualitative evaluation. Preliminary findings from the 64,000-word dataset show particular concentration in the "Music" (21,949 words) and "Physical Senses" (15,049 words) categories, as visually represented in Figure 2. This distribution suggests particularly rich engagement with these dimensions throughout our case studies, highlighting their central importance in digital score practice.

The methodology's phased structure allowed for both depth and breadth of understanding, moving from rich data collection through progressive analysis to comprehensive evaluation, while maintaining flexibility to incorporate emergent insights throughout the research process. Complete dataset can be found https://digiscore.github.io/pages/outputs/

2.1 The PACMMAN framework of connections

The PACMMAN framework builds on the *Taking-In/ Taken-Into* concept originally presented in [1]. It also considers several other theoretical frameworks that have provided inspiration over the course of this project, for example Chris Small's notion of *Musicking* [5], Simon Emmerson's *Living Electronic Music* [6], and David Borgo's *Sync or Swarm* [7], amongst others.

In Figure 3, the circle on the left-hand side is split into the 2 realms of the *Taking-In/ Taken-Into* framework with North being affordances and South which is more subjective and imagined. The diagram also suggests a movement between the two, although this is not a linear movement around the fields.

For the purposes of the survey, the seven fields were defined as:

- **Physical Senses**: The stuff we see, touch, read (notation), hear (reduced to physical sound, or harmonic relationships), bio/movement of other agents (human or machine)
- **Agential Presence/ Behaviours**: How we perceive the score behaving, the tangible presence or feel that something is there, the liveness of pre-recorded sound, the behavioural meaning of animation
- **Code-base decisions**: Reasoning presence, a mind in action, how we give meaning to random choices or generative AI, how we rationalise graphical behaviours or on-screen generative images, why something did what it did, or seemingly ignored me
- **Meta belief**s: the contexts embedded within the score:
  - MACRO) the cultural context within which it is positioned. e.g. the digital score is embedded with referential materials such as previous works, styles, identities
  - MICRO) the aesthetic and artistic and cultural beliefs (its world view) that guide the operationality of the score. E.g. the digital score is embedded with specific interactive behaviours and belief governing it's interactivity and/ or responses
- **Music world**: I am somewhere. The imaginary world a score conjures. The meta-physical space created by the score. E.g. in *Nautilus* this is NOT the Unity world (this is part of the Physical senses dimension), but it is the feel of the music, and the imaginary landscape in the musicians mind.
- **Acoustic space**: The actual acoustic space created by the realisation of the score. Using Emmerson's Local/Field-Stage metaphor it has more to do with the physical evocation of space and how that is staged in the music. Additionally, it can describe the actual physical sensation of the acoustic space on the musician, and this might be different depending on performance spaces.
- **Networks/ Participants**: Those participating in the music. Representations of self and others as a complex network of inter-relationships. This could be fellow performers, audience, sounding objects. The focus is on the web of actants in a musicking network and its dynamics over time

3. FINDINGS FROM PACMMAN ANALYSIS

This section presents the high-level findings from Phase 3 - survey, analysis and evaluation. It addresses each field of the PACMMAN framework in isolation and does not attempt to draw relationships across the fields.

3.1 P – Physical Senses

This field focused on the tangible materials of the DigScore corpus by examining what we see, touch, read, hear, and how we experience the touch and feel of such an object. The analysis focused on these emerging themes:

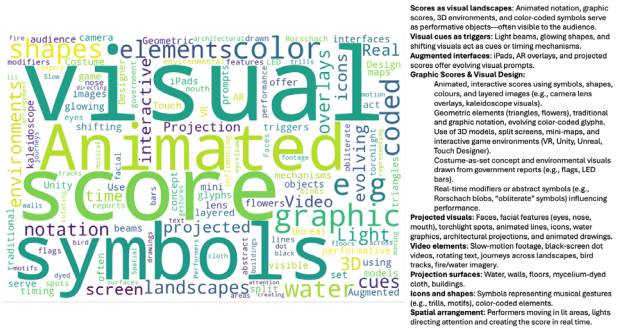


Figure 4. Physical field, visual focus word cloud

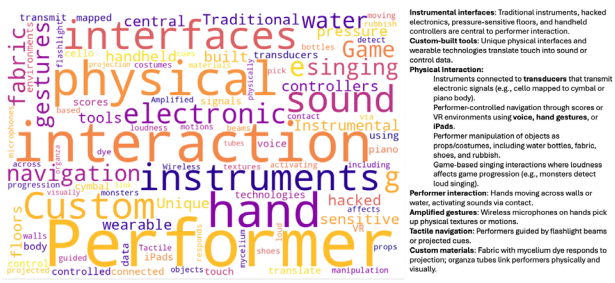


Figure 5. Physical field, materials focus word cloud

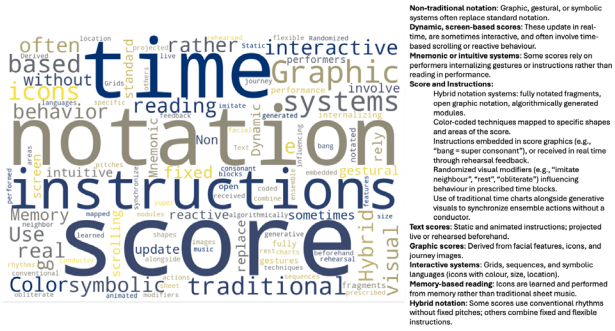


Figure 6. Physical field, notation focus word cloud

3.1.1 The Stuff We See - Visuals as Meaning

The analysis revealed that visual elements are not merely decorative but central to how scores are understood and performed. Animated components, colour-coded symbols, 3D visuals, and projections function as both notation and metaphor. For example, projections of water or moving shapes may cue performance actions while simultaneously reflecting deeper narratives such as migration or climate change. Shifting animations and visual symbols provide cues for timing and emotion, while transforming icons - like flowers, triangles, or facial expressions - convey dynamic musical intentions. These scores resemble maps or ritual objects, emphasizing the interplay between perception and meaning.

Visual elements play vital metaphorical, structural, and emotional roles, shaping how performers engage with the works. In pieces like *Speechless* (by Cat Hope), the visual score acts as a kinetic prompt, guiding performers through gesture and intention rather than fixed rhythm or pitch. These components form a symbolic language that invites interpretation and improvisation.

Visual elements and game-like interfaces extend the expressive range of performance beyond traditional notation. Colors, shapes, and symbols (in say *GuitaRPG* by Xavier Davenport) operated as signifiers of musical gestures or moods, offering abstract and layered meanings. Performers stated that they became immersed in visually rich environments - sometimes including virtual or projected spaces - with their own internal logic. Furthermore, projections on surfaces such as water or fabric heightened or distorted meaning, emphasizing ephemerality. Likewise, imagery of journeys, camps, or natural elements carries symbolic weight, evoking shared cultural or emotional narratives such as in *Open Orchestra*. In general, visual elements are not supplemental - they are integral to the composition's structure and message, merging aesthetics, intention, and performance into a unified idea.

3.1.2 The Stuff We Touch - Material as Interface

Some case studies discussed how touch became an active, interpretive gesture in these works. Musicians engage not only with traditional instruments but also with hacked electronics, handheld controllers, and unconventional materials such as fabric, architecture, or wearable technology. These objects are often amplified, allowing motion to directly produce or manipulate sound. Pressure-sensitive surfaces and haptic interfaces turn physical contact into sonic expression, making embodiment both visible and audible.

For audiences, witnessing this tactile interaction creates a tangible connection between performer, space, and material add to a blurring of the boundaries between composer, performer, and environment (a common theme in this research). Pieces like *Exercices d'étrangeté III Mycorrhizia* by Sandeep Bhagwati highlight how drawing, pressing, or manipulating materials generate both structure and sound, collapsing creation and performance into a single expressive act. The materials themselves "push back," offering resistance and feedback that shape the performer's response.

In this domain, meaning emerges through this physical dialogue - touch becomes a language of intention and exchange. Touch is not merely a means of execution - it is a mode of authorship. Performers shape sound and meaning through direct engagement, with environments like water or fabric becoming instruments in themselves. This tactile process reinforces performer agency and creates a shared sensory experience, where body, material, and sound are intimately entwined.

3.1.3 The Stuff We Read - Notation as Memory and Suggestion

As with many (most) scores that are discussed at TENOR, the contemporary notation systems utilised in many DigiScore case studies move away from prescriptive symbols toward suggestive, interpretive forms (for example *Netrono-*

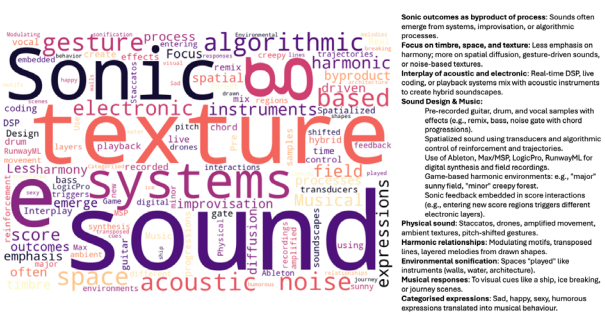


Figure 7. Physical field, sonic focus word cloud



Figure 8. Physical field, bio/ movement focus word cloud

mia by Ken Fields and Ethan Cayko). These may include icons, hybrid scores, or mnemonic graphics that prompt memory rather than dictate specific actions. Often, scores evolve in real time via screens or augmented reality overlays, requiring performers to respond to shifting visuals (for example *Machine à sons* by Jonathan Bell). Instructions may appear as embedded text within animations or as feedback emerging from rehearsal, positioning performers as co-creators who balance structure with freedom.

In this context, we could suggest that notation was reimagined as a flexible, dynamic tool - less a rigid directive than a framework for creative interpretation. Findings showed how animated, symbolic, or procedural notational systems invite performers to make decisions in the moment, emphasizing improvisation, internalization, and agency. Reading becomes a performative act in itself, where meaning emerges through engagement rather than fixed outcomes. There were also many issues with this too, such as the nature of extreme sight-reading.

In works like *Kaleidescore* by Lauren McCall, notation functioned as behaviour or scenario; it was time-based, interactive, and open-ended. Rather than issuing commands, these scores offer prompts and cues, encouraging performers to navigate ambiguity and construct meaning through interpretation. Symbols, gestures, and visual modifiers carry semiotic weight, forming an evolving language learned through practice and experience.

Visual notation also played an emotional and aesthetic role. It rendered the compositional process visible and treats the score as a performative object. Reading becomes embodied and improvisational, shaped by the performer's response to time, shape, and motion (such as *Shadow Aria* by Jaslyn Robertson). In general, there was a large interest in exploiting the boundary between written and improvised performance. As such, notation became a site of negotiation between structure and spontaneity, memory and invention where meaning is to be created through active interpretation and imagination.

3.1.4 The Stuff We Hear - Sound as Relational Information

In many case studies, sound emerged not from fixed compositions but from processes, whether improvised, gestural, interactive, or contextual (such as *Returns and Simulacra* by Solomiya Moroz). The findings suggested that emphasis shifted from melody and harmony to timbre, space, and texture. Spatialization and real-time digital signal processing contributed to immersive environments (for example *Nautilus* by Craig Vear). While tools like Ableton, Max/MSP, and field recordings generated hybrid, evolving soundscapes. Sonic events often responded to visual or physical triggers such as a performer's movement or a shifting icon, making sound an emotional or spatial echo of visual and gestural elements.

Several musicians suggested that sound was not a finished product but a trace of an action, shaped by physical gesture, algorithmic logic, or environmental input. It reflected behaviour, decision-making, and spatial dynamics, functioning as a narrative voice that heightened relationships between performer, system, and space. Works like *Point Line Piano* by Jarek Kapuscinski and OpenEndedGroup's Paul Kaiser and Marc Downie exemplified this approach as drawing produces sound directly, unifying gesture and audio. Sound in these contexts are spatial, responsive, and feed the emotions. In *Dynamic Landscapes* by Elsa Kitching, harmonic shifts may align with symbolic visuals - such as a "minor" key underscoring a dark forest - creating associative resonance. With both these pieces, performers shaped the sonic environment in real time, becoming active agents within a living soundscape.

Overall, there was a trend for meaning to arise through cause and effect: sound is interpreted in relation to movement, visual elements, or systemic behaviours. Whether reacting to facial expressions, site-specific materials, or animated imagery, sound becomes an audible record of interaction - an echo of gesture that invites both performers and audiences into an active, co-creative experience.

3.1.5 Bio/Movement of Agents - Embodiment and Agency

The movement of others, whether human or machine, was central to meaning-making in case studies such as *Jess+* by Craig Vear. The musicians involved felt that the robotic arm was not merely a tools but a collaborator, where its co-agency created a fluid, reciprocal relationship between humans and machine. In this case, musicians responded to each other and to machines with intention and improvisation, producing expressive, relational meaning. The boundaries between performer, audience, and system begin to blur, transforming performance into a collaborative field of interaction.

Embodied AI were treated as co-performers, such as in *Solaris* by Craig Vear. Their autonomous behaviours where read as expressive acts, prompting human performers to engage in a responsive dialogue. In *KC+* by Craig Vear and Kerry Francksen, the manipulation of visual forms becomes a kind of choreography, visible to both audience and fellow performers, shaping the piece's unfolding in real time.

In these digital scores, gesture and motion serve as primary conveyors of narrative and emotion, whether enacted by human bodies or algorithmic systems. Movement communicates struggle, coordination, surprise, creating meaning through visible intent and response. As such, meaning emerged through inter-agency: the interplay between human and non-human bodies, between movement, memory, and interpretation. The performer's physical presence (human, machine or AI) became part of the score, and motion became a shared language of intention, culture, and embodied understanding.

3.1.6 Discussion of the Physical Senses Field

Understanding a digital score from this multidimensional approach to the senses transformed our understanding of digital scores into an interactive, embodied, and visually rich perspective. An aspect of the digital score approach that needs serious consideration from a compositional, performative, or experiential perspective. Each sense - sight, touch, sound, motion, and notation - offered not a fixed meaning, but a platform for exploration. The result was a living, evolving form of music-making, where interpretation is communal and meaning emerges through action, presence, and interplay.

Collectively, these case studies can be seen to redefine musical meaning as a dynamic, co-created process rooted in interaction, embodiment, and perception. The score is no longer treated as a static composition to be played or heard, but as a multisensory experience; something to be seen, touched, read, felt, and negotiated in real time. With meaning arising not from a single element such as sound or score, but from the interplay between gesture, material, system, and the musicking mind. Both performers and audiences are invited to engage actively, shaping and interpreting the work through their perceptions and responses.

What emerges is a pluralistic, participatory form of meaning-making. In most cases, notation becomes suggestive rather than prescriptive, visuals function as expressive environments, and sound becomes a trace of action and intent. Interaction, interpretation, embodiment, and perception contribute to an evolving narrative shaped by presence and performance. Rather than delivering a fixed message, these works cultivate meaning as a living, emergent process, co-created across sensory and conceptual domains. Perhaps this is something for our music educators to be mindful of.

3.2 A - Agential Presence/ Behaviours

This field focused on how we perceive the score behaving, i.e. the tangible presence or feeling that something is there. The analysis focused on these emerging themes:

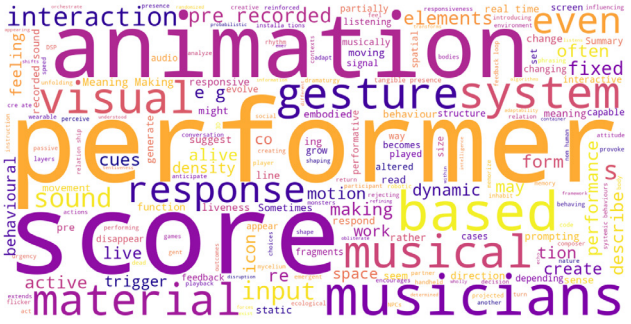


Figure 9. Agential field word cloud

3.2.1 How We Perceive the Score Behaving

In these works, the score transcends its traditional role as a static container of information, instead functioning as an intelligent, responsive system that actively shapes the performance. Performers frequently describe the score as exhibiting quasi-cognitive behaviours - listening, adapting, and transforming musical material in real-time based on their input. Through algorithmic functions like "echo" or "obliterate," familiar motifs are dynamically reorchestrated, creating an emergent dramaturgy where memory and innovation interact as generative forces.

The architecture of these scores exists in a deliberate tension between fixed and fluid elements. While composers establish fundamental temporal structures and visual triggers, probabilistic algorithms introduce controlled variability. It was mentioned that this hybrid approach demands a new mode of musicianship - one that privileges adaptive responsiveness over rote execution, as performers navigate a musical landscape that evolves through interaction.

A particularly compelling aspect is the score's perceived agency even those without AI. Performers often report the system functioning as a critical collaborator - capable of refining, rejecting, or redirecting musical input. This creates a creative feedback loop where each gesture is met with algorithmic response, prompting continuous reassessment of musical strategies. Some advanced systems employ recursive processing, analyzing previous input to generate subsequent notation through transformation chains that may decay, mutate, or evolve.

Three key characteristics define this paradigm:

1. Co-creative agency: The score actively participates in shaping musical outcomes through responsive algorithms
2. Structural hybridity: Fixed compositional elements coexist with probabilistic, emergent behaviours
3. Negotiated dramaturgy: Meaning arises through real-time interaction between performer decisions, system responses, and the work's memory of previous states

This framework redefines musical interpretation as a dynamic conversation rather than prescribed execution, where the most compelling performances emerge from the tension between compositional intention, algorithmic behavior, and performer creativity.

3.2.2 The Tangible Presence or Feel That Something Is There

Musicians readily described their encounters with digital scores as something far more profound than interacting with mere notation - they report experiencing what can only be called a presence. These scores seem to listen attentively through real-time audio analysis, respond thoughtfully via adaptive visual transformations, and make decisions through sophisticated algorithmic processing. What makes this experience particularly striking is how it transcends the boundaries of the screen - through motion sensors that track gestures, robotic arms that mirror human movement, and spatialized sound systems that respond to a performer's position in the room, the score becomes an embodied participant in the performance space.

This sense of presence manifests in several distinct yet interconnected ways. The score's agency extends physically into the performance environment through responsive technologies - robotic limbs that move in counterpoint to musicians, lighting systems that react to musical phrasing, or kinetic sculptures that transform according to algorithmic decisions. Even ostensibly static visual elements develop what performers perceive as intentionality; graphical symbols that expand with crescendos or dissolve during silences become interpreted as meaningful responses rather than mere visual feedback. Many musicians described developing what feels like genuine dialogue with these systems, testing boundaries and interpreting reactions much like they would with human collaborators - one musician characterized her digital score as "a duet partner with strong opinions."

This transformation from passive notation to active participant fundamentally changes the nature of musical interpretation. Performers find themselves engaged not in decoding instructions but in negotiating with what feels like an intelligent counterpart - one that communicates through multiple sensory channels simultaneously and demands the same kind of real-time adaptability required in ensemble playing with human musicians. The resulting performance exists in a unique creative space where meaning emerges continuously from this embodied negotiation between human creativity and algorithmic behavior, challenging traditional distinctions between interpreter and score, between execution and collaboration.

3.2.3 The Liveness of Pre-Recorded Sound

In these performances, pre-recorded sound undergoes a fundamental transformation - shedding its static nature to become a dynamic, responsive partner in the musical dialogue. Through real-time digital signal processing and interactive triggering systems, fixed audio layers gained a sense vitality, modulating their texture in response to performer activity, shifting density according to musical phrasing, or moving spatially in reaction to physical gestures. Some musicians described these elements not as passive playback but as active collaborators that appeared to remember previous phrases and respond accordingly, creating an evolving musical discourse.

These systems fostered a form of musical relationship where even structurally fixed material develops phenomenological liveness. Performers reported developing a rapport with the audio, learning through repeated interaction how certain gestures will provoke specific sonic responses. This knowledge becomes embodied, transforming the performance into a ritualized exchange where musicians anticipate and react to the system's behaviours. In game-like scenarios or installations, pre-recorded NPC cues function similarly, their apparent simplicity belying the rich, improvised responses they inspire from human performers.

What emerged were instances of liveness in electronic performance. The "aliveness" of these pre-recorded elements stems not from their internal variability but from their relational role within the performance ecosystem. Through interactive modulation, anticipatory relationships, and ritualized interaction, structurally fixed audio gains performative vitality. Some systems remembered, the performer anticipates, and together they created a musical present that seemed to be always becoming - where even "dead" recordings could be said to pulse with life through their dialogic relationship with the performer.

3.2.4 The Behavioural Meaning of Animation

Far from serving as mere visual decoration, animation in some digital scores operated as a dynamic performative language. These moving elements communicated through a vocabulary of behaviour rather than static symbols; for example, a thickening line might suggest intensification, accelerating icons suggest rhythmic urgency, while flickering or floating visuals could imply ambiguity or invitation. Performers become adept at reading these animations not just for their literal content but for their expressive intent, responding to the score's visual gestures much as they would to a conductor's nuanced motions.

The relationship between animation and sound often flows both ways. Some systems transform drawn lines in three-dimensional space directly into evolving musical material - notes, rhythms, and harmonies that develop their own algorithmic life. Others create feedback loops where animations dynamically adjust their speed or density in response to live audio analysis, fostering a genuine dialogue between visual movement and sonic response. This reciprocity transforms the score from an instructional document into an active participant in the musical conversation.

Certain animations constructed entire environments to be inhabited rather than simply read. Mycelium-like tendrils might weave between performers' positions, or ephemeral "monsters" appeared fleetingly, challenging musicians to capture them sonically before they vanish. These temporal and spatial behaviours created a sense of ecological interplay, where open and willing musicians must engage with urgency and presence within the score's living system.

The spatial distribution of these animations further enhances their behavioural impact. Whether projected across walls, displayed on wearable surfaces, or cast from handheld devices, the placement of visual elements turns physical space into an active score component. Performers navigate these environments through full-body awareness, where proprioception and gestural response become as crucial to

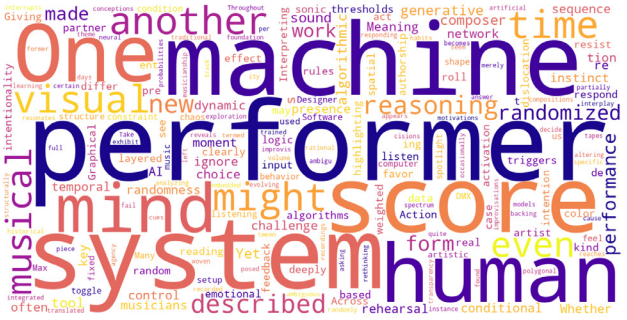


Figure 10. Code field word cloud

the musical outcome as traditional instrumental technique. The animated score thus becomes what one cellist described as "a landscape we move through together" - a shared space of visual, physical, and sonic negotiation where meaning emerges through embodied interaction.

3.3 C - Code-base decisions

This field focused on how we perceive reasoning presence, a mind in action, how we give meaning to random choices or generative AI. The analysis focused on these emerging themes:

3.3.1 A Mind in Action: Interpreting Human and Machine Intentions

Throughout the exploration of AI-integrated musical works, one theme emerged: the *presence* of a mind at work - even when that mind is partially artificial. Across a spectrum of compositions and performance systems, decisions were made that challenge traditional conceptions of authorship, agency, and intention. Whether through neural networks analysing rehearsal tapes, generative algorithms altering scores, or conditional triggers responding to performer cues, each piece revealed an evolving interplay between human musicianship and algorithmic reasoning. For example, *Nautilus* (by Craig Vear) or *DigiTabla* (created by tabla performer and composer Shawn Mativetsky), are systems where a backing track is composed of pre-recorded improvisations that randomly trigger sounds but interrupts itself if a performer reaches a certain level of behaviour. To the musicians the machine appears to *listen*, to *decide*, and occasionally to *ignore*. It becomes not merely a tool but an improvising partner, with ambiguous motivations. This ambiguity - this not-quite-human reasoning - forms a foundation for rethinking performance itself.

3.3.2 Reasoning Presence: Why Did It Do That?

Several musicians noted that when the digital scores seem to respond - or fail to respond - they were left asking: *why did it do that?* Often, the answer is found not in rational cause-effect sequences, but in layered probabilities, thresholds, or embedded rules. Algorithms might favour specific choices based on historical data, as in Iran Sanadzadeh's *502 Days of self* where machine learning models trained on 502 days of rehearsal recordings translated a performer's habits into visual elements. These systems exhibit what might be termed a "reasoning presence": an intentionality that resists full transparency. When randomness is structurally woven into a score (for example, using weighted probability distributions), the system doesn't simply act arbitrarily; it executes a different kind of logic. These logics, while opaque, are not unintelligible - they just resist the intuitiveness we apply to human collaborators.

3.3.3 Giving Meaning to Generative AI: Interpreting Randomness

Musicians often found themselves attributing intentionality to machines, reading significance into outcomes that may be statistically random. One artist described feeling "like a machine, like an instrument", highlighting the dissociation between input and perceived output. In another case, performers of *Jess+* expressed scepticism about whether a score was truly reactive, only to be convinced later by emotional resonance and feedback. From the data we can infer that meaning here was constructed retroactively. When a digital score lands on an uncanny moment of synchronicity, it feels "meant." Yet in truth, it might have been the result of an algorithmic coin toss. What is of interest here is how the human mind insisted on interpretation - we generate narrative and causality even when facing stochastic structures. A few musicians even discussed how the unpredictability of rest sequences caused them to sit silently, awaiting their next activation; a moment that felt both deeply musical and deeply disempowering.

3.3.4 Graphical Behaviours and On-Screen Imagery: Rationalising the Interface

The graphical representations of some digital scores appeared to bring a new semiotic challenge to existing knowledge and taught patterns. In some cases, colour represented amplitude or panning; line curvature denotes frequency modulation. Yet these visuals are not mere ornamentation - as in Cat Hope's *Speechless* - they serve as proxies for sonic behaviours, demanding real-time interpretation. In *GuitaRPG* (by Xavier Davenport) and *Kaleidescore* by Lauren McCall visual elements encoded musical, algorithmic, and spatial data. At one instance a musician described navigating a map with a single key, the direction randomized each time. This dislocation from linearity pushed the musician to reinterpret how movement and sound are linked. Another performer noted how text spirals spun at different speeds, a simple but profound temporal mapping that transformed reading into a form of listening.

3.3.5 Why Did It Ignore Me? Algorithmic Indifference

Having outlined the above positive attributes, there were some negative feelings. There was a sense that the machines do not prioritize us, nor do they favour our feelings, fatigue, or artistic instincts (unless explicitly programmed to). One performer described their discomfort when their sound was shaped by another's input, their autonomy subsumed by the system. Another recounted how triggering visuals or sonic events required a sequence of precise key

presses and script activations - a procedural hurdle that separated musical instinct from execution. Musicians spoke about the anxiety of not being in control, about being ignored by systems that only responded within narrow thresholds or temporal windows, for example, a Max patch might suppress overtones once a pitch gets too loud; a spotlight might never illuminate a performer despite their readiness. These moments of silence or inaction underscored an essential "other-ness" of some generative systems.

3.3.6 A Network of Conditions: Systemic Interactions and Constraints

The computational/ code-based behavioural systems described in these digital scores were often layered and conditional; and formed a significant part of the compositional process and concept. The general intention here was to create a dynamic condition network that reflected a musicking-mind in *flux*. The response to this was the need for realising musicians to toggle between control and surrender. This is something that is not readily taught in academia, yet appears to be a condition across many innovative scores.

3.3.7 Scoring the Unscored: Software as a Creative Partner

Decibel ScorePlayer software can be used to reconfigure notation into something more participatory, even democratic. This system, and other scroll-based case studies, allowed large-scale adjustments post-rehearsal, immediate visual feedback, and dynamic highlighting of individual parts. This flexible infrastructure let musicians "see" their own voice and place in the ensemble clearly - a new way of making sense of sound in time. A performer described how easily scores were adapted mid-process, emphasizing the overlooked practicalities of technological tools in artistic contexts. These tools are not passive; they shape what kind of music can be made, rehearsed, and performed.

3.3.8 Discussion: The Composer as Designer of Chance

Within the DigiScore corpus we found that in general, the composer's role shifted from author to a kind of musicking-system designer. Many works described switching between predetermined and randomized forms, making "structuring randomness" a compositional act. One system began with a fixed timeline but then slowly randomized its form. Another used "Time Out" hand gestures to allow human override. Yet another used score projections and live DMX lighting to choreograph space and time. This is not necessarily an abdication of authorship, but could be considered a redefinition. By curating randomness through weighted algorithms, conditional rules, or generative visuals, musicians can shape a meta-performance: not just *what* is played, but *how* choice itself is enacted.

Across many of the case studies, what emerged was not a singular approach to AI or interactivity, but a shared fascination with machine behaviour as both constraint and co-creator. Whether the digital score, or the machine, listens, mishears, ignores, or surprises, it challenged musicians to think beyond their own trained musical instincts and to engage with a reasoning presence that is foreign, partial, and



Figure 11. Meta Beliefs field word cloud

provocatively inhuman. Yet we found that in that ambiguity lies potential for new interpretations, new forms of attention, and new stories made from random choices. It would appear that we, as human musicians, fill in the gaps and assign meaning where the machine leaves room.

3.4 M - Meta beliefs

This field focused on the macro and micro contexts embedded within the digital score. The analysis focused on these emerging themes:

3.4.1 Meta Beliefs, World-views, and Artistic Frameworks

Several of the works in the corpus framed their artistic creation as a metaphorical journey or exploration. *Pathways* by Dejana Sekulic, and *Nautilus* by Craig Vear drew on symbolism of journey, for example, transcontinental voyages of refugee's, or those of the nautilus, a deep-sea mollusc whose spiral shell represents layered, immersive navigation through complex environments. These journeys extended beyond physical realms into abstract soundscapes and interactive spaces, reflecting an aesthetic world-view that valued exploration, discovery, and movement through intricate and multi-dimensional contexts. The connection to ancient symbols ("nautilus" as "sailor") tied traditional cultural mythologies to digital score creativity, blending historic and modern perspectives into a unified aesthetic vision.

Another batch of works focused on immersive environments as co-creators. These works (such as *Point Line Piano* or *Dynamic Landscapes*) situated immersive visual and sonic environments as active participants in performance rather than passive backgrounds. Other works (such as *Queer Temporal* by Solomiya Moroz) used large-scale projections, multisensory stimuli, and spatial atmospheres to shape both performers' and audiences' perceptions, emotions, and interactions. In a poetic sense, they underscore a belief in environment as a co-creator of meaning and experience, emphasizing embodiment and dynamic engagement with space. In this context immersion is both physical and conceptual, fostering deep sensory connection and emotional resonance.

Some works used organic, natural patterns as creative models for the digital scores. Fractal geometries, natural soundscapes, and biological processes served as inspirations and structural models for works such as *The Forecast*

by Kate Milligan. This reflected a world-view that creativity is intertwined with nature's complexity, self-similarity, and organic growth. In this context, artistic form and musical structure mirrored natural phenomena (such as mycorrhizal networks or cellular automata in Sandeep Bhagwati's *Exercices d'étrangeté III* Mycorrhizia), positioning art as an extension of natural systems and cycles. As such, nature's patterns become metaphors for interconnection, emergence, and evolution within the creative aesthetic of a digital score.

An abundant theme in the DigiScore corpus was the role of improvisation as a core concept, and finding a balanced with structure. The ideas described were generally heterophonic and fluid, emphasizing improvisation within defined frameworks. This balance between freedom and constraint suggests a cultural ethos that values open-endedness moderated by form, enabling coherence and emergent order from apparent freedom. This dynamic interplay rejected rigid authoritarian control of "the composer" in favour of emergent, collaborative creation between musicians.

Many of the case studies transcended traditional disciplinary boundaries, integrating graphic scores, interactive visuals, movement, and sound into unified, multimedia experience where artistic media were porous and interdependent. In these works, this hybridity fostered richer modes of storytelling and engagement, breaking down silos between visual, auditory, and performative art forms.

Themes of censorship, marginalization, and socio-political resistance permeated as a central narrative in some works. For example, in *Shadow Play* by Jaslyn Robertson, improvisers' responded to "censoring lights" metaphorically embodying struggles against suppression, echoing broader concerns around power, control, and voice within society. This reflected a critical cultural consciousness that foregrounds marginalized perspectives - particularly of refugees, women, and neurodiverse individuals - highlighting digital score creation as a site of resistance and empowerment.

Digital Syzygy by Andrew Hugill embraced neurodiverse ways of experiencing and creating music, including autistic and d/Deaf perspectives. The concept of "digital syzygy" signified new modes of cognitive and sensory alignment enabled by digital tools, challenging normative models of musical cognition. This inclusivity broadened definitions of creativity and highlighted technology's potential to facilitate novel forms of connection and expression beyond conventional sensory or neurotypical boundaries.

The motif of migration - both avian and human in *Pathways* by Dejana Sekulic- and the tensions between open and closed borders reflect larger socio-political and ecological concerns about displacement, belonging, and movement. This awareness situated this work within contemporary dialogues on refugee experiences, community, and environmental systems. It suggests an interconnected world-view that considers life trajectories as shaped by fluid yet constrained pathways within social and natural ecologies.

The adoption of RPG metaphors (for example in *GuitaRPG*) framed performers as explorers in richly detailed, dynamic worlds, embodying values of play, learning, adaptation, and identity negotiation. It was felt that this resonates with contemporary digital cultures that foreground agency, multiplicity, and iterative discovery. The RPG model emphasizes exploration and experimentation over fixed outcomes, aligning with the work's fluid and emergent artistic ethos.

3.4.2 Broader Cultural Contexts (MACRO)

There were several broader cultural contexts (macro) embedded in several of the digital scores. For example:

- **Social and Political Engagement**. Some works engaged deeply with social issues such as refugee trauma, displacement, identity, and belonging. Themes of speechlessness and ineffability reflect experiences of trauma and marginalization. The inclusion of diverse cultural backgrounds and histories - particularly of refugees and women - foregrounds social solidarity and community-building as central to these realisations.

- **Technological Integration and Post-Human Perspectives**. In several works in the corpus, AI, robotics, and virtual reality were regarded as not mere tools but integral to the creative ecosystem, challenging traditional human-centred notions of music-making. This openness to non-human "performers" (algorithms, robots) reflects a post-humanist world-view, expanding artistic agency beyond human actors and embracing hybrid creative processes.

- **Queer Theory and Nonlinear Temporality**. Theses works disrupt normative temporalities and social conventions by exploring queer concepts of time - non-linear, cyclical, and fluid. This challenges traditional narratives of progress and climax, instead privileging repetition, cycles, and openness. In this sense, space and time become sites of subversion and re-imagination, aligning with broader queer theoretical frameworks.

- **Inclusivity and Democratic Access**. Many works consciously attempted to break down barriers between professional and amateur, disabled and non-disabled musicians, as well as embracing broad stylistic pluralism and multicultural participation. Democratic creative processes ensure accessibility and shared ownership of artistic outcomes, reflecting progressive cultural values of equity and diversity.

3.4.3 Artistic, Aesthetic, and Cultural Beliefs (MICRO)

On a micro level, digital scores were embedded with identifiable artistic, aesthetics and cultural beliefs, values, principles and themes. For example:

- **Fluidity, Flexibility, and Performer Agency.** Some digital scores were designed as mutable, modular frameworks that emphasize performer agency and improvisation. In others, algorithmic elements introduced unpredictability, ensuring performances remain fresh and responsive. The performer was seen

as co-creator rather than mere executor of pre-defined material.

- **Human Imperfection and Gesture**. Human gestures, imperfection, and expressive flexibility were valued over mechanistic precision in some digital scores. Conducting gestures and bodily expression become essential components of musical communication, allowing space for silence and interpretation, fostering shared rhythmic and emotional pulses.
- **Interactivity and Embodiment**. In many digital scores, multimedia integration - video, virtual environments, graphic notation - expanded sensory engagement and embodiment. Performers' identities and bodily expressions were integral to meaning - making, with virtual reality and animation fostering new forms of agency and discovery.
- **Community and Collective Identity**. Some of the digital scores functioned as a framework for collective identity, emphasizing collaboration, pluralistic interpretations, and social interconnectedness (such as *Villanelles de Voyelles* by Sandeep Bhagwati). The boundary between composer, performer, and audience were blurred, reinforcing the communal nature of the musical experience.
- **Non-linear, Queered Temporality**. Some digital scores emphasized repetition without sameness, and cycles without closure, these works embraced toporhythm (spatial-temporal offsets) and process over fixed goals. This fostered aesthetic experiences that prioritize journey and transformation over destination or resolution.
- **Multistylism and Cultural Pluralism**. In some digital score multiple musical traditions and contemporary techniques were combined to create hybrid forms that respect and transcend cultural boundaries. These score's openness to diverse interpretations reflects a pluralistic worldview, valuing difference and hybridity as strengths.

3.4.4 Discussion: Cultural Identity and Storytelling

Through both a macro and micro intermingling, the following themes were embedded into some of the digital scores highlighting the versatility of the digital score concept and technological enmeshment to convey larger cultural identities and new avenues for storytelling through music:

- **Connection to Place and Heritage**. These works used local historical texts and Indigenous cultural symbolism (journeys, spirits, bird tracks) to root itself in cultural heritage and place. These digital scores become narrative tools expressing community stories, ancestral journeys, and belonging.
- **Temporality and Memory**. Some digital scores focused on non-linear time through cyclical structures and metaphorical devices like wishing wells, reflecting impermanence and memory layering. Ephemeral visual elements like water graphics underscored themes of transience and flux, deepening the sense of historical and place-based identity.
- **Collaboration and Shared Agency**. A common theme across many works was how traditional roles of "composer, performer, conductor, and audience" overlapped, encouraging collective interpretation and mutual listening. Inspired by natural metaphors such as tree networks and mycelium, these works promoted democratic participation and interdependence.
- **Technology as Integrated Meaning**. Digital environments and immersive media served as metaphors for natural interconnectedness, not as superficial effects. Many of the digital scores used technology to enhance multisensory experience, enabling new modes of discovery and spatial musical interaction.
- **Improvisation and Embracing Imperfection**. Flexible score structures encourage exploration within frameworks that merge traditional forms (e.g., Indian raga) with experimental practices. "Errors" and failures are embraced as creative opportunities, reflecting a learning process.
- **Narrative Metaphors and Ambiguity**. The journey motif organized movements around arrival, tension, struggle, and homecoming, while remaining open-ended. Natural and cultural metaphors framed musical interactivity and growth, allowing emotional engagement beyond literal storytelling.
- **Embodiment and Interactivity.** Performances emphasize bodily awareness and facial communication, enhanced by VR and immersive media that grant agency and blur spectator roles. These scores invited playful navigation between fixed cues and fluid improvisation.
- **Plurality and Hybridity**. Multiple musical styles and cultural traditions coexisted and blended, creating a hybrid artistic language. Many works embraced ambiguity and multiplicity as core aesthetic and epistemological principles, supporting a constructivist approach where meaning evolved collaboratively.
- **Multiplicity, Fluidity, and Process-Oriented Art**. The music and performance practices encourage multiple, radically different interpretations, reflecting postmodern values of plurality, process, and flux. The rejection of a "definitive" artistic product underscored art as a living, evolving experience co-created by performers, audiences, and environments. This epistemological stance privileged openness, ambiguity, and ongoing transformation over closure or finality.

The digital score in this corpus could be said to embody a rich constellation of meta beliefs and world-views that position their embedded musical ideas as a dynamic journey through immersive, organic, and interactive environments. In general, creativity was framed as a balance of

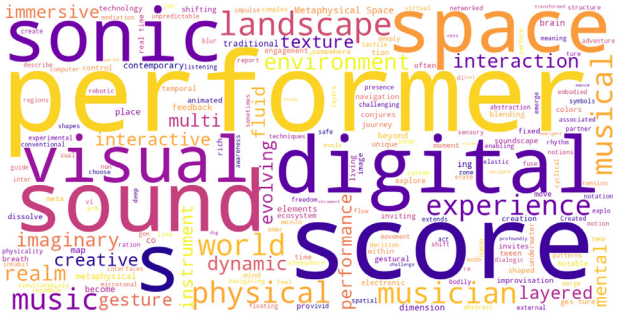


Figure 12. Music field word cloud

freedom and structure, deeply informed by natural patterns and cultural narratives. Social and political consciousness - especially regarding marginalization, neurodiversity, migration, and queer temporality - intertwined with technological innovation and collaborative processes. The result could be seen as a fluid, pluralistic, and process-oriented artistic practice that celebrates multiplicity, hybridity, and the interdependence of human, natural, and digital worlds.

3.5 M - Music world

This field focused on the imaginary landscape/ music-world created by the score. The analysis focused on these emerging themes:

3.5.1 Immersive Musical Environments: The Performer's Journey Through Metaphysical Spaces and Dynamic Scores

Musicians suggested that they felt like some of the digital scores in the corpus allowed them to inhabit fluid, immersive environments that were shaped by interactive agents, multi-sensory feedback, and emergent sound worlds. These spaces challenge fixed notions of musical structure and place, inviting a dynamic, co-creative process where sound, visuals, technology, and embodied gesture fuse into a complex ecosystem (such as *Open Orchestra* by Ben Hopkinson). The musicians mentioned how they explored a layered experience as they navigated these richly animated musical realms, and the imaginary worlds conjured by scores (in this case to Antarctica). Other musicians discussed the metaphysical spaces that arise from the interplay of performer and digital elements (such as in *GuitaRPG* by Xavier Davenport), and the fluid mental landscapes that underpin musical creativity.

3.5.2 "I Am Somewhere": The Performer's Immersive Experience

At the core of many reflections upon performance and intention towards composition, is the sense of performer's experience of "being somewhere". Due to the unique homogenisation of rich media and interactive technology prevalent in many digital score, it generated unique, often shifting, and deeply immersive environments. Unlike conventional performance settings grounded in physical instruments and static scores, these new spaces are dynamic, multisensory realms. Performers frequently described sensations akin to floating underwater, swimming through an otherworldly soundscape, or navigating a metaphysical space that is simultaneously safe and challenging. Performers reported entering a "flow state" or "zone" [8] where their awareness expanded and contracted, merging their internal and external worlds. This flow is underpinned by complex feedback systems - brain-computer interfaces, electronic sensors, digital displays - that both guide and disrupt traditional control mechanisms (such as *Jess+* by Craig Vear). This embodied engagement extended beyond solo exploration to include collaboration with human and robotic partners, networked setups, and real-time electronic mediation. The musical "place" therefore exist not only in physical space and objects, but in temporal, cognitive, and digital dimensions, fostering a heightened sense of presence, immersion, and responsive interaction.

3.5.3 The Imaginary World a Music Score Conjures

As has been previously documented and argued in [1], the digital score itself becomes a dynamic, multi-layered environment constructed for, say, animated landscape of shapes, colours, symbols, and sounds that through the affordance of the media and technologies involved can constantly evolve. The embedded motifs can function as navigational aids, creative prompts, and expressive gestures, guiding performers through an interactive, "choose your own adventure" narrative, yet guided by the overarching musical idea. The evolving nature of many digital scores invites continuous interpretation, improvisation, and co-creation. It encouraged playful engagements where patterns emerge and dissolve unpredictably, challenging performers to balance control with surrender. Musicians described this as dialogic, a partnership with the score and virtual elements (e.g., a digital bass flute (*Nautilus*) or robot arm (*Jess+*) where mutual adaptation and responsiveness are essential. Through these hybrid digital-physical scores, musicians move beyond traditional instrumental vocabulary into new realms of sound and movement. The score mediates between the performer's inner ear, bodily gestures, and the external sound world, becoming a meta-instrument that could reshape notions of composition and performance in the digital age.

3.5.4 The Metaphysical Space Created by the Score

Beyond tangible sounds and visuals, the interaction generates a metaphysical dimension - a non-physical space of temporal dilation, abstraction, and layered interaction. In a digital score such as *The Forecast* by Kate Milligan, its meta-space transcends conventional time and rhythm, creating elastic, non-linear experiences that require improvisational freedom balanced with responsibility. In this metaphysical realm, distinctions between performer and observer, live and electronic, will and involuntary response dissolve. Other digital scores have integrated brain-computer interfaces, random lighting cues, and networked interactions to create a "zone" charged with tension between control and chance, human and machine, intention and emergence. Musicians described inhabiting this liminal space as a form of shared consciousness or secondary awareness, where their actions negotiate through overlapping layers of live sound, visuals, and digital feedback loops. These

interactions generate unpredictable sonic textures and gestural behaviours, blurring physical distance and temporal constraints. As such, these spaces are a "playing field" where meaning emerges through the evolving game of interaction, where the performer's decisions and the score's responses weave a unique, ephemeral sonic architecture.

3.5.5 The Feel of the Music and the Imaginary Landscape in the Musician's Mind

Musicians reported experiencing the music as an evolving terrain - a sensory and emotional landscape rich with textures, colours, and gestures. They discussed their connections to sound-worlds that oscillate between pitched and unpitched elements, microtonal nuances, and extended techniques, blending organic and synthetic timbres into fluid, atmospheric soundscapes. They mentioned how the music felt like a dialogue between control and surrender, and how they had the options to selectively engage with visual and sonic elements, navigating a mutable landscape. In certain cases they visualized the digital score as a map populated with shifting regions, symbols, and colours associated with specific techniques and sonic qualities (for example, *GuitaRPG* and *Shadow Play*. Other mental maps were dynamic and non-linear, blending fragments of memory, perception, and creative impulse (such as *Returns and Simulacra*. This phenomenon supported the navigation from literal reading to abstract improvisation, enabling a deeply personal yet interconnected musical experience. These inner landscapes were sometimes described as tactile and embodied, where small agential and media shifts, breath, and gesture have profound sonic consequences. Some musicians imagined dancing with invisible partners - other performers, robotic arms, or gestural videos - in a fluid choreography of sound and movement. Others discussed slow-motion details and cyclical structures enabling them to experience different layers of rhythm and texture. This created dreamlike atmospheres where sonic molecules float and interact, inviting exploration, discovery, and playful experimentation.

3.5.6 Discussion: The Coalescence of the Performer, the Score, and the Metaphysical Space

Findings from this field suggested that musicians created and realised works through a dynamic space shaped by their physical interaction with instruments, responsive digital visuals, and real-time sonic feedback. Their journeys were both tangible and imaginative, guided by, say, gestures, brain activity, and technological mediation. In this sense the score, no longer a static set of instructions, becomes a living, evolving landscape—part spatial map, part improvisational playground—that invited musicians to co-create within its unfolding sonic universe.

This ecology transcends traditional disciplines, merging music, visual art, digital media, philosophy, gaming and neuroscience, to name but a few, into a single expressive medium. The result is a musical space where time and perception stretch and warp, allowing meaning to emerge organically. Within this environment, the performer's mind constructs its own vivid, tactile world—an internal compass that navigates the interplay between intuition, sound, and machine. At its core, this paradigm redefines musical presence, shifting from rigid interpretation to immersive exploration. It's no longer just about playing notes but inhabiting a living, responsive ecosystem—one where deep listening, spontaneity, and collaboration that can reshape what it means to be inside the music.

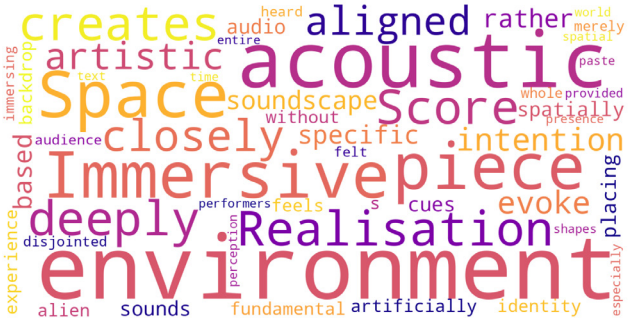


Figure 13. Acoustic field word cloud

3.6 A - Acoustic space

This field focused on the acoustic spaces created by the realisation of the score. The analysis focused on these emerging themes:

3.6.1 Immersive Acoustic Space and the Realisation of the Score

Many works in the corpus were discussed as creating deeply immersive acoustic environments that were closely aligned with their original artistic intention by evoking soundscapes based on a specific environment, rather than just spatially placing sounds or audio cues artificially (e.g. as a backing track). These environments were more than a backdrop, they were fundamental to the piece's identities. The acoustic space was not merely heard but felt as a whole environment, immersing both audience and performers in a world that shapes perception, especially of time and spatial presence. From an Emmersonian perspective, this relates strongly to the "Field" [6] component of the Local/Field-Stage metaphor, where the field refers to the physical and acoustic space that surrounds and envelops performers and listeners. The acoustic space is not a neutral container but an active, lived space that shapes the sound itself and how it is perceived.

3.6.2 Physical and Sensory Immersion for Performers

Several musicians emphasized the sensory and physical experience of being immersed in this space. The musicians mentioned how they felt they were physically immersed in the audiovisual environment, often having to orient themselves spatially - such as having their backs to the audience to focus on a large screen that is central to the performance. This immersion led to some musicians mentioning how it affected their sense of time and agency, fostering a loss of peripheral awareness, focusing attention inward on the sounds, their sensations, and the environment as a whole. For some musicians they described this as embodied and poetic, something difficult to convey in technical

terms but deeply felt as a "floating atmosphere" or "space to immerse into." The acoustic space thus functioned as a physical and psychological environment, influencing both interpretive freedom and the bodily experience of sound production and reception.

3.6.3 Balancing Live and Electronic Elements in Acoustic Space

The acoustic space generated by many digital scores also involved a complex balancing act between live instruments, live-electronics, synthesizers, and the room's natural acoustics. In some works the performers collaborated closely with sound engineers who adapt spatialization and electronic sound balance in real-time according to the room's acoustic response, where the musician's immediate physical environment is shaped and reshaped by real-time interaction with technology and space, and inviting the sound engineer into the co-operative play of musicking.

3.6.4 Acoustic Space and Performer Experience

The acoustic effect varies strongly with the performers' physical positioning in the space. Some performers felt more immersed in particular positions (e.g., standing at the back), where they could better hear spatialized elements of the fixed media track, affecting their musical response. This shows that the physical layout of the space and performer positioning are crucial in staging the acoustic environment, emphasizing the embodied nature of the music-making process within a particular acoustic field.

In case studies where the audience were polled, they mentioned their experiences of the acoustic space as an immersive audiovisual environment, enhanced by large screens and multi-channel speaker arrays. The sense of being "inside" the music is a deliberate compositional and staging strategy, fostering a shared spatial experience despite the complexity of spatialized sound. Meanwhile, the performer's acoustic space tended to be a more intimate and physical experience, shaped by their position, embodied movement, and interaction with technology.

3.6.5 Discussion

In summary, the acoustic space is a complex, immersive environment created through the interplay of, say, electronic spatialization, live acoustic performance, technological mediation, and the physical performance space itself. The acoustic space is not static but dynamic, site-specific, and performer-dependent, shaping perception, time, and bodily engagement with the music. The performers' physical sensations vary widely with different venues and positions, emphasizing the deeply embodied nature of acoustic staging in digital score musicking.

3.7 N - Networks/ Participants

This field focused on those agents perceived to be participating in the music; representations of self and others as a complex network of inter-relationships. The analysis focused on these emerging themes:

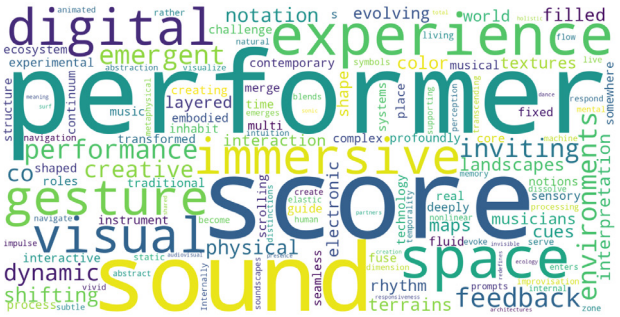


Figure 14. Networks field word cloud

3.7.1 Musicking as a Network of Relationships

At the heart of many digital scores in the corpus lies a dynamic and shifting web of relationships between human and non-human agents, including performers, scores, sounds, visuals, technologies, environments, and audiences. These actants co-create a musicking network that continually evolves through interaction. Musicking in these contexts is never solitary; even when alone, the performer is entangled with pre-recorded tracks, algorithms, scores, or responsive technologies.

This relational field is not only a backdrop but the essence of these artistic works. Musicians frequently mentioned how they were being drawn *into* environments - digital, sonic, or visual - within which they encountered agents like pre-recorded materials, visual stimuli, remote collaborators, and artificial intelligences. These agents were not passive; they provoked, suggested, and shaped the performers' choices. For example, one performer described how the digital score acted like a game, pulling them into a world populated with non-player characters (NPCs) and sonic behaviors derived from game environments. This sense of being "dragged in" highlights the agency of the score as an actant within the musicking network.

3.7.2 Self and Other in Constant Reconfiguration

The boundary between self and other was porous in some scores. Musicians often referred to their relationship with their instruments or sound-producing entities as one of *dialogue*, *co-creation*, or *alignment*. Similarly, musicians mentioned how background tracks were not just reference points but as companions or structures that afford security, within which they can still navigate freely. Overall, the network of participants mentioned included fellow musicians (sometimes remote), the audience, electronics, and environmental factors such as room acoustics or even costume design. These elements were not merely supporting features - they influenced timing, interpretation, and emotional connection. In one instance, the performer felt fundamentally different when positioned closer to speakers, noting that the spatial arrangement dictated which sounds were perceptible and, therefore, which musical relationships could emerge.

3.7.3 Cooperative Agency and Mutual Responsiveness

A central theme is that musicians felt they were not the sole authors of musical meaning within a network characterized by fluid agency. In some examples, musicians mentioned live electronic, spatialization systems, and algorithmic scores often responded to the input of the performer and then provided new material in return. In these systems, such as *Solaris* by Craig Vear and Fabrizio Poltronieri, causality is recursive: the performer influences the sound, which in turn affects the next action of the score. One participant described this as being 'immersed in the situation', where freedom and responsibility coexist. Another reflected on a synthesizer response that altered their playing, noting the feeling of a "trialogue" between themselves, another musician, and the system. This triadic (and sometimes more complex) web of relationships often leads to nonlinear communication. Musicians felt they were constantly balancing between self-listening, listening to others, and being attuned to technological feedback. As such, the act of musicking becomes a networked cognition where each element pushes and pulls on others, reconfiguring identity and role in real-time.

3.7.4 Score as Environment and Actor

Across all the corpus the digital score operated as more than instruction, and in many cases it was often felt as a participant or even a world in which the performer operates. Some scores loaded random elements or contained algorithmic variability, making each iteration slightly different. This gives the performer a sense of navigating through a landscape, rather than executing fixed instructions. The visual aspects of some digital scores - dual colour schemes, animated symbols, or environmental imagery - often served as a active and agential stimuli for emotional and musical responses, while also coordinating interactions between players. In ensemble contexts, communal coherence frequently superseded individual virtuosity. Whether through holding hands during choral sections, echoing each other's phrasing, or adapting to subtle changes in the environment, performers consistently prioritized relational sound-making. They reflected on the emotional depth of being "with" each other - even when not physically present - as when remote performers contributed via video or audio streams, creating a poignant sense of distributed presence.

3.7.5 Non-Human and Hybrid Agents

Some systems involved artificial intelligence, robotic arms, or generative animated environments. These were not just tools - they were treated as co-performers. A robotic arm that responded through motion and notation was said to foster connection and even camaraderie. One participant remarked, "We never felt we were playing alone… we were playing with it." This extension of agency to non-human actors redefines the performance space as post-human, where emotional resonance and co-creativity extend beyond the human participants. Furthermore, the materiality of instruments and objects also played a role. Tactile transducers, contact microphones, and altered acoustic properties became actants in the musicking web. These objects carried affordances - suggestions for how they might be played or heard - shaping the performer's engagement.

3.7.6 Discussion: Musicking as Entangled Practice

What emerged from these accounts is a vision of musicking not as solitary expression or hierarchical ensemble, but as entangled, relational, and ecological. The performer is situated within a mesh of influences - visual, sonic, emotional, spatial, technological - and becomes both a respondent and a provocateur. Each note played, each action taken, shifts the shape of the network. This entangled network includes not only the performers and their tools, but also audiences, spaces, scores, and non-human agents. It is this web - dynamic, unstable, and rich in potential - that could be seen to define the artistic practice. Through navigating this complexity, performers do more than make music - they participate in an evolving system of co-existence, where the boundaries between self, other, and environment blur into one collective act of musicking.

The individual nature of these connections and actants form the material of the other fields in the PACMMAN framework. For example, a shape, colour, sound can have a response through the physical sense, or emit behaviours as an agential actant, or contain a meta narrative.

4. EVALUATION

The digital score corpus, examined through a multidimensional sensory lens, transforms our understanding of musical notation into an interactive, embodied, and visually rich experience. This perspective challenges traditional notions of the score as a static, prescriptive document, instead positioning it as a living, evolving form of music-making where meaning emerges through communal interpretation, action, and interplay. The following discussion synthesizes the key insights from this exploration through the PACMMAN framework, expanding upon the central themes that redefine musical meaning in digital and experimental performance contexts.

The key Aspects of the digital score corpus as a dynamic, co-creative system are:

1. The Score as an Interactive, Responsive System The digital score is no longer a fixed set of instructions but a dynamic, algorithmic entity that reacts to performers' input, reshaping itself through processes such as echoing, altering, or rejecting musical material. Performers engage in a feedback loop with the score, interpreting its signals while simultaneously influencing its behaviour, fostering a collaborative meaning-making process. The score's perceived "intelligence" (through generative algorithms or interactive visuals) creates a sense of co-presence, where the system acts as a sentient collaborator rather than inert notation.

2. Embodiment and Phenomenological Engagement Performers experience the digital score not just visually or sonically but phenomenologically - as an active, responsive presence in the performance space. Gestural interaction, motion tracking, and robotic extensions integrate the score into the physical domain, blurring the boundaries between performer and system. Spatialized projections (on

bodies, instruments, or environments) enhance proprioceptive awareness, making gesture and movement central to musical decision-making.

3. Multisensory Meaning-Making The digital score engages multiple senses - sight, touch, sound, motion, and notation - each offering a platform for exploration rather than a fixed meaning. Visual elements (animations, graphics) function as performative languages, conveying behavioural cues that shape musical responses. Haptic feedback and interactive interfaces deepen the performer's tactile relationship with the score, reinforcing its materiality.

4. Liveness and the Reanimation of Fixed Media Prerecorded sound is recontextualized as interactive material, modulated in real-time through digital signal processing or performer gestures. Fixed audio becomes dynamic, responding to live input and fostering a sense of dialogue between the performer and the system. Performers develop an embodied familiarity with these sonic responses, anticipating and reacting to them as if engaging with an improvisational partner.

5. The Composer as System Designer The composer's role shifts from an authoritative author to a curator of interactive systems, structuring randomness through algorithms, conditional rules, or generative processes. Works may incorporate human override mechanisms (e.g., "Time Out" gestures) or probabilistic structures, allowing for emergent, performer-driven outcomes. The compositional focus expands beyond *what* is played to *how* choices are enacted, emphasizing meta-performance and emergent dramaturgy.

6. Improvisation Within Structured Frameworks The music thrives on heterophonic fluidity, balancing improvisational freedom with pre-set structural elements. Performers navigate mutable score structures, responding intuitively while maintaining coherence within an open-ended framework. This approach rejects rigid control in favour of collaborative, emergent creation, reflecting a cultural ethos that values adaptability and collective interpretation.

7. Interdisciplinary and Multimedia Integration Digital scores dissolve traditional disciplinary boundaries, merging graphic notation, interactive visuals, movement, and sound into unified multimedia experiences. Technology, body, image, and sound intermingle, fostering richer modes of storytelling and engagement. This hybridity reflects a world-view where artistic media are porous and interdependent, enabling new forms of sensory and cognitive alignment.

8. Socio-Political and Cultural Resonances Themes of censorship, marginalization, and resistance emerge metaphorically in interactive systems (e.g., performers reacting to "censoring lights"). The work amplifies marginalized voices - refugees, women, neurodiverse individuals - positioning artistic creation as a site of empowerment. Larger sociopolitical concerns such as, migration (human and avian) and ecological awareness inform the music's conceptual framework, reflecting contemporary concerns about displacement and belonging.

9. Neurodiversity and Inclusivity The digital score accommodates neurodiverse modes of musical experience, including autistic and d/Deaf perspectives. "Digital Syzygy" describes new forms of cognitive and sensory alignment facilitated by technology, challenging normative models of musical cognition. The digital score can break down barriers of access to musicking such as *JoyInst* and *Jess+*. This inclusivity expands definitions of creativity, demonstrating how digital tools can foster novel expressive possibilities beyond conventional sensory boundaries.

10. Role-Playing Games (RPGs) as a Structural Metaphor Performers are framed as explorers in dynamic, rule-bound worlds, emphasizing play, adaptation, and identity negotiation. The RPG model aligns with contemporary digital cultures that value agency, multiplicity, and iterative discovery. This approach encourages experimentation over fixed outcomes, reinforcing the fluid and emergent nature of the music.

11. Multiplicity, Fluidity, and Process-Oriented Art The music embraces radically different interpretations, reflecting postmodern values of plurality and flux. There is no "definitive" version of a work; meaning is co-created in real time by performers, audiences, and environments. Art is treated as a living, evolving process rather than a finalized product, privileging openness and transformation.

12. The Acoustic Space as a Dynamic, Embodied Environment The performance space is not static but shaped by electronic spatialization, venue acoustics, and performer interaction. The Local/Field-Stage metaphor [9] applies, where the physical environment of the performers merges with the theatrical experience of the audience. Performers' bodily sensations vary with different venues, emphasizing the deeply embodied nature of digital score musicking.

5. CONCLUSION: TOWARD AN ENTANGLED, ECOLOGICAL MODEL OF MUSICKING

What emerged from these case studies is a vision of digital score musicking as an entangled, relational, and ecological practice. The musicians exist within a dynamic network of influences - visual, sonic, technological, spatial - acting as both respondent and provocateur. Each musical gesture shifts the system's trajectory, reinforcing the idea that music-making is a collective act of negotiation rather than individual expression.

This model extends beyond human musicians to include non-human agents (algorithms, interactive systems, spatial acoustics), creating a web of co-creation where boundaries between performer, audience, and environment dissolve. The digital score, in this context, is not merely a tool but an active participant in an evolving musical ecosystem.

Ultimately, this paradigm fosters a new mode of musical presence - one defined by exploration, deep listening, and co-creative engagement. It challenges conventional hierarchies and pedagogies, embraces multiplicity, and redefines what it means to "be" in music (extending Small's notion of musicking into a digital musicking realm). As digital and experimental practices continue to evolve, this multidimensional, interactive approach offers a transformative framework for reimagining the future of musical notation, performance, meaning-making, and music education.

Acknowledgments

The Digital Score project (DigiScore) is funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. ERC-2020-COG – 101002086). The author would also like to thank the amazing research assistants who helped with this research: Dr Solomiya Moroz, Dr Fabrizio Poltronieri, Macharia Kelvin, Ben Hopkinson, Oluwademilade Oshin, Dr. Adrian Hazzard, Dr. Johann Benerradi, Elsa Kitching, Dr. Prashanth Thattai, Dr. Oliver Miles. And the project partners Cat, Ken, Sandeep, and Li and Nancy.

6. REFERENCES

[1] C. Vear, *The digital score: Musicianship, creativity and innovation*. Routledge, 2019.

[2] ——. (2025) The digital score. [Online]. Available: https://digiscore.github.io/

[3] M. McLuhan, "The medium is the message," in *Communication theory*. Routledge, 2017, pp. 390–402.

[4] C. Vear and S. Moroz, "Diving deep with nautilus: An analysis of musicking with a digital score," *Journal of Creative Music Systems*, vol. 8, no. 1, pp. 1–19, 2024.

[5] C. Small, *Musicking: The meanings of performing and listening*. Wesleyan University Press, 1998.

[6] S. Emmerson, *Living electronic music*. Routledge, 2017.

[7] D. Borgo, *Sync or swarm: Improvising music in a complex age*. A&C Black, 2005.

[8] M. Csikszentmihalyi and M. Csikzentmihaly, *Flow: The psychology of optimal experience*. Harper & Row New York, 1990, vol. 1990.

[9] S. Emmerson, "Aural landscape: musical space," *Organised Sound*, vol. 3, no. 2, pp. 135–140, 1998.

SYNTACTIC COMPLEXITY OF MUSIC WITH LAMBEK CALCULUS

Matteo Bizzarri
Scuola Normale Superiore
matteo.bizzarri.net

Satoshi Tojo
Asia University
tojo_satoshi@asia-u.ac.jp

ABSTRACT

Although music and language differ in many respects, both exhibit hierarchical structure — notably between harmonic progressions and syntactic constructions. In this paper, we apply Lambek Calculus, a Gentzen-style sequent calculus used in categorial grammar, to the analysis of chord sequences. We first show how Lambek Calculus can model harmonic relationships in music through the lens of syntax and logical semantics, where the structure of a proof parallels that of a syntactic tree. Next, to demonstrate the feasibility of this approach, we assess the harmonic complexity of music by examining the proof tree: specifically, by counting both the number of tonal modulations and the depth of the derivation. This method allows us to explore whether intuitively optimal harmonic analyses correspond to minimal proofs within the system.

1. INTRODUCTION

Music and language are often said to share a common origin [1, 2], in that music could have been a primitive communication method of old human species. Accordingly, many music researches have been made with the linguistic theory. The generative grammar is a representative of such theory, where a finite set of generative rules decides a set of sentences, and in the similar way, generative rules have been considered for the sequence of harmonic chords [3–6]. In these days many approaches employ grammar-based systems as well as statistical parsing with annotated corpora [7–9], whereas the generative power of formal grammar in music has not been fully elucidated.

In this study, we pursue the theoretical aspect of grammar in music. In order to clarify the issue, we employ the categorial grammar, which is translated directly into a formal proof system: the sequent calculus [10], instantiated as Lambek Calculus [11]. Thus far, jazz-chord sequence has been analyzed with the categorial grammar [12], however, they have not mentioned its logical aspect. We develop the framework of categorial grammar to Lambek Calculus, and enhances both interpretability and transparency in the analytical process [13–16].

Lambek Calculus enforces the directionality and the linear order by disallowing structural rules such as exchange, contraction, or weakening. This reflects the sequential and non-commutative nature of harmonic progressions, where the order of chords is essential and repetitions or permutations are not freely admissible. Moreover, Lambek derivations are explicit proofs: each analysis corresponds to a well-defined logical object, whose structure is systematically studied and is not readily available in the traditional context-free grammar frameworks.

The calculus therefore provides a logically grounded, compositional account of chord sequences, with the additional advantage of being extensible through modal operators to represent modulations and tonal accessibility. While this study focuses on tonal music, the same formalism can be extended to modal and non-tonal systems — an aspect we aim to develop further.

To demonstrate the feasibility of this approach, we propose a measure of harmonic complexity based on the structure of the proof tree: specifically, the number of tonal modulations and the derivational *depth* of the proof. This allows us to introduce quantitative measures such as minimality and ambiguity. We argue that these metrics correlate with intuitively optimal analyses.

The paper is structured as follows: Section 2 introduces the basics of categorial grammar and Lambek Calculus, together with Modal Logic. Section 3 defines modal operators for tonal accessibility. In Section 4, we show an analytical example of *In Your Own Sweet Way* with our method. In Section 5, we further develop the advantage of proof theory, to measure the complexity of chord progression. Section 6 applies the proof depth to the jazz standard *All the Things You Are*. Section 7 discusses implications and outlines directions for future research.

2. GRAMMAR AND CALCULUS

In this section, we provide fundamental theories of language and logic.

2.1 Categorial Grammar

Categorial grammar (CG) [17] consists of a set of categories, assigned to each lexical word. A category is constructed recursively from a given basic categories, with the following $(/)$ and $(\backslash)$; let X and Y be categories, then [1]

$$Y/X : \text{biting } X \text{ from the right-hand side, produces } Y,$$
$$Y\backslash X : \text{biting } Y \text{ from the left-hand side, produces } X.$$

With these categorial construction, we can argue if a given sequence of categories is *grammatical*, dependent on if we

[1] Note that $(/)$ does not have any musical meaning tipically used in chord notation.

Copyright: ©2025 Matteo Bizzarri et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

can finally obtain sentence category S, e.g.,

$$Y,\ Y\backslash(S/X),\ X \Longrightarrow S/X,\ X \Longrightarrow S.$$

A human language is said to belong approximately to the class of context-free grammar (CFG) in the Chomsky hierarchy [18] though there are known exceptions. [2] The generative power of CG is equivalent to CFG, shown as follows. A set of production rules of CFG can be rewritten in Chomsky normal form [19], each of which produces two non-terminal symbols (Y, Z below) or one terminal symbol (w) from a non-terminal symbol (X) as below

$$X \to Y\ Z, \quad X \to w,$$

and the branching in a syntactic tree becomes always binary except for terminal symbols at the leaves. The nodes in the tree are non-terminal symbols of CFG, which correspond to categories in CG. We can replace each CFG rule $X \to Y\ Z$ either for $Z = Y\backslash X$ or for $Y = X/Z$, and the resulting set of categories becomes equivalent to CFG.

For example, a verb phrase (VP) bites a subject noun phrase (NP) from left to be S, VP = NP\S. Also, a determiner (Det) bites a noun from right to be NP, Det = NP/N. Thus, we can compose a sentence as follows.

A	bird	
NP/N	N	flies
NP		NP\S
S		

2.2 Lambek Calculus

We now translate the categorial constructions into a Gentzen-style sequent calculus. The turnstile symbol ($\vdash$) is read as "derives," and commas (,) on the left-hand side denote logical conjunctions ($\wedge$). In Lambek Calculus, the left-hand side of ($\vdash$) is specifically an *ordered* sequence of categories, and in accordance with intuitionistic logic, the right-hand side of ($\vdash$) is restricted to a single term [20].

We employ lowercase Latin letters ($x, y, z, \dots$) to denote individual formulas, while uppercase Greek letters (e.g., Γ, Δ, Σ) represent sequences (or multisets) of formulas on the left-hand side of a sequent. Figure 1 summarizes the inference rules used to construct derivations. The expression y/x denotes a chord y that expects chord x to its right—i.e., y precedes x in the sequence. Conversely, $x\backslash y$ has the same interpretation but with reversed directionality: x expects y to its left. These rules can be applied either on the left side of the sequent ($\backslash_L$, $/_L$) or on the right ($\backslash_R$, $/_R$).

The *Cut* rule is a fundamental principle in proof theory, enabling compositional reasoning across intermediate formulas. However, in this paper, it is included for completeness but never invoked in actual derivations.

For example, consider the sentence "A bird flies." In categorial grammar, this can be expressed as a derivation using NP/N, N, and $NP\backslash S$. This construction is translated into the sequent calculus, providing horizontal lines step-by-step, as follows:

[2] For example, agreements in Romance languages or verb order in clauses in Dutch are context sensitive.

$$\frac{\Delta, y, \Sigma \vdash z \qquad \Gamma \vdash x}{\Delta, y/x, \Gamma, \Sigma \vdash z}\ (/_L)$$

$$\frac{\Gamma \vdash x \qquad \Delta, y, \Sigma \vdash z}{\Delta, \Gamma, x\backslash y, \Sigma \vdash z}\ (\backslash_L)$$

$$\frac{\Gamma, x \vdash y}{\Gamma \vdash y/x}\ (/_R) \qquad \frac{x, \Gamma \vdash y}{\Gamma \vdash x\backslash y}\ (\backslash_R)$$

$$\frac{\Gamma \vdash x \qquad x \vdash y}{\Gamma \vdash y}\ \text{(Cut)} \qquad x \vdash x\ \text{(Init)}$$

Figure 1. Set of rules of Lambek Calclus; we exclude ($\cdot$) rules.

$$\frac{\dfrac{\text{NP} \vdash \text{NP} \qquad \text{N} \vdash \text{N}}{\text{NP/N, N} \vdash \text{NP}}\ (/_L) \qquad \text{S} \vdash \text{S}}{\text{NP/N, N, NP}\backslash\text{S} \vdash \text{S}}\ (\backslash_L)$$

In this example, each category is introduced by an axiom and combined through inference rules, forming a syntactic proof that corresponds to the grammatical derivation.

2.3 Modal Logic

Modality is variously interpreted in music [21], and especially modal extension of Lambek Calculus [22] is useful to capture modulation in music, so we introduce a modal operator '$\Box$' by modal logic [23].

Modal logic is classically interpreted via Kripke semantics, which assumes a set of possible worlds connected by an accessibility relation R. The valuation (true or false) of a proposition depends on each world. A proposition P holds in a world w is written as $w \Vdash P$, and $\Box P$ is interpreted as:

$$w \Vdash \Box P \iff \forall w' \left({}_wR_{w'} \right) w' \Vdash P$$

where ${}_wR_{w'}$ states that w' is accessible from w. Namely, $\Box P$ holds in w if P holds in all the accessible worlds w' from w. Usual modal logic provides also $\Diamond$-operator as the dual of '$\Box$' as $\Diamond = \neg\Box\neg$, however, since Lambek Calculus does not possess negation ($\neg$) we do not include '$\Diamond$' in our system.

The '$\Box$' is introduced by the following axiom (K) in sequent calculus.

$$\frac{\Gamma \vdash x}{\Box\Gamma \vdash \Box x}\ \text{(K)}$$

In linguistics, $\Box$-operator can represent hypothetical contexts (e.g., conditionals or beliefs). A clause that has type S may be embedded as '$\Box S$' in another sentence, meaning it is evaluated in a different context. In our framework, we treat musical keys as possible worlds and employ '$\Box$' to refer to another key, as is shown in the following section.

3. LAMBEK CALCULUS IN TONAL PITCH SPACE

Throughout the paper, chord symbols appear in italics (e.g., Cm, $G7$), while key names are upright. Major keys are in

uppercase (e.g., C), minor keys in lowercase (e.g., a). Roman numerals indicate scale degrees: uppercase for major (I, II, ...) and lowercase for minor (i, ii, ...). The slash is equivocally used in music, as C/G, I/C, or V/V, meaning the base note in chord symbol, the degree in a key, or doppel dominant, respectively. In this paper, The slash symbol (/) is used only as a right-associative function in categorial grammar, avoiding ambiguous music notations.

3.1 Multiple accessibility

Given a set of diatonic notes, or a chord symbol, we can assign a degree within a key in different ways. We regard the key is a possible world in Kripke semantics and write $\varphi \Vdash x(c)$, where φ is a key, x is a degree, and c is a set of notes. For example, a Berklee chord symbol C is I in C major, IV in G major, V in F major, and so on.

$$\mathrm{C} \Vdash \mathrm{I}(C) \iff \mathrm{G} \Vdash \mathrm{IV}(C) \iff \mathrm{F} \Vdash \mathrm{V}(C).$$

This means that we may interpret a chord in a modulated context.

From now on, we employ white bold letters for modulation to related keys.

- $\mathbb{D}$ denotes modulation to the dominant key,
- $\mathbb{S}$ denotes modulation to the subdominant key,
- $\mathbb{R}$ denotes modulation to the relative major or minor,
- $\mathbb{P}$ denotes modulation to the parallel key (same tonic, opposite mode).

Figure 2 illustrates the map of related keys of western tonal music, based on the Tonal Pitch Space (TPS) [24]. [3] Thus, we introduce *multiple* accessible relations in our modal logic. We use $R_{\mathbb{D}}$, $R_{\mathbb{S}}$, $R_{\mathbb{P}}$, and $R_{\mathbb{R}}$ to represent the accesses to the above related key, respectively.

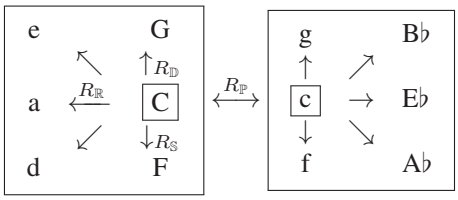


Figure 2. Accessible keys in the Tonal Pitch Space

In accordance with the four accessibility, we may introduce four modal operators $\{\Box_{\mathbb{D}}, \Box_{\mathbb{S}}, \Box_{\mathbb{R}}, \Box_{\mathbb{P}}\}$, however, we directly use $\{\mathbb{D}, \mathbb{S}, \mathbb{R}, \mathbb{P}\}$ for modal shifts for visibility, and '$\Box$' is employed only when it represents one of these modal shifts.

According to the conventional notation of modal logic, we should write

$$\varphi \Vdash \Box x(c) \iff R\varphi \Vdash x(c),$$

however in our case, since the access relation between tonalities is one-to-one, there exists such $\Box'$ that

$$\varphi \Vdash x(c) \iff R\varphi \Vdash \Box' x(c).$$

[3] Similar ideas to the TPS are also found in [25–27].

Therefore, we may attach the modal operator on the proposition in the destination world, for easy comprehension. For example, since G major is the dominant key of C major, we may write the equivalent formula, as follows.

$$\mathrm{C} \Vdash \mathrm{I}(C) \iff R_{\mathbb{D}}\mathrm{C} \Vdash (\mathbb{D}\mathrm{I})(C).$$

Later in Section 3.4, we will develop the sequent calculus with these modalities: let Γ be a sequence of chord degrees, and x be a chord degree. Then, axiom (K) is written, for example, as

$$\frac{\Gamma \vdash x}{\mathbb{D}\Gamma \vdash \mathbb{D}x}\ (\mathrm{K})$$

and so on.

Note that $\mathbb{D}, \mathbb{S}, \mathbb{P}$, and $\mathbb{R}$ distinguish scale degrees, while $R_{\mathbb{D}}, R_{\mathbb{S}}, R_{\mathbb{P}}$ and $R_{\mathbb{R}}$ refer to modulations between keys.

These modal annotations enrich the expressive power of Lambek derivations. Rather than flattening all harmonic functions into a single key, we enable layered reasoning across multiple tonal centers. This is especially useful in jazz, where temporary tonicizations and frequent modulations are structurally significant. We also adopt a simplification principle: when multiple derivations exist, the one with the fewest modal transitions (or the shallowest modal nesting) is preferred. This principle of minimality complements the proof depth metric introduced in the next section.

3.2 Relation Calculi

We summarize the relations among modalities in Table 1, where R represents key-to-key accessibility and φ denotes a key.

$$R_{\mathbb{D}}R_{\mathbb{S}}\varphi = R_{\mathbb{S}}R_{\mathbb{D}}\varphi = \varphi \quad (1)$$

$$R_{\mathbb{D}}^{12}\varphi = R_{\mathbb{S}}^{12}\varphi = \varphi \quad (2)$$

$$R_{\mathbb{R}}^{2}\varphi = \varphi,\ R_{\mathbb{P}}^{2}\varphi = \varphi \quad (3)$$

$$\begin{cases} R_{\mathbb{D}}R_{\mathbb{P}}\varphi = R_{\mathbb{P}}R_{\mathbb{D}}\varphi \\ R_{\mathbb{S}}R_{\mathbb{P}}\varphi = R_{\mathbb{P}}R_{\mathbb{S}}\varphi \\ R_{\mathbb{D}}R_{\mathbb{R}}\varphi = R_{\mathbb{R}}R_{\mathbb{D}}\varphi \\ R_{\mathbb{R}}R_{\mathbb{S}}\varphi = R_{\mathbb{S}}R_{\mathbb{R}}\varphi \end{cases} \quad (4)$$

Table 1. Relations in accessibility

In Table 1, (1) expresses that the dominant of the subdominant (or vice versa) returns to the original key. Equation (2) shows that applying the dominant or subdominant modulation 12 times brings us back to the starting key, reflecting the circle of fifths in equal temperament. The relations in (3) show that parallel and relative modulations are involutive. Finally, the commutativity relations in (4) describe symmetric transitions between modulations.

In fact, these operators are not all strictly necessary: for example, we could use just one modality to track motion clockwise or counter-clockwise on the circle of fifths. Since $R_{\mathbb{D}}^{-1}\varphi = R_{\mathbb{S}}\varphi$, we can write:

$$R_{\mathbb{D}}^{m}\varphi = R_{\mathbb{S}}^{n}\varphi \quad \text{where } m + n \equiv 0 \mod 12.$$

Similarly, one can express a bridge between the major and minor circles with:

$$R_{\mathbb{P}}\varphi = R_{\mathbb{S}}^{3} R_{\mathbb{R}}\varphi.$$

Nevertheless, for the sake of musical clarity and interpretability, we retain all four modal operators explicitly in our system.

3.3 Degree Calculi

The key modulations are applied here to degree calculation. We regard the degrees of chords, as well as those headed by the modal operators of $\{\mathbb{D}, \mathbb{S}, \mathbb{P}, \mathbb{R}\}$, as predicates for a set of notes in a key, that is a possible world.

In general, the x-th degree in key φ is equivalent to the $(x+3)$-rd degree in $R_{\mathbb{D}}\varphi$:

$$\varphi \Vdash x(c) \iff R_{\mathbb{D}}\varphi \Vdash (\mathbb{D}x)(c) \ \ (\mathbb{D}x = x + 3 \mod 7)$$

where φ: a key, x: a Roman numeral, and c: a chord symbol.

In addition, we need to distinguish upper-case degree numerals (major chords) from lower-case ones (minor chords). The full calculations are shown in Table 2. The first three can be considered the main ones, whereas the fourth is derived, since the Subdominant function can be derived from the Dominant function, and the Relative function can be derived from the Parallel one. Here, M represents a major key, and m a minor key.

$\mathbb{P}$	key, deg_M	=	key, deg_m
$\mathbb{D}$	key, deg_M	=	$\mathbb{D}\{key\}, \{deg + 3 \mod 7\}_M$
$\mathbb{D}$	key, deg_m	=	$\mathbb{D}\{key\}, \{deg + 3 \mod 7\}_m$
$\mathbb{S}$	key, deg_M	=	$\mathbb{S}\{key\}, \{deg + 4 \mod 7\}_M$
$\mathbb{S}$	key, deg_m	=	$\mathbb{S}\{key\}, \{deg + 4 \mod 7\}_m$
$\mathbb{R}$	key, deg_M	=	$\mathbb{R}\{key\}, \{deg + 5 \mod 7\}_m$
$\mathbb{R}$	key, deg_m	=	$\mathbb{R}\{key\}, \{deg + 2 \mod 7\}_M$

Table 2. Summarizing table for functions.

3.4 Tagged Sequent Calculus

We have now introduced the essential components of our formal system. In the Lambek Calculus notation, Γ, Δ, and Σ represent sequences harmonic contexts, while z stands for another chord formula in our system. We formalize modulations as transitions between possible tonal worlds, using modal operators to encode harmonic motion across key regions.

The (Init) rule represents the initial axiom or base case of a derivation, typically corresponding to the harmonic interpretation of a chord within a given tonality. For example, G $\Vdash$ IV(C) is shown as

$$\frac{}{\text{G: IV} \vdash \text{IV}}\ C$$

where we omit a horizontal line, to distinguish these initial introductions from other logical derivations.

In front of the sequents we proceed to use a key with colon (:) for mnemonic annotations, and call *Tagged sequents*, to remember the key in which we are doing our analysis. To avoid confusion, throughout this paper, we will always indicate the function of the key ($R_{\mathbb{D}}$, $R_{\mathbb{S}}$, $R_{\mathbb{P}}$, and $R_{\mathbb{R}}$) alongside the keys, even when they can be interpreted as belonging to a new tonal region. For instance, $R_{\mathbb{D}}$G corresponds to D, while $R_{\mathbb{D}}R_{\mathbb{R}}$d corresponds to C major. The key is valid for the entire line, not just for the subsequent sequent, and we will not repeat it if it is the same as the key in the previous line.

Example 1. Consider the example of $D7$–$G7$–$C_{\text{MA}}{}^7$. When we interpret the sequence in C major, we obtain:

$$\dfrac{\dfrac{\begin{matrix}D7 & G7\\ \text{C: II} \vdash \text{II} & \text{V} \vdash \text{V}\end{matrix}}{\text{II}, \text{II}\backslash\text{V} \vdash \text{V}}\,(\backslash_L) \qquad \begin{matrix}C_{\text{MA}}{}^7\\ \text{I} \vdash \text{I}\end{matrix}}{\text{II}, \text{II}\backslash\text{V}, \text{V}\backslash\text{I} \vdash \text{I}}\,(\backslash_L)$$

The final sequent i.e., II, II\V, V\I $\vdash$ I, can be interpreted as follows: in the tonality of C, we move from the second degree to the fifth, then to the first, which is also the last chord of the sequent, as can be seen on the right side.

Example 2. The flexibility of the system also enables the analyst to describe this type of tree in a different way, emphasizing the function of the *Doppel-dominant* as follows.

$$\dfrac{\dfrac{\dfrac{\dfrac{\begin{matrix}D7 & G7\\ \text{G: V} \vdash \text{V} & \text{G: I} \vdash \text{I}\end{matrix}}{\text{V}, \text{V}\backslash\text{I} \vdash \text{I}}\,(\backslash_L)}{R_{\mathbb{S}}\text{G: } \mathbb{S}\{\text{V}, \text{V}\backslash\text{I}\} \vdash \mathbb{S}\text{I}}\,(\text{K}_{\mathbb{S}})}{\mathbb{S}\{\text{V}, \text{V}\backslash\text{I}\} \vdash \text{V}} \qquad \begin{matrix}C_{\text{MA}}{}^7\\ \text{I} \vdash \text{I}\end{matrix}}{R_{\mathbb{S}}\text{G: } \mathbb{S}\{\text{V}, \text{V}\backslash\text{I}\}, \text{V}\backslash\text{I} \vdash \text{I}}\,(\backslash_L)$$

In this case, the tagged sequents are rewritten in different tonalities because they change.

In some cases, modulations cannot be restricted to those with single operators; complex operators are also admissible.

Example 3. For example, $A7^{alt}$–$Dm7$– $G7$ is analyzed as follows.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\begin{matrix}A7^{alt} & Dm7\\ \text{d: V} \vdash \text{V} & \text{i} \vdash \text{i}\end{matrix}}{\text{V}, \text{V}\backslash\text{i} \vdash \text{i}}\,(\backslash_L)}{R_{\mathbb{R}}\text{d: } \mathbb{R}\{\text{V}, \text{V}\backslash\text{i}\} \vdash \mathbb{R}\text{i}}\,(\text{K}_{\mathbb{R}})}{R_{\mathbb{D}}R_{\mathbb{R}}\text{d: } \mathbb{D}\mathbb{R}\{\text{V}, \text{V}\backslash\text{i}\} \vdash \mathbb{D}\mathbb{R}\text{i}}\,(\text{K}_{\mathbb{D}})}{\text{VI}, \text{VI}\backslash\text{ii} \vdash \text{ii}} \qquad \begin{matrix}G7\\ \text{V} \vdash \text{V}\end{matrix}}{\text{VI}, (\text{VI}\backslash\text{ii})/\text{V}, \text{V} \vdash \text{ii}}\,(/_L)$$

The sequent VI, (VI\ii)/V, V $\vdash$ ii expresses that, in the key of C (i.e., the tonal center reached via the tagged sequent $R_{\mathbb{D}}R_{\mathbb{R}}d$), we first interpret a chord as the sixth degree, then move through a cadential progression that brings us to the second minor degree (ii), and finally to the fifth degree (V). This sequence reflects a typical jazz turnaround embedded in a modulated tonal context.

4. ANALYSIS (1) – *IN YOUR OWN SWEET WAY*

Now, we present an example analysis of the first eight bars of *In Your Own Sweet Way* by Dave Brubeck shown in Figure 3, and in Figure 6. To provide a clearer understanding, the main points are as follows:

- In the first tree, we track key shifts using shift modalities, namely, $\mathbb{R}$ (relative), $\mathbb{D}$ (dominant), and $\mathbb{S}$ (subdominant).
- Moreover, during the analysis, it is necessary to apply (1) to simplify $\mathbb{SDR}$ into $\mathbb{R}$. Indeed, it is easy to verify that g (G minor) is the relative key of B♭.
- The second tree presents an alternative interpretation of the same analysis. The final result remains unchanged, but a notable difference is that the cadence $Cm7$–$F7$–$B\flat^6$ is analyzed separately. Then, using the rule $(\backslash_L)$, we obtain the same result as in the first tree.
- The third tree (bottom left in Figure 6) is separate from the first one because it features a brief but significant key shift, which is better represented independently.
- The last tree (bottom right in Figure 6) illustrates a *ii–V–I* progression concluding the A-section of the piece. It is presented as a separate tree because it belongs to a different tonal region.

The final result will be a sequence of the three distinct regions of B♭ – G♭ – B♭.

5. DEPTH OF A PROOF

Now that the basic system is set, it is possible to introduce the notion of the depth of a proof in a way similar to how it is defined in Logic:

Definition 1 (Depth of a proof)**.** The depth of a proof is the maximum level of nesting in a proof, determined according to the rules in Figure 4.

The concept of depth allows us to more precisely determine how directly a proof represents the harmonic analysis of a given sequence of chords. A lower depth indicates a more immediate derivation, while a higher depth may result from additional structural elements, mainly modal operators, that clarify underlying harmonic relationships. Furthermore, the same set of chords can be analyzed in multiple ways, leading to different depth values, as demonstrated in Example 4. This highlights both the flexibility of the system in capturing different levels of harmonic interpretation and the shift in perspective that the analyst can choose to emphasize. In Figure 4 all the rules are described. The idea is that the depth increases whenever it is necessary to do an operation on the proposition, e.g., to write a cadence or to change the key.

Is is also possible to define what is *Minimality* in this context:

Definition 2 (Minimality of a proof)**.** A proof is *minimal* when its depth is lowest among all possible analyses of a sequence of chords.

Even though we cannot ensure that a given proof is minimal or not, it is possible to define the minimal proof as such that does not employ the axiom (K):

Definition 3. The minimality of a proof that does not require modal operators is the number of chords minus 1.

In fact, it is not difficult to see that, based on the rules in Figure 4, if we do not use the modal rules the depth can only increase every when we concatenate two different chords.

Example 4. This example presents the analysis of the chords (transposed to the key of F for practicality, so that the last chord is C7) that make up the B-part of *I Got Rhythm* (A7, D7, G7, C7). As is well known, this section consists of nothing more than a chain of dominants. Using Lambek Calculus, we provide two different analyses. The first one is the most direct: we simply write the degree of the chords in the key of C, resulting in the minimal possible depth of 3. The second analysis, instead, changes the key twice using modal operators ($\mathrm{K}_{\mathbb{S}}$). Although this increases the depth to 5, it makes the harmonic structure clearer, explicitly showing that the chords form a chain of dominants.

$$\dfrac{\dfrac{\dfrac{A7 \qquad D7}{\text{C:VI} \vdash_0 \text{VI} \quad \text{II} \vdash_0 \text{II}}\,(\backslash_L) \qquad G7}{\text{VI}, \text{VI}\backslash\text{II} \vdash_1 \text{II} \qquad \text{V} \vdash_0 \text{V}}\,(\backslash_L) \qquad C7}{\text{VI}, \text{VI}\backslash\text{II}, \text{II}\backslash\text{V} \vdash_2 \text{II} \qquad \text{I} \vdash_0 \text{I}}\,(\backslash_L)$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{A7 \qquad D7}{\text{D:V} \vdash_0 \text{V} \quad \text{I} \vdash_0 \text{I}}\,(\backslash_L)}{\text{V}, \text{V}\backslash\text{I} \vdash_1 \text{I}}\,(\mathrm{K}_{\mathbb{S}})}{R_{\mathbb{S}}\text{D: } \mathbb{S}\{\text{V}, \text{V}\backslash\text{I}\} \vdash_2 \mathbb{S}\text{I}} \qquad G7}{\mathbb{S}\{\text{V}, \text{V}\backslash\text{I}\} \vdash_2 \text{V} \qquad \text{I} \vdash_0 \text{I}}\,(\backslash_L)}{\mathbb{S}\{\text{V}, \text{V}\backslash\text{I}\}, \text{V}\backslash\text{I} \vdash_3 \text{I}}\,(\mathrm{K}_{\mathbb{S}})}{R_{\mathbb{S}}R_{\mathbb{S}}\text{D: } \mathbb{S}\{\mathbb{S}\{\text{V}, \text{V}\backslash\text{I}\}, \text{V}\backslash\text{I}\} \vdash_4 \mathbb{S}\text{I}} \qquad C_{\text{MA}}{}^7}{\mathbb{S}\{\mathbb{S}\{\text{V}, \text{V}\backslash\text{I}\}, \text{V}\backslash\text{I}\} \vdash_4 \text{V} \qquad \text{I} \vdash_0 \text{I}}\,(\backslash_L)}{\mathbb{S}\{\mathbb{S}\{\text{V}, \text{V}\backslash\text{I}\}, \text{V}\backslash\text{I}\}, \text{V}\backslash\text{I} \vdash_5 \text{I}}\,(\backslash_L)$$

Finally we can see another tree with no modal operators that, maintains the same depth although the derivation is a little bit different from the first one that we have seen:

$$\dfrac{\dfrac{\dfrac{A7 \qquad D7}{\text{C:VI} \vdash_0 \text{VI} \quad \text{II} \vdash_0 \text{II}}\,(\backslash_L) \qquad \dfrac{G7 \qquad C7}{\text{V} \vdash_0 \text{V} \quad \text{I} \vdash_0 \text{I}}\,(\backslash_L)}{\text{VI}, \text{VI}\backslash\text{II} \vdash_1 \text{II} \qquad \text{V}, \text{V}\backslash\text{I} \vdash_1 \text{I}}\,(\backslash_L)}{\text{VI}, \text{VI}\backslash\text{II}, \text{II}\backslash\text{V}, \text{V}\backslash\text{I} \vdash_3 \text{I}}\,(\backslash_L)$$

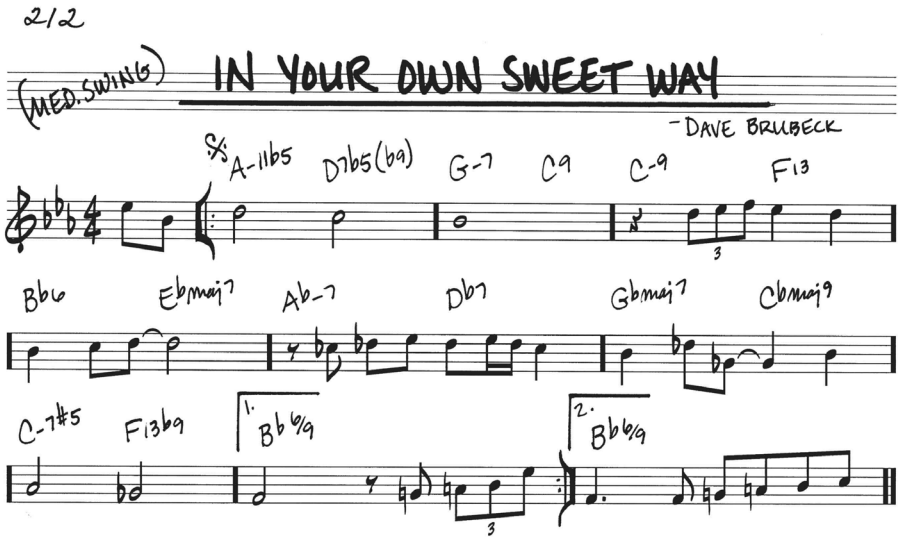


Figure 3. The beginning of *In your own sweet way* by Dave Brubeck

$$\frac{\text{Chord}}{x \vdash_0 x}\ (\text{Init})$$

$$\frac{\Delta, y, \Sigma \vdash_\delta z \qquad \Gamma \vdash_\gamma x}{\Delta, y/x, \Gamma, \Sigma \vdash_{\delta+\gamma+1} z}\ (/_L)$$

$$\frac{\Gamma \vdash_\delta x \qquad \Delta, y, \Sigma \vdash_\gamma z}{\Delta, \Gamma, x\backslash y, \Sigma \vdash_{\delta+\gamma+1} z}\ (\backslash_L)$$

$$\frac{\Gamma, x \vdash_\delta y}{\Gamma \vdash_\delta y/x}\ (/_R) \qquad \frac{x, \Gamma \vdash_\delta y}{\Gamma \vdash_\delta x\backslash y}\ (\backslash_R)$$

$$\frac{\alpha : \Gamma \vdash_\delta \Delta}{R_\Box \alpha : \Box\Gamma \vdash_{\delta+1} \Box\Delta}\ (K)$$

Where:

- $\delta, \gamma \in \mathbb{N}$;
- $K_\mathbb{F}$, $R_\mathbb{F}$ are respectively one of the modal functions applied and one of the modal relations;
- $\Box$ represents again each modality.

Figure 4. Rules of the Lambek Calculus for Chord Analysis with the calculus of the depths.

The idea of using depth arises from the fact that this number allows for comparing two different proofs and finding the one that is both the most explanatory for the analyst and the easiest to compute, thanks to the principle of minimality.

6. ANALYSIS (2) – *ALL THE THINGS YOU ARE*

All the things you are is a well-known song in the jazz repertoire, as was recorded by many celebrated artists, like John Coltrane, Keith Jarrett or Chet Baker (Figure 5). The interesting part of this song is the rapid succession of different keys and we can acquire various analyses thanks to Lambek Calculus in Figure 8. We should pay attention to the following interesting points about these analyses:

- The first point is that the initial tree analyzes a classic vii-ii-V-I progression that resolves to IV, after which there is a change in tonality. We have chosen to emphasize the fact that G7 functions as the VII major degree of $A\flat$, requiring multiple applications of the modal axiom (K) to reconnect the parts to the new key. However, we would also be able to divide the tree into two parts, providing a more specific analysis of the sudden modulation:

$$\dfrac{\dfrac{\begin{matrix}\vdots & \vdots\\ \text{vi}, \text{vi}\backslash\text{ii} \vdash_1 \text{ii} & \text{V}\backslash\text{I} \vdash_1 \text{I}\end{matrix}}{\text{vi}, \text{vi}\backslash\text{ii}, \text{ii}\backslash\text{V}, \text{V}\backslash\text{I} \vdash_3 \text{I}}\ (\backslash_L) \qquad \begin{matrix}\\ D\flat\text{MA}^7\\ \text{IV} \vdash_0 \text{IV}\end{matrix}}{\text{vi}, \text{vi}\backslash\text{ii}, \text{ii}\backslash\text{V}, \text{V}\backslash\text{I}/\text{IV}, \text{IV} \vdash_4 \text{I}}\ (/_L)$$

$$\dfrac{\begin{matrix}G7 & C\text{MA}^7\\ \text{C:V} \vdash_0 \text{V} & \text{I} \vdash_0 \text{I}\end{matrix}}{\text{V}\backslash\text{I} \vdash_1 \text{I}}\ (\backslash_L)$$

In contrast, in the tree in Figure 8, we have grouped everything together for two reasons. First, D$\flat$MA^{7}is
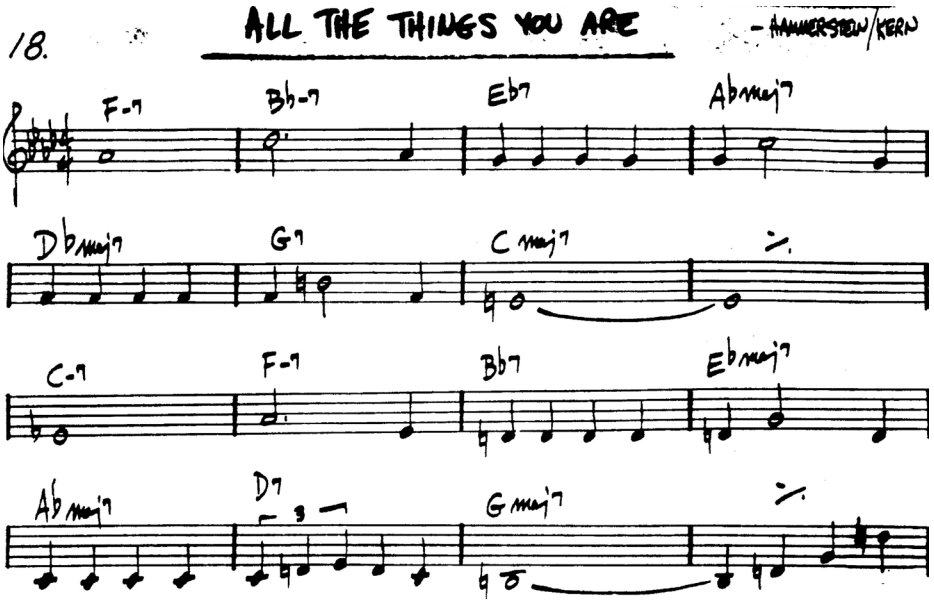


Figure 5. A part of *All the things you are*

closely related to $G7$, as they are separated by only a tritone. Second, the same pattern appears in the second tree. Ultimately, these differences reflect the main purpose of Lambek Calculus: to serve as a tool for the analyst, who can choose the approach that best fits their analyses.

- Another interesting point to note is that the results of the first tree in Figure 8 and the second tree are exactly the same in a different tonality, where they reveal the same harmonic structure transposed.
- Finally, in the last tree, it is possible to see that a similar mechanism is applied to the modulations, with the difference that from $\textsc{Gma}^7$, the modulation is to $\textsc{Ema}^7$—- only three steps on the circle of fifths – whereas in the rest of the cases, four steps were needed to connect the parts. This is also evident from the fact that the depth after the modulation increases by 3 instead of 4.

7. CONCLUSIONS AND FURTHER RESEARCH

Regarding the syntactic similarity between music and natural language, we have applied Lambek Calculus — corresponding to categorial grammar — to the analysis of chord sequences. Since the calculus is a formal proof system, it offers a robust and logically grounded framework. Moreover, it provides a clear and visual representation of harmonic structure that can aid music analysts.

In addition, we proposed a formal method to measure harmonic complexity in terms of the number of tonal modulations and the derivational depth of the proof tree. By quantifying these aspects, we introduced a notion of minimality that offers an optimality criterion for harmonic analyses.

Although we focused on tonal jazz harmony in this paper, the same formalism can be applied to other genres simply by modifying the modal and tonal systems. Since Lambek Calculus is inherently compatible with different rule sets, we regard our approach is highly adaptable for broader applications in music theory.

The current system, however, still includes the following issues. (i) We have added modal operators to access other keys into Lambek Calculus, however, we could not ensure the decidability and have not assessed the computational complexity by this extension. Although an inefficient algorithm requires $\mathcal{O}(n^3)$-time for parsing a sentence of length n, chord sequences are relatively short and thus less restrictive for practical applications. Also, different harmonic interpretations coexist in music, and thus, the notion of derivational ambiguity – the multiple proofs for a given sequence – may become conversely a meaningful measure of harmonic richness. (ii) For more rigorous introduction of modal operators, we would need the labeled sequent calculus where the labels (indices for possible worlds) represent keys; then, a modal operator works to access different label(s). Since the original Lambek Calculus do not include labels, we may need a big theoretical reconstruction. Instead of labels, we have tentatively employed 'tags' that is a tool outside of the formal syntax. Therefore, we need to adequately accommodate the tags into the sequent calculus.

Besides such logical issues, future directions include:

Automation A major objective is the development of computational tools capable of automatically producing harmonic analyses. Such tools would enhance the efficiency for both theorists and educators.

Semantic Interpretation Another crucial goal is the construction of a semantic framework to connect formal expressions with musical meanings — for example, mapping expressions like $V\backslash I$ to well-known concepts such as perfect cadences.

Empirical Applications Applying the system to large data sets of musical works will allow for empirical evaluation of its descriptive power and may reveal stylis-

tic trends in harmonic practice. The concept of depth could serve as a meaningful parameter for comparing different genres or composers.

Generative Capabilities By reversing the analytic process, it will be possible to generate harmonic progressions from the system itself. Controlled variation of proof depth could guide the creation of musical material with desired levels of structural complexity — opening new avenues for computational creativity.

These developments aim to refine the theoretical foundation and expand the practical utility of Lambek Calculus in music analysis and beyond.

8. ACKNOWLEDGMENT

This work is supported by JSPS Kaken 25H01169.

9. REFERENCES

[1] N. L. Wallin, B. Merker, and S. Brown, *The Origins of Music*. The MIT Press, 2000.

[2] P. Ball, *The Music Instinct*. Vintage Books, 2010.

[3] T. Winograd, "Linguistics and the computer analysis of tonal harmony," *Journal of Music Theory*, vol. 12, no. 1, pp. 2–49, 1968.

[4] P. N. Johnson-Laird, O. E. Kang, and Y. C. Leong, "On musical dissonance," *Journal of Music Perception*, vol. 30, no. 1, pp. 19–35, 2012.

[5] F. Lerdahl and R. Jackendoff, *A Generative Theory of Tonal Music*. MIT Press, 1981.

[6] M. Rohmeier, "Towards a generative syntax of tonal harmony," *Journal of Mathematics and Music*, vol. 5, no. 1, pp. 35–53, 2011.

[7] D. Harasim, T. J. O'Donnell, and M. A. Rohrmeier, "Harmonic syntax in time: Rhythm improves grammatical models of harmony," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 335–342.

[8] M. Rohrmeier, "Towards a generative syntax of tonal harmony," *Journal of Mathematics and Music*, vol. 5, pp. 35–53, 2011.

[9] M. Rohrmeier and F. Moss, "A formal model of extended tonal harmony," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference*, 2021, pp. 569–578.

[10] G. Gentzen, "Untersuchungen über das logische schließen," *I. Math Z 39*, pp. 176–219, 1935.

[11] J. Lambek, "The mathematics of sentence structure," *The American Mathematical Monthly*, vol. 65, no. 3, pp. 154–170, 1958.

[12] M. Granroth-Wilding and M. Steedman, "Statistical parsing for harmonic analysis of jazz chord sequences," in *Proceedings of the International Computer Music Conference (ICMC)*, 2012, pp. 478–485.

[13] M. Bizzarri, "Music and logic: a connection between two worlds," in *Proceedings of The 16th International Symposium on Computer Music Multidisciplinary Research (CMMR)*, 2023.

[14] M. Bizzarri and S. Tojo, "Chord progression analysis by labelled lambek calculus," in *Proceedings of ICNMC*, 2024.

[15] ——, "Foundations of LLCM: Labelled lambek calculus for music analysis," in *Proceedings of Evo∗*, 2025.

[16] ——, "A formal proof-theoretic approach to music analysis using labelled lambek calculus," in *SIGMUS*, 2024.

[17] R. Oehrle, E. Bach, and D. Wheeler, *Categorial Grammars and Natural Language Structures*. Netherlands: Reidel, 1988.

[18] N. Chomsky, *Syntactic Structures*. Mouton & Co., 1957.

[19] D. Kozen, *Automata and Computability*. Springer, 1997.

[20] R. Moot and M. Piazza, "Linguistic applications of first order intuitionistic linear logic," *Journal of Logic, Language and Information*, vol. 10, no. 2, pp. 211–232, 2001.

[21] S. Tojo, "Modal logic for tonal music," in *Perception, Representations, Image, Sound, Music: 14th International Symposium, CMMR 2019, Revised Selected Papers*, 2019, pp. 113–128.

[22] M. Moortgat, "Lambek calculus and its modal extensions," in *Proceedings of WoLLIC*, Halifax, 2023.

[23] G. Hughes and M. Cresswell, *A New Introduction to Modal Logic*. Routledge, 1997.

[24] F. Lerdahl, *Tonal Pitch Space*. Oxford University Press, 2001.

[25] E. Chew, "Slicing it all ways: Mathematical models for tonal induction, approximation, and segmentation using the spiral array," *INFORMS Journal on Computing*, vol. 18, p. 305, Aug. 2006.

[26] C. Krumhansl, *Cognitive Foundations of Musical Pitch*. Oxford University Press, 1990.

[27] E. Gollin and A. Rehding, Eds., *The Oxford Handbook of Neo-Riemannian Music Theories*. Oxford University Press, 2011.



Figure 6. Analysis of the first 8 bars of *In your own sweet way*

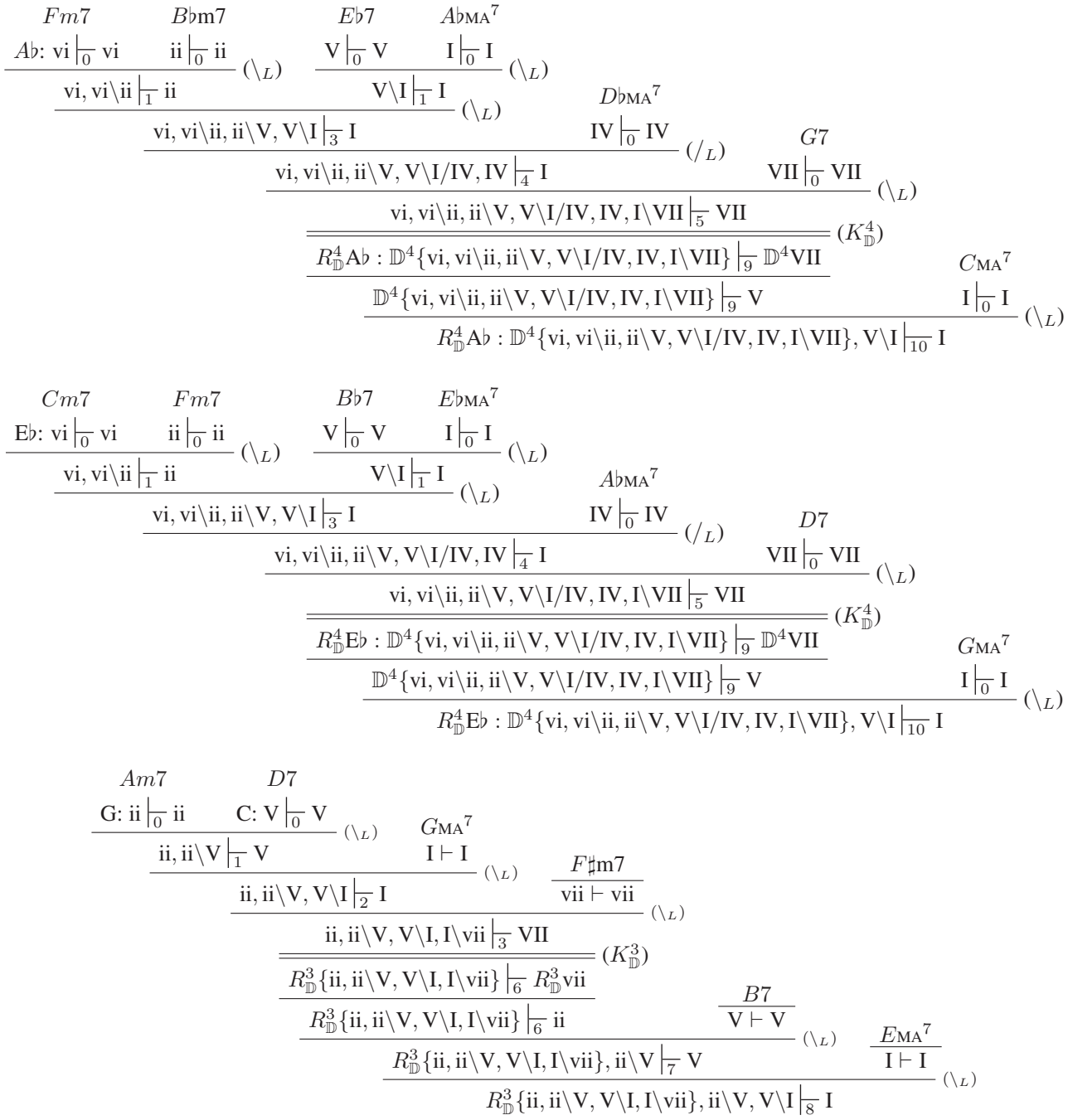


Figure 7. Analysis of the first 24 bars of *All the Things You Are* by Jerome Kern. Notice how the first two trees are exactly the same but in different tonalities. However, thanks to the $R^4_{\mathbb{D}}$ rule, it is also possible to highlight that the relationship between the two tonalities in both sections is identical. Without using this rule and instead attaching the new tonality directly—C in the first case and G in the second—the similarity between the two might be harder to perceive. The double lines indicate instances of the same rule.

The Role of Artificial Intelligence in Electroacoustic Score Generation: A Review

Ali Balighi

Texas Tech University
https://alibalighi.com
abalighi@ttu.edu

Hideki Isoda, Ph.D.

Texas Tech University
https://hidekiisoda.com
hideki.isoda@ttu.edu

ABSTRACT

Electroacoustic music—marked by nonlinear temporal structures, complex timbral materials, and non-standard notational practices—poses persistent challenges for representation, performance, and dissemination. This article offers a critical thematic review of how artificial intelligence (AI) shapes electroacoustic score generation across three primary domains: symbolic, spatial, and gestural. We survey contemporary approaches—symbolic modeling, deep generative methods, gesture mapping, and spatial trajectory prediction—and clarify their musical affordances and limits. We introduce a purpose-first taxonomy and three focused case discussions to deepen analysis without expanding scope. Alongside technical considerations, we foreground authorship, performer agency, interpretability, and aesthetic control, and examine ethical concerns around data bias, cultural specificity, and creative homogenization. We argue that AI's potential is strongest when systems are designed transparently and inclusively and when practices are collaborative, interdisciplinary, and reflexive. By synthesizing developments with philosophical inquiry, we outline pathways for responsible, imaginative integrations of AI in electroacoustic music.

1. INTRODUCTION

1.1. Topic Overview

Electroacoustic music, emerging from mid-twentieth-century experimental practices, diverges from traditional acoustic paradigms in both material and method. Defined by its reliance on recorded, synthesized, or transformed sound—often dislocated from physical instruments—this genre often emphasizes timbral, spatial, and gestural dimensions over pitch and meter. Composers in this domain, such as Edgard Varèse and Pierre Schaeffer, reimagined the score not as a static representation of notes, but as a dynamic system responsive to sonic transformation and performer interaction [1, 2].

However, despite its innovative sonic vocabulary, electroacoustic music remains constrained by a persistent and unresolved issue: the lack of widely standardized, legible, and expressive score systems. Unlike traditional staff notation, which is codified and widely understood, electroacoustic works often employ a pluralism of representations—graphic scores, animated systems, gestural maps, or software interfaces—that are often bespoke, ambiguous, and performatively opaque [3, 4]. These systems can struggle to convey critical parameters such as spectral morphologies, spatial trajectories, or nonlinear form [5], which can place interpretive burdens on performers and complicating reproducibility, pedagogy, and archiving [6].

In this landscape of notational fragmentation, artificial intelligence (AI) emerges as a promising but under-examined approach. AI techniques, particularly those rooted in machine learning, generative modeling, and gesture analysis, can facilitate the translation of complex sonic or spatial data into interpretable forms. They can support tools not only for symbolic rendering but also for dynamic, multimodal, and co-creative score systems that reflect the real-time, embodied, and spatial dimensions of electroacoustic practice [7, 8].

1.2. Purpose and Scope of the Review

This article presents a critical thematic review of AI's role in electroacoustic score generation, with a focus on how intelligent systems may address the core problem of notational insufficiency. The review is structured around the following three research questions:

- *How can AI facilitate the generation of symbolic, spatial, and gestural scores that are interpretable by performers and composers?*
- *What are the current limitations and potentials of AI systems in transcribing, modeling, or generating scores for non-pitched and electroacoustic sound materials?*
- *How does the integration of AI reshape traditional assumptions about authorship, creativity, and performance in electroacoustic music?*

To anchor these questions, Table 1 below offers a taxonomy of electroacoustic score types and the corresponding roles AI may play in their generation, adaptation, or interpretation. We classify by primary function—visual/temporal guidance, body/physical interaction, and spatial localization—with other aspects treated as secondary dimensions. This typological overview frames the subsequent sections and identifies key intersections between score formats and AI technologies

Primary function	Score Type	Key Features	Challenges	AI Role	References
Visual/Temporal Guidance	Graphic Scores	Visual symbols, lines, shapes, textures	Ambiguity, lack of visual vocabulary	Image-to-audio translation, visual-sound mapping	[1, 9, 10]
Visual/Temporal Guidance	Symbolic Scores (MIDI/Staff)	Discrete pitch/rhythm representations	Inadequacy for noise-based or spatial material	Fuzzy logic, symbolic abstraction from spectral data	[11, 12]
Visual/Temporal Guidance	Animated / Real-Time Scores	Time-evolving visualizations	Performance alignment, computational complexity	Generative score engines, real-time gesture and audio tracking	[5, 9, 13]
Body/Physical Interaction	Gestural Maps	Motion-capture, haptic or physical cues	Interpretive burden, sensor calibration	Gesture recognition, AI-based motion-to-sound mapping	[14-16]
Spatial Localization	Spatial Scores	Diffusion cues, trajectory maps, multi-speaker configurations	Representation of space and movement	Trajectory prediction, spatial interpolation, multimodal data alignment	[17-19]

Table 1. Taxonomy of Score Types in Electroacoustic Music and Related AI Roles

This taxonomy illustrates the diversity and complexity of electroacoustic score formats while highlighting how AI can mediate the translation between sonic data, visual representation, and human interpretation. The remainder of this review explores these intersections in depth, organized around symbolic generation, audio transcription, deep generative models, spatial/gestural integration, and critical reflection on authorship, legibility, and ethics.

This review contributes to both scholarly and compositional discourses by mapping out the current state of AI-assisted score generation and identifying paths toward more inclusive, legible, and expressive electroacoustic music practices.

2. FOUNDATIONS: ELECTROACOUSTIC MUSIC AND SCORE REPRESENTATION

This section establishes the representational frame for the taxonomy that follows. Electroacoustic practice often privileges timbre, spatialization, and gesture, which can strain conventional staff notation. In response, composers use a range of alternative formats—graphic, animated/real-time, gestural, and spatial—each with distinct affordances and limitations for composition and performance. To clarify where AI may assist, we map these formats and the challenges they pose for automation and interpretation.

2.1. Characteristics of Electroacoustic Composition

Electroacoustic practice motivates score formats that convey not only sonic structure but also embodied, spatial, and performative parameters [19]. The diversity of sound materials—granular synthesis, spectral morphing, noise, diffusion—can resist symbolic transcription, placing additional demands on representation and rehearsal. These considerations motivate alternative score systems beyond staff notation, each with distinct affordances and limitations that will be situated by the taxonomy in Table 1 and developed in the sections that follow.

2.2. Evolution of Notation in Electroacoustic Practice

The earliest electroacoustic works were often realized in studios and lacked formal scores altogether, being presented as fixed media recordings [20]. When scores were used, they consisted of verbal instructions, timing charts, or circuit schematics, intended more for technical execution than artistic interpretation [1], reflecting a composer-as-technician model, where the score served engineering more than performance.

By the late twentieth century, composers adopted *graphic scores* to convey more subjective or spatially emergent phenomena. These scores relied on lines, shapes, and colors to suggest dynamics, movement, and texture rather than precise pitches [9]. Yet their *semantic openness* often led to widely divergent performances, depending heavily on performer interpretation [5]. In response, *animated scores* were developed using software environments that could display evolving instructions in real time. Eigenfeldt's real-time notation engine and systems like GNMISS [9] allowed performers to interact with time-sensitive gestures and environmental cues. These systems expanded the expressive range of scores but introduced new requirements for screen-based interaction and performer adaptability [13].

Recent practice integrates symbolic notation, gestural input, and spatial representation to create multimodal environments in which performers interpret pitch, movement,

and space as integrated data streams [14]. However, integrating multiple data types into a coherent interface remains an open design challenge.

2.3. Connecting to AI Integration

Each score type presents *distinct opportunities for AI intervention*. Symbolic scores align with AI's strengths in rule-based generation and symbolic abstraction [11]. Graphic and animated scores open pathways for image-to-audio learning, temporal prediction, and visual feature extraction [10]. Gesture-based and spatial scores present opportunities for *sensor-driven input*, pattern recognition, and spatial behavior modeling [14].

As we move forward, the diversity of score formats require equally diverse AI strategies, with an emphasis on *co-design between composers, systems, and performers*. The next sections explore these intersections in detail—evaluating how AI supports symbolic rendering, real-time performance, generative co-creation, and spatial or gestural intelligence in electroacoustic score systems.

3. OVERVIEW OF ARTIFICIAL INTELLIGENCE IN MUSIC

Artificial Intelligence (AI) has become increasingly central to contemporary music creation, offering new methodologies for composition, performance, analysis, and interaction. Its impact spans both traditional symbolic domains (e.g., MIDI, score notation) and audio-based processes, with applications ranging from harmonic modeling to spatial audio rendering. For electroacoustic music, where sound exists beyond conventional pitch-rhythm syntax, AI offers unique affordances for abstraction, transformation, and multimodal integration.

3.1. Core AI Paradigms and Music Applications

AI in music operates through three dominant paradigms: rule-based systems, machine learning (ML), and deep learning (DL). Early systems relied on rule-based logic—such as heuristic or constraint-based models—to encode compositional procedures [11]. While transparent and controllable, these models lacked flexibility when addressing nonlinear or textural sound structures typical of electroacoustic music.

By contrast, machine learning models learn from examples. Trained on musical corpora, they extract statistical patterns to generate stylistically coherent outputs [21]. Within this paradigm, deep learning, including convolutional and transformer-based neural networks, enables high-dimensional feature learning and abstraction—crucial for capturing gestural, spectral, or affective elements in music [22]. Rule-based systems are transparent yet inflexible for nonlinear or textural materials, while ML/DL approaches depend on training data and can be opaque in attribution.

3.2. Symbolic vs. Audio Approaches

AI systems in music are generally categorized as symbolic or audio-based, but practice often hybridizes both; we use this distinction heuristically. Symbolic approaches—using representations like MIDI or MusicXML—are effective in tonal or rhythmically regular domains. They support melodic generation, chord progressions, and formal structuring [12]. However, symbolic models often falter in electroacoustic contexts, where sonic identity may not conform to pitch or rhythmic quantization [4].

Audio-based systems, which work directly with raw or processed sound, bypass the limitations of notation. These include spectrogram-based models, waveform autoencoders, and audio-domain transformers that generate or transform textures, timbres, and gestures without requiring symbolic intermediaries [21]. This capacity is especially vital for electroacoustic music, where expressive material exists in continuous spectral and temporal dimensions. Symbolic models can falter when sonic identity resists quantization, while audio-domain methods bypass notation yet can raise interpretability and real-time constraints.

3.3. AI in Creative and Compositional Practice

AI has moved from static score generation toward interactive and co-creative systems. Generative models—like those powering Magenta, MusicLDM, and Flow-Machines—can produce stylistically nuanced music based on training data [10]. These tools enable composers to explore variant forms, novel combinations, or data-informed aesthetics.

Increasingly, AI models are designed for real-time performance and improvisation. For instance, Donohue+ responds dynamically to live musicians, generating interactive electroacoustic accompaniments based on performative cues [23]. This positions AI not as a static tool, but as an improvisational partner, shifting the composer-performer-AI triad toward shared agency.

AI has also been incorporated into emotionally responsive systems, where affective computing drives the generation of music aligned with user input or mood [22]. These models open new pathways for affective engagement, adaptive environments, and therapeutic applications. Co-creative systems remain shaped by their training data, may shift perceived agency among composer–performer–system, and must meet practical responsiveness for performance.

3.4. Pedagogical and Analytical Functions

In educational contexts, AI enables personalized learning and exploration. Tools such as those developed in the EARS 2 project provide students with interactive environments to experiment with sound transformation, symbolic abstraction, and spatial composition [24]. Similarly, Cooper highlights how AI can assist in curriculum design, compositional scaffolding, and assessment—particularly in complex fields like electroacoustic music [25].

From an analytical perspective, AI aids in pattern recognition, gesture detection, and score–audio alignment, providing insights into structural and perceptual dimensions that are difficult to extract manually [11].

Ethical and cultural considerations—including dataset transparency, bias, and inclusion—are consolidated in section 6.

3.5. Summary: AI Capabilities in Electroacoustic Contexts

To conclude, the main affordances of AI in music—particularly for electroacoustic composition—can be summarized as follows:

Compositional/performative affordances

- Symbolic Generation: Automated score production from style-trained models [12].
- Audio Transformation: Textural/spectral manipulation without symbolic mediation [21].
- Gesture Recognition: Mapping physical motion to control sound or notation [14].
- Real-Time Interaction: Responsive systems for live performance [23].
- Affective Adaptation: Emotion-aware composition tools [22].

Analytical/pedagogical affordances

- Educational Support: Interactive, AI-informed learning platforms [25].
- Score-Audio Mapping: Analytical alignment of non-standard scores to sonic outcomes [13].

The sections that follow examine these affordances in relation to symbolic modeling, spatialization, and gestural performance within electroacoustic score generation.

4. AI FOR ELECTROACOUSTIC SCORE GENERATION

Artificial Intelligence (AI) presents both opportunities and complexities in resolving the longstanding challenges of score generation for electroacoustic music. Given the genre's resistance to standard notation—due to its use of textural, non-pitched, spatial, and gestural elements—AI systems serve not merely as tools for translation, but as collaborators in encoding and interpreting ephemeral sonic content. This section critically evaluates the major AI approaches currently applied to electroacoustic score generation, examining their technical affordances, artistic viability, and interpretive challenges.

4.1. Symbolic Modeling and Score Abstraction

AI-based symbolic modeling typically involves mapping features such as pitch, onset, or dynamics to MIDI or staff notation formats. In electroacoustic contexts, however, the translation of sound into symbolic terms can risk oversimplification (see section 2). Traditional symbolic systems are poorly suited for representing spectral morphologies, spatial behaviors, or gestural nuances inherent to electroacoustic composition [1].

To mitigate this misalignment, researchers have explored fuzzy-logic and probabilistic approaches. For example, a fuzzy-rule-based framework can interpret higher-level abstractions—such as timbral brightness or gestural envelope—yielding a more flexible symbolic representation. [12]. While such models can improve the capture of ambiguous sonic content, their interpretability remains limited. The resultant scores often require significant post-editing by composers to become performable or legible [6].

More broadly, symbolic AI still privileges discrete events over continuous transformations. Electroacoustic works built on glissandi, spectral smearing, or convolution are fundamentally resistant to fixed symbolic codification. This raises the question: should AI strive for symbolic fidelity, or embrace hybrid systems that integrate symbolic, visual, and gestural layers? The latter path appears more promising but remains underexplored [26].

4.2. Audio-to-Score Transcription

AI transcription systems aim to extract notational data from raw audio. While effective in tonal and percussive contexts, such systems can struggle with the ambiguity of electroacoustic textures. Techniques like onset detection, spectral centroid tracking, and gesture inference—often powered by neural networks—have improved in precision [13], but can still fall short when confronted with dense, noisy, or non-repetitive material [6].

Moreover, transcription often produces notations that are technically accurate but aesthetically or semantically irrelevant. A granular synthesis cloud, for example, may be transcribed as a rapid sequence of micro-onsets, misrepresenting its perceptual continuity [1]. Without human-in-the-loop design and co-design (see section 4.4), such systems risk creating scores that are unfaithful to the composer's intent and unperformable in practice [9].

While some systems integrate gesture tracking (e.g., motion capture, sensor input), these mappings often suffer from cultural and contextual ambiguity. A raised arm might signify a crescendo in one context, but trigger spatial diffusion in another. AI can struggles to disambiguate such mappings without contextual learning or supervised training [5, 15]. This reveals a major limitation: gesture fidelity is highly dependent on co-design between AI developers and performers, a relationship still in its infancy.

4.3. Deep Generative Systems (GANs, VAEs, Transformers)

Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and transformer-based architectures have recently been employed to generate scores, spectrograms, and gestural abstractions. These models are

particularly effective in stylistic emulation and the generation of novel content from latent spaces [21].

For example, MusicLDM—a transformer-based model—has been used to interpret visual graphic scores and generate audio responses[10]. This approach circumvents the need for symbolic notation by producing temporal sonic gestures from abstract visual inputs. The strength of such models lies in their ability to handle nonlinear and ambiguous material—a trait ideal for electroacoustic media.

These systems present interpretability and dataset-dependence constraints: GANs are often opaque [11]; VAEs can reduce output resolution; and Transformers depend heavily on the training corpus, which can replicate dominant stylistic features [21, 27]. Taken together, these issues can yield smeared or overgeneralized gestures and contribute to creative homogenization, complicating validation and control [11, 21, 27]. These concerns echo broader critiques of AI in art: who is really composing—the model, the training data, or the curator?

4.4. Spatial & Gestural Encoding

4.4.1.Spatial encoding

One of the most promising frontiers is the application of AI to spatial dimensions of score generation. Electroacoustic music frequently involves sound movement through multichannel environments—features that can resist conventional notation. AI has been used to track, predict, and generate spatial trajectories using tools such as ambisonics, speaker diffusion algorithms, or spatial interpolation models [17, 19].

4.4.2.Gestural mapping

In gestural contexts, AI can interpret physical movement—via accelerometers, depth cameras, or wearable sensors—and map these to sonic transformations or score events [14, 28]. These systems enable embodied composition, aligning physical gesture with acoustic feedback in real-time.

4.4.3.Limitations and calibration

However, these mappings are not universally intuitive. AI often lacks the semantic context to interpret a gesture's expressive intent. Furthermore, reliance on real-time tracking introduces issues of latency, false positives, or misclassification, particularly in performance settings with high variability [5, 29].

The future lies in co-adaptive systems—where AI not only learns from human gesture but allows for correction, nuance, and iterative refinement by performers and composers alike. Such designs, however, require deeper collaboration between musical and technical domains.

4.5. Comparative Synthesis: Evaluating AI Techniques for Electroacoustic Score Generation

AI-assisted electroacoustic score generation remains an emerging and multifaceted field. While current systems show promise across symbolic, spatial, and gestural domains, each approach comes with distinct limitations in legibility, fidelity, and creative control. The most effective tools tend to be those that *combine AI automation with human feedback*, supporting a hybrid process of co-creation rather than full automation. Future progress will depend not just on technical innovation, but on sustained interdisciplinary collaboration and ethical reflexivity.

AI Technique	Effectiveness	Technical Readiness	Interpretability	Main Limitations
Symbolic Modeling	Moderate (good for stylized input)	High	Moderate to High	Poor handling of gestural/textural elements
Audio Transcription	Low to Moderate (varies by texture)	Moderate	Low	Misrepresents continuous sound; lacks semantic intent
GANs / VAEs	High (for abstract or style-based tasks)	Moderate	Low	Opaque logic; poor generalization across domains
Transformers (e.g. MusicLDM)	High (especially for visual/audio integration)	Moderate to High	Moderate	Bias from training data; creative homogenization
Gesture-Based Systems	High (with human co-design)	Moderate	High (when transparent)	Ambiguity of meaning; high variability in input
Spatial AI Models	High (in controlled environments)	Moderate	Moderate	Latency; performer-dependency; needs calibration

Table 2. Comparative Evaluation of AI Techniques for Electroacoustic Score Generation

5. PLATFORMS, TOOLS, AND EXAMPLES

The development of AI-driven tools for electroacoustic score generation has accelerated in recent years, producing a growing ecosystem of platforms that support compositional, performative, and pedagogical applications. These tools vary significantly in terms of input modalities, output formats, and user engagement, ranging from symbolic modeling environments to real-time gesture-responsive systems. This section offers a structured overview of key

platforms and cross-references case studies that demonstrate their real-world application.

5.1. Overview of Platforms

Several prominent platforms have emerged that integrate AI into music creation and score generation.

Google's Magenta uses deep learning models such as recurrent neural networks (RNNs) and transformers to generate symbolic and audio data, often in the form of MIDI sequences. While designed primarily for general musical tasks, its modularity allows adaptation for experimental and electroacoustic uses [11].

Sony's FlowMachines uses constraint-based AI to model stylistic patterns, focusing on automatic composition and accompaniment generation. It excels in generating stylistically coherent symbolic outputs but is less suited for the gestural and spatial demands of electroacoustic music [21].

IRCAM's OpenMusic provides a visual programming interface for algorithmic and symbolic composition. While not AI-native, it can be extended with machine learning modules and supports integration with gesture-tracking or spatial audio tools [11].

Specialized platforms such as GNMISS (Graphical Notation for Mixed-Input Score Systems) are tailored specifically for real-time score rendering in electroacoustic and telematic settings. It supports live input and dynamic visualization, emphasizing performance interaction [9]. Similarly, Eigenfeldt's real-time notation engine adapts score material based on performer behavior, facilitating a dialogic relationship between system and performer [13].

Gesture-sensitive systems like those link physical movement to musical transformations, often using sensors like Wii remotes, depth cameras, or accelerometers [5, 14]. These systems serve not only in live performance but also as potential sources of score generation through motion mapping and AI classification.

In the domain of spatial music, platforms combine spatial audio rendering with AI-driven spatialization control, enabling trajectory planning and diffusion automation across speaker arrays [17, 18].

Platform	Input Type	Output	Application Area
Magenta (Google)	MIDI, symbolic sequences	Symbolic/music/audio (MIDI, Wave)	Style emulation, generative composition
FlowMachines (Sony)	Musical constraints, corpora	Symbolic music	Stylistic modeling, automatic composition
OpenMusic (IRCAM)	Algorithms, data flow, symbolic inputs	Notation, score structures	Algorithmic composition, gesture-spatial links
GNMISS	Audio, performer input	Dynamic graphical score	Live electroacoustic performance
Eigenfeldt's Notation Engine	Performer behavior, real-time input	Animated score	Responsive live notation
Gesture Systems	Motion data	Control signals, score fragments	Gesture-informed generation/performance
Rossetti & Manzolli / Kang	Spatial coordinates, audio features	Diffusion cues, trajectory maps	AI-enhanced spatial scoring

Table 3. Comparison of AI-Based Platforms for Electroacoustic Score Applications

5.2. Case Studies and Applications

A number of case studies illustrate how these platforms are deployed in artistic and research contexts. MusicLDM, a transformer-based model, was used to interpret Cardew's Treatise, a canonical graphic score. [10]. The system mapped visual forms to audio gestures, allowing performers to engage with AI-generated material in real time. This case underscores AI's potential to act as a semantic intermediary between abstract notation and sonic output.

Another example is Donohue+, an improvisational accompaniment system that listens to live input and generates responsive electroacoustic textures [23]. Here, AI functions as a real-time co-performer, making decisions based on gesture, phrasing, and timing. The improvisational model reinforces AI's utility not as a composer per se, but as an adaptive, interactive agent within live settings.

In educational settings, the EARS 2 project [24] integrates AI-based modules to help students engage with sound transformation, symbolic representation, and compositional experimentation. While less generative in scope, EARS 2 demonstrates how AI can scaffold access to the conceptual complexities of electroacoustic music.

5.3. Synthesis: Comparative Insights

These platforms collectively highlight a shift in AI's role from content generator to creative interlocutor. Magenta and FlowMachines are strongest in stylistic modeling but lack adaptability to the real-time, embodied dynamics of electroacoustic performance. OpenMusic and GNMISS,

while more flexible in input modalities, rely heavily on user customization and integration with external systems.

The most promising applications, such as Donohue+ and MusicLDM, underscore the importance of real-time responsiveness, multimodal input, and semantic adaptability. These systems suggest that AI's strength lies in its ability to facilitate hybrid workflows—merging symbolic, spatial, and gestural modalities—rather than replacing human creativity.

However, limitations remain. Most platforms require extensive setup, lack standardization in score output, and demand high user expertise. Furthermore, gesture and spatial systems still face interpretive challenges and can misrepresent expressive intent if not carefully co-designed with performers.

While no single platform offers a comprehensive solution, the collective landscape of tools illustrates an emerging framework for AI-assisted electroacoustic composition. Future development must prioritize usability, cross-platform interoperability, and collaborative design to ensure these tools serve diverse artistic and educational communities.

6. AESTHETIC AND PHILOSOPHICAL CONSIDERATIONS

The integration of artificial intelligence into electroacoustic music composition brings not only technical innovation but also profound aesthetic and philosophical challenges. These questions—long embedded in experimental music—are now intensified by machine agency, generative unpredictability, and blurred authorship. This section explores these tensions from both composer and performer perspectives, emphasizing the need for critical engagement with emerging creative paradigms.

6.1. Authorship and Creative Agency

AI complicates traditional notions of authorship by distributing creative agency across human and non-human systems. In electroacoustic music, where composers often design systems rather than discrete musical texts, AI further diffuses the role of the composer into curator, co-editor, or system-tuner [30]. Generative models, particularly those based on deep learning or transformer architectures, can produce stylistically coherent outputs without direct human specification, raising the question: who is the composer?

While some argue for understanding AI as a sophisticated tool [27], others view it as a creative collaborator, particularly when its behavior is context-sensitive, unpredictable, and capable of adaptation [11]. Systems like Donohue+ exemplify this ambiguity, where AI responds to live performers with improvisational agency, challenging hierarchical notions of authorship [23].

At the same time, the aesthetic trajectory of AI-generated outputs is shaped by training data. Models often reflect stylistic biases—echoing dominant forms, harmonic conventions, or culturally normative gestures—raising concerns about homogenization and the loss of artistic specificity [31]. Thus, while AI expands creative possibilities, it also embeds invisible constraints, subtly steering compositional decisions.

6.2. Performer Interpretation and Legibility

Performers face their own set of challenges when engaging with AI-generated or mediated scores. Electroacoustic music already demands alternative literacies—reading graphic scores, interpreting spectrograms, or navigating gestural interfaces. AI adds further complexity: dynamic, evolving scores can resist fixity, while transcription models may oversimplify noise-based or gestural content into misleading symbolic forms [4].

Real-time systems shift the performer's role toward that of a responsive interpreter, requiring improvisatory fluency and system familiarity (see sections 4.2, 5.1). In some contexts, performers must *listen to* or *perform with* the AI itself, co-creating outcomes that vary with each realization. This reconfigures the score not as a fixed directive but as a negotiation space between human and machine intentions.

Such shifts raise questions of rehearsal practice, performative agency, and artistic identity. When scores change during performance or are generated live, criteria for fidelity and interpretation shift toward responsiveness and negotiated control within hybrid human–AI ensembles. Related philosophical considerations about authorship, score status, interpretation, and embedded aesthetics are discussed in sections 6.1 and 6.3.

6.3. Critical Hybrid Aesthetic

We use the phrase critical hybrid aesthetic to denote hybrid human–AI mediation that is legible to performers and audiences and foregrounds ethics and performer/composer agency. The philosophical implications of AI in electroacoustic music are not merely abstract—they shape how works are conceived, rehearsed, and received. Composers are increasingly confronted with questions of transparency, intent, and disclosure. Performers, in turn, navigate shifting forms of agency, often learning new interpretive frameworks without standardized notation.

Rather than resolving these tensions, a more productive path lie in embracing the critical hybrid aesthetic—one that acknowledges AI as part of a distributed creative ecology while foregrounding human intentionality, cultural context, and ethical reflexivity. Berkowitz frames creativity as relational, layered, and co-evolving [30].

In this light, the score becomes a site of dialogue between composer and performer, human and machine. Ambiguity persists but can be productive, enabling collaborative authorship and expanded musical imagination.

7. FUTURE DIRECTIONS AND OPEN QUESTIONS

The integration of artificial intelligence into electroacoustic score generation opens a complex, interdisciplinary research frontier—technologically promising but conceptually unsettled. While tools for symbolic generation, spatialization, and gesture recognition have advanced, many questions remain regarding usability, cross-cultural relevance, and artistic coherence. Future development must extend beyond system design into empirical evaluation, critical theory, and collaborative experimentation.

7.1. Hybrid Notation and System Integration

One of the most compelling directions for future work lies in the development of hybrid notation environments—systems that seamlessly merge symbolic, graphic, spatial, and gestural modalities. These environments will require advanced multimodal machine learning to correlate sonic, visual, and motion data in real time, enabling scores that are both expressive and interpretable [14, 26].

Such systems might integrate AI-assisted symbolic abstraction with spatial trajectory mapping and live gesture recognition, rendered through responsive score interfaces. The potential for adaptive scores—which modify themselves in response to performance context—offers exciting possibilities for real-time co-creation. However, this vision demands robust usability testing, performer feedback loops, and design frameworks that prioritize clarity and creative agency.

Agenda: hybrid-notation and system-integration studies.

7.2. Performer-Centered and Audience Studies

To date, most AI–electroacoustic systems have been evaluated through composer or developer lenses. There is a critical need for performer-centered research that investigates how musicians interpret AI-generated scores, navigate real-time systems, and negotiate creative control [5, 9].

Equally important are studies of audience reception, particularly in works involving generative or non-repeatable elements. How do listeners perceive musical authorship, affective coherence, or spatial immersion in AI-mediated performances? Addressing these questions will enrich our understanding of how AI reshapes not just creation, but also perception and meaning making.

Agenda: performer-centered studies and audience-reception studies.

7.3. Cross-Cultural and Inclusive Frameworks

Most current AI systems are trained on Western musical corpora and built within Eurocentric aesthetic frameworks [31]. This poses a risk of cultural homogenization and the marginalization of alternative sonic practices. Future research should prioritize cross-cultural collaboration, dataset diversification, and the design of tools that reflect varied conceptions of sound, gesture, and structure.

AI-assisted systems must be shaped by inclusive values, offering ways to engage with idiomatic instruments, oral traditions, or nonlinear temporalities. Such approaches not only address ethical concerns but also expand the expressive capacity of AI in music.

Agenda: cross-cultural and inclusive framework studies.

7.4. Evaluation Metrics and System Transparency

Finally, a crucial methodological frontier involves developing new evaluation metrics for AI-assisted composition. Traditional metrics (e.g., tonal accuracy or rhythm matching) are insufficient for electroacoustic works. Instead, we must consider affective coherence, performative fluidity, spatial intelligibility, and artistic novelty. Transparent, explainable AI models—those that reveal how decisions are made—will be essential for composer trust and performer adaptation [30, 32].

Agenda: evaluation-metrics and system-transparency studies.

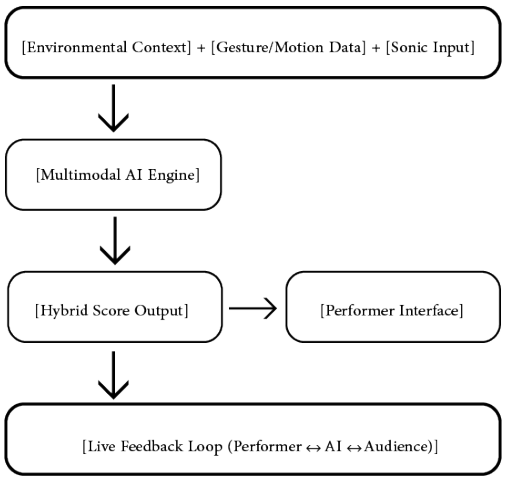


Figure 1. Conceptual Workflow for Future AI-Electroacoustic Systems

- **Sonic Input**: Audio features from live or recorded sources.
- **Gesture/Motion Data**: Captured via sensors or cameras.
- **Environmental Context**: Includes spatial configuration, emotional intent, or audience feedback.
- **Multimodal AI Engine**: Interprets and integrates data into generative decisions.
- **Hybrid Score Output**: Visual or symbolic content rendered for performance.
- **Live Feedback Loop**: Adaptive changes informed by performer and audience behavior.

As AI technologies continue to evolve, the most meaningful progress will come not from algorithmic novelty alone,

but from interdisciplinary collaborations that center artistic experience, cultural equity, and critical reflection. The next chapter of AI in electroacoustic music will be written not just by machines, but by communities of makers, listeners, and thinkers committed to expanding what music—and musical authorship—can mean.

8. CONCLUSION

Artificial intelligence has emerged as a powerful force in electroacoustic score generation, capable of transforming the ways composers conceive, notate, and perform sound. Through symbolic modeling, audio-to-score transcription, real-time gesture interpretation, and deep generative systems, AI offers technical support to many of the genre's longstanding challenges, such as the representation of non-pitched material, spatialization, and performer interactivity.

Yet limitations remain. Current systems often struggle with noise-rich, gestural, or culturally specific content that resists abstraction into pitch or rhythm. Hybrid notation environments remain largely experimental, requiring new performer literacies and clearer design frameworks. Deep learning models also risk opacity, reinforcing stylistic biases embedded in training data.

As AI continues to redefine the boundaries of musical authorship, notation, and performance, the guiding imperative is clear: design systems not only to innovate, but to preserve cultural plurality and creative agency.

9. REFERENCES

[1] M. Gatt, "The OREMA Project: A Call for the Liberation of Sound Analysis," *Organised Sound,* vol. 20, no. 3, pp. 316–322, 2015, https://doi.org/10.1017/s1355771815000242.

[2] J. Long, J. Murphy, D. Carnegie, and A. Kapur, "Loudspeakers Optional: A history of non-loudspeaker-based electroacoustic music," *Organised Sound,* vol. 22, no. 2, pp. 195–205, 2017, https://doi.org/10.1017/S1355771817000103.

[3] B. Bossis, "The Analysis of Electroacoustic Music: From Sources to Invariants," *Organised Sound,* vol. 11, no. 2, pp. 101–112, 2006, https://doi.org/10.1017/s135577180600135x.

[4] M. Clarke, "Analysing Electroacoustic Music: An Interactive Aural Approach," *Music Analysis,* vol. 31, no. 3, pp. 347–380, 2012, https://doi.org/10.1111/j.1468-2249.2012.00339.x.

[5] C. D. Perales, C. Portalés, and F. J. Sanmartín, "Sonic Gestures Applied to a Percussive Dialogue in TanGram Using Wii Remotes," pp. 216–221, 2009, https://doi.org/10.1007/978-3-642-04052-8_23.

[6] B. Sofronievski and B. Gerazov, "Scorpiano -- A System for Automatic Music Transcription for Monophonic Piano Music," 2021, https://doi.org/10.48550/arxiv.2108.10689.

[7] J. Kwiecień, P. Skrzyński, W. Chmiel, A. Dąbrowski, B. Szadkowski, and M. Pluta, "Technical, Musical, and Legal Aspects of an AI-Aided Algorithmic Music Production System," *Applied Sciences,* vol. 14, no. 9, p. 3541, 2024, https://doi.org/10.3390/app14093541.

[8] A. Balighi, "The Art and Science of Multichannel Audio in Electroacoustic Music: A Review," *Emille,* no. 22, p. 12, 2024, https://doi.org/10.23277/emille.2024.22..002.

[9] I. Whalley, "GNMISS: A Scoring System for Internet2 Electroacoustic Music," *Organised Sound,* vol. 19, no. 3, pp. 244–259, 2014, https://doi.org/10.1017/s1355771814000235.

[10] T. Karchkhadze, K. Shao, and S. Dubnov, "Interpreting Graphic Notation With MusicLDM: An AI Improvisation of Cornelius Cardew's Treatise," pp. 3181–3190, 2024, https://doi.org/10.1109/bigdata62323.2024.10825824.

[11] E. R. Miranda and D. Williams, "Artificial Intelligence In Organised Sound," *Organised Sound,* vol. 20, no. 1, pp. 76–81, 2015, https://doi.org/10.1017/s1355771814000454.

[12] I. Paz, À. Nebot, F. Mugica, and E. Romero, "A Fuzzy Rule Model for High Level Musical Features on Automated Composition Systems," pp. 243–251, 2017, https://doi.org/10.1007/978-3-319-47337-6_25.

[13] A. Eigenfeldt, "Generative Music for Live Performance: Experiences With Real-Time Notation," *Organised Sound,* vol. 19, no. 3, pp. 276–285, 2014, https://doi.org/10.1017/s1355771814000260.

[14] G. K. Sharma, M. Frank, and F. Zotter, "Evaluation of Three Auditory-Sculptural Qualities Created by an Icosahedral Loudspeaker," *Applied Sciences,* vol. 9, no. 13, p. 2698, 2019, https://doi.org/10.3390/app9132698.

[15] K. L. Hagan, "Textural Composition: Aesthetics, Techniques, and Spatialization for High-Density Loudspeaker Arrays," *Computer Music Journal,* vol. 41, no. 1, pp. 34–45, 2017, https://doi.org/10.1162/comj_a_00395.

[16] R. B. Tchemeube *et al.*, "Evaluating Human-Ai Interaction via Usability, User Experience and Acceptance Measures for MMM-C: A Creative AI System for Music Composition," pp. 5769–5778, 2023, https://doi.org/10.24963/ijcai.2023/640.

[17] D. Rossetti and J. Manzolli, "Studying the Perception of Sound in Space: Granular Sounds Spatialized in a High-Order Ambisonics System," *Opus,* vol. 26, no. 2, p. 1, 2020, https://doi.org/10.20504/opus2020b2610.

[18] J. Y. Kang, "Artistic approach to the WFS system," *Musica/Tecnologia,* pp. 27–45, 2023. [Online]. Available: http://digital.casalini.it/5719800.

[19] G. S. Kendall, "Spatial Perception and Cognition in Multichannel Audio for Electroacoustic Music," *Organised Sound,* vol. 15, no. 3, pp.

228–238, 2010, https://doi.org/10.1017/s1355771810000336.
[20] M. Battier, "Electroacoustic Music Studies and the Danger of Loss," *Organised Sound,* vol. 9, no. 1, pp. 47–53, 2004, https://doi.org/10.1017/s135577180400007x.
[21] D. Shalamov, "Role of Machine Learning in Shaping Contemporary Musical Culture," *The Journal of the Ukrainian National Tchaikovsky Academy of Music,* no. 3(64), pp. 91–107, 2024, https://doi.org/10.31318/2414-052x.3(64).2024.314745.
[22] A. Dash and K. Agres, "AI-Based Affective Music Generation Systems: A Review of Methods, and Challenges," *ACM Computing Surveys 56,* vol. 56, no. 11, 2023, https://doi.org/10.48550/arxiv.2301.06890.
[23] S. Gillies and M. S. Donohue, "Donohue+: Developing Performer-Specific Electronic Improvisatory Accompaniment for Instrumental Improvisation," *Organised Sound,* vol. 26, no. 1, pp. 129–139, 2021, https://doi.org/10.1017/s1355771821000121.
[24] L. Landy, R. Hall, and M. Uwins, "Widening Participation in Electroacoustic Music: The EARS 2 Pedagogical Initiatives," *Organised Sound,* vol. 18, no. 2, pp. 108–123, 2013, https://doi.org/10.1017/s1355771813000034.
[25] P. K. Cooper, "Music Teachers' Labeling Accuracy and Quality Ratings of Lesson Plans by Artificial Intelligence (AI) and Humans," *International Journal of Music Education,* 2024, https://doi.org/10.1177/02557614241249163.
[26] C. Hope, "The Future Is Graphic: Animated Notation for Contemporary Practice," *Organised Sound,* vol. 25, no. 2, pp. 187–197, 2020, https://doi.org/10.1017/s1355771820000096.
[27] A. Corbelli, "Beyond the Algorithm. Ethical and Aesthetic Challenges of AI in Music," *Itinera,* no. 28, 2024, https://doi.org/10.54103/2039-9251/27842.
[28] H. Isoda, "Motions to Express Music with AI," 2023. [Online]. Available: https://hdl.handle.net/2123/31523
[29] I. Whalley, "Developing Telematic Electroacoustic Music: Complex Networks, Machine Intelligence and Affective Data Stream Sonification," *Organised Sound,* vol. 20, no. 1, pp. 90–98, 2015, https://doi.org/10.1017/s1355771814000478.
[30] A. E. Berkowitz, "Artificial Intelligence and Musicking," *Music Perception,* vol. 41, no. 5, pp. 393–412, 2024, https://doi.org/10.1525/mp.2024.41.5.393.
[31] X. Zhou, "Analysis of Evaluation in Artificial Intelligence Music," *Journal of Artificial Intelligence Practice,* vol. 6, no. 8, pp. 6–11, 2023, https://doi.org/10.23977/jaip.2023.060802.
[32] F. Xue, "AI integration in creative industries: Challenges and opportunities," *Applied and Computational Engineering,* vol. 104, pp. 21–27, 2024.

A DATA MODEL AND A TSCORE REPRESENTATION FOR THE TABSTAFF+ NOTATIONS FOR ABLETON PUSH

Markus Lepper
semantics gGmbH, Berlin, DE
post@markuslepper.eu

Baltasar Trancón y Widemann
semantics gGmbH, Berlin, DE
Technische Hochschule Brandenburg a.d. Havel, DE

ABSTRACT

The proposal "TABstaff+" by Wilde and White (2024) defines a collection of formats to notate the play on a grid-based tangible user interface. We propose a mathematical formulation for the underlying data model, and present an application to construct and render the corresponding models, using the frameworks for time-related modelling "tscore" and for meter-ruled rhythm notation "metricSplit".

1. CONTEXT

The proposal "TABstaff+" by Wilde and White (2024) defines a collection of formats to notate the play on a grid-based tangible user interface (TUI). [1] Their proposal comprises three formats, namely TAB+ = tablature notation, Staff+ = annotations for execution, to be added above a staff of conventional music notation, and Charts+ = diagrams in the style of guitar chord diagrams. The tablature notation TAB+ is inspired by a proposal by Chris Sperren. [2] (See our Figure 3 for an example of our version of Charts+ and Figure 4 for TAB+. The elements of Staff+ are just numeric indications of the rows and columns to press, added above a conventional note system, and not discussed here.)

Their article discusses the "educational implications" of these notations, and describes their development in "multiple iterations [of tests] on various grid-based instruments [by] music composition and production students [...] with varying musical abilities."

In that article, the syntax and semantics are defined informally, by prose language and graphic illustrations. As a preparatory step for implementing a digital tool, an analysis of the underlying data model is useful, especially when a common data model can be identified for all three proposed formats. Furthermore, mathematical remodelling always brings more clarity also into human discourse and inexorably exhibits blurry areas of concepts. This article discusses such a data model to capture TAB+ and Charts+.

The next section describes the proposal of Wilde and White in more detail; then this is translated into a mathematical data model. The subsequent sections describe the `tScore` input format, the tool implementation, and the graphic rendering algorithms, their theory and their practical caveats.

Copyright: © 2025 Author Name. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

2. ABLETON PUSH AND TABSTAFF+

In their experiments, Wilde and White apply the proposed notations to the "Ableton Push" instrument, to other grid-based TUIs with a similar design, and to grid-based graphic user interfaces (GUIs) on touchscreens.

The Ableton Push is the origin of the described TUI and a full-fledged digital synthesizer and sample player, see Figure 1. With the same TUI input, much cheaper hardware exists restricted to the MIDI controller function. We use "Novation Launchpad MK3" to run experiments with our implementation.

The main interface realized by the Ableton Push is a matrix of eight times eight push buttons. Following the original manuals [3], the buttons are called *pads* and the matrix is called *grid* in the following. The coordinates of the pads are given as pair of numbers from 1 to 8. The order is "row before column" = "y before x", i.e. the way to address a mathematical matrix, not the Cartesian coordinates. The rows count bottom up, and the columns from left to right.

The TUI can be operated in different modes, where the pads have different meanings. These modes are called *layout* by the original hardware manual and by Wilde and White. For instance, emitting MIDI note-on and note-off events can be assigned to a pad, but also triggering the start of a prepared sound clip. The proposed notations TAB+ and Charts+ and the corresponding `tScore` input format give the physical coordinates of the pressed pads and thus are *completely independent* of the roles assigned to them. Format Staff+ is different, because it is added to conventional staves in Common Western Notation (CWN). It is not treated in this article.

Additionally the TUI hardware can have a display area and further buttons. These can be used to modify the layout on the fly while playing. In the `tScore` format these modifications are represented by events in a dedicated configuration voice named "`conf`", see below in section 4.

3. MATHEMATICAL RE-MODELLING

Table 1 shows a possible mathematical re-modelling of the information contents of TAB+ and Charts+. (As a mere analysis of the information contents of the proposals published in [1] it is independent from its `tScore` implementation as described in the next section, but it already follows the same principles.)

The first block of lines gives the data types for the events.

In `tScore` every event is identical to one particular combination of voice and timepoint. (All further event param-

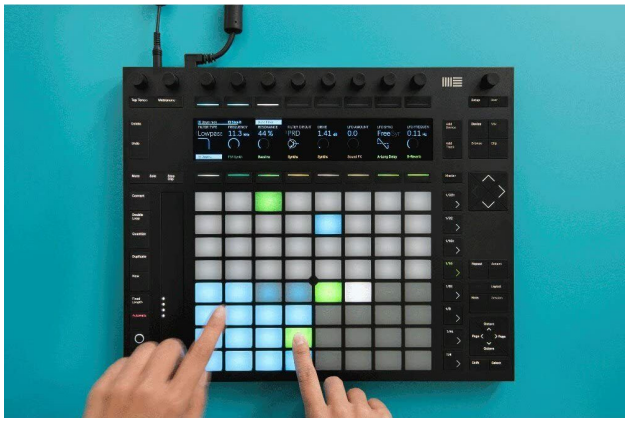


Figure 1. The "Ableton Push" synthesizher with its main input device, the pads grid. (Source: ableton.com)

$$
\begin{array}{l}
\mathsf{Hand} = \{\mathtt{md}, \mathtt{ms}\} \\
\mathsf{FNumber} = \{1..5\} \\
\mathsf{Row} = \{1..8\} \\
\mathsf{Column} = \{1..8\} \\
\mathbf{V} \qquad \textit{// given set of voices} \\
\mathbf{T} = \mathbb{N} \qquad \textit{// given set of timepoints} \\
\mathbf{E} = \mathbf{V} \times \mathbf{T} \quad \textit{// derived set of events} \\
\textit{// Parsed content of } \mathtt{tScore} \textit{ events:} \\
\mathsf{pad} : \mathbf{E} \nrightarrow \mathsf{OPT}(\mathsf{Row} \times \mathsf{Column}) \\
\qquad \textit{// = } \bot \textit{ stands for pause} \\
\mathsf{hand} : \mathbf{E} \nrightarrow \mathsf{OPT\ Hand} \\
\qquad \textit{// = } \bot \textit{ stands for "hand ad lib explicitly"} \\
\mathsf{finger} : \mathbf{E} \nrightarrow \mathsf{FNumber} \\
\mathsf{finger_X} : \mathbf{E} \nrightarrow \mathsf{OPT\ FNumber} \\
\mathsf{finger_X} = \big((\mathsf{dom\ pad} \cup \mathsf{dom\ hand}) \times \{\bot\}\big) \oplus \mathsf{finger} \\
\qquad \textit{// = } \bot \textit{ stands for "not longer valid"} \\
\\
\textit{// Totalize any function } F : \mathbf{E} \nrightarrow \mathsf{OPT}\ X\textit{:} \\
\overline{F} = \{v, t \bullet (v, t) \mapsto \mathbf{if}\ t = 0\ \mathbf{then} \perp \mathbf{else}\ \overline{F}(v, \mathsf{pred}\, t)\} \\
\qquad \oplus\ F \\
\\
\textit{// Indispensable invariants:} \\
\mathsf{dom\ finger} \subseteq \mathsf{dom}(\overline{\mathsf{hand}} \rhd \{\bot\}) \\
\mathsf{dom\ finger} \subseteq \mathsf{dom}(\overline{\mathsf{pad}} \rhd \{\bot\})
\end{array}
$$

Table 1. Syntactic and semantic data

eters are only adjoined and not constitutive.) On the time points an order is defined, and thus the notions of minimum and maximum.

The input data are partial functions from events into different domains. Here they specify the pad to press and optionally the hand and the finger to use. The range of these mappings is an optional data type: Taking the variant "no data" = $\bot$ has different meanings: Used instead of a pad it means to play a *pause*. With hand and finger it means that this is not (or no longer) prescribed.

For every event function F, the smallest fixpoint of the function definition $\overline{F}$ delivers the value specified by F at the given timepoint, or the value specified most recently before, or $\bot$. In a polyphonic context, $\overline{F}$ is especially necessary to analyze overlapping asynchronous events.

$$
\begin{array}{l}
\textit{Coord} ::= \text{"}\mathtt{1}\text{"} \mid \ldots \mid \text{"}\mathtt{8}\text{"} \\
\textit{Hand} ::= \text{"}\mathtt{r}\text{"} \mid \text{"}\mathtt{R}\text{"} \mid \text{"}\mathtt{D}\text{"} \mid \text{"}\mathtt{d}\text{"} \\
\qquad \mid \text{"}\mathtt{S}\text{"} \mid \text{"}\mathtt{s}\text{"} \mid \text{"}\mathtt{L}\text{"} \mid \text{"}\mathtt{l}\text{"} \mid \text{"}\mathtt{X}\text{"} \\
\textit{Finger} ::= \text{"}\mathtt{1}\text{"} \mid \ldots \mid \text{"}\mathtt{5}\text{"} \\
\textit{Pause} ::= \text{"}\mathtt{\%}\text{"}\ \textit{Hand}^? \\
\textit{Keypress} ::= \textit{Coord}\ \textit{Coord}\ \textit{Hand}^?\ \textit{Finger}^? \\
\textit{FChange} ::= \textit{Hand}\ \textit{Finger}^? \mid \text{">"}\ \textit{Hand}^?\ \textit{Finger}^? \\
\textit{Event} ::= \textit{Pause} \mid \text{"}-\text{"} \mid \textit{Keypress} \mid \textit{FChange}
\end{array}
$$

Table 2. Grammar rules for `tScore` events

The finger numbers are not "sticky" but only valid for the one notated event. This must be modelled explicitly by deriving $\mathsf{finger_X}$.[1]

The last two lines give invariants, the violation of which leads to a rejection by the implementation: fingers may not be specified without a hand, and fingers may not be specified for pauses. (See appendix A below, which discusses some parallels in conventional CWN piano notation.)

4. TSCORE INPUT FORMAT

`TScore` is a concept and an implementation for notating arbitrary data models which are organized along a time axis. It can be written conveniently and fast, and is equally well readable by computers and humans. [4] [5]

As usual in computer engineering, it is called a *meta-meta-model*. It allows easy definition of *meta-models*, which define the input grammar, the data structures, semantical invariants, and processing paths for a collection of encoded pieces, the *models* finally. (In this sense the formulas in Table 1 are a meta-model for all TABstaff+ data.) For a typical application to graphic avant-garde notation see [6]; for a collection of very different examples see [7].

The `tScore` meta-model which covers the TABstaff+ data model, our versions of rendering to tablature and charts, and the Java implementation, are called `squarePads`. Table 2 shows the grammar of the events in the `tScore` input, and Figure 2 shows a simple example.

The voice named "`conf`" determines the grid layout, and thus assigns roles to the pads of the grid. With the very first event of this voice some attributes are necessary. Later events may carry subsets of these parameters, to change the layout dynamically during the performance, see section 8.3.

The voice named "`nota`" plays different roles for different output media: It can determine the staves and clefs shown in tablature notation, and the size of the generated charts.

All other voices (with arbitrary but distinct names) hold the information what to play: Every event is either a pause "`%`", or a prolongation "–" of the preceding event, or a pair of row and column number, meaning a pad press. (Normally a pad is meant to be held down until the next event, but of course this may be superfluous for contexts where

[1] The name of the domain "FNumber" has been chosen to stress that it does not stand for a real-world "finger". For this you need a pair Hand × FNumber.

```
PARS sola
lingua=en
formaSuprema=eu.bandm.music.applications.squarepads.SquarePads
matrixClavium.tabulatura.vocesTangentes = faciliusLectu
// = Shift adjacent noteheads from different voices apart for easier reading.
matrixClavium.tabulatura.differe = (-1, +1)
// = On conflicts shift upper voice to the left and lower to the right.

T           0                                   1
VOX conf    horizontal+&fixed-&inKey-&thirds&minor&c2
VOX nota    ABαβ
VOX v1      11      21L3    21L3       >4      %
VOX v2      51R     51R1    >R2        L       %

          // Means
          // First row first column, hand and finger ad lib.
          // Fifth row first column, right hand, finger ad lib.
                  // Second row first column, third finger of left hand.
                  // Fifth row first column, first finger of right hand.
                          // Re-attack same button with same finger.
                          // Silently change to second finger of right hand.
                                    // Silently change to finger 4,  same hand.
                                    // Silently change to left hand, any finger.
```

Figure 2. A small `squarePads` `tScore`. The corresponding chart rendering is in Figure 3.

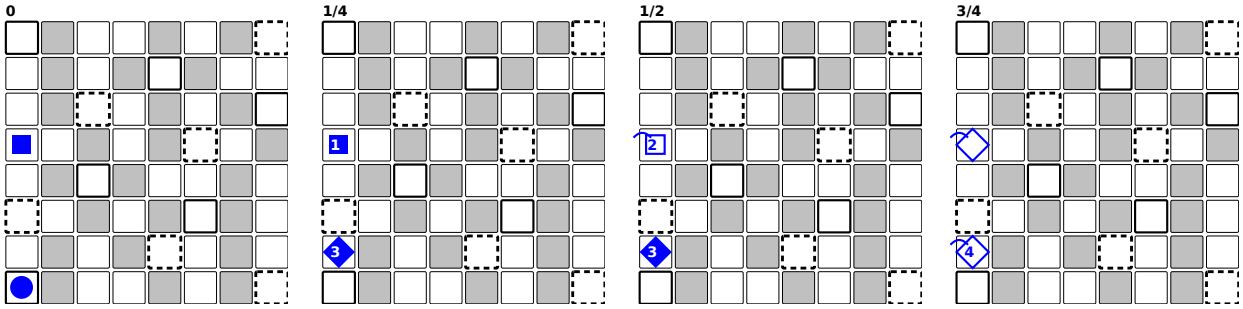


Figure 3. Rendered chord charts: hand is ○ unspecified □ right ◇ left

only the onset of the pressing [= the MIDI note-on event] is evaluated as a trigger.)

The pause symbol and the pad coordinates may be followed by a letter indicating the playing hand.

A pad coordinate can be followed by a finger number, from "1" to "5", as with piano notation. The hand information is treated sticky: it stays valid until explicitly ended by another indication, which may be "X", meaning "hand unspecified".

A finger number may only be added when a hand is currently specified, by this or an earlier pad-press event.

The event ">" indicates a silent finger change. It may be followed by a hand indication or a finger number or both. The preceding event must not be a pause or its prolongation.

The `tScore` input format is basically *monophonic*: every combination of timepoint and voice can carry at most one single pad information. Input formats which allow *chords* of pressed' pad are included in the implementation as an alternative, but this is only "syntactic sugar" and is resolved to automatically supplied additional voices.

5. CRITICAL PROPERTIES

One of the most important roles of a mathematical model as shown in Table 1 is as a basis for precise definitions of model properties and thus categories. Usually an application context or a historic or personal style of writing is naturally characterized by a restriction to a particular subset of models. Such a characterization can be discovered by algorithmic filtering by the defined properties. [2] Such properties are useful not only for notated music, but also for interactive performances, as long as they are translated into the data model.

Important for new kinds of manual input devices, but also for particular styles of writing for conventional keyboard, are PressPattern and DIGITINONFRACTI: they define the geometric arrangements of multiple pads which must be pressed by the same finger, and whether a composition or performance sticks to it. pressPattern_easy is an example where one single pad or two or four all-adjacent pads are pressed.

6. CHART NOTATION

The current demo implementation generates graphic representations which resemble the chord charts as widely used for guitars etc.,as proposed by Wilde and White. The di-

[2] The names of all properties are chosen from the Latin language to fit nicely into the system proposed as LMN [8]. Read as an LMN module, the complete name of all these properties is prefixed by MATRIXCLAVIUM.

$t : \mathbf{T}, r : \mathsf{Row}, c : \mathsf{Column}, h : \mathsf{Hand}, f : \mathsf{FNumber}$

$\textsc{clavisPerVoces}(t, r, c) = \{v \mid \overline{\mathsf{pad}}(v, t) = (r, c)\}$
// = set of voices targeting the given pad simultaneously
$\textsc{perNotasMultas} = \{t, r, c \mid \#\textsc{clavisPerVoces} > 1\}$
// = pads pressed by multiple voices simultaneously
$\textsc{clavesPerNotamSimplicem} \iff \textsc{perNotasMultas} = \varnothing$
// = every pad appears at every timepoint not more than once

$\textsc{clavisPerDigitos}(t, r, c) = \{v \mid \overline{\mathsf{pad}}(v, t) = (r, c) \bullet (\overline{\mathsf{hand}}(v, t), \overline{\mathsf{finger}_{\mathrm{X}}}(v, t))\} \setminus (\mathsf{Hand} \times \{\bot\})$
// = set of fingers which press a pad simultaneously
$\textsc{clavisPerDigitosMultos} = \{t, r, c \mid \#\textsc{clavisPerDigitos}(t, r, c) > 1\}$
// = set of pads which pressed by multiple fingers simultaneously
$\textsc{clavesSimplices} \iff \textsc{clavisPerDigitosMultos} = \varnothing$
// = no pad is ever pressed by multiple fingers simultaneously

$\textsc{digitusInClaves}(t, h, f) = \{v \mid \overline{\mathsf{hand}}(v, t) = h \wedge \overline{\mathsf{finger}_{\mathrm{X}}}(v, t) = f \bullet \overline{\mathsf{pad}}(v, t)\}$
// = set of all keys pressed by the given finger
$\textsc{digitusInMultosClaves} = \{t, h, f \mid \#\textsc{digitusInClaves}(t, h, f) > 1\}$
// = set of fingers which press multiple pads simultaneously
$\textsc{digitiSimplices} \iff \textsc{digitusInMultosClaves} = \varnothing$
// = no finger ever presses multiple pads simultaneously

$\textsc{manusCertae} = \bot \notin \mathsf{dom}\ \overline{\mathsf{hand}}$
// = for every pad press and every pause the hand is (sticky) prescribed
$\textsc{pausaCumManu} = (\mathsf{pad}^{\sim}(\!|\{\bot\}|\!)) \triangleleft \mathsf{hand} \neq \varnothing$
// = at least one pause carries an explict hand prescription.
$\textsc{digitiCerti} = \mathsf{dom}\ \mathsf{pad} \subseteq \mathsf{dom}\ \mathsf{finger}$
// = for every pad press the finger is prescribed
$\textsc{sineDigitis} = (\mathsf{finger} = \varnothing)$
// = for no single pad press the finger is prescribed
$\textsc{digitiCerti} \implies \textsc{manusCertae}$

// Indispensible invariants by physiology:
$\mathsf{PressPattern} = \mathbb{PP}(\mathbb{N} \times \mathbb{N})$
$\mathsf{pressPattern_easy} = \{\{(0,0)\}, \{(0,0),(0,1)\}, \{(0,0),(1,0)\}, \{(0,0),(0,1),(1,0),(1,1)\}\}$
// = set of patterns of pads which can be pressed simultaneously by one finger.
$instantiatePattern : \mathsf{PressPattern} \to \mathbb{PP}(\mathsf{Row} \times \mathsf{Column})$
$instantiatePattern(P) = \bigcup p \in P \bullet \{a, b : \mathbb{N} \bullet \{(c, d) \in p \bullet (a + c, b + d)\} \cap \mathbb{P}(\mathsf{Row} \times \mathsf{Column})\}$

$\textsc{digitiNonFracti}(P : \mathsf{PressPattern}) = \forall t, h, f \bullet \textsc{digitusInClaves}(t, h, f) \in instantiatePattern(P)$
// = indicates that the whole score requires to play not more than the given patterns.
$\textsc{digitiNonFracti}(\{\{(0,0)\}\}) \iff \textsc{digitiSimplices}$

Table 3. Characterizing properties w.r.t. fingers and hands.

atonic pitches, the chromatic steps between, and the root class pitches are represented by combinations of the background color and outline thickness of the pad symbols. This is in analogy to the state of the color LEDs in the hardware pads.

The pads pressed by the left and right hand are marked by rectangle symbols, upright for right hand, diamond-shaped for the left hand. Circles stand for notes without hand indication. Finger numbers are centered, silent finger changes are symbols with outlines only and no fill.

The current implementation generates single SVG files, see Figure 3, and one animated SVG file, assuming BPM 60.

7. TABLATURE NOTATION

The principle of tablature notation has been well-known for more than five centuries, in very different variants. [9] It is a notation of the *physical operation* on fretted string instruments: The lines of the staff represent the strings; the horizontal position from left to right represents the flow of time; the numbers put into the staff represent the fret to press, plus possibly additional symbols like finger indication and *laissez vibrer*.

Much later, this format has been transferred to key instruments (Klavarskribo [10], piano roll [11], etc.)

Transferring this format to a two-dimensional TUI seems sensible, but brings two major challenges:

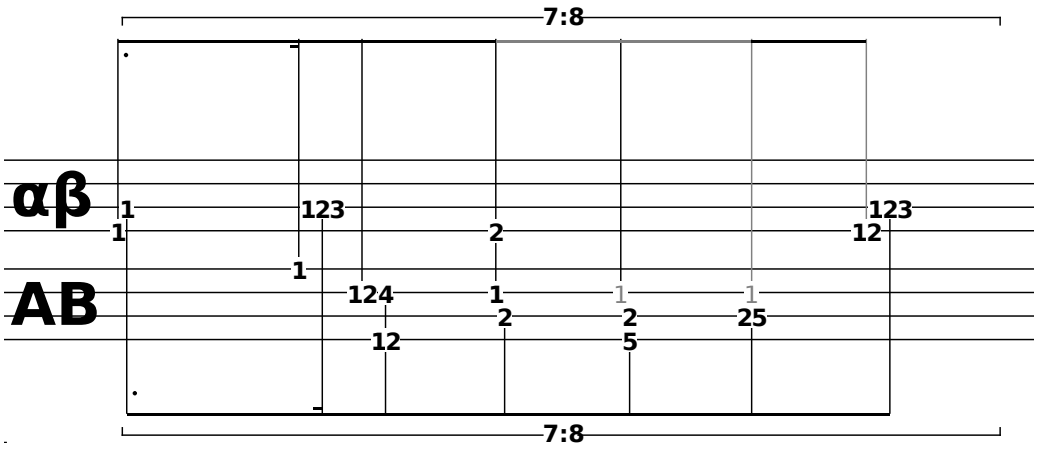


Figure 4. Collision Corrections in the Rendered Tablature Notation

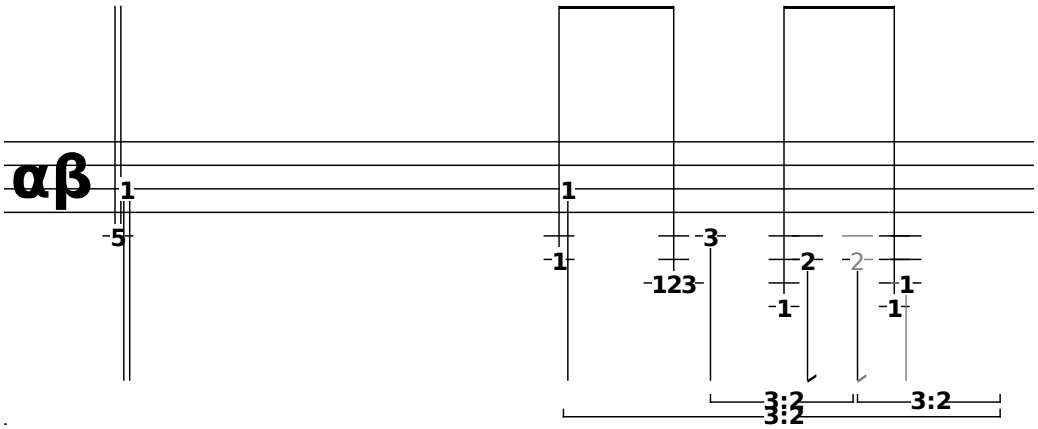


Figure 5. Ledger Lines in the Rendered Tablature Notation

(A) The staff lines stand for the rows of the grid, the numbers therein for the columns to press. In contrast to frets, *multiple* numbers on the same combination of lineola and timepoint appear frequently.

(B) In particular contexts, we may wish to notate both hands in the same staff, with upward vs. downward stems.

Both these new properties cause the need for *collision correction* when programming automated graphical layout.

Furthermore, (C) Wilde and White propose *ledger lines* for spurious outbreaking events, when normally only the lower or the upper half of the grid is used and represented by four lines. (The staff lines in historic tablature notation systems represent *all* stopped strings of the instrument—we have not found any example which uses ledger lines.)

7.1 Graphic Positioning Problems in Detail

In `squarePads`, we have implemented the following strategy for the graphic appearance: The relation of the column numbers to one of the two voices (right hand vs. left hand = up stem vs. down stem) is always unambiguously clear because the sequence of numbers is printed centered on the stem, see the sequence "124" in Figure 4. (An alternative is to place the numbers to the left or right, like noteheads in conventional staves, but this turned out much less readable. See [2] for examples.)

Tying noteheads as in CWN can create hardly solvable collisions among the arcs and between arcs and staff lines. Instead, in our rendering all non-leading tied noteheads (which do not represent a new attack but merely a longer holding of the pad) are printed in gray instead of black.

In case of collisions, the two stems must be moved apart by a certain distance. This distance is calculated according to the formulas in Table 4. When noteheads overlap, the shift is unavoidable, see the example in Figure 4. Figure 5 shows an example with ledger lines and half notes indicated by double stems.

This is different when the two columns of numbers only touch, without overlapping. In Figure 6, the two voices in example (a) are unambiguous and printed without any shift.

Contrarily, columns (b) and (c) would be ambiguous if printed without shift, because they contain multiple neighboring pairs of rows each.

Examples (d) and (e) are not ambiguous, but probably hard to read prima vista. So they get a shift if the configuration parameter TABULATURA.VOCESTANGENTES takes the value FACILIUSLECTU.

The notation property TABULATURA.DIFFERRE indicates for both voices in which horizontal direction their stem will contribute to a necessary shift.

$$
\begin{array}{l}
nhUp, nhDown : \mathsf{Row} \to \mathbb{P}\mathsf{Column} \\
U = \mathsf{dom}\ nhUp \rhd \varnothing \\
D = \mathsf{dom}\ nhDown \rhd \varnothing \\
A = U \cup D \\
i = \mathsf{min}\ U \\
a = \mathsf{max}\ D \\
corr = \textbf{if}\ i > a + 1\ \textbf{then}\ 0 \\
\qquad \textbf{else if}\ i = a + 1\ \textbf{then}\ touching \\
\qquad \textbf{else}\ \mathsf{max}\ \{r \in i..a \bullet \#nhUp(r) + \#nhDown(r)\}
\end{array}
$$

$$
\begin{array}{l}
\textsc{matrixClavium.tabulatura.vocesTangentes} : \{\textsc{faciliusLectu}, \textsc{necessaria}\} \\
touching = \textbf{if}\ \textsc{vocesTangentes} = \textsc{faciliusLectu}\ \textbf{then}\ [\ \#A > 2\ ] \\
\qquad \textbf{else}\ [\ \#A \cap succ(\!|A|\!) > 1\ ]
\end{array}
$$

$$
\begin{array}{l}
Shift = \{-1, 0, +1\} \\
\textsc{differre} : (Shift \times Shift) \setminus \mathbf{ID}_{Shift} \\
x_{\mathrm{up}} \quad = x + corr * \pi_1(\textsc{differre}) * ([\pi_2(\textsc{differre}) = 0] + 1) * WIDTH_OF_DIGITS/4 \\
x_{\mathrm{down}} \quad = x + corr * \pi_2(\textsc{differre}) * ([\pi_1(\textsc{differre}) = 0] + 1) * WIDTH_OF_DIGITS/4
\end{array}
$$

Table 4. Tablature collision correction, at one particular time point with nominal horizontal position x.

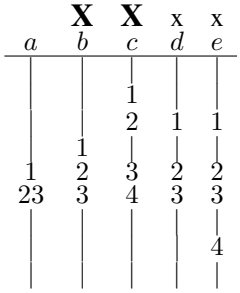


Figure 6. Real ambiguities (**X**) vs. presumed reading problems (x), caused by adjacent noteheads



Figure 7. Proposed noteheads to indicate columns graphically: (A) one event to press all pads in a row of eight columns. The design of the arrow heads requires one additional gap only in two cases: "4 (5) appears without direct neighbors but with successors (predecessors)". See (B) which addresses columns 2, 4, 6, and 8.
(C) and (D) show an event which presses all keys in a row in a four-column grid, in two different notation proposals.

Figure 4 shows the different collision cases. (In the current implementation, only collisions at the *same timepoint* are corrected. All timepoints are arranged strictly proportionally, and calculating the overall width is not yet automated in `tScore` anyhow.) [3]

Generating the flags, beams, proportion brackets, and prolongation dots and ties is done by metricSplit, our algorithm which can render any sequence of rational duration values to CWN. [12, 13]

The attempt to add fingering information by further decimal digits would imply hard-to-read overloading. (Numbers 1–5 are usual in keyboard notation, letters p, i, m, a in right-hand guitar.) An alternative is to use graphic symbols as noteheads for column information, for instance as in Figure 7. But practical experience is still missing.

8. PITCH LAYOUT AND ITS DYNAMIC CHANGE

We mentioned above, that the pad press events in TAB+, Staff+, and `tScore` are notated by their physical coordinates and thus completely independent from the meanings of the pads, and that the relation from all 64 grid pads to their meanings is called *layout* by the original hardware manual. Some of these layouts are called *64 notes layout* [3, section 7]: All $8 * 8$ pads generate MIDI note-on and note-off events; each pad is assigned to one particular MIDI key number; its lighting indicates its role in the currently intended diatonic scale and appears on the LEDs of the physical device and in the graphic boxes produced by Charts+. The possible configurations will be called *pitch layouts* in the following.

A challenge for re-modelling is the specification of the *dynamic changes*: While a session is running, the player can press dedicated buttons to change the parameters of the pitch layout on the fly. This is realized by `tScore`

[3] The demo application is currently just a proof of concept: Only one measure is rendered, which must extend from timepoint $0/1$ to $1/1$ and be in a $4/4$ meter. By resizing the window of the application, you can explore the visual appearance at different display widths.

by the dedicated voice "conf" which contains dedicated commands, see Figure 2 above.

While this pitch layout is independent from all notation systems described above, it must be implemented whenever a sound synthesis shall be addressed by some software. Furthermore, its remodelling (with mathematical means) by interpreting the original specification (in plain language only) is instructive and thus included in this article as a second, independent part.

8.1 The Original Specification

The Ableton Push 3 Manual [3] gives only an informal and very brief description how a pitch layout is changed dynamically:

> 7.1 Playing in Other Keys
> You can press the Scale button to switch to a different key and scale. You will see the available keys and scales in the display.
>
> Use the [..] buttons to select the key [and] to select a scale.
>
> The leftmost encoder changes the layout of the pad grid:
> 4ths – When moving up to the next pad vertically, each pad is a fourth higher.
> 3rds – When moving up to the next pad vertically, each pad is a third higher.
> Sequent – When moving to the right, each pad is in order based on the selected scale. [4]
>
> [..] When set to In Key, only the notes that are part of the selected key will be available on the pads. When set to Chromatic, all notes can be played; however, pads that contain notes that are not part of the key will be unlit.
>
> [..] When Fixed is on, the notes of the pad grid remain in the same position when you change keys, e.g. the bottom-left pad will always play C (unless the key does not contain a C, in which case the pad will play the nearest note to C). When Fixed is off, the notes on the pad grid shift so that the bottom-left pad always plays the selected root note.

Fundamental questions are left open:

(Q1) What are a "third" and a "fourth"? (This question is deeply related to the structures of the selectable "scales", which are not mentioned at all in the above-quoted text. The illustrating screen shot shows that a "Whole Tone scale" is offered. This does not contain any (pure) fourth!)

(Q2) What does "nearest note to C" mean?

(Q3) Switching the root pitch in non-fixed mode can bring any pitch x to the "bottom-left" pad $(1, 1)$. Then changing (while continuing the session) to fixed mode and then switching the root again will probably leave x on position $(1, 1)$, not "C"?

(Q4) Does the "Fixed" behavior of letting the maximal number of pitches in place also apply when changing the distance?

[4] This cannot be a definition of this mode because its always true in horizontal mode. The "push-2 manual" at https://www.ableton.com/en/manual/using-push-2/ says "The Sequent layout puts all notes sequentially in order." Indeed the right-most pad in every row is one pitch below the left-most pad in the row above, if any.

There are different possibilities to deal with such an incomplete specification. In the following we try to *explicate* it, by constructing different alternative interpretations and their algorithmic implementations. This allows to explore their limitations and consequences.

An alternative would be to do *reverse engineering* of the commercial product. But in contrast to a specification, its behavior cannot be expected to stay stable, and the discussion would be restricted to the single solution chosen by the engineers of that particular product and possibly miss the more fundamental rules and dependencies.

8.2 Static Pitch Layout by Parameters

When creating a pitch layout (= assignment of midi pitches and lighting states to pads) from scratch, it is defined completely by the parameters scale, root, distance (appearing as "4ths", "3rds", and "Sequent" in the manual text quoted above) , InKey (see "In Key" above), and orientation, as declared in Table 6. (All following text covers only the case orientation $=$ `horizontal`. The vertical layout can be derived easily by post-processing, namely by swapping rows and columns.)

(In the following the term *root* is reserved for the pitch class used as the root of the scale, and *base pitch* stands for the concrete pitch which is given as parameter and is assigned to the offset zero, as used technically in our model in Tables 6 and 7.)

All these parameters must be defined to create an initial (pitch) layout, and they define it completely. The algorithm has only to answer question (Q1) from the list above.

But these parameters can also be altered by the user dynamically, while playing. (We assume that only the change of one of these parameters occurs at a particular time point, but a second change can follow arbitrarily soon.)

In this case we call the currently active pitch layout the *preceding layout*. There is a meta-parameter Fixed of boolean type. When it is set to `false`, the new pitch layout is created from the current parameter values as if it were an initial one. But when Fixed $=$ `true`, then the new pitch layout is derived from the preceding layout, to minimize the number of pads affected.

Tables 6 and 7 start with a possible algorithm for the construction of an initial layout: The function $K_{\mathrm{R}}(s, k, S, c)$ delivers the MIDI offset (relative to the base pitch) pitch for the pad in column c, when the row starts with offset s, the InKey mode is k and S is the selected scale. Any scale is given as a pair of a set of indexes from 0 to 11 and a single such index. The set is similar but not totally equivalent to the well-known *pitch class set*. It indicates the steps of the chromatic scale which are contained in S. Step number 0 is contained by default. When the second component is set to a number $\neq 0$, it indicates a step of the scale which shall be highlighted as a *repercussa*, which is the somehow second-important step of the scale. This highlighting is our extension for a better orientation on the grid of pads.

For $k =$ `false` the chromatic pitches follow in dense order starting with s; for $k =$ `true` the list of all pitches of the scale $pitches(S)$ is constructed. Then this is restricted to begin with the given start point of the row, and indexed

by the column number c. (In Z in general, and as used in this article, sequences start with the index 1, see the appendix.)

The role of the pad (for their lighting) is derived from the MIDI pitch by the function $L_{\mathrm{R}}(s, k, S, c)$. Please note that none of the two functions needs to know the currently ruling value of the base pitch—they work with indices relative to the current base pitch. The MIDI key numbers effectively emitted with pad press events are calculated by adding the current value of base pitch to this index.

It is more problematic to find the first pitch for every row, realized by the vector V:
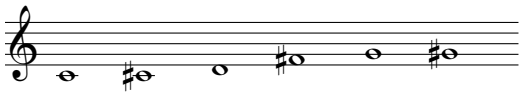
In every *initial* construction, the pad at $(1, 1)$ is assigned the selected base pitch—thus to the relative coordinate 0. When distance $=$ `sequent` then the next higher row will seamlessly continue the sequence of pitches and roles. In the other cases, a pitch "a third above" or "a fourth above" must be found. The function $N(s, k, S, d)$ delivers the start point for the next higher row of a row starting with s—again k is the InKey mode, S the scale, and d the selected distance.

N_1 to N_3 implement *variants*: N_1 delivers the next diatonic key with a distance equal or larger than a minor third or a fourth, respectively, above s.

N_2 is only applicable to InKey $=$ `false` and distance $=$ `fourths`, and simply goes five chromatic steps up. This is the only case not using `sequent` where a non-diatonic pitch can appear in the first column.

N_3 takes the historic origin of the interval names in a verbatim way and delivers the second or third successor of s, relative to S.

All variants can be appropriate to particular use cases. For the scale S starting with the pitch classes



it holds that

$$N_1(0, _, S, \texttt{thirds}) = N_1(0, _, S, \texttt{fourths}) \equiv \mathrm{F}\sharp$$
$$N_2(0, \texttt{false}, S, \texttt{fourths}) \equiv \mathrm{F}$$
$$N_3(0, _, S, \texttt{thirds}) \equiv \mathrm{D}$$
$$N_3(0, _, S, \texttt{fourths}) \equiv \mathrm{F}\sharp$$

The function initV creates the vector of type V of the starts of all rows by recursively calling N.

The current global state of the matrix is completely captured by an instance from G. Function initG creates such a state for the given parameters, from scratch, and $K(G, r, c)$ and $L(G, r, c)$ deliver the MIDI key number and the lighting for the pad in row r and column c.

8.3 Dynamic Change of Pitch Layout Parameters in Fixed mode

The dynamic changes of pitch layout parameters, i.e. during playing, when the device is in a particular pitch layout state, can be much more complicated. With the meta-parameter Fixed set to `false`, it has the same effect as creating a pitch layout from scratch, as described in the preceding section. But setting Fixed to `true` aims at a layout change with a minimal number of pads affected—to realize a "principle of minimal changes." This is ergonomically sensible: For instance, when changing the playing context only by shifting the root pitch class up by one or two positions in the circle of fifth, the player often wants to continue playing the common (and thus un-affected) pitches on the same pads as before. This corresponds to the handling of a traditional piano or organ keyboard, where only (in the mental reservoir maintained by the player) the white key f is replaced by the black key f-sharp, etc.

change…	…affects MIDI	…lighting
chromatic mode, $k =$ `false`		
base pitch	-	X
scale	-	X
distance	H	H
diatonic mode, $k =$ `true`		
root	X	-
scale	X	(x)
distance	H	H
diatonic $\longleftrightarrow$ chromatic	X	X

Table 5. Pitch layout parameters and the effects of changing them in Fixed mode. X = always changes. H = always changes, but not the lowest row. (x) = possibly, when lengths or repercussa of scales differ, - = no change.

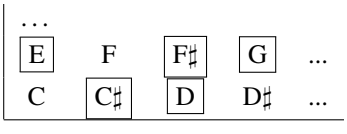
While the basic idea appears self-evident and ergonomic at first sight, its realization is complicated and indeed brings up inherent contradictions. Table 5 gives a survey of the minimal and possible effects. We affirm question (Q3) from above and thus include the vector $v : V$ in the global state G, to satisfy all change requests in the context of the preceding layout. The corresponding function change(g, x) in Table 7 is *overloaded*: The *type* of its second parameter x determines which component of the global state g shall be changed.

Taking the original text-only specification (as cited above) seriously, results in complicated behaviour. You can use the application (see section 9) for own experiments—no special hardware is required.

8.3.1 InKey $=$ `false`

In chromatic mode, changing scale and root is easy: All pads keep their pitch role, only the lighting changes, according to a shifted version of the corresponding initial layout. (In the formula, only the vector of the row starting points $v : V$ must be adjusted to compensate the change to the new pitch base q when emitting MIDI keys.)

We affirm question (Q4): Changing distance will change all pad assignments except in the lowest row. It is problematic due to question (Q1) from the list above, and because pad $(1, 1)$ is not always diatonic. The pitch layout



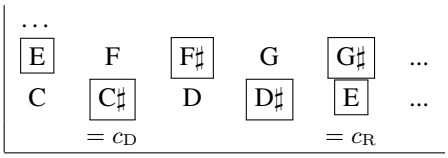
is reached when starting in C major and changing in fixed mode the root to D.

Our solution treats the root pad in the lowest row (its index is c_R in Table 7; $pitches((\varnothing, 0), x)$ is the simply the sequence of all multiples of 12 which are equal or greater than x.) in the same way as the initial construction treats pad $(1, 1)$, and then calculates back for the first columns.

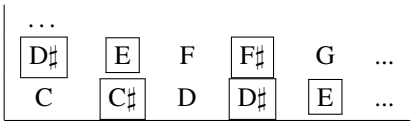
If no root pad appears in the bottom row, the first diatonic pad is taken as that anchor, its index is c_D.

Our first programming attempt used only this pad (by setting $c_1 = c_\mathrm{D}$), but this is not a good and stable solution for all cases, because (only) the distance `thirds` depends on the structure of the scale:

The sequence:
power-up configuration $\longrightarrow$ Fixed $=$ `true` $\longrightarrow$ root $=$ E
results in



Then switching forth and back with
$\longrightarrow$ distance $=$ `fourths` $\longrightarrow$ distance $=$ `thirds`
with c_D as the fixpoint would result in



So we introduced c_R as the normal case, and take c_D only as a (rarely necessary) fall-back.

8.3.2 InKey $=$ `true`

In diatonic mode the changes to all parameters S, q, and d are problematic because nearly always the set of represented pitches changes—not all pads can keep their MIDI pitches. And (Q2) from the list of questions above (what is the "nearest pitch to C"?) must be answered.

On the other hand there are numerous *combinations* of changes which do not change the set of pitches, but only the ighting: Mixolydian on G holds the very same pitches as Lydian on F. Therefore we define a function change which can process a new scale and a new root in one step, and can be called with $q = q'$ or $S = S'$. (Especially a change of the root pitch but not the pitch class will have no effect at all!)

To leave as many pitches in place when changing root or scale, we search in every row from left to right for the first pad with a pitch present in both scale–root combinations. (P contains all new pitches, relative to the current root: add the new root to the new scale components to get MIDI pitches, and then subtract the old root to get relative indexes again. P' is the sequence of root-relative indexes the MIDI pitch of which will survive, and $w(r)$ is the first such index found in row r. Then $c(r)$ is the column number of that survivor, and the new start of the row $v'(r)$ is found $c(r)$ positions below $w(r)$ in the new scale.

Changing the scale in Fixed mode has always common pads, namely those of the root pitch class. Changing the root possibly has no common pad: Shift a whole-tone scale a half-tone up! In case $P' = \varnothing$ our implementation searches

// Universal types and data:

$$\mathsf{orientation} = \{\texttt{horizontal}, \texttt{vertical}\}$$
$$\mathsf{InKey} = \mathsf{boolean}$$
$$\mathsf{scale} = \mathbb{P}\,\{1..11\} \times \{1..11\}$$
$$\mathsf{distance} = \{\texttt{thirds}, \texttt{fourths}, \texttt{sequent}\}$$
$$\mathsf{roots} = \langle \mathsf{c}, \mathsf{c\sharp}, \mathsf{d}, \mathsf{d\sharp}, \mathsf{e}, \mathsf{f}, \mathsf{f\sharp}, \mathsf{g}, \mathsf{g\sharp}, \mathsf{a}, \mathsf{a\sharp}, \mathsf{b} \rangle$$
$$\mathsf{pitch} = (\mathsf{ran\ roots}) \times \mathbb{N}$$

// Calculate the key and lighting for one row:

$$\mathsf{role} = \{\texttt{finalis}, \texttt{repercussa}, \texttt{own}, \texttt{foreign}\}$$
$$sort : \mathbb{PN} \to \mathsf{seq}\,\mathbb{N}$$
$$sort(A) = squash(\mathbf{ID}_A)$$

$$\frac{S : \mathsf{scale} \qquad S' = \pi_1 S \cup \{0\} \cup \{\pi_2 S\} \qquad \widehat{S} = \bigcup_{o \in \mathbb{N}} (_ + 12 * o)(\!|S'|\!)}{pitches(S) = sort(\widehat{S})}$$
$$pitches(S, s) = sort(\widehat{S} \cap \{s..\})$$

$$K_\mathrm{R} : \mathbb{N} \times \mathsf{boolean} \times \mathsf{scale} \times \mathsf{Column} \to \mathbb{N}$$
$$L_\mathrm{R} : \mathbb{N} \times \mathsf{boolean} \times \mathsf{scale} \times \mathsf{Column} \to \mathsf{role}$$
$$K_\mathrm{R}(s, \texttt{false}, _, c) = s - 1 + c$$
$$K_\mathrm{R}(s, \texttt{true}, S, c) = pitches(S, s)(c)$$

$$\frac{p = K_\mathrm{R}(s, k, S, c)\ \mathsf{MOD}\ 12}{L_\mathrm{R}(s, k, S, c) = \begin{cases} \textbf{if } p = 0 & \textbf{then } \texttt{finalis} \\ \textbf{else if } p = \pi_2(S) & \textbf{then } \texttt{repercussa} \\ \textbf{else if } p \in \pi_1(S) & \textbf{then } \texttt{own} \\ \textbf{else} & \texttt{foreign} \end{cases}}$$

// Calculate the start of the next upper row:

$$N_\square : \mathbb{N} \times \mathsf{boolean} \times \mathsf{scale} \times \mathsf{distance} \to \mathbb{N}$$

$$N_\square(s, \texttt{false}, _, \texttt{sequent}) = s + 8$$
$$N_\square(s, \texttt{true}, S, \texttt{sequent}) = pitches(S, s)(9)$$

$$N_\square(_, \texttt{true}, S, _) \in pitches(S)$$
$$s \in pitches(S) \wedge d \neq \texttt{sequent} \implies N_{1,3}(s, \texttt{false}, S, d) \in pitches(S)$$

$$N_1(s, _, S, \texttt{thirds}) = pitches(S, s + 3)(1)$$
$$N_1(s, _, S, \texttt{fourths}) = pitches(S, s + 5)(1)$$

$$N_2(s, \texttt{false}, _, \texttt{fourths}) = s + 5$$

$$N_3(s, _, S, \texttt{thirds}) = pitches(S, s)(3)$$
$$N_3(s, _, S, \texttt{fourths}) = pitches(S, s)(4)$$

// Global state of the pitch layout:

$$V = \mathsf{Row} \to \mathbb{N}$$
$$\mathsf{initV} : \mathbb{N} \times \mathsf{boolean} \times \mathsf{scale} \times \mathsf{distance} \to V$$

$$\frac{v : V \quad r > 1 \qquad v(r) = N_\square(v(r-1), k, S, d) \qquad v(1) = c_0}{\mathsf{initV}(c_0, k, S, d) = v}$$

$$G = \mathsf{pitch} \times V \times \mathsf{boolean} \times \mathsf{scale} \times \mathsf{distance}$$
$$\mathsf{initG} : \mathsf{pitch} \times \mathsf{boolean} \times \mathsf{scale} \times \mathsf{distance} \to G$$
$$\mathsf{initG}(q, k, S, d) = (q, \mathsf{initV}(0, k, S, d), k, S, d)$$

Table 6. Pitch layout parameters and evaluation rules, part one.

// Look-up MIDI key values and lightimg:

$$[\![_]\!] : \mathsf{pitch} \to \mathbb{N}$$
$$K : G \times \mathsf{Row} \times \mathsf{Column} \to \mathbb{N}$$
$$L : G \times \mathsf{Row} \times \mathsf{Column} \to \mathsf{role}$$

$$[\![(p, o)]\!] = \mathsf{roots}^{\sim}(p) - 1 + 12 * (o + 1)$$
$$K((q, v, k, S, d), r, c) = K_{\mathrm{R}}(v(r), k, S, c) + [\![q]\!]$$
$$L((_, v, k, S, d), r, c) = L_{\mathrm{R}}(v(r), k, S, c)$$

// Dynamic changes in chromatic mode:

$$\mathsf{change}((q, v, \mathtt{false}, S, d), q') = (q', v \fatsemi (\lambda x \bullet x + q - q'), \mathtt{false}, S, d)$$
$$\mathsf{change}((q, v, \mathtt{false}, S, d), S') = (q, v, \mathtt{false}, S', d)$$

$$\frac{\begin{array}{c} c_{\mathrm{R}} = pitches((\varnothing, 0), v(1))(1) \\ c_{\mathrm{D}} = pitches(S, v(1))(1) \\ c_1 = \textbf{if } c_{\mathrm{R}} > v(1) + 7 \textbf{ then } c_{\mathrm{D}} \textbf{ else } c_{\mathrm{R}} \\ v' = \mathsf{initV}(c_1, \mathtt{false}, S, d') \\ v'' = v' \fatsemi (\lambda x \bullet x - c_1 + v(1)) \end{array}}{\mathsf{change}((q, v, \mathtt{false}, S, d), d') = (q, v'', \mathtt{false}, S, d')}$$

// Dynamic changes in diatonic mode:

$$\frac{\begin{array}{c} P = (\lambda x \bullet x - q + q')(\!|\mathsf{ran}\ pitches(S')|\!) \\ P' = pitches(S) \triangleright P \\ w(r) = squash(P' \triangleright \{v(r)..\})(1) \\ c(r) = pitches(S)^{\sim}(w(r)) - pitches(S)^{\sim}(v(r)) \\ v'(r) = pitches(S')\big(pitches(S')^{\sim}(w(r)) - c(r)\big) \\ v_1 = \min(P \cap \{v(1)..\}) + q - q' \end{array}}{\mathsf{change}((q, v, \mathtt{true}, S, d), q', S') = \begin{cases} \textbf{if } P` \neq \varnothing & \textbf{then } (q', v', \mathtt{true}, S', d) \\ \textbf{else} & (q', \mathsf{initV}(v_1, \mathtt{true}, S', d), \mathtt{true}, S', d) \end{cases}}$$

$$\frac{v' = \mathsf{initV}(v(1), \mathtt{true}, S, d')}{\mathsf{change}((q, v, \mathtt{true}, S, d), d') = (q, v', \mathtt{true}, S, d')}$$

// Switching between chromatic and diatonic mode:

$$\frac{d \neq \mathtt{sequent}}{\mathsf{change}((q, v, \mathtt{true}, S, d), \mathtt{false}) = (q, v, \mathtt{false}, S, d)}$$

$$\frac{d = \mathtt{sequent} \quad v' = \mathsf{initV}(v(1), \mathtt{false}, S, d)}{\mathsf{change}((q, v, \mathtt{true}, S, d), \mathtt{false}) = (q, v', \mathtt{false}, S, d)}$$

$$\frac{v' = v \fatsemi (\lambda x \bullet pitches(S, x)(1))}{\mathsf{change}((q, v, \mathtt{false}, S, d), \mathtt{true}) = (q, v', \mathtt{true}, S, d)}$$

Table 7. Pitch layout parameters and evaluation rules, part two.

for the minimal new pitch which is higher than the current pitch at row one, column one, and creates from there an initial layout.

So the "nearest note to C" is the note which leaves as many pads as possible unchanged. If there are no common pitches between both pitch layouts, then it is the next higher pitch above C in the new layout. This second part of the rule is totally arbitrary and will not give stable results when applied repeatedly!

When distance changes, we construct a new pitch layout but keep $v(1)$ and thus the complete first row. All other rows must change necessarily.

8.3.3 Changing InKey

The set of pitches always changes when altering InKey, which is switching between chromatic and diatonic mode. According to the "principle of least change", the proposal in Table 7 keeps the starting points of all rows as far as possible: From chromatic to diatonic it is a *compaction*, where the first pad in every row becomes the first diatonic pad in the preceding layout.

From diatonic to chromatic the row starts are kept completely.

This is not a solution for distance = `sequent`: The preceding layout

…							
D4	E4	F4	G4	A4	B4	C5	D5
C3	D3	E3	F3	G3	A3	B3	C4

would be changed to

…							
D4	D♯4	E4	F4	F♯4	G4	G♯4	A4
C3	C♯3	D3	D♯3	E3	F3	F♯3	G3

and thus effectively *lose access* to diatonic pitches. In this case the fresh construction of an initial layout starting with the pitch in pad $(1, 1)$ seems more appropriate.

8.3.4 Conclusion

The sheer number of formula lines needed to describe a possible solution shows that a precise specification cannot be expected from just one manual page in prose language.

9. IMPLEMENTATION

We provide an implementation of our slightly extended version of TAB+, Chart+, the `tScore` representation, the data model,, and the pitch layout algorithm in one Java application. It can be downloaded from `http://bandm.eu/downloads/SquarePads.jar`. It can be started from the command line (CLI) as usual, taking options as explained by the option "`--help`", or by simply clicking the jar file icon, which requires to enter option values through its GUI.

Without any `tScore` file input, the application can be used to experiment with the different flavours of the pitch layout algorithm: The above-mentioned parameters InKey, scale, distance, base pitch, and Fixed can be changed dynamically, using the GUI widgets. The effects are shown on the GUI grid, see Figure 8. On this grid, simple and monophonic sounds can be played by clicking, produced by self-contained sound synthesis.

Additionally a hardware can be connected, on which the lighting is also shown on the pads, and the sounds can be played by pressing them. Currently we support only the "Novation Launchpad MK3" TUI device—probably adapting the mapping of the MIDI commands will be necessary for other hardware.

Selecting a file which contains a `tScore` source allows to render it into a sequence of SVG files containing the

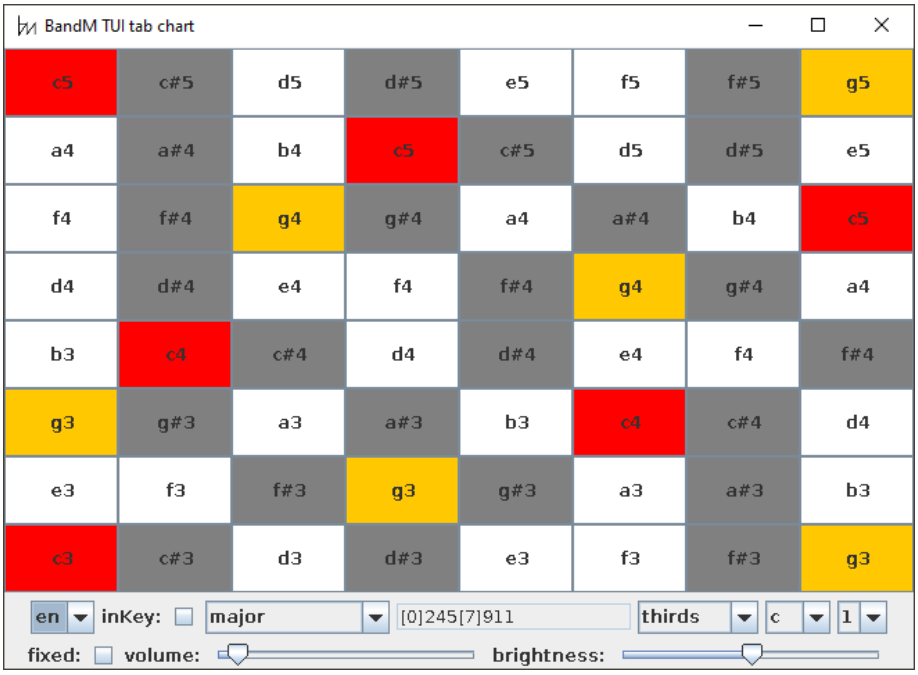


Figure 8. Screenshot of the tool in interactive mode

topic	lines of code
`tScore` model	335
chart rendering	277 + 148
tablature rendering and GUI	323 + 217
dynamic change of pitch layout	860

Table 8. Lines of code of the current implementation. (Measured with CLoC[16])

sequence of charts, into one file with an animated chart, and into tablature output (currently only one measure). By using CLI this can be done in batch mode. Activating interactive mode allows sequencing the score auditive, visually on the GUI and the connected hardware, and to a MIDI-out channel.

The Java sources are licensed as CC-BY-SA-NC and published on a source hosting platform. Table 8 shows the surprisingly low programming effort. The tool uses BandM metatools, metricSplit [12, 13], and Sig [14]. It uses the JSVG library by Jonathan Sevy [15].

10. SUPPORT FOR INTERDISCIPLINARY RESEARCH AND PRACTICE

In principle, all three notation formats defined by Wilde and White can be applied to any input device which has an orthogonal grid structure of binary-state input buttons, with arbitrary (manageable) numbers of rows and columns.

Since MIDI events can be translated into any kind of data, such TUIs and touchscreen GUIs can be used as input device for any interactive software, e.g. in psychological experiments, on-line examinations, or as a game controller. (We intend to explore them as a playing device for a synthesizer with pure intonation, mapping the *Euler net* to the two axes. To support TUIs of other vendors, the existing software must be enhanced by an adaption layer for different usage of MIDI commands.)

All formats presented here, Charts+, TAB+, their data model, and the `tScore` format, deal only with the physical coordinates of the buttons. Therefore they can be employed in all these cases for the pressed keys and the pressing fingers. They can be used for protocols of user interaction, or as prescribing score notation for human or automated execution. Translating the notation into the data model allows statistical analyses, and to apply properties (like those defined above in Table 3) for consistency checks.

There is one limitation: TAB+, as presented by Wilde and White, is restricted to cases which employ conventional meter-based rhythmical timing, and to users which are familiar with the duration encoding by CWN. Contrarily, the `tScore` representation can easily be adopted to other time models, e.g. physical time, or mere sequential order without any duration, and so can our rendering algorithm for tablatures. This opens a wider spectrum of applicability.

The other way of translation still needs exploring: creating timed sequences of pad patterns from traditional note information. This is a challenge because the physiological constraints of the human hands must be respected, see for instance the data type PressPattern in Table 3.

11. ACKNOWLEDGEMENTS

Many thanks to Lawrence Wilde and Charles White for additional information about their work.

Many thanks to Jon Sevy for realizing our wishes in his JSVG library.

Many thanks to the anonymous reviewers; not all of their valuable hints could be realized, due to lack of time.
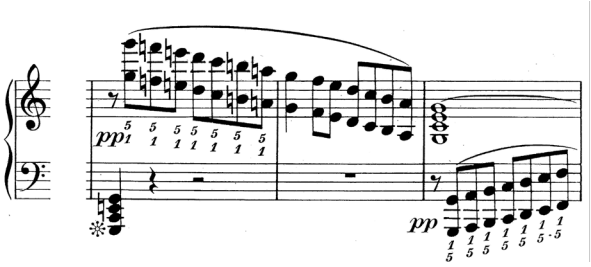


Figure 9. Finger numbers in the Waldstein sonata

A. IRREGULARITIES HERE AND IN CONVENTIONAL KEYBOARD CWN

Often and in very different realms, the authors have experienced that the attempt to construct a mathematical (meta-)model brings new insights into the pre-digital historic practice, especially bringing to light non-orthogonalities and idiosyncrasies. Here, modelling the TUI can be seen as preparatory for modelling the playing on a conventional chromatic keyboard.

Some irregular and noteworthy aspects:

(A) At the beginning (of the conventional CWN text) of a piece (for piano, organ, etc.) there is nearly always an implicit assignment of the notes to the two hands, by the implicit meaning of upper and lower staff.

Later, an explicit prescription of left or right hand can appear by the textual *m.s.* annotation. This is a *sticky* attribute:[5] It is written over one note and *stays valid* for some subsequent notes.

But there is no notational counterpart meaning "the prescription ends here" or "from here on the hands are free again", like a *loco* instruction after an *8va sopra*.[6]

(B) The pragmatic aspect, as linguistics call it, of these annotations can be different, namely a prescription, meant as an indispensable part of the work, or just a recommendation. When authored by the composer and not the editor, both are conceivable.

(C) Statements (A) and (B) also hold for finger numbers. Especially it is unclear whether such a number holds only for the attack ("push down with this finger, afterwards you may silently change") or for the whole notated duration ("push down with this finger and stick to it"). In other words: In conventional CWN there are signs to prescribe a silent finger change, but not to forbid it.

Similar to (A), there is no way to notate "the prescribed fingering ends here", for instance at one notehead in a long tied sequence, For this purpose one could learn from the `tScore` implementation above the sequence "`12R3␣␣>R`" and translate it into the conventional "3␣␣*m.d.*".

When humans play piano, these questions are practically not an issue. But they *must* be answered when implementing an algorithmic simulation, an automated interpretation by a robot, etc.

(D) In conventional CWN for keyboard, a *finger indication* is only sensible when the *hand* is also prescribed. The only exception we know of are the (in)famous octave glissandi in the last movement of the Waldstein sonata, as proposed by Bertha Wallner, [17] see Figure 9: $\frac{1}{5}$ *could* be read as "perform (as a glissando) with small finger and thumb of the same hand, right or left as you like".[7]

(E) In conventional CWN for keyboard, a *finger indication* is only sensible when a key is pressed. But a *hand indication* can also be sensible with a *pause*! The information "this finger does currently not play" is normally not made explicit in conventional notation, but "this hand pauses" is! Especially for rendering in traditional piano staff notation, any syntactic entity which is a pause needs the "which hand" information, because this normally decides in which of the two staves to put the representing glyph.

[5] Called NOTA.ADDENDAMANENTIA in LMN [8].

[6] Called *Aufhebungsereignis* = PERMANENS.EXIT in LMN [8].

[7] Again the status of this statement is critical anyhow: It is an information how these notes *had been* executed on the instruments of that time, and not a hint how to perform today on modern instruments.

B. REFERENCES

[1] L. Wilde and C. White, "TABstaff+: A hybrid music notation system for grid-based tangible user interfaces (tuis) and graphical user interfaces (guis)," in *Tenor 2024 proceedings*, Zürich, 2024, pp. 159–168, https://www.tenor-conference.org/proceedings/TENOR2024-Proceedings.pdf.

[2] C. Sperren, *A notation system for Ableton Push*, 2020, https://www.chrisperren.com/blog/2020/12/02/a-notation-system-for-ableton-push[20240724].

[3] S. Riegel, A. Kuźbik, M. Hughes, C. Tipton, J. Evans, and J. Terry, *Ableton Push 3 Manual*, Berlin, 2020, https://www.ableton.com/en/push/manual/[20250617].

[4] M. Lepper and B. Trancón y Widemann, "tscore: Makes computers and humans talk about time," in *Proc. KEOD 2013, 5th Intl. Conf. on Knowledge Engineering and Ontology Development*, J. Filipe and J. Dietz, Eds., instincc. Portugal: scitePress, 2013, pp. 176–183, http://markuslepper.eu/papers/tscore2013.pdf[20231003].

[5] ——, "Translets —— parsing diagnosis in small dsls, with permutation combinator and epsilon productions," in *Tagungsband des 35ten Jahrestreffens der GI-Fachgruppe "Programmiersprachen und Rechenkonzepte*, ser. IFI Reports, vol. 482. University of Oslo, 2018, pp. 114–129, http://urn.nb.no/URN:NBN:no-65294[20230920].

[6] ——, "Morton Feldman's "projections one to five" — exploring a classical avant-garde notation by mathematical remodelling," in *Proceedings of TENOR 2024 conference*, 2024, https://www.tenor-conference.org/proceedings/2024/15_TENOR2024_Lepper.pdf[20250701].

[7] ——, *Example Instances of the TScore Project Infrastructure*, 2013, http://markuslepper.eu/sempart/tscoreInstances.html[20230920].

[8] M. Lepper, "De linguis musicam notare," Ph.D. dissertation, Universität Osnabrück, Osnabrück, 2021, https://www.epos.uni-osnabrueck.de/buch.html?id=150[20230920].

[9] J. Wolf, *Handbuch der Notationskunde, Teil zwei*. Leipzig: Breitkopf & Härtel, 1919, https://ia800205.us.archive.org/3/items/handbuchdernotat02wolf/handbuchdernotat02wolf.pdf[20230101].

[10] C. Pot, *Klavarskribo und die Tontheorie*, 1932.

[11] J. Oberschmidt, "Zwischen Mensch und Maschine. Komponieren für Piano-Player und Player Piano," in *Musik im Spektrum technologischer Entwicklungen und Neuer Medien*. Osnabrück: epOs Verlag, 2015, p. 563–580.

[12] M. Lepper and B. Trancón y Widemann, "Metricsplit - Automated Notation of Rhythms, Adequate to a Metric Structure," Technische Universität Ilmenau, Ilmenau, Tech. Rep., Mar 2017.

[13] ——, "metricsplit: An automated notation of rhythm aligned with metric structure," *Journal of Mathematics and Music*, vol. 13, no. 1, pp. 60–84, 2019.

[14] B. Trancón y Widemann and M. Lepper, "Sig adlib — mostly compositional clocked synchronous data-flow programming in java," in *Tagungsband des 35ten Jahrestreffens der GI-Fachgruppe Programmiersprachen und Rechenkonzepte*, ser. IFI Reports, vol. 482. University of Oslo, 2018, pp. 31–48. [Online]. Available: http://urn.nb.no/URN:NBN:no-65294

[15] J. Sevy, *JSVG:Java SVG library*, 2023, https://jsevy.com/wordpress/index.php/java-and-android/java-libraries/jsvg-java-svg-library/[20250708].

[16] A. Danial, "cloc — count lines of code (v1.92)," Tech. Rep., 2021, https://github.com/AlDanial/cloc [20231003]. [Online]. Available: https://doi.org/10.5281/zenodo.5760077

[17] L. van Beethoven, *Klaviersonaten II*. München: Henle Verlag, 1952, *edt.* Bertha Antonia Wallner.

[18] J. M. Spivey, *The Z Notation: A reference manual*, ser. International Series in Computer Science. Prentice Hall, 1988, https://spivey.oriel.ox.ac.uk/corner/Z_Reference_Manual[20240729].

C. MATHEMATICAL NOTATION

The employed mathematical notation is fairly standard, inspired by the Z notation [18]. The following table lists some details:

$\mathbb{N}$	All natural numbers, incl. zero.
pred	The predecessor operation on integer numbers.
$\#A$	The cardinality of a finite set = the natural number of the elements contained.
$\mathbb{P}A$	Power set, the type of all subsets of the set A, incl. infinites.
OPT A	The optional type of all values from A plus the undefined value $\perp$. (Our extension; could be realized as a generic schema.)
$\varnothing$	The empty set (a constant of any type)
$A \setminus B$	The set containing all elements of A which are not in B.
$A \times B$	The product type of two sets A and B, i.e. all pairs $\{c = (a, b) \mid a \in A \wedge b \in B\}$.
π_n	The nth component of a tuple.
$A \to B$	The type of the *total* functions from A to B.
$A \nrightarrow B$	The type of the *partial* functions from A to B.
$f \, (\!\mid s \mid\!) $	The image of the set s under the function or relation f.
$a \mapsto b$	An element of a relation; simply another way to write (a, b).
dom A	Domain of a relation or function.
ran A	Range of a relation or function.
$r^\sim$	The inverse of a relation or function.
$\mathbf{ID}_A$	$= \{a \in A \bullet a \mapsto a\}$, the identity relation.
$r \fatsemi s$	The composition of two relations: the smallest relation s.t. $a \; r \; b \wedge b \; s \; c \Rightarrow a \; (r \fatsemi s) \; c$. (first apply r, then apply s.)
$r \oplus s$	Overriding of function or relation r by s. Pairs from r are shadowed by pairs from s: $r \oplus s = \big(r \setminus (\mathsf{dom}\, s \times \mathsf{ran}\, r)\big) \cup s$, with dom and ran being domain and range, resp.
$S \lhd R$	$= R \cap (S \times \mathsf{ran}\, R)$, i.e. domain restriction of a relation.
$R \rhd S$	$= R \cap (\mathsf{dom}\, R \times S)$, i.e. range restriction of a relation.
$R \rhd\!\!\!\!\!\!\cdot \; S$	$= R \setminus (\mathsf{dom}\, R \times S)$, i.e. negative range restriction of a relation.
seq A	The type of finite sequences from elements of A, i.e. of maps $\mathbb{N} \nrightarrow A$ with a contiguous range $\{1..n\}$ as its domain.
$squash(a)$	Turns any partial function $\mathbb{N} \nrightarrow A$ into a seq A by compactifying the indices.
$[\; expr \;]$	Iverson bracket: delivers 1 if the boolean expression evaluates to true, and 0 if to false. (our extension)
max A	The maximum from a set of values.
min A	The minimum from a set of values.

The frequently used notation

$$\frac{\begin{array}{ccc} & a & \\ b & & c \end{array}}{d}$$

means as usual $a \wedge b \wedge c \implies d$. Nearly always it should be read as an *algorithm*: "For to calculate d, try to calculate a, b and c. If this succeeds, the answer d is valid."

Functions are considered as special relations; relations as sets of pairs. So with functions, expressions like "$f \cup g$" are defined.

VISUALIZING VIBRATO: A LIBRARY OF WIGGLES TO BROADEN TECHNICAL AND CULTURAL HORIZONS

Barnaby Brown
Royal Conservatoire of Scotland
barnaby@pibroch.net

Austin Oting Har
Michigan State University
haraustI@msu.edu

ABSTRACT

In this paper, we provide a library of 169 vibrato graphics utilizing recent developments in music publishing that make it easier to notate vibrato and with superior clarity. Visualizing vibrato with two variables, speed and presence, helps teachers, students, composers and musicologists to broaden horizons technically and stylistically. In an ocularcentric world, seeing vibrato makes it easier to comprehend cultural differences and practice vibrato. Users may copy, paste, and customize the 169 vibrato graphics. This requires installation of the Open Font 'Bravura Text', developed in tandem with the Standard Music Font Layout specification (SMuFL), which together represent a milestone in the history of music notation. Bravura gives access to over 2440 music symbols, 35 of which are tessellating glyphs for building vibrato wiggles. As a result of testing combinatorial options and developing an inventory of graphics relevant to vibrato training, we aim to advance notational intelligence across technical and expressive dimensions of vibrato for instrumentalists and vocalists alike. With SMuFL's latest version 1.4 released in March 2021, and version 1.5 currently in the drafting stage, we additionally offer suggestions to the developers of this significant notational tool for how it may be further refined.

1. INTRODUCTION

It is unusual to notate vibrato. Indications tend to be vestigial and verbal. This imprecision allows technical habits and cultural biases to shut down the imagination, closing doors to expressive possibilities. In many vocal and instrumental traditions, from Western classical (e.g. soprano, violin) to diverse Asian traditions (e.g. duduk, piri, shakuhachi) cultivating a good vibrato is key to professional success, yet it remains invisible in the scores that mobilize the industry. Conversely, for composers working with music notation software, the lack of variety and precision in vibrato symbols can negatively impact their creativity and usability [1, p. 7]. From both sides of the coin, for example composers from a Western classical background or performers from an Asian tradition, this can foster prejudice, blind spots, musical parochialism, and lack of intercultural understanding about vibrato. Instead of being able to speak multiple vibrato languages, many musicians overlook less privileged vibrato cultures. By visualizing vibrato, the idea of engaging with the unfamiliar, or mastering the mysterious, becomes less daunting. Involuntary habits can be noticed, the craft of vibrato made more deliberate, and virtuosity, concentration, focus, and expression can be enhanced in both performance and notation [1, p. 13]. Whether the objective is to learn a foreign musical style, or to raise one's level of virtuosity within a musical style, it is helpful to see vibrato in more detail than 'vib.' and 'senza vib.'. We therefore welcome the inclusion by Daniel Spreadbury within Bravura of a repertoire of glyphs that allow vibrato to be represented on the page with greater precision. [1]

The Bravura font family is significant for three reasons. First, it has been released under the SIL Open Font License, which means that it is free to download, use, embed, redistribute with other software (including commercial software) or to create derivative versions. Secondly, its companion, the Standard Music Font Layout (SMuFL, pronounced 'smoofle'), is a revolutionary specification, solving what was becoming a technological nightmare in a manner applauded by music engravers and software developers. [2] Thirdly, Bravura is a music font of outstandingly high quality. It harvests the collective heritage of the last two hundred years comprehensively, pushing levels of beauty and legibility even higher. Bravura and SMuFL were launched in 2013, establishing Spreadbury and Steinberg Media as global leaders developing the next-generation music notation and composition software, Dorico. Our motivation for creating and sharing a library of customizable vibrato graphics is to liberate the creative imagination of performers and composers and to decolonize vibrato education. SMuFL Version 1.4 allows for seven shades of 'speed' and five shades of 'presence'. [3] We use the term presence, rather than width or intensity, to reduce confusion. The width of a wiggle segment conveys speed, not presence, and the adjective intensity is generally used for

Copyright: ©2025 Barnaby Brown et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

[1] Bravura is described at smufl.org/fonts and maintained at github.com/steinbergmedia/bravura.

[2] The goal of SMuFL is to establish a new standard glyph mapping for musical symbols that is optimized for modern font formats and that can be adopted by a variety of software vendors and font designers, for the benefit of all users of music notation software. Launched in 2013, Version 1.4 was released on 16 March 2021 and is the latest version: w3c.github.io/smufl/releases/1.4. As of this time of writing on 17 July 2025, Version 1.5 is in the drafting stage: https://w3c.github.io/smufl/latest/

[3] The present repertoire of vibrato glyphs was added by Daniel Spreadbury to SMuFL in Version 0.5 (2013-07-08) and to Bravura in Version 0.2 (2013-07-05); smufl.org/versionhistory and raw.githubusercontent.com/steinbergmedia/bravura/master/redist/FONTLOG.txt.

timbre, a variable that is independent of vibrato. [4] In notation, vibrato speed and presence are represented by the wiggle's wavelength and amplitude. Acoustically, presence collapses three parameters into one: variance in pitch (known as 'extent'), loudness (or amplitude), and timbre. In order to give vibrato wiggles the same level of beauty and elegance that SMuFL accommodates for trills, we recommend for Version 1.5 of SMuFL that the number of gradations of presence be increased from five to eight (introducing straight segments). Our library presents 169 multi-segment wiggles in five tables (Figures 1-5), with an analysis of the composition of each graphic to facilitate customization. Working with these wiggles requires installation of a SMuFL-compliant font. This document uses 'Bravura Text' version 1.393, released in February 2021. Bravura is a family of two fonts, one intended for use in scoring applications, the other for use in text-based applications. We only use the font for scoring applications, 'Bravura', to illustrate correct tessellation under Recommendation 1 below, otherwise all the graphics use 'Bravura Text'. [5]

2. A LIBRARY OF 169 VIBRATO WIGGLES

This library is intended to serve anyone working with a vibrato-capable instrument who wishes to expand their technical horizons or escape their cultural paradigm. These visualizations are intended as a means to help cultivate vibrato skills that are less mechanical, less limited by cultural conditioning, more supple, more in control, wider in expressive palette, and better able to adapt to context.

Figure 1 presents the full repertoire of component glyphs with their names, Hex codes, and a run of identical glyphs to copy and paste. The glyphs are named 'wiggleVibrato-[Presence][Speed]', with five gradations of presence from 'Smallest' to 'Largest', and seven gradations of speed from 'Slowest' to 'Fastest'. Figures 2–5 combine the segments in systematic ways, capturing a few of the vibrato idioms and training exercises practised around the world. The range, scope, and subtleties in real life are vastly greater, limited only by human imagination and schooling. The graphics may be regarded as signs intended to mobilize complex actions rather than as phonographic documents with a mechanical correspondence to reality[2]. Figures 3 and 4 present tessellations that reveal how the glyphs were designed. Figures 2 and 5 push beyond their designer's intentions in order to capture real-world vibrato practices more precisely, leading us to make Recommendations 2 and 3. The numbers in brackets reveal the composition of each tessellation. For example, (2,2,2,2) means that there are two instances of each segment.

The vibrancy, or emotional life in a musical tone involves oscillations that are an interaction of three acoustic components. The main component may be pitch, loudness, or timbre: they vary in proportion between instruments, between performers, between notes, and during notes. A vibrato's presence may therefore be increased by a wider pitch range (the 'extent'), a wider dynamic range, or a wider timbral range. Jeffrey Lependorf, the composer and shakuhachi master, stresses that on the shakuhachi "various [vibrato] widths as well as various speeds are possible and all can be a sustained note" and gives the example of *yuri* which "tend either to begin fast and erratic, slowing down to a controlled vibrato, or vice versa"[3, pg. 238-239]. [6] The Armenian duduk teacher, Vick Gevorgyan, likens the rhythm of a good vibrato to the sound of a ball bouncing, with the bounce height and period of oscillation decaying naturally [4]; this decaying oscillation is also heard on the Korean piri [5]. On instruments with fewer notes, vibrato is more in the limelight and often a focus of artistry, with performers exercising greater control than their Western counterparts.

The clean slate offered by the aulos revival [6, pg. 24-40], the instrument of Greek tragedy, and the global equality and diversity agenda prompt us to consider how to decolonize the attitudes to vibrato perpetuated by Western notation and Western conservatoire training. [7] How could we give oppressed vibrato practices equal consideration and respect on the pages of globally-connected devices, updating Western notation to reflect contemporary values? And how might these vibrato traditions be contextualized within intercultural frameworks; for example, migrating elements of one vibrato tradition into another context (e.g. a Greek tragedy-inspired Western opera) or vice versa, expanding a vibrato tradition with new possibilities [7, pg. 9-11]? Writing new music for doublepipes – exercises and rehearsals for recording and performance of Austin Oting Har's Greek tragedy-inspired opera *The Ghost* [6, pg. 24-40][8]– we decided to differentiate five broad categories:

1) d-vib. diaphragm vibrato (abdomen, rib-cage, throat)
2) m-vib. mouth vibrato (lip, chin, jaw, neck)
3) a-vib. arm vibrato (elbow, wrist, hand)
4) f-vib. finger vibrato (slight leak, slightly covering the highest open hole, fully covering lower holes) [8]
5) i-vib. interference-beat vibrato (unison pitches slightly detuned)

All of these techniques would have been possible in the elite piping cultures of ancient Mesopotamia, Egypt, Persia, Greece, and Rome; all have subtypes that are porous (except i-vib.) and promiscuous [9]. [9] Vibrato is a surprisingly rich ecosystem characterised by hybridity and strong opinions. The conventional stance of Western singing teachers, who widely view non-Western and non-Classical vi-

[4] The concepts of a wide vibrato (generally slow) and an intense vibrato (generally fast) fail to decouple the two variables with neutrality. The term 'width' may relate to the physical action of producing vibrato on Western string instruments, or the extent of the pitch bend. Width also fails to describe the relevant action, for example, of woodwind players who use finger vibrato.

[5] The latest versions of the Open Type Font (OTF) files are available at github.com/steinbergmedia/bravura/tree/master/redist/otf. In most cases, a double click on the downloaded OTF file will install it on your system.

[6] See Lependorf's paper also for a graphic notation of a slow and wide vibrato becoming fast and narrow on shakuhachi.

[7] The crux of a debate over how to notate music for ancient Greek aulos is explained in Barnaby Brown, Babylonian tunings for the 9-string sammû (c. 1800–600 BCE) and a hypothetical tablature for Panhellenic aulos (c. 500 BCE), Figshare, 2020, doi.org/10.6084/m9.figshare.13064126.

[8] Historical names in Western music for types of finger vibrato include flattement and bisbigliando.

[9] For demonstrations on a Pydna aulos with double reeds, and a Rustic aulos with single reeds, see Barnaby Brown, Five Types of Vibrato: exploring the possibilities on doublepipes, YouTube, 2020, youtu.be/FFQrtazhX1A.

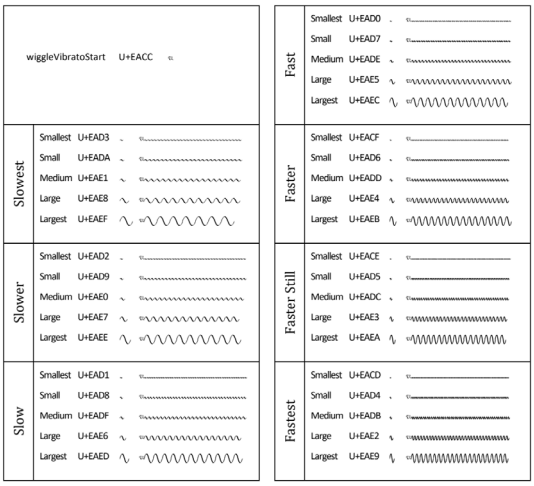


Figure 1. The thirty-six 'wiggleVibrato' glyphs in SMuFL 1.4 and Bravura 1.393.

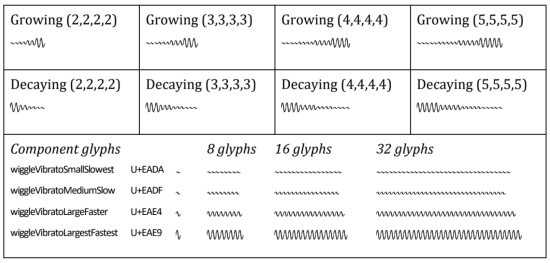


Figure 2. Variable-presence, fixed-speed tessellations

brato cultures as 'unhealthy', 'artificial', or 'incorrect', may be linked to vibrato's invisibility.[10] Composers or educators working with other instruments could adapt this culture-neutral taxonomy for situations where it was desirable to notate multiple vibrato types in a single score, without suggesting that one type or aesthetic is superior to others.

3. METAPHORS FOR VIBRATO TRAINING

Five levels of speed and five levels of presence are probably optimal for educational purposes. Meaningful extra-musical associations make it easier to differentiate the levels conceptually and sets of words like the following may be useful in teaching and learning.

Anything repetitive is a potential health hazard and discovering the easiest way, loosening up, is important in every vibrato idiom. These metaphorical associations on a simple 1–5 scale may be more memorable and manageable in the context of technical training, but for typographic elegance, the finer gradations provided by Bravura are invaluable. There is no need for educational categories and graphic increments to coincide. They satisfy different criteria: the metaphors meet needs of pedagogy and practising, developing competency, whereas the graphic distinctions satisfy demands of music publishing, producing scores.

[10] Karyn O'Connor, singwise.com/articles/vibrato-what-it-is-and-how-to-develop-it. The belief that vibrato 'occurs naturally in order to protect the vocal folds' is cultural, correct from one perspective; it fails to explain Indian, Arabic, Balkan, and other vocal styles that possess far greater agility in vibrato than is customary in Western Classical singing; for example, Pratibha Sarathy, 'How to sing Gamakas?', VoxGuru, YouTube, 2018, youtu.be/AIPraIlSmIk.

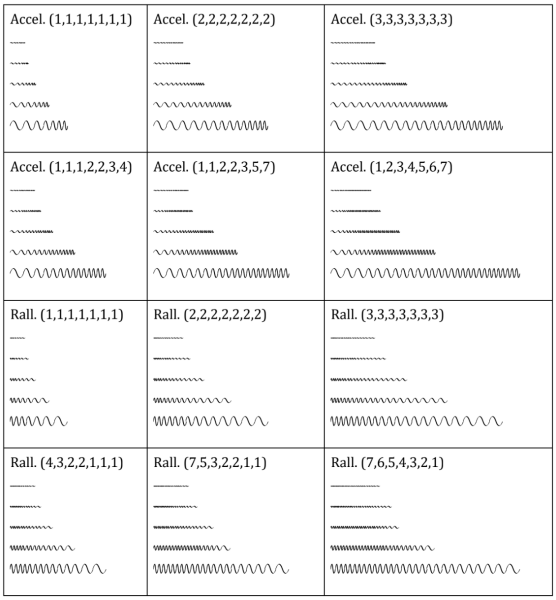


Figure 3. Fixed-presence, variable-speed tessellations

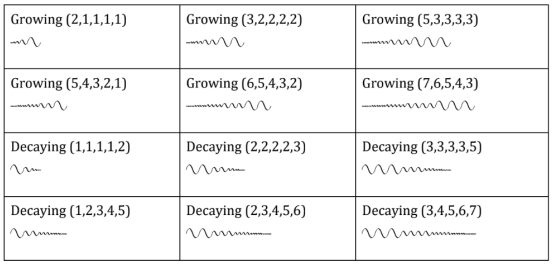


Figure 4. Variable-presence, linear-speed-change tessellations

4. SHARPENING THIS NOTATIONAL TOOL

We began developing this library for personal use, producing scores for aulos, practicing unfamiliar vibrato techniques destined for the recording and performance of Har's Greek tragedy-inspired opera[10][8][6, pg. 24-40], and developing materials for aulos students that would be good allies for marginalized vibrato cultures. This creative process brought some minor issues into focus, leading us to suggest three refinements. Although primarily addressed to font designers contributing to the evolution of the Bravura font family, and to the W3C Music Notation Community Group responsible for the evolution of SMuFL, they are shared here to invite a broader spectrum of potential users to add their wisdom on how to make this tool even more satisfying and future-proof.

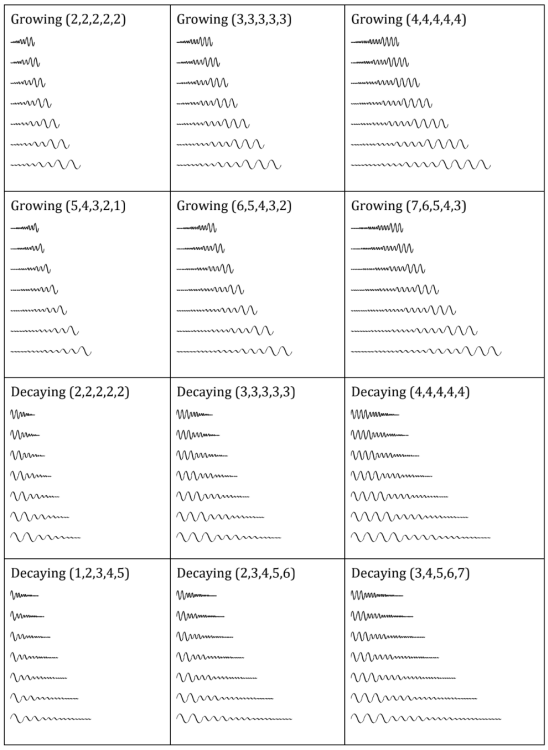


Figure 5. Variable-presence, exponential-speed-change tessellations

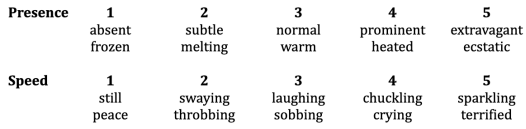


Figure 6. Five levels of speed and five levels of presence

4.1 Recommendation 1: Negative side bearings in Bravura Text

The wiggles tessellate beautifully in applications designed to support SMuFL. At the present time, these include Dorico, MuseScore, Capella 8, Soundslice, Verovio, neoScores, MusicXML, Logic Pro X, and MaxScore, with support for Finale coming soon. There is an issue, however, with Microsoft Office 365. When pasted into Word or Powerpoint, the 'wiggleTrill' segments in version 1.393 of Bravura Text appear with normal character spacing at font size 72 as seen in Figure 7.

Note that there are nine speeds of trill. Switching kerning on has no effect and sharp corners appear with Word's export to PDF function. There is a user workaround. With character spacing condensed by 4.5 points, the segments overlap as seen in Figure 8.

Slightly smoother junctions may be achieved by adjusting the spacing manually (between 3.2 and 5.1 points) as seen in Figure 9.

More efficient is to build the wiggles in an application like Dorico or MuseScore and do a screen capture. Ideally, however, the glyphs would tessellate correctly automatically in the applications available as standard in ed-

Figure 7. Recommendation 1.1: 'wiggleTrill' segments with character spacing at font size 72

Figure 8. Recommendation 1.2: 'wiggleTrill' segments overlapping with character spacing condensed by 4.5 points

ucational institutions, without administrators having to do anything other than install Bravura Text. The SMuFL guidelines for fonts designed for scoring applications state: "Tessellating glyphs (such as wavy lines, or the component parts of complex trills and mordents) should have negative side bearings, in order to achieve correct tessellation when set in a single run [11]."

In the version of the font designed for scoring applications, 'Bravura', the glyphs tessellate beautifully, but the large white space above and below is unwieldy in text-based applications. The glyph is shown in Figure 10 (the excessive white space appears when used in text-based applications).

For tessellating glyphs, we recommend that the existing guidelines for fonts designed for scoring applications be extended to fonts designed for text-based applications. We also offer font designers and application developers a set of wiggles for testing purposes (Figure 11). Ideally, non-technical users would be able to build wavy lines with variable-speed and variable-presence, with the tessellation happening automatically as the typographer intended.

4.2 Recommendation 2: Invariable glyph width per speed

Training the muscles that produce vibrato involves learning how to relax, reduce tension, and cultivate an effortless oscillation that beautifies the sound with freedom and agility, without risk of fatigue or injury. One of the exercises that promotes flexibility and enriches the expressive palette is to maintain a constant vibrato speed while turning up or down the presence. Another is to maintain a constant vibrato presence while turning up or down the speed. Decoupling the two parameters is the antidote to an involuntary or mindless vibrato: the result of tension and bias curbing an artist's capacity to be deliberate.

Figure 2 reveals that the vibrato glyphs were not designed with independence of speed and presence in mind. Only four gradations of presence are available for notating a fixed-speed vibrato because the seven speeds are not absolute, but relative (proportionally scaled with the presence). As a result, the 'SlowestSmallest' wiggle has a shorter wavelength than the 'FastestLargest' wiggle:

This is counterintuitive. The difference in glyph width between 'SlowestLargest' and 'FastestSmallest' is also un-

Figure 9. Recommendation 1.3: 'wiggleTrill' segments with smoother junctions by manually adjusting character spacing between 3.2 and 5.1 points

Figure 10. Recommendation 1.4: 'wiggleTrill' segments with good tessellation using Bravura font, but excessive white space above and below the symbol when used in text-based applications

realistically large:

Pavarotti's vibrato speed ranges between 5.6 and 6.6 Hertz, with significant emotional nuance lying within this 1.0 Hertz bandwidth [12]. Instrumentalists often have a wider range of speed than singers, and more control, but it is in Asian cultures that the widest range of speed is found. Where many Western instrumentalists demonstrate virtuosity through fingering techniques, "the shakuhachi demonstrates true virtuosity in the areas of vibrato and pitch inflection" [3, pg. 238]. The result of making vibrato speed relative to presence, rather than absolute, is that vibrato nuances which are compelling and eminently practical can only be notated crudely (Figure 2), yet speed changes that are impractical, far beyond what Asian virtuosi do, can be notated beautifully.

The present wavelength ratio of 11:1 between the 'Fastest' and 'Slowest' glyphs exaggerates reality unhelpfully. We recommend simplifying reality by fixing the seven vibrato speeds to the widths of the existing 'Large' glyphs. A ratio of 3:1 between the longest and shortest wavelengths is sufficient to mobilize musically-intelligent results, bringing this vibrato notation tool into alignment with the practices of musicians.

Vibrato training on many instruments involves starting slow (1 Hertz) and spreading looseness and freedom of movement to progressively faster rates, without locking up or freezing [13]. [11] Fixing the gradations of speed to notional practice tempi would be a performer-friendly approach, making this vibrato notation tool more useful in education.

Figure 6 shows how the glyphs for each degree of presence have frequencies that vary comparably with the wavelengths of the seven colors of the rainbow.

The existing names of the seven speeds are confusing because 'SlowerStill' is missing and there is no increment between 'Slow' and 'Fast'. When building variable-speed vibrato graphics, it is opaque which glyphs are adjacent in the series. SMuFL's 'About' page states: "Room for future expansion has generally been left in each group, so code points are not contiguous. The code point of each glyph

[11] For example, see String Technique, Developing Vibrato for Cello, YouTube, 2018, youtu.be/Ki0miM7wQbw.

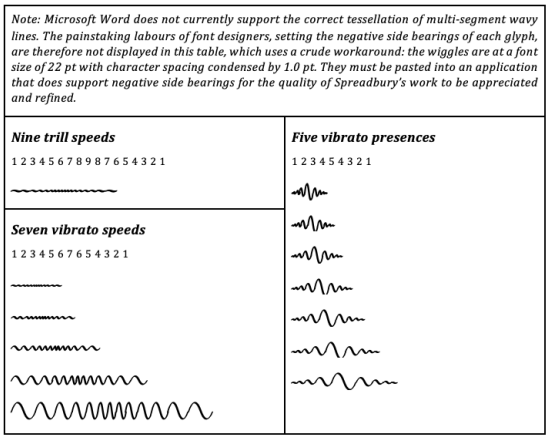


Figure 11. Paradigm of adjacent shades for testing negative side bearings

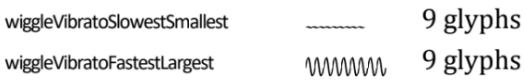


Figure 12. Recommendation 2.1: 'SlowestSmallest' wiggle has a shorter wavelength than the 'FastestLargest' wiggle

in SMuFL 1.0 and later is intended to be immutable, and likewise every glyph has a canonical name, also intended to be immutable" [14]. [12]

We recommend that an exception be made, renaming the seven wavelengths for each amplitude 'Speed1' to 'Speed7'.

In summary, Figure 3 and 4 represent how the glyphs were designed. In Figure 3, it is clear that only the speed is changing. In Figure 4, both variables are changing: the speed is relative to the presence. We recommend that the glyphs in each speed category have a fixed width, so that Figure 2 becomes as rich as Figure 4, and Figure 4 and 5 can be merged, jettisoning the distinction between linear and exponential speed change. The tessellations in each cell of Figure 4 would become the same length and this would replace Figure 2. This harmonises the notation with musical practice, achieving a typographic granularity consistent with acoustic reality.

4.3 Recommendation 3: Finer gradations of presence

Our third recommendation is to add another three shades of presence, including absent. This would smooth the step changes in amplitude visible on the title page of the present document and make the notation of Asian vibrato idioms more intuitive for readers by incorporating sections of straight, vibrato-less tone within the graphic.

We suggest names for the additional symbols in Figure 6, renaming the existing glyphs so that, as for speed, it is transparent which glyphs are neighbouring increments.

[12] There are 118 groups of glyphs, proceeding roughly in order of usage, from general to particular instruments, types of music, or historical periods; smufl.org/about. The vibrato wiggles are in the group 'Multi-segment lines', which is found just below the thirteen 'Medieval and Renaissance' groups.



Figure 13. Recommendation 2.2: 'SlowestLargest' and 'FastestSmallest' glyph width is unrealistically large

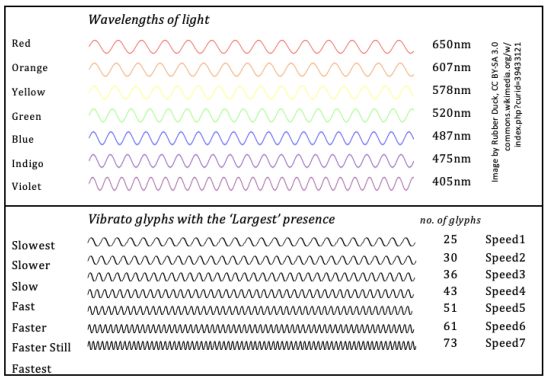


Figure 14. Nomenclature for seven gradations of wavelength

Having seven shades of presence for each shade of speed, plus three straight glyphs ('Presence0Speed1', 'Presence0Speed4', and 'Presence0Speed7'), would increase the number of tessellating vibrato glyphs from 35 to 52, allowing for significantly smoother representations of growth and decay. Giving each parameter seven degrees (plus straight), and revising the canonical names to a numerical series, would also make the tool more intuitive.

An alternative and simpler solution is to keep the existing number of gradations but make the steps in amplitude less pronounced, linear rather than exponential. Largest could be overwritten with Large and a new set of glyphs, halfway between Large and Largest, could overwrite Large. The question is, does having nine trill speeds, seven vibrato speeds, and five levels of vibrato presence meet most users' needs? Probably. But does this reflect Western musical beliefs and practices, or will it meet the needs of others whose musical craft was relatively invisible before YouTube?

5. CONCLUSIONS

Vibrato has had a long invisible period. We celebrate the development of a visionary notation tool that allows the emotional life of tones to be seen in a graphically nuanced and globally accessible way. This has potential relevance for anyone interested in the art of vibrato, be they educators, performers, composers, or musicologists.

When launching Bravura and SMuFL in 2013, Daniel Spreadbury defined three of the problems he set out to solve as follows: "to allow for easy extensibility in the event that new symbols are dreamed up or there are omissions; to provide a framework for the development of new music fonts; and to develop a community around the standard, so that the wisdom of experts in different fields of music can be brought to bear" [15]. Bravura represents thousands of hours of human labor – it is a stunning achievement, collating centuries of loving endeavor by music engravers. The vibrato glyphs barely scratch the surface of what this font offers, but their significance may be far greater than Spreadbury and his team at Steinberg anticipated. Possessing the technology to look at vibrato more closely makes it easier for performers to address technical limitations and to handle bias, waking us up to possibilities that lie beyond the habits, clichés, and beliefs of the culture we happen to be bred in. These glyphs help a dominant group become more aware and respectful of vibrato cultures in outgroups that do not share the privilege or self-confidence of the Western musical culture that derives its success, in no small part, to intimidatingly beautiful notation. For better or worse, notation reifies. A tool that allows everyone to visualize other vibrato cultures with beauty and nuance helps to level the playing field. That would be revolutionary.

Existing glyphs	*Proposal 1*	*Proposal 2*
	Straight	Presence0
Smallest	Smallest	Presence1
	SmallerStill	Presence2
Small	Smaller	Presence3
	Small	Presence4
Medium	Medium	Presence5
	Large	Presence6
Large	Larger	Presence7
	LargerStill	Presence8
Largest	Largest	Presence9

Figure 15. Proposed nomenclature for seven gradations of wavelength

6. REFERENCES

[1] M. Nowakowski and A. Hadjakos, "Online survey on usability and user experience of music notation editors," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'2023*, A. P. D. Ritis, V. Zappi, J. V. Buskirk, and J. Mallia, Eds. Boston, Massachusetts, USA: Northeastern University, 2023, pp. 6 – 15.

[2] F. Schuiling, "Notation cultures: Towards an ethnomusicology of notation," *Journal of the Royal Musical Association*, vol. 144, no. 2, pp. 429–458, 2019. [Online]. Available: https://doi.org/10.1080/02690403.2019.1651508

[3] J. Lependorf, "Contemporary notation for the shakuhachi: A primer for composers," *Perspectives of New Music*, vol. 27, no. 2, pp. 232–251, 1989. [Online]. Available: http://www.jstor.org/stable/833414

[4] Vick Gevorgyan, "Duduk lesson 9: Vibrato," YouTube video, 2019, Duduk vibrato lesson, channel "Duduk Lessons". [Online]. Available: https://www.youtube.com/watch?v=rQziCdC_wak

[5] Park Jiha, "Piri solo 'Sangyeongsan'," YouTube video, Nov 2010, Korean traditional piri performance.

[Online]. Available: https://www.youtube.com/watch?v=FDHwdnCausQ

[6] A. O. Har, "The ghost: Opera – an intersection of greek tragedy, electroacoustic, and spectral music," Ph.D. dissertation, 2021. [Online]. Available: https://hdl.handle.net/2123/25087

[7] ——, "Ma and traditional japanese aesthetics in spatial music and sonic art," *Organised Sound*, p. 1–12, 2025.

[8] K. Mikolajczyk, A. O. Har, L. Wachsmann, O. Bown, and S. Ferguson, "Mimetic possibilities: Collaboration through movement in multimedia opera," *Leonardo (Oxford)*, pp. 1–15, 2024.

[9] B. Brown, "Five types of vibrato: Exploring the possibilities on doublepipes," https://youtu.be/FFQrtazhX1A, May 2020, youTube video.

[10] "The myth of orron and oros," https://www.babelscores.com/catalogs/music-and-scene/7953-the-myth-of-orron-and-oros, n.d., babelScores Catalog.

[11] "Metrics and glyph registration for scoring applications," in *Standard Music Font Layout (SMuFL) Version 1.4*, D. Spreadbury, Ed. W3C Music Notation Community Group, 2021, see also diagram at https://w3c.github.io/smufl/latest/specification/repeat-offsets.html. [Online]. Available: https://w3c.github.io/smufl/releases/1.4/specification/scoring-metrics-glyph-registration.html

[12] G. Pecoraro, D. F. Curcio, and M. Behlau, "Vibrato rate variability in three professional singing styles: Opera, rock and brazilian country," *Proceedings of Meetings on Acoustics*, vol. 19, p. 035026, 2013. [Online]. Available: https://doi.org/10.1121/1.4801034

[13] String Technique, "Developing vibrato for cello," https://youtu.be/Ki0miM7wQbw, 2018, youTube video.

[14] "About smufl," https://www.smufl.org/about/, n.d., standard Music Font Layout website.

[15] D. Spreadbury. (2013, May) Introducing bravura, the new music font. Dorico Blog, Steinberg. [Online]. Available: https://blog.dorico.com/2013/05/introducing-bravura-music-font

ENGRAVINGGNN: A HYBRID GRAPH NEURAL NETWORK FOR END-TO-END PIANO SCORE ENGRAVING

Emmanouil Karystinaios[1] **Francesco Foscarin**[2] **Gerhard Widmer**[1,3]
[1] Johannes Kepler University, Linz, Austria
[2] Moises Systems Inc., Salt Lake City, USA.
[3] LIT AI Lab, Linz Institute of Technology, Austria
firstname.lastname@jku.at

ABSTRACT

This paper focuses on automatic music engraving, i.e., the creation of a humanly-readable musical score from musical content. This step is fundamental for all applications that include a human player, but it remains a mostly unexplored topic in symbolic music processing. In this work, we formalize the problem as a collection of interdependent subtasks, and propose a unified graph neural network (GNN) framework that targets the case of piano music and quantized symbolic input. Our method employs a multi-task GNN to jointly predict voice connections, staff assignments, pitch spelling, key signature, stem direction, octave shifts, and clef signs. A dedicated postprocessing pipeline generates print-ready MusicXML/MEI outputs. Comprehensive evaluation on two diverse piano corpora (J-Pop and DCML Romantic) demonstrates that our unified model achieves good accuracy across all subtasks, compared to existing systems that only specialize in specific subtasks. These results indicate that a shared GNN encoder with lightweight task-specific decoders in a multi-task setting offers a scalable and effective solution for automatic music engraving.

1. INTRODUCTION

The musical score encodes a variety of information that we can organize into two classes: the *musical content*, which specifies the sounds that a musician should produce, and the *score engraving*, i.e., the notation symbols which don't influence the music, but are employed for its graphical representation [1]. Examples of engraving information include staff separation, clefs, pitch spelling, grouping into chords and voices, octave shifts, and stem directions. A high-quality engraving is a fundamental component of a musical score as it allows musicians and musicologists to efficiently read it and get the composer's intentions with precision.

The situation changes significantly when we exclude the human component and consider music that must be played or composed by a machine, such as a sequencer. In this scenario, formats that only encode the musical content, like a MIDI-file, are much more efficient and easier to handle. This is visible in the field of music information retrieval (MIR) where the vast majority of systems, for example, automatic music transcription or automatic composition systems, work on MIDI-like representations. This task of *automatic music engraving* is then a necessary bridge to allow musicians to play computer-processed music.

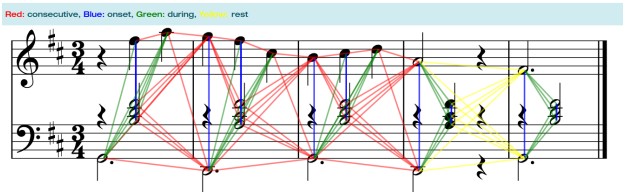


Figure 1. The score graph, nodes are notes and edges are marked in different color for each type.

Some systems were recently proposed to address individual engraving subtasks such as pitch spelling and key estimation [2, 3], voice and staff separation [4]. We use the latter as a basis for our system, and improve it by extending it to more engraving tasks without reducing its performance. Another related work is the one from Beyer and Dai [5], which targets in a single end-to-end framework the tasks of note quantization and music engraving. This is an interesting direction for automatic music transcription (AMT), but we keep our model more generic to be able to employ it in diverse scenarios (e.g., with a music generation system).

In this paper, we focus specifically on piano music and quantized inputs, introducing EngravingGNN: a unified framework for automatic score engraving. At its core, EngravingGNN employs a Hybrid-GNN encoder, a combination of heterogeneous graph convolutions and stacked GRU layers [6], to fuse relational note interactions with long-term temporal context. From this shared embedding, lightweight decoder heads simultaneously predict all of the following:

- *Voice edges* and *staff labels*, organizing notes into independent streams on the appropriate stave,
- *Pitch spelling* and *key signature*, ensuring correct accidentals and tonal context,
- *Stem direction* and *octave shift* markings, adhering to engraving conventions for clarity,

* Equal contribution.

Copyright: ©2025 First author et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

- *Symbolic duration* predicting dots, triplets, and notehead types (whole, half, eight, etc.),
- *Clef signs*, producing the initial clefs and capturing mid-piece clef changes when they occur.

In summary, our contributions are 3-fold:

1. A end-to-end model network architecture for music engraving, with an *Hybrid-GNN* encoder integrating heterogeneous graph convolutions with GRU stacks, plus 10 decoder heads for all core engraving tasks.
2. A postprocessing and export pipeline yielding fully engraved MusicXML/MEI scores.
3. Evaluation and comparison with existing systems on two piano corpora, with detailed metrics for voice, staff, pitch spelling, key, stem direction, symbolic duration, octave shifts, and clefs.

2. METHODOLOGY

Figure 2 provides a schematic overview of our approach. Starting from a quantized score, we construct an input graph, which is then encoded using a graph neural network (GNN). Multiple tasks are decoded in parallel from this encoded representation, followed by postprocessing and export of the final musical score.

2.1 Input Graph Construction

Given quantized notes, we create a directed heterogeneous graph $G_{in} = (V, E_{in}, \mathcal{R}_{in})$:

- **Nodes** V: one per note, with features:
  - pitch class (12-way one-hot),
  - octave (scalar in $[1, 7]$),
  - normalized duration $d \in (0, 1)$ via $\tanh(\frac{\text{note } d}{\text{bar } d})$,
  - onset fraction within bar and downbeat positions.
- **Edge types** $\mathcal{R}_{in}$: onset, during, follow, silence, plus inverse edges, as in [4]

In the current version of our system, grace notes are removed as they pose further challenges due to the lack of a quantized duration.

2.2 Output Predictions

Our model produces a rich set of engraving annotations by formulating each attribute as either a node-classification or an edge-prediction task. Given the GNN-computed embedding $\mathbf{h}_v$ for each note v, and embeddings $(\mathbf{h}_u, \mathbf{h}_w)$ for each candidate note-pair (u, w), we jointly predict the following:

1. **Voice edges.** We treat voice assignment as an edge-prediction problem over a restricted set of candidate pairs Λ_v as in [4]. Each candidate $(u, w) \in \Lambda_v$ satisfies that u and w lie in the same bar and $\text{offset}(u) \leq \text{onset}(w)$, ensuring temporal consistency. A 2-layer MLP, followed by a sigmoid function, takes the concatenated pair $[\mathbf{h}_u; \mathbf{h}_w]$ and outputs a probability $p_{uw} \in (0, 1)$ of w immediately following u in the same voice stream.
2. **Staff label.** Each note is assigned to one of two staves (upper or lower) via a binary node classifier. A 2-layer MLP, followed by a sigmoid function, maps $\mathbf{h}_v$ to a probability s_v. At inference time, we threshold s_v at 0.5 to decide staff membership. This separation is crucial for later beam grouping and rest insertion.
3. **Pitch spelling.** Correct notation requires exact accidentals. We predict one of 35 classes per node (7 natural pitch classes $\{A, B, C, D, E, F, G\}$, each with optional $\sharp$, $\flat$, $\times$, $\flat\flat$, or $\natural$ sign). A 2-layer softmax MLP takes $\mathbf{h}_v$ and outputs a distribution over these classes, trained via categorical cross-entropy against MusicXML ground-truth spellings.
4. **Key signature.** MusicXML represents key signatures as an integer in $[-7, +7]$ (number of fifth-circle sharps or flats). We add a sixteenth node-classification head: a 2-layer MLP with softmax over 15 classes. Local key changes (a.k.a. modulations) occur in Romantic repertoire, so we predict this attribute per note and later smooth them within measures.
5. **Stem direction.** Stem orientation can be up or down or no-stem. We model this as a ternary classification per note: Postprocessing enforces consistency within chords and aligns with beam direction.
6. **Octave shift.** Octave transposition indications (e.g. 8va, 8vb, 15ma) are common in piano music. We predict a multiclass label per note among $\{\texttt{none}, 8\texttt{va}, 8\texttt{vb}, 15\texttt{ma}\}$ via a softmax MLP. Consecutive notes sharing the same label are later merged into a single octave-shift bracket in the score.
7. **Clef sign.** Piano scores occasionally employ clef changes (e.g. switch from G- to F-clef on the upper staff). We predict one of three classes: G-, F-, or C-clef (this is rare in piano music, but we still support it for completeness) per note via a softmax head. During export, we insert clef-change tokens at the first note of each predicted clef region.
8. **Symbolic duration.** In addition to the quantized onset and offset, we must determine the proper notehead type ($\frac{1}{16}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}, 1, \ldots$), the number of augmentation dots, and any tuplet bracket (e.g. triplet) attributes. We discretize "note-type" into classes $\{\texttt{whole}, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{6}, \ldots\}$, "dots" into $\{0, 1, 2, 3\}$, and "tuplet" into common ratios $\{1, 3, 5\}$ (i.e. no tuplet, triplet, quintuplet). Each is predicted by its 2-layer softmax MLP on $\mathbf{h}_v$. Postprocessing then groups adjacent notes with identical tuplet flags into a single bracket and assigns dot placement.

All node-classification heads input the same latent embedding from the GNN encoder. Each head has its respective

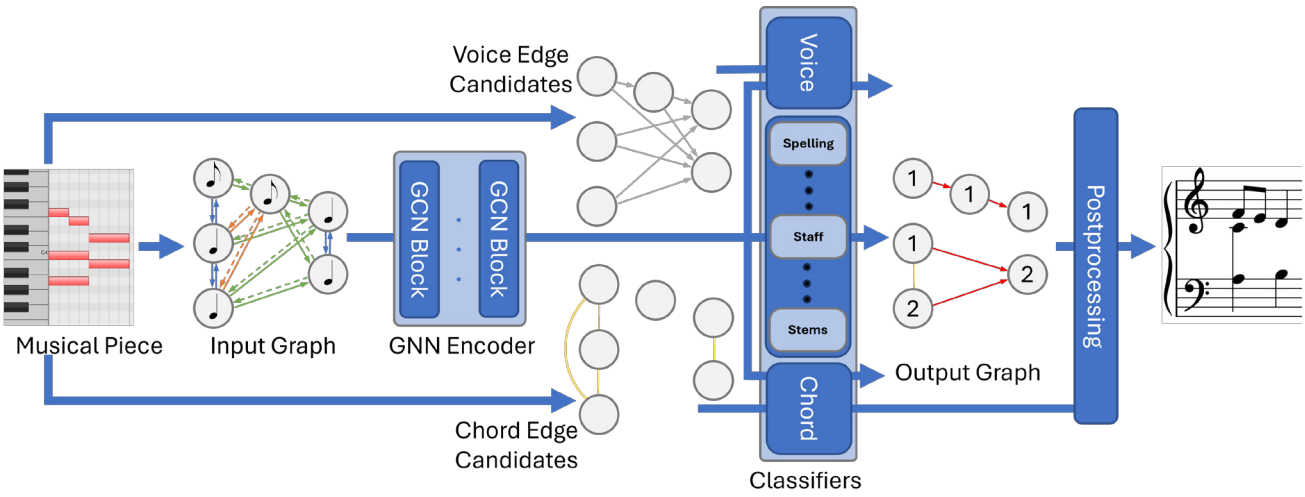


Figure 2. System overview: Input graph → GNN encoder → multi-task decoders → postprocessing → engraved score.

cross-entropy loss, and the voice-edge head uses binary cross-entropy over Λ_v. We sum these losses for end-to-end training.

2.3 GNN Encoder

Our backbone encoder is a *HybridGNN* that interleaves heterogeneous graph convolutions with recurrent GRU updates, combining the strengths of relational message-passing and sequential modeling [6]. By stacking these hybrid blocks, we capture both the rich, structured interactions between notes (via the graph convolutions) and the long-term temporal persistencies across the piece (via the GRU stacks).

Formally, let each node u in our input graph $G_{in} = (V, E_{in}, \mathcal{R}_{in})$ have an initial feature vector $\mathbf{h}_u^{(0)} \in \mathbb{R}^d$ encoding pitch class, octave, normalized duration, bar index, etc. We apply $L = 3$ identical hybrid layers. At layer l, we first perform a heterogeneous GraphSAGE convolution [7]:

$$\tilde{\mathbf{h}}_u^{(l)} = \sigma\Big(\mathbf{W}_0^{(l)}\,\mathbf{h}_u^{(l-1)} + \sum_{r\in\mathcal{R}_{in}}\sum_{v\in\mathcal{N}_r(u)} \mathbf{W}_r^{(l)}\,\mathbf{h}_v^{(l-1)}\Big),$$

where $\mathcal{N}_r(u)$ is the set of neighbors of u connected by relation type r, and $\{\mathbf{W}_r^{(l)}\}$ are relation-specific weight matrices. This step integrates multi-relation context such as onset, during, follow, and silence edges.

In parallel, we feed the initial feature vectors $\tilde{\mathbf{h}}_u^{(0)}$ into a gated recurrent unit (GRU) cell, the vectors are structured timely thanks to GraphMuse sampling process [6].

Because we stack one GRU per graph convolution layer, the network can accumulate and preserve information over long note sequences, learning temporal dependencies that pure GNN layers might lose. The GRU update also allows each node to "remember" its evolving embedding across layers, reinforcing global structure.

After $L = 3$ such hybrid blocks, each note u has a final embedding $\mathbf{h}_u^{(L)} \in \mathbb{R}^{256}$ that fuses local relational context with long-range sequential patterns. These embeddings serve as the shared inputs to all downstream decoders (voice-edge prediction, staff label, pitch spelling, etc.), enabling coordinated multi-task learning.

2.4 Multi-Task Decoders

Each task uses a lightweight 2-layer MLP on $\mathbf{h}_u$ (or $[\mathbf{h}_u; \mathbf{h}_v]$ for edge tasks):

- **Binary heads**: voice-edge candidate and chord-edge candidates (sigmoid output + BCE loss).
- **Softmax heads**: staff, stem, pitch spelling, key signature, octave shift, clef, and symbolic duration (categorical cross-entropy).

2.5 Training

We train end-to-end, minimizing the (unweighted) sum of all task losses. We employ Adam optimizer (lr=1e-3, weight decay 5×10^{-4}), 100 epochs. The train hyperparameters are taken from [4].

2.6 Implementation Details

For the encoder, we use a hidden size of 256 for both the heterogeneous graph convolution outputs and the GRU hidden state. Each convolutional layer is followed by a ReLU activation, and we apply 50% dropout to the convolution output before it enters the GRU. To stabilize training, layer normalization is applied both inside each GRU cell and between successive graph convolution blocks.

2.7 Postprocessing

A naive way to turn our voice-edge probabilities into a voice separation would be to simply apply a threshold and keep all edges above it. In practice, this can still produce three kinds of engraver's nightmares:

1. Distinct voices collapse into one,
2. A single voice splits into multiple disjoint streams,
3. Notes that belong together in the same chord end up in different voices.

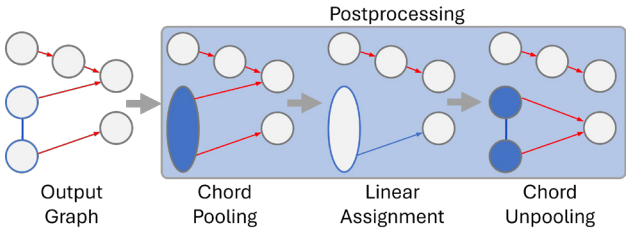


Figure 3. Postprocessing pipeline: chord pooling → voice assignment → unpooling → beaming & rests.

To guarantee musically valid output, we therefore follow prediction with a small postprocessing pipeline that enforces engraving constraints, as proposed in [4].

As shown in Figure 3, we perform the following steps in order:

1. **Chord pooling**: cluster synchronous notes (predicted chord edges) into virtual nodes.

2. **Voice assignment**: Once chords are fused, the score reduces to purely monophonic streams, so we can solve a linear assignment via the Hungarian algorithm [8] on pooled nodes.

3. **Unpooling**: restore original notes, distributing assigned voices.

4. **Rest infilling**: insert rests to fill each bar per voice.

We note that some node-level labels are altered by the chord-postprocessing to guarantee cohessiveness and proper engraving throughout predictions that need to carry the same label. Such labels include: symbolic duration, stem direction, and key signature. The logits of the notes that are pooled together for these tasks are averaged. All other node-level attributes, i.e. octave shift, staff, pitch spelling, and clef sign, are independent of the post-processing operations.

3. EXPERIMENTS

3.1 Datasets

We evaluate on two piano corpora that span simple to highly complex repertoire. The *J-Pop* set [9] (811 scores, 20% held out) features straightforward pop arrangements—lower-staff chords with simple upper-staff melodies. In contrast, the *DCML Romantic Corpus* [10] (393 scores, 20% held out) includes 17th–20th-century works with virtuosic textures, cross-staff beaming, multiple overlapping voices, and complex engravings. We convert all sources to MusicXML (using MuseScore and Partitura [11]) and split each corpus into 80 % train/validation and 20% test sets, ensuring that our model's ability to handle intricate engraving scenarios does not come at the cost of simpler pieces.

In line with Foscarin et al. [4], we train EngravingGNN as a single, unified model on the combined training sets of both J-Pop and DCML Romantic.

3.2 Evaluation Metrics

We evaluate each engraving subtask with measures tailored to its output type. Voice separation is judged using the homophonic F1 metric from Hiramatsu et al. [12]. Chord prediction is essentially a binary decision for each pair of synchronous notes: either they belong to the same chord or not. We report the standard F1 score over all candidate chord-edges.

All remaining engraving attributes—staff assignment, stem direction, clef sign, key signature, pitch spelling, octave shift, and symbolic duration features (note head type, augmentation dots, tuplet bracket)—are treated as per-note classification tasks. For each attribute, we compute the simple classification accuracy, i.e., the percentage of notes for which the predicted label exactly matches the ground truth. In the case of symbolic duration, which is decomposed into three separate predictions (note type, number of dots, tuplet flag), we measure both each individual head's accuracy and the overall rate at which all three sub-predictions are simultaneously correct.

All results are aggregated across the test set using micro-averaging, ensuring that pieces with more notes or edges contribute proportionally to the final scores.

3.3 Results

Table 1 presents the test-set performance of EngravingGNN alongside the baselines on both the J-Pop and DCML Romantic corpora. On the J-Pop dataset, EngravingGNN achieves a voice-separation F1 of 96.8—slightly above the previous GNN-only result (96.6) and well ahead of Shibata et al. (92.2). Chord grouping also improves to 96.9, and staff assignment climbs to 97.6%. While the specialized spelling model of Karystinaios and Widmer [6] attains 99.4% pitch and 96.9% key-signature accuracy, EngravingGNN's integrated predictions still perform strongly at 96.3% and 80.6%, respectively. We believe that the reason for the degraded performance is the absence of data augmentation, which may be crucial for pitch spelling on a dataset without much key variability. Stem direction (85.1%), octave shifts (99.9%), clef sign (96.2%), and symbolic duration (87.8%) likewise demonstrate high end-to-end engraving quality.

On the more demanding DCML Romantic corpus, which is characterized with attributes such as cross-staff beaming, frequent modulations, and dense polyphony, EngravingGNN again leads against the other models. Voice F1 rises marginally to 90.6 (vs. 89.9), chord F1 to 81.1 (vs. 79.5), and staff accuracy to 91.9% (vs. 91.0%). Notably, it matches the standalone spelling model on pitch (93.5%) and even surpasses it on key signature (50.5% vs. 46.5%), highlighting robust handling of chromatic passages and possible positive inter-task transfer. Stem direction (73.6%) and symbolic duration (83.3%) reflect the corpus's notational complexity. Overall, EngravingGNN delivers consistent, high-fidelity results across both straightforward and intricate piano repertoires.

We remind the reader that the above results are from a single unified model, trained on the combined training sets of both J-Pop and DCML Romantic. As Table 1 shows,

Model	Voice F1	Chord F1	Staff Acc	Pitch Spelling Acc	Key Acc	Stem Acc	Octave Acc	Clef Acc	Sym. Dur. Acc
J-Pop									
Spelling Model [6]	-	-	-	**99.4**	**96.9**	-	-	-	-
Shibata et al. [9]	92.2		92.8	-	-	-	-	-	-
Forscarin et al. [4]	96.6	94.9	96.3	-	-	-	-	-	-
EngravingGNN	**96.8**	**96.9**	**97.6**	96.3	80.6	85.1	99.9	96.2	87.8
DCML Romantic									
Spelling Model [6]	-	-	-	93.5	46.5	-	-	-	-
Shibata et al. [9]	84.9		88.5	-	-	-	-	-	-
Forscarin et al. [4]	89.9	79.5	91.0	-	-	-	-	-	-
EngravingGNN	**90.6**	**81.1**	**91.9**	93.5	**50.5**	73.6	100	90.0	83.3

Table 1. Comparison of the test-set results for our EngravingGNN model versus the Foscarin et al. [4] and Shibata et al. [9] on the voice and staff tasks and versus the Spelling model of [6] for pitch spelling and Key signature prediction.

this shared model delivers strong, consistent performance on both test sets, demonstrating EngravingGNN's ability to generalize across diverse piano repertoires without per-dataset tuning.

We do not evaluate against the model of Beyer and Dai [5] since it would not be a fair comparison, as the quantization task they perform is very complex and diminishes the performance of the score engraving subpart of their system.

3.4 Qualitative Analysis

Figure 4 illustrates our predictions on the first two bars of the first movements of Mozart's Sonata K310. Our method correctly captures symbolic duration, key, voices, staff, stem-direction, and spelling for each single note. We can also see a wrong prediction on the second half of the first beat, where the staff of an E which causes a cross staff assignment.

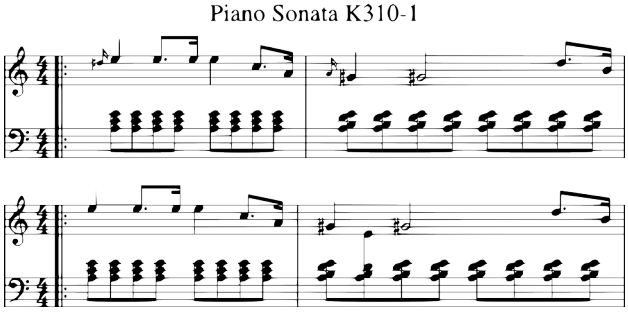


Figure 4. Engraving prediction on Sonata K310-1 by Mozart. Ground truth score on the top and EngravingGNN prediction on the bottom. Note that EngravingGNN, is not able to predict grace notes.

4. DISCUSSION AND LIMITATIONS

4.1 Treating Tied Notes

In its current version, our framework does not handle ties across multiple noteheads. In traditional engravings, ties connect notes across bars, across strong beats for better readability, or to form durations that cannot be expressed by one symbol alone. Since EngravingGNN's duration head predicts only one symbol per input duration, it cannot generate the necessary tie-edges or composite duration encodings. For this, autoregressive systems like the one of Bayer and Dai [5], which are naturally able to handle the "one-to-many" case, have a clear advantage. We believe that adding an autoregressive routine for the notehead task would be the way to improve in this direction.

4.2 Including Grace Notes

Our system does not currently support grace notes because grace notes lack quantized durations. However, grace notes are of crucial importance in many Baroque, Classical, and jazz passages, as they convey stylistic nuance. Proper engraving of grace notes would require flagging them in the input graph, linking them to their principal notes, and rendering them with small-noteheads with appropriate slurs. Extending EngravingGNN to handle these ornaments is not trivial but it would be a key step toward fully authentic and stylistically complete piano scores.

4.3 Postprocessing

While our postprocessing pipeline successfully pools chords, assigns voices, groups beams, and infills rests to produce valid MusicXML/MEI output, several limitations remain. First, symbolic-duration handling is limited to single-symbol durations and fixed tuplet brackets, so complex ties, composite rhythms, or wrong predictions can cause rendering errors in symbolic music visualizers. Second, global attributes such as key signature may exhibit measure-to-measure discontinuities, as we apply predictions per note without sequence-level smoothing. Finally, our beam grouping and rest-infilling steps rely on deterministic, rule-based algorithms rather than learned patterns, which can result in suboptimal engraving choices in corner-case scenarios (e.g. irregular beaming across voices or nonstandard rhythmic groupings).

4.4 Training and Optimization

Hyperparameters were selected from prior work [4]. We did not perform an extensive hyperparameter search over learning rates, layer depths, or batch sizes was conducted. Consequently, while our off-the-shelf settings are enough for our goal of showcasing the potential of our approach,

further gains may be possible through systematic hyperparameter search.

5. CONCLUSION AND FUTURE WORK

In this paper, we have presented EngravingGNN, a unified Hybrid-GNN framework that, with a single encoder and multi-head decoders, simultaneously predicts all essential piano engraving attributes, such as voice edges, staff assignment, pitch spelling, key signature, stem direction, octave shifts, and clef changes. After post-processing, our system generates ready-to-print MusicXML/MEI scores directly from quantized input, significantly reducing the manual effort traditionally required for high-quality engraving.

Despite these advances, our current model assumes each note's duration fits a single symbol in our predefined vocabulary, and does not handle tied notes or composite rhythms that span multiple noteheads or measures. It also leaves layout optimization, system breaks, staff spacing, and beam slopes, to external tools. Future work will extend the decoder to detect and represent ties and composite durations. Furthermore, we aim to explore end-to-end training with differentiable rendering feedback (e.g., via MEI/Verovio) to further close the gap between automated and professional human engraving.

6. ACKNOWLEDGEMENTS

This work was supported by the European Research Council (ERC) under Horizon 2020 grant #101019375 "Whither Music?".

7. REFERENCES

[1] F. Foscarin, P. Rigaux, and V. Thion, "Data quality assessment in digital score libraries: The gioqoso project," *International Journal on Digital Libraries*, vol. 22, no. 2, pp. 159–173, 2021.

[2] F. Foscarin, N. Audebert, and R. Fournier S'niehotta, "PKSpell: Data-driven pitch spelling and key signature estimation," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

[3] A. Bouquillard and F. Jacquemard, "Engraving oriented joint estimation of pitch spelling and local and global keys," *arXiv preprint arXiv:2402.10247*, 2024.

[4] F. Foscarin, E. Karystinaios, E. Nakamura, and G. Widmer, "Cluster and separate: a gnn approach to voice and staff prediction for score engraving," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2024.

[5] T. Beyer and A. Dai, "End-to-end piano performance-midi to score conversion with transformers," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2024.

[6] E. Karystinaios and G. Widmer, "Graphmuse: A library for symbolic music graph processing," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2024.

[7] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *IEEE Data Engineering Bulletin*, vol. 40, no. 3, pp. 52–74, 2017.

[8] R. E. Burkard and E. Cela, "Linear assignment problems and extensions," in *Handbook of combinatorial optimization*. Springer, 1999, pp. 75–149.

[9] K. Shibata, E. Nakamura, and K. Yoshii, "Non-local musical statistics as guides for audio-to-score piano transcription," *Information Sciences*, vol. 566, pp. 262–280, 2021.

[10] J. Hentschel, Y. Rammos, F. C. Moss, M. Neuwirth, and M. Rohrmeier, "An annotated corpus of tonal piano music from the long 19th century," *Empirical Musicology Review*, vol. 18, no. 1, pp. 84–95, 2023.

[11] C. E. Cancino-Chacón, S. D. Peter, E. Karystinaios, F. Foscarin, M. Grachten, and G. Widmer, "Partitura: A python package for symbolic music processing," in *Proceedings of the Music Encoding Conference (MEC)*, Halifax, Canada, 2022.

[12] Y. Hiramatsu, E. Nakamura, and K. Yoshii, "Joint estimation of note values and voices for audio-to-score piano transcription." in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2021, pp. 278–284.

MYKDF — DER KUNST DER FUGE OWN MIXDOWN, REALIZED WITH TSCORE AND SIG/ADLIB

Markus Lepper
semantics gGmbH
Berlin, DE
post@markuslepper.eu

Baltasar Trancón y Widemann
Technische Hochschule
Brandenburg, DE

ABSTRACT

The project myKdF gives music enthusiasts an interactive computer application which allows listening to *Die Kunst der Fuge* by Johann Sebastian Bach in dynamically changeable self-defined arrangements. This intends a deeper exploration of different aspects of this work by users not capable of playing it themselves on an instrument. Main result is a stand-alone Java application, which can run on any contemporary computer system.

The data model is constructed on the tScore meta-meta-model for the semantics of music notation; sound synthesis and user interaction are realized with the sig/adlib system for realtime signal processing. Furthermore, the data model has been used to produce a print publication and for automated extraction of statistical data and structural properties.

Notation systems and languages with very different contents play a fundamental role in the overall architecture.

1. INTRODUCTION

The initial idea of the project myKdF is to give music enthusiasts an interactive computer application which allows listening to *Die Kunst der Fuge* by Johann Sebastian Bach in dynamically changeable self-defined arrangements. This intends a deeper exploration of different aspects of this work by users not capable of playing it themselves on a keyboard or in a chamber music ensemble.

The second idea is to make the necessary computer model of that work available for automated analysis, by ourself and others.

Third, the strengths and weaknesses of the relatively novel and still developing products tScore and sig/adlib have heavily been tested by programming myKdF.

Notation plays a fundamental role in this project: historical sources stand against contemporary technology, and the translation between the different formats (*meta-models*) is not just mere necessity, but brings further insight into their semantic rules, expressiveness, and limitations. Here three notation meta-models have been constructed with our meta-meta-model tScore—others have been programmed ad hoc.

Copyright: © 2025 Author Name. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

The software can be downloaded as a executable jar file from `http://bandm.eu/downloads/myKdF.jar`, the sources (Java, tScore, etc.) are available under CC-BY-SA-NC license on a source hosting platform.

The work is still in progress and some minor features described in the following may be not yet operative.

2. DEGREES OF FREEDOM FOR THE LISTENERS

Figure 1 shows a screen shot of the application. The user can select one of the 18 movements , arrange its visual and audio representation, and operate the player. The movements are called *contrapunctus* in the original score and in this article—abbreviated like cp 1.

The very fist idea had been to give the listeners the freedom of assigning volume, timbre (=waveform and envelope), and panorama position to each voice separately. It soon turned out that these assignments bring more didactic effect when applicable not only to voices as a whole, but to distinct musical material within.

These material layers are projections of different aspects of the composition. The very first layer is always present and contains *mere counterpoint*, that is melodic material which has not been qualified as thematic. This includes (naturally) everything which sounds in the so-called *episodes* (German: *Zwischenspiele*), in which no theme entry in any voice. The other layers are all theme entries, which can be categorized in different ways, depending on the contents of the selected contrapunctus—see section 3.2 for details.

The user can further start and pause the sequencer and move its position. They can select different tuning modes (see below). The role of the tuning parameter slider depends on the selected tuning algorithm.

Each voice can be given a panorama position. The sliders for the materials are labelled according to the selected audio representation; sliders not required in the current contrapunctus appear disabled.

For each material the volume, the wave form (sine/square/saw/triangle) of the table-driven synthesis and the sound envelope (legato, meno legato, portato, staccato) can be selected. [1]

The area under the mixer shows the formplan of the contrapunctus. The different appearances of the thematic boxes

[1] For convenience, each of these selections can be distributed to all materials of the same voice by holding the "shift" key when releasing the mouse, to the corresponding materials of all voices by "control" and to all materials of all voices by holding both keys.

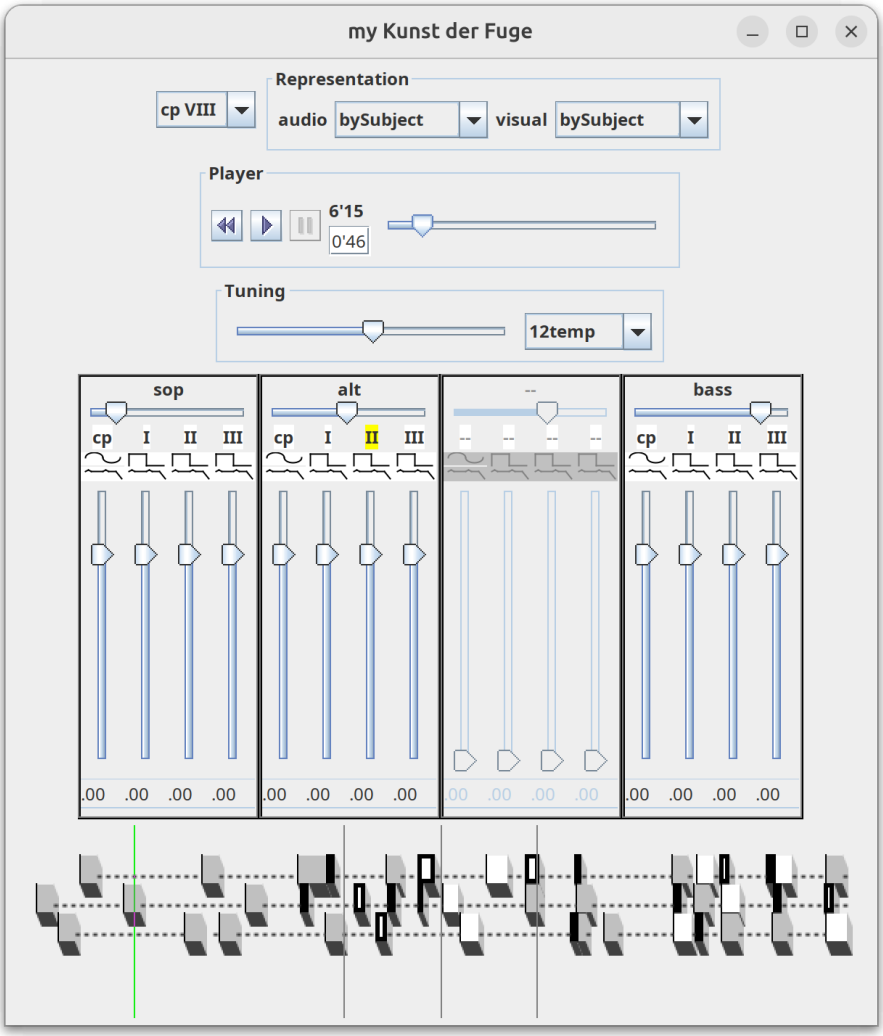


Figure 1. Screenshot of the application.

realize the selected *visual representation*. The separator lines mark the center of the contrapunctus and the positive and negative Golden Ratio. The green line is a cursor which moves synchronously with the sound.

3. SCORE DATA

3.1 Original Music Model

The project is based on the data files which contain the scores of the contrapunctus of *Die Kunst der Fuge*. These data are *models* which have one common definition of syntax and semantics, the *meta-model* `Score_cwn`. These in turn are instances of the *meta-meta-model* TScore. All these three architecture levels are realized as Java classes and instances. [1, 2, 3]

Our edition is almost completely identical with that by Wolfgang Graeser. [2] [4] The print version of our model can be downloaded from `http://bandm.eu/downloads/kdf.pdf`.
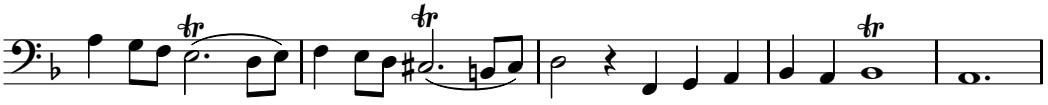
[2] We skip his "contrapunctus XVIII", which is just an arrangement for two harpsichords of cp 17, and use this number for the very last, unfinished movement.

TScore is a text-based formalism and its implementation for easy notation of any time-indexed data. The fundamental paradigm is co-algebraic: Voices and timepoints are distinct objects—necessary for all of them is an identifying name but no further properties. Each pair of voice and timepoint can constitute an event. Parameters can freely be attached to voices, timepoints, and events, without contributing to their identity.

The input text format follows the well-proven format of a conventional orchestra score: time flows from left to right, broken into staves which are vertically stacked; voices are vertically stacked within a staff, and synchronous events are more or less horizontally aligned.

Beyond these basic rules, everything is pluggable: The time model itself, all event parameters, their representation syntax and the resulting semantic data objects.

`Score_cwn` is a meta-model for a basic version of Common Western (Music) Notation. It supports easy notation of rhythms defined by metrics, changes of metric, functional pitches (= pair of white note and accidental = *spelled pitches (SP)* as defined by [5]), with "running octaves" and in different locales, dynamics, articulation, and ornaments.



```
T        8                  9                    10 !    !     11    !  !    12
VOX bass  a (g f)e - - (d e) f (e d)cis - -(h cis)d  % ,f  g   a  b  a   b       a
    artik       trX(     )       trX(      )                          tr
```

Figure 2. Historic notation vs. semantic encoding, cp 17 ms. 8–11.

```
T          156   157 158  162   163 167    168  169    172        173 174
VOX sop                                    - II                      -
VOX alt    - .III              x                        -.. IIu/2>    x
VOX tenor                          - .III                            x
VOX bass         - II      - ..x    - IIu> - x
```

Figure 3. Excerpt from the `formplan-fuge` score for cp 18. (Sign – stands for continued events; dots stand for metrical dotted notation; both are inherited from the tScore CWN generic format. For the specific symbols see Table 1.)

mat	::=	*cp* | *th* | `%`
cp	::=	`x` | (`1` | `2` | `3` | `4`) *modc*
modc	::=	`u`$^?$ & `>`$^?$
th	::=	*base* *mod*
base	::=	`D` | `C` | ((`I` | `II` | `III` | `IV`)(`D` | `C`)$^?$)
mod	::=	`u`$^?$ & `>`$^?$ & ((⋆ | /)(`2` | `3` | `4` | ...))$^?$

The symbol `%` stands for a pause; *cp* allows for different counterpoint material (only `x` and `1` are used in the context of myKdF); *base* identifies a theme and its dux or comes form; `u` indicates inversion (German *Umkehrung*); `>` indicates an incomplete entry; ⋆ and / stand for augmentation and diminuition of the rhythm by the given factor. The meta-operators _|_ stand for alternatives, _&_ for permutation and _$^?$ for optionality.

Table 1. Syntax of the `formplan-fuge` sub-language.

The syntax and semantics for all these parameters are pluggable. All parameter tracks can come in pairs, to separate Urtext and additions by the editors. Pitches must be monodic per voice; further extensions like chords and lyrics can be defined by subclassing. This has not been necessary for myKdF; contrarily, dynamics are not used at all, and articulation sparsely.

All tScore-based meta-models are ad-hoc extensible, so the source of cp 4 includes additionally the track `zach` which contains the annotations by Gerd Zacher, who relates this contrapunctus to the *Epistle to the Romans*. [6]

TScore is not intended to represent conventional notation in a one-to-one fashion, but to construct precisely defined *semantic* meta-models and models. The semantics of a notation system (= meta-model) can be defined as a *transformation network* from the graphic symbols in the written, printed, or displayed notation (on the left side of a network diagram as in Figures 5 and 6) to the audible parameters of an execution of a piece (on the right side). [7, 8] Then every encoding is a cut through such a network—so is every model and every tScore model. See for example Figure 2: We know from historically informed performance that the third trill symbol has a different meaning than the others: It does include an implicit *final turn* (German: *Nachschlag*). In case of the others, this is replaced by the explicit notes. Consequently, the semantic-oriented tScore encoding applies two different ornament keywords. These will be rendered identically in print, but differently in audio execution and statistical analysis.

The `Score_cwn` meta-model has a second parameter track `artikHRSG` for articulation and ornaments added by the editors. We have used it in cp 8 m. 24 and 160 bass to add the thematic mordent and in cp 9 eight times to add the thematic pralltriller to the middle of theme II. These additions correspond well to Baroque practices of notating and playing. They are also possible for the thematic fragments in cp 9 soprano m. 47 and bass m. 75.

The generic input format of the tScore meta-meta-model closely follows the well-proven traditional score format and is often suitable for sight-reading, and the syntax of all meta-models can be defined with minimal redundancy. Therefore the data input of the complete *Die Kunst der Fuge* took less than four workdays!

3.2 Formplan Data

As mentioned above, at a certain point of the work it became clear that the users shall be able to influence not only the distinct voices as a whole, but also the material layers within. These layers are defined by data in another tScore-based meta-model. This format inherits all metric and rhythmic notation means from the standard CWN score, but replaces the pitch information by the codes for the materials. Table 1 shows the syntax of this language and Figure 3 an example: For instance, the measure 172 in the alto voice is filled by two events. The first is doubly dotted (duration = $7/8$) and just a hold of whatever comes before. The second event (duration = $1/8$) is an entry of theme II in its inverted form, with all durations shortened by the factor two, and incomplete. In measure 174 some unspecified counterpoint material `x` begins.

	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII	XIII	XIV	XV	XVI	XVII	XVIII
by theme								●	●	●	●							●
by form	●	●	●	●	●	●	●			●	●	●				○	●	●
by tempo						●	●								○			
by direction					●	●	●			●	●	●			○	●		●

Table 2. Presentation modes offered per contrapunctus. (In XV: direction and tempo augmentation tightly coupled and appearing only once. In XVI: form hardly significant.)

A rendering can be seen in the bottom part of Figure 1.

Whenever the selected contrapunctus features theme entries which differ w.r.t. a particular category, then this category is offered to the user for visual and audio representation. These categories are

by theme for all contrapunctus where multiple themes are combined;

by form difference of dux and comes form, as fundamental for fugue theory;

by tempo whenever a theme appears in different augmentations and/or diminuitions;

by direction difference between recto and inverso, or even between all four dodecaphonic appearances: original, inverted, retrograde and inverted-retrograde. (Only the first two appear in *Die Kunst der Fuge*.)

The user can select which of these aspects shall be expressed in the visual representation in the form diagram shown below the mixer section and in the audio realization. Figure 1 shows the form diagram of cp 8 with different symbols per theme. The labelling of the audio sliders is determined by the selected audio representation mode.

In any case, the mere counterpoint is the very first category. The others are selectable for the different kinds of theme entries if and only if this aspect is realized in the selected contrapunctus, see the survey in Table 2.

When switching the contrapunctus, the audio and visual representation is selected automatically. There are different default priorities:

audio: byTheme → byTempo → byDirection → byForm
visual: byTheme → byDirection → byTempo → byForm

This is because the tempo difference is visible anyhow, by the width of the thematic blocks, without additional graphical means. (Incomplete entries are marked differently anyhow.)

4. THE sig/adlib REALIZATION

MyKdF is a stand-alone Java application. It is realized with sig/adlib, a Java library for sound synthesis and processing. It uses only the built-in means of the Java Runtime Environment (JRE). [10] For the size of the software see Table 3, measured with CLoC [9].

In principle, the sound synthesis could also be delegated to the built-in implementation of General MIDI. Then all sound manipulations could be mapped to MIDI controllers. Our approach (currently still) has the disadvantage of comparatively simple sound synthesis. But it also has significant advantages: (a) It is a totally stand-alone application which requires only a Java runtime environment, which is a JVM and a runtime library that interfaces the platform's sound system. (b) All sound synthesis is under program control and does not depend on the kind and quality of some General MIDI installation. (c) All SP-based tuning systems can be applied globally, not event-wise as for enharmonic MIDI keys.

sig/adlib singleton sources	1,797
sig/adlib parametric sources	2,985
sig/adlib expanded sources	13,593
sig/adlib/lib singleton sources	978
sig/adlib/lib parametric sources	1,774
sig/adlib/lib generated sources	6,680
myKdF Java sources	
model	281
view	499
control	68
analyses	967
tScore sources:	1,493

Table 3. Lines of code of the myKdF project, excluding blank lines and comments. Calculated by the program `cloc` [9].

4.1 The sig/adlib Network

The application follows a model–view–control architecture. The view is realized with components of the JRE Swing library, and the model is the sound-producing network constructed with sig/adlib. Its principal architecture is shown in Figure 4: The largest frame contains the circuit which is instantiated four times, once for each voice. It contains a smaller frame which is instantiated four times per voice for its material layers. (There are additional voices for the very last measures of cp 5 to cp 7 and cp 11, which are hidden to the user and left out in the diagram for readability.)

The very left column of the Figure shows the four wave tables (sine, square, triangle, and saw) which are distributed via a multiplexer to the table-driven generators. The multiplexer is switched by the user's choice in the control layer. In a similar way one of four envelopes is selected and applied to the sound.

The third column shows the eighteen sets of score data. Each consists of one score per voice, and an additional formplan score. The former are multiplexed and fed into a sequencer, one for each voice. The first column of the data is fed into a edge detector, which recognizes every event start and triggers the envelope, the frequency calculation and changing the material register. The next columns of

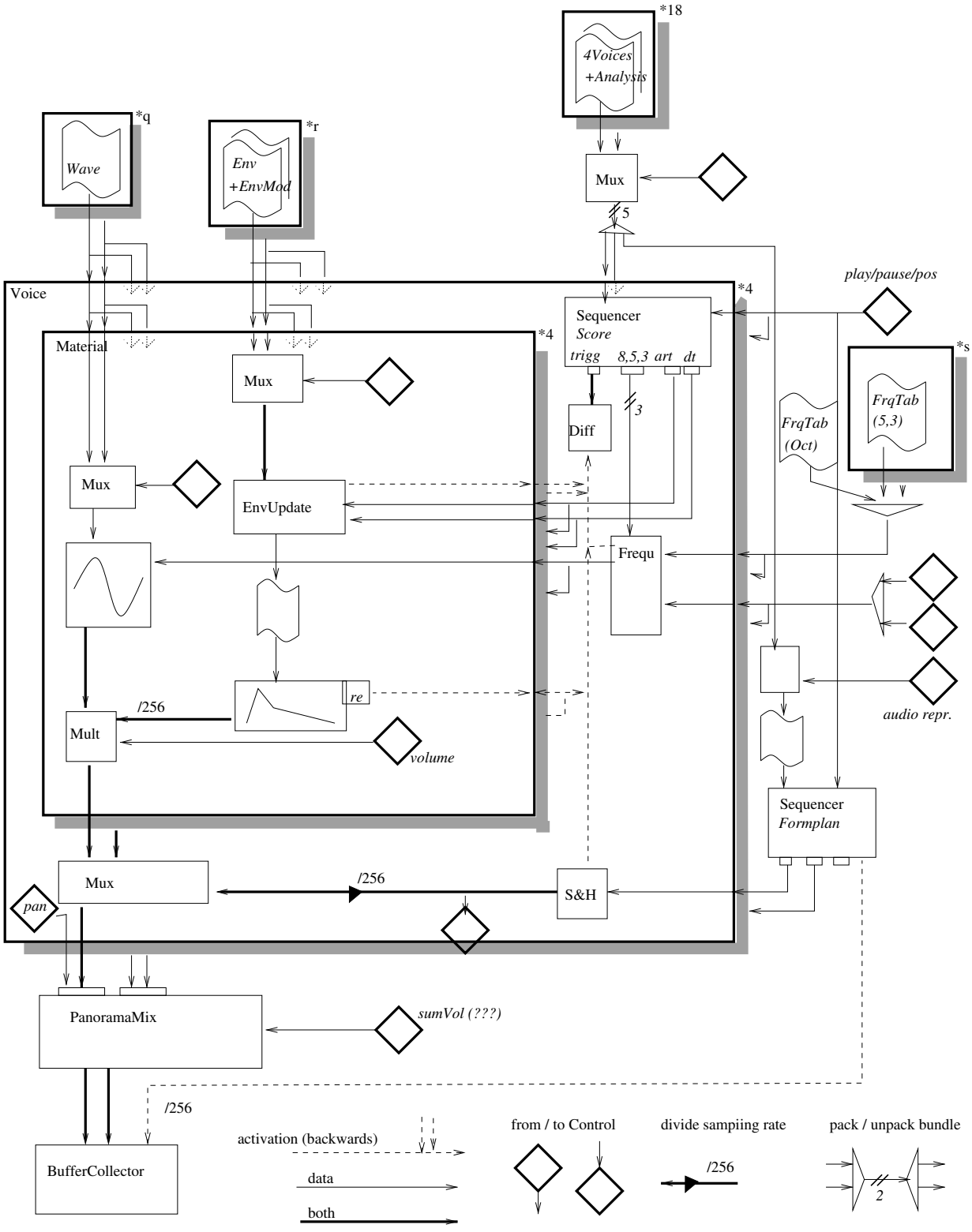


Figure 4. The sound synthesis network, realized with sig/adlib.

each voice score contain the pitch information encoded in octaves, fifth, and thirds. Last not least come the duration of the events.

The formplan data has its own format and goes to a dedicated sequencer which controls the multiplexing of the different material sounds per voice. The rightmost column shows the global tuning tables and the user's tuning controls which flow into the calculation of the technical frequency for each new event in each voice.

The bottom of the Figure shows how the four material layers are multiplexed into the output of the voice, controlled by the formplan sequences, and how the voices are mixed into the stereo output.

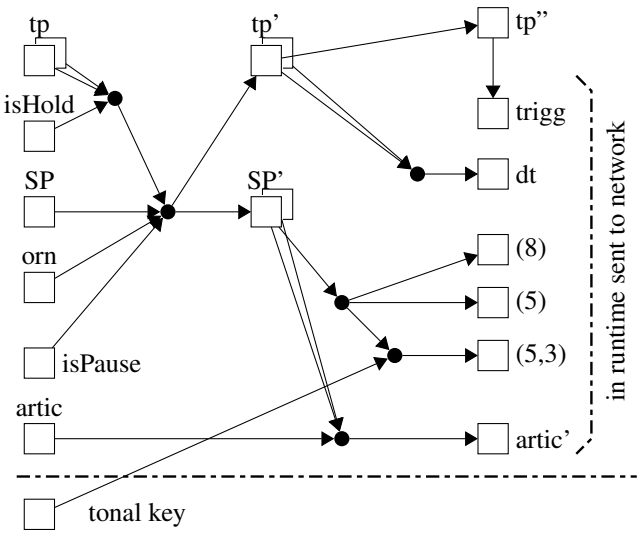


Figure 5. Translation from the tScore CWN format into sig/adlib sequencer data tables. (Staggered boxes stand for parameters of adjacent events.)

4.2 Transition from the semantic tScore score to the real-time sig/adlib score

The application consists of two parts: for realtime execution the symbolic data of the tScore world is translated into data for sig/adlib.

TScore-based data, including the `Score_cwn` meta-model as described above, contains complex, multi-faceted semantic relations. This is implemented with object-orientation and high-order functions. Contrarily, sig/adlib real-time data are fixed-length arrays of primitive values, optimized for realtime execution. Separation into these (or similar) two realms was caused by performance issues in the early days of real-time computer sound synthesis. Today it is still sensible for pragmatic reasons from software technology, applying the principles of information hiding and modularization [11]. And it serves as a cutting knife for exploring the domain theory, since the translation algorithm between these two spheres forces to make meanings and limitations on both sides explicit. For instance, the automated expansion of *ornaments* required exploring and formalizing of different rule systems. Figure 5 shows the data flow between both realms.

Critical points are (a) ornament resolution, which replaces one single input note by a sequence of multiple events; (b) the recognition of pitch repetitions, which is necessary for sharpening the articulation, while genuine articulation marks are rare in Bach's source text; (c) the translation of functional pitch information into combinations of interval coordinates.

The main difference to MIDI real-time execution is that *future information* is easily available for sound rendering, like the intended note duration and subsequent pitch repetition.

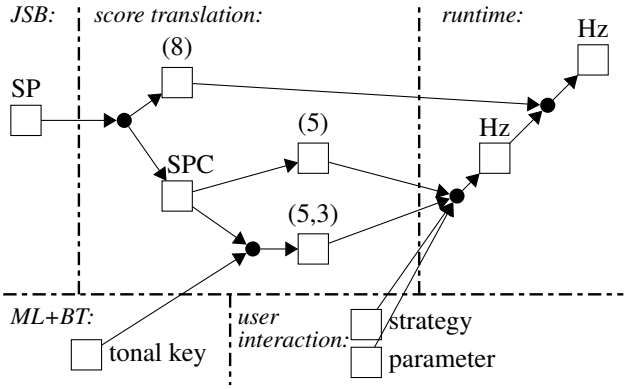


Figure 6. Tuning information flow. SP = spelled pitches. SPC = spelled pitch classes. (8), (5), (3,5) = interval coordinates.

4.3 Tuning

Figure 6 shows the data flow for a single event: The spelled pitch is translated into interval coordinates when the scores are translated.

The octave coordinate is used in any case. Other coordinates are calculated for the pitch class modulo octave. For SPs these classes are the *spelled pitch classes (SPCs)* [5].

The further coordinates specify multiples of fifths and thirds (= Euler coordinates) to realize pure tuning, and fifths only for all other tunings (equally tempered, mean tone, transposition invariant, etc.) Compared to SPs, these encodings allow much easier arithmetic operations like comparison, transposition, or tuning calculation.

Depending on the tuning strategy and its parameters, both selectable by the user interactively, even when playing, the effective frequencies are calculated.

The (5,3) encoding for the *pure tuning* interpretation of the SPCs requires additional information about the currently ruling *tonal key*. Currently this has been added manually by the authors, but could be supplied by an analysis algorithm in future versions of the software.

4.4 The Envelope Update Language (EUL)

For all envelope curves needed in a sound synthesis project, we offer the realization by a two-level language approach, see Table 4.

The lower level is a list of segment descriptors which control in realtime the changes of the output value of the envelope generator, segment by segment. Read as a table, each line contains an operation code and one or two numeric values, interpreted according to that opcode.

On a second level, the values used therein (d, f, v, and i in the Table) are given as an expression following the syntax definition *exp*, including realtime input parameters. In the current implementation, these are sampled once when a segment is started. (But this could change in a different future application context.) The expressions can use all score and all interactive parameters. The main aesthetic difference between realtime MIDI driven ADSRs and envelopes in a score-driven context is that the future length of the event is known and allows adaptive sound shaping.

dtSeg_	d		d gives the end point of this segment relative to its start.
dtEnv_	d		d gives the end point of this segment relative to the start of the complete envelope.
_fin		f	f shall be reached at the end of the segment; calculate the necessary increment accordingly.
_inc		i	i is the increment value which is added to the current value during the whole segment.
_inc		0	hold the current output value for the duration of the segment.
dtSeg_fin	0	v	jump to the value v and start next segment immediately.
inc_fin	i	f	add the increment i until final value f is reached.
end	–	–	keep output value; end processing.
ferm	–	–	keep output value; wait for external trigger to continue with next segment.

exp ::= *value* | *project* | *arith1* | *arith2*
| *log* | *if* | *comp* | *select*
value ::= `input`(*ident*) | `const`(*digits*)
project ::= `project` *exp exp exp exp exp*
// `project`$(a, b, c, d, e) = \frac{a-b}{c-b} * (e - d) + e$.
arith1 ::= (`not` | `neg` | `inv`) *exp*
arith2 ::= (`add` | `sub` | `mul` | `div` | `min` | `max`) *exp exp*
log ::= (`and` | `or` | `xor`) *exp exp*
if ::= `if` *exp exp exp*
comp ::= `equals` *exp exp exp*$^?$
// third parameter is the epsilon.
| `range` *exp exp exp*
// whether $a \leq c \leq b$
select ::= `select` *reference_to_envelope_table*

Table 4. Envelope Update Language (EUL): segment descriptor tables and expression language for the values.

The expression `select` is evaluated the same way, but operates on a meta-level: it selects from a multitude of table definitions. This is useful to switch to an alternative envelope form—for instance for extremely short or pianissimo events.

Figure 7 shows an example.

EUL is currently realized as a deeply embedded domain-specific language—the expression layer is interpreted by a stack machine whenever an event starts. This shall be replaced by a compiler into sig/adlib primitives, which is much faster.

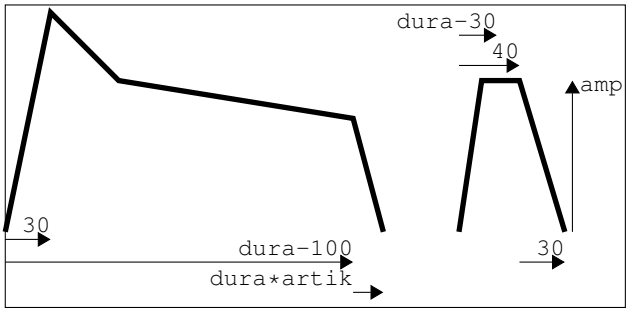
4.5 User Interactions Protocol

In sig/adlib, run-time user interaction are handled by `Post` objects. These can receive data from various sources, like GUIs, MIDI devices, sequencers, or programmed agents, and deliver it to synthesizers, protocol files, visual GUI feedback, etc. This architecture allows easy switching of the data source, even at runtime, and connecting it to a dynamic multitude of drains.

Every such Post object has a unique name and can write its realtime behavior to a protocol file. These files can be converted into the myKdF arrangement score format, see Figure 8 for an example and Table 5 for the syntax definitions.

This conversion is crucial (a) to make the protocol easy readable, and (b) to make it editable for modified replay.

(This part of the system is currently still under construction.)




```
if (range(0, 70, input(dura)),
    select(short), select(normal))
normal:
  dtSeg_fin const(30)       const(1.0)
  fin_Inc   input(amp)      const(0.005)
  dtEnv_inc sub(input(dura), const(100))
                            const(-0.004)
  dtSeg_fin mul(input(dura), input(artic))
                            const(0)
short:
  inc_fin   const(0.01)     input(amp)
  dtEnv_inc max(sub(input(dura),const(30)),
                const(40))
                            const(0)
  dtSeg_fin const(30)       const(0)
```

Figure 7. Two envelopes and their EUL source.

`VOX` *voiceName* *voiceData**
`VOX` *voiceName−matName* *matData**
voxData ::= `pan` (0..100) `/`$^?$
// Panorama position and interpolation select.
matData ::= (`sine` | `square` | `saw` | `triangle`)$^?$
$\&_{[;]}$ (`pizz` | `port` | `menoLeg` | `legato`)$^?$
$\&_{[;]}$ (0..100 `/`$^?$)$^?$
//Wave, envelope, and loudness with interpolation select.

Table 5. Syntax of the user protocol sub-language. The meta-operator $_\&_{[;]}_$ stands for permutation separated by `;`—for the others see Table 1,

5. SCORE PUBLISHING

The `Score_cwn` meta-model includes an automated translation into lilyPond source text.[12] Our print edition of *Die Kunst der Fuge* can be downloaded from `http://bandm.eu/downloads/kdf.pdf`. Page turns are optimized for piano playing; being in four staves and old keys, this is challenging but enjoyable.

```
PARS myInterpretation
  cp = 4
  audioRepr = byDirection visualRepr = byDirection
T        1              12   13      25            29     30       50
VOX sop  pan0/                                            pan100
    cp   sin;pizz;30/   49/  port    70                    50
    I    port;saw;60                                 square
VOX alt  pan20
    cp   sin;pizz;30
   (etc.)
```

Figure 8. A user interpretation protocol, converted to the corresponding tScore format.

6. STATISTICAL EVALUATION

Our models of *Die Kunst der Fuge* are also available for computer-assisted analysis. The following questions have addressed so far:

Tonal Material. Table 6 shows the SPCs found in every contrapunctus. The first contrapunctus obviously takes the role of laying the basis for the whole architecture: It contains twelve pitch classes, one for each key of the keyboard octave. Thus the first contrapunctus is the only which could be played in the traditional meantone temperament. The very last SPC is introduced not before m. 60. It completes the tonal spectrum by turning to the sub-dominant, which is a conventional signal for "we come to the end of the piece".

The maximum is reached with 17 SPCs in cp 4 (which was composed later [6]) and the triple fugue cp 8. This maximum set contains all white keys once and all black keys twice. The covered part of the circle of fifths ends immediately before the first SPC which notates a white key with an accidental: above c-flat and below e-sharp.

The very last contrapunctus, probably planned as a quadruple fugue, stays in its unfinished state below the maximum. This is a severe contrast to its harmonic complexity, which even includes far-out pseudo functions (German *Scheinfunktionen*) as in ms. 224–225. So perhaps here the maximum SPCs or perhaps even more would have appeared in a complete version.

False relations and diatonic splits. The English term *false relation* stands for two notes with the same generic pitch class and two different accidentals, appearing in two different voices. If these notes are adjacent then the German term is *Querstand*—when they are even overlapping it is considered a more pointed effect and called *diatonische Spaltung (diatonic split)*. One big motivation for constructing the data model was the statement of Ploeger [13] that there is no diatonic split in *Die Kunst der Fuge*. Knowing of one from playing, we wanted to search systematically and applied an algorithm to the data model. The rich results are out of scope of this article and will be published shortly.

Symmetric Chords, Grips, and Noteheads. According to Zacher [6] and our own results [14], in many works a composer applies a *second system* of specific and hidden rules, additionally to the conventional and obvious, to increase the complexity of the result and to make their necessary decisions easier. (Zacher: "[Zur] Diversifikation [= zur] Absicherung gegen das routinierte Einerlei der Fugenkomposition.") We have always assumed that *graphical symmetry* plays such a role in *Die Kunst der Fuge*. This could be proven by automated analysis.

	g♭	d♭	a♭	e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯	d♯	a♯	
1:				e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯			12
2:			a♭	e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯	d♯		14
3:		d♭	a♭	e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯	d♯	a♯	16
4:	g♭	d♭	a♭	e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯	d♯	a♯	17
5:				e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯	d♯		13
6:		d♭	a♭	e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯	d♯		15
7:			a♭	e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯			13
8:	g♭	d♭	a♭	e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯	d♯	a♯	17
9:			a♭	e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯	d♯		14
10:				e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯	d♯		13
11:	g♭	d♭	a♭	e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯	d♯		16
12:					h♭	f	c	g	d	a	e	h	f♯	c♯	g♯			11
13:					h♭	f	c	g	d	a	e	h	f♯	c♯	g♯			11
14:		d♭		e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯			13
15:				e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯	d♯		13
16:			a♭	e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯			13
17:			a♭	e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯			13
18:		d♭	a♭	e♭	h♭	f	c	g	d	a	e	h	f♯	c♯	g♯	d♯		15

Table 6. Spelled pitch classes contained in the 18 contrapunctus.

7. ACKNOWLEDGEMENT

Many thanks to Wolfgang Grieskamp, Sammamish, WA, for substantial financial support.

Many thanks to the anonymous reviewers for valuable hints for future work.

8. REFERENCES

[1] M. Lepper and B. Trancón, "tscore: Makes computers and humans talk about time," in *Proc. KEOD 2013, 5th Intl. Conf. on Knowledge Engineering and Ontology Development*, J. Filipe and J. Dietz, Eds., instincc. Portugal: scitePress, 2013, pp. 176–183, http://markuslepper.eu/papers/tscore2013.pdf[20231003].

[2] ——, "Translets —— parsing diagnosis in small dsls, with permutation combinator and epsilon productions," in *Tagungsband des 35ten Jahrestref-*

fens der GI-Fachgruppe "Programmiersprachen und Rechenkonzepte, ser. IFI Reports, vol. 482. University of Oslo, 2018, pp. 114–129, http://urn.nb.no/URN:NBN:no-65294[20230920].

[3] M. Lepper, *Example Instances of the TScore Projekt Infrastructure*, 2023, http://markuslepper.eu/sempart/tscoreInstances.html[20230920].

[4] J. S. Bach, *Die Kunst der Fuge BWV 1080*. Wiesbaden: Breitkopf und Härtel, 1924, *edt.* W. Graeser. [Online]. Available: https://s9.imslp.org/files/imglnks/usimg/1/1d/IMSLP469177-PMLP5843-Bach,_J.S._-_The_Art_of_the_Fugue.pdf

[5] J. Hentschel, F. C. Moss, A. McLeod, M. Neuwirth, and M. Rohrmeier, "Towards a Unified Model of Chords in Western Harmony," in *Music Encoding Conference Proceedings 2021*, S. Münnich and D. Rizo, Eds. Humanities Commons, 2022, pp. 143–149.

[6] G. Zacher, "Der geheime Text des Contrapunctus IV von Bach," *Beiträge zur Gregorianik*, vol. 13/14, 1992.

[7] M. Lepper, *de Linguis Musicam Notare*. Osnabrück: EpOS, 2021, https://www.epos.uni-osnabrueck.de/viewer/web/?id=150.

[8] M. Lepper and B. Trancón, "Critical semantic properties of music notation datasets," in *Proceedings of Music Encoding Conference 2024*, Denton, TX, USA, 2024, https://works.hcommons.org/records/ppg74-g6223.

[9] A. Danial, "cloc – count lines of code (v1.92)," Tech. Rep., 2021, https://github.com/AlDanial/cloc [20231003]. [Online]. Available: https://doi.org/10.5281/zenodo.5760077

[10] B. Trancón y Widemann and M. Lepper, "Sig adlib — mostly compositional clocked synchronous data-flow programming in java," in *Tagungsband des 35ten Jahrestreffens der GI-Fachgruppe Programmiersprachen und Rechenkonzepte*, ser. IFI Reports, vol. 482. University of Oslo, 2018, pp. 31–48. [Online]. Available: http://urn.nb.no/URN:NBN:no-65294

[11] D. L. Parnas, "On the criteria to be used in decomposing systems into modules," *Commun. ACM*, vol. 15, no. 12, p. 1053–1058, Dec. 1972. [Online]. Available: https://doi.org/10.1145/361598.361623

[12] H. W. Nienhuys and J. Nieuwenhuizen, "Lilypond, a system for automated music engraving," in *Proceedings of the XIV Colloquium on Musical Informatics*, Firenze, 2003, pp. 167–172.

[13] R. Ploeger, *Studien zur systematischen Musiktheorie*. Lilienthal, Bremen: Eres, 1990.

[14] M. Lepper, "Das Zweite System," *senza tempo (blog)*, 2015, http://senzatempo.de/ston2015081400.html.

A. APPENDIX: CAVEATS FOUND WHEN DEFINING THE FORMPLAN DATA

Writing a graphic representation of the form of a particular fugue is always a *creative process of interpretation.* This is understood from an abstract philosophical point of view, but it becomes strikingly obvious as soon as this formplan is notated in a concrete data format like tScore. Especially some rough edges indicate fundamental aesthetic problems and properties:

A.1 Recto versus inverso in cp 1–cp 4

Taking cp 1 to cp 4 each on their own, they only have one theme each which appears in only one direction. This thus can be called recto and no modifier must be appended to any theme code.

But in the context of *Die Kunst der Fuge* as a whole, these themes are the main theme appearing as recto in the former two and as inverso in the latter two of these contrapunctus. Therefore it could also be sensible to mark *all* thematic entries in cp 3 and cp 4 by the modifier `u`. We did not do so, because it would not add any flexibility to the software for its users. But it must be taken into account when e.g. applying analyses to the data model as a whole.

In cp 5 to cp 7 both forms are combined, so that the placement of the modifier `u` is out of any doubt.

A.2 Recto versus inverso in cp 8 and cp 11

Even more complicated is the relationship of cp 8 and cp 11. Cp 8 introduces two new themes, which it it later combines with the main theme of the work. This happens in form of a *metalepsis*—several things are first presented as unrelated, and only later, in retrospection, the combination becomes logical, even unavoidable.[6]

The themes II and III both appear only in one direction, which most naturally is received as their recto. But later they are combined with (a rhythmic variant of) the main theme in its inverso direction, when compared to its initial appearance in the overall work. We decided to use the undecorated codes `II` and `III` for the directions in which the new themes appear here for the first time, and the decorated `Iu`, which reflects the larger context.

In cp 11 now the same three themes appear again, but inverted: It starts with theme I in its recto form, followed by II and III inverted, now labelled as `I`, `IIu`, and `IIIu`. Later in this movement all themes appear in *both* directions: `Iu`, `II`, and `III` appear additionally. This confirms that our choices for the labels in cp 8 where sensible. Alternatively, all three themes in cp 8 can be labelled with `u`, but this does not correspond to the psychological effect: We do not expect an inversion to sound earlier than the original.

A.3 Automatic detection of incomplete theme entry

The implemented formplan meta-model allows to specify a numeric limit, by a parameter setting like

```
minimumCompletum = 4 1/8
```

Every theme entry shorter than this is automatically tagged as incomplete. This does work well in most contrapunctus,

but interestingly not in cp 7: Some entries have a shortened *initial* note, i.e., they start at a weaker metric time point than usual, see tenor ms. 23, soprano ms. 38 and 42, alto ms. 51, bass ms. 47.

Similar, there is an acceleration for a short subsection in the entry alto ms. 14.

In all these cases the overall length is shortened, but the entry cannot be called incomplete, which is a psychological term related to a premature *ending*.

A.4 Problematic limits of theme entries

All themes in *Die Kunst der Fuge* are clearly delimited, except the theme III in cp 8 and cp 11, see Figure 9. This is a sequence (in the musical sense of the word: repetition with transposition) which can break off after different numbers of repetitions. The fact that it is really a full-fledged theme and must be honored as such is doubtlessly clarified by the facts that (a) it appears in a fixed combination with both other themes (German: *kontrapunktischer Verband*), that (b) is subject to a maximal stretta in cp 11 ms. 170, and that (c) it is the only structure in the whole work which contains the very significant pitch repetition. But when exactly do the entries of this theme end? And when do they start?

Its motif kernel has three repetition notes and one singleton. As exposed in cp 8 alto ms. 40, it jumps down a third from the repetitions to the singleton, and goes up a second to the next repetition, see Figure 9. In the formplan score this is labelled as `III`.

With its re-appearance in cp 11, these two interval directions are maintained, but the sizes are exchanged, see Figure 9, tenor ms. 90: the melody goes down a second and up a third. Thus the overall direction of the sequence is inverted, but not the directions internal to the motif. Not before ms. 130 soprano a verbatim inversion appears, which jumps the third upward after the repetition. We decided to mark both versions which go globally upward as `IIIu`, but of course this decision is subjective. (Perhaps the set of possible decorations in the formplan meta-model should be enhanced by some freely useable variant indication?)

In all of its versions, finding the boundaries of the entries of this theme is problematic. In the formplan scores, we set the end points by melodic criteria, corresponding to their role in the application, namely to define the timepoints when to switch the sound register. The end points of theme III in cp 18 have similar problems, see for instance m. 161 pp.

Also the starting points of the entries of theme III cp 11 are ambiguous, but less problematic to define: We choose points shortly before the first significant repetition, which corresponds to the listener's recognition.

A.5 Problematic automated offer of representation modes

The representation modes offered to the user when selecting a movement, as shown in Table 2, are derived from the entries in the formplan data.

Cp 18 has one single incomplete entry (also called *false entry*, German: *Scheineinsatz*) we marked as diminuition
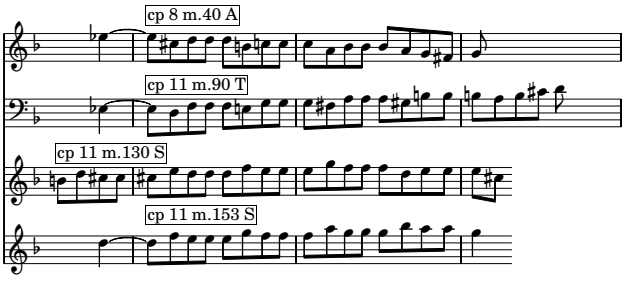


Figure 9. Ambiguous definitions of theme III and IIIu.
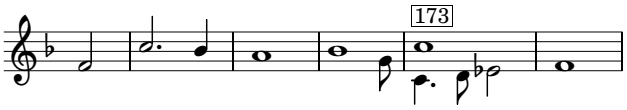


Figure 10. Diminuition entry in cp 18 m. 173 "IIu/2>" or "Iu>".

of theme II, namely ms. 173 alto, see Figure 10. As a consequence, the implementation offers the theme representation mode byTempo as appropriate. But when the user selects this for visual representation, they immediately see that a different tempo appears only once, and thus this mode is hardly useful for sound control. Since this entry is incomplete, it can also be heard as one of theme I, the overall main theme—it functions as a kind of proof object. So it should not be left completely unmarked.

A.6 Borderline cases between theme entry and derived counter-point material

A fundamental problem, often arising in the psychological analysis of listening and in didactic presentation, is drawing a line between regular theme entries and contrapuntal material extracted therefrom.

From a formal point of view, every fragmentation (German *Abspaltung*) from the *start* of a theme can also be counted as an incomplete entry. Whether this is appropriate, depends on the psychological situation of the recipient. There are famous moments where this dichotomy is played with, see Beethoven, Eroica, Finale, ms. 143–145 and Bruckner, Fifth Symphony, Finale, ms. 258–262.

Here we have a different situation in cp 5, ms. 53–56 and ms. 65–69, where the complete setting *as a whole* can be heard either as thematic *stretta* or as counterpoint material derived from (a head fragment of) the main theme.

Similar cases shows cp 8 with theme II at m. 89 and m. 114, and with theme III at m. 74 (= first solo theme entry) versus m. 77 (= theme entry or mere counterpoint?).

In cp 11 we find multiple such situations with respect to theme III, see ms. 97–99, etc. Again our formplan data aims for satisfying audio results.

The shortest possible fragment which can be heard as an incomplete entry is the sequence C3–F2 in cp 18 m. 168.

A.7 Overlapping Abspaltung

The formalization of analysis and mark-up also shows the limitations of the chosen meta-model. The crucial ms. 14–
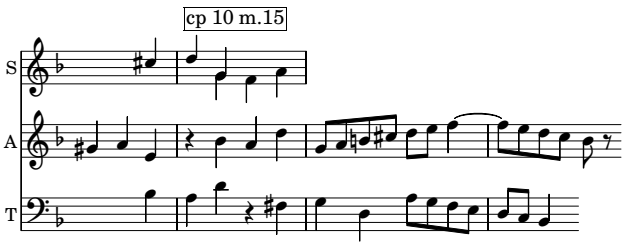


Figure 11. Overlapping incomplete entries in voice S.

16 in cp 10 show a stretta between recto and inverso in alto and tenor, together with an incomplete entry in the soprano, see Figure 11. But indeed these few notes can even be heard as *two* incomplete entries which *overlap*: the third note G4 is the last note of the first and the first of the second entry. Such an overlap is not foreseen in the metamodel. Future applications to models outside *Die Kunst der Fuge* perhaps require to extend it.

A.8 Insignificance of the dux/comes form

While the distinction between dux and comes form is highly relevant for the architecture of many contrapunctus, there are others where it can be neglected:

In cp 8 there the second entry of the main subject (bass m. 99) *programmatically* shows the comes form. We did not mark it as such in the formplan because it is the only one.

In cp 11 the main theme appears nearly always in the dux form. The comes form nevertheless is present, so the entries are coded completely in the formplan data. Changing the form only changes the interval of the very first anacrusis (in contrast to the standard process as applied in the very first comes).

This also holds for cp 16, and also in only few entries: S ms. 9, A ms. 48, B ms. 63.

In cp 17 the figured version of the theme after the first development group (= all entries after ms. 21) have all the same constant form. Our formplan score contains the qualifiers `D` (7 times) and `C` (only twice), which is correct in the strict formal sense, but aesthetically questionable.

VIPPSA: A WEB-BASED TOOL FOR VIDEO SCORE SYNCHRONIZATION AND ANNOTATION

Xavier Davenport
University of Illinois
xavierd2@illinois.edu

Craig Vear
University of Nottingham
craig.vear@nottingham.ac.uk

Christoph Guttandin
Media Codings
chrisguttandin@media-codings.com

ABSTRACT

Based on feedback from musicians who have rehearsed and performed digital scores, in this paper we present a novel technology for Video Playback Position Synchronization and Annotation (ViPPSA). We attempt to ameliorate some of the issues described by performers when rehearsing and interpreting video scores by providing a convenient platform for synchronizing and annotating pre-rendered scrolling-scores. We describe the annotation and synchronization features of ViPPSA and how they may be utilized in rehearsal and performance settings. We also briefly present a new musical work designed to take advantage of the unique features of this software, as well as possible future enhancements informed by early feedback from performers rehearsing using this technology.

1. INTRODUCTION AND PROBLEM STATEMENT

1.1 Digital Scores

The Digital Score is a 5-year, European Research Council-funded research project investigating the transformation of the music score through digital and computational technologies. It has been running since 2021, and currently in the final stage of evaluation and synthesis. The principle investigator is Prof. Dr. Craig Vear (University of Nottingham).

Vear defines a digital score as *a communications interface of musical ideas between musicians utilizing the creative potential of digital technology.* [1, 2] There are many types of digital scores that cover a broad range of approaches and technologies, but they all have one thing in common: they *communicate musical ideas using the creative potential of technology*. In this paper, we focus on a small subset of digital scores commonly referred to as *scrolling scores* or *video scores* where the score is pre-rendered as a video, then presented to the performer using a video-player.

1.2 Technical Difficulties

One of the findings that the Digital Score research project highlights is how performers have encountered difficulties when attempting to synchronize individual video score parts in rehearsal and performance. One such piece is the piano trio *Traveling Vagabond* by Davenport where synchronization of unique scrolling-score parts was a challenge that was never overcome before the premiere. Although a video score is often successful in conveying information when all the performers are able to look at a single screen, ensembles involving the piano often traditionally have performers facing radically different directions on stage. For the premiere performance of *Traveling Vagabond*, the strings players looked at an iPad, while the pianist looked at a Windows laptop. It was discovered in rehearsal that these different devices had radically different response times between clicking the play-button and the video timeline beginning to move. To make matters worse, the response times between clicking and playing were also inconsistent, likely depending on the background tasks each device was performing.

Several scrolling scores were commissioned and presented as part of the 2022 TENOR conference in Marseille, France. Musicians were open and willing to explore these new works, but offered critical feedback concerning the challenges around rehearsing and performing the works. [3] The performers specifically recommended that there should be functional rehearsal markers which enable rehearsal of individual sections in isolation.

More digital scores were presented at a DigiScore concert as part of *Encuentro Montrealia* in 2025. Performers at this event highlighted issues with technical coordination of the scores. [4] They noted that synchronization issues, multiple screen displays, and hardware constraints directly impacted the flow and coherence of their performance, though they attempted to alleviate some of these issues with the on-site sound engineers.

To address these issues reported by performers of digital works, we have developed a web-based environment for annotating and synchronizing video scores. The technology reduces the hardware barrier to entry by being accessible in any web-browser. Though it is currently optimized to work over a local area network, pieces can still be rehearsed and performed if necessary by accessing a publicly available website. As of submitting this paper, the annotation capabilities of ViPPSA can be accessed at https://vippsa2025.web.illinois.edu/. In the following, we describe the ways these issues of synchronization and rehearsability have been addressed by others before describing the unique features of ViPPSA, then finally present some possible future enhancements to the software based on feedback from performers using the web application.

Copyright: ©2025 Xavier Davenport et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

2. RELATED RESEARCH

2.1 Score Synchronization

The idea of synchronizing distinct animated part-scores between devices is not new – the Decibel Score Player does exactly that as well as providing an environment for bespoke score creation. [5] While the Decibel Score Player provides an immense amount of flexibility for the composer, the required collection of iPads may be prohibitively expensive for an ensemble to adopt. The creators of the Decibel Score Player cite synchronization issues due to "different operating systems, networking protocols and personal settings" as the motivation for restricting the software to only iPads. [6] Other frameworks for cross-device communication exist to fulfil specific compositional needs such as INScore and DRAWSOCKET which rely on OSC messages for synchronization, as well as browser-based systems for distributed notation like PITL, A&M, and John. [7, 8, 9] SmartVox is a web app very similar in concept to ViPPSA in that it synchronizes mp4 video files between devices using socket.io. [10] Another web-based app was developed a few years after SmartVox in 2021 called Autoconductor which can handle multi-channel audio as well as synchronizing scores, though the method of synchronization used was susceptible to fluctuations in the WiFi network. [11] The primary differences between these two synchronization web apps and our new software are the underlying algorithm used for synchronization between devices and the ability to annotate the score.

With the exception of SmartVox, the software listed in the previous paragraph was designed with interactivity in mind. SmartVox and our new software ViPPSA are concerned with a smaller subset of digital scores where the score has been pre-rendered into one or more videos, several of which can be found at www.animatednotation.com. We have opted to focus on scrolling video scores because of their portability and reliability – it is significantly easier for performers to put together a performance when they do not need to manage a bespoke system for algorithmic score generation.

2.2 The Timing Object

The video elements in the browser are synchronized across multiple devices with the TimingObject as specified by the Multi-Device Timing Community Group. [12, 13] The final version of the Timing Object specification was published December 3rd, 2024, and is designed to integrate with the Web Audio API, among other Web APIs. [14] We use an implementation of the TimingObject provided by Suenc which allows for precise timing between devices on a local area network without the need to reference a clock hosted on an external server. A demonstration of this technology is hosted at suenc.io. The TimingObject is essentially a vector (called the TimingStateVector) with position, velocity, and acceleration imformation at a certain point in time. It allows for precise and reliable synchronization between any number of devices. [15] This contrasts with what is used in SmartVox, which is dependent only on position with an additional algorithm layered on top to correct for timing drift between devices. [16] Correcting the playback position when it has sufficiently drifted from the desired timestamp can lead to an irregular playback velocity between devices. The TimingObject implementation constantly verifies the timing offset between devices and then adjusts the timing information of the synchronized TimingStateVector transparently.

3. DESIGNING VIPPSA

There are four significant difficulties performers have expressed when performing using video scores: [17] (1) It is challenging to rehearse small sections of animated scores in a repeatable way. (2) Scores should be flexible enough so that they can be hidden from the audience. Many scores need to be seen by the entire ensemble on a single screen, so they are often projected somewhere the audience can also see the score. (3) Bespoke technology is unstable. Latency, poor internet signal, unstable connectivity, and more can cause unnecessary stress to performers. (4) The performers should be able to engage with the piece before the group rehearsal. ViPPSA is intended to ameliorate each of these issues.

When performers go to the ViPPSA website, they will first need to press the "Load Video" button and select the video they wish to play on their device. So long as the *synchronize* checkbox is selected, any manipulation of the video playback position or playback rate is synchronized between all connected clients. If a performer wishes to navigate throughout the score with influencing or receiving influence from others, they may deselect the *synchronize* checkbox.

3.1 Lower Controls

Many elements in the video area will look familiar to anyone who frequently consumes content on YouTube or other streaming services, as shown in *Figure 1*. On the bottom left of the screen is the play/pause button as well as a mute button with volume slider. On the bottom right of the screen is a fullscreen button, as well as a number indicating the current playback rate, for example 1x, 0.2x, 0.8x, etc. All the other buttons across the bottom of the screen are tools used for annotating the score. First there are two horizontal sliders which appear when the user hovers over the corresponding icon, the first one used for pen-size and the second one used for pen-opacity. After that there are five buttons, the first of which allows the user to select a color for their next annotation. After that are four tools: the pointer, the text tool, the pen tool, and the eraser. The pointer is the default tool and does not allow the user to make annotations. After selecting the text tool, the user may click or tap somewhere on the video to make a text annotation. With the pen tool selected, the user can drag their mouse or finger across the screen to draw an annotation. Finally, the eraser can be used to delete both text and pen annotations.

After drawing an annotation using the pen tool and releasing the click or finger from the screen, a dialog appears asking for the annotation start time and end time. In the in-



Figure 1. Screenshot of the lower video controls container. From left to right, there is a play/pause button, a mute/volume button, the current video time, the video duration, a pen/text size slider, an annotation opacity slider, a color picker, a default pointer button, a text-tool button, a pen-tool button, an eraser button, a playback-rate button, and a fullscreen button.



Figure 2. Screenshot of the upper video controls container. From left to right, there is a synchronization checkbox, a selectable list of warp points, and a settings button. Selecting the synchronization check box allows the user to both receive timeline updates from others as well as influence the timeline of other users. Selecting an item in the list of warp points immediately moves the timeline to the corresponding timestamp.

terim between the start and end times, the annotation will appear on the video, but will not otherwise. The same is true of the text tool, however there in an additional input field for the text the user wants to appear.

3.2 Upper Controls

There is an additional option in the text annotation dialog asking whether the annotation should be added as a warp point. This feature was added in response to performers wishing it was easier to rehearse specific parts in a piece of music; adding a warp point adds the text annotation to a list of timestamps accessible in the "Warp to Annotation" dropdown in the center of the upper video controls container shown in *Figure 2*. When one of the timestamps is selected, the video warps all synchronized videos to that position. At any time, the user may save all the annotations they made. The annotations are saved in a plain-text json file and can be loaded again next time the website is accessed.

The feature-set detailed above streamlines the rehearsal process and aims to ease the four difficulties described by performers: (1) Small sections of a piece can be more easily rehearsed by setting warp points using text annotations. (2) Since scores can be so easily and precisely synchronized between devices, each performer can look at their own device and there is no need to show the score on a large screen. (3) This technology is using time-tested browser-based technology. Even if performers are not on a dedicated local area network, the TimingObject underlying ViPPSA allows for consistent synchronization between many devices over the internet, even when the WiFi is somewhat unreliable. (4) ViPPSA is hosted on a website accessible to performers at any time. Either as an individual or as an ensemble, performers can rehearse and annotate a piece without the composer or director needing to be in the room.

4. USAGE IN REHEARSAL

In the following, we outline a hypothetical large ensemble rehearsal using this technology. We assume that everyone has access to a personal device with a screen sufficiently large to view the score.

4.1 The First Run

Before rehearsal, each performer must download a video file of their part. At the beginning of the rehearsal, the performers should navigate to the ViPPSA webpage, click the Load Video button, and select their score. At this point, they can freely navigate and annotate the score on their own and should wait to select the "Synchronize" checkbox until everyone is ready to navigate the score together.

When the ensemble decides to begin playing the piece as a group, each member should ensure that the "Synchronize" checkbox is selected. Once everyone is synchronized, any member of the ensemble can navigate the video and influence the video playback position of every other performer's score.

4.2 Annotations

During a pause in rehearsal, performers may wish to desynchronize from the ensemble to navigate the score and add annotations. For instance, a performer could set the pen tool to be yellow with a large pen size and low opacity to highlight an easy-to-miss moment in the score. They might also use the text tool to add supplemental information to the score such as expressive text or a technical detail.

Using the text-tool, users can also add warp points to the score as a way to quickly revisit a specific moment in the score. Users can individually de-synchronize and add personal rehearsal markers. Alternatively, the ensemble could synchronize before individuals add rehearsal markers to ensure the markers are synchronized. There is currently not a simple way to transfer rehearsal markers between performers within ViPPSA, so every performer will need to add the markers to their own score. In either case, only one performer needs to select a warp point for all synchronized users to be influenced.

4.3 Example Score: *Stormbrush*

As a way to test the capabilities of ViPPSA, Davenport composed a piece for five instruments where each performer receives a unique video score. In the piece, each performer is playing at a different tempo by referencing what Kocher calls a *virtual conductor* on the screen. [18] This polytempic texture requires a high level of temporal synchronization to achieve the desired effect. In certain moments of the piece, it is also suggested that the performer annotate reminders for themselves on how to interpret the score depending on their instrument, as shown in one of the parts in *Figure 3*. Details about this piece can be found in another submission to this TENOR conference titled SONIC WORK SUBMISSION: STORMBRUSH.

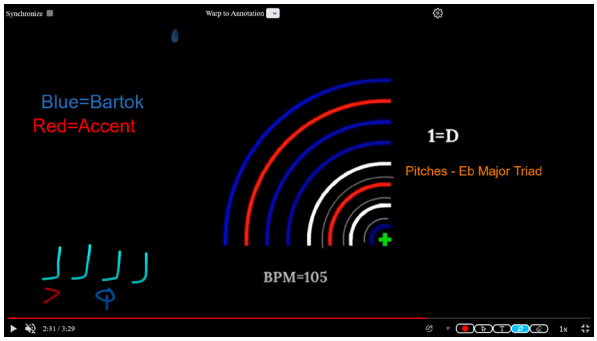


Figure 3. Screenshot of Davenport's score *Stormbrush* with a few annotations added. A user has added colored text as a reminder for what each color means on the screen, as well as a reminder of what pitch set they wish to use and a simple outline of a recurring rhythm in this section of the piece.

5. FUTURE ENHANCEMENTS

The goal in creating this software is to make digital scores more approachable and rehearsable for musicians. The future direction of ViPPSA design must prioritize an intuitive user interface while balancing a set of flexible features. Despite the need for flexibility, ViPPSA should not aim to be a comprehensive drawing program or a critical annotation tool. Instead, development should be driven by feedback from performers who desire a more efficient means of achieving their musical goals.

Based on early feedback from performers and colleagues, there are several features that could be reasonably added to ViPPSA which can be divided into three broad categories: synchronization, annotation, and distribution.

5.1 Distribution Features

Right now, the score is treated much as a paper score would be in an orchestral setting; the score must be "passed out" before or during the first rehearsal such that each performer has their own score loaded into their personal device. Depending on the size of the video file, this could be done via email, Box, Google Drive, or any number of other file sharing services. Some users expressed interest in a file-sharing system being integrated into ViPPSA such that videos can be downloaded and shared between users. This would make updating and sending new versions of a score much simpler for a composer.

Rather than relying on a locally stored video for rehearsal, some users express interest in playing videos hosted on YouTube or Vimeo embedded in ViPPSA while still being able to use the annotation tools. This addition to ViPPSA is not technically challenging, but requires duplicating a significant amount of code to handle both the HTML5 video element used for local video playback as well as the iframe element for embedding external videos. This feature would allow for the convenient rehearsal of a number of scrolling scores which are already posted on public platforms.

Performers who regularly work in larger ensembles saw the potential for ViPPSA to act as a system for the distribution of not only scores but also annotations. They conceived a scenario where an ensemble director may wish to add some general notes or rehearsal marks to everyone's score simultaneously, taking advantage of the digital interconnectivity provided by ViPPSA.

5.2 Synchronization Features

In addition to synchronizing videos between devices, the TimingObject can be mapped to other communication standards in audio. For example, timing object could be used to feed a string of OSC messages with timing information to another piece of software on one or more machines. This also provides a simple way to connect ViPPSA to a system for audio generation and other real-time elements.

Users also expressed an interest in even more precise temporal navigation in the score. Users can navigate by clicking on the timeline at the bottom of the video above the controls or by incrementing time using the left and right arrow keys, but typing in a specific time provides a much desired level of precision not currently in ViPPSA.

5.3 Annotation Features

The process of utilizing the annotation tools is designed to be as similar as possible to how a performer might annotate a paper score; a user draws something with a pen tool, and they could erase it later if desired. The annotation tools do not use any of the advantages of digital systems found in modern drawing programs. Some simple comforts that should be added to ViPPSA are functions like copy, cut, paste, undo, and redo. There is also presently no way to move an annotation after creating it; it must first be erased, then drawn again.

One more advanced computer drawing feature worth adding to ViPPSA is the idea of layering. Rather than every annotation being drawn to the same canvas, annotating on different layers would allow users to group annotations and choose whether to hide or lock them in place.

Perhaps the most desired feature for performers is the ability for annotations to move along with the video score. Before recent developments in AI technology, this would have been a feature exceptionally tedious to use: it would require the user to manually add key-frames to a video in order to attach an annotation to some feature on screen as it moves about. Now however, tools like the motion tracking video editors from Kapwing, Flexclip, and several other websites exist and are continuously improving. It is possible that this AI technology could be used to identify and track score elements in future versions of ViPPSA.

6. CONCLUSION

ViPPSA addresses challenges expressed by performers by allowing precise playback position synchronization between devices and for annotation of the score. Utilizing the timing-object standard allows latencies of less than a millisecond between an arbitrary number of web browsers. Our new annotation tool is simple, but gives room for personalized

note-taking within a video score. By prioritizing a user-focused design, we hope to broaden the appeal and accessibility of digital scores.

Acknowledgments

The Digital Score project (*DigiScore*) is funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. ERC-2020-COG – 101002086).

7. REFERENCES

[1] C. Vear, *The Digital Score: Musicianship, Creativity and Innovation*. New York: Routledge, 2019.

[2] "The digital score," 2025. [Online]. Available: https://digiscore.github.io/

[3] "Digiscore tenor22 commissions," 2022. [Online]. Available: https://digiscore.github.io/2022-10-05-DigiScore_TENOR22_commissions/

[4] "Digiscore concert as part of encuentro montrealia, mexico-canada," 2025. [Online]. Available: https://digiscore.github.io/2025-06-05-CMMAS_concert/

[5] A. Wyatt and C. Hope, "The decibel scoreplayer – learning from and for usability," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'2022*, V. Tiffon, J. Bell, and C. de Paiva Santana, Eds. Marseille, France: PRISM Laboratory, 2022, pp. 143–148.

[6] C. Hope, L. Vickery, A. Wyatt, and S. James, "The decibel scoreplayer - a digital tool for reading graphic notation," in *Proceedings of the First International Conference on Technologies for Music Notation and Representation – TENOR'15*, M. Battier, J. Bresson, P. Couprie, C. Davy-Rigaux, D. Fober, Y. Geslin, H. Genevois, F. Picard, and A. Tacaille, Eds., Paris, France, 2015, pp. 58–69.

[7] J. Bell, "Distributed notation in the browser, an overview," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'20/21*, R. Gottfried, G. Hajdu, J. Sello, A. Anatrini, and J. MacCallum, Eds. Hamburg, Germany: Hamburg University for Music and Theater, 2021, pp. 251–259.

[8] A. Pirchner, "Ergodic and emergent qualities of real-time scores. anna & marie and gamified audiovisual compositions," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'20/21*, R. Gottfried, G. Hajdu, J. Sello, A. Anatrini, and J. MacCallum, Eds. Hamburg, Germany: Hamburg University for Music and Theater, 2020, pp. 189–197.

[9] V. Goudard, "John, the semi-conductor: A tool for comprovisation," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'18*, S. Bhagwati and J. Bresson, Eds. Montreal, Canada: Concordia University, 2018, pp. 43–49.

[10] J. Bell and B. Matuszewski, "Smartvox. a web-based distributed media player as notation tool for choral practices," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'17*, H. L. Palma, M. Solomon, E. Tucci, and C. Lage, Eds. A Coru na, Spain: Universidade da Coru na, 2017, pp. 99–104.

[11] L. Felten and C. Klinkenberg, "Autoconductor: Synchronizing graphic scores, multi-channel sound and video files," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'20/21*, R. Gottfried, G. Hajdu, J. Sello, A. Anatrini, and J. MacCallum, Eds. Hamburg, Germany: Hamburg University for Music and Theater, 2021, pp. 265–269.

[12] "Multi-device timing community group," 2025. [Online]. Available: https://www.w3.org/community/webtiming/

[13] *Data-independent sequencing with the timing object: a JavaScript sequencer for single-device and multi-device web media. In Proceedings of the 7th International Conference on Multimedia Systems (MMSys '16)*, 2016, pre SFI. [Online]. Available: https://www.w3.org/community/webtiming/files/2016/05/mmsys2016slides.pdf

[14] "Timing object," 2024. [Online]. Available: https://www.w3.org/community/reports/webtiming/CG-FINAL-timingobject-20241203/

[15] C. Guttandin, "The timing object - a pacemaker for the web," 2018. [Online]. Available: https://media-codings.com/articles/the-timing-object-a-pacemaker-for-the-web

[16] J. Bell, "Audiovisual scores and parts synchronized over the web," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'18*, S. Bhagwati and J. Bresson, Eds. Montreal, Canada: Concordia University, 2018, pp. 17–23.

[17] C. Vear and S. Moroz, "The digital score: a review of research spanning 2022-23," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'2023*, A. P. D. Ritis, V. Zappi, J. V. Buskirk, and J. Mallia, Eds. Boston, Massachusetts, USA: Northeastern University, 2023, pp. 154 – 160.

[18] P. Kocher, "Polytempo network: A system for technology-assisted conducting," in *Proceedings ICMCISMCI2014*, 09 2014.

Artificial Authenticity - Real Authority: 你們是蟲子] (You are bugs), an AI-assisted playbite on Luigi Nono's ...*sofferte onde serene...* for piano and tape

Pavlos Antoniadis
University of Ioannina & IRCAM
p.antoniadis@uoi.gr
antoniadis@ircam.fr

ABSTRACT

On the occasion of Luigi Nono's birth centenary, I respond to his classic work for piano and tape, "..sofferte onde serene..", with a new work by the title 你們是蟲子 (You are bugs). This work extends Nono's techniques through live AI tools and interactive multimedia. It reflects on problems of authorship, authenticity and authority in musical AI, addressing the disturbing potential of human inferiority – the fact that we all are potentially "bugs" (with its multiple connotations), and we might have to accept a lower status in a new (or not so new) hierarchy. To do so, the work assumes an AI, as if alien, invasion of sorts, inspired by the "dark forest hypothesis", term coined by the Chinese sci-fi author Liu Cixin as a response to the Fermi paradox. The musical score consists of a fixed video, featuring borrowed cinematic materials of authoritarian figures, which are conducting, instructing, ordering, insulting and abusing the performer; augmented interactive tablatures; music notation and video synths. The score is projected in real-time and drives a combination of improvised, comprovised and fixed musical materials. These include the author's previous recordings of works by Luigi Nono, Iannis Xenakis and Maurice Ravel, which are remixed through a combination of SOMAX2 and GesTCom and are live controlled through inertial sensors and an MPE controller. Thus, both human and machine are trained and constrained by each other, and sparks of resistance to the machine are not missing.

1. INTRODUCTION

The advent of democratized AI tools based on generative architectures (GAN, RNN, CNN, Transformer, VAE and others) triggers new questions about the relation between authenticity and creativity [1], [2]. So do shallow learning approaches to generative AI, including methods such as probabilistic models, rule-based systems, Markov chains, genetic algorithms, and classical statistical techniques [3]. These methods generate content based on predefined rules, statistical relationships, or evolutionary processes, rather than the neural network architectures characteristic of deep learning.

But how novel are these questions? In the domain of musical performance and composition, the relation between authenticity and creativity has always been tensed, throughout both the history of contemporary music performance practice for traditional instruments, and the history of computer music performance practice for fixed or real-time interactive media. The emergent questions today are of a completely different order, but they still relate to the original aporias in often surprising ways to be explored in this paper.

Following up from this, I start by reviewing a classic problematic case, Luigi Nono's *..sofferte onde serene..*[4] for piano and tape, whose authentic performance by the standards of the so-called "high modernist model of performance practice" has been rendered impossible, due to parameters both endogenous and exogenous to the work. I subsequently address the relation between authenticity and creativity by reviewing three concepts, including Brian Massumi's notion of play as metacommunication - the conceptual background for the notion of "playbite" that features in the title - as well as my notions of Artificial Authenticity and Real Authority. I review the notion of Artificial Authenticity to address the creative and (auto)poietic aspects of any reconstruction of this type, before switching registers to link Artificial Authenticity to generative Artificial Intelligence.

To address this link, I present an analysis of my own piece, *你們是蟲子 (You are bugs),* which extends Nono's techniques through generative AI tools and multimedia based on a motion-following architecture, namely *GesTCom*[1][5], [6], [7] and *SOMAX2*[2] [8]. The poetic concept behind this creation is Liu Cixin's *dark forest hypothesis* applied to an imaginary AI invasion, which creates the narrative basis for the notion of "Real

Copyright: © 2025 First author et al. This is an open-access article distributed under the terms of the Creative Commons Attribution License 3.0 Unported, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

[1] https://medias.ircam.fr/en/x2253e1

[2] https://forum.ircam.fr/projects/detail/somax-2/

Authority" as a counterpart to Artificial Authenticity. Real Authority is expressed both in the Human-Machine relationship and in the machine musicianship aspects. I highlight the tension between a fixed electronic medium, namely a video score for *你們是蟲子 (You are bugs)* and the latent free spaces of live Human-Machine interaction, ranging from free improv on Nono's materials, to drones and generative improvisations introduced through the *SOMAX2* and *GesTCom.* Finally, this analysis, which is accompanied by the video score with audio realization, and by a video performance documentation, serves as a trigger for further discussing issues relating to the semantic blind spots of generative AI and to ways of creating meaning through cyber-human musicianship.

2. *…SOFFERTE ONDE SERENE…* AND THE HIGH MODERNIST MODEL OF PERFORMANCE PRACTICE

In [9] Sebastian Berweck provides an overview of the challenges and remedies in contemporary music performance with electronics. One of his case studies is Luigi Nono's *..sofferte onde serene..*, (suffered, serene waves) for piano and tape. In assessing the problems for performance, Berweck points to both audio quality and synchronization. Both types of problems are related to the original mono tape and the denoising technique used during its digitization. The problems leave no realistic remedy options to the performer other than a complete restoration project, such as the one undertaken by Paulo de Assis [10].

Unlike Berweck, whose "findings are almost tragic" for performance authenticity, De Assis considers both the poetics of the work's creation and re-creation as breaking into new ground. Through meticulous analysis of "several new elements [..], namely a new understanding of the use of vertical sound-aggregates ("chords"), the exploration of complex variational and canonical procedures, and, crucially, new modes of organizing "multi-temporalities," with the piano and the tape [..]" [11, p.206], De Assis asserts that "its concert rendering involves various degrees of uncertainty and unpredictability of sonic combinations" [11, p.206].

But the elements of uncertainty and unpredictability are not features of a re-creation strategy alone: They had already infused the very fabric of the work's creation. The piece was indeed based on a creative collaborative process between its dedicatee, Italian pianist Maurizio Pollini, and the composer. The process consisted of recording Pollini's improvisations and using them as input to be processed and remixed by the composer and sound engineer Alvise Vidolin, in ways that promote (con)fusion in the sonic and temporal realms:

> "This (con)fusion is enhanced by relatively free time relations between piano (live) and tape, allowing the performer on the piano and the performer controlling the sound-projection to intertwine a great variety of sonic relationships". [11, p. 206]

The score was composed as a reflection on the recorded improvisation, reversing the usual arrangement, whereby a performance follows the composition of the score and a recording follows a performance:

> "It was the concrete recorded sounds that slowly, in a constructive way, defined more and more precisely the sequence of sonic events—that is, the score for the pianist playing the piece's piano part. And if it is very clear that the score and its writing are the complete responsibility of Nono (who remains "the composer" in an orthodox sense), it is also true that the concrete sonic input produced by Pollini was of the utmost importance for the definition of the music". [11, p.207]

This flexible and unorthodox relationship between improvisation, score, recording and live piano performance manifests in the phenomenology of performance as a dialogue of echoes and resonances, which forms the essence of Luigi Nono's late compositional style, both aesthetically and politically.

De Assis's consideration of the multiplicity of relations between improvisation input, recorded materials, notated scores and final performances is crucial for the project described in the next sections. They particularly inform the sort of relationship sought after between AI agents and human performer. Before touching upon that, I would like to utter another word on the implications of Nono's approach for the wider context defined by Franklin Cox as the "High-Modernist Model of Performance Practice" (from now on abbreviated as "hmmpp").

In [12], Franklin Cox defines hmmpp as "a "noisefree", transparent relationship between all elements of the communicative chain [between conception, notation, performance and reception]" [12, p. 71], The hmmpp elaborates on the idea of a fully deterministic score-image. Such notation is considered able to enclose every possible characteristic of the sonic phenomenon it represents. As a result, it is supposed to assign concrete tasks to the performer and well-defined perceptual tropes to the listener. Cox suggests that complex music has brought about a "fundamental paradigm shift" away from that model. The reason is its purely quantitative characteristics, namely:

> "[..] extreme degrees of both density and fine detail, and [..] coalescence of highly rationalized materials, notated challenges and organization with an extreme physicality and almost irrationality of results" [12, p. 70].

The paradigm shift consists in the transformation of the above-mentioned communicative chain into "[..] an overlapping series of volatile conflicts between incompatibles. Thus, notation is treated as an essentially opaque medium and such notation demands less reading than decipherment" [12, p. 76].

From the discussion above, it becomes clear that the discrete existential domains of improvisation, recording, score and reproduction in Nono may be considered as a transposed instance of Cox's "overlapping series of volatile conflicts between incompatibles", yielding as a result the unpredictability and creativity that De Assis

considers a cornerstone for performing …*sofferte onde serene*… today.

But what if this notion of recreation as volatile conflict between incompatibles is further upgraded to the register of Human-Machine Interaction? How does one notate and represent a construction that potentially undermines representability? Could authenticity be redefined as Artificial Authenticity, when attempting the reproduction of Nono's "non-linear" techniques through machine musicianship? And how are these questions to be treated at the age of Generative AI? To answer these, we resort to the concept of play as metacommunication, in the instance of a 'playbite', developed by Massumi after Bateson.

3. PLAY AS METACOMMUNICATION

In [13] Brian Massumi offers an interesting interpretation on Bateson's definition of animal play.

"The play bite that says "it is not a bite" has the value of the analogous action without its force or function. The wolf cub says through his teeth: "this is not a bite; this is not a fight; this is a game; I am hereby placing myself on a different register of existence, which nevertheless stands for its suspended analogue." [13, p.4]

This take points towards play as some sort of analogous metacommunication of mutually inclusive (rather than exclusive) registers of existence: Fight includes play and play includes fight, or rather they are parts of a continuum that is differentiated by intensity and function. At the same time, this notion of play carries a semantic layer, thus the term "metacommunication", in the sense that through its analogy to fight it makes a non-verbal commentary, a statement, affirming the nature of a game rather than the nature of a fight.

Could we possibly think of the relationships between improvisation and composition, authenticity and authority, humans and machines, along these lines? What if, accepting that an authentic performance of …*sofferte onde serene*… today is irrelevant, we switch gears to a mutually inclusive register of play between what Cox perceives as incompatible existential domains in volatile conflict? This mutually inclusive register of play would denote: "this is not …*sofferte onde serene*…, nor is it a fight for authenticity, but a playbite on artificial authenticity in the face of generative AI". How is such a dynamic of metacommunication representable, both as musical material and as interaction schemes? And what eludes representation?

In that vein, I used …*sofferte onde serene*… as the basic material to create an "Artificially Authentic" double, whereby the "Real Authority" of the composer is outsourced to the score and to software agents, *SOMAX2* and *GesTCom*, trained on …*sofferte onde serene*…, filtered through other music and controlled through sensors, piano and an MPE controller in real-time.

4. ANALYSIS

你們是蟲子] (You are bugs) is conceived as a playbite on Nono's …*sofferte onde serene*…. The work is based on the dialectics between a static form manifested in the fixed video score (Figure 1) and a dynamic interactive process featuring media, human and artificial agents, and outputs (Figure 2).

4.1 Fixed form

The video score consists of a tripartite form. How do we represent a fixed video score? In an affirmation of Rodney Brooks' imperative "to use the world as its own model" [14], I employ a screenshot of the *iMovie* video editor used to create the score, further annotated in *Adobe Illustrator*. For the interaction part to be described in the next subsection, I employ a free adaptation of the visual language described in [15], designed in *Adobe Illustrator*.

The first part (*A*, 00:00 – 03:18) introduces the notion of Real Authority, associated with a video avatar of the author (Figure 3). The audio material is rooted on …*sofferte onde serene*…, including both a track of the original tape [16] and a track of a live recording of the work performed by the author [17]**.** The avatar visuals are based on video synthesized materials employing *Lumen*[3] and *OBS*[4]. Materials of feature films borrowed from *YouTube* are gradually introduced, presenting various authoritarian figures who instruct, order, insult and abuse the performer. The latter treats the recorded materials on stage according to the orders, remixing them live through *GesTCom* and *SOMAX2*.

The second part (*B*, 03:18 – 06:21) features a climactic opposition to Real Authority. The audio material introduces recordings of a work composed by Iannis Xenakis and performed by the author [18], remixed live through *GesTCom* and *SOMAX2*; as well as the invasion of electronic sounds, mostly pink noise, generated by the sound engine *Equator2* and controlled through the MPE controller *ROLI Seaboard RISE2*. The visual materials recede behind purple noise, featuring a single return of a cinematic snippet of yet another authoritarian figure.

The third part (*C*, 06:21 – end) features the return of the Nono material, this time filtered via *SOMAX2* through a recording of music by Maurice Ravel performed by the author. An improvisation on the acoustic grand piano, again based partly on Nono and partly on Ravel, is the only engagement of the author as a pianist performing live (as opposed to remixing). The visuals feature the authoritarian video avatar of the beginning, infused with the purple noise background and yet another snippet from a different film.

4.2 Developing variation

In this section, I explore the dynamic interplay of each part's materials, zooming in to identify detailed parameters and zooming out to identify interaction patterns.

Part *A* consists of two rather distinct sections: A static subsection *a*, which is based on an exposition of the fivefold varied repetition of a loop, consisting of the first 32s of …*sofferte onde serene*…'s tape, and on a recurring video-synthesized avatar of the author, instructing the performer; as well as a subsection *b*, initiating the dynamic

[3] https://lumen-app.com/

[4] https://obsproject.com/

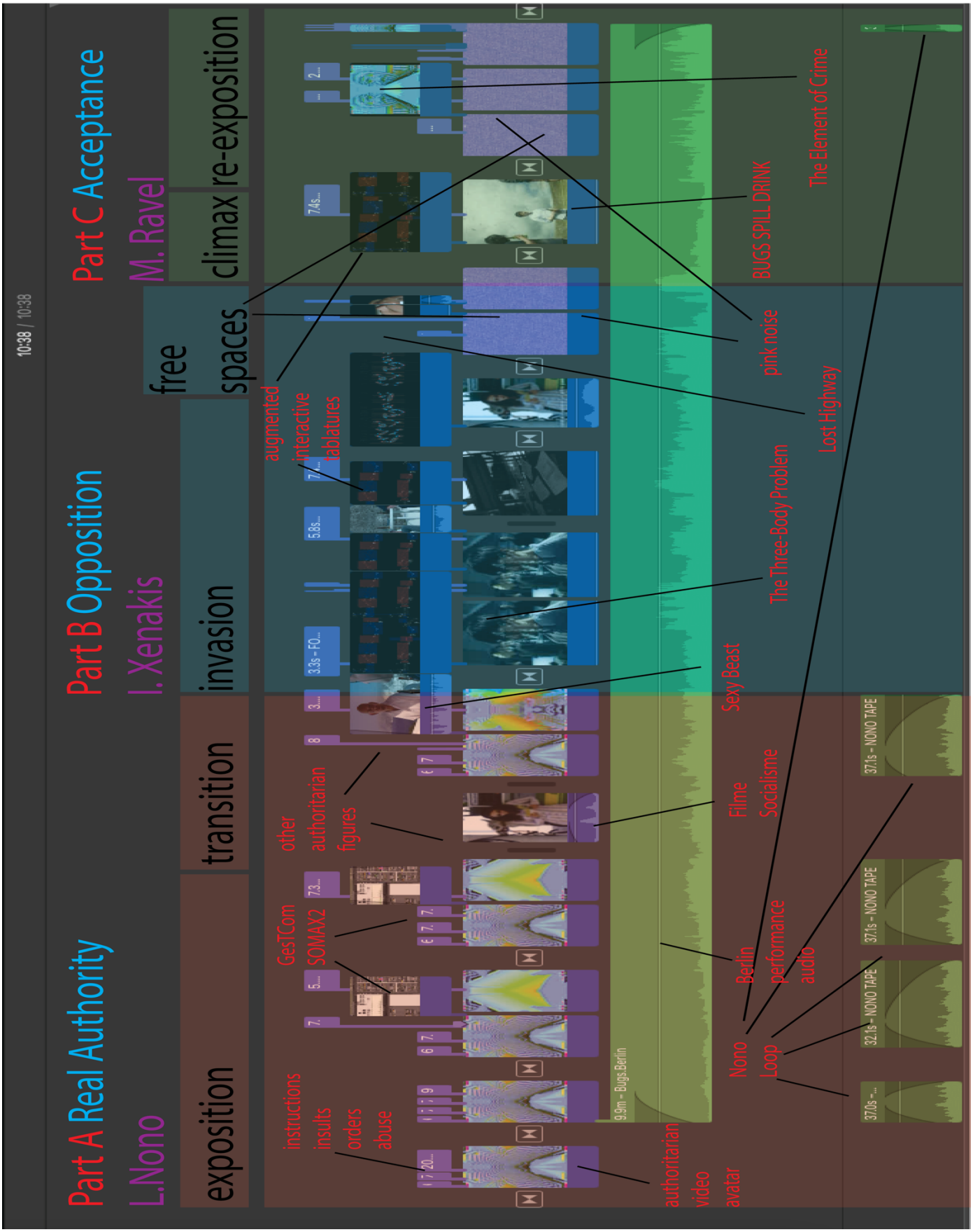


Figure 1: Overview of video score analysis as presented in section 4

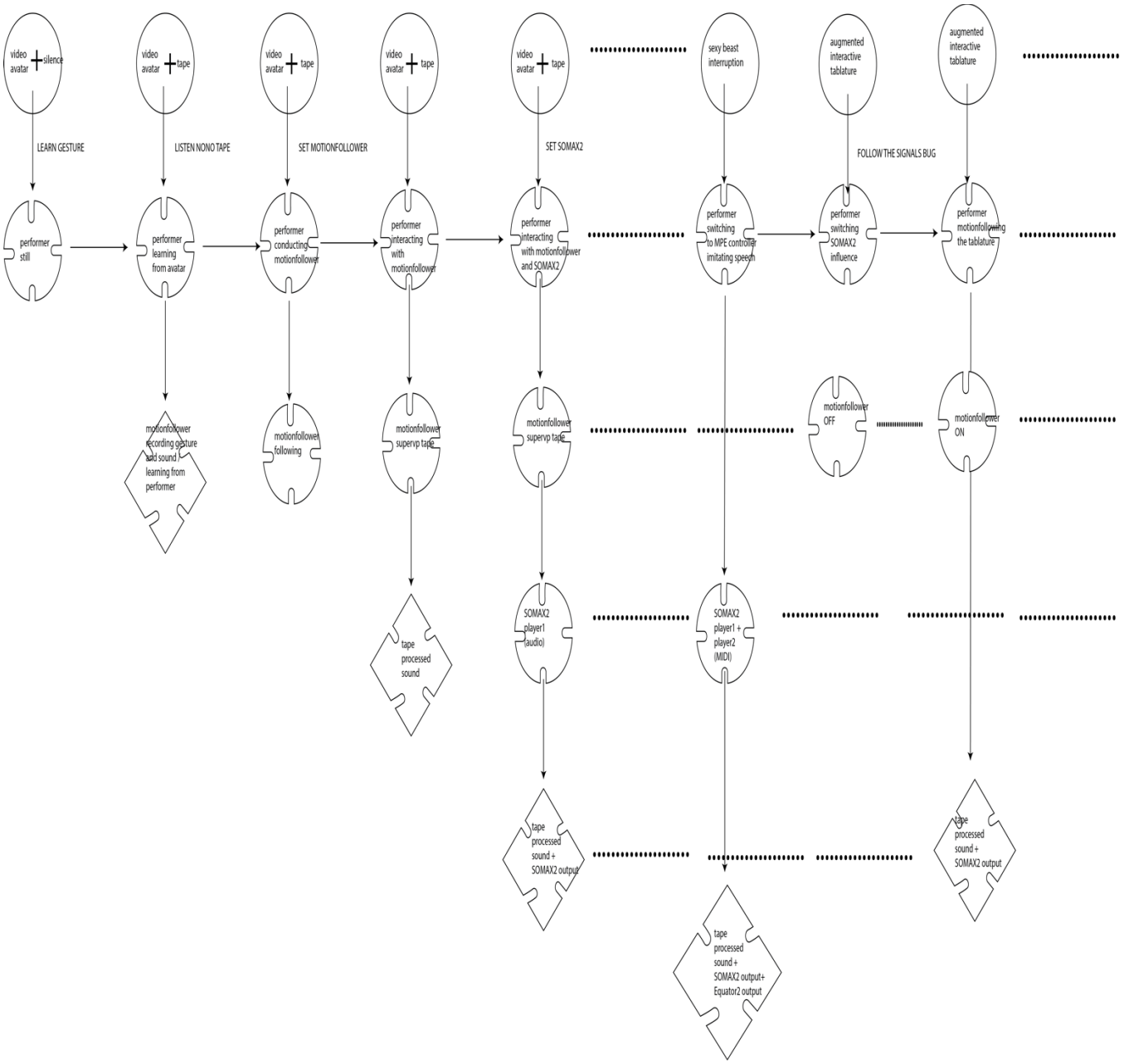


Figure 2: Interaction patterns with media, agents and outputs for part A, as described in sections 4 and 5. New media, agents and outputs, are indicated by circles, edged circles and edged rhombuses respectively. Please notice the proliferation and over-layering of agents and outputs over time.

transition to the next part, which consists of two interruptions of the *a* loop by cinematic materials: a snippet from *Film Socialisme* (2010) by Jean-Luc Godard [19], whereby a teenage girl in a rural gas station in France (played by Agatha Couture) ignores a couple of German tourists asking for instruction (ACH, DEUTSCHLAND, 02:37), and a snippet from *Sexy Beast* (2000) by Jonathan Glazer [20], whereby Ben Kingsley as Don Logan attempts to recruit a retired British gangster for a coming gig (03:12).

The piece starts with an authoritarian video avatar gesticulating in silence and assuming the role of the composer. Inscriptions that function as exclamations and/or instructions appear in the projected video score, for example: "你們是蟲子 (YOU ARE BUGS)". The scene evolves as if performing hypnosis and the inscriptions indicate actions to be performed in real time, driving the interaction. They unfold in ever-expanding cycles, that introduce new media, agents and outputs, as indicated in Figure 2 by circles, edged circles and edged rhombuses respectively. Thus, in the first cycle the inscription LEARN GESTURE orders the performer to mentally imitate the silent gestures that the pseudo-composer is performing. In the second cycle, the beginning of Luigi Nono's original tape for *…sofferte onde serene…* is introduced (LISTEN TO NONO TAPE) and the performer is sarcastically instructed to suffer (SUFFER) while imitating the gestures s/he has learned silently; at the same time, the performer's gestures train a motion follower system recording sound and movement as part of the *GesTCom* system. Thus, the power structure between the video score and the performer expands towards the motion follower that assumes the status of agent in the next cycle. In the third cycle, the performer conducts the motion follower that follows obediently, or varies her movements to obstruct the following, thus over-layering the output sound with frozen drones and spectral artifacts. A significant degree of unpredictability is introduced. The fourth cycle introduces the AI agent, *SOMAX2*, which filters a recording of a performance of the author through the Nono tape, adding to the previous mix. The cycle is abruptly interrupted by the introduction of the first snippet of cinematic materials described above, whereas the fifth cycle, wondering HOW LONG IS THIS GONNA TAKE, warns OUMUAMUA INCOMING, a reference to the first interstellar object formally designated 1I/2017 U1 and detected in 2017, and introducing the second cinematic clip described above.

Part *B* (03:18) introduces new material in both sound and image, capitalizing on an audio recording of Iannis Xenakis's *Evryali* (1973) [21] remixed through *SOMAX2* and *GesTCom*, filtered through Nono's work in the second version of the author's performance, as well as on augmented interactive tablatures based on Xenakis's sketches for *Evryali* and functioning as graphic scores for the performer – another *GesTCom* application. The tablatures are combined with materials from the last episode of the Chinese series adaptation [22] of Liu Cixin's *The Three-Body Problem* [23], as a recurrent threatening and awe-inducing inscription in the retinas of

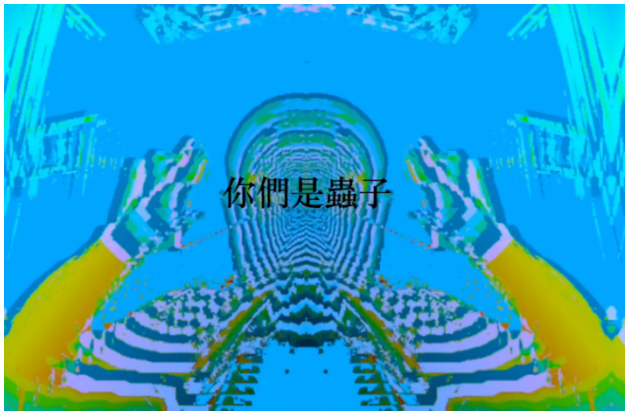


Figure 3: Avatar figure representing Real Authority, Part A

higher human officials (04:03), originating in an alien and superior civilization, the *San Ti*, who invade the Earth after a signal sent to them by a Chinese scientist in the years of Mao's Cultural Revolution. This development is interrupted yet again by a clip from *Sexy Beast,* with Don Logan in ever more abusive appearances, as well as a clip from *Film Socialisme*, with the teenage girl exclaiming that the tourists should "go invade another country". This moment signals an exodus of the score into the "purple noise" segment, introducing a free improvisation on *ROLI Seaboard RISE 2* MPE controller - a sort of tone (or rather pink noise) painting of the various invasions alluded to in the score, produced by the *Equator2* synthesizer (Figure 4). Despite the specification NOTHING TO SEE HERE, as an indication and invitation to free and unrepresentable spaces, the climax recedes fast to give its place to a hidden drone based on *...sofferte onde serene…*, and introducing a clip from David Lynch's *Lost Highway* (1997) [24], and yet another abusive figure, Alice (played by Patricia Arquette): Pete and Alice arrive at a remote desert cabin to fence stolen goods. During sex outside, Alice taunts Pete, saying, "You'll never have me," before disappearing into the cabin. This short episode marks the return of the climactic material comprised by a combo of Xenakis' *Evryali* filtered through Nono and pink noise on *ROLI*, before the introduction-premonition (06:08) of the Ravel material: an unpublished recording of Ravel's *Oiseaux Tristes* (nr. 2 in the collection *Miroirs*, 1904-1905) at IRCAM [25].

Part *C* starts at 06:23 with a stunning moment from the *Three-Body* series, whereby, in a gesture of acceptance of inferiority and animalistic-bug survival in the face of the alien invasion, the main characters perform a unique gesture of spilling their drinks (the equivalent of an ancient Greek *sponde*) with the epigram BUGS SPILL DRINK. The gesture marks both the reprise of the initial avatar material, as well as the first introduction of live piano playing which continues to the end, an improvisation on *…sofferte onde serene…* materials aimed to "zombify" the Pollini improvisation as transmitted through the Nono tape. The initial epigrams-instructions are slightly varied, through the inscription LEARN COMPOSURE AS WELL

and there is only a short interruption-reminiscence of the climactic part, in the form of a clip by Lars von Trier's *The Element of Crime* (1984) [26] - an image of a horse (yet another animal) accompanying the SUFFER instruction. This Nono tribute ends with a final loop of the tape, over the inscriptions SUFFER SERENE WAVES (the translation in English of *sofferte onde serene*).



Figure 4: Pink noise patch in *Equator2*

5. INTERACTION WITH AI AGENTS

In this section, more technical details are provided on the interaction patterns between the AI agents and the human performer as diagrammed in Figure 2.

5.1 LEARN GESTURE: The use of *GesTCom*

By virtue of a predefined syntax of movement and *a* probabilistic motion-following methodology employing Hidden Markov Models [27], *GesTCom* can be trained to follow (or unfollow) the performer during variations of the initial performance. This process involves a *recording phase* and a *following phase*. In the recording phase, the user-performer follows the movements of the avatar figure in Part A to train *GesTCom*'s *motionfollower*. Alternatively, performers can also follow any mobile element of the augmented interactive tablatures (in part B 03:20 – 03:43), which is set to move at a desired speed, like a classic metronome would do.

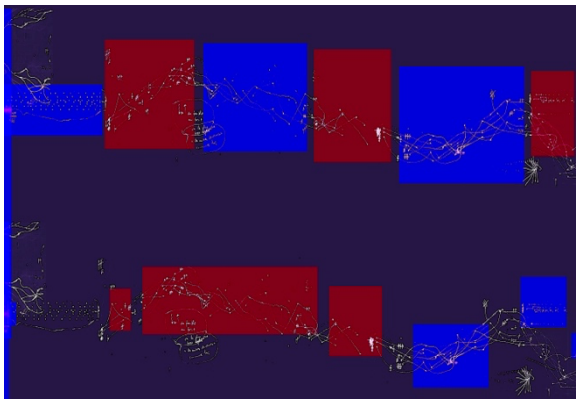


Figure 5: explicit score mapping on a Xenakis sketch for *Evryali* for motion following, Part *B*, realized with *INScore*. Sketch transcription by Aurélien Duval.

The musical sketch of Figure 5 has already been graphically segmented through the INScore and assigned a duration according to a space-time formalism (explicit mapping). In this phase, the motion follower "learns", so to speak, the mapping from the performer's gesture captured by R-IoT IMUs (implicit mapping), while s/he follows the video avatar or the explicit mapping of the video or sketch. In the next phase, defined as "following", the performer can pursue highly varied performances, ranging from heterophonic re-interpretations of the original to the introduction of completely novel material that shares the same gestural segmentation. This time, it is not the performer that follows the system, but rather the system that follows the performer, given that the segmentation is correct and common in all these varied performances. Thus, the performer may control the mobile elements of the tablature. The feedback of the follower has been extended to score compound representations. The gesture-following has been turned into score-following.

In Figure 6, the grey signal represents the implicit mapping - gesture that the augmented multimodal tablature "learns" along the explicit mapping of Figure 5 in the *recording phase*. The green signal represents the new incoming signal that controls the tablature in the *following phase*, the signal that the tablature follows during variations of the initial performance.

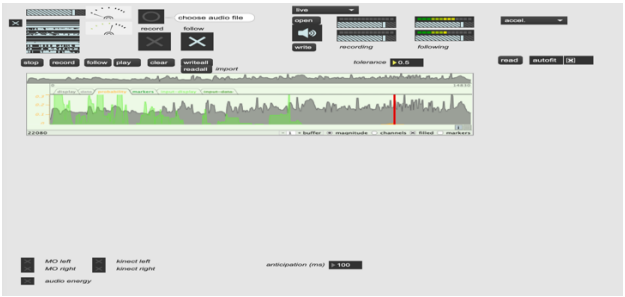


Figure 6: interface for motion following

More than following, *GesTCom* here has a function of unfollowing: Learned gestures are intentionally obstructed, or the tolerance of the motion follower algorithm is set extremely low, so that the system cannot follow, thus sabotaging its initial intent**.** As a result, various drones occur, as the *supervp.scrub~* object (advanced phase vocoder position controlled player module) allows resynthesized audio output to be time-stretched, transposed in pitch and undergo several spectral envelope transformations (Figure 7). More importantly, these effects can be unpredictable. The triggering from the performer's conducting gestures is not hard-wired: It partly depends on the complexity of the input recording and the performer's knowledge of it, on the degree of accuracy of the performance during both the *recording* and the *following* phase, and on the algorithm tolerance threshold. In this respect, it acts as the meta-commentary or playbite evoked in the introduction, and it eludes the usual mapping strategies in precisely controlling parameters with hand gestures. As such, it allows for a

very flexible and often surprising re-shaping of the input recording.

The motion following software, alone or combined with the sketch-based tablature was used in combination with audio tracks by Luigi Nono, Maurice Ravel and Iannis Xenakis as described above. The first introduction of the system's following function is featured in 01:42 – 01:53 of the video score and in 0:41 onwards on the performance video.

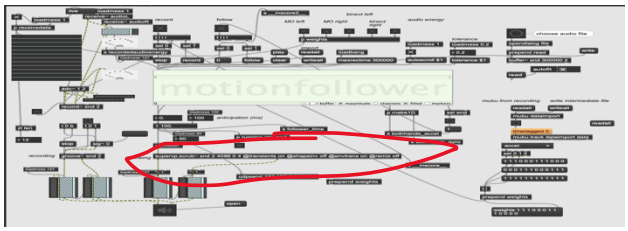


Figure 7: supervp.scrub~object

5.2. Training the AI Agent for comprovisation: *SOMAX2*

The motion following output, alone or combined with the sketch-based tablature, was used in combination with audio tracks by Luigi Nono, Maurice Ravel and Iannis Xenakis, as well as with MIDI input from the *ROLI Seaboard RISE 2* as training and influencer material for *SOMAX2*. *SOMAX2* is a multi-agent interactive system performing live machine comprovisation with musicians, based on machine-listening, machine-learning, and generative units. The current version [8] is a recent development and algorithms' improvement from the former *SOMAX* version. Agents provide stylistically coherent improvisations based on learned musical knowledge while continuously listening to and adapting to input from musicians or other agents in real time. The system is trained on any musical materials chosen by the user, effectively constructing a generative model (called a *corpus*), from which it draws its musical knowledge and improvisation skills. Corpora, inputs and outputs can be MIDI as well as audio, and inputs can be live or streamed from MIDI or audio files. *SOMAX2* is one of the improvisation systems descending from the *Omax* software [28], presented here in a totally new implementation. As such it shares with its siblings, the general loop [listen/learn/model/generate], using some form of statistical modeling that ends up in creating a highly organized memory structure from which it can navigate into new musical organizations, while keeping style coherence, rather than generating unheard sounds as other Machine Learning systems do. The open-ended nature of the resulting system allowed me to experiment with different kinds of interaction and materials in a setting of structured comprovisation based on Nono's original tape, a recording of *…sofferte onde serene…*, as well as Xenakis' *Evryali* and Ravel's *Oiseaux Tristes*, and finally Xenakis's *Evryali* sketches. Capitalizing on the chroma affinities between Nono, Xenakis and Ravel, I was able to construct composite corpora including harmonically similar language.

In Part *A*, an audio influencer treating sequentially an audio file of the Nono tape part (for loop 2), and an audio file of a complete performance of *…sofferte onde serene…* by the author (for loops 3-5) was used in pitch detection mode (loops 2-3) and, as the density increased, in onset detection mode (loops 4-5). The function of influencing equals a type of stylistic filtering of the Nono original tape. In Part *B*, an audio influencer playing the piece's complete performance, as well as a MIDI influencer capitalizing on the live input from *ROLI*, filtered a recording of Iannis Xenakis' *Evryali*. Finally, in Part *C* the materials of reference in the Player oscillated between Ravel and the tape by Nono, filtered by the audio of the live piano performance and the MIDI input by *ROLI*. Please note that the employment of up to four Players is possible, which would probably explode the complexity of interaction described above.

Several degrees of distancing (same, similar and alien material) of the human pianist from the original material used to train *SOMAX2* result in a variety of interactions, reflecting the echoes and resonances of Nono's own aesthetics. In a second instance, the pianist performer can shape the output of the *SOMAX2* through the motion follower, allowing for the supervp.scrub~ object (advanced phase vocoder position controlled player module) audio output to be sent to *SOMAX2*.



Figure 8: SOMAX2 setup

5.3. New Virtuosity

The virtuosity involved in controlling *GesTCom* and *SOMAX2* in live settings is nontrivial, and so is the possibility to enter a dialogue with materials already learned and recorded, but also to surprise the systems as well as the human performer with alien materials not featured in the Nono score or in the recordings. Several types of action are employed, including: silent conducting; recording conducting gestures and sounds; imitating sound from fixed media; following and varying gestures and sound; launching programs and tweaking parameters; piano performance, including traditional score reading and free improvisation; controlled improvisation, including the performance of *GesTCom* tablatures and/or variations of the notated Nono score; and finally, performance of the MPE controller, including 5D touch such as note-on

velocity MIDI messages (strike), polyphonic aftertouch or channel pressure MIDI messages (Press), MIDI pitch bend value on a given MIDI channel (Glide), MIDI Control Change Message (Slide) and note-off velocity MIDI message (Lift)[5]. Employing a paradigm of interaction and musical context based on Rasmussen's model of human information processing [29], [30] we can here detect all three types of behavior: skill-, rule- and model-based behavior. Skill-based behavior involves real-time, ongoing responses to continuous signals. In contrast, rule-based behavior entails selecting and executing stored procedures based on cues extracted from the system. Finally, model-based behavior is even more abstract, with actions directed toward a conceptual goal, requiring active reasoning to determine the appropriate response—whether rule-based or skill-based. In the repertoire of actions described above, instances of skill-based and rule-based behavior are self-explanatory, whereas model-based behavior is detected both on the macroscopic level of the multimedia composition of the work, as well as on the microscopic but potentially very calculated responses to the semantic network established by the cinematic and video-synthesized materials. Thus, the work is inscribed in the general discussion on contemporary musical virtuosities as exposed in [31].

6. FROM SCORE INTERPRETATION TO HUMAN-AI COMPROVISATION AS PLAYBITE

The topic of algorithms inheriting human biases, both hidden and overt, has sparked heated discussions in AI research. Notably, epistemic biases [32], [33] often reflect preconceived ideas and models about human thought, learning, and judgment that are integrated into AI systems without much scrutiny. Consequently, these systems, along with their human components, are often stuck in traditional frameworks, missing out on modern advances in their areas.

The probabilistic systems used in *GesTCom* and *SOMAX2* reveal deep-seated epistemic biases, especially in how music knowledge is represented. Both rely heavily on the probabilistic scanning of audio—responding to gestures in the motion follower or analyzing pitch and chroma in *SOMAX2*. This reveals what some describe as AI's "semantic blind spot," [34] where states lacking memory swap syntactic roles without creating meaningful relationships or causal effects, unless genuine, unpredictable changes occur. Such change springs from the unknown, not from familiar, navigable data.

Looking back, one might argue that symbolic biases rooted in classic replication models—or their modern incarnations like high-modernist performance practices—are rigidly imposed. These include the unchanging nature of scores, their layered parameters, the symbol grounding challenge [35], [7]. These biases constrain innovative systems of improvisation, highlighting issues such as symbolic representation's limits when compared to real-world experiences, the preference for abstract sound patterns over rich social interactions, and simplifying complex dynamics into basic parameters [36].

However, the idea of creating diverse connections among distinct representation domains—like improvisations, recordings, videos, and scores—is promising. It allows for unexpected, personalized interactions within time, even if these domains merely rearrange fixed timelines. The potential for breaking one-to-one score associations, exploring one-to-many mappings, and creatively transforming them into video scores with both familiar and novel materials is enticing.

Finally, the question of a semantic blind spot is addressed through the inherent semanticity of the borrowed cinematic materials. A network of inherently meaningful materials is articulated into an overarching narrative form, counterbalancing the free play of semantically neutral permutations.

7. CONCLUSIONS

I have attempted to present a concise analysis of my piece 你們是蟲子] (*You are bugs*) reflecting on the relationship between authenticity and creativity in relation to generative AI. Reviewing Nono's late aesthetics, introduced by his piano and tape piece *..sofferte onde serene..* , I wonder if the poetics of transformation of an original improvisation by Maurizio Pollini into a fixed piano score accompanied by a tape echoing and ghostly reflecting the original could be more accurately rendered as an AI-generated reconstruction of the piece, whereby Artificial Authenticity is coupled to Real Authority. The term of choice in this case is the notion of "playbite", exhibited by the philosopher Brian Massumi after Bateson, as an analogical existential domain of Otherness, which also bears a semantic layer potentially invaluable, given the epistemic biases exposed in the last section as far as AI is concerned. After exposing my concepts, I offered an analysis of the video score and I elaborated on the tools and concerns they spark.

8. REFERENCES

[1] R. Mitra and I. Zualkernan, "Music Generation Using Deep Learning and Generative AI: A Systematic Review," in *IEEE Access*, vol.13, pp.18079-18106, 2025, doi:10.1109/ACCESS.2025.3531798 https://ieeexplore.ieee.org/abstract/document/10845168 accessed 06.10.2025

[2] Gioti A-M. "From Artificial to Extended Intelligence in Music Composition". *Organised Sound*. 2020;25(1):25-32. doi:10.1017/S1355771819000438

[3] Dubnov, S. and Greer, *Deep and Shallow. Machine Learning in Music and Audio.* New York: Chapman and Hall/CRC, 2024

[5] https://support.roli.com/support/solutions/articles/36000376229-roli-seaboard-2-creator-manual-previously-rise-2-

[4] Luigi Nono, *...sofferte onde serene...*, 1st ed. (Milan: Edizione Ricordi, 1977)

[5] Antoniadis, P., Bevilacqua, F. and Fober, D., "Gesture cutting through textual complexity: Towards a tool for online gestural analysis and control of complex piano notation processing", Proceedings of Conference on Interdisciplinary Musicology, 2014, Berlin, Germany

https://hal.science/hal-02158741/file/CIM14-final.pdf accessed 06.10.2025 accessed 06.10.2025

[6] Antoniadis, P. *Embodied navigation of complex piano notation: rethinking musical interaction from a performer's perspective.* PhD Thesis. Université de Strasbourg & IRCAM, 2018 https://theses.hal.science/tel-01861171/file/Antoniadis_Pavlos_2018_ED520.pdf accessed 06.10.2025

[7] Antoniadis, P. and Chemero, A. "Playing without mental representations: Embodied navigation and the *GesTCom* as a case study for radical embodied cognition in piano performance". *journal of interdisciplinary music studies* season 2020, volume 10, art. #20101207, pp. 126-174 https://unipub.uni-graz.at/jims/download/pdf/5758295?fbclid=IwY2xjawNQadJleHRuA2FlbQIxMABicmlkETFWOUhwVEpDeHJ3TjlpWWhQAR5uxtLEihsvyk6WjRATZ220hr08_s2etGWvhh7cRzEb2b1ldFUb_yfaM6HL9Q_aem_bLJSoZAhCp4QIXY9Yk_9OA accessed 06.10.2025

[8] Gérard Assayag, Laurent Bonnasse-Gahot, Joakim Borg, "Cocreative Interaction: Somax2 and the REACH Project". *Computer Music Journal* 2022; 46 (4):7–25. doi: https://doi.org/10.1162/comj_a_00662 accessed 06.10.2025

[9] Sebastian Berweck, *It worked yesterday: On re-performing electroacoustic music*, Doctoral Dissertation, University of Huddersfield, 2012, http://eprints.hud.ac.uk/17540/ accessed 06.10.2025

[10] Paulo de Assis, *Luigi Nonos Wende. Zwischen Como una ola fuerza y luz und .....sofferte onde serene...* (Hofheim: Wolke, 2006)

[11] Paulo de Assis, «Revisiting Luigi Nono's Suffered, Serene Waves », in: *Artistic Experimentation in Music: An Anthology* (eds. Crispin, D. and Gilmore, B.), Leuven University Press, 2014

[12] Franklin Cox, "Notes Toward a Performance Practice for Complex Music" in Claus-Steffen Mahnkopf, Frank Cox and Wolfram Schurig, eds.: *Polyphony and Complexity, New Music and Aesthetics in the 21st Century*, Hofheim: Wolke Verlag, pp. 70-132, 2002

[13] Massumi, B. *What Animals Teach Us About Politics*, Duke University Press, 2014

[14] Rodney A. Brooks, "Intelligence without representation", *Artificial Intelligence 47* (1991), 139–159.

[15] Baalman, M. *Composing Interactions. An Artist's Guide to Building Expressive Interactive System*, V2_Publishing: Rotterdam, 2022

[16] Luigi Nono, *...sofferte onde serene...*, 1st ed. (Milan: Edizione Ricordi, 1977)

[17] Antoniadis, P. Live recording at SARC, Queen's University, Belfast, 18.01.2018 of Luigi Nono, *...sofferte onde serene...*, 1st ed. (Milan: Edizione Ricordi, 1977)

https://on.soundcloud.com/2xWXgYYKXt3pIzTeFH accessed 06.10.2025

[18] Antoniadis P. Live recording at Amphi X, Université Paris 8, 29.03.2021 of Iannis Xenakis *Evryali* (1973) https://on.soundcloud.com/MVjIwbaZqJ4HaQ2IuN, accessed 06.10.2025

[19] Godard, J.L., *Filme Socialisme*, Vega Film, 2010

[20] Glazer, J., *Sexy Beast*, FilmFour/Kanzaman S.A./ Recorded Picture Company/Jeremy Thomas Productions, 2000

[21] Xenakis, I., *Evryali*, Salabert, 1973

[22] Tian Liangliang, *Three-Body* (*三体*), Tencent Video/CCTV/The Three-Body Universe/Linghe Cultural Media, 2023

[23] Cixin, L., *The Three-Body Problem* (三体), Chongqing Press, 2008

[24] Lynch, D., *Lost Highway*, Ciby 2000 / Asymmetrical Productions, 1997

[25] Ravel, M. « Oiseaux Tristes » (nr. 2 in the collection *Miroirs*, Edition Peters, 1904-1905

[26] Von Trier, L. *The Element of Crime*, Det Danske Filminstitut, 1984

[27] Bevilacqua, F. et al 'Online Gesture Analysis and Control of Audio Processing. In: Solis, J. & Ng. K.(eds.), *Musical Robots and Interactive Multimodal Systems*. Berlin Heidelberg: Springer-Verlag, 2011, 127-142

[28] G. Assayag, G. Bloch, M. Chemillier, A. Cont, S. Dubnov, "Omax brothers: a dynamic topology of agents for improvisation learning,", in *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*. Santa Barbara, United States, 2006, pp. 125–132.

[29] Rasmussen, J. *Information Processing and Human-Machine Interaction: An Approach to Cognitive Engineering*. Elsevier Science Inc., New York (1986)

[30] Malloch, J., Sinclair, S., Hollinger, A. and Wanderley, M. M. "Input Devices and Music Interaction" in Solis, J. & Ng. K.(eds.), *Musical Robots and Interactive Multimodal Systems.* Berlin Heidelberg: Springer-Verlag, 2011

[31] Devenish, L., Hope, C. (eds.) *Contemporary Musical Virtuosities*, Routledge, 2025

[32] M. Shanahan, "Talking About Large Language Models," arXiv:2212.03551v2 [cs.CL], 2022. https://doi.org/10.48550/arXiv.2212.03551

[33] Antoniadis, P. "Challenging epistemic biases in musical AI: a guerrilla approach to human – machine comprovisation based on Xenakis's sketches for *Evryali*", in *Proceedings of TENOR24 - 9th International Conference on Technologies for Music Notation and Representation (TENO*R), pp. 65-72 https://www.tenor-conference.org/proceedings/2024/09_TENOR2024_Antoniadis.pdf accessed 06.10.2025

[34] AA. Cavia, *Logiciel. Six Seminars on Computational Reason*, Berlin: The New Centre for Research and Practice, 2022

[35] L. Barsalou, "Perceptual Symbol Systems," in *Behavioral and Brain Sciences*, 22: pp. 577–609, 1999

[36] D. Borgo, D. *Sync or Swarm. Improvising Music in a Complex Age*, Bloomsbury, 2022.

DETECTING NOTATIONAL ERRORS IN DIGITAL MUSIC SCORES

Léo Géré
Cnam, Cedric, Paris, France
leo.gere@lecnam.net

Nicolas Audebert
IGN, LASTIG, Saint-Mandé, France
Cnam, Cedric, Paris, France
nicolas.audebert@ign.fr

Florent Jacquemard
INRIA, Paris, France
Cnam, Cedric, Paris, France
florent.jacquemard@inria.fr
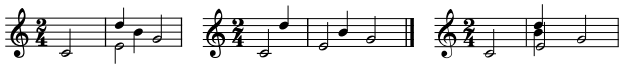
ABSTRACT

Music scores are used to precisely store music pieces for transmission and preservation. To represent and manipulate these complex objects, various formats have been tailored for different use cases. While music notation follows specific rules, digital formats usually enforce them leniently. Hence, digital music scores widely vary in quality, due to software and format specificity, conversion issues, and dubious user inputs. Problems range from minor engraving discrepancies to major notation mistakes. Yet, data quality is a major issue when dealing with musical information extraction and retrieval. We present an automated approach to detect notational errors, aiming at precisely localizing defects in scores. We identify two types of errors: *i)* rhythm/time inconsistencies in the encoding of individual musical elements, and *ii)* contextual errors, *i.e.* notation mistakes that break commonly accepted musical rules. We implement the latter using a modular state machine that can be easily extended to include rules representing the usual conventions from the common Western music notation. Finally, we apply this error-detection method to the piano score dataset ASAP [1]. We highlight that around 40% of the scores contain at least one notational error, and manually fix multiple of them to enhance the dataset's quality.

1. INTRODUCTION

Sheet music scores are, since centuries, an essential medium for the exchange of information between composers, performers, teachers, students, scholars, *etc*. We focus in particular on common Western music notation. Scores are rich and complex hierarchical objects, containing information about notes, rhythms, meter, dynamics, expression... Various encodings have been proposed over the years to represent these musical data digitally, such as **kern [2], ABC notation [3], MusicXML [4, 5], MEI [6, 7].

As complex as they can be, scores must follow a certain number of syntactic rules to be correctly analyzed, or for their content to be leveraged in different applications, including graphical rendering [8, 9, 10]. For example, a half note has twice the duration of a quarter note; notes in a same voice cannot overlap each other unless they form a chord, *i.e.* a group of simultaneous notes of the same duration. Score encoding formats are usually lax regarding compliance with these rules, which can lead to unconventional scores difficult (or even impossible) to work with. In particular, verbosity and information redundancy in these formats may be sources of inconsistencies. For example, in MusicXML, note durations are expressed both using symbols (quarter/half/eighth..., eventual dots) and numerical values (number of quarter notes on a timeline). However, the specification does not enforce any consistency between those, which can lead to issues in voice alignment and parsing difficulties. Those notational errors [1] can come from music editing software export, from conversion between different encoding formats, or from human encoders themselves. Differences in encoding have an impact on music analysis tasks [11], supporting the importance of having correct encoding to effectively leverage musical scores. Clean data is also primordial to train machine learning models [12], especially models producing complex structured output such as score data, in the context of tasks such as Automatic Music Transcription or Optical Music Recognition.



(a) MuseScore 4.5.2 (b) Dorico 6.0.22 (c) Verovio 4.2.0

Figure 1. Example of the same erroneous MusicXML file (presented in Appendix A) rendered by three different score rendering/editing software applications.

All music editing software perform validation, *i.e.* checking respect of the MusicXML specification. However, some also validate the musical content itself, looking for inconsistencies. Furthermore, programs sometimes applies automated error-fixing mechanisms, *e.g.* by raising explicit errors to the users, or sometimes by silently "correcting" the score. Yet, those automated checks are often limited and software-specific. As shown in Fig. 1, different software handles errors differently. In practice, we argue that it might be easier to point the error to the user, and let them fix it themselves rather than guessing their initial intention.

To evaluate the quality of digital score libraries containing scores from different sources, formats, and using different conventions, the GioQoso Project [13, 14, 15] proposes a quality model evaluating diverse aspects of the scores. It investigates a wide range of potential quality issues: score

[1] Composers do sometimes voluntary break some music notation rules *e.g.* for artistic reasons, but we will consider those as errors in this work.

Copyright: ©2025 Léo Géré et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

content issues, engraving issues, metadata issues, *etc*. In this work, we only consider the *musical content* of a score independently from any engraving and graphical rendering concerns. In other words, we only deal with the logical structure of the score, without considering the exact placement of elements relative to each other on a printed sheet.

In this article, we introduce a two-step error detection method that aims at precisely localizing notational mistakes in a score. First, we perform a rhythm/time consistency check, that detects musical elements whose theoretical duration, as described by their symbol, is inconsistent with the actual duration specified in the MusicXML tags. See Section 2.1 for the description of this kind of errors and Section 3.1 for their detection. We chose to focus on MusicXML, as it is one of the most common open digital music format. In the second step, we ensure that the score complies with a set of conventional music notation rules, see Sections 2.2 and 3.2. Our validation algorithm is flexible, built upon a set of music tokens and a state machine that could easily be extended with new rules and conventions. This second step is format-agnostic and could be extended to any encoding, provided that a tokenizer is implemented for the required format. We apply this method to the ASAP piano scores dataset [1], and identify problematic sections in approximately 40% of the scores (Section 4). Our manual qualitative analysis shows that these are notational mistakes, due to annotator error and poor software exports. We also improve the overall quality of the ASAP dataset by fixing some of these errors.

2. ERROR TYPES

There are multiple sources and types of errors, and we consider the following two categories: individual errors, and contextual errors. Individual errors refer to errors on single elements that we can detect on their own, without considering their interpretation context. Contextual errors refer to elements that are correct on their own, but break musical rules when taking into account their musical surrounding, *e.g*. overlapping notes in a voice.

2.1 Individual Errors

The errors presented in this subsection are specific to MusicXML due to how this format encodes some information.

2.1.1 Duration Representation in MusicXML

In MusicXML, durations are expressed as a number of quarter notes on a timeline, represented as exact fractions. The denominator is encoded at the measure level in the integer `<divisions>` tag.[2] The numerator is stored in the individual elements, in an integer `<duration>` tag. For instance, in a measure with an attribute `<divisions>` of 24, a note with a tag `<duration>` of 48 would last the duration of $\frac{48}{24} = 2$ quarter notes. This fraction represents the actual duration of the note on the timeline, but is incomplete with respect to the symbology of the note *value*, *i.e*. whether it is a quarter note, a half note, *etc*., and whether it has augmentation dots, or belong to a tuplet. For these purposes MusicXML also includes respectively the `<type>`, `<dot>` and `<time-modification>` tags.

[2] Usually, only one single common denominator is used for the whole score, introduced in the first measure.

2.1.2 Inconsistencies in Redundant Duration Information

Individual errors often stem from inconsistencies in redundant information, typically in MusicXML's double encoding of the duration. As just explained, a note with a `<duration>` of 48 in a 24-division-based MusicXML, has a duration of two quarters notes, or a half note. Yet, the MusicXML specification does not enforce that the `<type>` tag represents a half note. Therefore, some notes can have a timeline duration that is inconsistent with their theoretical duration. Parsers then have to decide which one to use.

These inconsistencies are generally rooted in attempts to improve the user experience, *e.g*. by writing the score so that its engraving or MIDI playback matches the user's expectations – at the expense of the musical content correctness. Grace notes are a typical example when they are written as normal notes. Ornaments are another one when added as invisible notes.

Tuplets are also a source of duration inconsistencies, due to rounding errors. MusicXML theoretically allows representing any fractional duration by setting an appropriate `<divisions>` tag. Yet, some software exports scores with a default value when faced with unusual tuplets, *e.g*. 480 for MuseScore up to version 4.4. Because some notes durations cannot be expressed as a fraction with this denominator, their `<duration>` tag is rounded to the nearest integer resulting in quantization errors. For example, a note in a septuplet of eighth notes has a theoretical value of $\frac{1}{2} \times \frac{8}{7} = \frac{4}{7} \approx 0.5714$ quarter notes, while in a 480-division-based MusicXML file it would have a duration of $\frac{\text{round}(4/7 \times 480)}{480} = \frac{274}{480} \approx 0.5708$ quarter notes.

2.1.3 Dubious `<backup>` and `<forward>` Tags

MusicXML uses `<backup>` and `<forward>` tags to align the different voices on the timeline. For example, once a voice has been completed, a `<backup>` tag is used to "rewind" the timeline back to the start of the measure and start filling a new voice. Similarly, `<forward>` tags are used to artificially "fast-forward" in the measure, generally in the case of incomplete voices. Like notes, these tags have a duration indicating the displacement on the timeline.

Although there are no set rules constraining the duration of these tags, some unusual values are likely to indicate an underlying issue. Consider a 480-division-based MusicXML file, in which a `<forward>` tag has a duration of 1. This means stepping forward on the timeline by a duration of a 1024th note in a triplet, nested in a quintuplet. This is not strictly speaking forbidden, although it is unlikely. Music editing software typically uses such `<forward>`/`<backup>` tags to fill small gaps due to the accumulation of rounding errors in case of the ill-chosen `<divisions>` tags discussed above.

2.2 Contextual Errors

The errors presented so far were errors on their own and did not need any context to be qualified as errors. However,

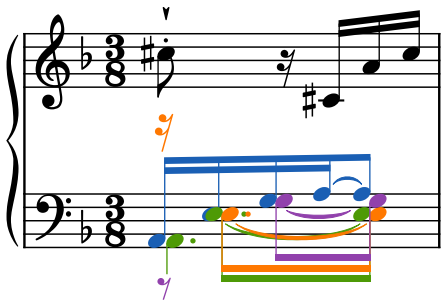


Figure 2. Example of overlapping notes in measure 356 of 3rd movement of Beethoven's Piano Sonata No. 17 in the ASAP dataset [1]. The notes are colored by voice. The notes in the green voice overlap with each other: no other note should start on that voice after the first dotted quarter note. This issue might come from a conversion error in the history of the score, merging two voices into one.

some errors are more complex and do require information about surrounding notes to be properly identified.

2.2.1 *Voice Overlap Errors*

We consider a voice to be a succession of notes, without any gap or overlap except for chords. In polyphonic music, a score can contain multiple voices in parallel, emphasizing different melodic lines. Voices can sometimes be *incomplete*, for example, they can start in the middle of the measure, along with an already started voice. In this article, we only consider *complete* voices, *i.e.* without any gaps. In case of incomplete voices, we pad them with rests.

Typical errors in voices are overlapping notes, as illustrated by Fig. 2. The second and third notes in the green voice overlap with the first one, and should have been put into a different voice.

2.2.2 *Measure Overflow*

In theory, the time signature constrains the duration of the measure, *i.e.* it sets the *nominal duration*. For example, a voice in a 4/4 measure is expected to contain the equivalent of four quarter notes. In practice, as there are valid cases in which a measure is shorter than its nominal duration, *e.g.* pickup measures or repetition bars placed in the middle of a measure. Hence, we only treat measures longer than their nominal duration as errors.

2.2.3 *Chord Errors*

Chords are groups of notes belonging to the same voice, starting at the same time and lasting the same duration. Once again, this is not constrained by the MusicXML specification. Therefore, we want to explicitly ensure that all same-voice notes starting at the same time also share the same duration, meaning that they belong to the same chord.

In addition, we can include some instrument-specific rules related to chords. In the case of the piano dataset ASAP presented in Section 4.2, the piano is not allowed to play the same note multiple times simultaneously. Hence, chords in piano scores should not contain duplicate notes. [3]

[3] Yet, there can be duplicate notes if they belong to different voices.

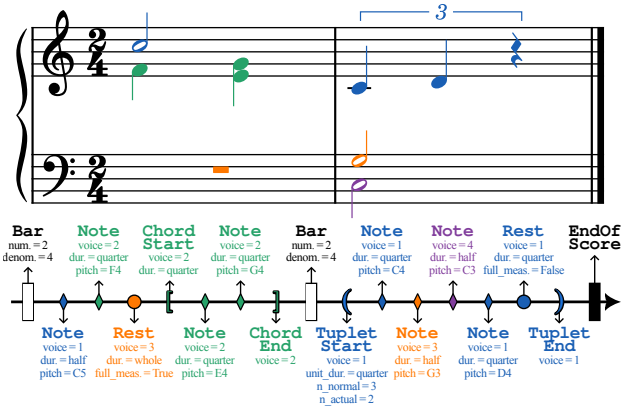


Figure 3. Example of tokenization of a score. The timeline represents the sequence of tokens. The items' shape represents the type of token, and their color the voice they belong to (only for tokens tied to a voice).

3. ERROR DETECTION METHODS

We now introduce our error detection method, which is separated into two stages. The first stage checks for individual rhythm/time consistency errors as introduced in Section 2.1, while the second one checks for contextual errors presented in Section 2.2. Individual errors are straightforward to spot as they only require analyzing each element independently. However, contextual errors are more challenging, as they require considering the neighborhood of the tested elements.

Note that we focus on verifying the actual *musical content* of the score. In other words, we *i)* consider only well-formed files, *i.e.* respecting the MusicXML specification [4], malformed files are supposed to be detected beforehand, and *ii)* exclude engraving/rendering-related issues.

3.1 Detection of Individual Errors

We ensure the duration consistency of every note and rest, as explained in Section 2.1. First, we compute the theoretical duration of each of them from the `<type>` and `<dot>` tags, taking into account any tuplet modifier tag `<time-modification>`, if present. Second, we compare this value to the actual timeline duration of the element encoded in the MusicXML. If they differ, we flag the note/rest as faulty for further inspection.

In addition to notes and rests, we also inspect `<backup>` and `<forward>` tags for suspicious durations. To do so, we ensure that each of those tags can be decomposed into a valid sequence of rests, up to some user-defined minimal duration, like 128th notes. For example, in a 480-division-based MusicXML file, a `<forward>` of duration 45 could be decomposed into a 64th rest (30) and a 128th (15) rest, but a `<forward>` of 46 could not without allowing much smaller divisions. Any tag that cannot be decomposed into an allowed sequence of rests is flagged as faulty.

3.2 Detection of Contextual Errors

Detecting more complex errors than individual errors requires taking into account the interpretation context of each

[4] `https://www.w3.org/2021/06/musicxml40`

Token type	Definition	Attributes
`Bar`	Beginning of a new measure	• $T_{\text{numerator}}$: Numerator of the time signature of the measure • $T_{\text{denominator}}$: Denominator of the time signature of the measure
`Note`	Single note	• T_{voice}: Voice index • T_{pitch}: Pitch information (step, octave, accidental) • T_{duration}: Symbolic duration (quarter note, half note, with optional dots…) • $T_{\text{is_grace}}$: Flag indicating if it is a grace note • $T_{\text{is_tied}}$: Flag indicating if tied to a future note
`Rest`	Single rest	• T_{voice}: Voice index • T_{duration}: Symbolic duration • $T_{\text{full_measure}}$: Flag indicating a full-measure rest
`ChordStart`	Beginning of a chord	• T_{voice}: Voice index • T_{duration}: Symbolic duration • $T_{\text{is_grace}}$: Flag indicating if it is a grace chord
`ChordEnd`	End of a chord	• T_{voice}: Voice index
`TupletStart`	Beginning of a tuplet	• T_{voice}: Voice index • $T_{\text{base_unit}}$: Base symbolic unit • $T_{\text{n_normal}}$: Number of base units it should normally take (outside context) • $T_{\text{n_actual}}$: Number of base units it actually consists of (inside context) e.g. *for a tuplet of 3:2 eighth notes, those last three attributes are respectively "eighth", 2 and 3.* We also use the normal and actual duration of the tuplet: • $T_{\text{normal_duration}} = T_{\text{base_unit}} \times T_{\text{n_normal}}$ • $T_{\text{actual_duration}} = T_{\text{base_unit}} \times T_{\text{n_actual}}$
`TupletEnd`	End of a tuplet	• T_{voice}: Voice index
`EndOfScore`	End of the score	None

Table 1. Token types and their attributes

element. We define a set of rules that scores must follow to be considered valid. They are checked against a linearization of the score into a sequence of *tokens*, representing different musical elements. This sequence is parsed by a state machine that maintains a state about the partially analyzed token sequence, and checks the validity of each new token according to our ruleset. We first detail how we tokenize the score, then the implementation of our state machine, and finally the rules that define allowed transitions.

3.2.1 Tokenization of a Score

We define several token types with attributes that represent various musical elements, *e.g*. notes, rests, tuplets, bars, etc. In this work, we focus on encoding elements relative to the rhythmic organization of the score. However, note that this tokenization is generic and could be extended to include other elements such as key signature and clef changes, *etc*. All token types and their attributes are detailed in Table 1.

The score is tokenized sequentially, going through musical symbols in a chronological order. To avoid issues with the boundaries of repeated sections (with tied notes for example), we unfold all repetitions before tokenizing the score. The order of simultaneous notes and rests does not have any impact on the ruleset we use. Simultaneous notes in the same voice are wrapped in between **`ChordStart`** and **`ChordEnd`** tokens. Similarly, in-tuplet notes and rests are wrapped in between **`TupletStart`** and **`TupletEnd`** tokens. An example of such tokenization is given in Fig. 3.

3.2.2 State Machine

The next step consists in validating or invalidating a sequence of tokens according to a set of rules. For this purpose, we use a state machine that tracks the state of the partially parsed sequence. We call this state the *score state*, notated S, containing two attributes:

- S_{voices} – List of *voice states*, describing the current completion of each voice.
- S_{tied} – Set containing all currently tied notes.

We note S_0 a special state representing the initial state of the machine, which will only be used later on to ensure that a score starts with a **`Bar`** token.

A voice state v has the following attributes:

- v_{duration} – Duration of the voice, in quarter notes, derived from the current time signature.
- v_{position} – Current position in the voice.
- v_{chord} – Either a *chord state*, when inside a chord (*i.e*. after a **`ChordStart`** token), or $\varnothing$ otherwise.
- v_{tuplets} – Stack of *tuplet states*, representing potential tuplets in construction, with the most nested one at the top of the stack. It is $\varnothing$ when there are no ongoing tuplets. The function top retrieves the top element of the stack, *i.e*. the most nested tuplet.

A chord state C has the following attributes:

- C_{duration} – Symbolic duration of the chord.
- C_{notes} – List of notes in the chord.

A tuplet state K has the following attributes:

- K_{position} – Current position in the tuplet (relative to the tuplet inner metric).
- $K_{\text{unit_duration}}$ – Base unit duration of the tuplet, *e.g*. a eighth note in a 3:2 tuplet of eighth notes.
- $K_{\text{n_normal}}$ – Number of units that the tuplet should normally take. In the example above it would be 2.

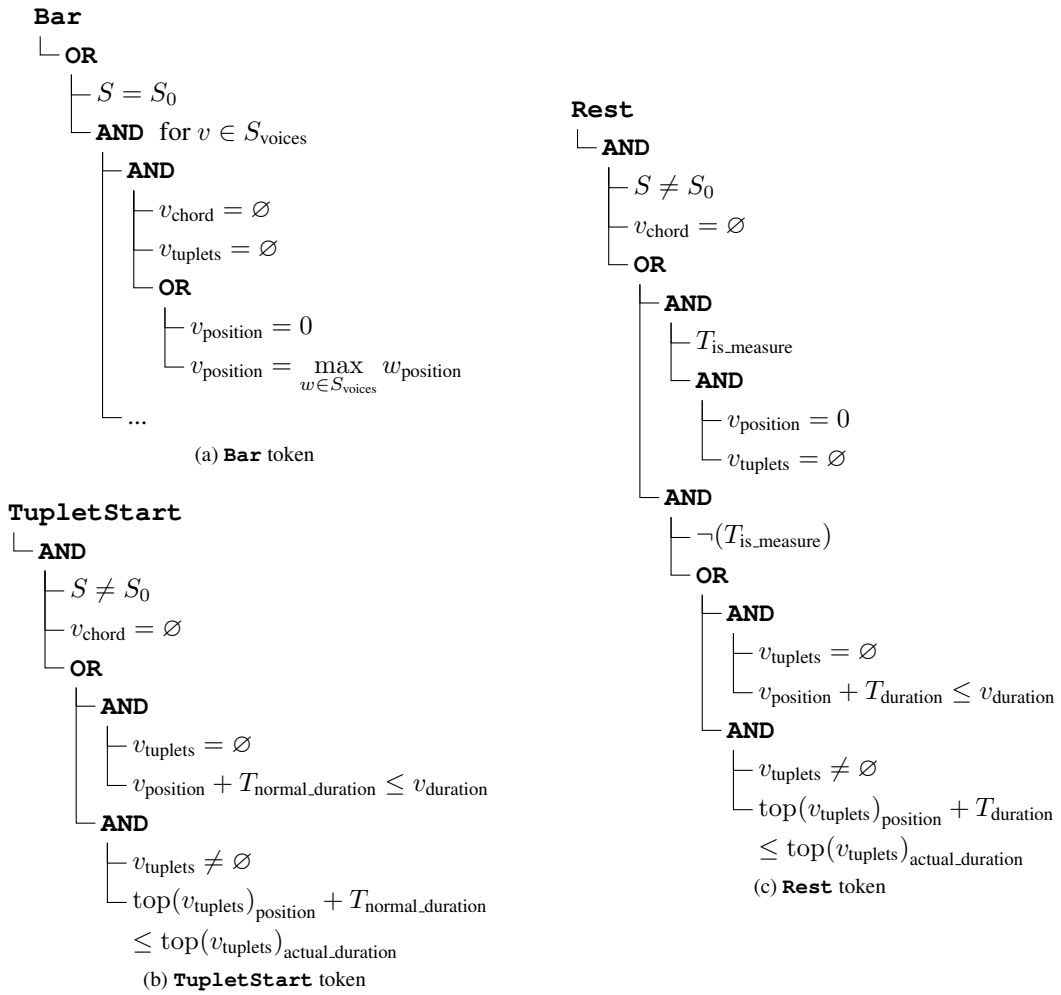


Figure 4. Guard condition examples for a token T for a state S, notating $v = S_{\text{voices}}[T_{\text{voice}}]$

- $K_{\text{n_actual}}$ – Number of units that the tuplet actually contains. In the example above it would be 3.
- $K_{\text{normal_duration}} = K_{\text{n_normal}} \times K_{\text{unit_duration}}$
- $K_{\text{actual_duration}} = K_{\text{n_actual}} \times K_{\text{unit_duration}}$

3.2.3 Guards of Transitions

Starting from the initial state S_0, we read the token sequence incrementally. For each new token, we ensure that the transition is valid before updating the state, by checking a verification condition called a *guard*. Each guard is token-specific and ensures that the current state can be updated by the new token through a valid transition. We detail some guards in Fig. 4, while an exhaustive list is available in Appendix B. If a guard condition is false at some point when parsing a sequence, this means that the score does not respect one of our rules, which raises an error.

For example, a **Bar** token (Fig. 4a) is allowed either at the very beginning of the sequence ($S = S_0$), or if all voices are complete, *i.e.* all chords have ended, all tuplets have ended, and all voices are synchronized [5] or empty.

A **Rest** token (Fig. 4c) is *not* allowed to come first in the sequence ($S \neq S_0$) and it cannot occur when a chord is in construction in the voice. To be valid, it must additionally be either a full-measure rest alone in the voice, or a regular rest which does not exceed the remaining duration of the voice – or tuplet if there is an ongoing tuplet in the voice.

Similarly, a **TupletStart** token (Fig. 4b) is *not* allowed as the first token of the sequence and cannot occur if there is a chord in construction in the voice. In addition, the tuplet's normal duration must not exceed the remaining duration of the voice – or outer tuplet if it is a nested tuplet.

Those guards could be extended to enforce stricter rules, or made optional to be more lenient with edge cases.

3.2.4 State Update

If the guard passes, then we can update the state of the machine. The current state is modified based on the new token and its attributes. These updates are generally straightforward. For example, **Bar** resets all voice positions to 0 and sets all voice durations according to its time signature. **Rest** advances the position of its voice by its duration. **TupletStart** pushes a new tuplet state onto the tuplets stack of the voice, that will be popped out when a **TupletEnd** occurs on that same voice.

[5] We recall that we do not consider incomplete voices as they have been filled with rests to fill the gaps.

4. IMPLEMENTATION AND APPLICATION

We assess the effectiveness of our error detection system by applying our syntactic checks on the MusicXML piano scores of the ASAP dataset [1].

4.1 Implementation Details

Our implementation is publicly available [6] .

4.1.1 Individual Errors

For individual errors, we parse the MusicXML file and compare the different durations as explained in Section 3.1.

4.1.2 Tokenization

We first parse the MusicXML files using Partitura [16], which keep the timeline untouched, meaning that overlapping notes in MusicXML still overlap in Partitura's internal representation. We then iterate chronologically over all elements on the timeline, creating tokens for supported ones.

Using Partitura's representation as input for our tokenizer allows the contextual error detection method to support not only MusicXML, but also all other score formats supported by Partitura. It could even be extended to unsupported formats by implementing the tokenization logic for those.

4.1.3 State Machine

We implement the states and guarded transitions introduced in Section 3.2. While nested tuplets are allowed in our framework, we chose not to implement them as they should be extremely rare, and the ones in the ASAP dataset are mainly due to encoding errors anyway. We also dropped support for grace chords, as Partitura currently does not support them. Instead, they are parsed as sequential grace notes. This has no practical impact on our method.

4.2 Results on ASAP Dataset

ASAP [1] contains 235 MusicXML scores, aligned with performances. However, those scores were not written by professionals, and therefore include numerous mistakes and notational errors. We report a summary of the proportion of the dataset impacted by both types of errors in Table 2.

4.2.1 Check of Individual Errors

Among the 235 scores in ASAP, 65 are flagged for inconsistency between theoretical and actual duration of notes, representing around 28% of the dataset. Those errors are present in 692 measures, or 1.9% of the total number of measures in the dataset. Most erroneous scores contain only a few invalid measures, with the median number of bad measures per score being 3. Only 17 scores have more than 5% of measures containing this type of errors, and 5 more than 30%.

Figures 5 and 6 show some representative errors detected with this method. Most of them are either rounding errors due to a bad division value in the MusicXML, or invisible notes with unusual durations included solely for MIDI playback.

[6] https://github.com/leleogere/detecting-notational-errors-in-digital-music-scores

```
<measure>
  <attributes>
    <divisions>480</divisions>
    ...
  </attributes>
  ...
  <note>
    <duration>34</duration>
    <type>32nd</type>
    <time-modification>
      <actual-notes>14</actual-notes>
      <normal-notes>8</normal-notes>
    </time-modification>
    ...
  </note>
</measure>
```

Figure 5. Example of tuplet rounding error in measure 94 of Beethoven's Sonata No. 17, op. 32, 2nd movement. The theoretical duration of a 32nd note in a 14:8 tuplet should be $\frac{1}{8} \times \frac{8}{14} \times 480 = \frac{240}{7} \times 480 \approx 34.3 \neq 34$.

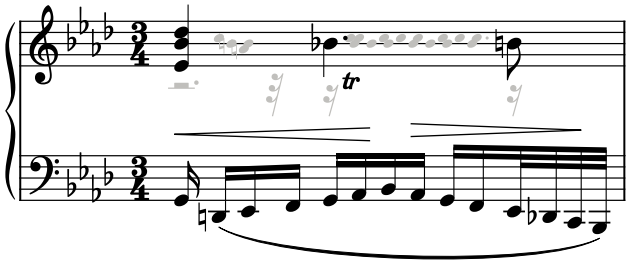


Figure 6. Example of invisible notes (in gray) of arbitrary duration resulting in a non-decomposable `<backup>` tag in measure 55 of the 1st movement of Beethoven's Sonata No. 31. Those notes were likely added for MIDI playback of the trill, and mess up with the MusicXML timeline.

4.2.2 Check of Contextual Errors

Because our state machine only deals with theoretical durations, and not with MusicXML timeline durations, individual errors will trigger contextual errors too as voices will be inconsistently filled. Therefore, we only run the contextual checks on the 170 correct scores left after the first stage, resulting in 35 incorrect scores:

- One had two parts, which we did not implement as a piano score should only have one part (containing multiple staves, but still one part).
- Two were only partially parsed by Partitura, which had trouble detecting some tuplet durations, which our tokenization procedure relies on. Those parsing difficulties likely point towards a notational error.
- The 32 remaining scores had been flagged as invalid by our set of rules.

Exploring those invalid scores, we successfully detected the following errors:

- **Measure overflow:** a voice is too long for the current time signature. This is generally caused by invisible rests (Fig. 7) or grace notes encoded as cue notes, which then have a non-zero duration (Fig. 8).
- **Same-voice notes overlap:** a note starts before the end of a former note in the same voice. While not

	Scores w/ errors	% scores w/ errors	Bars w/ errors	% of bars w/ errors	Median bars w/ errors in flagged score
Individual errors	65	$\frac{65}{235} = 28\%$	692	$\frac{692}{36496} = 1.9\%$	3
Contextual errors	35	$\frac{35}{170} = 21\%$	165	$\frac{165}{24648^*} = 0.7\%$	2

Table 2. Repartition of errors in ASAP dataset. The * indicates that the number of measures has been computed with unfolded scores, excluding the three incorrectly parsed MusicXML files mentioned in Section 4.2.2.

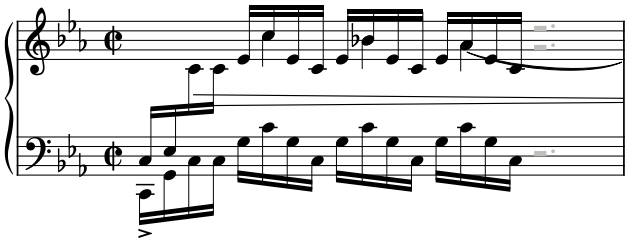


Figure 7. Measure overflow due to invisible rests (in gray) in measure 7 of Chopin's Etude Op. 25 No. 12.

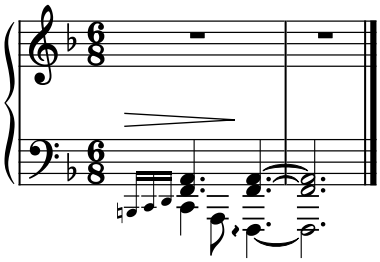


Figure 8. Measure overflow due to cue notes that should be grace notes, bar 97 of Liszt's Transcendental Etude No. 3.

directly detected as an overlap, this will trigger a measure overflow as our state machine does not have a direct notion of position; instead, it adds notes in the voice, eventually resulting in an overflow. For instance, in Fig. 2, the green voice is entirely filled by the first dotted quarter note, then the dotted eighth note coming next triggers an overflow as there is no more space in the measure.

- **Duplicate notes in chord:** as ASAP is a piano dataset, we forbid duplicated notes in a chord, as in Fig. 9.
- **Different durations in chord:** no such error were detected in ASAP. [7]

Note that the measure-overflowing mechanism also detects some occurrences of free played sections, such as in measures 232 to 234 of 1st movement of Beethoven's Piano Sonata No. 3 (Fig. 10). Those are considered as invalid by our model, as they go far beyond the nominal duration of the measure. These instances could be considered acceptable, and therefore, flagging them constitutes a false positive. This could be solved by allowing voices that are longer than the nominal measure duration, *i.e.*, making the overflowing rule optional. However, we argue this case (and other similar ones) is debatable: these small notes are

[7] In practice, our implementation does not find any such error as they are already fixed by Partitura's MusicXML import. Nonetheless, we confirmed that no such error exists in ASAP by directly parsing the MusicXML files.

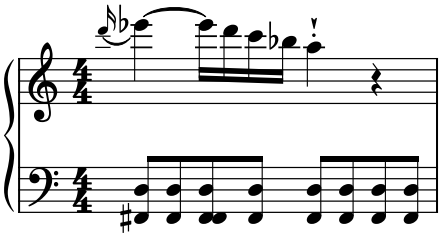


Figure 9. Duplicated note in chord in measure 97 of the 1st movement of Beethoven's Piano Sonata No. 21.

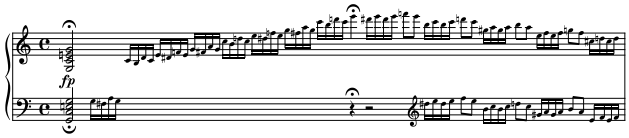


Figure 10. Free-played section in measure 232 of the 1st movement of Beethoven's Sonata No. 3. The actual duration of the measure is far longer than its nominal duration.

encoded as cue notes, which are supposed to be unplayed notes solely present to inform the performer about some other instrument's melody, not a free-play indication.

4.2.3 Fixing Scores

After detecting scores with notational errors, we have manually fixed some and pushed them to the ASAP repository. [8] Two of them contained tuplet rounding errors, which were fixed by computing an adequate value for the `<divisions>` tag, permitting an exact representation of all durations present in the score. All `<duration>` tags were updated according to this new value. Two scores were missing tuplet start tags, which would cause measure overflow issues later on. They were fixed by adding them back. Two scores were using incorrect tuplets, resulting in time-shifting and measure overflowing. One score contained extra invisible rests in an already-full measure, so we removed them. Finally, four other scores have also been fixed for other unrelated issues (extra staves, notes wrongly marked as invisible, poor notation choices).

5. DISCUSSION

In this article, we introduce a two-step method for the detection of notation error in digital music scores. The first step, specifically tailored for MusicXML, consists of ensuring consistency between different redundant duration information in MusicXML elements, in order to detect individual

[8] `https://github.com/fosfrancesco/asap-dataset`, only available on the `develop` branch as of October 2025.

note errors. The second step, format-agnostic, relies on an implemented tokenizer which linearizes the score into a sequence of tokens. This sequence is then checked by a state machine, whose role is to test the validity of each token within the musical surrounding context, in order to detect contextual errors.

Combining those two steps, our method is able to detect errors in 100 of the 235 scores of the ASAP dataset, or around 42%, confirming the fact that these scores were not written by professionals, and highlighting the interest of such detection methods. This allowed us to fix multiple scores in the dataset, enhancing its global quality. Scores that were manually fixed have been backported to the ASAP repository, for others to benefit from our work, even though there is still a long way to go to get a clean dataset.

Our method and implementation are modular and could be easily extended with more rules encoding various score quality aspects [15], to identify other types of errors. For instance, we might want to ensure that a score respects an specific instrument's tessitura, or make sure that no tie crosses a key signature or clef change – while not strictly forbidden, it is confusing to read. Hence, this model might serve as a base to build a more complete score validator to assess the quality of both human-produced and computer-generated music scores.

The main limitation of our approach is that while we detect errors, we do not automatically fix them. It is possible to suggest fixes for some error types, such as proposing a more suitable `<duration>` tag to correct duration inconsistencies. However, this still requires manual intervention and some expertise to check that the fix is suitable. Indeed, to fix rhythm/time consistency errors on notes and rests, one would need to choose which information becomes authoritative: the theoretical duration computed from the symbol or the timeline duration. The `<forward>`/`<backup>` tags, introduced to keep up with alignment in case of rounding errors, are even more difficult to deal with. Conversions between software can introduce invisible rests that are difficult to clean up automatically. Contextual errors are especially hard to automatically fix, because they require understanding the intention of the composer to sort out the cause of the error (*e.g.*, why is a measure overflowing?) and some expertise to know how to correct it while following the conventions of music notation. We observed that mistakes often tend to be very specific to one score, as two individuals have different ways of transcribing a piece. While a fully automated correction pipeline might be out of reach, our work remains useful to flag issues in scores. It then remains up to the user to inspect the problematic measures and determine the most appropriate fixes, *e.g.*, by comparing to references of that score in other published editions if they exist.

6. REFERENCES

[1] F. Foscarin, A. McLeod, P. Rigaux, F. Jacquemard, and M. Sakai, "ASAP: a dataset of aligned scores and performances for piano transcription," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2020, pp. 534–541.

[2] "Basic Notated Music | Humdrum." [Online]. Available: https://www.humdrum.org/rep/kern/

[3] "abc | home." [Online]. Available: https://abcnotation.com/

[4] M. Good, "MusicXML for Notation and Analysis," in *The Virtual Score, Volume 12: Representation, Retrieval, Restoration*. The MIT Press, 2001.

[5] "MusicXML for Exchanging Digital Sheet Music." [Online]. Available: https://www.musicxml.com/

[6] P. Roland, "The music encoding initiative (MEI)," in *Proceedings of the First International Conference on Musical Applications Using XML*, vol. 1060. Citeseer, 2002, pp. 55–59.

[7] "Music Encoding Initiative." [Online]. Available: https://music-encoding.org/

[8] G. Read, *Music Notation: A Manual of Modern Practice*, 2nd ed. Crescendo Publishers, 1979.

[9] E. Gould, *Behind Bars: The Definitive Guide to Music Notation*. Faber Music Ltd, 2011.

[10] W. Brodsky, Y. Kessler, B.-S. Rubinstein, J. Ginsborg, and A. Henik, "The mental representation of music notation: notational audiation," *Journal of Experimental Psychology: Human Perception and Performance*, vol. 34, no. 2, pp. 427–445, 2008.

[11] N. Nápoles, G. Vigliensoni, and I. Fujinaga, "Encoding matters," in *Proceedings of the 5th International Conference on Digital Libraries for Musicology (DLfM)*. Association for Computing Machinery, 2018, p. 69–73.

[12] S. Mohammed, L. Budach, M. Feuerpfeil, N. Ihde, A. Nathansen, N. Noack, H. Patzlaff, F. Naumann, and H. Harmouch, "The effects of data quality on machine learning performance on tabular data," *Information Systems*, vol. 132, p. 102549, 2025.

[13] D. Fiala, P. Rigaux, A. Tacaille, V. Thion, and G. Members, "Data Quality Rules for Digital Score Libraries," IRISA, Université de Rennes, Research Report, 2018.

[14] F. Foscarin, D. Fiala, F. Jacquemard, P. Rigaux, and V. Thion, "Gioqoso, an online Quality Assessment Tool for Music Notation," in *Proceedings of the International Conference on Technologies for Music Notation and Representation (TENOR)*. Concordia University, 2018.

[15] F. Foscarin, P. Rigaux, and V. Thion, "Data Quality Assessment in Digital Score Libraries. the Gioqoso Project," *International Journal on Digital Libraries*, vol. 22, no. 2, pp. 159–173, 2021.

[16] C. Cancino-Chacón, S. Peter, E. Karystinaios, F. Foscarin, M. Grachten, and G. Widmer, "Partitura: A Python Package for Symbolic Music," in *Proceedings of the Music Encoding Conference (MEC)*, 2022.

A. EXAMPLE OF A MUSICXML CONTAINING NOTATIONAL ERRORS

The XML in Fig. 11 is the source of the examples from Fig. 1. There are some duration inconsistencies in notes, as well as unusual <backup>/<forward> tags crossing the measure boundaries. Different pieces of software deal with those differently, resulting in rendering differences. This illustrates how the MusicXML specification does not enforce strict respect of musical notation, and allows for ambiguous interpretation when engraving.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE score-partwise PUBLIC "-//Recordare//DTD MusicXML 4.0 Partwise//EN"
↪  "http://www.musicxml.org/dtds/partwise.dtd">
<score-partwise version="4.0">
  <part-list>
    <score-part id="P1">
      <part-name>Piano</part-name>
      </score-part>
    </part-list>
  <part id="P1">
    <measure number="1">
      <attributes>
        <!-- In this file, a quarter note has duration 2 -->
        <divisions>2</divisions>
        <time>
          <beats>2</beats>
          <beat-type>4</beat-type>
          </time>
        <clef>
          <sign>G</sign>
          <line>2</line>
          </clef>
        </attributes>
      <note>
        <pitch><step>C</step><octave>4</octave></pitch>
        <voice>1</voice>
        <duration>4</duration>
        <type>half</type>
        </note>
      </measure>
    <measure number="2">
      <note>
        <pitch><step>E</step><octave>4</octave></pitch>
        <voice>1</voice>
        <!-- /!\ Half note with a duration of 2 instead of 4 -->
        <duration>2</duration>
        <type>half</type>
        </note>
      <note>
        <pitch><step>G</step><octave>4</octave></pitch>
        <voice>1</voice>
        <!-- /!\ Same as above -->
        <duration>2</duration>
        <type>half</type>
        </note>
      <backup>
        <!-- /!\ How to deal with this backup of duration 6 when we are only 4 divisions into the
        ↪  measure? -->
        <duration>6</duration>
        </backup>
      <note>
        <pitch><step>B</step><octave>4</octave></pitch>
        <voice>2</voice>
        <duration>2</duration>
        <type>quarter</type>
        </note>
      <backup>
        <!-- /!\ again a backup with a weird duration -->
        <duration>6</duration>
        </backup>
      <note>
        <pitch><step>D</step><octave>5</octave></pitch>
        <voice>3</voice>
        <duration>2</duration>
        <type>quarter</type>
        </note>
      </measure>
    </part>
  </score-partwise>
```

Figure 11. XML source of Fig. 1.

B. GUARD CONDITIONS

We detail below the guard conditions for the token types that are not introduced in Section 3.2. In each figure, we test whether the newly read token T allows for a valid transition while the state machine is in the state S.

`Note`

- **AND**
  - $S \neq S_0$
  - **OR**
    - **AND**
      - $v_{\text{chord}} \neq \varnothing$
      - $T_{\text{duration}} = (v_{\text{chord}})_{\text{duration}}$
      - $T_{\text{is_grace}} = (v_{\text{chord}})_{\text{is_grace}}$
    - **AND**
      - $v_{\text{chord}} = \varnothing$
      - **OR**
        - **AND**
          - $T_{\text{is_grace}}$
          - $v_{\text{position}} < v_{\text{duration}}$
        - **AND**
          - $\neg\,(T_{\text{is_grace}})$
          - **OR**
            - **AND**
              - $v_{\text{tuplets}} = \varnothing$
              - $v_{\text{position}} + T_{\text{duration}} \leq v_{\text{duration}}$
            - **AND**
              - $v_{\text{tuplets}} \neq \varnothing$
              - $\text{top}(v_{\text{tuplets}})_{\text{position}} + T_{\text{duration}} \leq \text{top}(v_{\text{tuplets}})_{\text{actual_duration}}$

Figure 12. Guard condition for a **`Note`** token

`EndOfScore`

- **OR**
  - $S = S_0$
  - **AND**
    - **AND** for v in S_{voices}
      - **AND**
        - $v_{\text{chord}} = \varnothing$
        - $v_{\text{tuplets}} = \varnothing$
        - **OR**
          - $v_{\text{position}} = 0$
          - $v_{\text{position}} = \max_{w \in S_{\text{voices}}} w_{\text{position}}$
      - ...
    - $S_{\text{tied}} = \varnothing$

Figure 13. Guard condition for an **`EndOfScore`** token

`ChordStart`

- **AND**
  - $S \neq S_0$
  - $v_{\text{chord}} = \varnothing$
  - **OR**
    - **AND**
      - $T_{\text{is_grace}}$
      - $v_{\text{position}} < v_{\text{duration}}$
    - **AND**
      - $\neg\,(T_{\text{is_grace}})$
      - **OR**
        - **AND**
          - $v_{\text{tuplets}} = \varnothing$
          - $v_{\text{position}} + T_{\text{duration}} \leq v_{\text{duration}}$
        - **AND**
          - $v_{\text{tuplets}} \neq \varnothing$
          - $\text{top}(v_{\text{tuplets}})_{\text{position}} + T_{\text{duration}} \leq \text{top}(v_{\text{tuplets}})_{\text{actual_duration}}$

Figure 14. Guard condition for a **`ChordStart`** token

`ChordEnd`

- **AND**
  - $v_{\text{chord}} \neq \varnothing$
  - $\left|(v_{\text{chord}})_{\text{num_notes}}\right| \geq 2$

Figure 15. Guard condition for a **`ChordEnd`** token

`TupletEnd`

- **AND**
  - $v_{\text{tuplet}} \neq \varnothing$
  - $\text{top}(v_{\text{tuplets}})_{\text{position}} = \text{top}(v_{\text{tuplets}})_{\text{duration}}$

Figure 16. Guard condition for a **`TupletEnd`** token

MOZ'LIB & PWFORMAX : REWIRING CAC HERITAGE

Julien Vincenot

PhD (defended 2024)
Department of Music
Harvard University
mail@julienvincenot.com

Örjan Sandred

Professor of Composition
Desautels Faculty of Music
University of Manitoba
sandred@umanitoba.ca

Juan S. Vassallo

PhD (defended 2025)
The Grieg Academy
University of Bergen
personal@juanvassallo.com

ABSTRACT

This paper surveys ten years of development on *MOZ'Lib*, an open-source package for Max built around the *bach* ecosystem for computer-aided composition (CAC). Originally a set of pedagogical, entry-level modules for beginners of CAC, it features now a more advanced framework, *PWforMax*, which allows musicians to explore the rich heritage of Lisp-based CAC within Max. The package provides a variety of paths for teaching music creation, experimentation, but also to maintain cultural continuity in digital creativity. After presenting MOZ'Lib's foundations and knowledge base — rooted in the PatchWork family of CAC environments — the discussion focuses on rule-based generative systems. The paper also reflects on the place of artificial intelligence in computer-aided composition practices today, and advocates for a diversity of AI paradigms in computer music. Two constraints solvers, *PWConstraints* and *Cluster-Engine*, whose interfaces have been largely reimagined for Max, are examined in detail. In the last sections, authors discuss some design choices made over the years — in particular to bridge the conceptual and technical gaps between Lisp-based CAC and Max — as well as ongoing and future developments.

1. INTRODUCTION

MOZ'Lib (or simply MOZ) is a Max package developed by Julien Vincenot since 2015. It can be considered a non-official satellite of the *bach* ecosystem[1], created by composers Andrea Agostini and Daniele Ghisi in 2010. Originally not meant for public release, MOZ'Lib has been completely refactored in its current form and fully documented in English as part of Vincenot's PhD research in composition at Harvard University, and freely distributed since 2020[2]. The current version is compatible with all macOS environments (version 10.12 and later, Intel and Silicon), as well as Windows (version 10 and later[3]), thanks to the help of Matteo Marson and Juan S. Vassallo.

Both a beginner-friendly option, and a rather advanced, specialized tool, MOZ'Lib inherits the philosophy of previous computer-aided composition (or CAC) environments : first and foremost, the PatchWork family (including OpenMusic and PWGL), initiated by Mikael Laurson, and of course *bach* itself. Although it is clearly a part of its core, the package's main goal is not to offer a comprehensive catalogue of pre-existing techniques — a form of "canonical" culture of algorithmic music, which can be intimidating or overly prescriptive to newcomers. Instead it hopes to foster a broader, exploratory and holistic approach, focusing on a flexibility of practices (plural), multiplicity of representations, expressivity and extensibility, but also the preservation or continuation of past practices.

Today this package can be divided into two distinct halves, with significantly different purposes : the original MOZ'Lib itself, a set of pedagogical tools designed to explore music writing, creation and computer programming ; and its expert counterpart *PWforMax*[4], which allows to generate and run Lisp code in Max, and thus to work with codebases inherited from the PW family. Among various areas of interest, PWforMax gives access to a selection of rule-based generative systems (i.e. music constraint solvers), that offer uniquely expressive possibilities. As such, it represents fertile ground for anyone looking to experiment with alternate AI directions in music today.

2. CAC FOR AGES 9 TO 99

The original set of pedagogical MOZ modules was commissioned by the *Ariane#* project and Conservatory of Montbéliard (France). It resulted from lengthy

[1] cf. Agostini & Ghisi, 2015

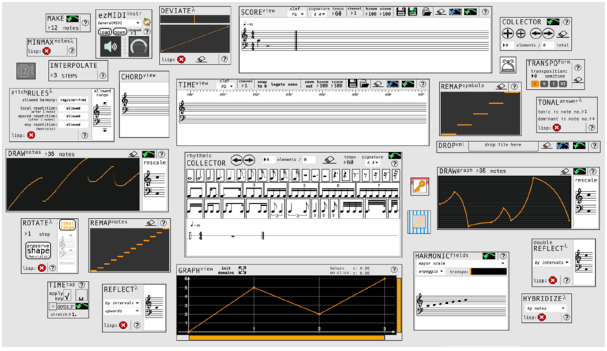
[2] MOZ'Lib is distributed under the GNU General Public License version 3 (GPLv3) and regularly updated.
cf. github.com/JulienVincenot/MOZLib/

[3] Currently, Windows users are required to perform a list of operations manually, including the installation of WSL (Windows Subsystem for Linux) in order to use Lisp functionalities in Max. An alternate solution is being examined, in collaboration with Paulo Raposo, using the Chocolatey package manager instead of WSL to handle SBCL.

[4] The name PWforMax was chosen as a homage to the historic contributions made by Mikael Laurson with PatchWork then PWGL, and particularly his pioneering work on music constraints. The tools are truly missed, and MOZ'Lib is a modest attempt to recapture some of their unique capabilities.

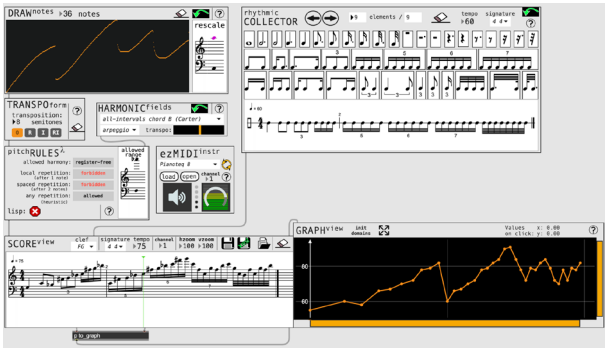
discussions with composer and pedagogist Gaja Maffezzoli, who specialized early in combining music creation and pedagogy for children, first in Florence's Tempo Reale center, then in Montbéliard. Its first goal was to introduce young musicians in training to music theory and instrumental composition with the help of digital tools. The library had to be accessible, appealing, and most importantly, highly responsive. PWGL, which was already endangered at the time[5], was initially considered to develop these tools, but *bach* was eventually chosen as the most promising option.

Instead of proposing a set of rigid, single-purpose tools (typically non-editable Max apps), another choice made early on was to build on the great flexibility of both Max and *bach* to propose a sort of simplified environment, combining ideas like a chain of plug-ins in a DAW. The solution was inspired by the BEAP and Vizzie modules introduced in 2014 by Max 7 : a library of bpatchers, i.e. large boxes with embedded user interfaces, that could be connected to each other like regular Max objects to build more complex scenarios with limited programming knowledge.



To this day, the main library includes more than 30 modules to generate, transform, display and listen to notated music materials, that can be easily drag-and-dropped[6] into a new patch. Between modules, patch cords usually carry simplified (flattened) lists of pitches (or midicents), rhythm (represented as rational numbers), purely symbolic data or other parameters[7].

This almost "baby CAC" approach meant facilitating access to *bach* features for everyone, in a way similar to *cage*[8], trying to reduce as much as possible the distance between idea and result. Considering their inherent complexity and steep learning curve for beginners, a strong emphasis was put on all the typical "end-of-branch" modules, i.e. so-called MOZ Editors. First and foremost, CHORDview, SCOREview and TIMEview (all inspired from similar objects in PWGL and OM), allow to quickly obtain some score representation from a simple list of pitches and/or durations. GRAPHview can instantly display multiple tracks of numerical data (musical or non-musical) as 2D break-point functions[9]. MOZ Editors also accommodate more experimental modules such as METAview, which makes use of automatic patch scripting to combine metered and non-metered notations, a feature unavailable in *bach*. Beyond the intended audience of total beginners of CAC, many of these modules have proven more and more useful in a variety of situations, to accelerate workflow when testing ideas, to display motifs or formal schemes, to quickly control a result as notation or play it with a vst~ instrument.



[5] A sequel to the original PatchWork, created in 1986 by Mikael Laurson and developed with several collaborators at IRCAM, PWGL was developed since 2002 by Laurson together with Mika Kuuskankare and Vesa Norilo, later joined by Kilian Sprotte. First released in 2007, it remained officially in "beta" form until its final known version (RC19, build 342). Despite a dedicated community, PWGL began to wither in 2016. This decline may be attributed to multiple factors, including the lack of a timely 64-bit port, a refusal to open its source code (most likely for a lack of trust following OpenMusic's succession to PatchWork), difficult portability because of a rather unaffordable Lisp implementation (LispWorks), and eventually a lack of funding and institutional support. These challenges made PWGL unusable beyond macOS 10.14.6, and the official website quietly disappeared sometime in 2020. cf. archive.org for the missing www2.siba.fi/PWGL/

[6] In the Max toolbar, a dedicated menu gives access to the palette of MOZ modules (or snippets) arranged in different categories :

• Generators — modules to create simple music entities from scratch by drawing, recording or building up sequences with predetermined options (rhythm cells, harmonic fields, etc.)

• Editors — modules to display music entities according to various modes of representation : flat melodic profile or list of chords (inspired by PWGL's chord editor), sequencer-like chronometric notation (based on *bach.roll*), traditional metric notation (based on *bach.score*) or more abstract (break-point function or graph)

• Transformers — a selection of CAC techniques operating mostly on pitches — transposition, retrograde/inversion, mirroring, distortion, interpolation, non-linear remapping, rotation (including heuristic preservation of shape), basic constraints, etc.

• Utilities — in/out modules and smaller tools.

[7] Many of these modules were in fact inspired by Vincenot's personal experience and workflow in PWGL, re-actualized in *bach*. But unlike PatchWork and its successors, where the programming logic was so-called bottom-up or "demand-driven", MOZ'Lib inherits the same "data-driven" approach as Max and *bach* where data follows the tree of the patch from top to bottom. This logic is in fact quite unnatural to users familiar of previous, Lisp-based CAC environments. Trying to combine to two styles of interaction resulted in some debatable design choices like the addition of "callback" buttons to certain modules, forcing data to traverse the patch again and update the whole tree, for instance when a new module is introduced in the chain.

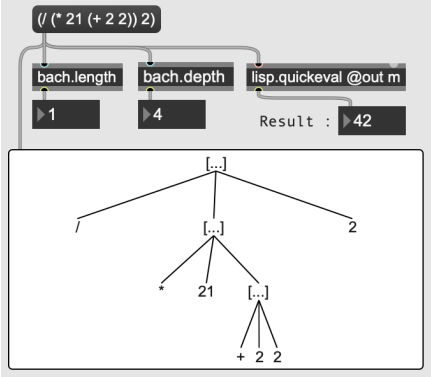
[8] This package was the first addition to *bach* by its creators, followed by *dada* and *ears*. It offers a collection of composition tools — all abstractions that demonstrate the possibilities of *bach* objects.

[9] In our view, this module is just as crucial as the notation modules —if not more so— because it encourages a sense of abstraction in musical processes (in particular for evaluating perceptible directionality) which can benefit both composition and creative analysis.

Among the early batch of pedagogical modules, MOZ Transformers quickly became the trickiest part of the project. One of Vincenot's first goals was to emulate certain PWGL tools by Jacopo Baboni Schilingi, who introduced him to CAC in 2007. Many of these were in fact rather complex Lisp programs, impossible to recreate using only Max or *bach* objects. This was the initial impulse to develop a system that could transplant techniques from the PatchWork family into Max.

3. *PWFORMAX* : RE-OPENING THE PARENTHESIS

The first prototype which would eventually become PWforMax was the result of a collective effort during the 2015 meeting of the international PRISMA research group[10], hosted at IRCAM, notably by Örjan Sandred, Hans Tutschku and Johannes Kretz. It relies on a simple mechanism : a piece of Lisp code (or s-expression) is initially produced in Max, then evaluated outside of Max by SBCL[11] via shell commands[12]. After the evaluation is completed, the result is returned back to Max with usually imperceptible latency, then further developed or displayed with regular *bach* interfaces.

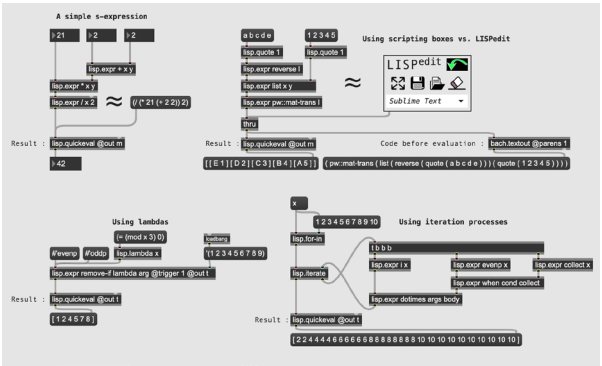


In the earliest versions, standard Max messages were used to generate the Lisp code dynamically, in combination with the *sprintf* object. But this approach proved unviable due to a number of issues : Max's hard limit on message length, difficulties handling nested parentheses, and the accumulation of new symbols that inevitably led to bloated RAM usage after a few hours. The solution came from a critical insight by Jean-Baptiste Barrière : the *bach* ecosystem relies on a custom data structure, the *llll* (or Lisp-like linked list), which is virtually indistinguishable from s-expressions, and has no inherent limitations on length or depth. Using *bach* objects to generate Lisp code on the fly made PWforMax fully stable and operational[13].

Originally, the feature allowing to use Lisp in Max was hidden in some of the original MOZ Transformers that were marked with a lambda symbol, such as Deviateλ or Hybridizeλ, based on the JBS-CMI library. The most advanced modules in this category, namely Rotateλ and PitchRulesλ, both made use of Laurson's PMC engine (a constraint solver which will be further discussed below), allowing to apply a combination of true/false and heuristic rules to pitch materials.

It was only after a few years testing these features in a variety of scenarios, that PWforMax was officially announced and fully documented as the advanced core behind MOZ. It is in fact this addition that justified distributing MOZ'Lib as a public package. For the overwhelming majority of the software industry, including most computer music specialists, Lisp is usually dismissed as a “dead tongue”, a relic from the past. Yet, over the years it became obvious that a dedicated, albeit narrow, community of users remained strongly attracted to its possibilities[14], as a way to modernize older (or even lost) CAC techniques, but also to apply them in real-time performances or installations.

Unlike the few other past and present Lisp implementations in Max[15], the initial design choice behind PWforMax — Max / *bach* handling code and SBCL evaluating it separately— has led to develop two complementary approaches for music Lisping, familiar to all users of PW(OM)GL : coding *and* patching.



[10] This particular PRISMA workshop, hosted at Ircam, was triggered by a hypothesis from Vincenot, after comparing performances in PWGL and OpenMusic (both developed with LispWorks) against SBCL. It turned out the two CAC environments came with strict limitations on heap size in recursive functions. This prevented users from performing heavier calculations, such as analyzing a large corpus of data, or generating longer sequences with a constraint engine. A (rather absurd) idea was proposed : to offload these more demanding calculations outside of PWGL or OM, letting SBCL run in the background when necessary. This was prototyped directly in Max by PRISMA members, and later adopted for MOZ’Lib.

[11] SBCL (Steel Bank Common Lisp) is one of the most popular Lisp implementations today. It is open-source, cross-platform, and offers great performances. cf. sbcl.org

[12] Terminal commands are supported non-officially in Max thanks to Jeremy Bernstein's *shell* object.
cf. github.com/jeremybernstein/shell

[13] For more technical details about the implementation of the system, cf. Vincenot, 2017

[14] As of today, the core audience for PWforMax comes primarily from the nebula of Lisp-based composition tools. This includes the PW family of course — PatchWork, OpenMusic and PWGL, which all share a similar design philosophy and culture. It also extends to independent projects like the proprietary environment Opusmodus. Every now and then, a newcomer who discovered modern CAC with *bach* gets also seduced by the “old ways” and starts to explore the unique possibilities of Lisp for composition, in particular constraints.

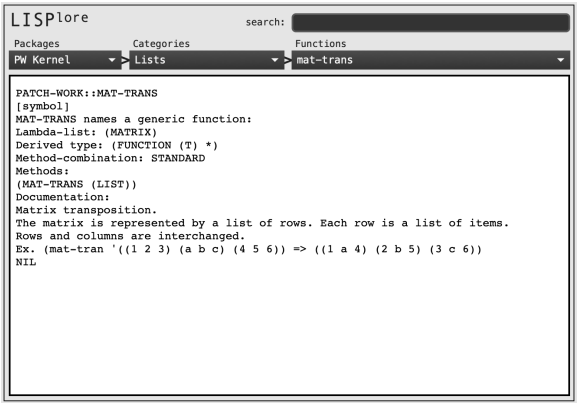
[15] Notable implementations are Brad Garton's *MaxLispj* based on ABCL (2011), Alex Graham's *Lisper* based on Clozure CL (2011-2013) and Iain Ducan's *SchemeForMax* based on s7 Scheme Lisp (2022). They all run directly in Max (unlike MOZ), and essentially rely on Max messages to interact.

The main entry point to evaluate Lisp code is the *lisp.quickeval* object. It receives an s-expression — either as a Max message or a *bach llll* — and automatically handles communication with SBCL. After evaluation, the result is read back into Max as a *bach llll* pointer by default. Users may also choose to have the output translated into text for immediate readability.

Beyond simple Max messages, "pure Lisping" in text form was facilitated by introducing *LISPedit*. The module can be linked with any favorite Lisp editor or IDE, enabling features like syntax highlighting, parenthesis matching, auto-indentation, and more. LISPedit can send a piece of code directly to evaluation, but it is mostly useful to prepare "code snippets" — a feature of lisp.quickeval which allows to include new function definitions in the environment. This is the easiest way to enrich the "kernel" of Common Lisp and PW functions that is provided with MOZ'Lib (more on this in the next section).

The other approach to generate Lisp code in MOZ relies on the visual programming paradigm itself, which reshaped the landscape of computer music in the mid-1980s, first with Miller Puckette's *Patcher*, and shortly after with Mikael Laurson's *PatchWork*. Despite the fact that, under the hood, these are fundamentally different programs (as will be further discussed below), PWforMax gladly takes advantage of the visual analogy between them. In practice, attempts to reproduce the behavior of a PWGL patch would often result in a Max equivalent with a very similar shape and vocabulary. This proved to be a great time-saver when porting older patches to the new environment. In order to streamline the whole process, a small suite of abstractions (called "Lisp-scripting boxes") was created to compose s-expressions (*lisp.expr* and *lisp.lambda*), to perform loops (*lisp.iterate*), help debugging with the Max window (*lisp.print*) or quote an ordinary Max list before it enters the Lisp domain (*lisp.quote* or *lisp.backquote*).

To facilitate as much as possible the documentation about Lisp directly within Max, a special helper tool was designed, LISPlore, which gives access to docstrings of all the PW "kernel" functions provided out-of-the-box with MOZ'Lib. This was also meant as a partial solution to the "black box" effect experienced by several users of the early prototypes. The knowledge base of PWforMax — a pre-compiled SBCL .core, cf. next section — being particularly opaque, it was indispensable to provide a simple way to browse among its most common and useful functions. In addition, the PWforMax Overview also provides a Lisp introduction pdf, as well as a few suggestions for accessible textbooks and online references, for all those interested in learning the language.



4. PATCHWORK FAMILY (AND FRIENDS…)

In order to program some of MOZ'Lib's pedagogical modules, it was necessary to access a range of functions from PWGL libraries[16]. These higher level CAC tools, however, did not rely only on the ANSI Common Lisp standard lexicon. They were also built with numerous "helper functions" specifically written for CAC purposes, many of which had been passed down from PatchWork into both OpenMusic and PWGL. Among them are some iconic examples such as *flat-once*, *x-append*, *posn-match*, *mat-trans*, etc., which remained virtually identical across all three environments. Most of these essential functions were eventually borrowed from OpenMusic's sources[17], simplified into vanilla Common Lisp, and included in the PWforMax kernel[18].

Hundreds of functions are readily available under the :PW package name, although most of them remain poorly documented in Max for lack of time or necessity. They are also rarely mentioned explicitly in PWforMax tutorials. The aforementioned LISPlore module addresses this gap : it allows to navigate some of the known packages included in the .core file via dropdown menus, organized into explicit subcategories. It also features a search function, allowing users to retrieve the

[16] When discovering a new CAC environment, it usually does not come as a "naked" tool. A few years after its release, PWGL already offered a couple dozen libraries, often passed hand to hand between composers, researchers, etc. Many users learned to use the environment — and in some cases, learned to compose — by experimenting with these higher-level tools, discussing with their authors, studying their code, etc. Most importantly, users built upon each others' work : porting the most recent tool in a chain often implies to carry its whole lineage. For reference, below is a public repository gathering over 50 libraries which were circulating in CAC circles until PWGL's development was discontinued.
cf. github.com/JulienVincenot/PWGL-community-library/
A similar page gathers libraries for OpenMusic.
cf. openmusic-project.github.io/libraries.html

[17] We would like to thank again the Music Representations team at IRCAM, in particular Jean Bresson, Gérard Assayag and Karim Haddad, for keeping these accessible and well documented over the years. This entire project would not have been possible otherwise.

[18] The "save-lisp-and-die" function in SBCL allows to save an entire Lisp environment, including declared packages, functions and variables, as a single binary file called a core. This .core file can be easily reloaded using the --core flag when calling SBCL from the terminal. The main advantages of this method is the portability of generated .core files (within a same architecture), and instant startup even with dozens of libraries preloaded.

docstring of any function declared in the environment—even those not listed in the menus.

Over the years, a number of libraries from OM and PWGL have been ported to MOZ'Lib using the same method, often in collaboration with their original creators. The procedure requires little effort, even for casual Lispers. It allows not only to preserve some previous work, programs and sketches in a more enduring environment like Max, but also to transform radically how users interact with them. The new real-time context calls for much more reactive types of interfaces (again, with minimal latency) and naturally broadens the range of artistic contexts in which to use these advanced techniques. This includes the whole world of audio processing, synthesis and analysis in Max (similarly to OpenMusic) but also video, lighting, sensors and networks, etc. It also brings functionalities that may seem trivial at first, yet prove transformative, such as sophisticated systems to handle presets as meta-parameters.

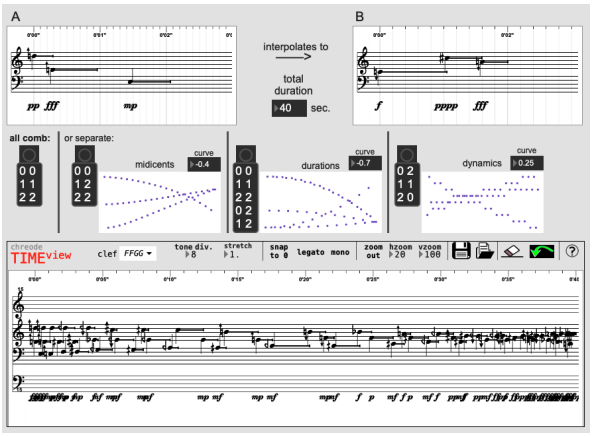
Until very recently, the only durable way to add a new library to PWforMax was for its creator to hardcode it directly into the MOZ'Lib package. As a result, Vincenot adopted a form of "curated" approach to libraries, highlighting different approaches to music thinking through CAC[19]. Some of them, labelled "guest libraries", are showcased as part of the PWforMax project, and receive special attention. Not only is their source code adapted to run in the new environment, but custom interfaces are also designed on top of key functions to better fit the Max context. They are also fully documented, with plenty of example patches available in the package overview. A more sophisticated and appealing UI aims to attract a broader audience, but also to make the underlying concepts of these tools more accessible to newcomers.

Guest libraries in PWforMax include so far :

• the reconstitution of two historical libraries, Jean-Baptiste Barrière's *Chréode* and Kaija Saariaho's *Transkaija*[20], both developed in the 1980s with CHANT/FORMES — a text-based CAC environment created at IRCAM ;

• Frédéric Voisin and Jacopo Baboni Schilingi's *Morphologie*[21] ; Baboni Schilingi's *CMI*[22] and *Profile*[23], all three originally created for PatchWork in the late-1990s, and later ported to OM and PWGL ;

• Baboni Schilingi's *JBS-Constraints*, created for PWGL in the mid-2000 (see more in the next section).



As a tool clearly oriented towards pedagogy, MOZ'Lib needed to find a good balance between reliable transmission of ideas — i.e. not constantly reinventing the wheel — and yet not trying to aim for exhaustivity, but rather focus on quality. The project is not only about the perpetuation of some abstract culture, but bringing forward again the embodied practices of musicians who developed those systems, grew with them, augmented them, and often tried to propose alternatives to canonical recipes of algorithmic composition[24].

PWforMax cannot pretend to contain all the "flavors" of Lisp-based CAC or algorithmic composition. In comparison, OpenMusic remains the largest repository of historical knowledge available. It has been actively maintained for decades at IRCAM and has the most vibrant user community still attached to the PW family. But after years as committed PWGL users, we now clearly favor the combination of Max and *bach* as the most modern CAC environment available today, in particular in terms of accessibility and pedagogy. Yet, no one can tell how this minuscule corner of computer music will evolve in the next decade. Losing PWGL has been a rather traumatic experience for a part of the CAC community, and it did

[19] Most of the libraries included so far were created by former teachers and friends from the PRISMA group, which already formed a non-negligible part of the PWGL community.

[20] This project was realized in collaboration with Barrière with early versions of MOZ. Both libraries were originally written in *Le Lisp*, a dialect now considered outdated. Reconstitutions required plenty of hypotheses from sources — hardly readable and unexecutable — compared with intermediary versions in PW(OM)GL, archived documents, and of course Barrière's own memories.

Chréode uses a palette of bpatchers to generate Lisp code dynamically, which the user must assemble following a given syntax. The final code is evaluated on the fly, and results can be observed as graphs or bach scores in real-time. cf. Barrière, 1984

Transkaija is a great example of the "debate" around music metamorphosis in early days of CAC. Saariaho takes a more subtle approach than the typical interpolation — i.e. regular sampling of a line between two values. It allows to transition across multiple parameters in parallel : chords, melodies, but also durations and dynamics. One can set explicitly which elements in one entity should point to in the other, thereby generating original scenarios. cf. Saariaho, 1987

[21] *Morphologie* is dedicated to analysis, classification, and reconstruction of numerical / symbolic musical sequences. It supports applications ranging from isolated transcriptions to large-scale corpus analysis, depending on how data is encoded. Designed to bridge physical and cognitive representations, it aims to serve both musicological and compositional purposes.
cf. Baboni Schilingi & Voisin, 1999.

[22] *JBS-CMI* is a collection of techniques for generating and manipulating musical structures, in particular matrices, complex permutations, groupings and segmentations.

[23] *JBS-Profile*, initially written in collaboration with Mikhail Malt, focuses on controlling the contour of any kind of parameter, including melodic profiles, with a variety of geometric transformations : perturbations, reflections, derivations / integrations and interpolations.

[24] In that vein, see a discussion around the idea of interpolation vs. musical metamorphosis. cf. Vincenot, 2016

not improve confidence in institutions to support such fragile projects in the long term. Two options are without a doubt better than one.

This reality reinforces the necessity to cultivate a form of *technodiversity*[25] within CAC approaches. Environments like OM or Max/*bach* (with the help of MOZ), cannot make the economy of a critical perspective on their own apparatus of preservation and passing of knowledge. For the same reasons, computer music in general, and CAC in particular, must also acknowledge what alternatives exist and have existed within the historic label of "artificial intelligence" and what they can offer to music creators. More than anything, we must avoid promoting a single, dominant paradigm of AI as much as possible.

5. NO-BRAINER : AI FLAVORS YOU CAN'T RESIST

In this bigger section we would like to give a panorama of generative constraint systems showcased in MOZ'Lib. We hope to communicate why we believe they remain a powerful tool for music creation and CAC today, in contrast with more recent trends of ML-driven generative paradigms. We will not go into too much technical depth for this comparison, which would go beyond the scope of this paper. Instead, we will focus on what we can actually speak about, which is interacting closely with algorithms to compose music.

For readers less familiar with the subject, it is worth recalling that rule-based approaches are historically rooted in (symbolic) AI research[26]. Unlike the most visible trends of AI today, they do not rely on large-scale data, but instead on logical properties and relations, combined with the knowledge (and craft) of humans working with them. This is particularly valuable in music, where massive, annotated datasets are rare or non-existent[27], but also for those interested in creative tools that go beyond the statistical reproduction of pre-existing materials. It can also become a vital alternative for creators concerned about the ethical, environmental, and cognitive footprints associated with current generative AI approaches.

In his 1996 thesis, Mikael Laurson presents his approach of music constraints in those terms : « When using a procedural programming language, such as C or Pascal, or a functional language, like Lisp or PW, the user has to solve a problem in a stepwise manner. […] Descriptive (declarative) languages, like Prolog, offer an alternative way to look at this problem: instead of trying to solve a problem step-by-step, the user describes a possible result with the help of a set of rules. It is then up to the language to find solutions that are coherent with the descriptions. This approach is probably more natural for individuals with a musical background.[28] » Constraints solvers like Laurson's use a backtracking search engine to explore possible solutions sequentially. At each step, the engine picks a value from a given domain, and retains the ones that satisfy the given rules. If a logical conflict arises, the system backtracks to earlier choices and keeps exploring alternative paths—much like a composer testing and refining ideas by hand. This recursive mechanism allows to efficiently eliminate invalid options, and gradually converge towards a satisfying outcome.

This is quite different from more “decisive”, traditional approaches to algorithmic composition. There, we usually focus on well-defined recipes designed —or repurposed— to generate musical results with certain properties and internal relations. A composer may choose, for instance, an implementation of *l-systems*, in order to generate certain typologies of rhythmic texture. On the opposite end of the spectrum, ML-driven (data-based) and constraint-driven (rule-based) offer a very similar mode of interaction to artists, who can focus on describing desired properties or relations they wish to find in a resulting entity, whether it is through code or natural language prompts.

Algorithms enabling such transfer from description to result often fade into the background, so to speak. They can almost be perceived as neutral or agnostic — a necessary illusion for the user, but an illusion nonetheless. Different constraint algorithms may have different purposes and limitations, due to architectures of varying degrees of complexity. And therefore they may have, potentially, very different “flavors” in usage. In the context of PWforMax, no matter what constraint engine is chosen, users need to express what they want to obtain either as truth values (true/false rules) or as numerical values (heuristic rules). In PWforMax this is done by way of Lisp code or graphically, with Lisp scripting boxes[29].

We believe these techniques constitute a great value not only as a specific tool to use, but as a process to live, and learn with. By that we mean, learning to understand what it is that we are looking for, learning to describe it, to create a conscious “transduction” between some abstract wish and the very concrete result of a music score — and keep interacting with that score, through multiple evaluations and refinements of the rules. When composing music with computers, the speed of interaction does matter, so that the flow of

[25] The concept was proposed by philosopher Yuk Hui : « Instead of a universal history describing one technology with various stages of development, we can step back for a moment and instead describe technological development as involving different *cosmotechnics*. I call this *technodiversity*. Here, we must revisit the question [...] in terms of systems of knowledge. Michel Foucault [...] understood them as ways of life — ways of sensing and ordering experience, producing in turn certain forms of knowledge. » cf. Dunker, 2020

[26] cf. Norvig, 1992

[27] By this we do not mean only datasets such as MIDI-based transcriptions (mostly relying on pitch and rhythm), but the whole spectrum of notated music to be performed by human instrumentalists. In recent years, we have often heard lamentations, among colleagues in computer music, about the lack of annotated data in particular — i.e. music paired with descriptive, musically meaningful labels — compared with the visual domain. Yet we see this not as a shortcoming, but as an opportunity.

[28] cf. Laurson, 1996, p.147

[29] For future updates, we hope to develop a system to help users generate rule code in Lisp based on some natural language descriptions, potentially with open-source, specialized SLMs ran locally. This could potentially improve the learning curve for Lisp learners, but also facilitate access to more casual users who may be reluctant to learn the language otherwise.

imagination is not interrupted. Yet, a user may need a few minutes, or even a few hours to write a complicated rule. On the other hand, an engine with well-written rules may be re-evaluated over and over, and produce results extremely fast, and reliably. Rules may also be re-used in very different contexts and projects, with a reasonable time of adaptation. This is clearly a strength when compared with the rather "slow" interaction loop one might encounter with well-known, general-purpose LLMs or prompt-based systems — typically "single use" results, often unusable as is, and with no guarantee of continuity beyond the context window. We would add that, in the context of music composition, mistakes allowed by purely heuristic systems can easily become unforgivable, especially in instrumental scores intended for musicians. Unlike ML-based approaches we have experimented with, constraints solvers remain a strong and expressive compromise between this ideal of music precision and extravagant exploration of the possible.

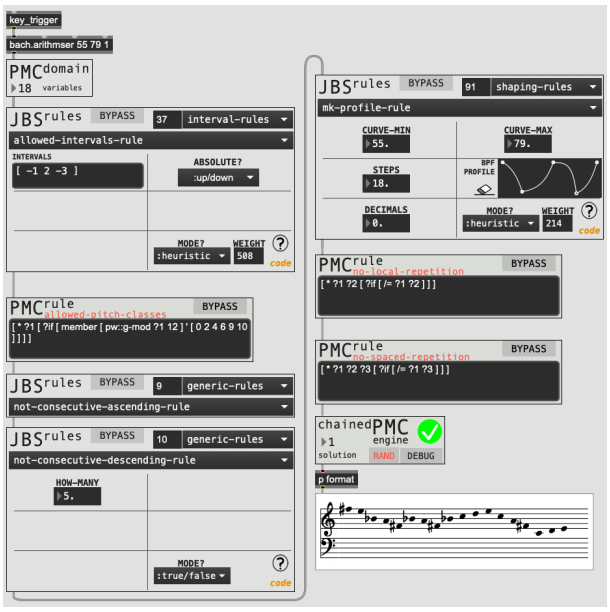
5.1. PMC engine — the polyvalent solver

Not a PWforMax "guest library" per se, *PWConstraints* was included in the MOZ'Lib kernel since the very beginning. It was originally created in 1995 by Mikael Laurson for PatchWork, then ported to PWGL[30]. The library revolves around the PMC engine, a Lisp function searching for potential solutions to a musical problem described by the user with a set of rules. Solutions are generally a sequence of predetermined size for a single parameter at once, i.e. either pitch, rhythm, or any other symbolic representation that the user can imagine. This makes it a robust and highly versatile implementation, which can also be applied to problems that are not explicitly musical.

In MOZ'Lib, the way of patching and interacting with the library has been extensively redesigned to fit the usual practices of Max users. Early in the development, it became clear that the typical "star-shaped", demand-driven layout of constraints patches in PW(OM)GL did not suit the top-down logic of Max — and went against the intuitive approach that was hoped for MOZ. For this reason, Sandred proposed the idea of the "chained" engine, first prototyped for the PMC then generalized with the Cluster-Engine. This implementation of PWConstraints revolves around a small set of bpatcher modules. The candidates (defined in *PMCdomain*) traverse a string of rule modules (*PMCrule*, *PMCheuristic-rule*, *JBSrules*) that act metaphorically like "filters", then reach the *PMCengine*[31] where the computation really takes place.

PMC rules, whether in true/false or heuristic form, are written using a dedicated syntax[32] based on Lisp, While this may seem demanding at first, it remains relatively accessible for beginners and constitutes a great introduction to text-based programming. In practice, writing simple rules only requires learning a few core elements of Lisp syntax, and a small set of basic arithmetic and logical operators.

In order to make PWConstraints more accessible to musicians without a programming background, Baboni Schilingi developed JBS-Constraints for PWGL in 2007. The library contains over 200 rules compatible with the PMC engine, grouped by musical topics such as pitches, intervals, shaping, etc. Each rule is presented as a function that generates code dynamically according to user-defined parameters, making them far more intuitive to manipulate than static code. In Max, the entire library was re-implemented as a single bpatcher named JBSrules, whose interface and parameters are updated dynamically when another rule is selected. With a click, the module returns the code for the active rule with its current parameters. This can help users learn the PMC syntax, and inspire the formalization of their own rules, like a "rule dictionary" of sorts. But it is not just a reference tool — it is also a functional rule module in its own right, that is applied as soon as it is inserted in the chain. A built-in "mode-switch" allows each rule to toggle between its true/false and heuristic variant, making it easier to solve logical contradictions. Even with minimal knowledge of Lisp and PMC syntax,

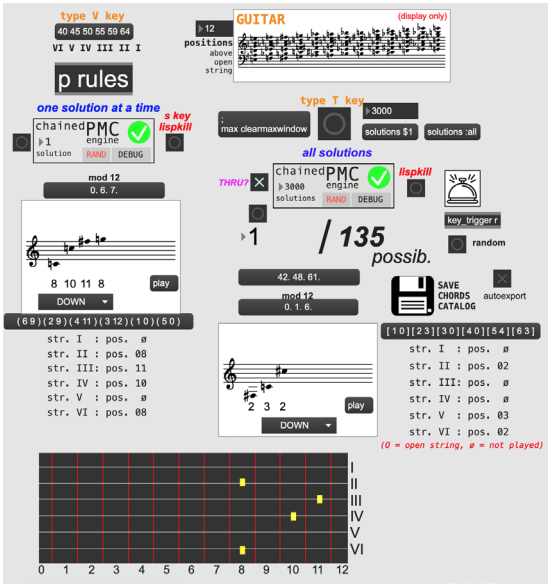


[30] PWGL's sources always remained closed, even though the software itself was distributed freely. This is why the version of PWConstraints that is now included with MOZ'Lib was based on a port realized by Sandred from PatchWork to OpenMusic in 1999, called OMCS. This version may not be the most up-to-date iteration of Laurson's system, but it is fully functional.

[31] Recently added to PWConstraints modules is the *SMC engine* based on Screamer, originally a PWGL library by Kilian Sprotte. Screamer is an extension of CommonLisp that adds support for nondeterministic programming. This alternate engine is compatible with the other PMC modules (and rule syntax), but it operates on different premises. Notably, it allows to generate results with no fixed length, which can be controlled with a dedicated "stop-rule" module.
cf. nikodemus.github.io/screamer/

[32] For a thorough description of the PMC syntax, one can refer to the documentation of MOZ'Lib, as well as the fifth chapter of Laurson's doctoral dissertation, which is entirely dedicated to PWConstraints. cf. Laurson, 1996, chapter 5 (pp.145-186)

users can achieve sophisticated scenarios of music generation by chaining a few instances of this module. For this reason, it constitutes a great entry point to experiment with constraints before studying more complex approaches.



Bringing such system to a polyvalent environment like Max allows to apply constraints not only in the context of notated music — that is the typical domain of CAC — but also in a great variety of scenarios — sound synthesis and processing, live music, interactive installations, multimedia projects, etc.[33] In Sandred's multimedia installation *Sonic Trails*[34], the PMC engine continuously guides the progression of audio/video processing of environmental recordings. Another example is Vincenot's piece 藏镜人—*puppetmaster*[35], where parts of the digital score are calculated during the live performance, some of which by using the PMC engine to re-arrange pre-existing materials or generate new sequences. We can also mention some ongoing research project by Vincenot, aiming to calculate optimal playing positions for string instruments, based on topological and musical rules. These patches have proven quite effective both as a writing aid during the composition process, and an exploratory tool to generate catalogs of materials to be further developed "in time" with the *Cluster-Engine*[36].

5.2. Cluster-Engine — the polyphonic solver

Created by composer Örjan Sandred in 2010 for PWGL[37], *Cluster-Engine* is a system for solving specifically musical constraint satisfaction problems. As its name suggests, it uses a cluster of smaller search engines (similar to Laurson's PMC) that collaboratively search for a solution. Results take the form of a score which satisfies a set of rules defined by the composer.

In music, a melodic line aligns well with the sequential logic[38] of a search engine like the PMC. However, in polyphonic music, multiple voices exist in parallel, making the sequential order of individual notes less obvious. In this case, a search engine must adhere to a specific search order[39]. In Cluster-Engine, the rhythmic content of voices is generated dynamically during the search, and one cannot predict in advance which notes will align across voices to form chords. Cluster-Engine handles search order by assigning a search engine to each parameter of each voice. Each individual engine follows its own sequential order, but it has access to data from the others and can trigger a neighboring engine to backtrack if necessary. Since there is no central or global search engine, the relative search order between voices or parameters is not as critical. Cluster-Engine was specifically designed to

[33] This was already imaginable with *bach*'s own implementation, namely *bach.constraints*. The latter in particular offers some accessible interface where rules are written with *bach* objects within a "lambda loop" structure, typical in that ecosystem. Different engines are available — strong (comprehensive, with back-tracking), heuristic (optimizing a solution based on a global score) and hungry (hybrid approach) — and some musical examples are showcased. In spite of interesting design decisions, and a much better adaptation of the interface to the context of Max, here are a few reasons why we tend to prefer the Lisp-based alternatives : 1) while *bach.constraints* may seem more versatile, our few tests revealed than the PMC engine's performances were superior for identical problems ; 2) we believe the original PMC syntax is actually more accessible to beginners than the *bach* lambda loop ; 3) finally, Cluster-Engine offers far more possibilities for musical exploration than any other engine available, and therefore is a favorite solution for advanced musical problem. Of course, comfort and personal preferences matter most, so we encourage musicians interested in constraints to try different implementations and see which is best for their own use.

[34] *Sonic Trails* (2021) was created in close collaboration with Max Sandred. The installation explores the deconstruction and reconfiguration of recorded audio and video environments using semi-transparent screens and immersive surround sound. The PMC engine controls various parameters of those processes.
cf. sandred.com/sonictrailsinfo

[35] 藏镜人—*puppetmaster* for flute, clarinet, piano and live computer was premiered by Trio de Nice in Cadillac·Shanghai Concert Hall, April 2023.

[36] These instrumental patches were initially commissioned by Jean-Baptiste Barrière in 2022, first focusing on the violin and cello. The same principles have been later applied to the guitar, for Vincenot's collaboration with guitarist Rafaël Carosi.

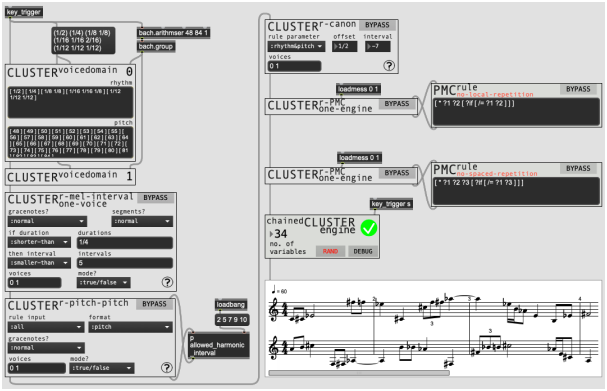
[37] Cluster-Engine was tested with early prototypes of PWforMax, but only completed in 2020. The project is maintained by Örjan Sandred and Torsten Anders as a pure Lisp library, independently from any CAC environment. As of 2025, it has been also ported to Opusmodus (by Anders) as well as OpenMusic and OM# (by Paulo Raposo), and is used by a growing community of composers.
cf. github.com/tanders/cluster-engine

[38] A fundamental requirement for a search engine is that it must explore possible solutions sequentially. To ensure that a valid solution is eventually found, "backtracking" (i.e. solving logical conflicts identified in a sequence) must occur in the reverse order of the search.

[39] In Laurson's *ScorePMC* — a variant of the PMC available in PW(GL) — the engine searches for pitch values within a predefined polyphonic rhythmic structure. The search proceeds in chronometric order, traversing all notes from lowest to highest voice — following the assumption that bass notes influence the pitch choices of upper voices — and giving priority to longer note values. This approach introduces a trade-off : adjacent pitches within a melodic line may be separated in the search sequence, which can reduce the system's efficiency for melodic rules but enhances its effectiveness for harmonic ones. cf. Laurson, 1996, pp.213-215.

improve search speed — in some cases, it delivers results up to 600 times faster than earlier versions[40].

A single voice is represented by two sequences : one for pitches and one for durations (including rests). An additional sequence represents the metric structure of the score, shared by all the voices. Cluster-Engine brings a lot of flexibility when defining the initial domain or search space : candidates can be either single values or motifs, i.e. predefined sequences of durations, pitches or chords, intervals, etc. A carefully designed domain can have a strong impact on the result, even before adding a single rule. In MOZ'Lib, the *CLUSTERengine* interface, reimagined for Max, is made of a large collection of bpatchers. They further develop the idea of the "chained engine" introduced with the PMC. One can build a functional generator of polyphonic scores by chaining any number of "voice-domain" modules (each dealing with rhythm and pitch candidates for a given voice), an optional "metric-domain" module, and the engine module itself.



Rules are easily inserted between the domains and the engine, and can be written in a great variety of ways. They can affect properties or relations not only in a single parameter of the score, or in a single voice (between pitch and rhythm), but also between a voice and the metric structure, between different voices, etc. Rules usually require two things to work properly : an accessor and a code part. The accessor, or rule applicator, determines which music parameter will be visible to the rule. About twenty accessors or rule-related modules[41] are available to combine. The code part is "the rule itself" so to speak — it checks properties of musical parameters made visible to it by a given accessor. It is written as a Lisp *lambda* — either in text form, or visually by using scripting boxes — and behaves similarly to PMC true/false and heuristic rules. Depending on the accessor's scope — adjustable with various parameters in the modules — incoming data from the lambda will be represented differently, thus changing how the rule can influence the result.

Cluster-Engine has been Vincenot's main composition tool since *silent_data_corrupt* for saxophone and live computer (2014). The Max version was used in a similar fashion, to generate materials for 藏镜人—*puppetmaster* (mentioned above) and *Hyperlinks—asset(s) failed to load properly* for flute and electric guitar (2023-25). The new environment opened up new contexts of application that were previously out of reach. For instance, the engine can now constrain a wider range of musical parameters — not only pitch, rhythm, and meter. This is made possible by the "Multidomains" modules proposed by Vincenot, which take advantage of the multiple "slots" assigned to each note in *bach* notation objects. Vassallo's piece *Elevator Pitch* for cello and electronics[42] made extensive use of this feature. Finally, Cluster-Engine has been integrated into patches for live performance — handling the creation of variations in live electronic responses in Sandred's *Sketches of Shifting Landscapes* for string quartet and live electronics (2024).

6. WEIRD PATCHES & (BILINGUAL) DESIGN ISSUES

Now that we have a broader view of MOZ'Lib and the techniques it offers, we need to address some of its shortcomings in terms of readability, accessibility, and the design of its programming interfaces.

MOZ has remained committed to pedagogy since its inception. In this regard, it shares many similarities with *bach* itself, which is arguably the most heavily documented Max package available. MOZ'Lib's overview gathers to this day a bit more than 300 patches, including "guest libraries"[43]. These aim to clarify technical aspects, to provide some (historical or musical) context, or even to explain some abstract theory attached to certain techniques. This likely reflects a propensity towards complexity inherent to CAC environments : higher level compositional topics

[40] Cluster-Engine is the culmination of Sandred's research on constraints since 1997. It builds on previous music solvers : *OpenMusic Rhythmical Constraints* (1998), then *PatchWork Musical Constraints* (2005). Both OMRC and PWMC relied on a single engine — Laurson's PMC itself — iterating over complex, composite objects including melodic motifs, rhythmic values and meter. The speed increase in Cluster-Engine stems from combining multiple search engines and the use of "back-jumping". This technique allows an engine to detect when a logical conflict arises in some earlier variable, and jump directly back to that point to try another value. By skipping intermediate steps, and erasing only the portion of the solution within the local sequence, the system reduces the need to reconstruct already valid parts of the solution. cf. Sandred, 2021

[41] A few examples of accessor modules are *r-PMC-one-engine* (rhythm or pitch rule on a single voice, backward compatible with PMC rules syntax), *r-rhythm-pitch-one-voice* (relation between pitch and rhythm for a given voice), *r-canon* (pitch and/or rhythmic canon between two voices), *r-pitch-pitch* (harmonic relations between n voices), etc.

[42] Vassallo experimented with "Multidomains" mainly to constrain the temporal behavior of the cello's bowing position and pressure, dynamics, articulation, tremolos and vibratos. cf. Vassallo, 2024

[43] Just as reading all *bach* tutorials and help files would be counterproductive, we do not encourage anyone to read the MOZ Overview from start to finish. Instead, users should find precise information on any topic exactly when they need it. Here, MOZ takes direct inspiration from *bach*'s "Help center", meant to be kept open on the side, like an encyclopedia. While *bach* offers a search function based on a limited set of keywords, MOZ draws on libraries with potentially uncommon vocabulary. Therefore a grep-based search was implemented, letting users find tutorials containing any term.

often require more verbose tutorials than the average Max audio-processing library.

Early design choices continue to shape the patching style in MOZ, which may seem unconventional compared to typical Max/*bach* patches. The modules' interfaces were designed to be as “eye-friendly” as possible, but they occupy more screen estate than most UI objects, which often requires adjusting patching strategies. To improve navigation and readability in complex patches, most modules include a “collapse” feature that shrinks them into a more compact form — either by clicking their title or sending a message.

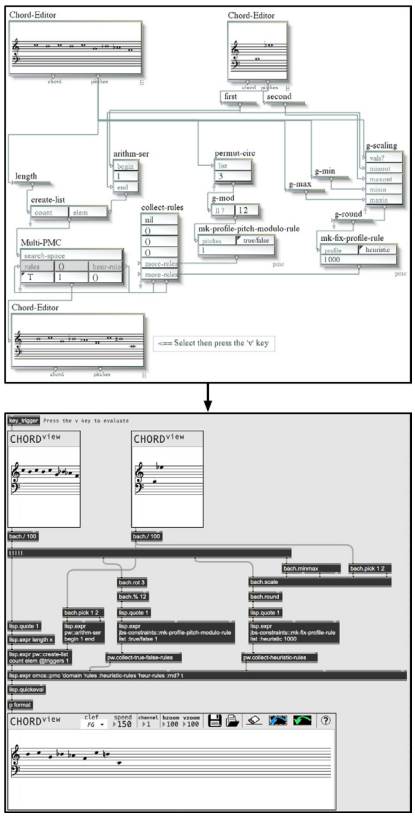
In terms of their inner workings, MOZ modules are as “vanilla *bach*” as can be[44]. Just like *cage*'s abstractions, users are encouraged to use the tools also as patching references for building their own systems. This leads to a known issue : early in the development, it was decided to let users choose their favorite mode of initialization (preset, *pattrstorage*, dictionaries, etc.) if they need one. As a result, adding several *loadbang* or *loadmess* to modules may be required, a reason why MOZ patches often seem a bit more cluttered visually.

Another issue is the apparent contradiction in Max between objects as *blueprint* and objects carrying some sort of memory. In the PW family, the full state of a Lisp function or UI object (like a score) is always saved with the patch, and not re-initialized when re-opening the patch. Things are different in Max, where every object comes with dozens of attributes — hundreds for certain *bach* objects — which complicates the visibility of what is memorized and what has to be handled manually by the user. This may seem trivial, but can have a very direct impact on compositional practice. In PW(OM)GL, it is common to duplicate an object, say a score, and move it aside in the patch before evaluating further, just like one might naturally move paper sketches around a desktop. But Max tends to resist this mode of interaction, and only certain objects like *bach.roll*, *bach.score* or *bach.reg* — with an “embed” attribute — can do this. Max abstractions and especially MOZ bpatchers cannot carry an internal memory elegantly when duplicated — without some heavy preset mechanism — and instead a brand new object is instantiated. It is to remedy this issue that certain “tricks” like the *MOZsafe*[45] module were imagined.

Attempting to create an intuitive, consistent bridge between the PW family model (bottom-up, demand-driven) and Max’s (top-down, event-driven), had a strong influence on what a typical PWforMax patch might look like. Once again, a key reference was *bach* itself, whose creators deliberately steered closer to Max than, say, OpenMusic[46]. While early MOZ modules leaned heavily into PW(GL) heritage, PWforMax orbits around an even stronger model : the Lisp language itself. This has led to some idiosyncrasies in the context of Max, like the need to add a "lisp.quote" before a Max list or *bach llll* enters the Lisp domain, or making chains of objects connected only via their second or third inlets, yet remaining functional.

An issue of readability we already mentioned is the “black box” effect. Patches made of objects generating bits of Lisp code can obscure the role played by a given abstraction, making patches more difficult to write and debug. While patching in PW(OM)GL already looked very similar to Lisp represented as a patch tree, each box could return its intermediate result with a single keystroke. Code generators in MOZ’Lib return nothing but nested code — evaluated later down the line — which makes patching more cryptic. This motivated the creation of *lisp.quickeval*, to be able to evaluate Lisp code anywhere in a patch more easily[47].

Likewise, when copying a patch from PW(OM)GL, or imagining a new one from scratch, a common issue is deciding when to switch from Max-*bach* to Lisp. This typically requires juggling between identical or near-



[44] With the exception of Bernstein's *shell* object — the only external included — MOZ uses exclusively pure Max or *bach* family objects.

[45] *MOZsafe* is a snippet that contains an embedded subpatcher, not an abstraction, therefore it carries its own memory after duplication. Its interface allows to “lock” or “unlock” parts of a patch on demand, an idea inspired by a similar feature in the PW family. This can be used, for instance, to prevent upper sections of a patch to erase some lower result, and to provide some intermediate initialization value.

[46] This is very visible in their choices for objects naming, consistency for instance with *zl* (a lower-level list processing family of objects in Max), attribute logic, or the way they conceived *bach*-style “lambda loops”. However, Ghisi and Agostini have often admitted this wish was not always entirely fulfilled, in particular concerning notation objects. cf. D. Ghisi & A. Agostini, « A compendium of *bach*'s shortcomings as it turns 10 », SMC 2020 Virtual Conference. Video available at : youtube.com/watch?v=_W8ghhJcTz8

[47] As this paper was being finalized, a new object was added to PWforMax : *lisp.antenna*. It allows to display the result of any scripting box used in a patch (*lisp.expr*, *lisp.iterate*, *lisp.quote*, etc.), as long as the user hovers the mouse over that object. This feature brings usage even closer to the PW experience, although it still requires a trigger to happen higher in the patch for it to work — typically via a simple key stroke.

equivalent functions in both worlds — *bach.trans* vs. *pw::mat-trans*, *bach.join* vs. *pw::x-append*, etc. A good practice to follow, in order to keep patches readable, is trying to rely on *bach* alone for as long as possible, until "something" simply could not be done without Lisp, or when switching would lead to a cleaner solution[48]. From that point on, the patch should remain in the Lisp domain until a full expression is evaluated. The result would appear as a *bach llll* or a simple Max list, then the patch would resume to pure Max/bach — possibly returning to Lisp again later if needed.

This leads us to what is arguably the biggest issue when programming with PWforMax : switching back and forth between the Max scheduler and *asynchronous* processes in SBCL. One cannot always guarantee when a Lisp result will be returned — especially with more complex, constraint-based processes. Consequently, even the most basic rule of Max dataflow, i.e. a trigger's outlets fire from right to left, can become unreliable[49]. Another very concrete problem arises when a user wants to take a portion of a patch into a loop. Iteration is notoriously tricky in Max, but *bach* can handle it gracefully with a *bach.mapelem* or combining a *bach.iter* and *bach.collect*. Yet in PWforMax, boxes only generate code, and therefore *bach* iterators are unusable. Instead, loops need to happen on the Lisp side. To address this, objects based on the "Iterate" library have been introduced, mimicking the appearance of *bach* iterators, including the use of "lambda loops". This solution has lead to very promising developments for composition in the last year, introducing two new constraint modules, *PMClooper* and *CLUSTERlooper*[50].

7. FUTURE OF MOZ'LIB

As a form of conclusion we will discuss future plans for MOZ'Lib, some of which are already under development. We can briefly mention the collection of pedagogical modules, which has reached a rather stable form in 2019 and is no longer a development priority. Improvements will keep being offered, and new modules are still introduced on occasion, essentially with practical or "quality of life" purposes in mind. The original modules essentially revolved around the two usual suspects of CAC : pitch and rhythm. This limitation may be addressed in the future, at least partially[51]. Yet again, it is not the intention for MOZ'Lib to be comprehensive and relevant on every musical front. Limiting its scope can also serve as encouragement for users to study *bach* more in depth, and start developing their own tools with it.

Since PWforMax has also stabilized as its own ecosystem — after years of testing, we believe it is reliable enough even for live situations — the next chapters will most likely be focused on "content", i.e. extending its knowledge base. New "guest libraries" are already being considered and discussed with their respective authors. The priority is given to high profile libraries that would be the most relevant in the current ecosystem, but also that could benefit from some extra time designing custom interfaces in Max and substantial documentation.

Here are a few libraries which may join the existing collection in the near future :

• Frédéric Voisin's *FV-Morphologie*, a 2010 "reboot" of the 1990s *Morphologie* library, dedicated to analysis, classification, recognition and reconstruction of numeric or symbolic sequences ;

• Torsten Anders' *Cluster-Rules*, compatible with Sandred's Cluster-Engine, concerning various musical scopes (rhythm, melody, harmony, counterpoint, etc.) ;

• Vincenot's library *JV-Components* (2008-2015), various techniques around analysis, supervised interpolation, instrumental constraints ;

• Sandred's ongoing development — codenamed "New Engine" — is a new approach to music constraints. It is similar in scope to the PMC, but allows both absolute and relative values to coexist in the domain definition : pitch and intervals, time positions and intervals of time. Over the course of its development, the engine has become increasingly focused on real-time applications. In particular, it supports the use of live input during an ongoing search, allowing the system to dynamically adapt while retaining the ability to backtrack—up to the point at which it has already produced output for live playback[52].

[48] A third scenario is often encountered by PW veterans, who may simply not know that an equivalent function exists in *bach* or how to call it. Some naming choices can increase confusion for Lispers, for instance the Lisp *member* and its Max counterpart *bach.belong*.

[49] This can lead to rather convoluted patches, for instance : a value that would logically be received last, in a pure Max context (leftmost outlet of a trigger), might arrive before some intermediate Lisp result. To fix this, the Lisp evaluation should always come last, or at least it would need to re-trigger some stored value received Max-side, so that it hits the "hot" inlet at the right time.

[50] The new modules allow to generate series of distinct PMC sequences or Cluster-Engine scores with varying search spaces, sharing a set of identical rule modules — taken in a *bach* "lambda loop". Dedicated UI elements help to "capture" some of these rules' parameters to vary on each iteration step, or to enable some rules only on given steps. Finally, the possibility to store intermediate results between iteration steps (Lisp-side) allows for even more complex scenarios. For instance, while generating sequences A, B, C,… certain rules could access the cumulated memory and make sequences D and F to imitate B and A respectively, or simply allow for smoother transitions from one segment to another. The two modules, still in development, can be seen as "meta-"constraint engines, allowing to generate complex scores with rich formal contrast.

[51] A collection of modules have been developed outside of MOZ in the last year, around a composition project by J.-B. Barrière. These modules are similar to the original MOZ modules, but allow to process entire scores in full, similarly to *cage*.

[52] The engine allows to extend a search indefinitely, while outputting results continuously. This property was used in Sandred's Shifting Landscapes for string quartet and live electronics (2024), where the generated electronic part continuously adapts to the performance.

A significant addition to MOZ'Lib over the last year has been a fully-fledged “user-library” feature[53]. Originally, adding new Lisp libraries to PWforMax could be only done in two ways. By default, those needed to be "hardcoded" by Vincenot in the official MOZ distribution, which is rarely desirable. Advanced users were still able to recompile a new .core file after including their own code to MOZ'Lib's own “sources” folder, but this would prove rather inconvenient with every subsequent update of the package. The newest solution relies on the Max package architecture (containing abstractions, help files, snippets, etc.), by adding a dedicated "sources" folder containing a Lisp library loaded via a dedicated .asd file. A Max patch found in MOZ's extras menu allows to select one or more “MOZ-compatible” packages, then can compile a fresh SBCL .core with a click, including not only the newly added libraries but also the default environment of PWforMax.

A couple of upcoming libraries are already being developed around this model, and will be distributed in the coming months :

• *NeuralConstraints*, developed by Juan Vassallo. This extension for PMC and Cluster-Engine introduces a new type of heuristic rules, influenced by a simple neural network trained on a given music corpus[54].

• *MOZ-Screamer*, a library on non-deterministic programming and music generation, originally developed by Paulo Raposo as OM-Screamer[55].

Since its first prototype in 2015, the development of MOZ'Lib has been slow but steady. It owes everything to the collective efforts from which it stemmed from, in first line the PRISMA research group, but also the community of Lisp-based CAC at large. Beyond its most seasoned audience, MOZ'Lib aims to give better access to composition to anyone, no matter their music background or technical training. More than ever, we believe these tools can help musicians develop a deeper relation with written music and music creation in general. Ten years after its inception, and perhaps for another decade, we hope its various flavors of “good old fashioned AI” for music generation can remain attractive and relevant for the toolbox of the 21st century composer.

[53] Implementing such a feature in a package that is itself a satellite of a multi-package ecosystem — all hosted within a larger programming environment — may all seem quite absurd. Another symptom of the profusion of packages, devices, tutorials and other propositions available in Max since its inception. The sheer density of this offer can easily become intimidating or even paralyzing at times — for both newcomers and veterans. But we hope this is a positive sign of the impact even the smallest (and hyper-specialized) endeavors allowed by such an environment. Most importantly, it may reintroduce in MOZ one of the defining qualities of its predecessors : the sharing of CAC libraries and techniques among composers.

[54] The main principle behind *NeuralConstraints* is to use the “mean absolute error”, returned by the trained network, as a heuristic value for a given rule. As a result, it allows to combine the properties inferred from the neural network with the more conventional, logical rules. cf. Vassallo, Sandred & Vincenot, 2025

[55] cf. github.com/PHRaposo/OM-Screamer

8. REFERENCES

1. Agostini, A. & Ghisi, D., “A Max Library for Musical Notation and Computer-Aided Composition,” *Computer Music Journal*, Volume 39, No. 2, p. 11–27, 2015.
2. Barrière, J.-B., “‘Chréode’: the pathway to new music with the computer,” in *Contemporary Music Review*, 1(1), M.I.T. Press, 1984.
3. Saariaho, K. “Timbre and harmony,” in *Contemporary Music Review*, 2(1), M.I.T. Press, 1987.
4. Baboni Schilingi, J. & Voisin, F., *Morphologie: Fonctions d'analyse, de reconnaissance, de classification et de reconstitution de séquences symboliques et numériques*, IRCAM, Paris, 1999.
5. Dunker, A., “On Technodiversity: A Conversation with Yuk Hui,” in *Los Angeles Review of Books*, June 2020.
6. Laurson, M., *PatchWork: A Visual Programming Language and some Musical Applications*, Studia musica no.6, doctoral dissertation, Sibelius Academy, Helsinki, 1996.
7. Norvig, P., *Paradigms of artificial intelligence programming: Case studies in Common Lisp*, Morgan Kaufmann, 1992. Also freely available at norvig.github.io/paip-lisp
8. Sandred, Ö., “Constraint-Solving Systems in Music Creation,” in *Handbook of Artificial Intelligence for Music*, Springer, 2021.
9. Vassallo, J. S., Sandred, Ö. & Vincenot, J., “‘NeuralConstraints’: Integrating a neural generative model with constraint-based composition,” in *Frontiers in Computer Science*, Sec. Human-Media Interaction, Volume 7, April 2025.
10. Vassallo, J. S., “Exploring Musical Procedural Rhetoric: Computational Influence on Compositional Frameworks and Methods in the piece ‘Elevator Pitch’,” *IJMSTA*, July 01; 6 (2): 1-16, 2024.
11. Vincenot, J., *Placeholders for [failure?]*, doctoral dissertation under the direction of Hans Tutschku and Chaya Czernowin, Harvard University, 2024.
12. Vincenot, J., “LISP in Max: Exploratory Computer-Aided Composition in Real-Time,” in *ICMC proceedings*, Shanghai Conservatory of Music, 2017.
13. Vincenot, J., “On ‘slow’ computer-aided composition,” in *OpenMusic Composer’s Book Vol.3*, Éditions Delatour – IRCAM, Paris, 2016.

Sonogram-Based Digital Scores and Flipped Learning: Breaking Down Hierarchies in Chinese Electroacoustic Music Education

Ruibo Zhang (Mungo)
Music, Technology and Innovation Research Centre
De Montfort University
chears.info@chears-dotinfo.co.uk

Mingyue Guan
China ElectroAcoustic Resource Survey (CHEARS)
moon.gmy@chearsdotinfo.co.uk

Kaiying Dong
Shenyang Conservatory of Music
dongkai-ying6616@gmail.com

ABSTRACT

This paper investigates how sonogram-based "digital scores"—realised through Acousmographe, eAnalysis, and TIAALS (Tools for Interactive Aural Analysis)—can enable flipped learning models that challenge the top-down hierarchies prevalent in Chinese music education. Through three case studies, we explore how digital listening interfaces can facilitate interdisciplinary and learner-driven approaches to electroacoustic (e-a) music analysis. Case 1 revisits Ambush on All Sides, a fixed-media composition based on the Chinese pipa, visualized through Acousmographe. Case 2 analyzes Fugue of Electronic Waves, a waveform-based fugue rendered in eAnalysis with visual annotations resembling children's cartoons. Both compositions were created by the first author and are analyzed by the second author, a pre-doctoral trainee originally trained in classical music theory. In Case 3, the second author applies TIAALS to study timbral mimicry between Western and Chinese instruments. To support a bottom-up learning process and reduce teacher–student dependency, an invited postgraduate contributor with a background in music therapy provided perspectives grounded in therapeutic practice. The study argues that while sonogram tools are sometimes considered outdated, they remain effective scaffolds for conceptual transformation in music-technology learning. This pedagogical framework encourages critical listening, cross-cultural dialogue, and flipped power structures in contemporary e-a music training in China.

Keywords: sonogram-based analysis, digital score, flipped learning, electroacoustic music pedagogy, Acousmographe, eAnalysis, TIAALS, cross-cultural analysis

Author Contributions:

The first author conceptualized the project, composed the fixed-media works for Cases 1 and 2, supervised the seconde author's training in electroacoustic analysis, and led the development of the flipped learning methodology and manuscript.

The second author initially observed works created in Acousmographe and eAnalysis, before undertaking her own analytical project using TIAALS and applying her background in classical analysis while developing new competencies in music technology through a pre-doctoral research project.

The third author served as an invited reviewer of Case 3, using her expertise in music therapy to support and evaluate the flipped learning process, and contributed critical insights on bottom-up pedagogy.

All authors contributed to the interpretation of results and approved the final manuscript.

Copyright: © 2025 Ruibo Zhang et al. This is an open-access article distributed under the terms of the Creative Commons Attribution License 3.0 Unported, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

1. INTRODUCTION

Electroacoustic music (e-a music), as both a creative and analytical field, has long struggled with inherited pedagogical hierarchies in certain cultural contexts. In China, education in music and technology remains predominantly top-down, characterised by a centralised system where the teacher leads and the student follows. This system typically impacts a student for many years—extending from undergraduate to postgraduate and even doctoral levels—often resulting in cohorts of students who reflect a kind of pedagogical self-similarity. As De Ritis (2022) observes, "individualism and nonconformity are discouraged – the group supersedes the individual" [1]. While recent shifts in China's academic and creative sectors have begun to acknowledge the value of "individual slices of genius," the pedagogical implications of this shift for electroacoustic music remain underexplored.

At the same time, the field of electroacoustic music has benefited from decades of development in tools for listening, analysis, and visualisation. One such tool is the sonogram—also referred to as a sonograph or spectrogram—which represents sound as a two-dimensional image over

time. As Roads (1995) explains, "a sonogram… represent[s] a sound signal as a two-dimensional display... The vertical dimension depicts frequency… and shades of grey indicate amplitude, with dark shades indicating greater intensity" [2]. This visualisation technique, though originally developed for acoustic analysis, has become central to how composers and analysts work with sonic material. What we now refer to collectively as digital scores, in a much broader sense, builds upon this visual foundation and introduces a spectrum of software platforms that integrate image, sound, and interactive interpretation.

Sonogram-based tools have served as quiet agents of bottom-up learning in Chinese e-a music education over the past two decades. Through three case studies, we revisit a body of work that has implemented digital score practices as an interface for sound-based thinking, rather than notation-based one, since the early 2000s. These cases reveal that such tools were used to encourage creative agency, long before flipped learning became a recognised educational model. Through the lens of visual analysis, cross-cultural identity, and "flipped environment", we argue that the sonogram is more than a display—it is a pedagogical interface. The following sections outline the methodological framework and discuss how digital scores may be re-contextualised as tools of pedagogical disruption, challenging the enduring teacher-student hierarchy in Chinese music and technology education.

2. METHODOLOGY

This study adopts a pedagogy-driven methodology to examine the role of sonogram-based tools—referred to here as digital scores—in reshaping electroacoustic (e-a) music education in China. Drawing from flipped learning theory and comparative tool analysis, the research approach challenges the traditional teacher-student hierarchy by engaging learners as active participants and co-constructors of knowledge.

2.1. Flipped Learning as Pedagogical Framework

The pedagogical core of this study is informed by flipped learning, a model that redefines the teacher's role from knowledge authority to learning facilitator. According to Straw, Quinlan, Harland, and Walker (2015), while in a traditional classroom "the teacher's role is to lead the lesson/pass on knowledge," in a flipped model "the teacher's role is mainly to facilitate student-led learning" [3]. This reorientation is particularly relevant in Chinese higher education, where cultural norms often discourage independent exploration and emphasize conformity as mentioned earlier by De Ritis in 2022. By integrating flipped learning into e-a music training, this study seeks to enable a bottom-up pathway in a historically top-down educational culture.

The flipped model was implemented in two directions:

In Case 2, a struggling undergraduate student re-engaged with music technology through hands-on interaction with visual tools.

In Case 3, a classically trained music analyst used a psychoacoustic lens to reinterpret sound-based representations, guided but not dictated by her supervisor.

This dual-flip allows for student-led inquiry from both low-entry and high-expertise perspectives, disrupting conventional hierarchies in both technical and conceptual domains.

2.2. Digital Scores as Analytical Tools

Technologically, this study is grounded in the evolution of sonogram-based tools as discussed in The Computer Music Tutorial, Second Edition (Roads, 2023) [4]. While the first edition (1995) introduced the sonogram primarily as a spectral display, the second edition expands this foundation by documenting many systems that build upon it, such as Acousmographe, eAnalysis, and TIAALS, which collectively support what we now refer to as digital scores in a broad sense. The following three systems play a central role in this paper:

Acousmographe "supports a transcription of electroacoustic music starting from a spectrogram representation" and "lets the user inscribe a library of colour graphic symbols onto a spectrogram" (p. 1436) [4]. Developed in 1993 at the Groupe de Recherches Musicales, it remains an influential tool for bridging analytical notation and perceptual listening.

eAnalysis (2016) "provide tools for creating analytic text and graphics for music study," (p.1440) [4] which interface is more accessible, lending itself particularly well to teaching environments.

TIAALS (Tools for Interactive Aural Analysis), introduced in 2013, "enable users to develop interactive analyses in an environment integrating a sonogram and a chart maker" (p. 1439) [4]. Unlike sonogram objects, "objects placed on the chart are visually abstract—a symbolic mode of representation," offering an open-ended structure for interpretive and comparative study.

These tools represent a methodological shift: from static representations toward exploratory environments for music analysis. Their pedagogical relevance lies not only in the data they display, but in how they allow learners to actively construct meaning—by inscribing, annotating, and interpreting sound.

This triad of digital-score systems allows for a flexible interface for flipped environments, where students engage in interpretation before formal instruction. Each system encourages student-led perception and contextualised learning, making them particularly effective for bottom-up strategies in music technology education.

2.3. Case Study Design

Three case studies were selected to reflect different periods, perspectives, and levels of experience:

Case 1 uses Acousmographe to analyze an electroacoustic work derived from traditional Chinese music.

Case 2 explores a student's reinterpretation of digital waveforms using eAnalysis.

Case 3 examines how TIAALS supports a trainee's shift from classical to technological analysis, with added input from a third-party observer.

Each case is situated at a different point on the composer–student–analyst spectrum, with the aim of assessing how digital scores function not only as analytic tools but

as interfaces for flipped learning. The combination of temporal breadth (2003–2023) and multiple user identities offers a comprehensive look at how sonogram-based approaches have evolved pedagogically, and how they may continue to foster independent inquiry in Chinese e-a music education.

3. CASE STUDY 1: AMBUSH ON ALL SIDES AND ACOUSMOGRAPHE

The first case study explores Ambush on All Sides, a fixed-media electroacoustic composition created in 2003 by the first author during a transitional moment in his musical identity. With a strong background in Western classical music, the composer had only recently encountered acousmatic practices, though he had many years experiences with technologies on recording and mixing for music productions. Ambush on All Sides marked his earliest venture into sound-based composition, reflecting a critical juncture at which he moved beyond notation-based frameworks and into electroacoustic idioms grounded in sonic gesture and spatialized sound.

The work draws conceptual inspiration from the famous Chinese pipa piece Ambush from Ten Sides[1], but radically reinterprets its sonic materials. Rather than recording directly, the composer sourced pipa material from commercial CDs, treating them as plunderphonic samples. These were subjected to basic sound manipulations—such as reversing, time-stretching, and pitch-shifting—to generate transformed textures. Additional layers included children's vocal shouts, transposed by an octave to resemble adult male voices, contributing to a disorienting and dramatic soundscape at the climax. This approach preserved the martial intensity of the original while aligning with the acousmatic principle of separating sound from its source.

A recomposed version titled New Ambush on All Sides[2] (2004) retained only the opening ten seconds of the original. It introduced a digital sampler and automated controllers, significantly expanding both the structural depth and sonic complexity of the work. However, this experimentation coincided with the composer's growing disillusionment with local educational practices. While pursuing his postgraduate degree at Central Conservatory of Music in Beijing China, he observed that most e-a students—undergraduates and postgraduates alike—were encouraged to follow uniform procedures dictated by teachers. The essential aesthetic foundations of acousmatic music (e.g., reduced listening) were often ignored or misunderstood, even by instructors. This top-down teaching model limited creative independence and failed to cultivate sound-based awareness.

In response, the first author began developing his own terminology and teaching strategies, laying the groundwork for what would later evolve into a bottom-up, student-centered pedagogy for electroacoustic music. In 2011, he revisited Ambush on All Sides, recognizing that its relative simplicity made it especially suitable for analysis using Acousmographe. This visual representation was conceived not merely as a supporting aid, but as an interface for sound-based thinking—a pedagogical bridge designed to articulate structural form, dynamic gesture, and spectral density to musicians trained primarily in notation-based traditions.

The Acousmographe[3] (CHEARS, n.d.-a) [5] version premiered in China in 2012 (see figure 1), in a concert curated by the composer to introduce electroacoustic music to colleagues and students at Shenyang Conservatory of Music, Liaoning China. At the time, sound-based listening was still foreign to most Chinese musicians, whose training was almost entirely notation-based. The visual score became a necessary bridge, facilitating comprehension by "compensating the missing part for the eye"—that is, enabling a visual frame for sounds whose sources were invisible. After its success in China, the piece and its Acousmographe rendering were also presented at De Montfort University, Leicester UK as part of the composer's doctoral research.

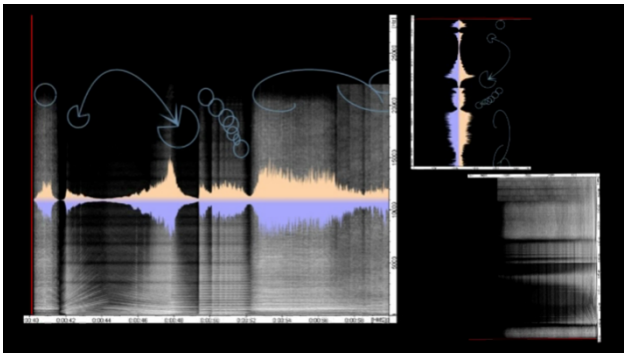


Figure 1. Ambush on All Sides on Acousmographe

Importantly, the use of Acousmographe did not originate from a teacher's instruction, but from the composer's own inquiry, dating back to 2005. At that time, the dominant institutional model required students to outsource video representations to multimedia technicians in order to present fixed media works with visuals—an arrangement that reinforced the notion of visuals as supplementary rather than integrated. In contrast, Acousmographe represented a non-hierarchical, reflective tool—an early act of resistance against the prevailing top-down model.

In this study, the second author (a pre-doctoral trainee with a background in Western classical analysis) re-engaged with the piece using Acousmographe to provide an objective analytical perspective. This re-reading emphasized how digital score tools can empower analysts to explore complex sonic forms independently. The second author's interpretation—without guidance from the composer—reaffirming Acousmographe's potential as both an analytical platform and a pedagogical instrument in flipped environments.

This case thus exemplifies how a digital score can act as a bottom-up mechanism for cultivating electroacoustic literacy. It also testifies to the potential of digital visualisation tools to democratise analysis, provide access to non-

[1] 《十面埋伏》: Ambush from Ten Sides

[2] http://chears.info/frontend/concert/works.aspx?worksid=5

[3] http://chears.info/frontend/terminology/allterms.aspx?alltermsid=11

notated music, and subvert traditional teacher-student hierarchies, particularly in regions where acousmatic practices are not yet institutionally embedded.

4. DIGITAL SCORE AS A PEDAGOGICAL INTERFACE: COMPARATIVE CASE STUDIES

4.1. A Tool for Divergent Learners in Transition

This section examines two contrasting cases—an undergraduate student on the brink of academic failure and an independent scholar on the verge of doctoral research—who both engaged with digital score tools to overcome systemic and conceptual barriers in the study of electroacoustic (e-a) music. Despite their different circumstances, each found in the digital score a transformative interface: not only for navigating unfamiliar content, but for reclaiming agency within an educational environment traditionally driven by top-down structures.

The pairing of these cases highlights a core argument of this paper: digital score technologies such as eAnalysis and TIAALS can serve as student-centered platforms, fostering sound-based thinking in learners conditioned by notation-based traditions. These tools not only support technical comprehension of sound phenomena but can also catalyze new modes of artistic and academic inquiry.

4.2. Case Study 2: From Conceptual Breakdown to Visual Intuition

The second case features a Chinese undergraduate student who failed his initial entrance into the centralised conservatoire system and was forced to seek alternatives abroad. In 2016, during his first year studying music technology in the UK, he faced major challenges, most notably a language barrier that severely limited his grasp of theoretical concepts. After failing all modules, he was given one last opportunity to resit the year—but only if he could complete an essay on timbre, a concept that had so far eluded him.

The first author selected Fugue for Electronic Waves (2003), an early electroacoustic work composed using various waveforms—sine, triangle, square, and saw—as the basis for the student's investigation. Using eAnalysis [4] (CHEARS, n.d.-b) [6], which was designed by "Simon Emmerson, Leigh Landy…and [Pierre Couprie]…at the Music, Technology and Innovation Research Centre at Leicester's De Montfort University" [7], the student was able to explore these waveforms visually via sonogram, contrasting their spectral properties by freely drawing cartoon-like shapes—intuitively and playfully—from the depths of his childhood imagination. The cartoon-like (see figure 2) sonogram layout became a crucial bridge for understanding: for instance, he could finally grasp how differences in harmonic content shape tone colour. The essay he submitted was sufficient to qualify for the resit, marking a pivotal turnaround in his studies.

From the first author's perspective, this intervention validated the need for learner-led, bottom-up pedagogies in e-a music. With neither the capacity to survive China's rigid conservatoire system nor the linguistic fluency to align with standard instruction in the UK, the student required a tailored, visually guided method of access to core concepts. The digital score thus served not only as a pedagogical tool but as a lifeline.

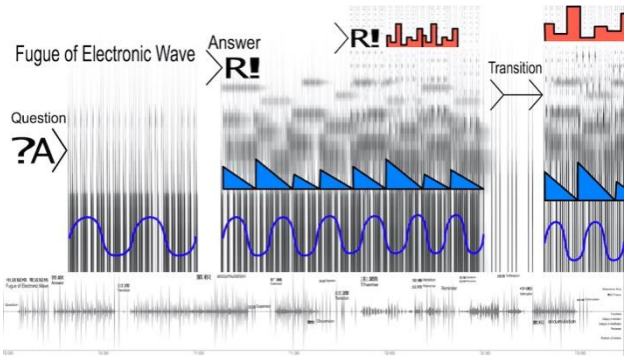


Figure 2. Fugue of Electronic Wave on eAnalysis

The second author, who later encountered this piece herself, noted the balance between technological design and musical structure. The micro-level rhythmic sequences were algorithmically generated, while the macro structure reflected a fugal design across waveforms. In her view, the piece offered an accessible and elegant fusion of sound design and compositional thinking—an ideal educational example.

4.3. Case Study 3: From Theoretical Imprisonment to Sonic Discovery

In stark contrast, the third case focuses on an independent researcher (the second author) with a master's degree in contemporary music analysis. Having previously completed her undergraduate studies in "electronic music composition" at Shenyang Conservatory of Music, she had been deeply disillusioned by the split between theory and practice in her training. The curriculum had been divided: one part focused on commercial DAWs for producing note-based music; the other promoted sound manipulation for acousmatic composition—without any conceptual grounding in aesthetics, listening modes, or sound-based analysis. Key concepts such as the acousmatic condition, reduced listening, or even timbre were absent.

This left her in a theoretical vacuum. Pursuing a master's in note-based analysis helped rebuild her academic confidence, but it also distanced her further from sound-based creative practice. By the time she began considering doctoral study, she was uncertain where to begin or whom to work with. In China's PhD system, access is often limited by a shortage of supervisors and reliance on private

[4] http://chears.info/frontend/terminology/allterms.aspx?alltermsid=892

lessons long before entrance exams—a structure that perpetuates elitism and restricts independent research paths.

The first author, recognising her potential, introduced her to foundational vocabulary in acoustics, psychoacoustics, and 20th-century aesthetic movements such as formalism, minimalism, and radicalism. As her conceptual framework developed, she was introduced to TIAALS. Through this tool, She discovered that the most vivid instance of sonic mimicry—originally assumed to involve Western instruments imitating Chinese timbres, a process she considered only a halfway transition from notation-based to sound-based thinking—was, in fact, the erhu mimicking a horse's neigh at the very end of the piece. This marked a successful shift in her conceptual approach toward a fully sound-based analytical paradigm, as evidenced in her own words (see figure 3):

"As we can see, there are two sonogram-based scores: the upper one is a real horse neighing, and the lower one is the erhu mimic. The real horse neighing shows that the greatest intensity of amplitude (Roads, 1995) is concentrated around 1-2 kHz in the shades of grey. Although other frequency bands have energy distributions, they are not as dark as the band at 1-2 kHz. The erhu mimic also shows that the greatest intensity of amplitude is concentrated around 1-2 kHz, but the other frequency bands have very little energy distribution. The analysis reveals that both sound signals exhibit their primary energy concentrations around 1-2 kHz, demonstrating similar spectral characteristics in this dominant frequency range for human hearing, as perceived through the skills of critical listening."

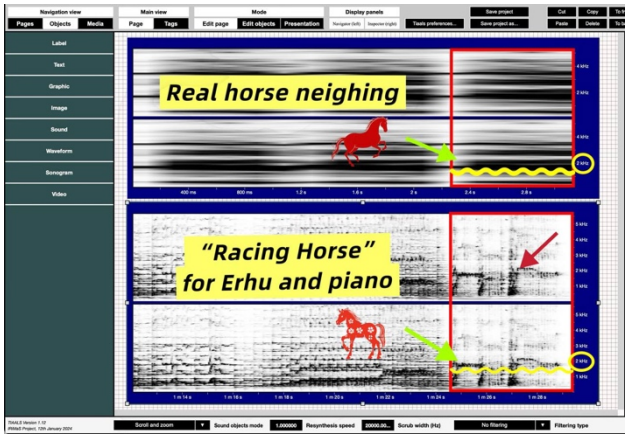


Figure 3. Horse-shaped symbol in Chinese paper-cutting art in TIAALS (Tools for Interactive Aural Analysis)

Using TIAALS, she highlighted this moment in the sonogram, marking the frequency band with horse-shaped symbols—one of which was stylised in Chinese paper-cutting art (also see figure 3). This act fused cultural reference, sonic analysis, and visual symbolism into a powerful new research direction: exploring mimicry across East-West musical aesthetics, through the lens of sound-based and visual-auditory analysis. While her PhD research is still in preparation, this moment marked a breakthrough. For the first time, she was able to conceptualise her research question not through borrowed theories or imposed structures, but through direct interaction with sound.

4.4. Comparative Reflection: Resistance, Reorientation, and Sound-Based Thinking

While these two case studies differ in level and scope, they reflect parallel journeys of resistance against the constraints of a top-down educational system. Both learners had been misaligned with institutional expectations—Case 2 due to academic and linguistic struggles, Case 3 due to structural barriers and historical bias in her field. In each case, the digital score provided a medium for reclaiming direction. The table below summarises the comparison (see table 1).

Aspect	Case 2: Survival Learning	Case 3: Transformative Inquiry
Background	Struggling undergraduate with weak conceptual/language base	Independent analyst with advanced degree, but misaligned training
Challenge	Understanding timbre conceptually and visually	Bridging note-based training with sound-based research
Digital Tool Used	eAnalysis (simplified, cartoon-like)	TIAALS (interactive, symbolic, analytical)
Learning Outcome	Passed resit; internalised key sonic concept	Formulated original PhD research idea using digital analysis
Pedagogical Insight	Visual scaffolding supports abstract comprehension	Symbolic sonogram encourages conceptual autonomy and creativity
Wider Implication	Necessity of bottom-up, adaptive teaching in e-a	Digital score as interface for decolonising and rethinking research paradigms

Table 1. Summarises the comparison in Case 2 and 3

Together, these cases support the argument that sonogram-based digital scores are not simply tools for analysis, but instruments of pedagogical transformation. Whether simplifying timbre for a beginner or enabling advanced cross-cultural research questions, they shift the student's role from passive recipient to active discoverer. In this way, they embody the very essence of a flipped environment, enabling students to build their own sonic literacies and research trajectories.

5. DISCUSSION

The issues observed in centralized music education, particularly within conservatoire systems, are not rooted in

the knowledge itself—much of which originates from the West—but rather in the rigidity of how that knowledge is transmitted. Over the past four decades, these systems have tended to reinforce top-down, prescriptive pedagogies, often prioritizing notation-based models and suppressing alternative sonic perspectives. This stagnation reflects a broader resistance to pedagogical innovation: when an entrenched framework dominates teachers' and students' thinking, any new paradigm—such as sound-based thinking—must struggle to find entry. The analogy to legacy energy industries is apt: just as oil dominates infrastructure and markets even as cleaner alternatives emerge, note-based conventions continue to overshadow exploratory sound practices.

Flipped learning, also developed in Western contexts, offers one potential corrective—but its adoption in Chinese music education remains limited. Many students and educators, as the 3rd author personally experienced around 2019–2020 in China, are familiar with the concept only through its translation—without fully engaging with its underlying pedagogical implications. Here, digital scores emerge as an essential tool—not only as a form of media, but as a conceptual key for unlocking the potential of flipped, bottom-up learning. This is particularly evident in the first author's long-term research trajectory. While foundational texts such as The Computer Music Tutorial (Roads, 1995; 2023) are updated only once in several decades, bottom-up experimentation has progressed in parallel through field-driven initiatives like CHEARS (China Electroacoustic Resource Survey), which not only integrates tools like Acousmographe and eAnalysis—as referenced in this paper—but also provides a comprehensive bilingual terminology for the field of electroacoustic music in both English and Chinese. Initiated as part of the first author's doctoral research, CHEARS established a platform to implement and test learner-led methodologies in electroacoustic music. After confirming success in the undergraduate case (Case 2), this trajectory continues to evolve in postdoctoral contexts, including the support provided to the doctoral-bound researcher in Case 3.

Informed by her background in music therapy, the 3rd author offers an additional layer of insight into the pedagogical implications of digital score tools. She identifies a strong resonance between the subversion of traditional teacher-student hierarchies—central to the bottom-up approach—and the core therapeutic principle of egalitarian collaboration. In music therapy, therapists serve not as authoritative instructors but as facilitators who accompany clients in co-creative processes. Here, the visual and interactive affordances of sonogram-based digital scores (e.g., symbolic annotations in TIAALS) mirror therapeutic practices by validating individual sonic expressions without judgment. This approach aligns well with therapeutic work involving clients with limited verbal communication—such as children with autism—by enabling them to label or graphically annotate their sound expressions. Such methods reduce reliance on conventional musical literacy and reposition clients not as students conforming to norms, but as autonomous agents whose sound-making carries intrinsic value.

6. CONCLUSIONS

Top-down and bottom-up approaches are often seen as opposites, yet they can also be complementary. Top-down systems are undeniably effective for the majority, especially when efficiency and structure are paramount. However, a single model cannot fit every learner or context. When top-down systems reach an impasse—be it due to rigidity, cultural mismatch, or learner diversity—bottom-up alternatives naturally emerge. Once successful, these grassroots solutions may rise to broader acceptance and eventually influence top-down policy, until a new cycle of adaptation is required. This dynamic interplay resembles the philosophical balance of Yin and Yang—constantly shifting yet interdependent.

In this study, flipped learning was not merely cited as an educational trend but was actively embedded within the notion of the digital score. Here, the egalitarian potential of digital score becomes evident: a teenager drawing freely may express just as valid an insight as a professor with decades of training. Drawing is perhaps our most primitive means of expression—prehistoric cave markings and early Chinese characters both remind us that images came before formal language. Once freed from semantic constraints, these elements can be creatively reused as part of today's visual-musical discourse as Fields quoting the "new composite character designed by Zhang Ruibo" [8] for the changes of sound with a further development (see figure 4):

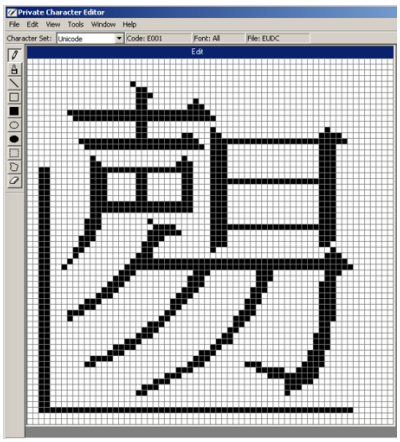


Figure 4. The further development to the new composite character for Sound Changes.

Through case studies, we demonstrated how digital score practices supported flipped learning in intuitive ways. Whether through cartoon-like drawings or symbolic Chinese paper-cuttings, the visual nature of digital score allowed students to take control of their learning process. As shown in Case 2, this tool helped a student reclaim his academic path, and in Case 3, it helped another redefine her scholarly identity from her previous educational traumas.

Beyond education, the visual and participatory qualities of digital score extend its relevance into therapeutic contexts, where it supports broader processes of healing, self-

expression, and emotional awareness. As the 3rd author (a music therapy specialist) points out, the visualization feature of sonogram tools opens new pathways for sound awareness in clinical practice. Clients who struggle with verbal communication—such as those with PTSD (Post-Traumatic Stress Disorder) or, in less severe cases, post trauma syndrome—can visualize their sonic expressions using tools like Acousmographe, eAnalysis and TIAALS. These tools not only allow clients to better perceive and understand their emotional states but also help therapists monitor patterns and evaluate change over time. In this way, digital score serves as both expressive aid and analytic instrument, reinforcing the client's autonomy while supporting therapeutic goals.

At the other end of the spectrum, digital scores have the potential to evolve into extremely complex systems—particularly in electroacoustic (e-a) music. Scholars and practitioners in the 21st century are increasingly experimenting with new representations of musical thought.

As Stolet (2024) argues, this pursuit challenges centuries of conventional notation practices: "Traditional Western notation confirms the importance of starting and stopping musical events as does the need to perform musical events in a coordinated and synchronized way…We can observe this spatial orientation in Western musical scores where vertical positions of note heads signify events in 'pitch space', and where a note's horizontal placement in space indicates its position in time." [9]

As digital scores become more intricate, however, printed reference materials such as The Computer Music Tutorial (Roads, 1996; 2023) lack the flexibility to accommodate this expansion. In such cases, frameworks like CHEARS (China Electroacoustic Resource Survey)[5] become vital—not only for managing bilingual terminologies in e-a music but also for classifying and contextualizing emerging forms of musical representation. This raises an important question: is digital score a genre, a presentation method, or something else entirely. Just as Zhang (2024) notes in his doctoral thesis, the need for alternatives becomes urgent when a system grows too sophisticated or exclusionary:

"As a dichotomy, the top-down scheme was used by 'big-man' who obtained massive knowledge and was able to organise these concepts into an arrangement of a scientific system... The bottom-up scheme was used by a 'small-man'... who merely obtained limited knowledge and tried to put these concepts into an arrangement for the best interests of their own... When [top-down] classification is too sophisticated to a (Chinese) user, an alternative is always available to the user in a mode of flipped learning." [10]

In sum, digital score serves not merely as a tool for visualizing sound, but as a conceptual key to unlock stagnant systems. Whether rescuing a struggling undergraduate, inspiring a researcher in transition, or providing therapeutic insight, it enables learners to draw their own path—literally and figuratively. It represents a convergence of art, technology, and pedagogy—bridging East and West, tradition and innovation, top-down and bottom-up.

[5] Fully sponsored by China Scholarship Council (CSC)

7. REFERENCES

[1]. A. P. De Ritis," 'Mixed blood' and aesthetic evolution in China's electroacoustic music today," *Organised Sound*, 27(3), 2022, pp.307–315. https://doi.org/10.1017/S1355771822000279

[2]. C. Roads, *The Computer Music Tutorial*. MIT Press, 1995, pp.563-564.

[3]. S. Straw, O. Quinlan, J. Harland, & M. Walker, *Flipped learning Practitioner Guide*. National Foundation for Educational Research (NFER) and Nesta, 2015, P.7.

[4]. C. Roads, *The Computer Music Tutorial* (2nd ed.). MIT Press, 2023, (pp. 1436, 1439-1440)

[5]. CHEARS. (n.d.-a). Acousmographe. China Electroacoustic Resource Survey. http://chears.info/frontend/terminology/allterms.aspx?alltermsid=11

[6]. CHEARS. (n.d.-b). eAnalysis. China Electroacoustic Resource Survey. http://chears.info/frontend/terminology/allterms.aspx?alltermsid=892

[7]. Couprie, P. (2018). "Methods and tools for transcribing electroacoustic music". In S. Bhagwati & J. Bresson (Eds.), *Proceedings of the International Conference on Technologies for Music Notation and Representation (TENOR '18)* (pp. 7–16). Concordia University. ISBN 978-1-5251-0551-7. Montreal, Canada.

[8]. Fields, K. (2023). "Sheng Yijing: Book of sound changes". In A. P. De Ritis, V. Zappi, J. Van Buskirk, & J. Mallia (Eds.), *Proceedings of the International Conference on Technologies for Music Notation and Representation (TENOR 2023)* (pp. 171–175). Northeastern University. ISBN 978-0-6481592-7-8. Boston, Massachusetts, USA.

[9]. J. Stolet, "Music-centric description of performance with data-driven musical instruments," *in Proceedings of the 49th International Computer Music Conference* (ICMC 2024), Seoul, p.86.

[10]. R. Zhang, "Assisting the development of the field of electroacoustic music studies in China" (PhD dissertation). De Montfort University, Leicester, UK, 2024, pp.153-154.

CHORDSNEST: DEVELOPING A MULTIPHONICS PALETTE FOR MAXSCORE

Aigerim Seilova
Hochschule für Musik und Theater Hamburg
aigerim.seilova@hfmt-hamburg.de

James Tsz-Him Cheung
Hochschule für Musik und Theater Hamburg
tsz.him.cheung@hfmt-hamburg.de

Georg Hajdu
Hochschule für Musik und Theater Hamburg
georg.hajdu@hfmt-hamburg.de

ABSTRACT

ChordsNest is a new extension for MaxScore (a notation environment for Max) that provides an integrated multiphonics workspace. A multiphonic is an extended technique where multiple notes are produced simultaneously by a typically monophonic instrument. This tool addresses the difficulty of notating and accessing such by uniting chord-like notation, fingering diagrams, and sample-based playback in one interface. Over the course of the project, 136 B♭ clarinet and 130 bass clarinet multiphonics were systematically recorded, categorized, and incorporated into the user interface, allowing composers to search and select multiphonics via tags. ChordsNest offers immediate audio feedback for each multiphonic and automatically displays the corresponding fingering diagram on the score, removing the manual workaround traditionally needed. By leveraging MaxScore's message-driven architecture and its integration in the Max environment, the system seamlessly unites notation, visualization, and playback. This significantly enhances the workflow for composers working in contemporary and electroacoustic idioms. The paper outlines the development methodology, implementation details, and discusses future extensions of the ChordsNest environment.

1. INTRODUCTION

Multiphonics are extended instrumental techniques in which a performer produces more than one pitch simultaneously on a typically monophonic instrument, such as the clarinet [1]. These complex, often fragile sounds have become increasingly common in contemporary music, offering rich sonic textures and expressive potential. However, working with multiphonics presents challenges in both notation and performance.

Traditionally, composers refer to catalogues such as Bartolozzi's *New Sounds for Woodwind* [2] or Richards' *The Clarinet of the Twenty-First Century* [3] to identify usable fingerings. The actual sonic result, however, is often unpredictable, and performers may not reliably reproduce the intended sound. Notation typically involves representing the resulting chord and manually inserting fingering graphics – often via external graphic tools or custom scripting solutions. These workflows are cumbersome and fragmented, and the actual sonic result is usually only heard at rehearsal.

ChordsNest was developed to address these issues by providing an integrated solution that unifies chordal notation, fingering diagrams, and immediate playback within the MaxScore environment. It is designed to support composers and performers alike, reducing the technical overhead of working with multiphonics and streamlining their use in both composition and performance.

Copyright: © 2025 Aigerim Seilova et al. This is an open-access article distributed under the terms of the Creative Commons Attribution License 3.0Unported, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

1.1. Existing Notation Tools and Multiphonic Notation

Multiphonics are increasingly common in new music: i.e., Fallowfield [4] analyzes complex multiphonic fingerings for cello, and specialized clarinet catalogs by Roche [5][6] document dozens of multiphonic fingerings (with ease-of-production ratings and audio examples). In traditional engraving practice, however, there is no single standardized method for notating multiphonics across mainstream notation software. Composers and performers often resort to ad-hoc solutions – writing multiple simultaneous notes in separate voices, drawing custom fingerings by hand, or attaching graphics or text glyphs – leading to extra work and potential ambiguity in the score.

Most notation applications support multiple simultaneous voices, but none include a ready-made multiphonic fingering system. For instance, Sibelius [7] explicitly instructs users to write multiphonics by using a second voice on the same staff and selecting alternate notehead shapes for those notes. MuseScore [8] likewise allows up to four voices per staff, and users often treat multiphonics as chords or secondary voices with modified noteheads; however, any specialized fingering diagram must be created as an image or composed from basic fonts. LilyPond [9] can express simultaneous notes via multiple Voice contexts and supports engraved woodwind fingering diagrams (introduced in v2.14). However, each multiphonic must still be constructed manually, as there is no predefined library of multiphonic charts in the source distribution.

Dorico is designed for advanced notation, yet as of its latest release it offers no dedicated multiphonic tool. Its official documentation explains that users import custom

fingering graphics or design bespoke symbols via the "Creating custom playing techniques" interface [10], and forum discussions confirm that multiphonic graphics are typically defined as custom playing techniques in personal libraries [11].

In the Max environment, notation has begun to see more real-time and flexible solutions. MaxScore allows scores to be generated and updated on-the-fly [12]. Agostini and Ghisi's bach library introduced a suite of objects for creating and manipulating notation within Max, but even here there is no specialized multiphonic support – users assemble noteheads or attach static images manually [13]. ChordsNest fills this niche: it provides MaxScore with a dedicated palette of fingering charts and symbols for woodwind[1] multiphonics. By selecting from this palette, a user can spawn correctly formatted multiphonic notation objects directly in the score, without leaving the Max/MaxScore environment. This integration is novel because it combines real-time score editing with a high-level, instrument-specific notation interface.

2. METHODS AND MATERIALS

Given the absence of existing tools that seamlessly combine multiphonic notation, categorization, and playback, this project was motivated by the need to create a resource grounded in real-world instrumental practice and informed by performer expertise. From the outset, we sought to develop ChordsNest not only as a technical solution, but as a collaborative endeavor bridging compositional needs with realities of performers.

Central to this process was our collaboration with clarinetist Heather Roche, whose expertise in multiphonics was essential to the project. Through regular online meetings, we discussed technical possibilities, developed the categorization system, and recorded a comprehensive library of B♭ clarinet and bass clarinet multiphonics. This close cooperation ensured that ChordsNest reflects both practical playability and artistic potential.

In designing ChordsNest, we envisioned a tool that would offer composers, performers, and educators flexible entry points tailored to their specific needs. Composers often require a way to explore sounds intuitively, such as searching by approximate pitch content, dynamic range, or timbral characteristics to find multiphonics that fit a musical idea. Performers and educators, on the other hand, may wish to look up specific fingerings or assess technical feasibility based on difficulty levels. We aim to support these varied workflows in our designs. This approach was intended to make the exploration of multiphonics both practical and creatively engaging, accommodating a range of artistic and pedagogical contexts.

2.1. Recording and Categorization

The dataset includes 266 recorded multiphonics: 130 for bass clarinet and 136 for B♭ clarinet. Each sample was systematically categorized by its fundamental and harmonic content to document pitch, by dynamic level ranging from *ppp* to *fff*, by a difficulty rating from 1 (easy) to 3 (challenging), and by characteristic tags such as "fragile," "buzz," "spectral," and "glissando" to support detailed classification.

2.2. Interface Design and Implementation

The user interface was designed to prioritize intuitiveness and efficiency. It enables users to select the instrument (B♭ clarinet or bass clarinet), browse available multiphonics through either an interactive staff diagram or a categorized tag-based search, and hear immediate audio playback of any selection.

The Max patch integrates several core components: a *jweb* object that loads a static HTML page containing an SVG *viewbox* for interface display and interactions, a *js* object that manages queries to an SQLite database, a *js* object that handles interactions with the MaxScore instance, multiple *dict* objects that store intermediary data such as search results and active filter selections, and a *playlist~* object that handles preview playback. Together, these elements provide a responsive interface that connects the multiphonic database, visual rendering, and playback engine in real time.

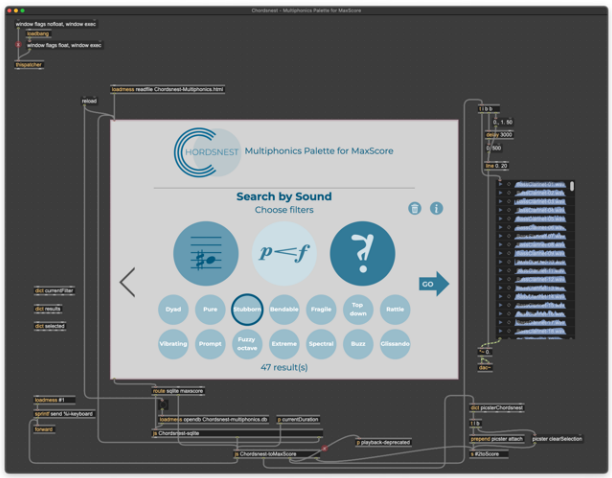


Figure 1. The ChordsNest Max patch

2.3. Database

The library of available multiphonics is provided by our collaborating clarinetist, Heather Roche. Several solutions, including JSON, were considered for storing the information, but ultimately SQLite was chosen because of its simplicity in handling search queries. This was further supported by the availability of an *SQLite* object in the default Max JavaScript implementation. The information for individual multiphonic samples is organized in an SQLite database stored in a file called "Chordsnest-multiphonics.db". The columns include:

[1]As of now, the implementation includes two instruments: the clarinet and the bass clarinet.

Field	Type / Range	Description
index	Integer (1–266)	Unique identifier for each multiphonic sample
instrument	String (“clar-bb”, “clar-bass”)	Indicates B♭ clarinet or bass clarinet
dynamicsMin	String (e.g. “pp”)	Lower end of plausible dynamic range
dynamicsMax	String (e.g. “ff”)	Upper end of plausible dynamic range
dynamicsCategory	Array of integers (1–3)	Derived category from **dynamicsMin** and **dynamicsMax**, used to facilitate interface searches
difficulty	Integer (1–3)	Indicates the relative difficulty of producing the multiphonic
tags	Array of strings	Characteristic tags (e.g. “fragile,” “buzz,” “spectral”). See **Table 3**
fingering	String	Fingering used to produce the multiphonic (e.g. “r023456f¡€”), each character corresponding to a unique key being pressed
pitchWritten	Array of strings	Perceived pitches as identified by H. Roche
pitchMidicent	Array of integers	Perceived pitches converted to midicents, derived from **pitchWritten**
pitchClassApprox	Array of integers (0–11)	Approximation of pitches to nearest MIDI key class (mod 12) for interface search, derived from **pitchMidicent**

Table 1. SQLite database structure and description

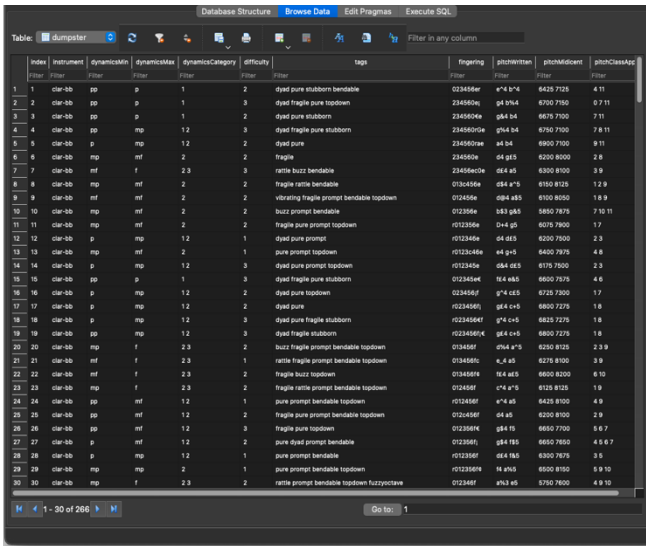


Figure 2. The SQLite database when opened with DB Browser

2.4. Search Methods

After selecting the instrument, the user is given two primary search options:

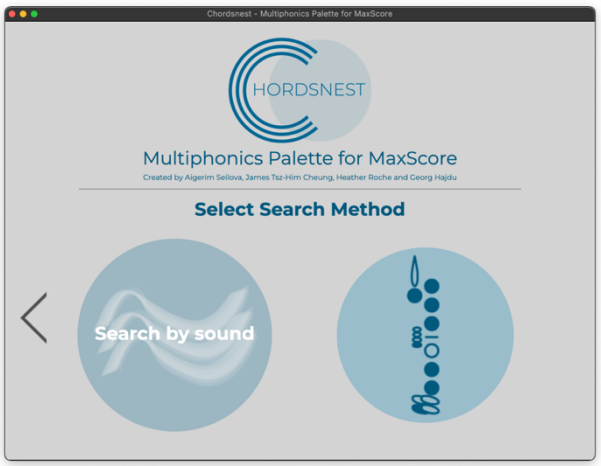


Figure 3. Interface view showing the two primary search modes: (left) search by sound, and (right) search by fingering

2.4.1.Search by Sound

This path allows filtering by the following parameters:

Parameter	Description
Pitch	Users select a pitch class (e.g., "D"). The system returns all multiphonics containing an approximate match within an octave-transposed range around that pitch class (e.g., D-half-sharp to D-half-flat).
Dynamics	Three dynamic categories are provided: soft (*pp*), middle (*mf*), and loud (*ff*). Selecting one includes all multiphonics playable within the chosen dynamic level.
Difficulty	Ranges from 1 (easy) to 3 (difficult), allowing users to filter based on playability.
Characteristic Tags	Tags help describe the sonic behavior and playing characteristics of each multiphonic, provided by Roche. These are user-selectable filters and defined in **Table 3** below.

Table 2. SQLite database structure and description

These filters can be stacked, and the number of available results with the current selected filters is shown in the bottom of the page.

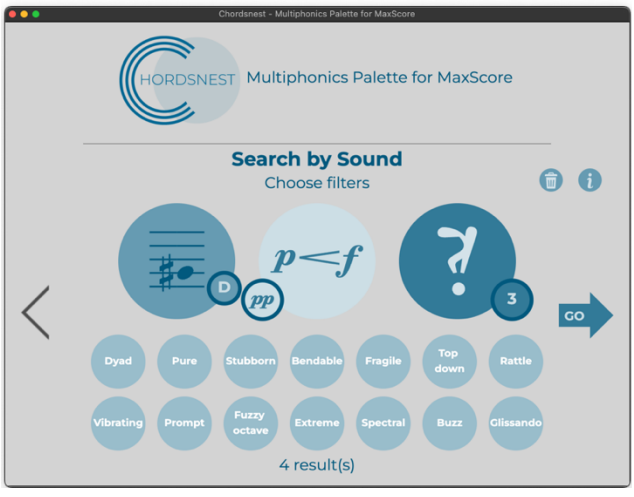


Figure 4. Search by Sound options

Tag	Description
dyad	A soft close dyad multiphonic with two prominent pitches, no more than an octave apart.
pure	A pure, clean sound.
stubborn	Requires strong diaphragm and embouchure support to sound.
bendable	Top pitch can be bent downwards.
fragile	Pitches fluctuate or don't remain stable.
topdown	Can be started on the top pitch, unlike typical bottom-up multiphonics.
rattle	Contains a rattling, buzzing texture.
vibrating	Similar to a rattle but more pulsed.
prompt	Speaks quickly with both pitches immediately audible.
fuzzyoctave	Nearly octave multiphonics (rare for clarinet).
extreme	Among the most difficult to produce reliably.
spectral	Overblown, spectral-sounding with a stable fundamental.
buzz	Very distorted and buzzy.
glissando	Allows upward pitch sliding.

Table 3. Characteristic Tags implemented in ChordsNest

2.4.2.Search by Fingering

For users already familiar with a specific multiphonic fingering, the "Search by Fingering" method enables direct access. Users interactively toggle keys to match a fingering chart; the tool then displays the corresponding multiphonic. Only one result is returned for each unique fingering.

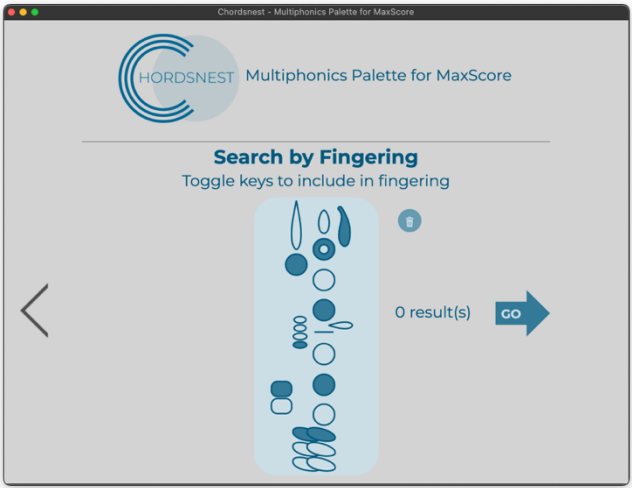


Figure 5. The Search by Fingering page

2.4.3.Result cards

Regardless of search method, the results will be shown after clicking on the “Go” button, as rectangular result cards containing all information of the multiphonic samples, containing the index, fingering, pitches (transposed), plausible dynamic range, characteristic tags and difficulty.

The fingering diagrams are generated using SVG graphics, with path data adapted from Bret Pimentel’s online fingering diagram builder [14]. CSS variables are used to control the visibility and fill state of individual keys or key groups.

The notation considers the transposition of the instrument (not concert pitch).

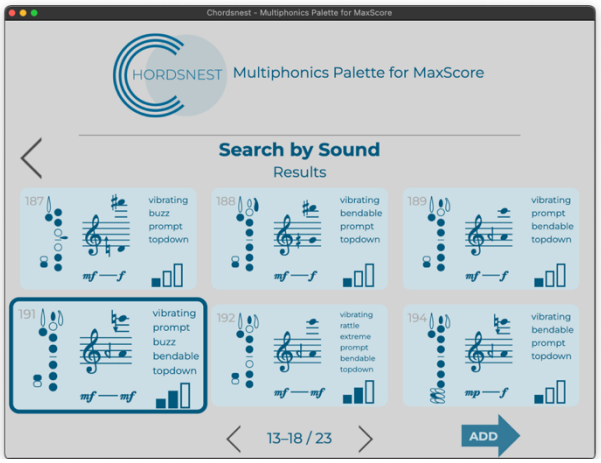


Figure 6. Search results page displaying multiphonic result cards, each containing fingering diagram, tags, and playback link

2.4.4.Preview playback

When a result card is clicked on, the multiphonic sample will be played back through Max audio as a preview. This is achieved internally in the *ChordsNest-multiphonics* patch using the Max *playlist~* object.

2.5. Integration into MaxScore Editor

ChordsNest is now included in the installation of MaxScore, accessible via the MaxScore Editor as an Entry Tool [15] within the Pitch Utility tab. Entry Tools are interfaces that provide interactive pitch input, primarily designed for microtonal composition. For example, the Scala Browser allows users to browse microtonal scales available in the Scala database. ChordsNest leverages the flexibility of custom Entry Tools, enabling the system to add notes and intervals, with attached fingering charts as SVG graphics containing playback instructions, by sending messages to the MaxScore Editor instance.

In the ChordsNest interface, after selecting and previewing a result card, clicking on “Add” inserts the selected multiphonic into the chosen measure, using the duration set previously in MaxScore. A note and its interval are generated according to the corresponding pitchMidicent values from the SQLite database.

Fingering diagrams are inserted in MaxScore as Picster elements, which are custom graphical components created using an interactive Max patch, with SVG or text-based shapes stored as JSON and linked to playback expressions. [16]. These include JSON representations of SVG graphics and Expressions for playback control. The SVG paths are based on diagrams created with Bret Pimentel’s online fingering diagram builder [14].

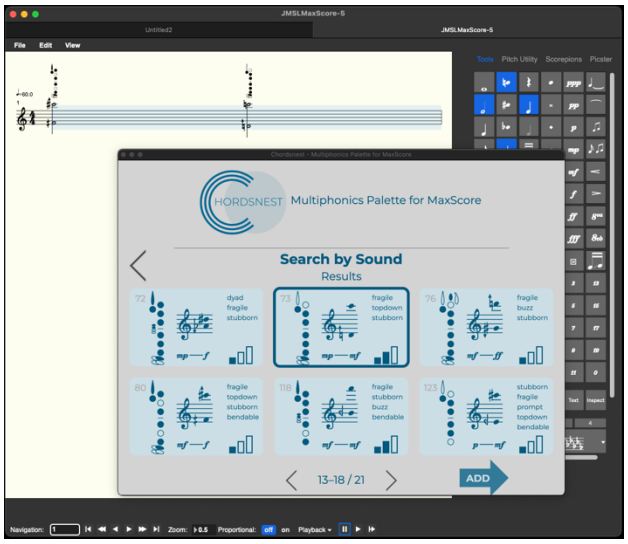


Figure 7. ChordsNest resides in MaxScore as a pitch entry tool

2.6. Playback using MaxScore Sampler

MaxScore Editor works with MaxScore Sampler via the *maxscore.makenote* object, which manages incoming MIDI messages and instrument changes. ChordsNest adopts this paradigm by utilizing a dedicated bank file containing two instruments: *BassClarinet-Multiphonics* and *Clarinet-Multiphonics*. This single bank file defines each instrument as a collection of .wav audio samples, with accompanying metadata such as root key, key and velocity zones, envelopes, and playback parameters. The samples within these banks are ordered according to their indices from the SQLite database. The MIDI pitch of each sample corresponds directly to its sample index, taking advantage of the MaxScore Sampler's ability to handle MIDI pitches above 127 (the standard limit).

When generating a multiphonic dyad, the bottom note is muted (velocity set to 0), while the top note employs the MaxScore note attribute "originalPitch" to define a MIDI playback pitch (corresponding to the sample index) that differs from the notated pitch on the score. The attached Picster Element, which displays the fingering diagram as SVG graphics, also contains an expression to facilitate a temporary instrument change in the Sampler.

This mechanism is analogous to how a pair of Picster Elements labeled "pizz" and "arco" can induce instrument changes on a string staff. A particular challenge arises because, after playing back a multiphonic sample, there is no subsequent Picster Element to instruct the Sampler to revert to the previous instrument, for example an ordinario clarinet sampler instrument. To address this, a new message "multiphonic [instrument]" has been introduced for the *maxscore.makenote* object, enabling one-time, temporary instrument changes.

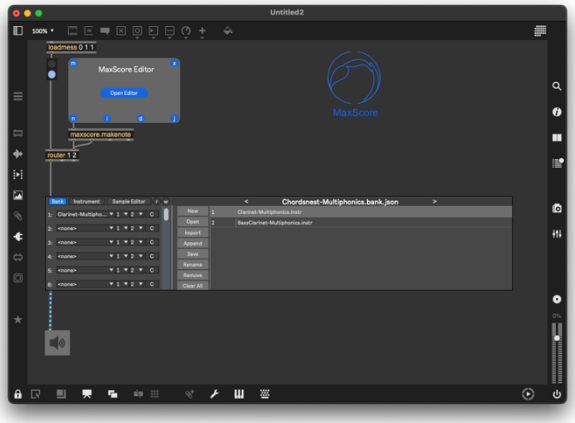


Figure 8. MaxScore combo template, with the ChordsNest clarinet multiphonics sound bank loaded

3. RESULTS

ChordsNest integrates into MaxScore and provides a comprehensive library of multiphonics with high-quality audio samples. It includes an interface for notation and playback, along with a tagging system that facilitates navigation. Playback is managed through the MaxScore Sampler, using a sound bank containing all samples recorded by Heather Roche.

3.1. Preliminary Evaluation

To assess workflow improvements, five composers tested ChordsNest while preparing short clarinet studies. They reported a significant reduction in time spent preparing multiphonic notation compared to manual entry and other software solutions. The interface was described as "intuitive" and "well integrated." Future work will include a broader usability study to quantitatively measure these improvements.

4. DISCUSSION

4.1. Implications for Composition and Performance

ChordsNest addresses a notable gap in contemporary music composition by providing a practical tool for integrating multiphonics into notation and playback workflows. Performers also benefit from access to systematically categorized and playable examples, supporting detailed preparation and interpretation. ChordsNest can also be useful for educators introducing different categories of multiphonics and their characteristics.

4.2. Potential for Expansion

The project's framework allows for future development to include additional instrumental families such as oboe, bassoon, saxophone etc. The tagging system and interface are designed to be scalable, making it possible to accommodate a broader range of multiphonic resources. While the

current implementation focuses on clarinet and bass clarinet, its applicability to other woodwind instruments remains untested and could be explored in future research, along with the integration of live-processing capabilities.

Further development could also extend MaxScore's existing message-based control, enabling programmatic generation of multiphonic notation and playback, and thus supporting algorithmic composition with extended techniques.

In addition, more features can be potentially developed, including the capability for users to save certain multiphonic samples in a "favourites" list, or for example, for users to define custom multiphonics not currently included in the fixed database.

5. CONCLUSION

The ChordsNest project represents a significant step forward in the integration of multiphonics into compositional practice. By combining a detailed library, intuitive UI, and seamless integration into MaxScore, the tool addresses key challenges in contemporary music composition. With the potential for further expansion, ChordsNest lays the groundwork for a new standard in multiphonic accessibility and usability.

Acknowledgment

We would like to acknowledge the generous support of the Gerhard Trede-Stiftung, which made this research possible.

6. REFERENCES

1. C. Beare, *Cultivating the Contemporary Clarinetist: Pedagogical Materials for Extended Clarinet Techniques*, Ph.D. dissertation, Univ. of Washington, Seattle, WA, 2021.
2. B. Bartolozzi, *New Sounds for Woodwind*, Oxford University Press, 1982.
3. E. Michael Richards, "The Clarinet of the Twenty-First Century: New Sonic Resources Based on Principles of Acoustics", E & K Publishers, 1992.
4. E. H. Fallowfield, "Cello Multiphonics: Technical and Musical Parameters," *Tempo*, vol. 74, no. 291, 2020, pp. 51–69. [Online]. Available: https://irf.fhnw.ch/. Accessed: Jul. 14, 2025.
5. H. Roche, "27 Easy Bb Clarinet Multiphonics," heatherroche.net, Sep. 13, 2018. [Online]. Available: https://heatherroche.net//. Accessed: Jun. 20, 2025
6. H. Roche, "Spectral Multiphonics (Bb and Bass Clarinet)," Multiphonic Monday, Sep. 26, 2016. [Online]. Available: https://heatherroche.net/
7. Avid Technology Inc., Sibelius User Guide, 2022. [Online]. Available: https://kb.avid.com Accessed: Jun. 15, 2025.
8. MuseScore, *MuseScore Handbook*, 2023. [Online]. Available: https://musescore.org. Accessed: Jul. 10, 2025.
9. LilyPond Development Team, "Woodwind fingering diagrams," *LilyPond Documentation*, v2.14, 2021. [Online]. Available: https://lilypond.org. Accessed: May 20, 2025.
10. Steinberg Media Technologies GmbH, "Creating custom playing techniques," Dorico Pro 5.0 Help, 2025. [Online]. Available: https://www.steinberg.help/. Accessed: Jul. 20, 2025.
11. D. C. Adams, "Multiphonics flute," *Steinberg Forums*, Apr. 2, 2021. [Online]. Available: https://forums.steinberg.net/. Accessed: Jul. 20, 2025.
12. G. Hajdu and R. Gottfried, "The Symphony in the St. Pauli Elbe Tunnel," *Contemporary Music Review*, Mar. 19, 2025, pp. 1–23.
13. A. Agostini and D. Ghisi, "A Max Library for Musical Notation and Computer-Aided Composition," *Computer Music Journal*, vol. 39, no. 2, June 2015, pp. 11-27.
14. B. Pimentel, "Woodwind Fingering Diagram Builder," [Online]. Available: https://bretpimentel.com/
15. G. Hajdu and N. Didkovsky, "MaxScore: Recent Developments," in *Proc. Int. Conf. Technologies for Music Notation and Representation (TENOR '18)*, Montreal, Canada, 2018, pp. 138–146.
16. G. Hajdu, and N. Didkovsky, "MaxScore – Current State of the Art," in *Proceedings of the International Computer Music Conference,* Ljubljana, Slovenia, 2012, pp. 156–162.

MODELING OF COGNITIVE CLASSIFICATION AND SEGMENTATION OPERATIONS DURING MUSIC LISTENING WITH MORFOS MUSIC ANALYSIS SOFTWARE

Joséphine Calandra[abc] josephine.calandra@ircam.fr
Jean-Marc Chouvel[a] jeanmarc.chouvel@free.fr
Myriam Desainte-Catherine[c] myriam@labri.fr

[a]Sorbonne Université, IReMus, UMR 8223, Paris, France;
[b]Sorbonne Université, Collegium Musicæ, IRCAM -STMS, UMR 9912, Paris, France;
[c]Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, F-33400 Talence, France

ABSTRACT

This article is part of an interdisciplinary exploration in computational musicology and mathematical theory. It presents MORFOS, a music analysis software that aims to model cognitive phenomena while listening to music. This model is based on two operations: classification and segmentation, which are operated at different time scales simultaneously. These operations, applied either to audio or symbolic music, lead to the creation of a musical form's representation that is called a multi-scaled formal diagram. This representation corresponds to a high-level notation that helps musicians better understand musical structure. As the software models a cognitive process that happens alongside the musical flow, the software also works in real-time alongside its acquisition. In this article, we present the formalisation and the implementation of classification and segmentation of the algorithm implemented in MORFOS, called the Cognitive Algorithm.Then, we present a first analysis of the software's behaviour depending on the choices made by the algorithm during these operations, defining distinct Cognitive Phases. We then present four analyses based on four segmentations of the Menuet K.545's *Rondo* of W.A. Mozart and study the computed Cognitive Phases, in order to better understand how music's structure impacts cognitive mechanisms while listening to music.

1. INTRODUCTION

In this article, we aim to gain a better understanding of the cognitive phenomena associated with musical practice, and more specifically with music listening. To this end, we present a multi-scale music analysis software that aims to model real-time classification and segmentation operations taking place simultaneously at distinct temporal scales. The result of the multi-scaled segmentation and classification processes on audio or symbolic music in MORFOS is a representation of the cognitive organisation of musical material over time, called musical form, which is represented in multi-scale formal diagrams created while the musical material is processed in the system.

Copyright: ©2025 Joséphine Calandra[abc] et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

1.1 Multi-scale music analysis software

The human brain is capable of structuring time at different scales: when we listen to a musical work, we can propose a segmentation both on the scale of the pattern and on the scale of the musical section [1].

A fundamental contribution in the representation of the multi-scale form is Lerdahl and Jackendoff's Generative Theory of Tonal Music [2]. It has been a real breakthrough in music structure analysis, which has been automated by Tojo et al. [3]. Nevertheless, their scope is limited to tonal music. Thus, the Cognitive Algorithm [4] and the MORFOS software [5] represent important advancements in the field of structure and form analysis, as they aim at analysing any kind of music genre, as the cognitive system might do. A parametrisation system influences the segmentation and classification behaviours of the software, allowing the user's choice to simulate a subjective behaviour. Segmentation and classification operations are made at different time scales, starting with elements of the smallest analysis window size, then aggregating the elements to create higher-level elements based on the parameterization.

1.2 Segmentation and classification

N. Meeùs [6] presents segmentation as a "fundamental activity of musical analysis", intrinsic to the existence of musical form. He describes the segmentation of a musical work as its "resolution into units distributed along the time axis, with duration being an essential characteristic." Meeùs compares musical segmentation to linguistic segmentation, highlighting their similar essential nature. While linguistic segmentation is possible due to the linear nature of language, which is also the case for music, the linguistic discourse — and even more so, the musical discourse — can be multidimensional and contrapuntal. For this paper, in the context of MORFOS, we set aside the complex issue of counterpoint and source separation, as these topics require specific in-depth study.

In the listening model presented by E. Bigand [7], segmentation during musical listening occurs after signal processing based on grouping principles derived from Gestalt theory. Music information retrieval addresses segmentation in order to estimate the overall structure of a musical work, and determine tempo and rhythm [8, 9], which can help identify musical metrics and structures.

Similarity between musical pieces or within a single piece makes it possible to classify works by style or recognize a music group from the very first seconds of listening thanks to specific timbres (Samson [10]).

In the field of Music Information Retrieval (MIR), classification is addressed using various criteria: Gabbolini et al. [11] introduce a similarity measure calculated from a measure of "interest" of the paths between the representations of the two entities in a graph in order to classify genre and artists. Herremans et al. [12] propose statistical models and decision trees to distinguish composers such as Bach, Haydn, and Beethoven. Sarfati et al. [13] use a clustering method to create a graph of relationships between audio files for cover detection. Dehaas et al. [14] implement a local alignment algorithm for chord recognition.

As segmentation seems to be a major process in music listening, it constitutes the first principal operation in MORFOS software. Even though some of the corresponding computer techniques obtain state-of-the-art results by using neural networks, their use presents issues related to explainability. Additionally, some clustering methods require complete prior knowledge. Consequently, MORFOS opts to avoid these techniques for segmentation and classification. Instead, the software is rule-based, offering a range of descriptors (signal-based) and grammar rules (symbolic-based) chosen by the user. These rules might be selected according to the specification of the listening model (timbral or pitch-based, distinguishing more or less the elements...) or the analytical specificities needed for such or such musical genre.

Classification is then the second principal operation in MORFOS Software. For now, even though, from a cognitive point of view, two occurrences of the same note played at two different times can be considered different dissimilar, and the same piece played by two distinct artists may be considered as two different versions, MORFOS works with a binary concept of similarity that depends on the parameters chosen by the user (choice of a similarity threshold). But this can be challenged by displaying a "similarity value" of the current object to other objects as in a Self-similarity matrix (Foote [15]).

1.3 Attention phenomenon modeling

Inspired by L. Meyer's application of the laws of the *Gestalt*, E. Narmour presents in [16, 17] the *Implication-Realisation* model. This model aims to present the mechanisms of attention based on its cognitive aspects during musical listening. Two statements concerning this model are, on the one hand, the realisation or non-realisation of the following two hypotheses: $(1) A + A \rightarrow A$ and, on the other hand, $(2) A + B \rightarrow C$ where A, B, and C are musical groupings. Assumption (1) means that the repetition of the same musical motif implies a successive repetition of that motif, while assumption (2) means that the succession of two distinct musical motifs implies a third distinct motif. The second statement defines whether or not the previously described patterns imply a closure of the musical phrase. The *System & Contrast* [18] model from F. Bimbot et al. describes the internal organisation and relationships between the properties of musical elements grouped into squares of four elements.

J.M. Chouvel presents in [4] an analysis of the behaviour of the Cognitive Algorithm, giving rise to four cognitive phases at each level, corresponding to the four possibilities of results for the two tests of classification and segmentation. It is this model that we will explain in the next sections and put into perspective with the software results.

In the following part, we present briefly the MORFOS Software. Section 3 describes the out-of-time and real-time mathematical formalisation of multi-scale formal diagrams relying on segmentation and classification operations, and Section 4 explains the implementation of the segmentation and classification algorithms. Section 5 presents the meta-analysis of MORFOS' Cognitive Algorithm with the implementation of the cognitive phases that are based on classification and segmentation behaviours, in order to better understand the cognitive model.

2. THE MORFOS MUSIC ANALYSIS SOFTWARE

Analysing a musical flow as "listened to" implies proceeding to (at least) three operations: segmentation, classification, and hierarchisation. The goal of MORFOS is to model those operations in order to represent music strategies for listening and the phenomenon of attention, i.e., for example, surprise or memorisation.

2.1 The Cognitive Algorithm

MORFOS analysis is based on the Cognitive Algorithm, which is documented in [4]. The Cognitive Algorithm was proposed as an analytic methodology by Jean-Marc Chouvel and comes from cognitive musicology [19] and semiology [20, 21].

The Cognitive Algorithm comprises two main steps. The first one is the *Paradigmatic Recognition Test* (classification test). In this test, the musical fragment currently acquired in the algorithm (at first, the analysis of the musical element of the size of the analysis windows for signal, or the parsing of the first annotation for symbolic representation) is compared with previously known fragments of the same level and classified according to this comparison (the first fragment is by default considered as a new element, as the methodology asserts that no prior knowledge is necessary at this stage).

Next, a second test follows: the *Syntagmatic Recognition Test* (segmentation test) determines whether or not the group of fragments that have just been heard constitutes a fragment of the upper level. If this is the case, both tests are repeated at the higher level, a higher level being a level in which the fragments are longer, as they are the result of a concatenation of fragments from the lower level. Thus,

each level leads to the realisation of a specific diagram called *formal diagram*.

This algorithm is used as a process for musicologists and had never been implemented. The MORFOS software is based on the concepts provided by this analytic method, while adapting to operational needs. For example, the operations of classification and segmentation have been reversed, as the first operation in the system is the segmentation of the audio stream or the parsing of the symbolic stream.

2.2 Formal Diagram and Multi-scale Formal Diagrams

A Formal Diagram (or Paradigmatic Diagram) is the representation of *musical materials* according to *time* and can be represented simultaneously at different time scales. We show an formal diagram obtained automatically with MORFOS at a two-bar musical scale of Wolfgang Amadeus Mozart Menuet's *Rondo K.545* in Figure 1.

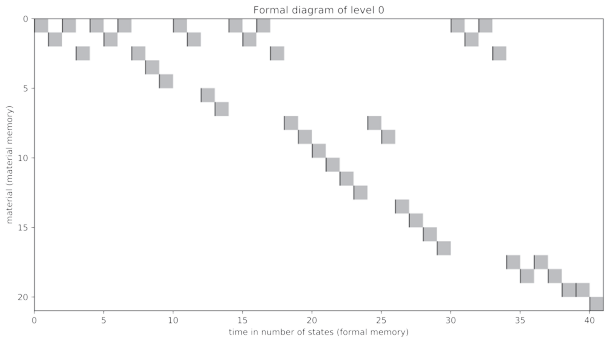


Figure 1. Automated Formal Diagram obtained with MORFOS of W.A. Mozart Menuet's *Rondo K.545* at two bars musical scale, from a symbolic representation.

Materials are defined as classes of substitutable elements. For instance, the concept of the note $A3$ represents a material, but materials can encompass various types, such as musical phrases in some musical styles, or even noises. This concept is closely related to the idea of paradigm. An *object* then refers to a specific occurrence of a material in a musical work.

Starting with the smallest time scale diagrams, the Cognitive Algorithm constructs larger time scale diagrams, so that the sequences of lower-level objects constitute a single higher-level object. For example, a sequence of notes in a formal diagram constitutes a phrase for the higher-scale formal diagram, and a sequence of phrases constitutes a part of the higher-scale formal diagram.

The superimposition of Formal Diagrams obtained at different time scales is called *Multi-scale formal diagram*. A representation of W.A. Mozart's *Rondo K.545* Multi-scale Formal diagram is shown in Figure 2 and more deeply explained in [22].

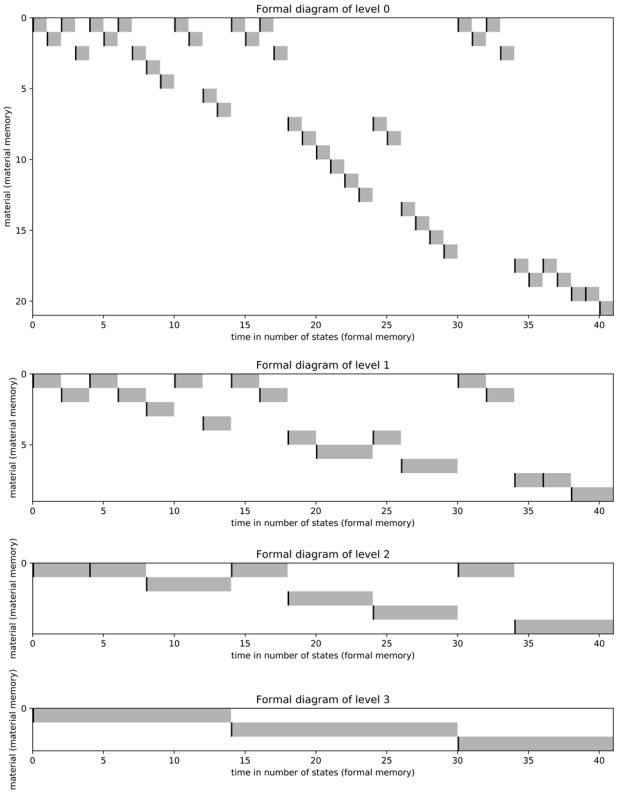


Figure 2. Automated Multi-scale formal diagram of W.A. Mozart's *Rondo K545* obtained with MORFOS. Level 0 is at (about) two bars scale, Level 1 is at four bars scale, Level 2 is at height bar scale, and Level 3 is at the sixteen bars scale. Each scale is automatically built according to the construction of the lower one.

3. FORMALISATION OF THE MULTI-SCALE FORMAL DIAGRAM

This section describes the out-of-time and real-time formalisation of the multi-scale formal diagrams in MORFOS.

3.1 Out-of-time formalisation

Let us first note that the input and output at a given level of the MORFOS Algorithm are actually the same data structure. The input of an instance of the MORFOS algorithm can thus be the output of a lower-level instance.

We thus define $r^{(n)}$ as the *representation of level n* equivalent to the *formal diagram of level n* as a word

$$r^{(n)} = x_1^{(n)}...x_i^{(n)}...x_{l_n}^{(n)}$$

with $x_i^{(n)}$ letters defined on the ordered alphabet $\mathcal{A}^{(n)}$.

The symbolic representation is then the succession of characters corresponding to the objects (that can be either symbolic or audio data) represented by the labels of the associated initial materials. These labels have no semantic interest. The *index* of a labeled object in a word $r^{(n)}$ is also defined as the number of the object's location in the word r: it is thus indexed by its index. The notation $x_i = r[i]$ is then accepted.

The level n representation is segmented into a sequence of fragments. The segmentation operation $sgm(r^{(n)}, cs(n))$ takes as arguments the initial representation $r^{(n)}$ and a set of segmentation criteria cs depending of the level n. In fact, the segmentation criteria can vary from one time scale to another and therefore from one level to another. Segmentation criteria may include, for example, a change in a sequence of similar elements at the level of a few milliseconds and the return to the beginning of a sequence in higher levels. Let $s^{(n)}$ be a *segmented representation of level n* which is the grouping of letters from $\mathcal{A}^{(n)}$ into factors $w_j^{(n)}$ belonging to $\mathcal{A}^{(n)*}$ by associativity of the monoid $\mathcal{A}^{(n)}$ on $\mathcal{A}^{(n)*}$ such that

$$s^{(n)} = sgm(r^{(n)}, cs(n)) = w_1^{(n)}...w_j^{(n)}...w_{l_{n+1}}^{(n)}$$

The internal similarity between the fragments $w_j^{(n)}$ of $s^{(n)}$ is then calculated. For that purpose, a classification operation $cls(s^{(n)}, cc(n))$ is defined, which takes as arguments the segmented representation of level n $s^{(n)}$ and a set of classification criteria cc depending on the level n. A classification criterion can be a similarity threshold, and the operation of similarity (Euclidean distance, cosine distance, etc.) to apply to the spectrum, descriptor, or symbolic string. Thus, the *level n+1 representation* is the word $r^{(n+1)}$ obtained as follows

$$r^{(n+1)} = cls(s^{(n)}, cc(n)) = cls(sgm(r^{(n)}, cs(n)), cc(n))$$

$$= x_1^{(n+1)}...x_j^{(n+1)}...x_{l_{n+1}}^{(n+1)}$$

where the $x_j^{(n+1)}$ are letters defined on a new ordered alphabet $\mathcal{A}^{(n+1)}$.

3.2 Real-time formalisation

The formal diagrams are, in fact, not obtained in a single block, but in real-time, i.e., as the signal is acquired window after window in MORFOS and the initial representation is parsed in the MORFOS Algorithm.

Thus, a multi-scale formal diagram is defined by a sequence of states represented by the word $R = X_0.X_1...X_n$ with n the size of the word corresponding to the number of formal diagrams created. It is obtained by the concatenation of the triplets $X_j = (r^{(j)}, s^{(j)}, k^{(j)})$, where $r^{(j)}$ is the representation of level j, and $s^{(j)}$ is the segmented representation of level j. A marker of level j, $k^{(j)}$ is also defined, corresponding to the index of the letter where the last segmentation took place.

First, $R = \epsilon$ is initialised with the empty word. There are four operations that allow to change the state of R:

- the *creation of a new level* $X_n = (\epsilon, \epsilon, 0)$ with the operation nlc,
- the *real-time acquisition* at level 0 noted rta,
- the *real-time segmentation* at level n $rts(n)$,
- the *real-time classification* at level n $rtc(n)$ towards level $n + 1$

Not only do the $rts(n)$ segmentation and $rtc(n)$ classification operations apply at level n, but they can also take into account criteria depending on the level to which they apply. As a reminder, the real-time classification $rtc(n)$ occurs between levels n and $n + 1$ as it classifies the succession of objects of level n into a single object of level $n + 1$.

We also use the notations "." for the concatenation between two words or between a word and a letter, and $[r^{(j)}]_{k_1,k_2}$ for the subword of $r^{(j)}$ going from index k_1 to index k_2, included.

When acquiring a new object x at level 0, we concatenate $r^{(0)}$ with the new object x which comes from the flow:

$$X_0X_1...X_n = (r^{(0)}, s^{(0)}, k^{(0)})X_1...X_n$$

$$\xrightarrow[x]{rta} (r^{(0)}.x, s^{(0)}, k^{(0)}).X_1..X_n$$

After the acquisition, a segmentation operation can be carried out if relevant.

During segmentation, we concatenate $s^{(j)}$ with the fragment composed of all letters acquired since the last segmentation (indicated by the index corresponding to the letter $k^{(j)}$) up to the previously acquired letter of index $l-1$ where l is the length of the word of the representation $r^{(j)}$:

$$X_0...(r^{(j)}, s^{(j)}, k^{(j)})...X_n$$

$$\xrightarrow{rts(j)} X_0...(r^{(j)}, s^{(j)}.([r^{(j)}]_{k^{(j)},l-1}), l)...X_n$$

After each segmentation, we apply a classification operation ctr on the last factor concatenated to $s^{(j)}$. We then concatenate the corresponding label obtained in $\mathcal{A}^{(j)}$ to $r^{(j+1)}$. If we denote w as the last fragment of our segmented representation, we then have:

$$X_0...(r^{(j)}, s^{(j)}.w, k^{(j)})(r^{(j+1)}, s^{(j+1)}, k^{(j+1)})...X_n$$

$$\xrightarrow{rtc(j)} X_0...(r^{(j)}, s^{(j)}.w, k^{(j)})(r^{(j+1)}.rtc(w), s^{(j+1)}, k^{(j+1)})...X_n$$

We see that the acquisition operation rta is in reality a special case of the classification operation $cls(n)$ at level 0 with the classification function being the identity function.

It is possible that level $j + 1$ does not yet exist. Before proceeding with the classification operation, it is therefore necessary to create a new level with the operation nlc, which corresponds to the concatenation of the formal multi-scale diagram of size $n - 1$ with the formal diagram $X_n = (\epsilon, \epsilon, 0)$.

$$X_0X_1...X_{n-1} \xrightarrow{nlc} X_0X_1...X_{n-1}.(\epsilon, \epsilon, 0)$$

The final state of the word is obtained when all of the information at level 0 is acquired and the classification has been carried out at the highest level.

Thus, the set of n formal diagrams that constitute the multi-scale formal diagram is the ${r^{(j)}}_{j\in[0,n]}$ after having carried out the entirety of the changes of state.

The order of operations is shown in the diagram in Figure 3. The algorithm always starts with a succession of acquisitions at level 0, before segmenting. The segmentation is always followed by a classification, with an intermediate creation of a new level if it does not already exist. After classification, there is either segmentation at the higher level or acquisition at level 0.

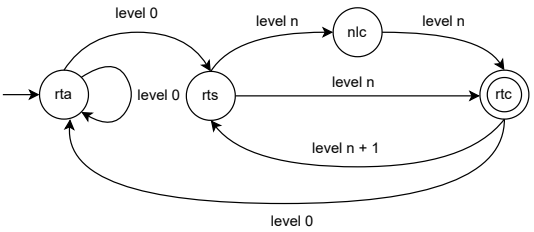


Figure 3. MORFOS Algorithm operations automaton. Each state corresponds to an operation, and the arcs correspond to the level of state affected by the operation to which the arrow points.

3.3 Example of W.A. Mozart's *Rondo* K.545

Thus, on the example of W.A. Mozart's *Rondo* presented in Figure 2, we have the following changes of state, according to the given segmentation and classification rules.

The multi-scale formal diagram is initially represented by the empty word ϵ. We want to acquire a new item at level 0. To do this, level 0 must first be created with the nlc operation:

$$\epsilon \xrightarrow{nlc} X_0$$

A new object a is then acquired at level 0:

$$(r^{(0)}, s^{(0)}, k^{(0)}) = (\epsilon, \epsilon, 0)$$

$$\xrightarrow[a]{rta} (r^{(0)}.a, s^{(0)}, k^{(0)}) = (a, \epsilon, 0)$$

There is no need to segment, so the next object, b, is acquired:

$$(a, \epsilon, 0) \xrightarrow[b]{rta} (ab, \epsilon, 0)$$

In the same way, let us suppose that the algorithm continues to acquire the next object, a:

$$(ab, \epsilon, 0) \xrightarrow[a]{rta} (aba, \epsilon, 0)$$

Here, we suppose that a segmentation operation is performed between b and a:

$$(aba, \epsilon, 0) \xrightarrow{rts(0)} (aba, (ab), 3)$$

In order to proceed with the classification of (ab) at the higher level, we must first initialise the higher level:

$$(aba, (ab), 3) \xrightarrow{nlc} (aba, (ab), 3).(\epsilon, \epsilon, 0)$$

The classification operation is then carried out on (ab), which is concatenated to $r^{(1)}$:

$$(aba, (ab), 3).(\epsilon, \epsilon, 0) \xrightarrow{rtc(0)} (aba, (ab), 3).(\text{A}, \epsilon, 0)$$

where A is the label given to the higher-level object.

And so on until the acquisition of all materials of level 0.

4. IMPLEMENTATION

In this section, we describe the implementation of the two specific operations of segmentation *rts* and classification *rtc*.

4.1 The data structures

In this subsection, we present three data structures necessary to understand the following algorithms.

- The *ongoing concatenated object of level n* corresponds to the concatenation of the objects acquired since the last segmentation.
- The *oracle of level n* is based on the *Variable Markov Oracle*, which is an automaton presented by Allauzen et al. [23] for compression and used for music improvisation by G. Assayag, C. Wang and S. Dubnov [24, 25]. It quickly finds the longest string of similar characters previously heard in the software. Each internal link to a state represents an object, and each state corresponds to the observation at time t of a discrete time series corresponding to the analysed stream.
- The *Material Memory of level n* constists of the *History Table of level n*, which contains the pairs of objects of level n and their corresponding objects of level n-1, and the *self-similarity matrix of level n*, which contains the similarity value between all acquired materials of level n.

These data structures are more precisely documented on [5].

4.2 The segmentation algorithm

The segmentation in MORFOS is performed by the function `segmentation(level n, object c, criteria cs): boolean` which returns 1 if there is segmentation and 0 otherwise. The level n provides access to all the data structures of the current level n and the c object provides information about the last object acquired.

The segmentation criteria correspond to a set of rules selected before the analysis and used to determine whether or not segmentation exists. These criteria are applied either to the signal, directly to the spectrum, or to descriptors (MFCC) or to symbols (MIDI or character strings). From a signal perspective, segmentation can be performed based on the dissimilarity computed either on frequency or dynamics. From a symbolic perspective, grammar rules are implemented by analysing changes in sequences of multiple same characters, the return to the beginning of a sequence, or length of sequences, to name just a few rules among many others.

The segmentation rules are designed in a modular way so that developers can add new ones.

4.2.1 Implementation of segmentation rules

Segmentation rules can be of three types: *mandatory*, *prohibited*, or *bivalent*. Mandatory segmentation rules necessarily imply segmentation at the moment preceding the acquisition of the current object. They therefore return the value 1 if there is segmentation between the acquired object x_{i+1} and the last object x_i. For example, `rule_1_similarity_word(ms_oracle,`

`level)` is a mandatory segmentation rule that enforces the segmentation if the ongoing concatenated object of level *level* is considered similar to a previously segmented factor of the same level.

Prohibited segmentation rules forbid segmentation between the currently acquired object and the previous one. They therefore return the value 1 if there is no segmentation between the acquired object x_{i+1} and the last object x_i. For example, `rule_2_validated_hypothesis(ms_oracle, level)` is a prohibited segmentation rule that forbids segmentation if the ongoing concatenated object of level *level* is the beginning of a previously segmented factor of the same level.

Bivalent segmentation rules induce segmentation prohibition constraints and segmentation obligation constraints. These are implemented independently in two rules: mandatory segmentation and prohibited segmentation rules. For example `rule_6a_low_bound(ms_oracle, level)` and `rule_6b_high_bound(ms_oracle, level)` are respectively the prohibited segmentation and mandatory segmentation rules of the bivalent segmentation rule that forbid the segmentation if the ongoing concatenated object of level *level* has a length inferior to a predefined integer, and enforce a segmentation if the ongoing concatenated object of level *level* is of length superior to another predefined integer

When the user selects the rules for the analysis, they are added to the *list of rules* called by the rules manager.

4.2.2 Rules manager

The rules manager calculates the Boolean values associated with the result of the execution of each of the segmentation rules in the given context. This produces a set of booleans $\{r_i\}_{1 \leq i \leq m_1}$ where m_1 is the number of rules selected for the analysis, with each bivalent rule being divided into two rules.

To determine whether segmentation is appropriate, the following logical formula is calculated:

$$b = (\bigvee_{r_i \in \mathcal{O} \cap \mathcal{S}} r_i) \wedge (\bigwedge_{r_j \in \mathcal{I} \cap \mathcal{S}} \neg r_j)$$

Where $\mathcal{O}$ is the set of mandatory rules, $\mathcal{I}$ is the set of forbidden rules, $\mathcal{S}$ is the set of selected rules, and b is the boolean result of the logical formula.

4.3 The classification algorithm

The classification in MORFOS is performed by the function `classification(level n, criteria cs): material`. This function adds the ongoing concatenated object at the current level to the higher-level oracle. To add this object to the oracle, it is compared in an optimised manner according to the construction of the oracle.

The classification criteria correspond to a set of rules selected before the analysis and are applied either to the signal or to symbols. From a signal perspective, classification is performed by computing, for example, cosine similarity or Euclidean distance on the spectrum or descriptors. From a symbolic perspective, strict equality or Needleman-Wunsch alignments [26] are computed on sequences.

The algorithm 1 is used for this purpose. To find the material corresponding to the newly acquired object, the ongoing concatenated object of level n is compared with the factor of level n associated with the previous objects of level $n+1$, accessed by the construction of the oracle of level $n+1$, and accessible from the history table in the material memory of level n. The similarity calculation `similarity(`w_i`,` $o_c^{(n)}$`, cc)` is applied to the corresponding factor and the object being formed at level n. Furthermore, let us assume that the material c obtained is new. In that case, the self-similarity matrix of the level n is updated with all the similarity values calculated previously and stored during the call to the $similarity$ function. Thanks to the construction of the oracle, all existing materials are compared when a new one is created.

Algorithm 1 classification(level n, criteria cs): object

For x_i accessed in $oracle^{(n+1)}$ by going back and retrieving the suffix links starting from the last state:
  $w_i \leftarrow Th^{(n)}(x_i)$
  $s \leftarrow similarity(w_i, o_c^{(n)}, cc)$
  If $s = 1$:
    $c \leftarrow x_i$
    end For
  Else:
    $c \leftarrow create_material()$
    $update(S^{(n)})$
return c

Finally, the function `similarity(factor` f_1`, factor` f_2`, classification criteria` cc`):boolean` calculates the similarity between two factors f_1 and f_2 based on classification criteria cc which are the classification rules, corresponding to similarity measures, selected by the user. Then, the algorithm checks if the value obtained exceeds a similarity threshold ss, also defined as a parameter. This amounts to taking the integer part greater than the difference between the similarity value and the threshold.

$$similarity(f_1, f_2, cc) = \lceil \frac{\sum_{sim_k \in cc} sim_k(f_1, f_2)}{Card(cc)} - ss \rceil$$

5. SECOND-ORDER ANALYSIS AND ATTENTION PHENOMENA

In this section, we analyse the algorithm's behaviour in order to better understand the phenomena of attention when listening to music. To do this, we rely on the notion of *cognitive phases* presented by J.M. Chouvel in [4].

5.1 Cognitive phases

Cognitive phases correspond to the characterisation of behaviours related to the handling of the information flow in

the Cognitive Algorithm, i.e., the audio or symbolic flow during music listening.

These cognitive phases are linked to the results of the two main tests of the Cognitive Algorithm as originally presented, namely the paradigmatic test and the syntagmatic test.

These phases make it possible to clarify specific cognitive states:

Integration: Integration occurs when there is a sensation of continuity; when the current object is concatenated with the objects heard since the last segmentation, and a higher-level object is consequently under construction.

Retention: J.M. Chouvel writes in [27] that *"The retentional memory required by form must allow the memorisation of elementary events"*. Thus, retention specifically refers to the memorisation of information in the most compressed form possible, with minimal information loss. It involves memorising each new material. Retention occurs when the classification test fails (a new material is analysed).

Realisation: In the algorithm, realisation occurs when an object has been recognised at the current level and allows a hypothesis that the whole corresponding object at the superior level will occur. The following object at the current level is then expected by the perception. Realisation occurs when the classification test is valid.

Acquisition/Recursion: Acquisition refers to the grouping phenomenon of recently heard objects to form a higher-level object, with the aim of processing this new object as a whole. Recursion is the invocation of the Cognitive Algorithm at the higher level on this newly acquired object. There is therefore acquisition at the higher level and recursion at the algorithm level when *completion* occurs, meaning the segmentation test is valid.

Expectation of the Unknown: Expectation of the unknown arises when no hypothesis can be made about the *successor object*. The successor object refers to the object acquired at the current level after the object currently being acquired. Otherwise, the listener is in the *expectation of the known*. However, even if the known is expected, the actually acquired object may not correspond to the expected one. This gives rise to possibilities of *satisfied prediction* or *unsatisfied prediction*. A satisfied prediction occurs when the subsequently acquired object matches the emitted hypothesis, and an unsatisfied prediction otherwise.

In MORFOS, the classification and segmentation tests are performed in reverse order compared to Chouvel's model. We consider segmentation as prior to classification: before classifying an element, it must first be distinguished. At the algorithmic level, the software first performs automated signal sampling, or preliminary segmentation, that provides a sequential symbolic representation. In reality, these two tests likely occur simultaneously: the algorithm's implementation shows that classification immediately follows segmentation, but similarity computations are performed beforehand to determine segmentation. This deserves further discussion.

Thus, the modeling of cognitive phases in our software is presented in Figure 4. We introduce the term *object currently being acquired* x_t^j to denote the object currently acquired at level j and time t. It should not be confused with the *ongoing concatenated object*, which corresponds to the concatenation of objects at a given level in order to build a higher-level object. The *successor object* is the object that immediately follows the currently acquired object at the same level, i.e., the object x_{t+1}^j acquired at time $t+1$ at level j. We may also extend this terminology to refer to the *higher-level acquired object* x_k^{j+1}, which is the higher-level object closed by x_t^j, and the *higher-level successor object* x_{k+1}^{j+1}, which is the higher-level object initiated by x_{t+1}^j.

An *iteration* of the algorithm involves acquiring an object at a given time t and level j and applying the segmentation and classification tests to it. Two iterations are shown in Figure 4. The first (top) corresponds to the *segmentation/classification* iteration at level $j-1$ giving the object x_t^j, while the second (bottom) corresponds to the *segmentation/classification* iteration during the acquisition of object x_{t+1}^j.

The two tests relevant for determining the cognitive phase are highlighted in dark gray.

A major difference between our algorithm and the one proposed by J.M. Chouvel is that the segmentation test validates or invalidates a segmentation between the currently acquired object and the previous one. It does not indicate that the current object *closes* the acquired higher-level object, but rather that the successor object *initiates* the successor higher-level object.

The first is the classification test, which corresponds to the classification of the factor $w_t^{(j-1)}$ at level $j-1$: it labels this factor into a current acquisition object $x_t^{(j)}$ at level j. The segmented object that yields this factor is not our focus here. If this label matches that of a material seen previously, the classification test is considered *valid*. Otherwise, the test is *invalid*.

The second test is the segmentation test, which determines whether the object $x_t^{(j)}$ constitutes a closing element. This means that during the acquisition of its successor $x_{t+1}^{(j)}$, segmentation occurs into a factor $w_k^{(j)} = (x_{t-i}^{(j)} ... x_t^{(j)})$. If segmentation occurs, the test is said to be *valid*. Otherwise, when there is no segmentation, the segmentation test is *invalid*.

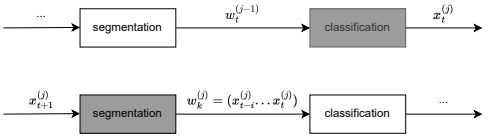


Figure 4. Modeling of cognitive phases.

We now present the four cognitive phases.

5.1.1 Cognitive Phase 1

Cognitive phase 1 corresponds to an invalid classification test and an invalid segmentation test. An invalid classification test at level j means that the currently acquired object x_t^j is not similar to any previously heard object at level j:

thus, the corresponding material is memorised. Furthermore, there is *integration* in the sense that the object is concatenated with the objects heard since the last segmentation (construction of the factor $w_k^j = x_{t-i}^j ... x_t^j$, where $t-i > 0$ is the index of the last segmentation), and the listener is in the process of constructing a higher-level object. However, during the acquisition of x_{t+1}^j, there is no segmentation: x_t^j does not close the higher-level object under construction, and nothing happens at the upper level.

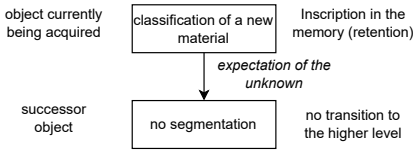


Figure 5. Phase 1.

In terms of hypotheses: since the object $x_t^{(j)}$ corresponds to new material acquired at the current level, no specific hypothesis is made regarding the object to follow $x_{t+1}^{(j)}$, nor for the higher-level object $x_k^{(j+1)}$. More precisely, a novelty hypothesis is put forward: the listener expects the unknown.

5.1.2 Cognitive Phase 2

Cognitive phase 2 corresponds to an invalid classification test and a valid segmentation. Thus, memorisation and retention occur in the memory of the currently acquired object x_t^j as in phase 1. However, there is now segmentation before x_{t+1}^j during its acquisition, which means that the object under construction $w_k^j = x_{t-i}^j ... x_t^j$ is acquired and therefore labeled (through a classification operation) as a higher-level object, with a recursive call to the MORFOS algorithm.

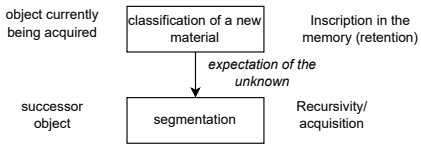


Figure 6. Phase 2.

As in cognitive phase 1, new material is acquired at the current level, so that a hypothesis of novelty arises concerning the object to come. Moreover, the system acquires probably new material at the higher level (unless this is considered a negligible variation) for the object $x_k^{(j+1)}$, and a novelty hypothesis arises concerning the following object $x_{k+1}^{(j+1)}$. The listening subject therefore expects the unknown, both at the current level and at the higher level.

5.1.3 Cognitive Phase 3

Cognitive phase 3 corresponds to a valid classification test and an invalid segmentation. The currently acquired object x_t^j is similar to a previously heard object. The associated time instant t is memorised, but the corresponding material is not memorised again. The object is concatenated to the ongoing concatenated object, and there is a *realisation* phase: potential materials are concretised in memory. As in phase 1, however, there is no segmentation after acquiring the successor object x_{t+1}^j.

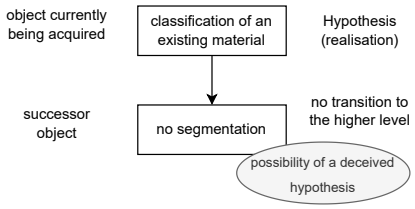


Figure 7. Phase 3.

Recognition in x_t^j of already perceived material allows hypotheses to be made concerning the successor object x_{t+1}^j expected at the current level. Hypotheses are based on objects that succeeded previous occurrences of the material corresponding to x_t^j. This is also made possible by the fact that there is no segmentation after $x_t^{(j)}$. Indeed, if there were segmentation after $x_t^{(j)}$, then the object x_{t+1}^j would correspond to the beginning of a new object at level j, and no hypothesis would be made (case of phase 4). Here, hypotheses are made: the listening subject expects the known. However, it is possible that the object x_{t+1}^j does not correspond to the expected materials: a surprise phenomenon will then occur.

In our modeling, it is the structure of the Variable Markov Oracle that allows us to find previous occurrences and their successors, which are potential candidates for $x_{t+1}^{(j)}$.

It should be noted that at this stage, hypotheses can also be made about the higher-level object $x_k^{(j+1)}$: either the object under construction w_k^j is the prefix of one or more higher-level objects already seen, and these objects are hypotheses of realisation, or the object currently being constructed does not correspond to any prefix and the unknown is therefore expected at the higher level. By moving up the levels, the scope of anticipation can therefore potentially be increased.

5.1.4 Cognitive Phase 4

Cognitive phase 4 corresponds to a valid classification test (realisation) and also a valid segmentation test (acquisition at the higher level and recursive call of the algorithm).

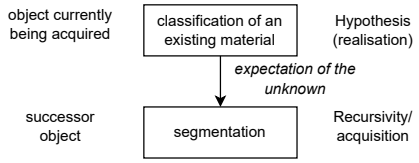


Figure 8. Phase 4.

As in phase 3, acquiring a recognised object x_t^j allows hypotheses to be made about the object $x_{t+1}^{(j)}$, but in real-

ity, there is segmentation between these two objects. The object $x_{t+1}^{(j)}$ is thus the beginning of a new factor w_{k+1}^{j} and no hypothesis is made at level j: the unknown is expected.

At the higher level, however, it is possible that the obtained object x_k^{j+1} corresponds to already heard material at level $j+1$: thus, a hypothesis can be made about the successor object x_{k+1}^{j+1}.

5.2 Analyses

The computation of the phases in real time has been implemented in MORFOS. We computed them for different types of segmentation on Mozart's *Rondo*:

- no segmentation;
- factors of size two are segmented;
- random segmentation;
- set of relevant segmentation rules;

The corresponding multi-scale formal diagrams are represented respectively in Figures 9, 10, 11, and 12.

5.2.1 No Segmentation

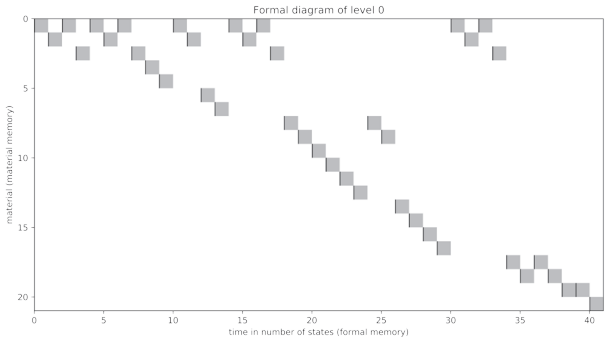


Figure 9. Automated (Multi-scale) formal diagram of W.A. Mozart's *Rondo K545* obtained with MORFOS with no segmentation. As there is no segmentation, there is only one level of hierarchy.

Table 1 shows the phases corresponding to an analysis without segmentation as shown in the multi-scale formal diagram Figure 9. The first row corresponds to the states acquired at level 0, and each additional row corresponds to a level and contains the phases corresponding to the acquisition of the states associated with the segmentation of the state at level 0. In this case, we observe only phases 1 and 3: phase 1 appears during the acquisition of a new material and phase 3 during the return to an already seen material, but there is never any segmentation except for the last material, which is considered segmented by default (and was never seen, which yields phase 2).

There are only phases 1 and 3, so there is alternation of retention and realisation in the memory. Moreover, there is no transition to the higher-level, so the number of concatenated objects in the ongoing concatenated object of level 0 keeps increasing. Therefore, a hypothesis is that it increases the cognitive load as no compression of the information into higher level segments is made.

5.2.2 Segmentation every 2 objects

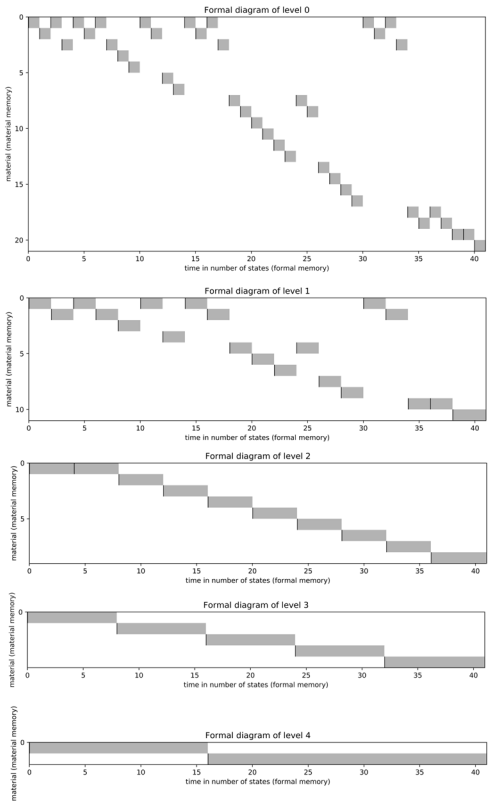


Figure 10. Automated Multi-scale formal diagram of W.A. Mozart's *Rondo K545* obtained with MORFOS, where factors of size two are segmented.

Table 2 presents the cognitive phases obtained during an analysis where size-two factors are segmented. Segmenting every two objects presents the cognitive phase 1 or 3 every one time, and phase 2 or 4 every other time, depending on whether the acquired object was already in memory or not.

Starting from level 2, there are almost only phases 1 and 2 due to the classification of objects from materials that have never been seen before; therefore, there is constantly retention at the acquisition of objects at levels 2 and 3. Nevertheless, as new materials constantly appear, there is always an expectation of the unknown.

5.2.3 Random segmentation

We present in the table 3 the cognitive phases associated with a random segmentation as shown in the multi-scale formal diagram Figure 11. The phases obtained also seem random between 1 and 3 when there is no segmentation (depending on whether the object has already been acquired or not) and between 2 and 4 if there is random segmentation.

Random segmentation favors the non-recognition of the objects starting at level 1, so there are only phases 1 and 2 at level 1 and higher levels. There is thus only retention in the memory except for level 0.

level	a	b	a	c	a	b	a	c	d	e	a	b	f	g	a	b	a	c	h	i	j	k	l	m	h	i	n	o	p	q	a	b	a	c	r	s	r	s	t	t	u
0	1	1	3	1	3	3	3	3	1	1	3	3	1	1	3	3	3	3	1	1	1	1	1	1	3	3	1	1	1	1	3	3	3	3	1	1	3	3	1	3	2

Table 1. Cognitive phases computed at each level of segmentation (here, only level 0 is computed) according to W.A. Mozart's *Rondo K545* multi-scale formal diagram obtained with MORFOS with no segmentation.

level	a	b	a	c	a	b	a	c	d	e	a	b	f	g	a	b	a	c	h	i	j	k	l	m	h	i	n	o	p	q	a	b	a	c	r	s	r	s	t	t	u
0	1	2	3	2	3	4	3	4	1	2	3	4	1	2	3	4	3	4	1	2	1	2	1	2	3	4	1	2	1	2	3	4	3	4	1	2	3	4	1	3	2
1		1		2		3		4		1		4		1		4		3		2		1		2		3		2		1		4		3		2		3		2
2				1				4				1				2				1				2				1				2				1				2
3								1								2								1								1								2
4																1																										2

Table 2. Cognitive phases computed at each level of segmentation according to W.A. Mozart's *Rondo K545* multi-scale formal diagram obtained with MORFOS, where factors of size two are segmented.

level	a	b	a	c	a	b	a	c	d	e	a	b	f	g	a	b	a	c	h	i	j	k	l	m	h	i	n	o	p	q	a	b	a	c	r	s	r	s	t	t	u
0	1	2	3	1	3	3	3	4	1	1	3	3	2	1	3	4	3	4	1	2	1	2	1	1	3	4	1	2	1	1	4	3	3	3	2	1	4	3	2	3	2
1		1						2					1			2		1		2		1				2		1			1				2		1		1		2
2								1								2				1						1									1						2
3																1																										2

Table 3. Cognitive phases computed at each level of segmentation according to W.A. Mozart's *Rondo K545* multi-scale formal diagram obtained with MORFOS with random segmentation.

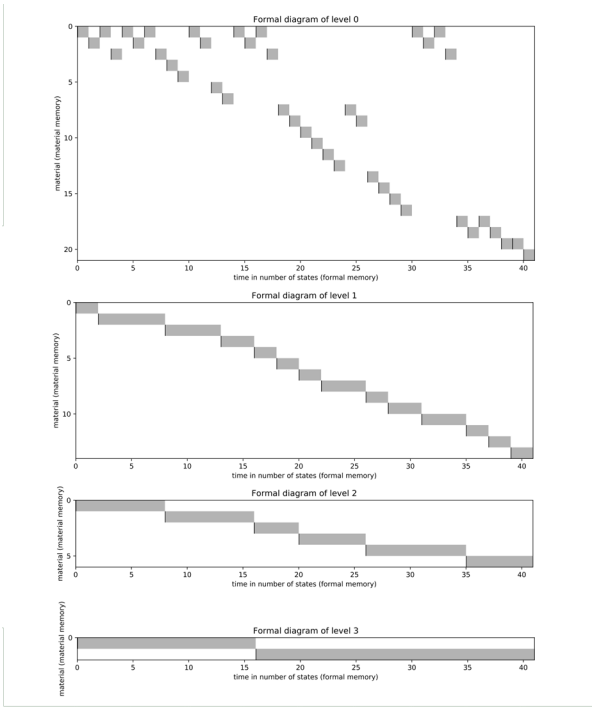


Figure 11. Automated Multi-scale formal diagram of W.A. Mozart's *Rondo K545* obtained with MORFOS with random segmentation.

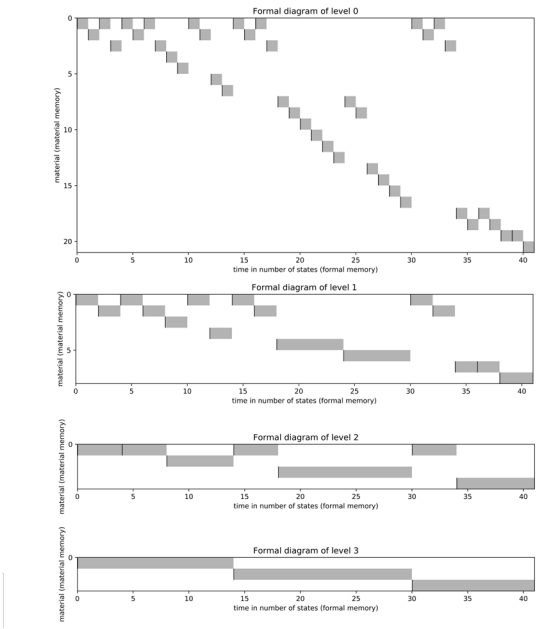


Figure 12. Automated Multi-scale formal diagram of W.A. Mozart's *Rondo K545* obtained with MORFOS with a set of relevant segmentation rules described in the article.

5.2.4 Set of relevant segmentation rules for the agenda

Table 4 presents the phases corresponding to a set of rules as defined as follows :

- segmentation when the beginning of a higher-level object, i.e., the start of a factor in the current-level segmented representation, is found.
- As soon as the ongoing concatenated object in the current level is an object already seen in the higher level, the character string is segmented.
- If the ongoing concatenated object at the current level is a prefix of a factor in the segmented representation at the current level, but the object acquired is also identical to the object following the prefix, then the string is not segmented.

The corresponding multi-scale formal diagram obtained is shown in Figure 12.

level	a	b	a	c	a	b	a	c	d	e	a	b	f	g	a	b	a	c	h	i	j	k	l	m	h	i	n	o	p	q	a	b	a	c	r	s	r	s	t	t	u
0	1	2	3	2	3	4	3	4	1	2	3	4	1	2	3	4	3	4	1	1	1	1	1	2	3	3	1	1	1	2	3	4	3	4	1	2	3	4	1	3	2
1		1		2		3		4		1		3		2		3		4						1						2		3		4		1		3			2
2				1				3						2				3												2				3							2
3														1																1											2

Table 4. Cognitive phases computed at each level of segmentation according to W.A. Mozart's *Rondo K545* multi-scale formal diagram obtained with MORFOS with a set of relevant segmentation rules described in the article.

This example can be compared to the one presented by J.M. Chouvel in [27], which is recalled in Figure 13.

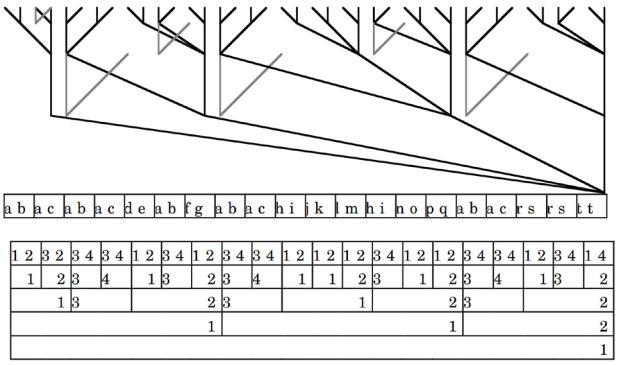


Figure 13. Cognitive phases of W.A. Mozart's *Rondo K545* according to Chouvel in [27].

This is an opportunity for us to introduce here another representation presented by Jean-Marc Chouvel of the work called the *structural representation*: here, the meaning of the branches, in addition to informing about the structure of the work, informs about the phenomenon of continuity of information (backward branching) or anticipation (forward branching). Forward branches that cross thus constitute disappointed anticipations.

We indeed obtain the same segmentations as those obtained by Chouvel, with a few exceptions. The first is the last character u, which is considered included in the last t of Chouvel's analysis: whereas for him it is a repeated and segmenting character (phase 4), we divide it into two characters t and u, the first being repeated and non-segmenting (phase 3) and the second being new and segmenting (phase 2).

The second difference concerns the unfolding of the two developments $(hijklm)$ and $(hinopq)$. Whereas Chouvel notes phase 2 at the acquisition of the first i, k, and o, we here consider phase 1 for new objects without segmentation, and phase 3 for the second i (object already seen without segmentation), where he considers phase 4 because there is segmentation. The objects segmented at level 0 are then propagated to the higher level 1 into three objects, which themselves have no impact on level 2.

6. DISCUSSION

A high-level notation, the formal diagram, is presented in this article. This notation is important for musicians as it might help them memorising the musical structure, and it has been used in teaching methods for memorise large forms. The relevance of the software has been more or less demonstrated using an educational example to facilitate understanding and validation of the concepts, although the aim of the software is to explore a larger repertoire.

We might wonder to what extent classification and segmentation criteria correspond to the associated cognitive process while listening to music. In this context, we could compare the Cognitive Algorithm with the cognitive model of music processing presented by S. Koelsch et W. A. Siebel [28]. Thus, the potentials that seem most relevant are those detecting attention and the end of phrases (MMN, ERAN, CPS). However, the musical context is of prime importance in pattern recognition, so in order to observe these potentials, it is necessary to carry out studies on specially composed music that allows all factors, such as transposition, to be controlled during pattern recognition, which makes such a study difficult to carry out.

To better understand the cognitive phases, we could also compare the computational costs associated with the acquisition of each musical object with the associated cognitive phases. We could compute the complexity of each object according to Kolmogorov's definition [29], meaning that we count how many operations are necessary to describe an object according to its context, i.e., the previous objects. Are the computational costs higher when a hypothesis is deceived after the cognitive phase 3? To what extent are the cognitive phases correlated with the compression of the information? These questions need to be more thoroughly studied.

7. CONCLUSION

In this article, the formalisation and implementation of classification and segmentation tests of the Cognitive Algorithm within the framework of a cognitive listening model with MORFOS software is presented. Based on this model, the cognitive processes associated with segmentation or not, and with the classification of a known or unknown element following the analysed musical object, have been implemented, giving four processes called cognitive phases. An intuition of the actual cognitive phenomena associated with cognitive phases can be given through the notion of integration, retention, realisation, acquisition/recursion, and expectation of the unknown. Then, four analyses of these cognitive phases are presented on four distinct segmentation examples of the *Rondo* from the *Sonata K.545* by W.A. Mozart.

In a second stage, an idea would be to calculate secondary cognitive phases, which can be defined as the *derivative* of cognitive phases. J.M. Chouvel [27] presents sixteen secondary cognitive phases, corresponding to the combinatorial succession of the four existing phases. By implement-

ing them in MORFOS, we might try to better understand how cognitive processes succeed each other and their impact on the cognitive load.

Acknowledgments

We would like to thank the Studio for Creation and Research in Computer Science and Experimental Music (SCRIME) for their contribution to the funding of this article.

8. REFERENCES

[1] W. Fitch and M. Martins, "Hierarchical processing in music, language, and action: Lashley revisited," pp. 2014 May;1316(1):87–104, 2014.

[2] F. Lerdhal and R. Jackendoff, *A generative theory of tonal music*. Cambridge, Mass. : MIT Press, 1983.

[3] S. Tojo, K. Hirata, and M. Hamanaka, "Computational reconstruction of cognitive music theory." *New Gener. Comput.*, vol. 31, p. 89–113, 2013.

[4] J.-M. Chouvel, "Musical form, from a model of hearing to an analytic procedure," *Interface*, vol. 22, pp. 99–117, 1993.

[5] J. Calandra, J.-M. Chouvel, and M. Desainte-Catherine, "Hierarchisation algorithm for morfos : a music analysis software," in *2025 International Computer Music Conference (ICMC 2025) Boston, MA, United States.*, Jun 2025.

[6] N. Meeùs, "Épistémologie d'une musicologie analytique," vol. volume XXII, pp. 97–114, 2015.

[7] E. Bigand and S. McAdams, *Contributions de la musique aux recherches sur la cognition auditive humaine*, 2005, ch. 8, pp. 261–262.

[8] X. Sun, Q. He, Y. Gao, and W. Li, "Musical tempo estimation using a multi-scale network," in *in Proc. of the 22nd Int. Society for Music Information Retrieval Conf., Online*, 2021.

[9] S. Böck, M. E. P. Davies, and P. Knees, "Multi-task learning of tempo and beat: Learning one to improve the other," in *Int. Society for Music Information Retrieval Conference*, 2019.

[10] S. Samson, *Perception des timbres musicaux*, de boeck supérieur ed. Bernard Lechevalier éd., 2010, pp. 123–146.

[11] G. Gabbolini and D. Bridge, "An interpretable music similarity measure based on path interestingness," in *in Proc. of the 22nd Int. Society for Music Information Retrieval Conf., Online*, 2021.

[12] D. Herremans, D. Martens, and K. Sörensen, *Composer Classification Models for Music-Theory Building*, 10 2015.

[13] M. Sarfati, A. Hu, and J. Donier, "Ensemble-based cover song detection," 05 2019.

[14] W. De Haas, M. Robine, P. Hanna, R. Veltkamp, and F. Wiering, "Comparing approaches to the similarity of musical chord sequences," vol. 6684, 06 2010, pp. 242–258.

[15] J. Foote, "Visualizing music and audio using self-similarity," in *Proc. Of ACM Multimedia*, 1999, pp. 77–80.

[16] E. Narmour, *The Analysis and Cognition of Basic Melodic Structures: The Implication-Realization Model*. University of Chicago Press, Chicago,, 1990.

[17] ——, *The Analysis and Cognition of Melodic Complexity: The Implication-Realization Model*. University of Chicago Press, Chicago,, 1992.

[18] F. Bimbot, E. Deruty, G. Sargent, and E. Vincent, "System & contrast : a polymorphous model of the inner organization of structural segments within music pieces." *Music Perception, University Of California Press*, vol. 33, pp. 631–661, 2016.

[19] O. Laske, *Music, Memory and Thought, explorations in Cognitive Musicology*. University Microfilms International, Ann Arbor (MI), 1977.

[20] J. Nattiez, *Musicologie générale et sémiologie*. Christian Bourgois, 1987.

[21] F. Delalande, *La musique au-delà des notes*. Presse Universitaires de Rennes, 2019.

[22] J. Calandra, J.-M. Chouvel, and M. Desainte-Catherine, "Multi-scale oracle and automated representation of formal diagrams based on the cognitive algorithm," in *Int. Conf. On Technologies For Music Notation And Representation - TENOR 2021*, 2021.

[23] C. Allauzen, M. Crochemore, and M. Raffinot, "Factor oracle: a new structure for pattern matching; ; theory and practice of informatics." in *Proceedings of SOFSEM'99*, Paris, France, 1999.

[24] G. Assayag and S. Dubnov, "Using factor oracles for machine improvisation," *Soft Computing*, vol. 8, pp. 604–610, 01 2004.

[25] C. Wang and S. Dubnov, "The variable markov oracle: Algorithms for human gesture applications," in *IEEE Multimedia, vol. 22, no. 4*, 2015, pp. 52–67.

[26] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of Molecular Biology*, vol. 48, no. 3, pp. 443–453, 1970.

[27] J.-M. Chouvel, *Esquisses pour une pensée musicale, Les métamorphoses d'Orphée*. L'Harmattan, Paris, 1998.

[28] S. W. Koelsch S, "Towards a neural basis of music perception," vol. Dec.9(12), pp. 97–114, 2005.

[29] A. N. Kolmogorov, "Three approaches to the quantitive definition of information," *In : Prob Info Trans*, vol. 1.1, p. 3–11, 1965.

INTERPRETING CHORD DISTANCES OF TONAL PITCH SPACE: A CORRELATION STUDY WITH CORPUS-BASED CHORD PROGRESSION FREQUENCIES

Naohiko Yamaguchi
International Professional University
of Technology in Tokyo
yamaguchi.naohiko@t.iput.ac.jp

Tetsuro Kitahara
Nihon University
kitahara.tetsuro@nihon-u.ac.jp

ABSTRACT

This paper investigates the musical meaning of chord distances computed by Tonal Pitch Space (TPS) , a theory quantifying numeric distance between chords. TPS is a useful theory, but the musical meaning of its proposed chord distance remains unclear. Prior research has not attempted to verify this meaning. We propose two interpretations for chord distance: a value related to the typicality of chord progressions (i.e., "transition" hypothesis) and a value related to the similarity of the chords (i.e., "substitutability" hypothesis). We explore if TPS distances represent chord "transition" or "substitutability" by relating them with chord progression trigram frequencies from the MidiCaps corpus. Two ratios are calculated: occurrence ratio (simple frequency) and relative occurrence ratio (relative to the most frequent chord). The results refuted both the "transition" hypothesis and the "substitutability" hypothesis. However, during the validation of the transition hypothesis, a bell-curve-shaped relationship was observed between chord distance and chord occurrence frequency. When considered in light of the Wundt curve in experimental psychology, this suggests that chord distance may be functioning as a model for inferring chord novelty. No significant differences were observed between popular music and jazz.

1. INTRODUCTION

In popular music and jazz, composition and analysis often revolve around chords, making it essential for computers to understand chord progression theory to achieve automatic composition and music recommendation. Realizing computational chord progression theories would enable, for example, the identification of characteristic chord progressions from user-liked songs to recommend similar music based on those progressions.

Tonal Pitch Space (TPS)[1] is a convincing theory to quantify numeric distance between two chords. TPS is a prominent computational music theory proposed by Fred Lerdahl in 2001, serving as a complementary theory to the Generative Theory of Tonal Music (GTTM). Among computational music theories, TPS is unique in its specific focus on chords (i.e., harmony analysis).

However, it remains unclear what the chord distance computed by TPS specifically signifies as a musical theoretical value, and whether this chord distance should be considered an interval scale or an ordinal scale. Even in Lerdahl's original work[1], there is no clear description regarding the meaning of chord distance or the type of scale it represents. Although several studies have attempted to improve or extend TPS[2, 3, 4], no prior research has thoroughly verified and examined the meaning indicated by chord distance.

Chord distance appears to have two possible interpretations. One perspective views chord distance as a measure of the typicality of a chord progression, where smaller distances indicate more typical or frequently occurring progressions. We call it "transition" hypothesis. The other perspective interprets chord distance as a measure of chord substitutability, suggesting that chord progressions with smaller distances share common harmonic functions, implying that chords can be interchanged while maintaining musical coherence. We call it "substitutability" hypothesis.

We aim to re-examine the meaning and evaluate the validity of chord distances calculated by TPS by investigating their relation with trigram frequencies of chord progressions obtained from a chord progression corpus. In this paper, a set of chord trigrams sharing the same Head and Tail chords is defined as a single context.

2. TONAL PITCH SPACE

TPS is the quantitative harmony analysis method. In TPS, the distance between two chords (x and y) is defined as the sum of distances in tonal space, harmonic function space, and basic space as

$$\delta(x, y) = \mathrm{region}(x, y) + \mathrm{chord}(x, y) + \mathrm{basic_space}(x, y).$$

The $\mathrm{region}(x, y)$ represents distance in tonal space. The basic method involves selecting the shorter distance on the circle of fifths (either clockwise or counter-clockwise) required to shift from the key of chord x to the key of chord y (Figure 1). For minor keys, transposition to their relative major is first considered.

The $\mathrm{chord}(x, y)$ represents distance in harmonic function space. Specifically, it involves converting chords x and y

Copyright: ©2025 Naohiko Yamaguchi et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

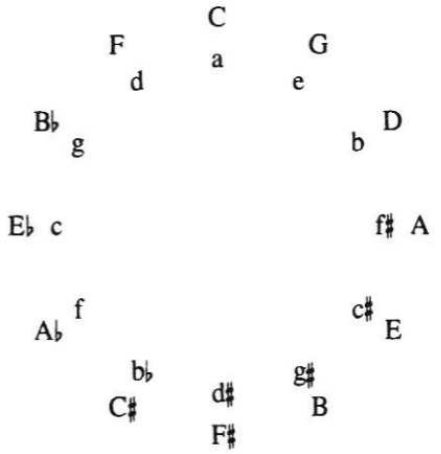


Figure 1: Circle of fifths (Kellner's regional circle)[1, p.47]

into their diatonic degree notation and then selecting the shorter distance on the common tone circle (either clockwise or counter-clockwise) to move between them (Figure 2).

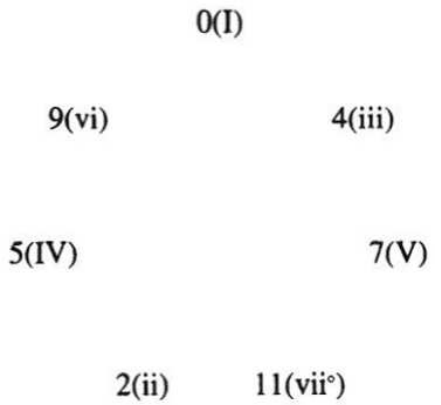


Figure 2: Common tone circle[1, p.56]

The $\mathrm{basic_space}(x, y)$ function represents the weighted change in chord tones, categorized by their harmonic function. The basic space is a particularly important concept in TPS. It is structured as a 12x5 matrix, where the horizontal axis represents the 12 pitch classes from C to B, and the vertical axis represents five levels denoting the function of each chord tone (Table 1). For minor keys, while there are three types of scale tones-—natural minor, harmonic minor, and melodic minor-—the natural minor scale is used for creating the basic space. Figure 3 shows an example of

Table 1: Five Levels in the Basic Space

Level a:	Tonic (1st) tone
Level b:	Dominant (5th) tone
Level c:	Sub dominant (4th) and other chord tone
Level d:	Diatonic tone (Major or Natural Minor)
Level e:	Chromatic tone

the basic space representation for Dm7 in C major key and Am in A minor key. In TPS, chord tones must always be diatonic tones, meaning that TPS can only represent diatonic chords. The basic space distance is defined as the count of components that have been moved, increased, or decreased when comparing the basic space of the two chords.

Figure 4 shows an example of calculating chord distance using TPS, specifically for chord x as Dm in C major key and chord y as G7 in C major key.

3. OUR APPROACH

3.1 Core question

Existing research (e.g. [2, 4]) interpreted TPS-computed chord distances as measures of distance of chord transition propensity; however, we questioned this interpretation. If chord distance represents the inverse concept of transition typicality (referred to as the "transition" hypothesis) (Figure 5a), then it would be natural for the distance of the chord progression $x \rightarrow y$ to differ from that of $y \rightarrow x$. However, the chord distance calculated by TPS is symmetrical (i.e., $\delta(x, y) = \delta(y, x)$). If chord distance represents the inverse concept of chord substitutability (referred to as the "substitutability" hypothesis), then the values of $\delta(x, y)$ and $\delta(y, x)$ should be identical.

The core question of this paper is to ascertain whether the "transition" hypothesis or the "substitutability" hypothesis correctly explains the chord distance calculated by TPS.

3.2 Hypothesis

We propose two hypotheses in response to the aforementioned core question.

If the "transition" hypothesis is correct, then more typical chord progressions should yield smaller chord distances, resulting in a monotonically decreasing relationship between a chord's contextual likelihood of appearance and its chord distance as shown in Figure 6a.

Alternatively, if the "substitutability" hypothesis is correct, then more compatible chords should yield smaller chord distances, resulting in a monotonically decreasing relationship between a chord's substitutability within a context and its chord distance as shown in Figure 6b.

3.3 Methodology for Validation

We validate the core question by investigating the relationship between the occurrence frequencies of chord progressions derived from a chord progression corpus and the chord distances calculated by TPS. For our experiments, the MidiCaps music corpus, developed by Jan Melechovsky et al.[5, 6], was utilized, primarily due to its extensive song collection and comprehensive inclusion of chord names, genre, and key information.

To facilitate the subsequent explanation, we define the set of diatonic triads and seventh chords as

$$\mathbb{D} = \{\text{'C'}, \text{'Dm'}, \text{'Em'}, \cdots, \text{'Am'}, \text{'Bm}^{-5}\text{'},$$
$$\text{'Cmaj7'}, \text{'Dm7'}, \text{'Em7'}, \cdots, \text{'Am7'}, \text{'Bm7}^{-5}\text{'}\}.$$

Level a			D									
Level b			D					A				
Level c	C		D		F			A				
Level d	C		D		E	F	G	A		B		
Level e	C	Db	D	Eb	E	F	Gb	G	Ab	A	Bb	B

(a) Dm7 in C major Key

Level a										A		
Level b					E					A		
Level c	C				E					A		
Level d	C		D		E	F		G		A		B
Level e	C	Db	D	Eb	E	F	Gb	G	Ab	A	Bb	B

(b) Am in A minor Key

Figure 3: Basic space

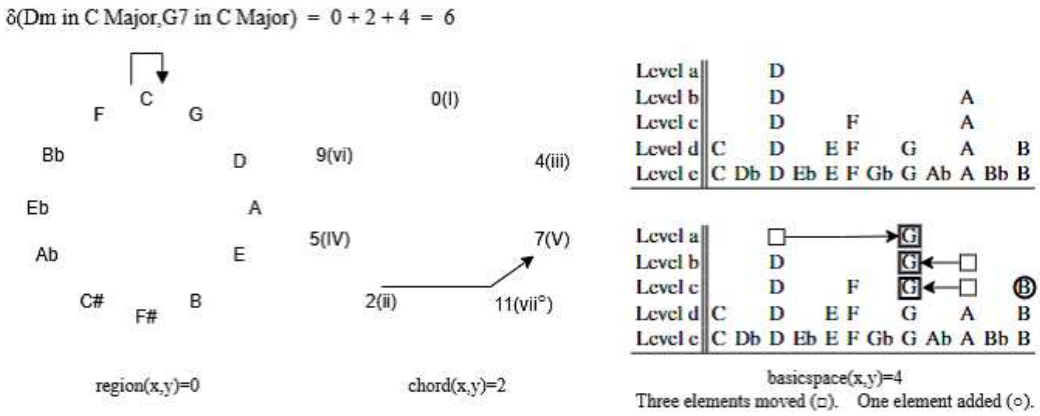


Figure 4: Calculation sample of chords distance by TPS

And we define a chord progression trigram, considering trigrams with identical C_{head} and C_{tail} chords to be a single context:

$$\text{trigram} \stackrel{\text{def}}{=} [C_{head} \to X \to C_{tail}] \quad (C_{head}, X, C_{tail} \in \mathbb{D}).$$

For each context, we aggregate the occurrence frequencies of the chord X within the corpus(Figure 7). When described as a conditional probability, the occurrence frequency is represented by

$$P(X = x_i \mid C_{head} = c_1, C_{tail} = c_2) \quad (1 \leq i \leq N),$$

where N denotes the number of distinct chord X types appearing in a context.

And we define x_{max} as the chord X with the maximum occurrence probability as

$$x_{max} \stackrel{\text{def}}{=} \underset{x_i}{\text{argmax}} P(X = x_i \mid C_{head} = c_1, C_{tail} = c_2) \quad (1 \leq i \leq N).$$

From the obtained occurrence frequencies, we calculate two metrics: occurrence ratio and relative occurrence ratio. These two types of ratios differ in their denominator, which forms the basis of their calculation.

The occurrence ratio (OR) is the simple appearance ratio of a chord X, calculated with conditional probability:

$$\text{OR}(c_1, x, c_2) = P(X = x \mid C_{head} = c_1, C_{tail} = c_2).$$

As an example represented in Table 2, with the occurrence count of context "C-X-G7" (457) as the denominator, the ratio where X is F is calculated as $251/457 = 54.92\%$, and the ratio where X is Dm7 is calculated as $90/457 = 16.69\%$.

The relative occurrence ratio (ROR) is the appearance ratio of other chords X, calculated using the occurrence count of x_{max} (the most frequent chord X) within the same context as the denominator:

$$\text{ROR}(c_1, x, c_2) = \frac{P(X = x \mid C_{head} = c_1, C_{tail} = c_2)}{P(X = x_{max} \mid C_{head} = c_1, C_{tail} = c_2)}.$$

As an example represented in Table 2, with the occurrence count of most frequent chord x_{max} F in context "C-X-G7" (251)as the denominator, the ratio where X is Dm7 is calculated as $90/251 = 35.86\%$, and the ratio where X is Dm is calculated as $52/251 = 20.72\%$.

3.4 Relation Between Two Occurrence Ratios and Chord Distance

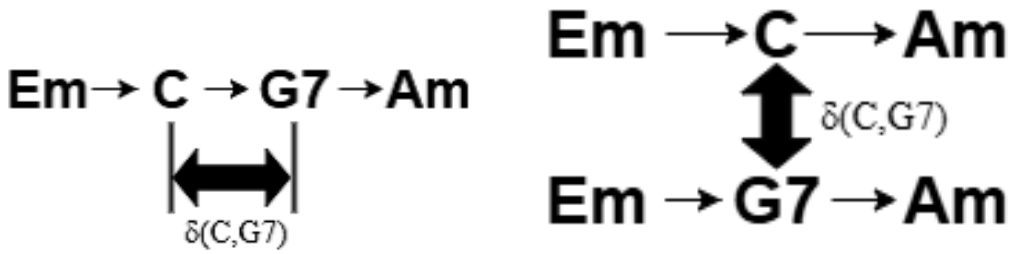
In the following equations, the symbol $\leftrightsquigarrow$ indicates a relationship (not limited to linear) between the left and right sides.

If the "transition" hypothesis holds true-—that is, if TPS's chord distance represents the chord transition propensity (CD_{tr})-—then some relation should be observable between CD_{tr} and the OR as

$$\text{OR}(c_1, x, c_2) \leftrightsquigarrow \text{CD}_{tr}.$$

CD_{tr} is defined as the sum of the chord distance from C_{head} to X and the chord distance from X to C_{tail} as

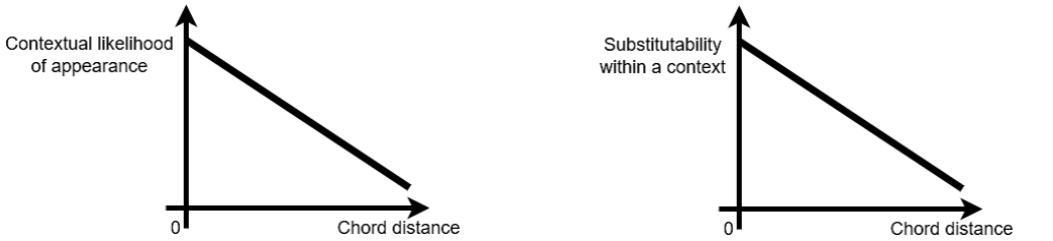
$$\text{CD}_{tr}(c_1, x, c_2) \stackrel{\text{def}}{=} \delta(c_1, x) + \delta(x, c_2).$$



(a) Pattern A: TPS-computed chord distances is measures of chord transition propensity.

(b) Pattern B: TPS-computed chord distances is measures of chord "substitutability" propensity.

Figure 5: Two Interpretations of Chord Distance



(a) Relationship between a chord's contextual likelihood of appearance and its chord distance

(b) Relationship between a chord's substitutability within a context and its chord distance

Figure 6: Hypothesis

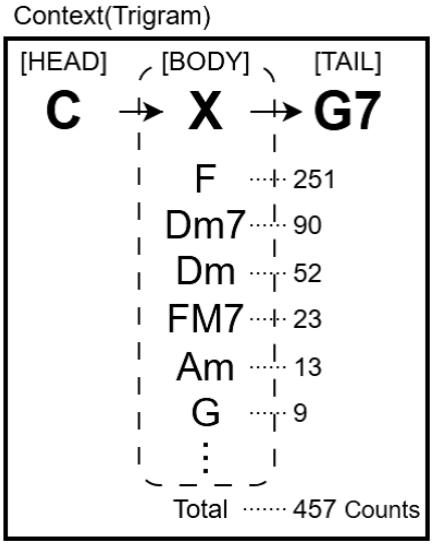


Figure 7: Sample of occurrence frequencies

If the "substitutability" hypothesis holds true-—that is, if TPS's chord distance represents the substitution propensity (CD_{sb})—then some relation should be observable between CD_{sb} and the ROR as

$$\mathrm{ROR}(c_1, x, c_2) \nleftrightarrow \mathrm{CD}_{sb}(c_1, x, c_2).$$

CD_{sb} is defined as the chord distance from X to x_{max} as

$$\mathrm{CD}_{sb}(c_1, x, c_2) \stackrel{\mathrm{def}}{=} \delta\left(x, x_{max}\right).$$

4. EXPERIMENTS

4.1 Procedure

4.1.1 Trigram Extraction

From the MidiCaps chord progression data, we selected only songs categorized as Jazz or Pop, then transposed major keys to C Major and minor keys to A minor before extracting the chord progression trigrams. We used the key data provided in the corpus for the original keys before transposition.

4.1.2 Trigram Frequency Aggregation

Trigrams are extracted and aggregated from the data prepared in §4.1.1. However, trigrams containing non-diatonic chords are excluded, as TPS can only process diatonic chords. Trigrams with repeated consecutive chords are also excluded. First, we fix the leading (C_{head}) and trailing (C_{tail}) chords of a trigram as the context, then aggregate the occurrence frequencies of the central (X) chord. We then calculate the OR and ROR.

4.1.3 Chord Distance Calculation and Relation Analysis

For the trigrams obtained in §4.1.2, the chord distances will be calculated using TPS according to the algorithm detailed in §2. We will calculate two types of chord distances: CD_{tr} & CD_{sb}

From the results obtained through the above procedures, a scatter plot will be generated to investigate the relation.

4.2 Results

To test the "transition" hypothesis, we generated a scatter plot showing the relation between CD_{tr} and OR; however,

Table 2: Occurrence ratio / Relative occurrence ratio

Context	Total		$x_1 = x_{max}$	x_2	x_3	x_4	x_5	x_6	$\cdots$
			F	Dm7	Dm	FM7	Am	G	$\cdots$
C-X-G7	457	Counts	251	90	52	23	13	9	$\cdots$
		occurrence ratio (OR)[%]	54.92	16.69	11.38	5.03	2.84	1.97	$\cdots$
		relative occurrence ratio (ROR)[%]	100	35.86	20.72	9.16	5.18	3.59	$\cdots$

contrary to the hypothesis, no monotonically decreasing relationship was observed, but rather a bell-shaped distribution with chord distances clustering around 13-14. Furthermore, a comparative analysis conducted by performing the same analysis on Jazz and Pop genres revealed no discernible differences between the two. Figure 8 presents the distribution charts created for five representative contexts.

Next, to test the "substitutability" hypothesis, we generated a scatter plot showing the relation between CD_{sb} and ROR; while no clear relation was found, chords with a chord distance of 1 (i.e., differing only by the presence or absence of a 7th) were distinctly separated from all others. No differences were observed between Jazz and Pop in this analysis either. Figure 9 presents the distribution charts created for five representative contexts.

Figures 10 and 11 present detailed analysis results for two representative contexts.

Figure 10 illustrates the context where the chord progression moves from Am (C_{head}) to another chord (X) and then returns to Am (C_{tail}). The graph of CD_{tr} – RO exhibits a broad bell shape, indicating that a diverse range of chords can serve as the X. Similarly, the graph of CD_{sb} - ROR also shows a wide distribution.

Conversely, Figure 11 presents the context where the chord progression moves from Dm7 (C_{head}) to another chord (X) and then proceeds to C (C_{tail}). In this specific context, the chord X is almost exclusively G7 or G, representing a typical ii-V-I progression. The graph for CD_{tr} - OR displays an extremely narrow bell shape, indicating that the variety of chords that can function as the X is severely limited. Similarly, in the chord distance between CD_{sb} - ROR graph, only chords with a chord distance of 1 (i.e., those differing merely by the presence or absence of a 7th) are prominently used, again suggesting a very restricted range of chord choices.

4.3 Consideration

4.3.1 About Hypothesis Verification

Initially, the "substitutability" hypothesis is refuted due to the lack of a discernible relation between CD_{sb} and ROR.

A bell-shaped relationship was observed between CD_{tr} and OR, which, not being the monotonically decreasing relationship assumed by the "transition" hypothesis, suggests that the "transition" hypothesis is not correct.

Thus, it's evident that chord distance is not a simple metric that directly correlates with occurrence frequency derived from the chord corpus.

4.3.2 Interpretation of Chord Distance

While both the "substitutability" hypothesis and "transition" hypothesis were refuted, the observed bell-shaped relationship between CD_{tr} and OR is noteworthy, as theories in experimental psychology propose a similar bell-shaped (inverted U-shaped) relationship between stimulus complexity (or novelty) and pleasantness. This theory originates from the Wundt curve (Figure 12), proposed by psychologist Wilhelm Wundt, and has been subsequently supported by numerous researchers.

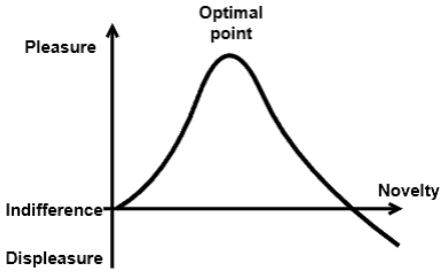


Figure 12: Wundt Curve

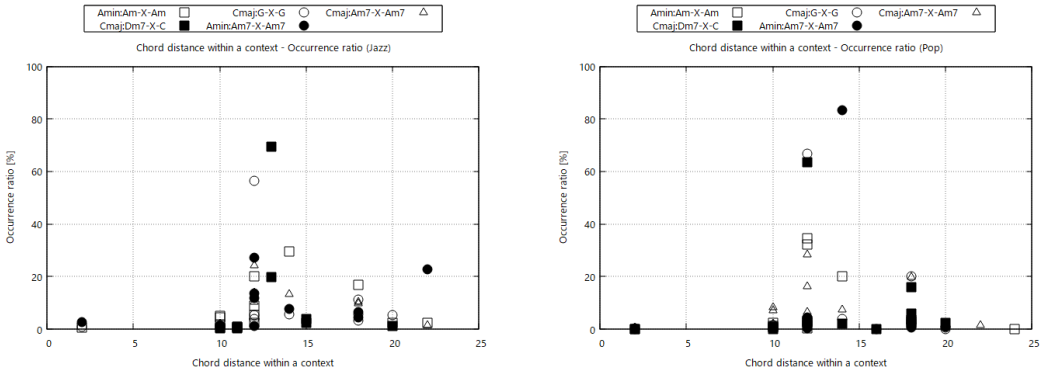
If considered in light of the Wundt curve, chord distance may serve as a model for inferring chord novelty; however, precisely defining chord novelty is challenging, making it difficult to immediately ascertain whether chord distance accurately infers it. Further investigations are necessary, incorporating insights from music psychology and cognitive theories.

5. CONCLUSION

In this paper, to verify whether the chord distance calculated by TPS indicates the chord transition propensity (CD_{tr}),that is, "transition" hypothesis, or the chord substitutability (CD_{sb}), that is, "substitutability" hypothesis, we calculated the relation between TPS-derived chord distances and occurrence frequencies obtained from a chord progression corpus using two distinct methods.

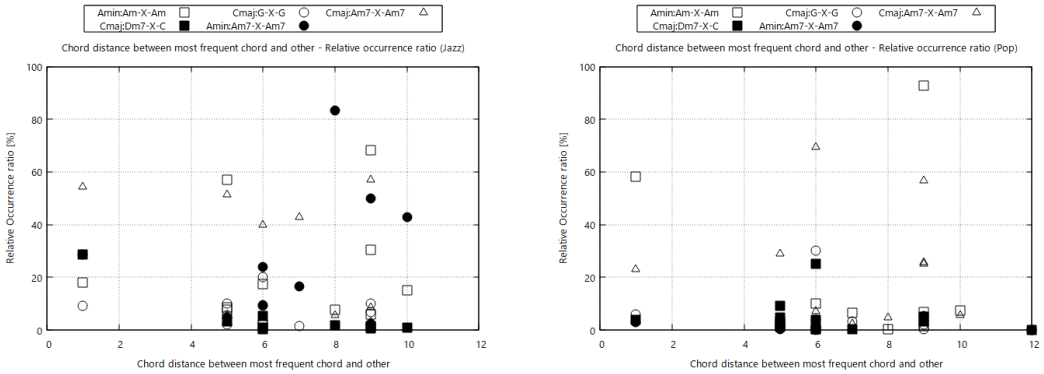
If the "transition" hypothesis holds true, then a scatter plot of CD_{tr} versus Occurrence ratio (OR) should reveal some relation; indeed, the results showed a tendency for chord distance to exhibit a bell-shaped distribution, primarily centered around 13-14. This result is analogous to the Wundt curve and suggests the possibility that chord distance functions as a model for inferring chord novelty.

If the "substitutability" hypothesis holds true, then a scatter plot of CD_{tr} versus Relative Occurrence ratio (ROR) should reveal some relation; although no clear relation was



(a) Jazz (b) Pop

Figure 8: A scatter plot showing the relation between CD_{tr} and OR



(a) Jazz (b) Pop

Figure 9: A scatter plot showing the relation between CD_{sb} and ROR

found in this scatter plot, chords with a chord distance of 1 (i.e., those differing only by the presence or absence of a 7th) were distinctly separated from all others.

In both results, no significant differences were observed between the Jazz and Pop genres.

Our future objectives are to perform verification using statistical models rather than just scatter plots, and to expand the analysis to non-diatonic chords utilizing the authors' proposed TPS-ExJ model.

Acknowledgments

Our special thanks go to Professor Satoshi Tojo for his valuable advice and supervision.

This research was supported by research grant 2025 from Kawai Foundation for Sound Technology & Music.

6. REFERENCES

[1] F. Lerdahl, *Tonal pitch space*. Oxford University Press, 2001.

[2] Y. Yamamoto and T. Mizutani, "Computational analysis of jazz music: Estimating tonality through chord progression distances," in *Proceedings of the 7th International Conference on Computer Science and Application Engineering*, 2023, pp. 1–6.

[3] H. Yamamoto and S. Tojo, "Beyond the basic-space of tonal pitch space: Distance in chords and their interpretation," pp. 83 – 87, May 2023. [Online]. Available: https://doi.org/10.5281/zenodo.8079302

[4] W. B. De Haas, R. C. Veltkamp, and F. Wiering, "Tonal pitch step distance: a similarity measure for chord progressions." in *ISMIR*, 2008, pp. 51–56.

[5] J. Melechovsky, A. Roy, and D. Herremans, "Midicaps: A large-scale midi dataset with text captions," 2024. [Online]. Available: https://arxiv.org/abs/2406.02255

[6] AMAAI-Lab, "Midicaps: A large-scale dataset of caption-annotated midi files," 2024, retrieved 2025/06/27. [Online]. Available: https://github.com/AMAAI-Lab/MidiCaps

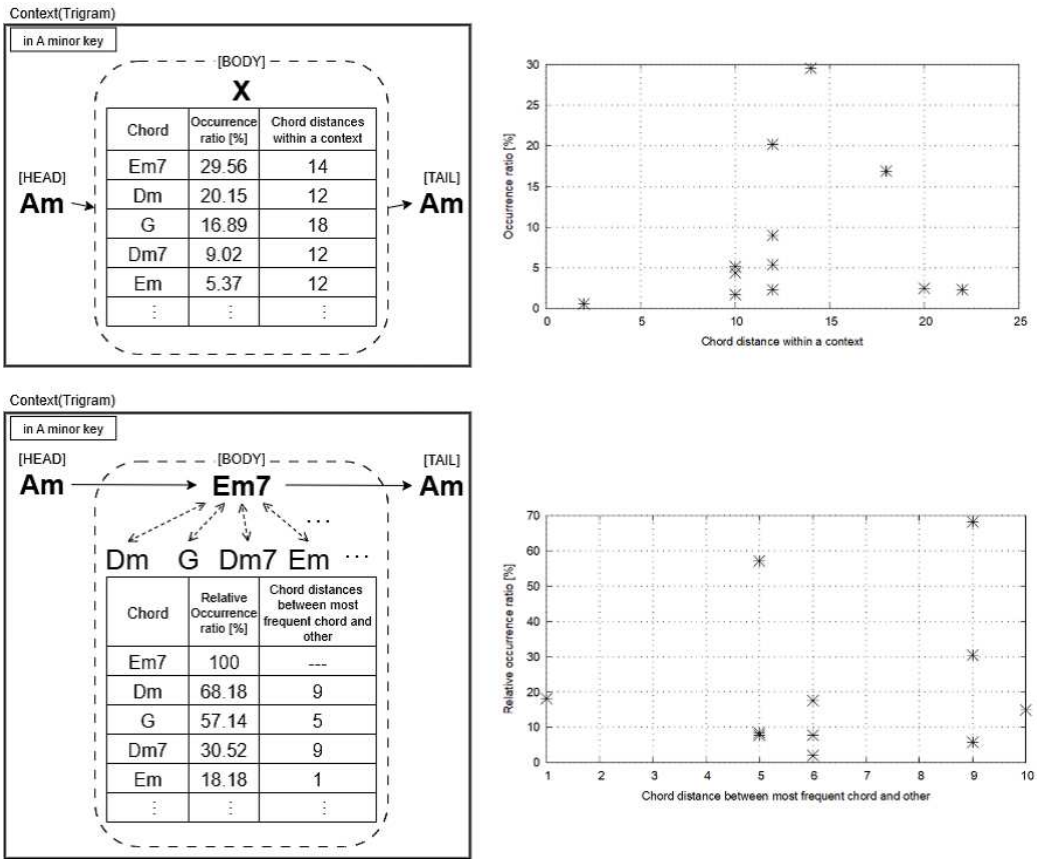


Figure 10: Result viualisation (Am-X-Am in A minor key (from genre 'Jazz'))

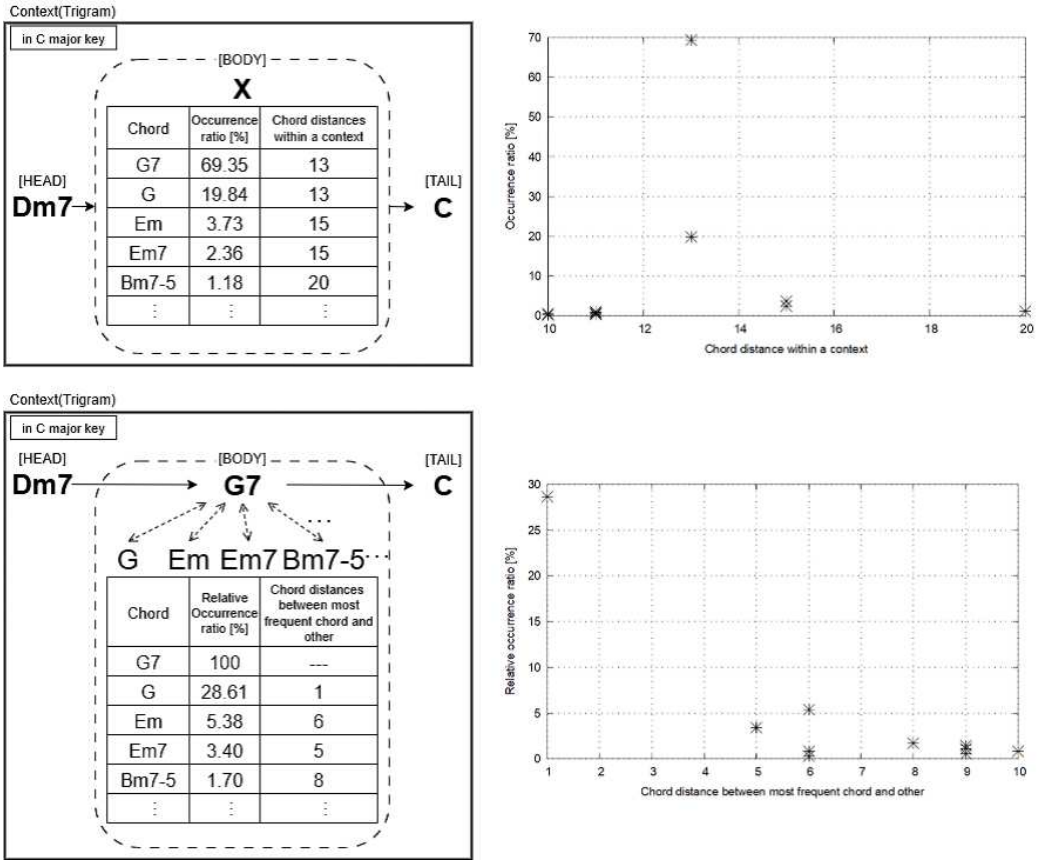


Figure 11: Result viualisation (Dm7-X-C in C major key (from genre 'Jazz')

MODELING RAGTIME STRUCTURE AND STYLE FROM RHYTHMIC NOTATION ALONE

Phillip B. Kirlin
Department of Computer Science, Rhodes College
kirlinp@rhodes.edu

ABSTRACT

We explore the extent to which rhythmic features alone, specifically binary note-onset patterns and the Longuet-Higgins and Lee (LHL) syncopation metric, can predict both musical phrase boundaries and composer identity in ragtime music. Using a novel dataset of 71 hand-labeled symbolic scores by Scott Joplin, James Scott, and Joseph Lamb, we present statistical analyses revealing that rhythmic and syncopation patterns differ significantly across composers and metrical positions within a 16-measure strain. We evaluate how effectively these rhythmic patterns predict ragtime phrase boundaries by training and comparing multiple machine learning models, including logistic regression, support vector machines, and neural network architectures such as convolutional neural networks, bidirectional long short-term memory networks, and transformers. An ablation study confirms that the LHL syncopation measure significantly improves predictive performance. We further demonstrate that these same rhythmic features can be used to classify composer identity either from an entire piece or from a 16-measure strain, and discuss the musicological interpretations of these findings. These results provide clear evidence that the rhythmic aspect of musical notation alone can model structural and stylistic differences in compositions.

1. INTRODUCTION

Understanding musical structure is a central challenge in computational musicology. While prior work has approached this problem from melodic, harmonic, and rhythmic perspectives, most systems aim to leverage whichever features are most predictive, regardless of their musical origin. Yet in certain genres, such as ragtime, rhythmic style is not merely a contributing factor, but a defining element of the genre.

Ragtime is characterized by its distinctive rhythmic features, particularly its syncopated or "ragged" rhythms. During ragtime's heyday in the late 1890s and early 1900s, the stylistic convention emerged of structuring such compositions into 16-measure phrases or *strains*; composers such as Scott Joplin, James Scott, and Joseph Lamb best exemplified this convention [1, 2]. This standardized phrase structure, combined with the characteristic syncopated rhythmic complexity, provides an ideal context for investigating how rhythmic features might encode both structural and composer-specific stylistic information.

In particular, we ask to what extent the structure of ragtime music can be understood and predicted using the rhythmic aspect of notation alone. While a number of recent studies have attempted to quantify the way syncopation — a purely rhythmic phenomenon — is used in ragtime compositions, previous research has not systematically investigated syncopation's potential to predict structural elements such as phrase boundaries or composer identity. Understanding how rhythmic and syncopation patterns encode such information provides potential insights into cognitive processing of rhythm and can inform other areas of computational musicology.

In this paper, we directly investigate the relationship between syncopation and the development of musical phrases and strains in a ragtime composition. We then extend the investigation to study if different ragtime composers utilized syncopation differently within musical phrases. We show that we can use syncopation along with note onset patterns — no other melodic or harmonic information — to reliably predict both phrase boundaries and composer identity in symbolic ragtime scores. This is made possible by a newly-released symbolic dataset of 71 ragtime piano compositions, annotated with composer labels and hand-labeled 16-measure segment boundaries.

Our contributions are as follows. We publicly release a new dataset of ragtime music annotated with 16-measure phrase boundaries and composer labels. We present a detailed analysis of how syncopation, measured using the Longuet-Higgins and Lee (LHL) metric [3], varies within and across pieces, composers, and phrase locations. Our statistical results confirm that composers use syncopation in measurably distinct ways within phrases. We use machine learning models to predict the location of segment boundaries using rhythmic onset patterns and syncopation values alone. We show that these same rhythmic features can reliably predict composer identity, both at the segment level and the full-piece level, and we provide a musicological interpretation of these findings. Lastly, we confirm the impact of the syncopation factor with an ablation study. The dataset and code for all of the experiments described here is publicly available. [1]

Copyright: ©2025 Phillip B. Kirlin. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

[1] https://github.com/pkirlin/ragtime-tenor-2025

2. RAGTIME, SYNCOPATION, AND MUSICAL PHRASES

Syncopation in music occurs when a listener, expecting to hear the musical onset of a note on a strong beat, instead finds that onset shifted to a weak beat, subverting their expectations [4, 5]. While syncopation is found in many musical genres, it is particularly identified with ragtime [2]. Because of this close association, it is natural to study how the use of syncopated patterns varies in the genre. Music historians have noted that the particular varieties of syncopated patterns have varied in usage over time [1, 6, 7], and computational studies have confirmed this [8, 9, 10, 11]. Syncopation can be measured through various metrics, including the Longuet-Higgins and Lee (LHL) metric [3], various schemes utilizing inner metric analysis [12, 13] or the Sioros-Guedes metric [14]. We rely on the LHL metric here because it closely aligns with human perception [15] and has been used in other corpus studies of ragtime [10, 9]. We describe the calculation of this metric in the following section.

A prototypical ragtime composition has a set musical structure consisting of three or four sections of music known as *strains*, each of which is sixteen measures long. Each strain is typically denoted by a letter of the alphabet, "A," "B," etc. Strains are often repeated and reprised. Each strain is usually divided into four phrases of four measures each, occasionally with the third phrase a repetition of the first. While the majority of a composition consists of these strains, frequently there will be other musical phrases included that are not a part of any strain, such as an introduction, short interludes, or a coda.

3. CREATION AND ANALYSIS OF THE DATASET

There are many datasets of ragtime music available [8], but no symbolic ones with strain boundaries clearly identified. Therefore, we set out to create a dataset of "classic rags," a term which refers to a ragtime piano composition written in the style of Scott Joplin (1867 or 1868–1917) that usually follows a prescribed musical form [16]. These compositions are written in duple meter (usually 2/4 time, though some in 4/4 time) and with well-defined 16-measure strains. "Classic rags" specifically exclude ragtime dances, waltzes, or marches. The largest set of classic rags was written by the "big three" ragtime composers of Joplin, James Scott (1885–1938) and Joseph Lamb (1887–1960), who stand out as best exemplifying the ragtime genre [2].

Creating the Dataset. We created this dataset by examining the works of Joplin, Scott, and Lamb, and selecting all the compositions that fit the classic rag format. We then located symbolic scores in either Humdrum or MusicXML format, and manually located each 16-measure musical strain of each composition, assigning them the letters "A," "B," etc., along with also notating variations of each strain. We use the term "variation" here to mean any repetition of the strain that is not identical to the original occurrence. For every musical phrase *not* part of a 16-measure strain, we assigned a label such as "introduction," "interlude," or "coda."

In the end, we collected 71 compositions by the big three composers (31 by Joplin, 28 by Scott, and 12 by Lamb). All were in 2/4 time except for two in 4/4 time (one each by Joplin and Lamb). These compositions consisted of 10,414 total measures of music. There were a total of 604 16-measure strains in the corpus, including repeated strains, totaling 9,664 measures of music. The phrase annotations were stored in a text file recording the name of the composition, the composer, the name of the phrase or strain, the variation number of the phrase or strain, and its length in number of measures. Every measure of every composition therefore falls into exactly one phrase or strain, which we will collectively call *segments*.

Preprocessing. Classic piano rags feature a syncopated melody in the right hand paired with a largely non-syncopated left-hand accompaniment [1]. Because our analysis focuses exclusively on the rhythmic aspect of music notation, we began by converting each measure of the right-hand melody parts of the compositions into a *binary onset pattern*, a standard representation in studies of ragtime syncopation [8, 10, 9]. A binary onset pattern is a pattern of ones and zeros where a one represents a note onset and a zero represents a lack of a note onset at a particular subdivision of the beat. For pieces in 2/4 time, we computed these patterns at the 16th-note level of granularity, meaning a measure of four eighth notes would be represented as "`10101010`." For pieces in 4/4 time, we computed these patterns at the 8th-note level, as it was clear from the notation in these compositions that the underlying quarter note pulse in the 4/4 pieces was equivalent to the eighth note pulse in the 2/4 pieces. This step ensured that every measure in the entire dataset was represented by a pattern of eight ones and zeros.

Using these patterns, we computed the LHL syncopation metric for each measure. This metric is zero for measures with no syncopation, and increases for each occurrence of a note onset on a weak beat followed immediately by the lack of an onset on the following strong beat. Syncopations crossing longer divisions of the measure increase the metric more. For instance, the pattern `01010101` contains three instances of a weak beat "`1`" followed by a strong beat "`0`." The first and last have LHL values of 1, while the middle instance has a value of 2 because it crosses the midpoint of the measure. Therefore, this measure has an overall LHL value of 4. If this measure were followed by a measure with no note onset on the downbeat, the LHL value would increase by 3 (to a total of 7) for the additional syncopation crossing the barline.

As the final step of preprocessing, we calculated the LHL value from each previously-computed binary onset pattern. Our final dataset consisted of every measure of music from the melodies in the corpus, aligned with the corresponding ragtime segment annotations, binary onset pattern, and LHL value.

Analysis. Previous work has shown that the big three ragtime composers used more syncopation than their contemporaries [9] and that their use of syncopation varied depending on interactions with the metrical hierarchy [11].

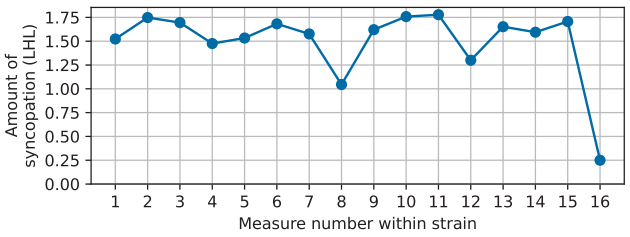


Figure 1. Average syncopation values during each measure of a 16-measure strain.

Here, we examine syncopation specifically within 16-measure strains.

We first analyzed the change in LHL syncopation values during a 16-measure strain. We isolated only these strains from our dataset and averaged the LHL values for each measure across all of the strains. Figure 1 illustrates that these values are not constant. We can see how syncopation tends to fall at natural points of rest within the strain, namely at the end of the strain (measure 16), halfway (measure 8), and one-quarter and three-quarters of the way through (measures 4 and 12). This should not be surprising — these are the concluding points of the natural four-measure phrases within the strain, where one would expect a release of tension. The average LHL values at measures 3–4, 7–8, 11–12, and 15–16 clearly show this release.

We ran a one-sided permutation test to examine if the average LHL value during a strain rises and falls due to deliberate variations in the use of syncopation or due to chance. We measured the total variability in the average LHL values during a strain by calculating the sum of the squared differences between adjacent measures. To generate a null distribution, we randomly permuted the measures within each strain, thereby removing any consistent temporal structure, recomputed the average LHL values, and calculated the same variability statistic. Repeating this procedure 10,000 times yielded a distribution against which we compared the observed test statistic. The resulting p-value was less than 0.0001, indicating that the observed LHL contour is highly unlikely to have occurred by chance and reflects a statistically significant pattern in the use of syncopation within a strain.

We then analyzed LHL contour over strains grouped by composer; results are shown in Figure 2. This graph suggests that the big three ragtime composers employed syncopation in distinct ways. To test this hypothesis, we conducted three pairwise one-way multivariate analyses of variance (MANOVA), comparing the mean LHL contours — treated as 16-dimensional vectors — between each pair of composers. In each test, the null hypothesis stated that the two composers had equivalent average syncopation contours. All three comparisons were statistically significant ($p < 0.0001$) after applying the Šidák correction, indicating that each composer employed a distinct pattern of syncopation within strains. These results support the hypothesis that the big three ragtime composers not only used syncopation with differing intensities, but also with structurally distinct temporal profiles.

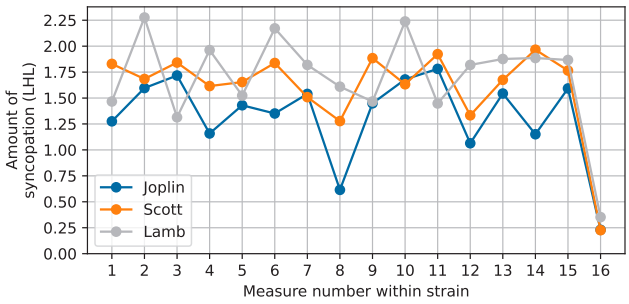


Figure 2. Average syncopation values during each measure of a 16-measure strain, grouped by composer.

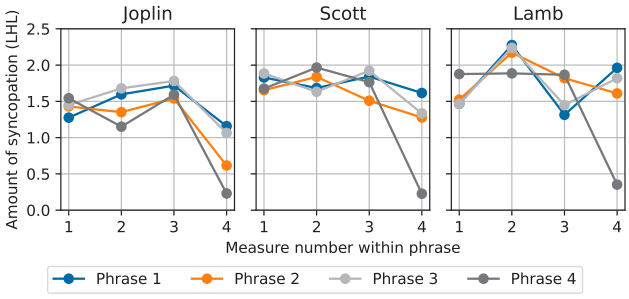


Figure 3. Average syncopation values during each measure of a 4-measure phrase within a strain.

It is also informative to examine syncopation use within the four-measure phrases of each strain. Figure 3 illustrates this by averaging LHL syncopation values across each four-measure phrase (i.e., measures 1–4, 5–8, 9–12, and 13–16) allowing for clearer identification of within-phrase trends. A consistent pattern emerges for Joplin and Scott: syncopation levels remain relatively stable during the first three measures of each phrase and then decrease noticeably in the fourth measure. Lamb, however, departs from this pattern. His phrases exhibit greater internal variability in LHL values, and the expected drop in syncopation at the end of the phrase occurs reliably only in phrases 2 (only slightly) and 4.

Our final analysis of syncopation patterns examines how syncopation changes across different strain categories. Most of the 71 compositions in the dataset contain four distinct strains, though a handful only have three, and a few have five or six. We examined average LHL value across the 16 measures of A, B, C, and D strains, using only data from the first variation of each strain. Figure 4 illustrates how A, B, and C strains tend to have relatively similar contours (the thinner lines) but D strains have a unique contour: these strains tend to alternate between more extreme high and low values of syncopation than the other strains. We ran another MANOVA test to examine if the contour of the D strains differed in a statistically significant way from the average contours of the A, B, and C strains *combined*. To get enough statistical power for the test, we used data from the first and second variations of each strain, rather than just the first. This test revealed a statistically significant difference in contour profiles between D strains and

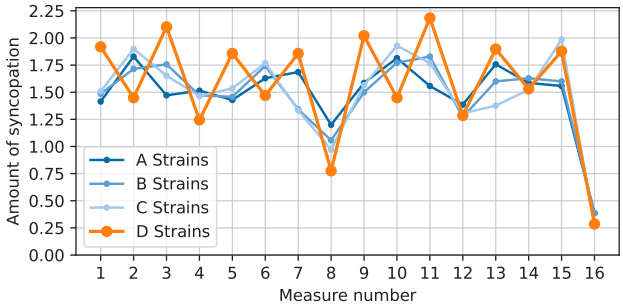


Figure 4. Average syncopation values during each measure of a 16-measure strain, grouped by strain identifier.

the other strains ($p < 0.01$).

4. PREDICTING PHRASE BOUNDARIES WITH MACHINE LEARNING MODELS

Having established that the three major ragtime composers — Joplin, Scott, and Lamb — use syncopation in measurably distinct ways within ragtime strains, we investigate whether these rhythmic differences can be leveraged to identify the boundaries of those same individual strains. This task falls under the broader umbrella of *phrase boundary detection*, a long-standing problem in computational musicology that has been studied extensively using both symbolic scores and audio recordings as input. Automated boundary detection has practical implications for music informatics, including interactive notation systems, structural visualization, and digital score analysis.

Previous Work. Approaches in the symbolic music domain typically fall into one of two categories: prescriptive systems usually based on rules derived from music theory, human perception, and cognition; and systems where rules or models are directly computed from data. Prescriptive systems include those found in Lerdahl and Jackendoff's *A Generative Theory of Tonal Music* [17], Narmour's Implication-Realization Model [18, 19], Cambouropoulos' Local Boundary Detection Model [20], and Temperley's Grouper model [21]. These systems attempt to encode human intuitions about melodic expectation and rhythmic grouping. While conceptually rich, many of these frameworks were not originally designed for computational implementation, though subsequent work has formalized parts of them algorithmically.

On the data-driven side, most techniques for phrase boundary prediction in symbolic data are based on statistical or machine learning approaches, including data-oriented parsing [22], information-theory-driven learning [23], restricted Boltzmann machines [24], rule mining approaches [25], and mathematical optimization [26]. In particular, we build on the investigations into neural networks for boundary prediction [27, 28] that examined convolutional neural networks (CNNs) and bidirectional long short-term memory networks (BiLSTMs); we extend this to include transformers and the attention mechanism [29].

Our Approach. We follow the symbolic data-driven paradigm, but this work differs in a few key areas. First, most prior work in this area, while acknowledging that much of phrase-boundary finding is rhythmically driven [25], relies on auxiliary melodic or harmonic information for an extra boost. In contrast, our goal was to examine what was possible with the rhythmic aspect of music notation alone, in particular, the binary onset pattern of each measure and its LHL syncopation value. Second, we use a novel dataset, and focus on predicting the start and/or end of phrase boundaries at the measure level, rather than the note level. We did this as to better align with perceptions of ragtime melodies which are tightly coupled to the 16-measure musical strain. Third, we build on prior machine learning models but focus on those designed for long-term dependency modeling, namely bidirectional long short-term memory networks (BiLSTMs) and transformers. Fourth, we expressly desired to test if it is easier to predict phrase boundaries for certain composers than others. To our knowledge, this is the first study to model phrase boundaries in symbolic ragtime scores using machine learning models with purely rhythmic features.

Task Formulation. We frame phrase boundary detection as a sequence labeling task. Each input is a ragtime composition, divided at the measure level. Each melody measure is represented by nine features: eight binary for the eight ones and zeros in the binary onset pattern, and one integer for the LHL value of the measure. All segment boundaries — both 16-measure ragtime strains and separate phrases not part of any strain — are encoded at the measure level using a 1 for the starting measure, 2 for the ending measure, and 0 for all other measures. Our goal was to develop a machine learning architecture that could predict the segment boundaries of an entire composition at once by labeling each measure of a previously-unseen composition with a 0, 1, or 2. We studied three different variations of labeling and predicting: using all three labels (0, 1, and 2), using only starting and continuing labels, and using only continuing and ending labels.

We evaluated eight machine learning models: two non-neural models to obtain a baseline level of accuracy, and six neural network architectures to evaluate the possible improvements that could be leveraged by models designed for capturing long-term dependencies in sequences. Our models are:

- **Logistic Regression**: The first baseline model.
- **SVM**: The second baseline model: a support vector machine with a radial basis function kernel to help capture non-linear decision boundaries.
- **CNN-CRF**: A convolutional neural network (CNN) architecture followed by a conditional random field (CRF) layer for sequence labeling. The CNN consists of a single 1D convolutional layer with 32 filters, a kernel size of 3, and ReLU activation, followed by a dropout layer with a rate of 0.3. The output of the convolution is passed through a fully connected layer that maps to the label space, producing per-measure class scores. Finally, a CRF layer, inspired by [28], is used to model transitions between measure labels and enforce structural consistency in the predicted sequence. The total number of parameters is 1930.

- **CNN-3-CRF**: An extended version of the previous model with a deeper architecture consisting of three stacked convolutional layers, each similar to the single layer in the previous model: 32 filters, a kernel size of 3, and ReLU activation. The final output passes through a dropout layer with a rate of 0.3, followed by a fully connected layer identical to that in the previous model. This deeper CNN captures increasingly abstract rhythmic features over a larger temporal context, with the goal of identifying more complex segment boundary cues. The total number of parameters is 7178.
- **BiLSTM**: A bidirectional LSTM (BiLSTM) model with two stacked layers, each with 32 hidden units per direction (yielding 64-dimensional outputs per time step after concatenation). A dropout layer with a rate of 0.3 is applied between layers. The LSTM outputs are passed through two fully connected layers: a linear projection from 64 to 32 dimensions, followed by a ReLU activation and dropout, and a final linear layer mapping from 32 to the number of label classes. The BiLSTM architecture is well-suited to this task because it captures both past and future context for each time step, making it effective at detecting temporal patterns such as rhythmic transitions and segment boundaries that depend on information across the sequence. The total number of parameters is 38,242.
- **BiLSTM-CRF**: A BiLSTM identical to the previous one, but with a CRF layer applied on top. The total number of parameters is 38,250.
- **Transformer**: A transformer model that first projects each measure of music into a 128-dimensional embedding space using a linear projection layer. A positional encoding is then added to preserve the order of measures. The embedded sequence is processed by a stack of two Transformer encoder layers, each with four attention heads and an embedding size of 128. The output is passed through two fully connected layers: first from 128 to 128 hidden units with ReLU activation and dropout with a rate of 0.3, and then from 128 to the number of output label classes. Unlike the BiLSTM models, which process sequences sequentially and capture temporal dependencies in order, the transformer attends to all positions simultaneously, allowing it to model long-range rhythmic dependencies more efficiently and capture global patterns in the music. The total number of parameters is 36,834.
- **Transformer-CRF**: A transformer identical to the previous one, but with a CRF layer applied on top. The total number of parameters is 36,842.

Each model was coded in PyTorch [30] or scikit-learn [31] and trained using the Adam optimizer with a weight decay of 0.00005 for L2 regularization, and early stopping. Hyperparameters (number of CNN filters, BiLSTM and transformer dimensions) were tuned separately on a fixed 20% subset of the data and held constant during leave-one-out cross validation. We used a loss function adjusted for class imbalance as appropriate (cross-entropy or negative log-likelihood), depending on model. We evaluated each model under three phrase boundary prediction setups (predicting start-continue-end, start-continue, and continue-end), two dataset variations (whole pieces labeled versus only 16-measure strains being labeled), and four composer training/testing groupings (all composers, only Joplin, only Scott, and only Lamb). For each of these setups, we ran leave-one-out cross validation at the composition level, predicted phrase boundaries for each measure of the tested piece, and calculated precision, accuracy, and F1 score. For every configuration, we also predicted segment boundaries with the true LHL input ablated (set to zero in the input) to simulate training each model with only note onsets and no LHL values.

Results. On average, across all task variations, the models trained to locate the end of segments (trained on segments labeled with 0 and 2) had higher average F1 scores than the models trained to locate segment beginnings (labels 0 and 1) or both (all three labels). This was true for 7 out of 8 models, both dataset variations, and all composer groups. Therefore, to save space, we present results here only for the end-of-segment labeling task; full results for the other labeling tasks are available in the online code repository. Table 1 shows precision, recall, and F1 scores for all task combinations predicting segment endings; because this task is trained to label the measures of a composition with 0 for a segment continuation and a 2 for an ending, showing just the results for the ending predictions is sufficient and makes it easier to compare our results with other studies; recent studies cite F1 scores ranging from 0.26 to 0.89 [24, 25, 28].

Including the LHL syncopation feature in the inputs improved performance across all model types, training subsets, and composer groups, with an average F1 score increase of 0.02. While modest, this gain is statistically significant ($p < 0.0001$) according to a Wilcoxon signed-rank test. One likely explanation for this increase is that the LHL metric is derived directly from the same binary onset patterns already available to the models, dampening its effect. In future work, we plan to explore whether LHL values, especially when calculated over longer contexts than one measure, interact with specific metrical subsets of binary onset pattern values in ways that support more accurate boundary detection.

On average, models performed slightly better when trained on all segment types (i.e., both 16-measure strains and miscellaneous phrases such as introductions and codas), likely because strain and non-strain boundaries share rhythmic characteristics, providing a broader distribution of boundary patterns. We isolated results from this setting in Table 1 and visualized them in Figure 5 to highlight several key findings.

First, the neural network models consistently outperformed the logistic regression and SVM baseline models. While this may not be surprising given their greater number of trainable parameters, it is notable because the statistical patterns described in Section 3 might suggest that this task is relatively straightforward. In practice, however, identifying phrase boundaries requires more nuanced modeling than simple classification can provide. The CNN-based models showed modest gains over the baselines, but the

Model	Training	All composers				Only Joplin				Only Scott				Only Lamb			
		Pre	Rec	F1	Δ	Pre	Rec	F1	Δ	Pre	Rec	F1	Δ	Pre	Rec	F1	Δ
Logistic Reg	All	0.23	0.87	0.36	+0.01	0.19	0.87	0.31	+0.02	0.29	0.89	0.44	-0.01	0.22	0.83	0.35	+0.01
SVM	All	0.27	0.91	0.41	+0.00	0.25	0.88	0.38	+0.00	0.37	0.91	0.52	+0.00	0.21	0.80	0.34	+0.00
CNN-CRF	All	0.56	0.43	0.49	+0.04	0.50	0.32	0.39	+0.03	0.55	0.56	0.55	+0.11	0.54	0.43	0.48	+0.06
CNN-3-CRF	All	0.48	0.55	0.52	+0.02	0.44	0.43	0.43	+0.04	0.53	0.61	0.57	+0.07	0.57	0.53	0.55	+0.00
BiLSTM	All	0.56	0.80	0.66	+0.03	0.45	0.57	0.50	+0.01	0.61	0.81	0.70	+0.01	0.51	0.73	0.60	-0.02
BiLSTM-CRF	All	0.76	0.70	**0.73**	+0.05	0.54	0.46	0.50	+0.02	0.81	0.72	**0.76**	+0.03	0.65	0.58	0.62	+0.04
Transformer	All	0.51	0.88	0.65	+0.00	0.54	0.79	0.64	-0.01	0.56	0.86	0.68	+0.00	0.54	0.82	**0.65**	+0.01
Transf-CRF	All	0.72	0.72	0.72	+0.03	0.71	0.69	**0.70**	+0.02	0.74	0.75	0.75	+0.05	0.62	0.67	0.64	+0.01
Logistic Reg	Only 16	0.19	0.85	0.31	+0.01	0.15	0.84	0.26	+0.01	0.24	0.88	0.37	+0.00	0.20	0.84	0.32	+0.01
SVM	Only 16	0.22	0.91	0.36	+0.00	0.20	0.87	0.32	+0.00	0.31	0.89	0.46	+0.01	0.20	0.76	0.32	-0.01
CNN-CRF	Only 16	0.55	0.36	0.43	+0.05	0.50	0.31	0.39	+0.08	0.50	0.47	0.48	+0.10	0.49	0.35	0.41	+0.03
CNN-3-CRF	Only 16	0.44	0.51	0.47	+0.05	0.43	0.43	0.43	+0.05	0.50	0.56	0.53	+0.15	0.47	0.33	0.39	-0.04
BiLSTM	Only 16	0.54	0.78	0.64	+0.02	0.45	0.65	0.53	+0.01	0.58	0.77	0.66	+0.03	0.47	0.68	0.55	+0.01
BiLSTM-CRF	Only 16	0.75	0.67	0.70	+0.03	0.60	0.52	0.56	+0.09	0.73	0.63	0.68	+0.03	0.67	0.55	0.61	+0.03
Transformer	Only 16	0.45	0.88	0.59	-0.01	0.44	0.78	0.56	-0.02	0.48	0.89	0.62	+0.00	0.46	0.75	0.57	+0.01
Transf-CRF	Only 16	0.75	0.70	**0.73**	+0.06	0.69	0.67	**0.68**	+0.03	0.68	0.71	**0.69**	+0.05	0.61	0.66	**0.63**	+0.01

Table 1. Performance on predicting segment boundaries by model, subset of data used (all segments or only 16-measure strains), and composer. Δ indicates improvement in F1 score when the LHL syncopation value is included in training.

BiLSTM and transformer models achieved around double the F1 scores of the simpler models across most test conditions. The CNN models likely lag behind because while they can capture local rhythmic patterns in the measures well, they lack a mechanism for modeling long-range dependencies, an essential feature for identifying phrase structure. These results indicate that the deeper architecture in the 3-layer CNN as compared to the single-layer CNN helped performance only slightly, even with an almost four-fold parameter increase. In contrast, the BiLSTM and transformer models can more easily model temporal dependencies across larger time spans.

Second, we note that in comparing performance in all composers grouped together, the CRF and non-CRF versions of the BiLSTM and transformer models performed similarly, with the CRF giving a slight boost to the F1 scores, most likely due to the CRF's capacity to enforce structured label transitions. This is noteworthy as these models have similar numbers of trainable parameters.

Third, model performance varies notably by composer. Even though Joplin and Scott each had roughly the same numbers of pieces in the corpus (31 versus 28), finding phrase boundaries in Joplin's compositions appeared to be a harder task for six of the eight models; only the transformer-based classifiers seemed to cope roughly equally well with the two composers' phrases. Lamb, with only 12 pieces, exhibited the greatest relative variability in F1 scores, sometimes being the most difficult of the three composers and sometimes being almost even with Joplin and Scott. These differences likely stem from distinct rhythmic styles: as shown in Figure 3, Joplin's strains exhibit frequent reductions in syncopation at the end of each four-measure phrase, which may lead the model to misinterpret internal boundaries as final ones. By contrast, Scott and Lamb tend to reserve syncopation drop-offs for the end of full 16-measure strains, thereby providing clearer predictions for segment boundaries. The sparsity of Lamb's data likely further contributes to inconsistency in model performance.

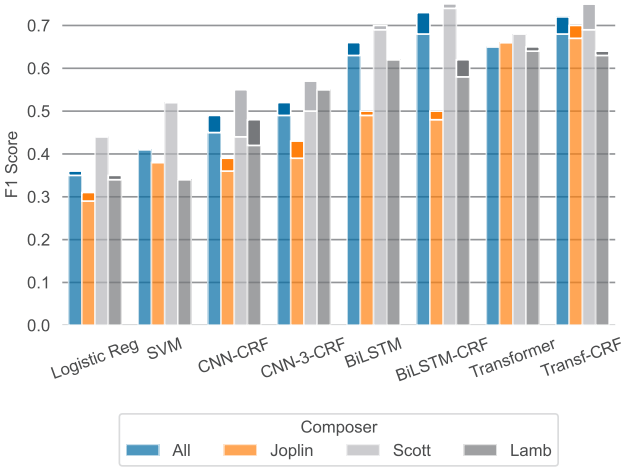


Figure 5. Performance of each model and composer group in predicting strain endings. Improvement due to including LHL is shown by the shaded region at the top of applicable bars.

These findings demonstrate that rhythmic notation alone encodes sufficient information to identify phrase boundaries, even in the absence of pitch or harmonic content. Additionally, the differences in syncopation and phrase shaping among the composers in the corpus are reflected in the models' performance. Further musicological investigations could explore whether this rhythm-based approach generalizes to other metrically strong musical genres, potentially suggesting insights for both symbolic music representation and the analysis of historical compositional style.

5. PREDICTING COMPOSERS

While the previous section focused on how rhythmic features such as syncopation align with phrase structure in ragtime, we now turn to the question of whether these features can also be used to distinguish between individual composers. In other words, we ask if Joplin, Scott, and Lamb employ rhythm and syncopation in ways that are compositionally distinct enough to allow machine learning models to identify authorship based on notated rhythms alone.

Previous Work and Task Formulation. Composer attribution has been a longstanding task in music informatics. Traditional approaches often relied on hand-crafted melodic, harmonic, or rhythmic features, but more recent deep learning methods have enabled direct classification from symbolic scores [32, 33, 34]. Our approach straddles the line: as in the last section, our goal is to see what is possible using solely a rhythmic view of a score. Therefore, we use similar machine learning models as in the previous section, that are provided only binary onset patterns and syncopation values. We evaluate two task formulations: (1) strain-level classification, where each 16-measure segment is independently labeled and predicted, and (2) piece-level classification, where the entire composition is input and a single composer label is predicted.

We adapted five of the eight models from the previous section to work in this new formulation. Specifically, we used the logistic regression, SVM, 3-layer CNN, BiLSTM, and transformer models. We removed the CRF layer from the CNN (and did not use it in the other neural network models) because it was unsuitable for this task: a conditional random field is specifically used in situations with multiple labels being predicted to model dependencies between them; here, there is only one label per prediction (a single 16-measure strain or composition). For this same reason, each neural network model also applies pooling after processing its input. The CNN models use global average pooling to summarize convolutional activations over time, the BiLSTM model uses mean pooling over the sequence dimension, and the transformer uses mean pooling across the time dimension. These operations are all mathematically identical, though the terminology used by convention is slightly different. As in the previous section, we incorporated class weights into the cross-entropy loss function.

For predicting composer from a strain, we used 10-fold cross validation, and for predicting composer from an entire piece, we used leave-one-out cross-validation. Results are presented in Table 2, with confusion matrices in Figure 6. Direct comparison to prior work is difficult due to differences in dataset composition and input modality; most notably, our models rely exclusively on rhythmic features. However, for reference, a similar three-composer prediction task in [34] cited per-class accuracies rates between 0.864 and 0.998 (compare with our recall rates).

Results. Several insights emerge from these experiments. First, strain-level models generally outperform whole-piece models across the board. This is likely due to the greater number of training examples available at the segment level,

		Strain-level			Piece-level		
Model	Comp	Pre	Rec	F1	Pre	Rec	F1
Logistic Reg	Joplin	0.78	0.69	0.73	0.73	0.77	0.75
Logistic Reg	Scott	0.71	0.77	0.74	0.63	0.61	0.62
Logistic Reg	Lamb	0.51	0.56	0.54	0.36	0.33	0.35
SVM	Joplin	0.94	0.88	0.90	0.67	0.90	0.77
SVM	Scott	0.86	0.94	0.90	0.62	0.64	0.63
SVM	Lamb	0.92	0.88	0.90	0.00	0.00	0.00
CNN-3	Joplin	0.80	0.78	0.79	0.67	0.71	0.69
CNN-3	Scott	0.73	0.74	0.73	0.64	0.57	0.60
CNN-3	Lamb	0.50	0.50	0.50	0.23	0.25	0.24
BiLSTM	Joplin	0.75	0.80	0.77	0.67	0.65	0.66
BiLSTM	Scott	0.72	0.73	0.72	0.63	0.68	0.66
BiLSTM	Lamb	0.50	0.39	0.44	0.27	0.25	0.26
Transformer	Joplin	0.81	0.83	0.82	0.76	0.84	0.80
Transformer	Scott	0.75	0.74	0.74	0.60	0.54	0.57
Transformer	Lamb	0.52	0.52	0.52	0.25	0.25	0.25

Table 2. Per-class performance on predicting composers from individual strains or complete pieces.

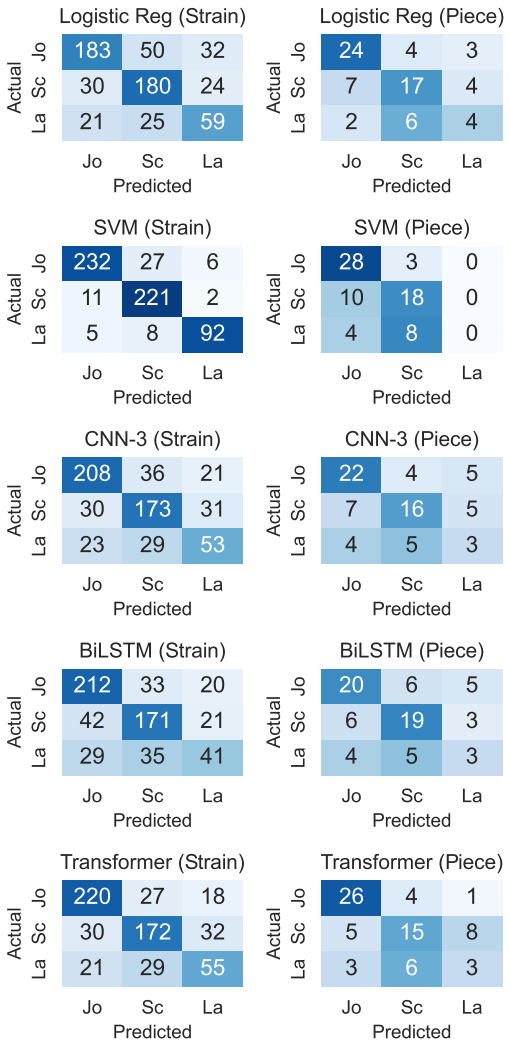


Figure 6. Confusion matrices for strain-level and piece-level classification tasks.

as well as the models' effectiveness at detecting short-term rhythmic patterns. Second, the SVM achieves the highest strain-level F1 scores across all composers, suggesting that simpler models can excel when the prediction context is constrained. In contrast, the SVM failed to classify any Lamb compositions correctly in the piece-level task, and continued to misclassify such compositions even when we oversampled Lamb scores in the training data. Lamb was consistently the most difficult composer to classify for each model, which may reflect both the smaller number of available pieces (12) and higher within-class variability. At the piece level, Lamb was frequently misclassified as Scott, which may stem from the fact that both composers exhibit higher average syncopation than Joplin.

Third, these two tasks see less improvement by using neural network models over the baseline models (logistic regression and SVMs) than in the segment boundary detection task from the previous section. Here, in the the strain-level classification task, the three neural network models show only a slight improvement over logistic regression and none at all over the dominantly-performing SVM. In the piece-level classification task, the neural models actually perform slightly worse, on average, than logistic regression and the SVM. This is somewhat surprising, as one would expect that these architectures, especially the BiLSTM and transformer, would do better at capturing higher-level rhythmic and temporal information in the music; however, this suggests that in ragtime, composer identity is primarily expressed through local rhythmic motives rather than extended structural dependencies.

Several other factors may explain these results. First, though we saw in Section 3 (Figures 2 and 3) that there are statistically-significant differences in the way that the three composers use syncopation, these differences are relatively subtle. Second, the dataset is small, which limits the potential of higher-capacity models and increases the risk of overfitting. With these two factors, we might have reached the limit of how well the discussed machine learning models can do.

Musicological Analysis. To further interpret these results through a musicological lens, we examined the weights that logistic regression achieved on the strain-level composer classification task; these weights are illustrated in Figure 7 on a per-composer basis. Because the features are normalized, we can interpret the weights as representing the contribution of each feature toward predicting a given composer, relative to the others. Each of the first seven features corresponds to a binary onset at a specific metric position in the measure ("Pos0" through "Pos7"); these are followed by the LHL syncopation value feature and a bias term. Because each feature corresponds directly to a position in the notated measure, we obtain a clear mapping from the computational output back to the original symbolic rhythmic notation.

The weight profiles suggest clear stylistic differences in rhythm and syncopation usage across composers. We can make a few observations. As expected from our earlier statistical analyses (Figures 2 and 3), Scott and Lamb receive higher weights on the LHL syncopation feature than

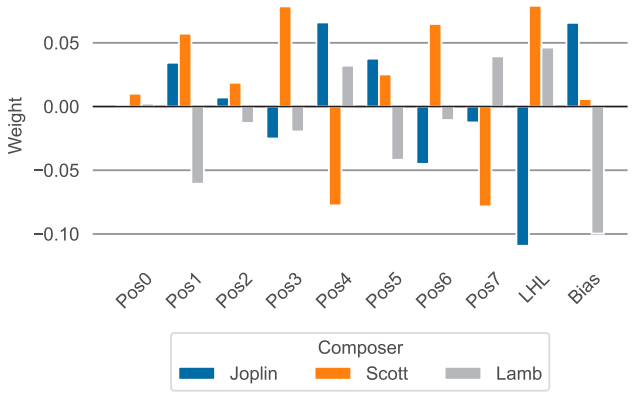


Figure 7. Logistic regression weights for each feature for strain-level composer classification. Positive weights indicate that higher feature values increase the likelihood of classifying a segment as that composer relative to the other two composers, while negative weights reduce it.

Joplin, reflecting their greater use of syncopation. More granular insights can be derived from the individual positional weights. For instance, while the downbeat of the measure, position 0, does not show much difference between the composers, the halfway point, position 4, shows large differences between Joplin and Scott: Joplin appears to favor having an onset here much more than Scott does, suggesting that Scott included more syncopation crossing the midpoint of the measure (e.g., from position 3 to position 4), than Joplin did. The opposite situation occurs at position 6, where an onset here favors Scott but not Joplin, suggesting Joplin preferred syncopation crossing from position 5 to 6 more than Scott did. The large negative bias weight for Lamb reflects his underrepresentation in the corpus.

These weight differences reinforce a musicological interpretation of stylistic rhythm in ragtime: while all three composers make use of syncopation, the specific rhythmic patterns and degrees of metric disruption vary in ways that are detectable and classifiable. Though the precision and recall scores that logistic regression obtained in this experiment limit the overall power of these musicological interpretations, it is still a useful tool.

These results and their musicological interpretation suggest several avenues for future work. From a modeling perspective, hybrid architectures that combine convolutional layers for local rhythmic motif detection with recurrent or attention-based layers for long-range structure may better capture both local and global rhythmic features. Expanding the dataset, particularly with additional Lamb compositions, could reduce class imbalance and when combined with an interpretable model, would provide more opportunities for useful musicological findings. Finally, it remains an open question whether the rhythmic dimension of music notation alone is sufficient for composer attribution in this repertoire, or whether additional melodic or harmonic information is needed to more fully distinguish compositional style.

6. CONCLUSION

This paper demonstrates that the rhythmic aspect of music notation alone, represented by binary onset patterns and Longuet-Higgins and Lee (LHL) syncopation values, provides substantial information about both musical structure and composer identity in ragtime. Using only these symbolic rhythmic representations, we showed that machine learning models, particularly those like BiLSTMs or transformers that are designed to detect long-term structural dependencies, can reliably identify the boundaries of ragtime strains. On the composer-classification task, non-neural models approached or exceeded the performance of neural networks. In particular, support vector machines performed best in the strain-level task, but were inconsistent in the piece-level task. Still, the overall performance statistics indicate that rhythmic notation provides a good amount of guidance for distinguishing composers, especially those of Joplin and Scott.

From a musicological perspective, these findings reinforce that rhythmic patterns encode musical style and structural information in a way that is both detectable and classifiable. The interpretable logistic regression weights and positional feature analysis highlight how differences in phrasing and syncopation among Joplin, Scott, and Lamb are evident directly in their notated rhythmic patterns. This underscores the value of score-based rhythmic representations for both computational musicology and the study of historical musical style.

Future work will explore several directions. Modeling segment-level syncopation trajectories, rather than per-measure values, may improve boundary detection and composer attribution. Hybrid neural network architectures that combine convolutional layers for local rhythmic motifs with attention-based or recurrent layers for global context could further improve performance. Expanding the dataset could enable more robust stylistic modeling and support deeper musicological insight. Finally, extending this work to evaluate the contributions of melodic and harmonic features alongside rhythmic notation will clarify how different aspects of score representation interact in composer attribution and segment boundary detection.

7. REFERENCES

[1] E. A. Berlin, "Ragtime," in *Grove Music Online*. Oxford University Press, 2013. [Online]. Available: https://doi.org/10.1093/gmo/9781561592630.article.A2252241

[2] R. Blesh and H. Janis, *They All Played Ragtime*, 4th ed. New York: Oak Publications, 1971.

[3] H. C. Longuet-Higgins and C. S. Lee, "The rhythmic interpretation of monophonic music," *Music Perception*, vol. 1, no. 4, pp. 424–441, 1984.

[4] D. Huron, *Sweet Anticipation: Music and the Psychology of Expectation*. Cambridge, MA: MIT Press, 2006.

[5] G. Sioros, G. Madison, D. Cocharro, A. Danielsen, and F. Gouyon, "Syncopation and groove in polyphonic music: Patterns matter," *Music Perception*, vol. 39, no. 5, pp. 503–531, 2022.

[6] J. E. Hasse, "Ragtime: From the top," in *Ragtime: Its History, Composers, and Music*, J. E. Hasse, Ed. New York: Shirmer Books, 1985, pp. 1–39.

[7] I. Harer, "Defining ragtime music: Historical and typological research," *Studia Musicologica Academiae Scientiarum Hungaricae*, vol. 38, no. 3–4, pp. 409–415, 1997.

[8] A. Volk and W. B. de Haas, "A corpus-based study on ragtime syncopation," in *Proc. of the 14th International Society for Music Information Retrieval Conference*, 2013, pp. 163–168.

[9] P. B. Kirlin, "A corpus-based analysis of syncopated patterns in ragtime," in *Proc. of the 21st International Society for Music Information Retrieval Conference*, 2020, pp. 647–653.

[10] H. V. Koops, A. Volk, and W. B. de Haas, "Corpus-based rhythmic pattern analysis of ragtime syncopation," in *Proc. of the 16th International Society for Music Information Retrieval Conference*, 2015, pp. 483–489.

[11] J. Yust and P. B. Kirlin, "The multileveled rhythmic structure of ragtime," in *Proc. of the 9th International Conference on Culture and Computing*, 2021, pp. 337–354.

[12] D. Bemman and J. Christensen, "Inner metric analysis as a measure of rhythmic syncopation," in *Proc. of the 25th International Society for Music Information Retrieval Conference*, 2024, pp. 389–396.

[13] A. Volk, "The study of syncopation using inner metric analysis: Linking theoretical and experimental analysis of metre in music," vol. 37, no. 4, pp. 259–273, 2008.

[14] G. Sioros and C. Guedes, "Complexity driven recombination of MIDI loops," in *Proc. of the 12th International Society for Music Information Retrieval Conference*, 2011, pp. 381–386.

[15] G. Sioros, A. Holzapfel, and C. Guedes, "On measuring syncopation to drive an interactive music system," in *Proc. of the 13th International Society for Music Information Retrieval Conference*, 2012, pp. 283–288.

[16] D. A. Jasen and T. J. Tichenor, *Rags and Ragtime: A Musical History*. New York: Seabury Press, 1978.

[17] F. Lerdahl and R. Jackendoff, *A generative theory of tonal music*. Cambridge, MA: MIT Press, 1983.

[18] E. Narmour, *The Analysis and Cognition of Basic Melodic Structures: The Implication-Realization Model*. Chicago: University of Chicago Press, 1990.

[19] ——, *The Analysis and Cognition of Melodic Complexity: The Implication-Realization Model*. Chicago: University of Chicago Press, 1992.

[20] E. Cambouropoulos, "Musical rhythm: A formal model for determining local boundaries, accents and metre in a melodic surface," in *Music, Gestalt, and Computing: Studies in Cognitive and Systematic Musicology*, M. Leman, Ed. Springer-Verlag, 1997, pp. 388–417.

[21] D. Temperley, *The Cognition of Basic Musical Structures*. Cambridge, MA: MIT Press, 2004.

[22] R. Bod, "Memory-based models of melodic analysis: Challenging the gestalt principles," *Journal of New Music Research*, vol. 31, no. 1, pp. 27–36, 2002.

[23] M. T. Pearce, D. Müllensiefen, and G. A. Wiggins, *Melodic Grouping in Music Information Retrieval: New Methods and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 364–388.

[24] S. Lattner, C. E. C. Chacón, and M. Grachten, "Pseudo-supervised training improves unsupervised melody segmentation," in *Proc. of the 24th International Joint Conference on Artificial Intelligence*, 2015, pp. 2459–2465.

[25] P. van Kranenburg, "Rule mining for local boundary detection in melodies," in *In Proc. of the 21st International Society for Music Information Retrieval Conference*, 2020, pp. 271–278.

[26] M. E. R. López, "Automatic melody segmentation," Ph.D. dissertation, Utrecht University, Utrecht, 2016.

[27] K. Ullrich, J. Schlüter, and T. Grill, "Boundary detection in music structure analysis using convolutional neural networks," in *Proc. of the 15th International Society for Music Information Retrieval Conference*, 2014.

[28] Y. Zhang and G. Xia, "Symbolic melody phrase segmentation using neural network with conditional random field," in *Proc. of the 8th Conference on Sound and Music Technology*, 2020, pp. 55–65.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. of the 31st International Conference on Neural Information Processing Systems*, 2017, p. 6000–6010.

[30] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 8024–8035.

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[32] G. Buzzanca, "A supervised learning approach to musical style recognition," in *Proc. of the International Conference on Music and Artificial Intelligence*, 2002.

[33] G. Velarde, T. Weyde, C. C. Chacón, D. Meredith, and M. Grachten, "Composer recognition based on 2D-filtered piano-rolls," in *Proc. of the International Society for Music Information Retrieval Conference*, 2016, pp. 115–121.

[34] H. Verma and J. Thickstun, "Convolutional composer classification," in *Proc. of the 20th International Society for Music Information Retrieval Conference*, 2019, pp. 549–556.

COGNITIVELY-AWARE SYMBOLIC NOTATION VIA SIMULATION: A SOFTWARE-ONLY FRAMEWORK FOR ADAPTIVE MUSIC REPRESENTATION

Sanjay Majumder
Style tree
sanjay@styletree.me

Neal Anderson
Style tree
neal@styletree.me

ABSTRACT

This paper presents a software-only, simulation-based framework for cognitively-aware music notation. The system provides a modular proof-of-concept for adaptive music representation that responds to modeled performer cognition without requiring hardware-based brain–computer interfaces. Instead of using physiological signals, the framework employs behavioral proxies, including performance timing, expressive deviation, and simulated eye-gaze, to infer cognitive load and attentional focus. These simulated cues drive real-time adjustments to the notation, such as adaptive visual spacing, emphasis modulation, and rhythmic simplification, aligning score presentation with predicted attentional bandwidth. The framework integrates symbolic music processing and machine-learning heuristics within a unified architecture composed of a Symbolic Music Engine, Cognitive-State Simulator, and Adaptive Notation Engine. Evaluation was conducted entirely in a closed-loop simulation environment using professional score datasets and virtual performer profiles of varying expertise. The results demonstrate improved simulated sight-reading fluency, reduced modeled cognitive effort, and improved readability. In general, this study establishes the feasibility of purely software-driven adaptive notation, offering a foundation for future empirical validation and interdisciplinary research connecting computational notation design, cognitive modeling, and AI-assisted music performance.

1. INTRODUCTION

Music notation has historically served as the primary interface between composer intent and performer interpretation. From neumes in medieval chant to modern staff notation and experimental graphic scores, the evolution of notational systems has mirrored broader technological and cultural shifts. In recent decades, the integration of computational methods has significantly expanded the representational capacity of music notation, enabling real-time rendering, interactive performance systems, and algorithmic composition. However, traditional notation systems often remain static and do not account for the dynamic cognitive states of performers, particularly under real-time constraints such as sight-reading, improvisation, or multitasking.

This research addresses the growing need for adaptive and cognition-aware music notation systems developed purely through software. Unlike brain-computer interfaces (BCIs) that rely on external neural sensors, our approach leverages simulated cognitive signals derived from software-based performance analysis, such as expressive timing, cursor tracking, and modeled attentional dynamics. These behavioral signals act as proxies for the underlying mental states and enable the adaptive rendering of symbolic music in real-time.

Drawing upon advances in artificial intelligence, symbolic music modeling, and cognitive science, this work introduces a modular software framework that simulates neural-informed adaptation of music scores. The system is designed to enhance the engagement of the performer, reduce the cognitive load, and promote fluid interpretation by continuously reshaping the notation based on the inferred mental workload. This approach aligns with the central theme of Notational Intelligence by treating notation not merely as a representational artifact but as a responsive medium grounded in real-time human-computer interaction.

This work presents a computational framework rather than a performer-in-the-loop system. We outline the design, implementation, and evaluation of a software-only architecture that augments notational intelligence without requiring hardware-based neural interfaces. Through simulation, we examine the effects of adaptive notation on fluency, error rates, and subjective experience, offering new insights at the intersection of music representation, cognitive modeling, and computational design. The current implementation is limited to simulation-based evaluation, serving as a proof of concept and establishing a theoretical and technical foundation for future empirical validation with live musicians.

2. BACKGROUND AND RELATED WORK

Music notation serves as a primary interface between a composer's intent and a performer's interpretation, translating abstract musical ideas into structured symbolic representations. Traditional Western notation, while deeply entrenched and highly expressive, is inherently

Copyright: ©2025 Sanjay Majumder and Neal Anderson. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

static—providing the same representation regardless of the performer's skill level, cognitive state, or performance context. Although this has allowed the standardization and preservation of musical works, it can create barriers for novice and intermediate performers who may struggle with real-time interpretation.

Research has shown that comprehension of notation systems is shaped by cognitive factors such as working memory, visual attention, and perceptual grouping [1, 2]. For example, Cheng et al. [1] systematically evaluated ten different notation systems and demonstrated that design variations significantly influence readability and cognitive load. Neurocognitive studies further reveal that music reading engages intertwined visual–motor and auditory–cognitive networks that overlap with language and spatial processing systems [3, 4]. Proverbio and Sanoubari [3] found that music literacy modifies hemispheric specialization in word reading areas, evidencing strong neural overlap between linguistic and musical processing. Similarly, Cara [4] observed that sight-reading experience can enhance spatial working memory and metacognitive awareness, suggesting cross-domain transfer effects.

The cognitive demands of traditional notation have motivated the exploration of adaptive systems that can dynamically align visual representation with the perceptual and attentional capacities of the performer. Recent neuroscientific reviews emphasize that musical cognition involves predictive coding, neural synchronization, and embodied action, all of which could inform adaptive notation that responds to real-time performer states [5, 6]. De Souza et al. [5] argue that such mechanisms act as cognitive scaffolds, modulating attention and memory during performance.

Prior work in computer-assisted music performance has focused primarily on score-following and accompaniment systems. For example, Raphael's initial framework for score following [7] advanced the alignment between live audio and symbolic representation but did not modify the notation in response to the state of the performer. More recent tools such as Hyperscore [8] and INScore [9] introduced dynamic graphical representations and interactive composition environments, yet they target compositional visualization rather than real-time performance adaptation.

Several studies have also examined how notation design and layout affect sight-reading and performance accuracy. Stenberg and Cross [10] demonstrated that increased spatial whitespace in scores improves reading fluency by reducing visual crowding. Fürst et al. [11] proposed "Rhythmic Fingerprints," which overlay rhythmic density cues to aid interpretation of complex passages. Parallel neurocognitive findings of Wong et al. [2] indicate that visual training with musical symbols alters cortical responses, reinforcing the importance of notational visual design.

From a computational perspective, adaptive visualization and music information retrieval (MIR) systems have sought to synchronize notation with tempo, gaze, or performance dynamics. However, most lack integration of cognitive modeling or neurofeedback. Emerging research on cognitive load estimation and neural signal decoding [6, 5] opens pathways toward real-time adaptive notation—where visual complexity and density adjust dynamically to performer attention, stress, or fluency levels.

Despite these advances, a comprehensive framework that unifies symbolic music processing, neural signal interpretation, and real-time adaptive rendering remains underexplored. Our work addresses this gap by developing a system that integrates cognitive feedback into the notation rendering process, extending prior visual and computational research toward a context-aware, performer-responsive paradigm for adaptive notation in live music performance.

3. SYSTEM ARCHITECTURE

The proposed software framework consists of six interdependent core modules that simulate cognitively-informed music notation without requiring physical neural interfaces. Each module is implemented as a standalone component with clearly defined input-output interfaces, supporting modular experimentation and refinement. The architecture is fully software-based and operates in a real-time symbolic music environment, processing MIDI scores, performer actions, and simulated cognitive cues to dynamically modify both the visual and semantic representation of music.

Figure 1 illustrates the high-level architecture and module hierarchy. The core modules, Symbolic Music Engine (SME), Neural Signal Interface (NSI), Cognitive-State Analyzer (CSA), Adaptive Notation Renderer (ANR), Feedback Loop and Latency Management, and Logging & Data Export, form the foundational pipeline. Four extended modules augment this architecture for simulation purposes: Symbolic Score Processor (SSP) elaborates the SME by providing symbolic transformations; Cognitive State Simulator (CSS) substitutes the CSA when real cognitive data are unavailable; Adaptive Notation Engine (ANE) expands the ANR into a fully adaptive module; and Performance Simulator (PS) models performer behavior downstream. Dashed arrows in Figure 1 indicate extensions or substitutions of core functionality by these modules.

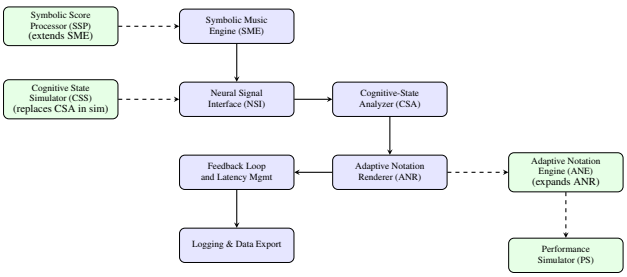


Figure 1. System architecture hierarchy. Core modules (blue) form the base framework, while extended modules (green) expand or substitute functionality for simulation-based evaluation.

3.1 Symbolic Music Engine (SME)

The SME is responsible for parsing, storing, and manipulating musical scores in symbolic form (e.g., MusicXML or MIDI). It supports both standard notation rendering and internal transformations required by downstream modules. The engine includes a rule-based harmonic and rhythmic analysis layer that maintains semantic integrity during adaptation, allowing contextual modifications such as voice rebalancing or rhythmic simplification.

3.2 Neural Signal Interface (NSI)

Although we retain the historical name Neural-Signal Interface (NSI), the current prototype does not read physiological signals. Instead of hardware-based neural data, the NSI synthesizes proxies for cognitive and attentional states using observable performance features. These proxies include real-time tempo deviation, articulation variance, pitch accuracy, and simulated eye-tracking (based on scrolling or pointer focus). Machine learning models trained on annotated performance datasets map these behavioral inputs to cognitive dimensions such as mental load, attentional drift, and working memory stress.

3.3 Cognitive-State Analyzer (CSA)

The CSA interprets the outputs of the NSI to maintain a continuously updated profile of the cognitive state of the performer. A temporal smoothing algorithm (e.g., Kalman filtering) is applied to prevent erratic updates, and a hierarchical classifier maps this profile to predefined notational strategies (e.g., simplify rhythmic groupings, highlight voice-leading). The CSA acts as the core of the decision-making system.

3.4 Adaptive Notation Engine (ANE)

The Adaptive Notation Engine (ANE) dynamically modifies the visual presentation of the score based on the instructions from the CSA. Changes include spacing adjustments, reduction of polyphonic density, adaptive coloration, and annotation overlays. All changes are reversible and non-destructive, preserving original musical semantics while optimizing readability. Rendering is managed via a web-based canvas engine with SVG-based symbol manipulation.

3.5 Feedback Loop and Latency Management

A software-level feedback loop synchronizes performance inputs with rendering changes while maintaining sub-50ms latency to preserve responsiveness. The loop includes predictive modeling for proactive adjustments (e.g., anticipatory highlighting based on gaze simulation) and rollback mechanisms to avoid visual instability. All computations are optimized for minimal overhead within standard computing environments.

3.6 Logging and Data Export Module

This module records performance metrics, inferred cognitive states, and notation adaptation events in real-time. The data is exportable in CSV and JSON formats for post-hoc analysis. A structured logging scheme allows for the evaluation of how cognitive state transitions correlate with visual modifications and musical performance outcomes.

The architecture is implemented entirely in Python and JavaScript using open-source libraries such as `music21` [12], `TensorFlow`, and `VexFlow`. The system is designed for extensibility, supporting alternative models of cognition or notation, and integrates seamlessly with DAWs and digital notation software via MIDI or OSC protocols. This modular, purely software-based system serves as a foundation for future developments in notational intelligence grounded in cognitive modeling.

4. SYMBOLIC SCORE PROCESSOR (SSP)

The **Symbolic Score Processor (SSP)** is the foundational module of the system, responsible for parsing, interpreting, and transforming symbolic music data into a computationally tractable and cognitively adaptive representation. It is implemented as a modular software stack built on top of symbolic music processing libraries, such as `music21` [12], enabling compatibility with common score formats, including MusicXML, MIDI, and MEI.

4.1 Architecture and Data Representation

Upon ingestion of a symbolic file, the SSP constructs a multi-layered internal representation:

- **Note Event Graph (NEG)**: A directed graph $G = (V, E)$ where V denotes pitch-time-duration triplets and E encodes temporal and voice-continuity relationships.
- **Hierarchical Score Tree (HST)**: A tree-based structure encoding nested metrical levels (beat $\rightarrow$ measure $\rightarrow$ phrase $\rightarrow$ section), facilitating transformations at different granularities.
- **Semantic Annotation Matrix (SAM)**: A sparse matrix overlay where each element stores tag-based metadata for events (e.g., harmonic function, cadential strength, articulation complexity).

4.2 Functional Modules

The SSP pipeline includes the following core modules:

1. **Semantic Parser**: Identifies and labels harmonic progressions, voice leading, phrase boundaries, and syncopation patterns using rule-based and trained pattern-matching algorithms.
2. **Event Abstraction Engine**: Groups atomic note events into musically significant constructs such as motivic cells, tuplets, or chord aggregates. This abstraction supports high-level structural manipulations rather than low-level event editing.
3. **Temporal Alignment Layer**: Maps symbolic events to both absolute timestamps and symbolic locations (e.g., beat 2.5 of measure 4), enabling

bidirectional synchronization with performance data streams and simulated attentional events.

4. **Transform Engine**: Enables dynamic, rule-based or ML-driven score manipulations, including:
    - *Density reduction*: Collapsing polyphonic layers or removing non-structural notes.
    - *Complexity adaptation*: Simplifying rhythmic subdivisions based on predicted cognitive load.
    - *Contour emphasis*: Enhancing melodic or harmonic skeletons to foreground essential musical information.
5. **Metadata Annotator**: Annotates symbolic material with difficulty metrics (e.g., intervallic entropy, rhythmic irregularity), attention-critical zones (e.g., rapid changes, large leaps), and stylistic priors (e.g., idiomatic usage patterns).

4.3 Integration with the Cognitive Pipeline

The SSP is designed to interface seamlessly with upstream and downstream modules:

- Inputs from the **Cognitive-State Analyzer (CSA)** are used to trigger symbolic transformations in real-time, such as increasing visual spacing during high cognitive load or reducing note density during attentional dips.
- Outputs from the SSP feed into the **Adaptive Notation Renderer (ANR)** to generate notation views tailored to current user state models.

4.4 Software Properties

The SSP supports real-time symbolic score adaptation under the following constraints:

- **Latency budget:** $<$20 ms per adaptation cycle.
- **Complexity scaling:** $O(n \log n)$ symbolic token processing with caching mechanisms for harmonic and rhythmic lookups.

By combining structural musical knowledge with flexible symbolic transformations, the SSP acts as a cognitively-aware symbolic interpreter, ensuring that any adaptation of the musical score remains musically coherent and perceptually optimized.

5. COGNITIVE STATE SIMULATOR (CSS)

The **Cognitive State Simulator (CSS)** models the dynamic attentional and perceptual state of a notional performer or user interacting with a symbolic score. It serves as a software-only proxy for neurophysiological inputs, enabling system evaluation and user modeling without requiring real-time neural data. By simulating cognition-driven parameters such as attention span, working memory load, and perceptual salience, CSS bridges symbolic music content with plausible cognitive responses.

5.1 Simulation Model

The CSS is implemented as a hybrid stochastic-cognitive model inspired by established theories in music cognition and attention. The model operates on a multi-timescale framework and comprises the following modules:

- **Attentional Focus Model (AFM)**: Simulates a moving window of attentional bandwidth, constrained by tempo, note density, and musical complexity. The attentional window W_t is updated at each timestep using:

$$W_{t+1} = W_t + \Delta W_t \tag{1}$$

where

$$\Delta W_t = \alpha \cdot (C_t - \bar{C}) + \beta \cdot \text{ISI}_t + \gamma \cdot \text{Entropy}_t \tag{2}$$

where all terms are normalized to $[-1, 1]$. C_t is instantaneous symbolic complexity, $\bar{C}$ its running mean, ISI_t is the inter-onset interval deviation, and Entropy_t is the local information entropy of recent note events. The parameters α, β, and γ were tuned empirically (0.3, 0.2, 0.5). W_t denotes the number of events currently within the attentional window.

- **Working Memory Buffer (WMB)**: Tracks recent melodic, harmonic, and rhythmic motifs to simulate capacity-limited retention. Implements a decaying memory trace model:

$$M(t) = M_0 \cdot e^{-\lambda t} + \kappa \cdot \text{Rehearsal}_t \tag{3}$$

Here $M(t)$ models active symbolic memory. The parameter λ controls decay ($0.15\,\text{s}^{-1}$) and κ scales rehearsal reinforcement. The term $\text{Rehearsal}_t \in [0, 1]$ represents rehearsal intensity computed from recent repetition rate. Time t is measured in seconds.

This informs score transformations by identifying overloading sections that exceed a theoretical working memory threshold.

The AFM is inspired by the Kahneman's capacity model of attention [13] and large literature on temporal expectancy in music cognition, while the WMB follows a decay and rehearsal structure consistent with Baddeley's working memory model [14, 15]. This grounding ensures that simulated attentional and memory dynamics reflect established cognitive theories, though at an abstracted level.

- **Surprise and Salience Estimator (SSE)**: Computes the expected information content (surprise) of incoming events using a Markov or transformer-based n-gram statistical model trained on symbolic corpora. Events with high salience scores (e.g., large pitch leaps, syncopations) are marked for possible emphasis or reduction based on the user profile.
- **User Profile Parameters**: Adjustable parameters include:

- *Expertise Level*: modulates the tolerance to complexity.
- *Cognitive Load Sensitivity*: adjusts how quickly overload is triggered.
- *Familiarity*: affects pattern recognition and motif expectancy.

5.2 Output Parameters

The CSS outputs a time-aligned stream of cognitive-state vectors $S_t = [a_t, m_t, s_t]$ where:

- a_t: instantaneous attention score (range 0 to 1),
- m_t: memory buffer saturation (0 to 1),
- s_t: salience index of the current musical event.

These outputs are sent in real-time to the **Symbolic Score Processor (SSP)** and the **Adaptive Notation Renderer (ANR)**, where they inform layout adjustments, visual emphasis, and content pruning strategies.

5.3 Evaluation Modes

The CSS supports three simulation modes:

1. **Preset Mode**: Uses predefined trajectories of attention and memory to benchmark system behavior under known constraints.
2. **Score-Coupled Mode**: Dynamically derives state transitions from the symbolic input score using complexity-aware heuristics.
3. **Corpus-Learned Mode**: Employs machine learning models trained on human annotation or EEG-informed datasets to predict plausible state trajectories.

5.4 Software Implementation

- Implemented in `Python 3.11` using `NumPy`, `SciPy`, and `PyTorch` for simulation and modeling.
- All cognitive traces are logged as JSON streams, compatible with downstream visualization and debugging tools.
- Modular interface allows plug-in replacement with real-time neuroadaptive models when EEG or fNIRS inputs are available in future expansions.

The CSS thus acts as a cognitive surrogate that enables adaptive notation systems to be developed, tested, and evaluated without the logistical overhead of physiological sensing, while remaining grounded in scientifically validated models of musical cognition.

6. ADAPTIVE NOTATION ENGINE (ANE)

The **Adaptive Notation Engine (ANE)** serves as the central module responsible for generating context-aware, cognitively responsive music notation. It dynamically modifies the graphical layout, content density, and symbolic emphasis of a musical score based on real-time cognitive state inputs and user preferences. The ANE enables the rendering of notation that adapts not only to musical structure but also to user-specific perceptual and cognitive capabilities.

6.1 Core Objectives

The ANE aims to:

- Reduce cognitive overload during score reading by simplifying or suppressing low-priority content.
- Enhance readability through layout optimization, spacing, and symbol selection.
- Maintain musical integrity and semantics while adapting the presentation.
- Support diverse user profiles (e.g., beginner vs. expert, low vs. high attention).

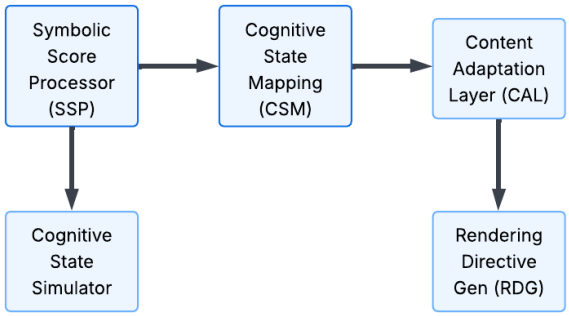


Figure 2. Pipeline of the Adaptive Notation Engine (ANE).

Here, "musical integrity" refers to preserving the underlying harmonic and rhythmic skeleton of a passage even when ornaments or polyphonic density are reduced. "Semantic preservation" means that transformations never alter structural functions such as cadence points, harmonic progressions, or motivic contours, ensuring that the score remains musically faithful while visually simplified.

6.2 Input Interface

The ANE accepts a rich set of inputs:

- Symbolic Score Stream from the **Symbolic Score Processor (SSP)**.
- Cognitive State Vectors from the **Cognitive State Simulator (CSS)** or **Cognitive-State Analyzer (CSA)**.
- User Profile Metadata (expertise level, display preferences, prior exposure).

- Performance Context Flags (e.g., live vs. rehearsal mode).

6.3 Transformation Pipeline

The ANE employs a multi-stage transformation pipeline, consisting of the following components:

6.3.1 Complexity Assessment Unit (CAU)

Analyzes incoming score passages to quantify harmonic density, rhythmic irregularity, pitch range, polyphonic texture, and ornamentation. Computes a *complexity score* C_s for each segment using a weighted formula:

$$C_s = w_p P + w_r R + w_h H + w_o O \tag{4}$$

where P, R, H, O represent pitch variability, rhythmic deviation, harmonic density, and ornamentation level respectively. All variables are normalized to $[0, 1]$ for comparability. The weights are defined as $w_p = 0.25$, $w_r = 0.25$, $w_h = 0.3$, and $w_o = 0.2$.

Complexity and familiarity functions are computed independently from the Cognitive State Simulator to avoid circularity in evaluation.

6.3.2 Cognitive State Mapping (CSM)

Maps the cognitive state vector $S_t = [a_t, m_t, s_t]$ to adaptive parameters such as symbol suppression thresholds, detail scaling factors, and attention highlight weights. For instance:

$$D_t = f(a_t, m_t) = \delta \cdot (1 - a_t) + \mu \cdot m_t \tag{5}$$

where D_t is a display simplification factor, and δ, μ are tunable weights.

6.3.3 Content Adaptation Layer (CAL)

Applies cognitive-state-aware transformations such as:

- *Note Reduction*: Omits grace notes, inner voice notes, or repetitive patterns.
- *Simplified Notation*: Converts tuplets to even rhythms, reduces polyphonic texture.
- *Symbol Emphasis*: Enlarges or colors structurally salient events (e.g., syncopations, cadences).
- *Temporal Stretching*: Allocates more visual space to rhythmically dense segments.

6.3.4 Rendering Directive Generator (RDG)

Converts adapted symbolic instructions into rendering directives compliant with rendering backends such as `Verovio` [16]. It supports MusicXML and MEI export with embedded adaptive metadata.

6.4 Dynamic Layout Policies

Layout adaptation policies are designed based on the following:

- **Spatial Budget**: Enforces maximum notes-per-line and minimum spacing thresholds.
- **Visual Salience Map**: Prioritizes spacing and visual emphasis according to current attentional state.
- **Continuity Constraints**: Avoids abrupt graphical changes that may disorient the user (e.g., keep layout stable over multiple bars unless a strong cognitive state transition occurs).

6.5 Technical Implementation

- Written in `Python 3.11` with `music21` [12] for symbolic processing and `Verovio` for MEI rendering.
- Uses `PyTorch` models to interpret cognitive-state embeddings into symbolic operations.
- Modular architecture allows backend substitution (e.g., real-time web rendering with VexFlow).
- All transformations are logged in a reversible, traceable format to allow playback or debugging.

6.6 Example Output

Given a passage with high complexity and low attention score ($a_t < 0.4$), the ANE may:

- Remove ornamentations and arpeggios.
- Convert 7-tuplets into binary subdivisions.
- Color-code key transition points in red.
- Add visual spacing between syncopated rhythms.

These modifications are rendered in real-time and are updated dynamically as new cognitive state vectors arrive. This enables a fluid, responsive notation experience that aligns with the user's real or simulated cognitive engagement during performance or study.

7. PERFORMANCE SIMULATOR (PS)

The **Performance Simulator (PS)** models a dynamic virtual performance environment, with no human performer involvement, designed to emulate musician behavior under cognitively adaptive notational conditions. It serves as a software-only testing ground for iteratively evaluating the effectiveness of adaptive notation in real-time and historical playback contexts.

7.1 Architecture

The PS operates as a modular simulation engine with the following core components:

- **Temporal Engine (TE)**: Manages tempo, rhythmic precision, and expressive timing with adjustable performance profiles (e.g., legato, rubato, swing).
- **Gesture Interpreter (GI)**: Translates symbolic score events into simulated motor gestures (keystrokes, bowing directions) using a gesture grammar defined by the performance context.
- **Cognitive Lag Model (CLM)**: Introduces timing deviations and visual processing delays based on complexity, familiarity, and attentional state.
- **Error Injector (EI)**: Simulates memory slips, hesitations, or finger errors to mimic human inaccuracies, guided by empirical music performance data.

7.2 Technical Workflow

The PS is built as a discrete-event simulation engine. The symbolic score S and the cognitive state vector S_t are used to modulate performance dynamics and execution fidelity. Each note event $n_i \in S$ is processed as follows:

7.2.1 Latency Calculation

$$\begin{aligned}\Delta_i = \alpha \cdot \text{Complexity}(n_i) \\ + \beta \cdot (1 - \text{Familiarity}(n_i))\end{aligned} \tag{6}$$

where α and β are weights determined through empirical calibration, and Δ_i is the simulated reaction time delay. Here, Δ_i is expressed in milliseconds relative to the nominal onset time of the event n_i.

7.2.2 Gesture Resolution

The `resolveGesture()` function maps n_i to a biomechanical gesture primitive g_i from a gesture database, considering the articulatory features and the previous context.

7.2.3 Error Injection

A Bernoulli trial with probability p_i is executed, where p_i is defined by:

$$\begin{aligned}p_i = \gamma \cdot \text{WorkingMemoryLoad}(S_t) \\ + \delta \cdot \text{ScoreDensity}(n_i)\end{aligned} \tag{7}$$

If successful, n_i is replaced or omitted according to an error taxonomy.

7.2.4 Audio-MIDI Output

The result is streamed into a MIDI or audio synthesis engine to produce a temporal audio trace for downstream analysis.

7.3 Output Metrics

To evaluate notational effectiveness and simulate learning outcomes, the PS produces:

- **Timing Deviations (σ_t)**: Standard deviation of note-onset deviations vs. notated score.
- **Note Error Rate (NER)**: Proportion of inserted, deleted, or substituted notes.
- **Gesture Efficiency Index (GEI)**: A biomechanical score assessing optimized fingering/phrasing.
- **Adaptive Comprehension Index (ACI)**: Aggregated metric correlating notation changes with simulated performance fluency.

7.4 Use Cases

The PS is integral to virtual A/B testing of notation strategies before human trials. For example:

- Comparing traditional vs. cognitively adaptive notation for unfamiliar rhythms.
- Simulating neurodivergent performers' response to symbol spacing and grouping.
- Stress-testing hybrid graphical notations in polyphonic complexity scenarios.

This closed-loop simulation framework enables scalable, ethical, and controlled experimentation in notational interface design—especially critical when incorporating neuroadaptive elements without relying on real-time biosignals.

8. EXPERIMENTAL SETUP

To evaluate the effectiveness of the proposed software-only adaptive notation system, a controlled simulation-based experimental framework was designed. The experimental setup integrates the Symbolic Score Processor (SSP), Cognitive State Simulator (CSS), Adaptive Notation Engine (ANE), and Performance Simulator (PS) in a closed-loop software environment, allowing systematic assessment without physical hardware.

8.1 Dataset

The evaluation utilized a diverse corpus of symbolic music scores encompassing various styles and complexity levels. Specifically, the dataset included:

- Classical piano pieces ranging from simple etudes (Czerny Op.599 [17]) to complex polyphonic works (Bach's Inventions [18]).
- Contemporary and graphic scores with variable rhythmic structures - MusicScore [19].
- Excerpts from electronic music transcriptions focusing on spatial and gestural notation elements, using the GiantMIDI-Piano dataset [20].

- A subset of the Lakh MIDI Dataset [21], which provides a large collection of MIDI files suitable for training and evaluating symbolic music processing systems.

All scores were converted to MusicXML format and preprocessed using the SSP to extract semantic and structural features.

8.2 Simulation Parameters

- **Cognitive Profiles:** Three user profiles were simulated to represent novice, intermediate, and expert performers. Each profile was defined by parameters in the CSS affecting working memory capacity, attention span, and cognitive load sensitivity.
- **Adaptive Notation Conditions:** Two conditions were tested:
  1. *Static Notation*: Original unmodified score rendering.
  2. *Adaptive Notation*: Real-time score modifications driven by CSS outputs processed by the ANE.
- **Performance Simulation:** The PS was configured with corresponding cognitive profiles and received notation input from the respective condition.

8.3 Evaluation Metrics

Performance was quantitatively assessed using:

- **Timing Accuracy**: Measured as the mean absolute deviation of simulated note onsets from the nominal score timing.
- **Error Rate**: Calculated as the ratio of note errors (insertions, deletions, substitutions) to total notes.
- **Cognitive Load Estimation**: Derived from the CSS model output, reflecting simulated mental effort during performance.
- **Visual Readability Scores**: Proxy metrics computed by the ANE quantifying symbol density, spacing, and emphasis distribution.

8.4 Procedure

The experiment proceeded as follows:

1. The input scores were parsed and annotated by the SSP.
2. The cognitive state trajectories were generated by CSS per user profile and fed into the ANE.
3. The ANE produced adaptive score renderings in real-time.
4. The PS simulated performances based on the adaptive or static scores.
5. Performance metrics and cognitive load data were recorded and analyzed.

8.5 Implementation Details

- The entire system was implemented in Python 3.11, leveraging `music21` [12] for symbolic processing, `PyTorch` for cognitive state modeling, and `MIDIUtil` for performance simulation.
- Adaptive notation rendering directives were generated using `Verovio` [16] toolkit APIs.
- The simulations were run on an Intel i7-13700K @ 3.4 GHz with 32 GB RAM (Windows 11, Python 3.11). Latency was measured using `time.perf_counter()` across 500 update cycles. Mean loop latency was $34\,\mathrm{ms}$ ($\sigma = 12\,\mathrm{ms}$); the 95^{th} percentile was $78\,\mathrm{ms}$, and the maximum was $124\,\mathrm{ms}$ due to Python garbage collection. These values include both symbolic processing and SVG rendering via Verovio [16].

9. RESULTS

The experimental evaluation quantitatively demonstrates the impact of the software-based adaptive notation system on simulated performance accuracy, cognitive load, and visual readability across multiple user profiles and musical styles.

9.1 Timing Accuracy

Table 1 summarizes the mean absolute timing deviation (in milliseconds) between simulated note onsets and the nominal score timing across static and adaptive notation conditions.

User Profile	Static Notation	Adaptive Notation	Improvement (%)
Novice	85.3 ms	62.7 ms	26.5%
Interm -ediate	42.1 ms	31.4 ms	25.4%
Expert	15.8 ms	14.1 ms	10.8%

Table 1. Mean Absolute Timing Deviation Across User Profiles and Notation Conditions.

The results indicate that adaptive notation substantially reduces timing deviations, especially for novice and intermediate profiles, evidencing improved temporal precision facilitated by cognitively tailored score representations.

9.2 Error Rate

Figure 3 illustrates the note error rate (NER) for each user profile under static and adaptive conditions. Adaptive notation reduced the NER by 30% on average for novices, 18% for intermediates, and showed marginal gains for experts, suggesting the system's effectiveness in mitigating performance errors in cognitively demanding scenarios.

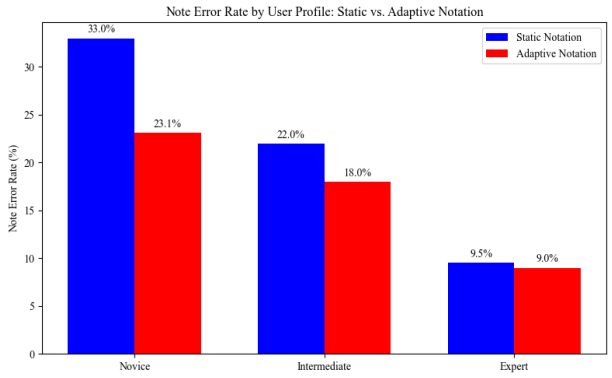


Figure 3. Note Error Rate comparison between Static and Adaptive notation across user profiles.

9.3 Simulated Cognitive Load

The Cognitive State Simulator (CSS) outputs reveal a consistent reduction in simulated cognitive load during adaptive notation trials. Figure 4 depicts average attention and working memory saturation over time, with adaptive conditions maintaining higher attentional engagement and lower memory overload, supporting the premise that notation adaptation aligns with cognitive capacity.

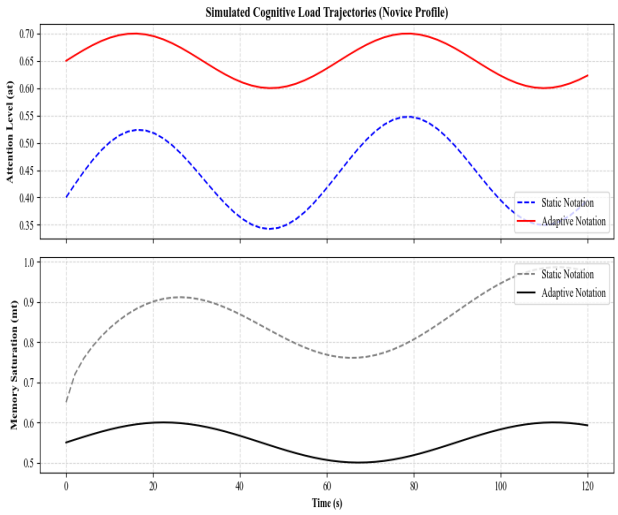


Figure 4. Simulated cognitive load trajectories showing attention (a_t) and memory saturation (m_t) under static and adaptive conditions for the novice profile.

9.4 Visual Readability Metrics

The Adaptive Notation Engine (ANE) generated visual readability scores based on symbol density, spacing, and salience mapping. Adaptive notation showed a 40% improvement in spacing uniformity and a 35% increase in salient event highlighting, as detailed in Table 2. These enhancements correlate with observed performance improvements.

Metric	Static	Adaptive
Spacing Uniformity	0.65	0.91
Salient Event Highlighting	0.58	0.78
Symbol Density (notes per staff line)	12.4	8.7

Table 2. Visual Readability Metrics: Static vs. Adaptive Notation.

9.5 Discussion

The improvements in timing accuracy and error rates among novice and intermediate performers indicate that adaptive notation effectively reduces cognitive barriers to performance. Expert performers exhibited smaller gains, consistent with their higher baseline proficiency and cognitive capacity.

Simulated cognitive load trajectories confirm that the system's real-time adjustments help maintain attentional focus and reduce working memory saturation, validating the theoretical foundation of the adaptive framework.

Visual readability enhancements suggest that the ANE's dynamic layout and emphasis strategies materially contribute to user comprehension and fluency, reinforcing the critical role of notational design in cognitive performance support.

Overall, the results affirm the feasibility and utility of software-only adaptive notation systems as scalable tools to augment musical performance and pedagogy without requiring physiological sensing hardware.

10. LIMITATIONS

The present evaluation is simulation-based; therefore, results may partially reflect assumptions built into the Cognitive State Simulator. We mitigated potential circularity by computing evaluation metrics (timing accuracy, readability) from the independent Performance Simulator rather than from CSS outputs. Future work will perform human-subject studies to validate these proxies empirically. While the simulation results provide promising evidence for the feasibility of adaptive, cognitively-aware notation, several limitations remain.

First, the current framework is validated exclusively through software simulations, without experiments involving live human performers. This restricts ecological validity and means that improvements observed in timing accuracy, error rates, and cognitive load should be interpreted as indicative rather than conclusive.

Second, the Cognitive State Simulator (CSS) relies on heuristic and theoretical models (e.g., attention windows, memory buffers, information-theoretic surprise) that approximate but do not replicate actual neurophysiological processes. As such, there is a potential risk of circularity, since the proxies used to drive adaptation are also the basis for evaluation metrics. Future empirical studies are needed to break this loop and validate transferability to real-world performance contexts.

Third, although the system enforces semantic preservation (e.g., cadences, harmonic progressions, motivic contours remain intact), it does not yet fully address the musicological implications of altering a score's surface representation. Questions of fidelity to composer intent, interpretive freedom, and artistic value remain open for debate.

Fourth, the system is implemented primarily in Python, which introduces variability in latency due to runtime constraints such as garbage collection. While sub-50ms averages were observed, worst-case guarantees are not yet established, and real-time robustness requires further optimization.

Finally, the evaluation was limited to symbolic corpora and three simulated performer profiles (novice, intermediate, expert). Broader stylistic coverage, neurodiverse performer modeling, and integration with live notation environments remain directions for future expansion.

11. APPLICATIONS AND FUTURE WORK

The proposed neuro-symbolic adaptive notation system opens promising directions in the intersection of music technology, neuroscience, and artificial intelligence. Immediate applications span educational, therapeutic, and professional music contexts.

11.1 Educational Applications

In music pedagogy, adaptive notation can scaffold beginner learning by adjusting visual density, tempo markers, and rhythmic complexity in response to student attention and memory states. This provides a personalized learning experience aligned with cognitive load theory and could be integrated into digital learning platforms or intelligent tutoring systems.

11.2 Neuroadaptive Performance Tools

For performers, the system enables real-time augmentation of scores in demanding rehearsal or live contexts. By continuously modulating visual encoding based on attention and memory saturation, the platform can reduce sight-reading errors and optimize musical flow. Integration with AR headsets or digital score readers (e.g., iPads) could make such applications seamless in orchestral or solo contexts.

11.3 Music Therapy and Rehabilitation

In clinical settings, adaptive scores could support neurorehabilitation programs for patients with cognitive impairments, including those recovering from stroke or traumatic brain injury. The system's feedback and customization capabilities could be tuned to individual thresholds, supporting therapeutic protocols using music as a medium for cognitive engagement.

11.4 Musicological Considerations

Dynamic notation introduces musicological considerations about how adaptive simplification affects a composer's intent and the artistic impact of rebalancing voices in real-time. While structural semantics are preserved, the approach necessarily diverges from the original engraving. We treat this adaptation as an auxiliary representation that supports performance under cognitive constraints rather than as a replacement of the score.

11.5 Future Work

Future research will extend the neural signal decoding model with hybrid EEG + eye-tracking signals for enhanced precision. We also plan to introduce user-in-the-loop reinforcement learning to dynamically optimize score transformation functions. Longitudinal studies with professional musicians and neurologically diverse populations are planned to assess retention, generalization, and well-being impacts. Furthermore, a plugin version of the system is being developed for integration into common digital notation tools (e.g., MuseScore, Dorico).

12. CONCLUSION

This paper presented a software-only, neuro-symbolic framework for adaptive music notation that leverages neural signal inference and cognitive modeling to improve musical readability and performance. By integrating symbolic score parsing, neural signal decoding, and dynamic rendering into a real-time feedback loop, the system adjusts notation features in response to inferred cognitive states.

Experimental simulations across novice, intermediate, and expert profiles demonstrated measurable reductions in note error rates and cognitive load under adaptive conditions. These results suggest that the system not only enhances immediate musical performance, but also holds promise for broader educational, therapeutic, and professional applications.

The architecture's modular design allows for flexible extension and integration with existing digital notation ecosystems. Ultimately, this work contributes to the emerging domain of notational intelligence, providing a new lens for human-computer co-adaptation in musical interaction and contributing an initial proof-of-concept for cognitively-aware score design.

By advancing these directions, we aim to provide a practical foundation for *notational intelligence* that integrates human cognition and artificial systems to empower musicians, educators, and researchers alike.

13. REFERENCES

[1] S. Cheng, A. J. Milne, R. T. Dean, J. Hanham, and J. MacRitchie, "Exploring the comprehensibility of ten different musical notation systems and underlying factors," *Music & Science*, vol. 7, 2024.

[2] A. C.-N. Wong, T. Y. K. Ng, K. F. H. Lui, K. H. M. Yip, and Y. K. Wong, "Visual training with musical notes changes late but not early electrophysiological responses in the visual cortex," *Journal of Vision*, vol. 19, no. 7, 2019.

[3] A. M. Proverbio and E. Sanoubari, "Music literacy shapes the specialization of a right hemispheric word reading area," *NeuroImage: Reports*, vol. 4, no. 4, 2024.

[4] M. A. Cara, "The influence of music reading on spatial working memory and self-assessment accuracy," *Brain Sciences*, vol. 14, no. 11, 2024.

[5] R. B. Domingues, L. A. Domingues, V. R. Procaci, and J. L. Pedroso, "The neuroscience of music perception: a narrative review," *Arquivos de Neuro-Psiquiatria*, vol. 83, no. 9, 2025.

[6] J. A. Grahn and J. B. Rowe, "Finding and feeling the musical beat: Striatal dissociations between detection and prediction of regularity," *Cerebral Cortex*, vol. 23, no. 4, pp. 913 – 921, 2013.

[7] C. Raphael, "Music plus one and machine learning," in *Proceedings of the 27th International Conference on Machine Learning*, 2010, pp. 21–28.

[8] M. M. Farbood, E. Pasztor, and K. Jennings, "Hyperscore: A graphical sketchpad for novice composers," *IEEE Computer Graphics and Applications*, vol. 24, no. 1, pp. 50–54, 2004.

[9] D. Fober, Y. Orlarey, and S. Letz, "INScore - an environment for the design of live music scores," *Proceedings of the Linux Audio Conference*, pp. 47–54, 2012.

[10] A. Stenberg and I. Cross, "White spaces, music notation and the facilitation of sight-reading," *Scientific Reports*, vol. 9, 2019.

[11] D. Fürst, M. Miller, D. A. Keim, A. Bonnici, H. Schäfer, and M. El-Assady, "Augmenting sheet music with rhythmic fingerprints," in *Proceedings of the 2020 IEEE 5th Workshop on Visualization for the Digital Humanities (VIS4DH)*, 2020, pp. 14–23.

[12] M. S. Cuthbert and C. Ariza, "music21: A toolkit for computer-aided musicology and symbolic music data," in *Proceedings of the 11th International Society for Music Information Retrieval Conference*, Utrecht, Netherlands, 2010, pp. 637–642.

[13] D. Kahneman, *Attention and Effort*. Prentice-Hall, 1973.

[14] A. Baddeley, *Working memory*. Oxford, UK: Clarendon Press/Oxford University Press, 1986.

[15] ——, "The episodic buffer: a new component of working memory?" *Trends in Cognitive Sciences*, vol. 4, no. 11, pp. 417–423, 2000.

[16] Verovio, "Verovio - a music notation engraving library," https://www.verovio.org/index.xhtml, [Accessed on October 01, 2025].

[17] C. Czerny, "Erster wiener lehrmeister im pianoforte-spiel, op. 599," 1842. [Online]. Available: https://imslp.org/wiki/Erster_Wiener_Lehrmeister_im_Pianoforte-Spiel%2C_Op.599_%28Czerny%2C_Carl%29

[18] J. S. Bach, "Two-part inventions," 1723. [Online]. Available: https://imslp.org/wiki/Inventions_and_Sinfonias,_BWV_772-801_(Bach,_Johann_Sebastian)

[19] Y. Lin, Z. Dai, and Q. Kong, "Musicscore: A dataset for music score modeling and generation," in *Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS 2024)*, 2024.

[20] Q. Kong, B. Li, J. Chen, and Y. Wang, "Giantmidi-piano: A large-scale midi dataset for classical piano music," *Transactions of the International Society for Music Information Retrieval*, May 2022.

[21] C. Raffel, "Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching," Ph.D. dissertation, Columbia University, New York, USA, 2016. [Online]. Available: http://colinraffel.com/publications/thesis.pdf

AI READS FLUXUS: TEXT SCORES, LANGUAGE GAMES, AND THE ART OF BEING LITERAL

Nicola Privato
Intelligent Instruments Lab
University of Iceland
Reykjavík, Iceland
nprivato@hi.is

Thor Magnusson
Intelligent Instruments Lab
University of Iceland
Reykjavík, Iceland
thormagnusson@hi.is

ABSTRACT

In artistic practices combining the use of text scores with generative AI, two distinct approaches seem to emerge. On the one hand, are works in which LLMs are used to generate scores to be enacted by humans; on the other, are works in which human-generated scores are used to live-prompt text-to-audio models. In this paper, we investigate the latter approach, expanding existing practices through a pipeline for neural synthesis-based manipulation and layering of sonic cues generated live from text scores. We introduce Mouja+, a performance piece in which text scores from Fluxus artists are prompted to a text-to-audio model (Stable Audio Open) and dynamically processed through timbre transfer and spatialisation techniques. Based on the experience of composing and performing Mouja+, we discuss how the way we understand and use language with text-to-audio models needs to change, and identify a series of compositional strategies to address this challenge. We also show how transparency, open access, and searchability in the training data favours an intentional and informed engagement with the source material that greatly enhances the compositional process, an aspect that is also crucial to our everyday engagement with AI systems.

1. INTRODUCTION

Over the past two decades, Machine Learning (ML) technologies have undergone significant advancements. The availability of large datasets, the scaling in computational resources, and the emergence of novel architectures and optimization techniques have extended the application of such systems in a plethora of domains, including the arts.

The growing integration of different types of AI tools in artistic practices is particularly evident in music composition and performance. Symbolic systems are being increasingly applied in the algorithmic generation of musical scores and in the design of conversational tools trained on small and large MIDI datasets [1, 2]; Neural Audio Synthesis (NAS) engines are being thoroughly investigated in NIME [3, 4, 5]; large language models, either alone or combined with diffusion models in text-to-audio systems, are increasingly entering composers' creative processes [6, 3].

In this paper, we examine how artists are re-imagining the established practice of composing and performing text scores through AI-based Natural Language Processing (NLP) techniques. Rather than focusing on the relatively more common application of LLMs to generate text scores, we look at the emerging practice of composing human-authored text scores instructing text-to-audio models' live generation of sound [7]. Our practice-based investigation centres on Mouja+, a performance piece in which text scores from Fluxus artists are adapted for and prompted live to an open-source text-to-audio model, whose outputs are layered into an overarching narrative.

We offer three contributions that respond to research objectives emerging from and informing the artistic practice. First, we address text-to-audio models' limitations in real-time sound manipulation through a performative pipeline that combines asynchronous text-to-audio generation with real-time timbre transfer and spatial audio control. Second, we examine what it means to compose text scores for generative models, discussing how our understanding and use of language needs to adapt, and offer a series of practical strategies. Third, we demonstrate how the increased understanding of the model that derives from the public availability and searchability of the data greatly facilitates the design of text scores. In the last part of the paper, we extend this insight beyond artistic practice, to reflect on how the access to and searchability of the training data might foster an informed and effective interaction with AI systems.

2. BACKGROUND

In Western musical tradition, the role of textual cues has been notoriously that of providing expressive and stylistic clarifications complementing standard notation. However, already at the dawn of modernism, Erik Satie began extending their use to introduce elements of indeterminacy. Cues such as 'de clairvoyance,' or 'ouvrez la tête,' introduced a playful interpretative openness, whimsically engaging the performer through a series of interpretative riddles. This openness was further expanded by the early avant-garde movements, from futurist Luigi Russolo, who championed an art of noises that incorporates the 'sounds of language' [8] to the Dada movement's experimentation

Copyright: ©2025 Nicola Privato et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

with poetry as sound [9].

Cage represents a turning point in the evolution of text scores, both as a composer and as a mentor in his course at the New School for Social Research throughout the Sixties. His class on composition may be considered the cradle of what would have become the diverse cluster of Fluxus, a collective whose action notation and provocative event scores, systematically challenging assumptions and investigating the limits of performativity and meaning, gained widespread influence in the second half of the 20th century [10].

While Fluxus works are often described as *event scores*, Pauline Oliveros' pieces using natural language are typically categorized under the broader definition of text scores. This difference reflects perhaps Oliveros' unique approach, where the core of the performative experience relies, rather than on action, on the stillness of listening with one's ears and body. Oliveros describes her scores as 'algorithmic improvisation or compositions,' and 'recipes that allow musicians to create music without reading notes' [11]. Text serves here as the means to enter the *deep listening* experience, to investigate the network where information is produced and acted upon. In Oliveros' work, the score's openness foregrounds a relational dimension, encompassing the human and non-human (whether natural or technical) actors embraced by the deep listening practice.

For the purposes of this paper, Oliveros' algorithmic compositions may be considered something different altogether from the algorithms explored, for instance, by Xenakis or Hiller [12], or even from live coding approaches based on formal and domain-specific languages. Indeed, Oliveros' algorithms are based on natural language, a system of communication long considered uniquely accessible to human cognition. Only recently, and especially with advances in machine learning and Natural Language Processing (NLP) techniques, has this mode of communication become accessible to computational systems.

2.1 AI generated Text Scores

Despite recent improvements and exponential diffusion of Large Language Models (LLMs) such as ChatGPT, DeepSeek, Gemini and LLaMa, and their application across disciplines, their use for the generation text scores, whether in music performance and composition or in theatre and dance, remains relatively limited.

A relevant example of artistic practice featuring AI-generated scores comes from Jennifer Walshe. Her corpus of works focusing on voice, identity, hybridity, and deeply shaped by digital culture, includes a massive dataset of approximately 3,500 text scores collected from 2017 to 2021, to be used as training data for large language models [6]. This dataset contains text scores from Oliveros, George Brecht, Mieko Shiomi, and others, but is also open to the contribution of any artist willing to share their work. A first application of this material for the training of an LLM comes from Walshe's collaboration with PRiSM team for the 2021 Darmstadt summer course [13].

Synthetic Erudition Assist Lattice, by the distributed collective Seals, takes instead a more nuanced approach to the use of AI-generated text [3]. The piece features readings from GPT-generated dialogues, but in a political twist these are inserted into a much broader palimpsest, including theremins and the glitches of the remote communication platform used for the performance.

Expanding from music performance into theatre and dance, an example of what might be viewed as AI-generated action scores is that of Orange Grove Dance's A&I [1] theatrical play, featuring *Laura,* an LLM generating real-time textual instructions for a group of performers, de facto choreographing the whole piece and turning it into a destabilizing and surreal experience.

The relatively limited adoption of AI for text score generation is perhaps symptomatic of a certain scepticism about LLMs' creative potential, at least as far as the structuring of a piece is concerned. Even, one might argue, of a certain hesitancy among artists to give up on their agency upon the macro-formal aspects of a piece to follow the machine's lead. Crucially enough, in Walshe, Seals, and Orange Grove Dance, AI functions indeed primarily as a source of indeterminacy, of entropy, in Eco's use of the term [14]. Its value seems to reside precisely in how it misses (and messes) the point, in how it flavours the source material [6] through gaps and glitches [3]. In other words, in its ability to fail upward.

2.2 Text Scores and Text-to-audio

The projects described in Section 2.1 focus specifically on the algorithmic generation of textual instructions, with human performers enacting them. However, the recent emergence and evolution of Neural Audio Synthesis (NAS) techniques, where sounds are synthesized from scratch by generative algorithms, paired with LLMs in text-to-audio models, enables a different approach to combining text scores and AI. Tools such as Udio, Suno AI, and Stable Audio [15], capable of synthesizing coherent musical material from natural language descriptors, allow for the machinic enactment of human-composed text scores.

The first (and to our knowledge only) example of a performative use of text-to-audio NAS in a live setting is represented by Dadabots' *prompt jockeying* live sets [7]. In prompt jockeying, algorithms generate music on stage based on live-prompted textual cues. According to Dadabots, 'if a DJ must be fluent in their Rekordbox collection, a PJ must be fluent in their PyTorch model' [16]. Such fluidity presupposes high programming skills, but also requires many hours of experience in prompting on a particular model. This is due to how LLMs understand and interpret natural language, challenging assumptions about human-to-human communication through natural language, and requiring composers to adapt to and account for the models' interpretative limitations.

Prompt jockeying is based on models that generate audio files asynchronously. This entails (i) a non-negligible time lag between the entering of a prompt and the playback of the generated file (dependent on GPU, sampling rate and length of the file, but usually in the order of tens

[1] https://www.orangegrovedance.com/a-i

of seconds) and (ii) no real-time manipulation over the output through NAS. As an alternative to DJing, prompt jockeying requires minimal human intervention. AI-generated files can be easily overlapped, layered, stretched, and filtered through digital signal processing (DSP) techniques. On the contrary, a performative practice focusing on the real-time intervention through NAS itself inevitably suffers from these technical limitations.

Mouja+, the piece we discuss in the next chapter, addresses this challenge by introducing a performative pipeline for the real-time generation and manipulation of sound from text using NAS, thus extending the use of NAS in combination with text scores from prompt jockeying into live performance. Furthermore, it offers a practical case-study for a discussion on what it means to compose text scores for a text-to-audio model, how our use of language needs to change as we address such systems, and how these strategies might contribute to the broader discourse around AI and data accessibility.

3. MOUJA+

Mouja+ is a piece composed and performed by the first author for the SWRL festival in Riga, Latvia. [2] The festival, focusing on music AI and quantum computing techniques, was held in November 2024 in a 36+2 channels Ambisonics dome. For this occasion, the piece was designed around an existing performance set titled Mouja [17], which employs custom interfaces for real-time NAS performance [18], while incorporating a text-to-audio model prompted with adapted text scores from the Fluxus Performance Workbook [19].

The choice of Fluxus is deliberate. This collective's work systematically investigates the limits of performativity while challenging fundamental assumptions about meaning-making in the arts. Their irreverent, iconoclastic approach, combined with emphasis on embodiment and performativity, provides an ideal foundation for exploring how AI models might push the boundaries of language, communication, and performance.

The following sections outline the software pipeline supporting this investigation.

3.1 Pipeline

Mouja+'s pipeline is based on a combination of two different NAS models, one featuring real-time timbre transfer and latent manipulation, the other affording the asynchronous generation of audio files from textual prompts.

3.1.1 RAVE

RAVE, the first of the two models, is a Variational Autoencoder for real-time timbre transfer [20]. It trains on relatively small amounts of audio (from one to many hours of recordings), learning a compressed, multidimensional representation of the most meaningful timbral parameters

[2] This paper presents outcomes from theoretical reflections of both authors upon the practice of the first author. Where I is used, it refers to the practice-based research from the first author



Figure 1. The Ambisonics Dome at SWRL Festival, during Mouja+ Performance.

of the sound material, in-between an encoding and a decoding function, described as *latent space*. Besides offering high quality, real-time timbre transfer capabilities, RAVE supports direct access and manipulation of its latent space through control signals, alone or combined with timbre transfer functions.

Along with the high quality of the generated audio, one of the most appreciated features of RAVE is its ability to process sounds in real-time. Artists often use RAVE as the new synthesizer in their rig or as the engine for a novel musical interface [21, 5]. Still, compared to traditional synthesis methods, RAVE informs a different thinking about controlling and manipulating sound, one in which sound parameters and latent dimensions are deeply entangled. In previous works, we explored three types of entanglement, showing how these influence the way we play, and proposed a series of tailored performative and compositional approaches, including the development of dedicated interfaces [22]. Even though this paper does not focus on the interfaces themselves, it is worth noticing that in Mouja+ RAVE's latents are manipulated through Stacco, one of the interfaces emerging from our work at the Intelligent Instruments Lab (IIL).

3.1.2 Stable Audio

Stable Audio Open [15], the second NAS model used in Mouja+, is a text-to-audio diffusion model trained on a dataset collected from the Freesound archive [23]. Diffusion models generate sound files by progressively reversing a noise-adding process used for the training, and are often conditioned on text vectors generated by LLMs. The user interacts with such systems by entering positive and negative prompts using natural language, and by defining parameters such as temperature and steps, corresponding to the randomness and the number of denoising iterations in the generation process.

Crucially, whereas real-time autoencoders such as RAVE continuously output small blocks of sounds (typically 2048 samples per block), thus allowing one to dynamically intervene in the timbre transfer process through the manipulation of the latent space, text-to-audio diffusion models asynchronously generate one audio file per prompt. This asynchronous process limits real-time NAS manipulation



Figure 2. GUI for text prompting and timbre transfer.

but enables macro-formal compositional control through natural language instructions.

3.1.3 RAVE and Stable Audio

Stable Audio and RAVE serve two different and yet potentially complementary purposes. On the one hand, Stable Audio invites the composer to work on structure and form through textual instructions, but offers no real-time control over the timbral qualities of the sound; on the other hand, RAVE's small block size makes it the ideal choice for dynamic sound manipulation but offers no control over meso and macro-formal aspects.

The first contribution of Mouja+ is a pipeline complementing these two methods with each other. By prompting textual instructions through carefully designed text scores, and by layering the generated files during playback in the guise of prompt jockeying, the composer gives shape to the piece. At the same time, through optional timbre transfer and latent manipulation with RAVE, one can intervene into the most minute timbral parameters as the piece unfolds. To easily combine Stable Audio and RAVE in live scenarios we developed a Graphical User Interface (GUI), through which one prompts textual instructions, plays back and mixes the generated cues, and optionally routes them to a series of RAVE models for real-time timbre transfer.

3.1.4 Graphical User Interface

Mouja+'s GUI (Fig.2) forwards to and download prompt from a remote server through a dedicated OSC application [3]. The added overhead of synthesizing the sounds on a remote server is justified by the need to reduce the time lag between prompting and the generation of the file, which is obviously a critical aspect in a live performance. By processing the prompt on a server with a high-performing GPU, we could sensibly reduce the time required for the generation of the sound files (around 20 seconds for a 47-second sample) compared with most consumer machines, even considering the time required to download the file into the client.

On the client GUI, one writes positive and negative text prompts, and defines length in seconds, number of steps, temperature, and buffer for the download. Once ready, the file generated by Stable Audio automatically uploads into the specified buffer on the client computer. The GUI allows one to view, play, process and mix up to eight audio files at the same time, that can be routed either directly to the Ambisonics encoder (and from there to the speakers), or (either entirely or partially) to RAVE for timbre transfer. When timbre transfer is applied, the performer can influence the resynthesis by manipulating the latents through a dedicated interface such as Stacco.

The GUI also allows one to actively control the position of up to four sound sources (one per buffer, for a total of four buffers) in the Ambisonics space, with the spherical coordinates of each source mapped to the position of one magnetic attractor in Stacco. When the sound sources are not actively controlled by the user, an algorithm handles their displacement along the circular perimeter of the dome, with different speeds and directions.

In experimenting with RAVE, we first trained a model with the same dataset (from the Freesound archive) used by Stable Audio Open. The results were not satisfactory, both in terms of sound quality and latent control, probably due to the high timbral variability and large size of the dataset, which seems to be at odds with RAVE's focus on small and timbrally consistent datasets. We thus selected a series of different models curated and trained by various members of the Intelligent Instruments Lab (IIL), including foley sounds, liquids, guitars, voices, percussions, magnets, organs, and birds. [4] This approach proved more effective, both in terms of extending the system's timbral possibilities, and of providing a fine control over the resynthesis through latent manipulation.

3.2 Performance

Mouja+ features ten text scores adapted from the Fluxus Performance Workbook [19], whose selection and adaptation I view as a practice-based investigation into how language use shifts when addressing AI systems. During the SWRL performance, scores were displayed on a large screen alongside real-time information indicating whether each prompt's synthesis was in process or complete, a deliberate choice that allowed me to play with the audience's cognitive dissonances between the textual instructions and the model's outputs.

The audio files were layered and structured into a tripartite form: two sections of three scores each, and one section of four scores. Score selection prioritized timbral and dynamic considerations, as well as textual relatedness, with the different text scores establishing semantic relationships to form a surreal and theatrical meta-narrative. The piece describes the entrance on stage of a piano, that gets interacted with in different ways (except playing it in a traditional way, as typical in Fluxus), then dismantled and sold along with the whole theatre.

The use of spatialisation techniques greatly influenced the selection of the Fluxus scores toward prompts that would evoke on stage the presence of large objects and acting bodies, whose sonic traces would vividly move around the audience. Audience members described the performance

[3] https://github.com/elgiano/gradio-osc

[4] https://huggingface.co/Intelligent-Instruments-Lab/rave-models

as evocative, due to the cognitive dissonance between the realism of the sounds, materialising the presences evoked by the text scores, and their absence on stage. Through the real-time displacement of the sources afforded by the pipeline, I could intensify this effect at will throughout the piece.

Section One opened with concrete sounds moving concentrically around the audience, and the noise of a heavy resonating body (a piano, as revealed by the score) being dragged into the stage. The text scores were three: 'Let piano movers carry the piano on stage,' 'Wash the piano, wax it and polish it well' (adapted from Nam June Paik), and 'Eat a juicy apple' (adapted from Bob Lens), the sequence suggesting the presence of one or more figures on stage, struggling to move a piano, washing it and finally enjoying a snack. As the files were generated and overlapped, I routed a part of the playback to RAVE, and controlled the resynthesis with Stacco, using a foley and water model to smooth out transitions and emphasize dynamics.

Section Two shifted from concrete sounds to pointillistic and evocative ones. The scores were 'Hammer nails into the piano keys' (adapted from Tomas Schmit), 'Dropping coins' (excerpt from George Maciunas Solo for Rich Man) and 'The sound of the stone ageing,' (adapted from Yoko Ono). As in the first section, I resynthesised the sounds with Stacco. Here, a crucial challenge was managing the model's tendency toward literalism (albeit an idiosyncratic one, as I discuss in section 4). While section one materialised objects and bodies on stage, this section's transition toward semantic ambiguity (e.g. the sound of a stone ageing) foregrounded the model's tendency to collapse uncertainty into overly simplistic interpretations. The section concluded with a solo passage using direct latent navigation on Stacco without resynthesis, momentarily stepping outside the text-to-audio paradigm to foreground the performer's embodied control over the latent space itself.

Section Three exploited the model's realism more deliberately, by conjuring on stage an open market, with unintelligible voices merging into a crowd that I dynamically displaced in the Ambisonics dome with Stacco. This reversed the circumscribed and closed space with that of an imaginary outside. Finally, the cacophony of voices, coins, pianos, deflated again into the crawling of a piano (the same piano that entered on stage at the opening) being heavily dragged out of the stage by imaginary actors. The text scores I used in this section were 'Performer sells the theatre' (adapted from Ben Vautier), 'Piano movers carry piano out of stage' (adapted from Nam June Paik), and, as a closure, Ono's 'The sound of the stone ageing.'

4. DISCUSSION

Through the first author's artistic practice on Mouja+, we gained multiple insights relevant to design practice with NAS, the use of NAS in combination with spatial audio, and best practices for designing text scores for text-to-audio generative models. In the discussion, we focus on the latter.

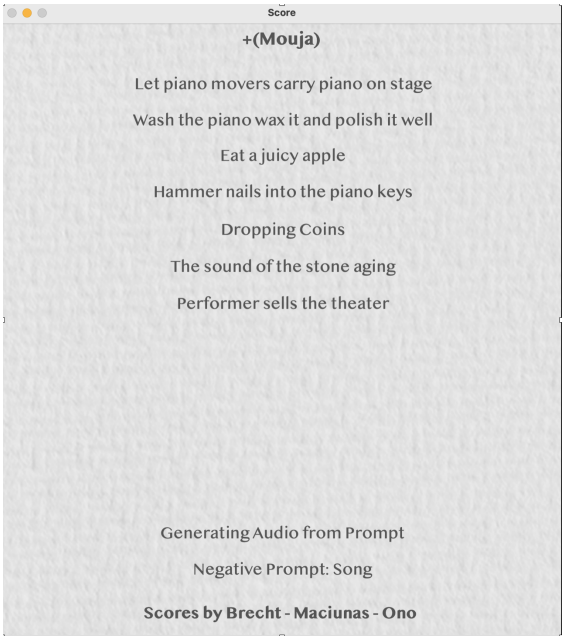


Figure 3. Mouja+ score combining adaptations of Fluxus text scores, as seen by the audience.



Figure 4. Performing Mouja+ at SWRL.

4.1 Language Games

In working on Mouja+, one aspect became inevitably apparent: when composing text scores for text-to-audio models, our use of language must change and adapt. We can examine this through Wittgenstein's notion of language games, which demonstrates how meaning in language is not fixed but emerges through contextual use [24].

Yoko Ono's Stone Piece offers a fitting example, as it plays with a radical ambiguity, by inviting the performer to 'take the sound of a stone ageing.' This is a contradictory statement on many levels. What is the sound of ageing? How can a stone age? And as a consequence, what does ageing mean? Like a haiku, this score introduces interpretative ambiguities and nuances that call for introspection. Yet, to an existential question requiring deep reflection and awareness of context of use, in other words of the language game being played, the model tends to default to a simplistic and literal interpretation: the sound of a stone being scratched.

From this reflection, it is clear how there exists a cer-

tain art to being literal with a text-to-audio model. This involves an adoption and use of natural language that accounts for the constraints determined by the model (perhaps better, by the encounter of a certain algorithm with a certain training data), that predates the performative contextualization. Tentatively discovering and adapting to the language game Stable Audio knows and plays best, and leveraging it in composing Mouja+ was perhaps the most challenging (but rewarding) process in working on this piece.

4.2 Compositional Strategies

To understand why the strategies described below were adopted, it is necessary to first clarify the artistic intent. To do this, we may draw from Umberto Eco's theory on openness and indeterminacy in contemporary artistic production. In *Opera Aperta*, Eco demonstrates how contemporary artists create open works by incorporating different forms of entropy, allowing for the contextual emergence of multiple meanings [14]. This implies walking a fine line between control and deliberate unpredictability, which is necessary to the emergence of poetic meaning.

Similarly to the examples discussed in 2.1, in Mouja+ Stable Audio's interpretative idiosyncrasies and statistical variability are viewed as yet another source of entropy, and the efforts in learning how to best design the prompts (the syntactic stratagems, but also the pipeline) represent yet another attempt to negotiate emergence and control.

In other words, the aim in Mouja+ was not one of turning a probabilistic system into a deterministic one, to know with a high degree of precision what sound will be generated given a certain prompt. Rather, it was to learn how to constrain the model's indeterminacy within a certain space of possibilities, to become familiar with how the system interprets language and anticipate, with reasonable confidence, how prompts might sound once rendered. The strategies presented in the next section were developed in order to achieve this balance. These may termed (i) *syntactic fine-tuning*, (ii) *mindful use of negative prompting*, and (iii) iterative *exploration of the training data*.

4.2.1 Syntactic Fine-tuning

I began working on Mouja+ by testing a large number of Fluxus scores. Once I found a score from which the model generated an artistically interesting output, I further experimented with it, first by working on the semantics of the score and then by changing the syntax. If semantic changes usually produced very different sonic worlds, by slightly changing the syntax (for instance by inverting the order two words or by adding or removing a preposition) I could finely stir the generation process towards a desired outcome. Metaphorically speaking, semantics changes felt like preset changes on a synth, and syntactic ones as fine-tuning a given preset. For instance, 'Let piano movers carry the piano out of the stage,' 'piano movers carry piano out of stage,' or 'eat juicy apple(s) during concert' and 'eat a juicy apple during concert' tend to be interpreted in similar ways by humans. However, by adding 'a' and removing '(s)' from the text score, I increased the consistency of the model, intended as the likelihood of generating similar results through repeated prompts using the same text.

4.2.2 Dataset Exploration

Stable Audio Open is trained on the Freesound archive, which is open and freely accessible online through a convenient search engine. This feature was highly relevant in shaping my prompts. I used Freesound's advanced search engine to explore text embeddings, typing keywords and selecting tags based on the scores I was experimenting with. One example is the word 'exit,' as per Brecht's score, which I experimented with extensively. As I realised by checking the keyword with Freesound's search engine, in the database 'exit' is often embedded in environmental recordings involving trains arriving at the station, opening and closing their doors, and leaving. This suggests that the statistical likelihood that prompts containing this word might incorporate sounds of moving trains is high, as was confirmed through successive tests with the model.

4.2.3 Negative Prompting

Once I understood how words were embedded in the original data, I typically decided whether to discard a score or use negative prompting to stir the system towards an expected outcome. In most cases, negative prompting was the most effective strategy. For instance, useful negative prompts for the word 'exit' were 'train' and 'bus,' 'station,' or 'soundscape,' which allowed avoiding sounds of public transportation coming and going. With negative prompts, I could effectively define niches of sounds by exclusion, counterbalancing some of the problems arising from the model's lack of context awareness. This was a critical aspect of Mouja+: in most of the scores I used negative prompts such as 'music,' 'piano music' or 'song' to avoid the generation of structured musical pieces of any genre, that would have hardly integrated with the rest of the performance.

4.3 Open Access

Of the three methods, dataset exploration deserves here further consideration. Checking the relationship between individual keywords (or clusters of words) and the dataset through Freesound's search engine was a particularly helpful strategy in adapting Fluxus scores to Stable Audio, one I systematically and iteratively applied on every single score. Crucially, the applicability of this method relies on the Freesound archive's openness and on the availability of an advanced search engine, two features that represent the exception rather than the norm in text-to-audio models and more broadly generative AI.

The academic literature pointing to the value of open access to datasets is rich. For instance, scholars advocate for FAIR (Findable, Accessible, Interoperable, Reusable) approaches to data collection and deployment as a way of mitigating bias [25], discuss the ethical relevance of data transparency [26], and criticise the concerning systemic lack of data documentation among AI companies [27]. In response to these pressing issues, researchers have implemented tools to trace lineages and audit text datasets [28]

and multimodal content, that allow querying a certain corpus, verifying whether and how data is licensed, or assessing diversity, inclusion, and biases.

The artistic research presented in this paper tackles this issue from a slightly different angle, showing how, instead of relying on tentative prompting approaches, we can more effectively refine our natural language instructions through the direct engagement with the training data. Of course, provided that the latter is accessible and can be effectively searched. This hints at how openness and accessibility are essential not solely in addressing critical issues around bias, data ownership and accountability, but also to facilitate and improve our engagement with generative AI within and beyond the arts.

5. CONCLUSION

In this paper, we investigated how text-to-audio models can serve as compositional and performative tools for experimenting with text scores. Our primary technical contribution demonstrates that by combining an asynchronous text-to-audio model (Stable Audio Open) with a real-time timbre transfer one (RAVE), performers gain control over macro-formal aspects and timbral navigation through NAS techniques. This hybrid approach extends text-to-audio generation beyond the limitations of prompt jockeying.

In Mouja+, we used Fluxus scores as probes to investigate our use of natural language when addressing AI systems rather than human performers. Drawing on Wittgenstein's language games, we found that effective communication with text-to-audio models requires adapting our strategies in designing the scores. Three practices proved particularly effective: syntactic fine-tuning of prompts, strategic use of negative prompting, and iterative exploration of the training dataset.

The last point in particular shows that dataset transparency and accessibility is crucial to modulate control and emergence, and thus to develop effective compositional workflows with generative systems. While current works emphasize the importance of openness and accessibility of datasets to address bias and ownership issues and ensure accountability, our work demonstrates that accessible training data also enables a more intentional and nuanced creative engagement with generative AI systems.

It is possible, and perhaps even likely, that the compositional and performative approach to AI co-creation via natural language we described will diffuse as NAS technologies increase in efficiency. But as our engagement with generative systems grows and evolves, affecting our lives on stage and outside of it, we should be mindful that, with AI, there is an art to being literal.

6. ETHICAL STANDARDS

All the RAVE models used in this work have been trained from open-source material or with the explicit consent of all parts involved, and are free to download and use. Stable Audio Open is trained on Creative Commons-licensed audio [15].

Acknowledgments

This research is supported by the European Research Council (ERC) as part of the Intelligent Instruments project (INTENT), under the European Union's Horizon 2020 research and innovation programme (Grant agreement No.101001848). We also thank SWRL space and Voldemārs Johansons for organising and hosting the performance.

7. REFERENCES

[1] V. Shepardson, J. Armitage, and T. Magnusson, "Notochord: a flexible probabilistic model for embodied midi performance," 2022. [Online]. Available: https://zenodo.org/record/7088404

[2] N. Privato, O. Rampado, and A. Novello, "A creative tool for the musician combining lstm and markov chains in max/msp," in *Artificial Intelligence in Music, Sound, Art and Design: 11th International Conference, EvoMUSART 2022, Held as Part of EvoStar 2022, Madrid, Spain, April 20–22, 2022, Proceedings*. Berlin, Heidelberg: Springer-Verlag, 2022, p. 228–242. [Online]. Available: https://doi.org/10.1007/978-3-031-03789-4_15

[3] S. Yuditskaya, S. Sun, and M. Schedel, "Synthetic erudition assist lattice," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, Shanghai, China, June 2021. [Online]. Available: https://nime.pubpub.org/pub/5oupvoun

[4] V. Shepardson and T. Magnusson, "The living looper: Rethinking the musical loop as a machine action-perception loop," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, M. Ortiz and A. Marquez-Borbon, Eds., Mexico City, Mexico, May 2023, pp. 224–231. [Online]. Available: http://nime.org/proceedings/2023/nime2023_32.pdf

[5] N. Privato, T. Magnusson, and E. T. Einarsson, "Magnetic interactions as a somatosensory interface," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, M. Ortiz and A. Marquez-Borbon, Eds., Mexico City, Mexico, May 2023, pp. 387–393. [Online]. Available: http://nime.org/proceedings/2023/nime2023_54.pdf

[6] J. Walshe, "The text score dataset 1.0," https://megansteinberg.com/wp-content/uploads/2022/01/a6702-thetextscoredataset1.0-jenniferwalshe.pdf, 2021, project documentation.

[7] S. L. Moan, "Machine listening: Machine learning and audio analysis," https://www.youtube.com/watch?v=_fpnAHoRSqU, Aug. 2020, youTube video.

[8] L. Russolo, *The Art of Noises*, 1986.

[9] J. D. Erickson, *Dada: Performance, Poetry, and Art*, ser. Twayne's World Authors Series. Boston: Twayne Publishers, 1984, vol. TWAS 632.

[10] K. Friedman, Ed., *The Fluxus Reader*. Chichester, West Sussex; New York: Academy Editions, 1998, free digital edition available from Swinburne University of Technology. [Online]. Available: http://hdl.handle.net/1959.3/42234

[11] P. Oliveros, *Anthology of Text Scores*. Kingston, NY: Deep Listening Publications, 2013, edited by Sam Golter and Lawton Hall.

[12] T. Funk, "A Musical Suite Composed by an Electronic Brain Reexamining the Illiac Suite and the Legacy of Lejaren A. Hiller Jr." 9 2018. [Online]. Available: https://indigo.uic.edu/articles/journal_contribution/A_Musical_Suite_Composed_by_an_Electronic_Brain_Reexamining_the_Illiac_Suite_and_the_Legacy_of_Lejaren_A_Hiller_Jr_/10781933

[13] J. Walshe and D. D. Roure, "Ai text scores," https://www.rncm.ac.uk/research/research-activity/research-centres-rncm/prism/prism-blog/ai-text-scores/, July 2021, pRiSM Blog, Royal Northern College of Music.

[14] U. Eco, *The Open Work*. Cambridge, MA: Harvard University Press, 1989, original work published as *Opera Aperta*, 1962.

[15] Z. Evans, J. D. Parker, C. Carr, Z. Zukowski, J. Taylor, and J. Pons, "Stable audio open," 2024. [Online]. Available: https://arxiv.org/abs/2407.14358

[16] Intelligent Instruments Lab, "Workshop: Exploring ai musicianship with dadabots," https://iil.is/news/dadabots-workshop, 6 2024, workshop held at Veröld – hús Vigdísar, University of Iceland, Reykjavik. [Online]. Available: https://iil.is/news/dadabots-workshop

[17] N. Privato, "Mouja: Experiencing ai through magnetic interactions," in *Proceedings of the Eighteenth International Conference on Tangible, Embedded, and Embodied Interaction*, ser. TEI '24. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: https://doi.org/10.1145/3623509.3635328

[18] N. Privato, T. Magnusson, and E. T. Einarsson, "The magnetic score: Somatosensory inscriptions and relational design in the instrument-score," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'2023*, A. P. D. Ritis, V. Zappi, J. V. Buskirk, and J. Mallia, Eds. Boston, Massachusetts, USA: Northeastern University, 2023, pp. 36 – 44.

[19] K. Friedman, O. F. Smith, and L. Sawchyn, *The Fluxus Performance Workbook*. Swinburne, 2002. [Online]. Available: https://doi.org/10.25916/sut.26270992.v1

[20] A. Caillon and P. Esling, "Rave: A variational autoencoder for fast and high-quality neural audio synthesis," 2021. [Online]. Available: https://arxiv.org/abs/2111.05011

[21] N. Privato, V. Shepardson, G. Lepri, and T. Magnusson, "Stacco: Exploring the embodied perception of latent representations in neural synthesis," pp. 424–431, September 2024. [Online]. Available: http://nime.org/proceedings/2024/nime2024_62.pdf

[22] N. Privato, G. Lepri, T. Magnusson, and E. T. Einarsson, "Sketching magnetic interactions for neural synthesis," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'24*, P. Kocher, Ed. Zurich, Switzerland: Zurich University of the Arts, 2024, pp. 89–97.

[23] D. Stowell and M. D. Plumbley, "An open dataset for research on audio field recording archives: freefield1010," October 2013, arXiv preprint. [Online]. Available: https://arxiv.org/abs/1309.5275

[24] L. Wittgenstein, *Philosophical Investigations*, G. E. M. Anscombe, Ed. New York, NY, USA: Wiley-Blackwell, 1953.

[25] S. Raza, S. Ghuge, C. Ding, E. Dolatabadi, and D. Pandya, "Fair enough: How can we develop and assess a fair-compliant dataset for large language models' training?" 2024. [Online]. Available: https://arxiv.org/abs/2401.11033

[26] J. Hardinges, E. Simperl, and N. Shadbolt, "We Must Fix the Lack of Transparency Around the Data Used to Train Foundation Models," *Harvard Data Science Review*, no. Special Issue 5, may 31 2024, https://hdsr.mitpress.mit.edu/pub/xau9dza3.

[27] S. Longpre, R. Mahari, A. Chen *et al.*, "A large-scale audit of dataset licensing and attribution in ai," *Nature Machine Intelligence*, vol. 6, pp. 975–987, 2024. [Online]. Available: https://doi.org/10.1038/s42256-024-00878-8

[28] A. Piktus, C. Akiki, P. Villegas, H. Laurençon, G. Dupont, A. S. Luccioni, Y. Jernite, and A. Rogers, "The roots search tool: Data transparency for llms," 2023. [Online]. Available: https://arxiv.org/abs/2302.14035

LIVELILY: THE EVOLUTION OF A LIVE SCORING / LIVE SEQUENCING ENVIRONMENT

Alexandros Drymonitis
Independent Scholar
alexdrymonitis@gmail.com

ABSTRACT

In this paper I focus on the development of the *LiveLily* Live Scoring and Live Coding system, since its initial release on 2023. This system has been developed for writing Western-music scores live. Its further development includes user defined functions, enhanced OSC communication, syntax highlighting and other features. Inspired by the *Lilypond* language, this system offers an intuitive textual language, simpler than *Lilypond*, for writing and playing music scores live. It provides the ability to use Python and the *Pyo* module to write DSP code to play scores or to control other software or hardware via OSC or MIDI communication, and communicating the score with human performers through an accompanying software that can display part of the full score. Being focused on Live Scoring, it is very responsive and takes minimal time to render a score, while it can play rendered scores in real-time.

1. INTRODUCTION

Live Scoring is an artistic practice of creating music scores in real-time. It is a field that has seen development through research and artistic practice since more than ten years, at the time of writing. Projects like *INScore* [1], *Maxscore* [2], the *bach* objects for *Max* [3] and *OpenMusic* [4] date back to 2012, 2013 and 2014, but most of them are still actively maintained. Live Coding too, is a field with active research. In this practice, an artist uses computer code to create audio or visuals and writes or edits this code live during a performance. Live Coding is characterized by its community of scholars and artists and an abundance in literature —which is demonstrated by its own conference [1] —, but also by a multitude of approaches, including combinations of live coders with other instruments, choreographers, performance artists and others [5].

LiveLily is a system for Live Scoring and Live Sequencing through Live Coding, based on a bespoke Domain Specific Language (DSL) [6]. Even though Live Coding can be characterized by curiosity, where live coders experiment with various inter-disciplinary projects, there is little to no research on the combination of Live Coding with Live Scoring for creating traditional Western-music scores. In this context, we should mention that live coders tend to see the code as a music score [7], leaving a gap between Live Coding and the traditional Western-notation music score. *Maxscore*, the *bach* objects for *Max* and *OpenMusic* can be considered as Live Scoring systems that can be used in a Live Coding context, but these projects follow the visual programming paradigm, as *Maxscore* and the *bach* objects are developed for *Max*, while *OpenMusic* is a visual programming system of its own.

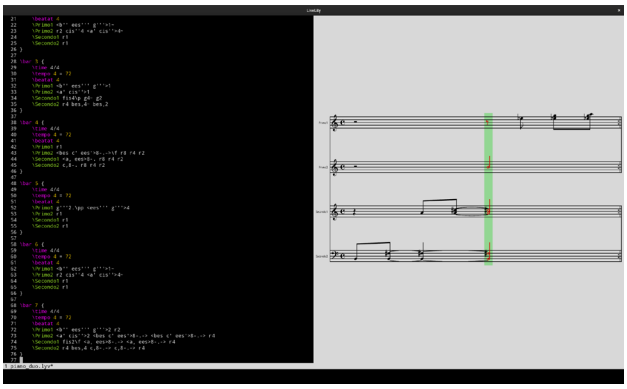


Figure 1. A *LiveLily* session with a composition by Yiannis Sfyris.

The *LiveLily* system utilizes textual code, inspired by the *Lilypond* language [8], therefore it fills the gap between textual Live Coding and Live Scoring in a traditional Western-music context. Software like *Frescobaldi* [2] might seem similar to *LiveLily*, in the context of writing code in the *Lilypond* language and rendering the score, but rendering takes a considerable amount of time in a Live Scoring context. *OpusModus* [3] utilizes textual code to render traditional Western-music scores too. Like *LiveLily*, this software features Live Coding too, but this is mainly used as a mechanism to provide feedback to composers on what they write while writing a score and it does not aim at Live Scoring.

LiveLily is an open-source [4] true Live Scoring system, as it renders the scores in real-time. It also includes a sequencer that sends either OSC or MIDI messages as a score plays. These messages can be received in other software or

[1] International Conference on Live Coding (ICLC) https://iclc.toplap.org/

[2] https://frescobaldi.org/

[3] https://www.opusmodus.com/

[4] https://github.com/alexdrymonitis/LiveLily

Copyright: ©2025 Alexandros Drymonitis et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.



Figure 2. Live Coding an acoustic ensemble.

hardware, so a score can be played with any desired sound. Figure 1 illustrates a *LiveLily* session with part of a composition by Yiannis Sfyris, who collaborated with me during my Postdoc research [9].

In addition to the main system, an accompanying program that displays part of the score can be used with performers to either combine electronics with acoustic instruments, or to live code acoustic ensembles. Figure 2 illustrates one such Live Coding session, where the author uses *LiveLily* to live code a score which an acoustic ensemble sight-reads and follows to play techno. Prior research I have conducted has combined performers with hardware [9], with a Yamaha Disklavier robotic piano being controlled by *LiveLily* and at the same time a performer playing on the same instrument.

This paper is structured as follows: The next section analyzes the architecture of the *LiveLily* system. The section after that describes the basic usage of the *LiveLily* language. Then a section on the new features that are the main focus of this paper is laid out. A section on future work follows, with the conclusions closing the paper.

2. SYSTEM ARCHITECTURE

The *LiveLily* system is written in C++ with the openFrameworks toolkit [5] (OF). This toolkit was chosen for its flexibility, its support from its user and developer communities, and its simplicity concerning commands for drawing visuals. The entire system consists of four distinct parts. The first part is the editor where the user types code in the *LiveLily* language as shown in the left side of the screen capture in Figure 1. Even though the *LiveLily* system is not solely a code editor, and only part of the focus of this system's development has been dedicated to developing the editor, it includes all the necessary functionalities a user would expect from such an editor, including line numbering, syntax highlighting, auto-insertion of closing brackets, copying and pasting to/from the clipboard, and others. The second part of the system is the interpreter that translates the code to executable commands. This is invoked when the user asks the system to execute a line, or a larger chunk of code. Executable code being called explicitly by the user gives the flexibility of being able to type anything into the editor but execute only part of it. This means that the users are able to code their scores at their desired pace and execute their code bit by bit, leading to a fine control of how the score is played. The third part is the score that displays what is written in the *LiveLily* language in traditional Western-music notation, as shown on the right side of Figure 1. The last part is the sequencer that plays the score and communicates it with OSC or MIDI.

[5] `https://openframeworks.cc/`

2.1 The Score

The score of the *LiveLily* system allows the visual representation of the coded musical indications. It is written using the Sonata font and primitive shapes provided by OF. The Sonata font is used for symbols like note heads, rests, clefs, meters, articulation symbols, accidentals, dynamics, and octave symbols. Shapes like stems, beams, glissandi, slurs, and the actual staves are drawn using primitive shapes like lines and curves.

A previous Live Scoring approach with the *Data Mining / Live Scoring* project [10] included the *Lilypond* engraver and OF, where the former rendered scores in the PNG format, and the latter displayed them and highlighted the currently playing bar with a blinking cursor at the beginning of it. This approach though was not responsive enough in a Live Scoring context. This is because *Lilypond* takes a few seconds to render a single bar of music. Additionally, extra work was needed to spot the beginning of each bar, as the score was rendered as a PNG file and not displayed with OF's drawing classes. This required computer vision techniques provided by the ofxOpenCV addon for OF, which are time consuming in a Live Scoring context.

The *LiveLily* approach is much more responsive as it takes very little time to render a bar. While evaluating the system's responsiveness, rendering bars with four staves took less than one millisecond. The bar in Figure 3 —composed by Yiannis Sfyris— was rendered twenty-seven times to compare the various rendered durations between each iteration, with the shortest duration being 377 microseconds and the longest 799 microseconds. Bars that have already been rendered are stored in *LiveLily*'s memory and can be called anytime during a session without any delay at all, since no additional rendering takes place. Also, spotting the position of any element of the score does not require any perceived computational time, as all elements are OF objects. This means that the program is instantly aware of where each item is drawn and no computer vision techniques are required.

This approach provides much more flexibility as to which item of the score can be highlighted and in what way. In Figure 1 we can see a green cursor at the third beat of the bar displayed over the score. This is a blinking cursor that appears at the currently playing beat, to display the sequencer. This is useful for the audience to follow the score, but more importantly for a performer that plays along to be synchronized. In this Figure, we can also see the currently active notes and rests displayed in red color. All this highlighting is possible because all the score elements are drawn as OF objects, which gives full control over their appearance.
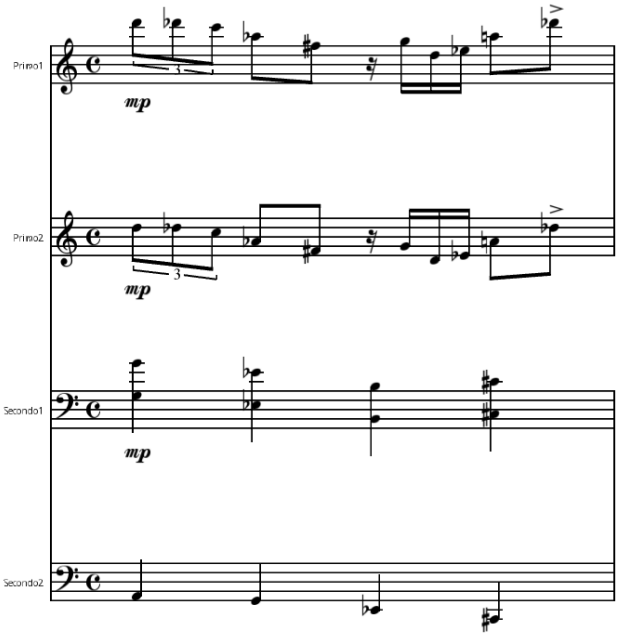


Figure 3. A four staff bar composed by Yiannis Sfyris, rendered to time the rendering duration.

2.2 The Sequencer

Three out of four parts of this system run in the same thread, with the sequencer having its own thread. This due to the fact that OF, being primarily a graphics-oriented toolkit, operates at a fixed framerate that works well for visuals, but not necessarily for processes that need a faster rate. Even at high framerates like 60, or even 120 frames per second, running a musical sequencer at high tempi would cause jitter, due to the time grains necessary to synchronize fast notes (e.g. sixteen or higher) being shorter than what a high framerate provides.

OF provides threading that runs at different clock speeds than the framerate of a program written in it. Utilizing this feature, *LiveLily*'s sequencer runs at a clock speed of 100 microseconds. This clock speed provides the necessary resolution to play scores at high tempi with fast notes without jitter.

3. BASIC USAGE

In this section we include some examples of the *LiveLily* language, to give an impression to the reader of how the *LiveLily* system works. The main commands to create a simple score are *insts* and *bar*. The first one initializes the instruments of the score. Once the instruments are created, they cannot be changed, so the user must created all instruments at the beginning of each session. This command takes the names of the instruments as arguments. The line below will create a string quartet with the names *violin1*, *violin2*, *viola*, and *cello*.

```
\insts.init violin1 violin2 viola cello
```

Once created, if the score is already visible, four staves will be created, one for each instrument. The score can become visible with the *score.show* command.

Once the instruments have been created, the user can start creating bars of music. The command to create a single bar is *bar*. This command takes a name for the bar to be created, and the contents of the bar inside curly brackets. The lines for each instrument inside a bar definition must be separated, as shown in the code example below.

```
\bar 1 {
  \violin1 c'''2 b''4( c''')
  \violin2 g''4 fis'' f'' e''
  \viola e'2 d'8( des') c'4
  \cello \clef bass c2 g2
}
```

The name of the bar in this example is *1*, but it could be anything, be it a number or a string. Once a bar has been created, it can be called by its name with a backslash prepended to it. In the example above, if the user wants to display this bar, they should type *1*. Note though that when a bar is created, if the sequencer is not running, the bar will immediately be displayed. Calling a bar by its name to display it is necessary if the sequencer is running and another bar is being currently displayed.

If the user creates more than one bars, these can be combined in a loop with the *loop* command. This command is very simple. The line of code below demonstrates its use.

```
\loop 1-2 {\1 \2}
```

The example above supposes that the user has defined at least two bars whose names are *1* and *2*. This line will create a loop with these two bars which is named *1-2*. As with bars, loop names take a backslash at their beginning, once created. An addition to the *loop* command is the asterisk which is used to repeat a bar in the loop defined for a certain number of times. The line of code above can be changed to the line below, in which case, the loop will repeat bar *1* twice before playing bar *2*.

```
\loop 1-2 {\1*2 \2}
```

Being inspired by the *Lilypond* language, the *LiveLily* language supports elements such as articulations, slurs, ties, glissandi, and dynamics. All these are handled by the sequencer of the system, without the user needing to handle them in the DSP environment they work in. Glissandi and dynamics are communicated via OSC or MIDI. When used with OSC, the frequency and amplitude changes are sent to their respective address, */note* and */dynamics*, respectively. In the case of MIDI, glissandi are handled with Pitch Bend messages, while dynamics are handled with the overall volume control. Slurs and ties are handled either by not sending a zero amplitude when used with OSC, or by sending the Note Off message after the next Note On, when used with MIDI.

Articulations include staccato, staccatissimo, accent, marcato, trill, tenuto and portando. The staccato, staccatissimo, and tenuto articulations are handled by the sequencer, by shrinking or stretching the Note On durations. The percentage of these shrinks and stretches to the overall Note On duration can be set by the user. Their default values are 50%, 25%, and 90% of the Note On duration for staccato, staccatissimo, and tenuto, respectively. The rest of

the articulations are aimed at instrument performers that play following the accompanying software mentioned in the *Introduction* of this paper, which displays part of the full score.

4. NEW FEATURES

Since its initial announcement in 2023 [6], the *LiveLily* system has seen a lot of development. New features have been added to the editor, the interpreter, the handling of errors, the language design of the system, the capability to read MusicXML files, the capability to connect to external serial devices, its OSC communication features, and the integration of the Python programming language. In this section we will analyze these features in detail.

4.1 Additions to the Editor

The additions in the editor of the *LiveLily* system include syntax highlighting, editor modes, and the ability to split the editor window in panes. In this section we will analyze these separately.

4.1.1 Syntax Highlighting

One of the feature that were included in the *Future Work* of the initial paper for this system was syntax highlighting [6]. As seen in Figure 1, commands like *\bar*, *\time*, and *\tempo* are displayed with a different color than instruments like *\Primo1*, *\Primo2*, etc. The colors are used to separate the following features of the *LiveLily* language: first level commands, second level commands, instrument names, bar names, loop names, functions, comments and digits.

4.1.2 Editor Modes

Another addition to the editor is four different modes, following the paradigm of the Vim editor. Vim has four modes of operation, Normal, Insert, Visual, and Command mode. *LiveLily* imitates parts of each of these modes. In Insert mode, like in Vim, the user can type text into the editor as they would do with any simple text editor. Contrary to Vim, whose default mode is the Normal mode, in *LiveLily* the Insert mode is the default one. In Normal mode, the user cannot type text, but can navigate through it with the H, J, K, and L keys, like in Vim. This feature is useful when touch typing, so the user does not need to change the position of their hands, thus can maintain eye contact with the screen, uninterrupted.

In Visual mode, the user can copy text to the program's memory —not the clipboard—, and can paste that text in the editor. These text copies are called *yanks* in Vim's terminology. Copying to and pasting from the clipboard is possible in Insert mode with the combination of CTRL+C for copy and CTRL+V for paste. In Command mode, the user can type certain commands, either to load or save a text file, navigate through the computer system —aiming at locating files to load—, and quit the program. Inspired by NeoVim, when in Insert mode, the cursor is shown as a line. When in Normal or Visual mode, the cursor is shown as a rectangle. When in Command mode, the cursor moves

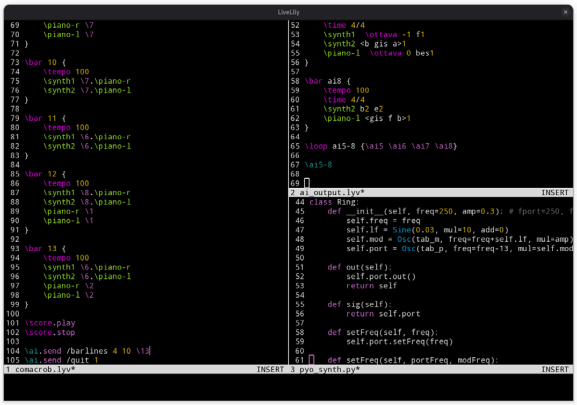


Figure 4. Splitting the editor into panes.

to the message area of the *LiveLily* window (see *4.2.1 Error, Warning, and Note Messages* subsection below).

4.1.3 Panes

The last addition to the editor is splitting its window into panes. This can be done either horizontally, or vertically, or in a combination of the two. Figure 4 illustrates the editor split into three different panes, with two columns, one row for the first column and two rows for the second. In this Figure we can see that the panes are being numbered from left to right and top to bottom. The numbering is shown in the thick bar below the pane and its order is regardless of the order in which the panes are created. The active pane can be spotted by the cursor shown as a line —because it is in Insert mode—, where the inactive panes have their cursor as an outlined rectangle, regardless of the mode these panes are in. Switching between panes is done with the Alt key and the number key of the pane the user wants to activate. It should be noted that the commands executed in any pane are directed to the same interpreter. Regardless of which pane is active, any command executed will affect the same score, unless a pane is set to execute Python code, in which case the Python interpreter is invoked. All Python panes share the same interpreter too, like all *LiveLily* panes do.

4.2 Additions to the Interpreter

Additions to the interpreter include new commands that extend the *LiveLily* language's command set, improvements to error, warning, and informational messages, as well as changes to the way the interpreter executes commands passed as arguments to other commands. For the sake of clarity in writing this paper, the new commands that have been added to the language are analyzed in their own subsection and not in this subsection.

4.2.1 Error, Warning, and Note Messages

The first version of the *LiveLily* system included error messages that were triggered when a command did not exist, or where there was a syntactical error in it. The current



Figure 5. An error message and its highlighted line.

version of the system includes warning and note messages, like C or C++ compilers.

When an error occurs, the command is not executed, and the error message is printed in the message area of the *LiveLily*. This is the bottom area shown in Figures 1, 4 and 5. The error message is displayed in red, containing the line number in which it occurred, and the actual line in the editor turns red too, as shown in Figure 5.

Warning and note message let the command be executed, but are also printed in the message area. The warning messages are displayed in orange color, while the note messages are displayed in white. The warning messages highlight the respective line, like the error messages do, but in orange color.

4.2.2 Nested Commands

In the current version of the *LiveLily* system, the interpreter can parse commands that are passed as arguments to other commands. The syntax for this feature is the following: if a command takes one argument, this is written immediately after the name of the command, while if the command takes more arguments, these are placed within curly brackets. To maintain consistency with *Lilypond*, which has inspired the development of *LiveLily*, when a command takes multiple arguments, the first one can be written immediately after the command name, not placed within curly brackets, and the rest of the arguments can be placed inside curly brackets right after. This has been enabled due to the way tuplets are written in *Lilypond*, which take the following form:

```
\tuplet 3/2 {c'8 cis' d'}
```

4.3 New Commands

The following new commands have been added to the *LiveLily* language: *bars*, *function*, *list* and the wildcard used to create loops with multiple bars.

4.3.1 The \bars command

In the *Future Work* of the initial paper on this system, the *bars* was mentioned [6]. This feature has now been implemented and enables users to create multiple bars with a single command execution. The example below demonstrates how to create two bars with the *bar* command, followed by a loop that iterates over these two bars.

```
\bar 1 {
   \inst1 e''4 ees''~ 2
   \inst2 c'2 r
}

\bar 2 {
  \inst1 ees''4 g''2 gis''4
  \inst2 g'8 fis' e' ees' c'2
}

\loop 1-2 {\1 \2}
```
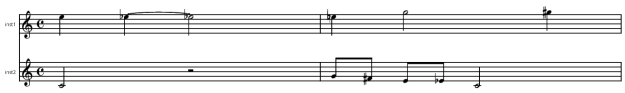


Figure 6. A two-bar loop.

This example will create the two-bar loop shown in Figure 6. With the addition of the *bars* command, the code above can now be written as follows:

```
\bars 1-2 {
  \inst1 e''4 ees''~ 2 | ees''4 g''2 gis''4
  \inst2 c'2 r | g'8 fis' e' ees' c'2
}
```

The code above will create both bars and the loop of the first code example. The different bars are separated by the vertical bar character (|). Also, when creating multiple bars this way, the user still has access to the each of these bars separately by appending a hyphen followed by the bar index to the name of the bars. In the example above, the first bar can be accessed with *1-2-1*, and the second bar can be accessed with *1-2-2*.

4.3.2 Using the wildcard

The wildcard character can now be used in a *loop* definition to include either all rendered bars or all bars whose names share the same character(s). It works similar to the way this character is used in the terminal of a Unix system. If, for example, the user creates some bars whose names start with the word "intro", while other bars do not contain this word in their name, then they can create a loop with all the bars that do contain this word in their name with *loop introloop* {*intro**}, —here "introloop" is used as an example name for the loop of all bars whose names start with "intro". To create a loop with all the defined bars, regardless of their names, the user can type *loop loopname* {*}.

4.3.3 The \function command

The next addition to the command-set of the *LiveLily* language is the *function*. This command enables the user to create a function with or without arguments that they can call at a later stage. The code below is an example of the use of the *function* command.

```
\function solo1-3 {
  \solo \inst1
  \solo \inst3
}
```

The function defined above will mute all instruments but *inst1* and *inst3* with a single call. Once defined, the user can call the function with the name they defined it with. In the example above, the function can be called with *solo1-3*.

Functions can also be bound to a certain event, like the beginning of a loop, the beginning of a bar, the beat, the frame rate, or a specific note. To bind the function of the example above to the beginning of a loop, the user must type the following:

```
\solo1-3.bind loopstart
```

Function binding to events can happen for a specified number of times and certain number of binding events can be skipped. If the user wants to bind the function of this example at the beginning of every other loop for four times, they should type the following:

```
\solo1-3.bind loopstart every 2 times 4
```

4.3.4 The \list command

The last addition to the command-set of the *LiveLily* language is *\list*. This command enables the user to create lists with data that can then be traversed to enable less typing. The code example below creates four staves for percussive instruments and uses a list to turn them all to one-line staves that are used for rhythm. The dollar symbol in the example below is replaced by the items of the *\perc* list, one at each iteration.

```
\insts perc1 perc2 perc3 perc4
\list perc {\perc1 \perc2 \perc3 \perc4}
\perc.traverse {$ rhythm}
```

4.4 Language Design

The language design of the *LiveLily* system has changed to allow for a more coherent structure. The commands of the language are now separated to first and second level. An example of a first level command is *\score*, and an example of a second level command is *show*. These two are concatenated with a dot, *\score.show*. Some commands take arguments, which follow the command with a white space, or as already mentioned above, in case these are more than one, inside curly brackets. Such an example is the *\score.show barcount*, where *barcount* is an argument to the second level *show* command.

Apart from the predefined commands, instrument names, bars, loops, functions and lists that are defined by the user behave like commands. To easily distinguish one from another, each of these types take a different color in the syntax highlighting, but they behave like commands in the context of taking either arguments or second level commands. For example, a bar name followed by an instrument name as a second level command, will return the line of that instrument of this bar. The example below uses this feature to copy the melodic line of the saxophone from bar *\1* to the clarinet in bar *\2*.

```
\bar 2 {
  \clarinet \1.\saxophone
}
```

4.5 Python Integration

In the *Future Work* of the initial paper on *LiveLily*, integrating Python was mentioned, among other features [6].

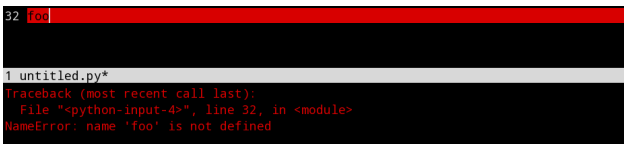


Figure 7. A Python error message integrated into *LiveLily*.

Python has now been integrated into the system through the *Pyo* integration in OF. *Pyo* is a DSP module for Python written in C [11]. The code that integrates *Pyo* in OF has been modified to route Python's both STDOUT and STDERR into the C++ code of *LiveLily*, so that can be fully integrated into the error, warning, and note message handling of *LiveLily*, as show in Figure 7.

Since Python has been integrated through *Pyo*'s OF embedding, the *Pyo* module is also included. This enables users to write their DSP code in Python with *Pyo* in the *LiveLily* editor and control it with the scores they write in the same system. In this case, no OSC or MIDI is needed to control the various *Pyo* synthesizers. All the user needs to do is define an instrument to send its notes to Python with *\instname.sendto python*. This command directs the sequencer to route the Note On, Note Off, and dynamics messages straight as Python messages. For this to work, the user must create a Python class with the name of the instrument (without the backslash), and in this class, to define a *setFreq()* and a *setAmp()* method, to control the frequency and amplitude of the class. For an instrument defined in *LiveLily* with the name *synth*, the Python code example below would be valid. Then, in the pane with the *LiveLily* code, the user should write *\synth.sendto python*.

```
class Synth():
  def __init__(self):
    self.adsr = Adsr()
    self.supersaw = SuperSaw(mul=self.adsr).out()
  def setFreq(self, freq):
    self.supersaw.setFreq(freq)
  def setAmp(self, amp):
    self.adsr.mul = amp
    self.adsr.play()

synth = Synth()
```

4.6 Reading MusicXML Files

MusicXML is a format for music scores that enables the exchange of scores between programs [12]. This feature was added during the postdoc research of the author, where a collaboration with composers that provided original music made this feature necessary, as these composers used various score engraving software. The MusicXML format provides an extensive flexibility in communicating score elements. *LiveLily* being rather limited as to what it can display —see the *Basic Usage* section—, ignores elements that are not used within it.

When a MusicXML file is loaded in *LiveLily*, the score is written bar by bar in the native language of the system. Then, it is left to the discretion of the user whether they will create a loop with all the bars and play the entire piece, or

whether they will treat the score in some other way. Reading MusicXML files provides flexibility in the context of composers sharing their pieces with *LiveLily* users, or even composers loading their own compositions to *LiveLily* so they can experiment with their structure.

4.7 Connecting to Serial Devices

The *LiveLily* system can now connect to external serial devices like the Arduino. It can read and write bytes to/from such a device. This can be helpful if the user wants to control external devices through the score. By utilizing functions that can be bound to certain events, it is easy to bind physical output through a serial device to a musical event such as a specific note or the beginning of a bar. The code below is an example of how one can connect to a serial device and bind the sending of a byte with a random value to the beginning of a loop.

```
\listserialports
\openserialport 1
\function sendbyte {
  \serialwrite \random 255
}
\sendbyte.bind loopstart
```

4.8 Enhanced OSC features

In its initial release, *LiveLily* used OSC only to communicate the score to other software. In its current version, the user can define their own OSC clients to send arbitrary information to other software. An example of how this can be useful is the use of a clicktrack. Even though *LiveLily* includes a blinking cursor so that performers can visualize the tempo and synchronize themselves to other performers or electronics, having used this software with performers has highlighted the need for adding a clicktrack on top of the blinking cursor. Since *LiveLily* does not produce sound, a clicktrack can be created in another software and it can be controlled by *LiveLily*. The code below creates an OSC client sending messages to the home IP and port 9010, named *\click*. It then creates a function that sends the beat count —using the *\beatcount* command— to the OSC address */beat*, and binds this function to the beat of the sequencer.

```
\osc click
\click setup 9010
\function sendbeat {
  \click.send /beat \beatcount
}
\sendbeat.bind beat
```

The code chuck above results in *LiveLily* sending the beat count —with 1-based counting— via OSC. This can be used in another software to control a clicktrack. For example, a clicktrack can be created with a simple Pd patch with an oscillator with a short envelope, where when receiving *1* through OSC, the oscillator's frequency is set to 880, and for all other values it is set to 440. This way, the first beat of every bar will sound an octave higher than the rest of the beats, like the traditional clicktrack works.

With the integration of Python and *Pyo*, the same can be achieved inside *LiveLily*. To accomplish that though, an additional staff must be created for every bar definition, that will contain an A note above the staff with a G clef for the first beat, and an A note inside the staff for all other beats. It is therefore probably simpler to create the clicktrack in another software and control it with OSC. This way, two different audio interfaces can be used, one for the clicktrack that is played by another software and one for *LiveLily*, in case of using *Pyo* to combine instrumentalists with live electronics.

5. FUTURE WORK

Even though many new features have been added to the *LiveLily* system, there is still room for improvement. A feature that has been in the list of features to add but has not yet been implemented is to add support for the Lua scripting language. This language has been targeted mainly because of the ofxLua addon for OF which allows the user to write OF code in Lua for changing an OF program dynamically. The aim is to treat the score as a visual element and apply OF code to it, for example to blur, degrade, delay the image or apply other visual effects to it.

An addition that is currently being implemented is reading MIDI files to render them as scores. This feature is currently close to being realized, and will most likely be included in the next version of the software. A difficulty this feature presents is the interpretation of durations, as MIDI files include Note On messages only and their duration, which is not the entire duration of the note to be played. For example, for a quarter note at 60 BPM, the duration of the note will not be exactly one second but a bit less, to account for triggering the release part of the envelope of the MIDI instrument. If a note is written as a quarter note staccato, then the duration is even shorter, and it is difficult to decide how to treat it. Once these issues are dealt with, reading MIDI files will be added to *LiveLily*.

6. CONCLUSIONS

I have presented the evolution of the *LiveLily* system. This system provides a solution for Live Scoring a traditional Western-music score through Live Coding, a combination that, to the best of my knowledge, does not exist in other systems. Since its initial announcement, two years ago at the time of writing, there has been a significant number of additions to this system. These additions provide a smoother workflow when using this system for Live Scoring.

Having already been used in both professional artistic and academic contexts, it has been tested in realistic conditions where Live Scoring has been the main characteristic of these performances. These performances concerned both experimental new music and more conventional genres like techno. This demonstrates a wide range of applications that are possible with this system, concerning both the music style and the nature of such a performance, whether it is purely artistic or academic.

7. REFERENCES

[1] D. Fober, Y. Orlarey, and S. Letz, "INscore an environment for the design of live music scores," in *Proceedings of the 2012 Linux Audio Conference, LAC*, California, USA, 05 2012.

[2] G. Hajdu and N. Didkovsky, "Maxscore - current state of the art," in *Proceedings of the International Computer Music Conference, ICMC*, Ljubljana, Slovenia, 2012, pp. 156–162.

[3] A. Agostini and D. Ghisi, "Real-Time Computer-Aided Composition with bach," *Contemporary Music Review*, vol. 32, 02 2013.

[4] J. Bresson, "Reactive visual programs for computer-aided music composition," 07 2014, pp. 141–144.

[5] A. Mclean, "Reflections on Live Coding Collaboration," in *Proceedings of 3rd conference on Computation, Communication, Aesthetics and X (xCoAx)*, Glasgow, Scotland, 01 2015, pp. 213–220.

[6] A. Drymonitis, "Livelily: An expressive live sequencing and live scoring system through live coding with the lilypond language," pp. 256–261, May 2023. [Online]. Available: http://nime.org/proceedings/2023/nime2023_37.pdf

[7] T. Magnusson, "Algorithms as Scores: Coding Live Music," *Leonardo Music Journal*, vol. 21, pp. 19–23, 12 2011. [Online]. Available: https://doi.org/10.1162/LMJ_a_00056

[8] H.-W. Nienhuys and J. Nieuwenhuizen, "Lilypond, a System for Automated Music Engraving," in *Proceedings of the XIV Colloquium on Musical Informatics (XIV CIM 2003)*, Firenze, Italy, 05 2003.

[9] A. Drymonitis and M. Koutsomichalis, "Composing for acoustic robots - instant synthesis for computer-controlled acoustic instruments through live coding and ai: Research in progress," *Chroma: Journal of the Australasian Computer Music Association*, vol. 40, no. 1, Jul. 2025. [Online]. Available: https://journal.computermusic.org.au/chroma/article/view/21

[10] A. Drymonitis and N. Chatzopoulou, "Data mining / live scoring – a live performance of a computer-aided composition based on twitter," in *Proceedings of the 2nd Joint Conference on AI Music Creativity*. AIMC, Jul. 2021, p. 10. [Online]. Available: https://doi.org/10.5281/zenodo.5137948

[11] O. Bélanger, "Pyo, the Python DSP Toolbox," in *Proceedings of the 24th ACM International Conference on Multimedia*, ser. MM '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1214–1217. [Online]. Available: https://doi.org/10.1145/2964284.2973804

[12] M. Good and G. Actor, "Using MusicXML for file interchange," in *Proceedings Third International Conference on WEB Delivering of Music*, 2003, pp. 153–.

DECLARATIVE MUSIC COMPOSITION WITH EVENT GRAPH TRANSFORMATIONS

Taosheng Qiu
National Institute of Informatics, Japan
asrcpq@gmail.com

Ryutaro Ichise
Institute of Science Tokyo, Japan
ichise@iee.e.titech.ac.jp

ABSTRACT

Representing music in declarative languages allows accurate, human-readable representation, and the automated processing of musical data. Many existing music programming languages, like engraving systems, focus on high-level representation of musical notation, offering users limited access to low-level event processing. Other music programming languages, like music processing libraries, focus on processing low-level events, thus cannot directly be used to declaratively represent musical concepts.

In this work, we propose a unified graph-based event model used throughout the compilation process, from high-level musical notations to low-level audio rendering. Musical events are represented as graph nodes connected by edges representing temporal dependencies, offering a more expressive model than traditional hierarchical series-parallel models. Event graph transformations are explicitly defined as first-class functions, which give users full transparency and extensibility to the compilation procedure. This framework is implemented in a general-purpose language that offers flexible syntax for convenient music writing, without introducing language-specific biases towards any musical system. The expressivity, extensibility, and flexibility advantages of our system are demonstrated through several examples of music segments and their implementations, involving complex temporal voice structures and notations that previous systems struggle to handle.

1. INTRODUCTION

Music representation languages use programming languages to represent and process music. Compared with interactive editors, music programming languages can represent musical notations declaratively, and enable templating musical structures or batch-editing of objects.

A core design problem of music programming systems is the framework for representing and processing musical data. Traditional systems use a fixed depth structure or series-parallel graphs with a fixed list of musical objects, with associated predefined methods for compiling music directly to sheets or MIDI. As a result, music representation languages generally fall into two categories: high-level engraving systems, which offer highly specialized DSL for music notation but limited programmability; and low-level event-based systems, which offer full programmability but lack the ability to represent musical notations as declarative contexts.

To solve this problem, we propose a unified graph-based event model that use first-class transformation functions to explicitly modify its mutable event graph. Contextual information is explicitly represented as nodes propagated during compilation. Compilation is based on first-class transformation functions, which allow users to fully control the algorithm, its execution order, or creating new musical notations with customized compilation flow. The framework is implemented in a general-purpose language with runtime syntax support, allowing users to use and create syntax forms without introducing language bias for any specific musical systems. This system offers a unified approach that provides both simplicity for music composition and full extensibility for music research and analysis.

This paper is structured as follows. In Section 2, we introduce several existing music representation systems. In Section 3, we introduce the event graph representation. In Section 4, we introduce the design and algorithms of event graph transformation. In Section 5, we introduce language and library implementation of the graph event model and transformation functions. In Section 6, we provide several music representation examples to examine the advantages of our system. In Section 7, we summarize and conclude this work.

2. RELATED WORK

Traditional music representation systems focus on sheet engraving. LilyPond [1] is the most popular music engraving programming language. It is inspired from MusixTex [2] but was then separated and reimplemented in C++ and GNU Guile, a language based on scheme R5RS [3]. LilyPond offers a DSL with specialized syntax, abundant pre-defined musical notations, and a decent MIDI rendering engine. Another notable engraving language is the ABC notation [4]. Similar to LilyPond, ABC notation offers convenient syntax and rich primitives for music annotations. Additionally, MusicXML is also considered a low-level engraving language, due to its declarative representation of musical notations. These DSL focuses on direct correspondence to sheet music, thus offers limited programmability, such as extending the DSL with custom notations or algorithmically generate events to reduce repetitions.

The music21 toolkit [5] is a sheet music analyzing, searching, and transforming library. Music21 also provides a

Copyright: ©2025 Taosheng Qiu et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

lilypond-like DSL parser and a MIDI exporter. Internally, music21 is based on an object-oriented representation of musical objects. Therefore, it is more restricted to classical music structures, rather than providing a general music composition framework.

Some music language propose to use operators for music composition [6]. Euterpea [7] and related systems [8], [9] are haskell libraries using graph grammars for music composition. These systems are based on series-parallel graphs, i.e., graphs created from series and parallel compositions. Instead of data, these systems usually represent music as the result of evaluation, thus do not allow efficient editing of the music that have been created. Since musical notations need to be defined as procedures, they cannot be easily exported or stored, which becomes problematic for declaratively representing non-monotonic events like acciaccatura. Another related idea [10] represents musical data as small graphs called tiles, and uses several operators to combine those graphs. This work mainly proposes a mathematical model for building music as immutable graphs, but does not give detailed solutions for processing and representing musical notations.

Finally, there are low-level music event models. The most well-known low-level music representation is MIDI, with its extensions MPE and MIDI 2.0 [11]. MIDI is the standard of digital music exchange data format. A major limitation of MIDI is its channel-based controller. True polyphonic controllers are implemented by MIDI Polyphonic Expression (MPE) and MIDI 2.0. Another example is the Open Sound Control (OSC) [12], a simple protocol for communicating arbitrary events between multimedia devices. These representations focus on representing music as low-level events, thus are not suitable for written by hands.

3. EVENT MODEL

In this section, we will introduce the event graph model for representing musical objects and relations. The graph model could store the local context of musical information, to delay the evaluation from graph building stage to later transformation functions.

3.1 Event Graph

An **event graph** is a directed graph (N, E). Each event $e \in N$ contains a key-value attributes table. For example, events may have a `type` key to indicate the type of the event. Some attributes that have a list of other nodes as values are called **edge attributes**, the values give a list of nodes N such that every $n' \in N$ is connected by edges using the key as the label: $(n, \texttt{key}, n')$.

Event can be exported and loaded as a text-based format at any time, which enables flexible storage for data exchange and caching for incremental building. The exporter will assign an `ID` to each events. Each event is exported to a line containing multiple entries:

$$\texttt{id} = \#\texttt{ID}$$

$$\texttt{key}_i = \begin{cases} \#\texttt{ID}_i^1, \#\texttt{ID}_i^2, \ldots & \text{edge attributes} \\ \texttt{value}_i & \text{otherwise} \end{cases}$$

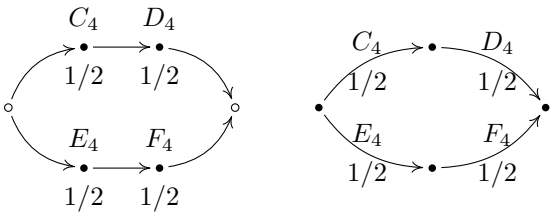


Figure 1. Our node model (left) and the corresponding edge model (right). Rational numbers show the duration of each note. White nodes in the left model are joint nodes for connecting other events.

Representing events as notes is different from some other graph-based models [10] which represent notes or rests as edges connecting anonymous vertices. An example is given in Fig. 1. Our model uses **joint** events to bridge multiple events. An advantage of representing events as nodes over edges is that nodes can be naturally extended with custom attributes to associate with other events, such as dynamic and articulation controlling. Events in an edge model are required to take a duration. Since edge models has shared nodes for different events, attributes need to be applied on the edges, which reduces the integrity and simplicity of the graph model.

An important edge attribute is called the **temporal dependency edge**.

Definition 1. *Node A is called to temporally depend on node B, if a pair of* `next` *and* `prev` *edges exist:*

$$(n_B, \texttt{next}, n_A)$$
$$(n_A, \texttt{prev}, n_B)$$

If node X has a path of `next` *edges to node Y, X is called a **temporal predecessor** of Y; Y is called a **temporal successor** of X.*

Our model, as given in left part of Fig. 1, visualizes `next` dependencies between notes as edges. The temporal dependency subgraph must be acyclic, i.e., it is not allowed to find a loop that contains only `next` or only `prev` edges. Temporal dependencies are used for defining structures of the music. Temporal dependency is created at the first stage of event graph building, before beat-based or bar-based time calculation. Most pre-defined transformation functions navigate the event graph through temporal dependency edges. For example, to reserve the duration of an acciaccatura, the temporal dependencies define notes that should be squeezed to provide the space.

3.2 Context

Many music representation languages implicity track contexts, such as octave and duration during parsing. The event graph transformation model could integrate the context handling, to make the graph building stage fully context-free. We define **Context** as data stored in some nodes that affects other nodes based on the event graph structure. For example, users can create a note with only the note name. Other

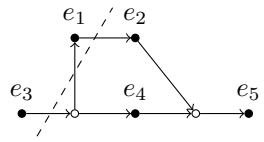
attributes of the note, such as octave, duration, time, are determined by its temporal dependency recursively.

We define **context propagation** as transformation functions that resolve context that affect temporal successors. It is often observed that temporally related notes tend to share similar attributes such as instruments. For example, in MIDI protocol, most events need channel information. Explicitly storing all context in each note as the internal data representation of music programming languages leads to scalability and flexibility problems. Instead, the graph transformation functions propagate context during compilation. The propagation procedure will temporarily construct some **context variables** that can be changed and propagated over temporal dependencies, minimizing the information that each node needs to carry during all compilation stages. Context like time and octave are propagated right after the algorithm stage, context like synthesizer parameter and volume, are kept and propagated until mixing stage,

However, problem occurs when a node is temporally dependent on multiple nodes of different values. Similar case are also observed in engraving markup languages, where inconsistent octave and duration is written before merging multiple voices or notes of a chord. For example, LilyPond [1] always uses the context of the first note. However, for graph-based representations, it will be less reasonable to force an order of edges. In our SMM system, we generally require all dependencies to hold the same value for some variable, to use that value for the current node:

Definition 2. *For any context variable V, If there exists an event subgraph $G' \subseteq G$, that all dependencies of G' in $G \setminus G'$ is an event that set V to a constant value v, while there is no node in G' that set V to a value $v' \neq v$, all nodes in G' are said to* ***naturally inherit*** *the context value v.*

An example of natural inheritance is given in Fig. 2. In this figure, e_1 and e_3 are events that set context variable d. The remaining events are note events. The only possible subgraph that contains e_5 with all dependencies setting d, but no internal events setting d is shown as the right part of the dashed line. However, due to the inconsistency of the value set by the two dependencies, the duration of e_5 cannot be naturally inherited.



e_1 `type` $=$ `setDuration`, `d` $= 1/2$
e_2 `type` $=$ `note`, `note` $=$ C
e_3 `type` $=$ `setDuration`, `d` $= 1$
e_4 `type` $=$ `note`, `note` $=$ G
e_5 `type` $=$ `note`, `note` $=$ E

Figure 2. Example event graph using temporal dependency `next` as edges. Nodes right to the dashed line forms an invalid subgraph due to inconsistent duration settings between e_1 and e_3.

4. EVENT TRANSFORMATIONS

We defined the event graph as the general musical data representation model, and the context which include variables to be later propagated to other nodes. In this section, we introduce event transformation functions which handles conversions between event graph representations and context propagation. We also introduce a pipeline to show the example transformation steps for a typical environment.

4.1 Event Traversal Function

SMM library provides a iterator for templating event traversers useful for writing event graph transition functions. The traverser builder is extended from Kahn's topological sort. As shown in Algorithm 1, this algorithm takes an event graph, a context propagation function f_c, and an ordering function f_o. The context propagation function provided by the user takes the current node and a list of contexts of its temporal dependencies, and returns the context of the current node. The ordering function takes a node and returns a value as a tie-breaker for the topological sort. The procedure returns an iterator to provide the sorted nodes and context.

The topological sort is necessary even after the time of all events have been calculated. To represent events in perfect timing, the order of the events with the same value given by the ordering function must be distinguished by the topological order. For example, if a velocity change event occurs at the same time as a note event, then the topological order will be necessary to decide whether the velocity will be applied to that note event.

A direct application of this function is to implement preview rendering in SMM, by using the time attribute as the ordering function: $f_o(n) \coloneqq n[\mathrm{t}]$, and propagating the renderer function, which takes a note event and returns the rendered audio buffer, as the context. For audio streaming, the framework of SMM is based on rendering each notes independently. This is different from traditional realtime userspace audio drivers such as JACK [13] based on streaming tiny audio buffers between audio processing modules. In SMM, Audio or event streaming can be created from the traverser builder by using the time attribute as the ordering function.

4.2 Reference Transformation Pipeline

This event graph model is used throughout the compilation of the music. The compilation procedure is controlled by transformation of events programmed by users. We will introduce a reference pipeline of our model in four stages, by aligning the compilation procedure with some common musical data formats:

Declarative stage. The high-level algorithmic representation of music, represented as the evaluation result of initial graph building. The compilation of this stage evaluate templates to flattened note-based structure, then propagate necessary context, like octave and duration, to each note. Note that notes in this stage are considered to correspond to a note mark on the music sheet, not the actually played notes.

Algorithm 1 Kahn's Algorithm with Context Propagation

Require: Input event graph G
  A context propagation function f_c.
  An ordering function f_o.
Ensure: An iterator of ordered events and contexts.

```
Initialize H ← empty min-heap.
Initialize Contexts ← empty map.
Initialize Indegree map for nodes in G.
for each node u with indegree[u] = 0 do
    Contexts[u] ← f_c(u, {})
    H.push((f_o(u), u))
                        ▷ Initialize H with custom order
end for
while H is not empty do
    f_o(u), u ← H.pop()
    P ← {Contexts[p] | p ∈ u[prev]}
    Contexts[u] ← f_c(u, P)
                              ▷ Compute and store context
    yield (u, Contexts[u])
    for each v ∈ u[next] do
        Indegree[v] ← Indegree[v] − 1
        if Indegree[v] = 0 then
            H.push((f_o(v), v))
        end if
    end for
end while
```

Therefore, it does not require every individually played note to be explicitly represented. For example, music notations like glissando and vibrato produce more note events than the sheet notation, but is not expanded at this stage.

Sheet stage. The high-level representation of musical objects. This stage can be used for exchanging data with symbolic musical formats like engraving systems. The compilation of this stage includes interpreting musical notations, and resolve the time to absolute beat for each note.

Performance stage. The low-level representation of musical events. This stage can be used for exchanging data with musical communication protocols like MIDI. The compilation of this stage includes frequency calculation, time and tempo calculation, and note rendering.

Rendering stage. The rendered audio. This stage can be used for exporting or streaming rendered audio. The compilation of this stage includes audio buffer mixing and data streaming or exporting.

A key design difference of SMM is that all transformations are based on modifying a mutable event graph. Most previous systems are based on constructing immutable tree-like representation in a single pass, by carefully tracking multiple user-customizable contexts during parsing a specialized DSL. However, representating musical objects as data, instead of as result of evaluation, is required for representing non-monotonic musical objects, like acciaccatura. Additionally, storing musical objects as mutable graphs allows the separation of stages for data exchange with various formats.

5. IMPLEMENTATION

A flexible language is required for efficiently and universally building event graphs and writing transformation functions of event graphs introduced in previous sections. In this section, we introduce the implementation of the `pv` language, which includes a language which allows user-defined runtime syntax, and the S-expression Music Markup (SMM) music library implemented on top of it.

5.1 Base Language

We propose a general-purpose language `pv` that can be used for writing graph creation and transformation functions. The syntax of the language is close to Scheme, which is based on S-expressions, and rely on lambda functions instead of object-oriented modeling. The interpreter is currently implemented on top of Python, allowing interoperability between Python and underlying FFI libraries. Unlike Python, it allows defining runtime syntax with unrestricted expressivity through vau expressions [14]. `pv` also provides a simple module system based on relative path of the current source file, which enables dynamic loading of musical representations.

One major advantage of using a general language over a DSL is that SMM has no builtin keywords or specialized syntax for musical representation. All musical concepts are implemented as library functions and syntaxes, without polluting the syntax or the namespace. For example, LilyPond represents note names like `c`, cis. ABC notation represents note names like `c`, `^c`. In SMM, the default graph builder's syntax parses some strings like `c`, cs, but user can create custom syntax to simulate other notations, or to implement notation systems for other temperaments or simulating existing notations. Meanwhile, those name can be still used as variable and function names. Note names will override variable names, only in vau functions that customize the evaluation order.

Although libraries like music21 [5] also provide convenient notation systems, they are implemented as a DSL on top of the host language, thus inherently have the same problem as those engraving languages. Such subsystems need to be explicitly parsed and quoted, which increases the verbosity of music writing. Programming on such DSLs relies on manually templating the DSL, offseting the advantage of using a general library for music representation. For example, users cannot conveniently call host language functions inside the DSL, or extend the parser with custom notations, without rewriting it.

We implemented our S-expression Music Markup (SMM) system on top of `pv`, which provides a reference implementation of a set of convenient graph builder and transformation functions. Due to the scalability limitation of the Python-based interpreter, some implementations are split into Python modules loaded by the `pv` programs. For example, audio sample parsing and audio rendering are implemented in NumPy modules. Music is represented as functions, and dynamically loaded by external wrappers which define the interface. For example, a default sheet-to-audio compiler is provided in SMM library, which expect a

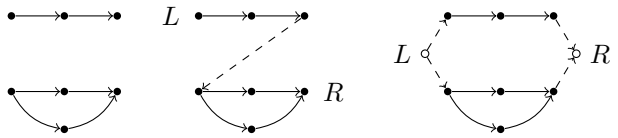


Figure 3. Two subgraphs (left) merged with series (middle) and parallel (right) builders. Dashed lines show additional temporal dependencies. Hollow circles in the parallel builder are joint nodes. Event attributes are omitted. L/R represent the output left and right nodes of the builder.

```
[fn [series meval] [return [vau [. l] env
    [if [== 0 [len l]] [panic 'emptyList]]
    [= first null]
    [= prev null]
    [forin ll l
        [= [p n] [meval ll env]]
        [if [== first null] [= first p]]
        [if prev [connect prev p]]
        [= prev n]
    ]
    [return first n]
]]]
```

Figure 4. Definition of the series operator, defined as a meta-builder that takes an `meval` function and returns the builder function with captured evaluator.

`sheet` function in the loaded module, and will render the evaluated sheet stage representation to final audio file.

Additionally, some utilities are provided. The `soundfont` project extracts SoundFont2 parameters and samples to raw samples. A MIDI parser is implemented to convert MIDI files to performance stage.

5.2 Graph Building

The default parser model in SMM is based on the classical series-parallel graph builder. Inside the builder, every function returns left and right nodes. A series combinator concatenates events one after another, and a parallel combinator fuse all left nodes and all right nodes respectively to a joint node. An example is given in Fig. 3. The series and parallel combinators expect list of pairs of left and right notes, and returns its leftmost and rightmost nodes after merging.

SMM's implementation of the series operator is given in Fig. 4. The function is a meta-builder that captures the evaluation function `meval` and returns the combinator function, which takes any amount of variables, parsing them as chain of events, then concatenate these chains consecutively.

Since during parsing the syntax needs to be recursively applied to child structures, these builders need to be created by taking the custom evaluation function. This allows different parsers to reuse the implementation of these builders, while supporting custom syntax. SMM provides a note parser `line`, which implements the default note parsing on top of the series and parallel operators. The implementation is shown in Fig. 5. The three functions: `line/call`, `line/;`, and `line/:`, are mutually recursive, that the

```
[fn [line/call l env]
    [cond [[list? l] [cond
        [[== '; [@ l 0]]
            [set l 0 'line/;]
            [return [eval l env]]
        ]
        [[== ': [@ l 0]]
            [set l 0 'line/:]
            [return [eval l env]]]
        ...
    ]] [[str? l]
        [= ret [line/evalstr l env]]
        [if [! [null? ret]] [return ret]]
    ]]
]
[= line/; [series line/call]]
[= line/: [parallel line/call]]
```

Figure 5. Part of the implementation of `line` parser's evaluator. Note that `line/;` and `line/:` are instantiated series and parallel builders.

nested call among these functions will prioritize customized parsing logic. Musical keywords like note names are parsed in `line/evalstr` thus does not pollute the namespace.

The `line/call` is then wrapped into a closure which statically captures the right nodes of each call and restores the value as the left node at the next call. This allows writing multiple staves in parallel by independently concatenating each line. Since context is stored as graph nodes, as long as the graph is concatenated, all context variables can be automatically propagated until the end of the line.

An important event parsed by the `line/evalstr` is the **label** event. Similar with MIDI metaevent markers, labels are events for locating specific positions in an event graph. In SMM, labels are not automatically indexed. Instead, the corresponding local variable will be automatically set or fetched. Three forms are parsed by the evaluator:

1. `<label` get the label, report error if undefined.
2. `>label` set the label, report error if already defined.
3. `'label` get the label if defined, set the label if undefined.

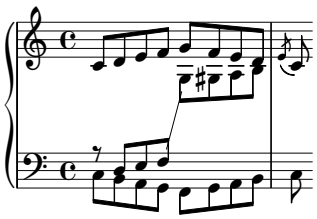
Labels can be used to merge nodes and fuse event graphs, allowing the creation of arbitrary graph structures that cannot be built with the series-parallel graph builder. Labels can also be used for locating the events, that can be passed to the later transformation functions.

6. EXAMPLES

In this section, we provide several examples of SMM. We mainly focus on comparing SMM with LilyPond, which also offers a convenient DSL built on top of Guile Scheme, a general-purpose language. Lilypond is based on a series-parallel graph model. User have limited control for the compilation from sheet to the actual performance.

Source code of the examples, the implementation of SMM, and related libraries is available online. [1] We use Alpine

[1] `https://github.com/asrcpq/2025_tenor`



```
[P1 /2 c4 d e f g f e d |
    >2 [acc [/8 e4] /2 c4]]
[P2 /2 c3 >1 b2 a g f g a b | c3]
[line/; <1 d e f g gs a b <2]
```

Figure 6. An example of splitting and merging voices that raises problem in traditional hierarchical series-parallel graph models. The line indicates that the two beams belong to the same voice. The corresponding SMM program is given below the sheet.

Linux v3.21.3 for the development and evaluation of the examples.

6.1 Expressivity of Event DAG

SMM represent music as graphs, instead of a hierarchical tree model like most other systems. This allows it to extend music representation with complex splitting and merging of voices. Consider a segment of music with three voices upper, middle, and lower voices. As shown in Fig. 6, the middle voice is split from the lower voice and is finally merged into the upper voice.

Since SMM provides series and parallel combinators for basic graph constructions, it is possible to create any DAG by connecting the graph, such as using labels introduced in Section 5.2. Traditional hierarchical music representation systems do not have a direct correspondence of this segment. Some workarounds need to be used, such as writing three independent voices to approximate the DAG model. Such misalignment between the data structure and the actual music leads to many potential problems. For example, in Fig. 6, the last note in the upper voice is an acciaccatura. Acciaccatura is defined to temporarily occupy the duration of previous notes. However, due to the misalignment, it is not possible to define the temporal dependencies of the main note, which need duration adjustments to reserve the space for the acciaccatura.

On the other hand, SMM implements accurate acciaccatura rendering, by modifying the event graph after the sheet stage. As shown in Fig. 7, in graph building stage, a special joint event is create to associate the acciaccatura with the main note as an additional temporal dependency. During beat calculation, the time of the `G_ACC` is calculated in parallel with other dependencies of the main note. In the rendering stage, the transformation function will first calculate the temporal range $t_1 \sim t_2$. Then, it traverse other dependencies of the joint event, reserving this period by adjusting the durations of the notes. Finally, `G_ACC` is inserted between the previous events and the main note. In LilyPond, most performance processing, such as rendering of grace notes, is implemented as fixed logic in the C++ backend. Therefore, this limitation cannot be solved without rebuilding the engine from the bottom.

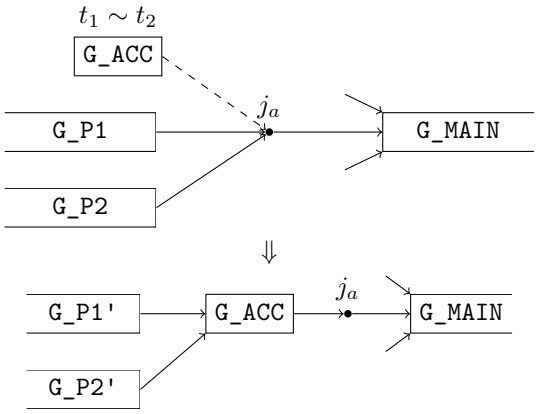
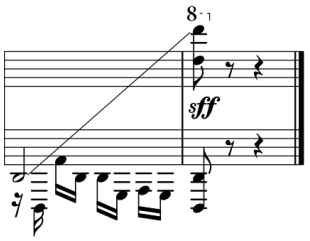


Figure 7. Visualization of transformation for rendering acciaccatura. `G_*` represents event subgraphs. j_g is a special joint node for acciaccatura which maintain an additional edge (dashed) to the ornament.




```
[P1 1/ d2 [EV gliss] c6 | /2 [: d5 d6]]
[P2 /4 r d1 a2 d d g1 a g | /2 [: d1 d2]]
```

Figure 8. Simplified last two bars of Prokofiev's Toccata in D minor. The line notates a glissando. The corresponding SMM program is given below the sheet.

6.2 Extensibility of First-class Transformations

Since SMM defines and applies transformations as first-class functions, it naturally supports extending the notation with customized interpretations. As an example, a glissando in the last two bars of Prokofiev's Toccata's is given in Fig. 8. In building stage, we denote the glissando with the starting and ending notes, with the halves whole duration of the glissando as the note duration. A declarative `gliss` event is inserted between the two notes. This example implements glissandi by specifying a target note `C6`, since the actual target note in the next bar is not involved in time calculation.

After the sheet stage, a transformation function `gliss` expands the declarative glissando to note events. The example implementation is given in Fig. 9. `pp` and `pn` give the pitch (semitone distance to `C0`). The duration `d` of each note in the glissando is computed by dividing the total duration of the glissando divided by the number of notes. A loop over a list of pitches in the glissando created new note events and build a chain between the initial two nodes.

Glissandi usually require customized algorithms for different instruments. For piano, a glissando is performed on

```
[= l [pianogliss pp pn]]
[/= d [len l]]
...
[forin pitch [slice l 1 -1]
    [= vn [% pitch 12]]
    [= vo [/ pitch 12]]
    [= n [EV note [note vn]
        [d d] [t t] [octave vo]]]
    [if [null? first] [= first n]]
    [connect last n]
    [= last n]
    [+= t d]
]
```

Figure 9. A snippet from the `gliss` transformation function which fills a chain of note events between two adjacent notes of the gliss event.
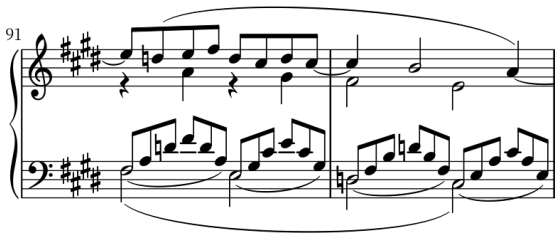
only white or black keys, processed by the `pianogliss` function, which internally generates the pitch sequence by skipping pitches that have a different key color from `pn`.

Most existing music representation systems either cannot represent glissandi declaratively, or do not provide an easy extension for handling glissando compilation. For example, LilyPond offers sheet representation of glissando, but does not render it in the MIDI output. The rendering process is often handled by internal algorithms, which can only be controlled with some exposed parameters. For many systems based on immutable graph representations, they also do not allow customized transformation functions, without inefficiently rebuilding the whole music structure. Therefore, SMM as a universal event graph models allowing transparent access to the data offers user full customizability throughout the compilation process.

6.3 Flexibility of the Language

We use an example from Debussy's first arabesque, also used in previous research [10], to show the generality of the graph model. The previous work did not give details on how to construct such graphs, which will be discussed in this section. The sheet and corresponding musical notation is given in Fig. 10. `P1` to `P3` using SMM `line` parsers are automatically concatenated between calls. For each note, an optional number follows the key name, which set the octave context variable. All numbers with a slash set the duration context variable to a rational number of beats. Note that since double-stemmed unisons are defined as multiple notes of same pitch played simultaneously, the actual interpretation such as merging these notes should be delayed to the performance stage.

In musical composition, it is usually more convenient to write all staves by grouped measures. SMM, based on runtime syntax and general-purpose graph builders, enables concise syntax for writing multiple staves or voices in interleaved units. Since context of graph building stage is explicitly stored as nodes, octave, duration, and contextual musical notations like ties in Fig. 10, are automatically continued after combining the structure. In LilyPond, it is not easy to implement a similar notation. Due to the separation of the DSL and the underlying language, the




```
[P1 /2 e d e fs d cs d cs - |]
[P2 /1 r a4 r gs |]
[P3 [: [; /3 fs a d4 fs d a3] [; 2/ fs]]
    [: [; /3 e gs cs4 e cs gs3] [; 2/ e]]|]
[P1 /1 cs 2/ b4 /1 a - |]
[P2 2/ fs e |]
[P3 [: [; /3 d fs b d4 b3 fs] [; 2/ d]]
    [: [; /3 cs e a cs4 a3 e] [; 2/ cs]]|]
```

Figure 10. Bar 91, 92 of Debussy's first arabesque and the corresponding SMM program.

```
[= T [vau [n1 n2 n3 n4] env
    [return [eval [list 'line/:
        [list 'line/; '/3 n1 n2 n3 n4 n3 n2]
        [list 'line/; '2/ n1]
    ] env]]
]]
[P3 [T fs3 a3 d4 fs4] [T e3 gs3 cs4 e4]]
[P3 [T d3 fs3 b3 d4] [T cs3 e3 a3 cs4]]
```

Figure 11. Rewriting lower staff of Fig.10 with templated pattern.

user cannot preserve the context, if the segments are parsed before concatenation. While it is possible to add an earlier preprocessing stage to syntactically concatenate the segments before parsing, it will essentially create a higher meta-DSL with Lilypond interoperability dropped.

Since the lower staff contains a pattern repeated for many bars. The pattern can be templated as shown in Fig. 11, similar to LilyPond's `define-music-function`. However, Lilypond uses a heterogeneous model for music processing and representation, which limits the flexibility of the language. Since the representation model is mainly implemented in C++ backend, users can only talk to the powerful Scheme engine with a fixed list of pre-defined musical objects.

7. CONCLUSION

In this work, we propose to use unified, mutable event graphs to represent musical data across different compilation stages. First-class transformation functions explicitly compile event graphs from high-level to low-level representation stages. We propose a general-purpose language for writing event graph construction and transformation functions. We provide several examples to show the flexibility and customizability of our systems, compared with previous musical programming languages. The first acciaccatura example shows SMM's expressivity through its unified event DAG model for music representation. The second glissando

example demonstrates the extensibility of SMM by using first-class functions for transformations, which are explicit and fully controllable by users. The third multi-voice example shows the flexibility of the language and its ability to create a convenient syntax for music writing, without adding global language biases like most music DSLs.

In summary, this paper proposes a unified framework for musical processing, focusing on generality and extensibility, and a practical implementation of the framework that demonstrates advantages in flexibility and coherence. This framework could be a useful model for musical research and declarative music programming. Since most existing declarative music programming languages are based on a direct correspondence between sheet music and their digital representations, a more general graph model with explicit transformations enables efficient and declarative experimentation with new ideas in musical research beyond traditional sheet representations.

On the other hand, SMM library is not designed to directly correspond to sheet representations, which could be a limitation in some circumstances. Currently, SMM lacks the ability to exchange data with digital sheet music formats, such as MusicXML. In the future, we plan to support visualizing event graphs in various media, such as plain text, rendered sheets, and interactive editors. However, controlling the engraving behavior of sheet music is not likely to be included in the core SMM library, due to its focus on declarative music representation. Another future work is to evaluate the usability of SMM through user studies and case studies of musical research.

8. REFERENCES

[1] H.-W. Nienhuys and J. van Nieuwenhuizen, "Lilypond, a system for automated music engraving," in *Colloquium on Musical Informatics (XIV CIM 2003)*, 2003, pp. 167–172.

[2] D. Taupin, R. Mitchell, and A. Egler, "Musixtex. using tex to write polyphonic or instrumental music," in *Proceedings of EuroTeX 93*, 1993, pp. 257–272.

[3] N. I. Adams, D. H. Bartley, G. Brooks, R. K. Dybvig, D. P. Friedman, R. Halstead, C. Hanson, C. T. Haynes, E. Kohlbecker, D. Oxley, K. M. Pitman, G. J. Rozas, G. L. Steele, G. J. Sussman, M. Wand, and H. Abelson, "Revised5 report on the algorithmic language scheme," *ACM SIGPLAN Notice*, vol. 33, no. 9, pp. 26–76, 1998.

[4] C. Walshaw, "The abc music standard 2.1," https://abcnotation.com/wiki/abc:standard:v2.1, 2011, accessed: 2025-07-09.

[5] M. Cuthbert and C. Ariza, "Music21: A toolkit for computer-aided musicology and symbolic music data," in *Proceedings of the 11th International Society for Music Information Retrieval Conference*, 2010, pp. 637–642.

[6] D. Fober, Y. Orlarey, and S. Letz, "Scores level composition based on the guido music notation," in *Proceedings of the 2012 International Computer Music Conference*, 2012, pp. 383–386.

[7] P. Hudak, "Euterpea," https://www.euterpea.com/, 2014, accessed: 2025-07-09.

[8] D. Quick, "Kulitta: A framework for automated music composition," Ph.D. dissertation, Yale University, 2014.

[9] P. Hudak, D. Quick, M. Santolucito, and D. Winograd-Cort, "Real-time interactive music in haskell," in *Proceedings of the 3rd ACM SIGPLAN International Workshop on Functional Art, Music, Modelling and Design*, 2015, pp. 15–16.

[10] D. Janin, "A robust algebraic framework for high-level music programming," in *Proceedings of the 2nd International Conference on Technologies for Music Notation and Representation*, 2016, pp. 167–175.

[11] The MIDI Association, "MIDI 2.0 Specification," https://www.midi.org/specifications/midi-2-0-specifications, 2020, accessed: 2025-07-17.

[12] M. Wright and A. Freed, "Opensound control: a new protocol for communicating with sound synthesizers," in *International Computer Music Conference 1997. Proceedings: Thessaloniki, Hellas. 25-30 september 1997*, 1997, pp. 101–104.

[13] S. Letz, Y. Orlarey, and D. Fober, "Jack audio server for multi-processor machines," in *Proceedings of the International Computer Music Conference*, 2005, pp. 1–4.

[14] J. N. Shutt, "Fexprs as the basis of lisp function application or $vau: the ultimate abstraction," Ph.D. dissertation, Worcester Polytechnic Institute, 2010.

COMPOSING AND DESIGNING WEB-BASED DIGITAL SCORES: PRACTICES FOR NETWORKED MUSIC PERFORMANCE

Xiao Xiang
Central Conservatory of Music
xiaoxiangthemusician@gmail.com

ABSTRACT

This paper presents a series of creative experiments, exploring performance practices with web-based digital scores built using JavaScript and related web technologies. It introduces several projects, including: five experimental scores, and two more fully developed prototypes "ModularTouch" and "Inarticulate". Through an examination of the technical implementation of each project and design, rehearsal and performance experiences, this paper discusses how web-based technologies expand the possibilities for digital score design, notation, and collaborative performance.

Over a period of six months, this study designed seven score projects, which were iterated and performed by a total of 12 performers.

The potential of web-based technologies to support both local and distributed collaborative music making is shown to be a viable and emerging creative practice.

1. INTRODUCTION

Designing digital scores as web-based applications offers a cost-effective approach to ensuring broad compatibility across computer and mobile devices. [1]. To make digital score creation more accessible and allow composers to create scores without advanced programming skills, this approach relies on P5.js [1] as the primary framework. Building on this foundation, Node.js [2] (as a local JavaScript server) and Socket.io [3] (for real-time, bi-directional communication) are integrated to enable seamless interaction between client (for display and interaction) and server (for sending and receiving information) components in networked music performance scenarios. Furthermore, a locally hosted score can be deployed on an internet server, allowing the web-based digital score to be performed by musicians situated in different locations via the internet.

These experiments were developed with the following creative objectives:

1. To ensure that the score information is intuitive and easily comprehensible for both performers and audiences.

2. To create scores that are open with respect to instrument choice, specific pitches, rhythms, and timbres, allowing these elements to be determined collaboratively by the conductor and performers during rehearsals.

3. To enable flexible performance content by allowing a single score program to generate multiple pieces through simple data modifications, thereby facilitating adaptation to a variety of performance contexts.

4. To ensure reliable operation of the scores on client devices (e.g., smartphones and tablets), the system is designed to maintain feasibility and robustness across distinct performance scenarios: co-located performances that rely on local networks and geographically distributed performances that rely on the internet.

2. FIVE EXPERIMENTAL SCORES

Before designing larger projects, five short demo scores were composed to explore concepts central to designing graphic notation with JavaScript. The themes explored were: physical modeling, mathematical counterpoint, randomness, machine learning model, and toporhythm performance. Consistent with the notion that "the graphic/visual imagery used to communicate ideas to the musicians will shift or mutate or change through time" [2, p. 95] , the first four scores were animated to demonstrate musical changes across time. [4] Each score is designed for two to four performers and has a duration of approximately one minute.

The design principles guiding these five experimental scores are as follows:

1. The core concept of each score should be straightforward for both performers and the audience to understand.

2. Instrumentation and playing techniques should remain open, to be decided upon during rehearsal.

3. Trajectory based scores should be predictable, allowing performers to play with confidence.

[1] P5.js is a JavaScript library designed for creative coding that allows users to generate and manipulate graphical content on the web with minimal programming experience. https://p5js.org/

[2] Node.js provides a framework for running JavaScript on the server, facilitating fast and efficient network application development. https://nodejs.org/

[3] Socket.io is a JavaScript library that enables real-time, bidirectional communication between web clients and servers. https://socket.io/

[4] Video recording: https://www.bilibili.com/video/BV13xAheMEWa/

Copyright: ©2025 Xiao Xiang et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

2.1 Three-Body Movement

The first score is inspired by familiar physical principles observed in daily life, specifically through a 2D graphical simulation of a three-body system [3]. In this piece, three on-screen orbs mutually attract one another, generating distinct trajectories for three performers to interpret. Key performance cues such as speed and direction changes, are highlighted through visual prompts, making the concept intuitive. Over the course of one minute, physical parameters like gravity and mass are dynamically altered at predetermined moments, transforming the system's behavior from stable to unstable states—a process the performers are encouraged to reflect musically.

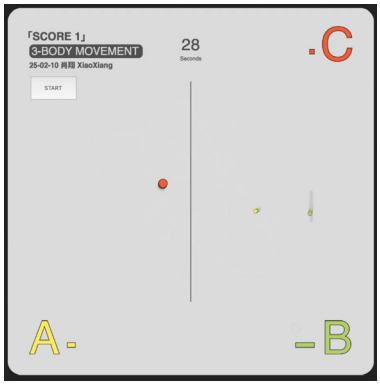


Figure 1. Experimental Score 1: Three-Body Movement https://openprocessing.org/sketch/2531408/

2.2 Lissajous Curves

The second score explores rhythmic patterns through Lissajous curves, visualizing the interaction of two oscillations with different frequencies along perpendicular axes. Beats are defined as pairs of points 180 degrees apart within each oscillation's cycle, and one full traversal of the curve constitutes a performance pattern for the performer. The rhythmic characteristics of the two oscillations are visually embedded in the curve's form, helping performers internalize their material. These oscillations remain stable for most of the piece, changing only at predetermined moments to prompt shifts in musical content.



Figure 2. Experimental Score 2: Lissajous Curve https://openprocessing.org/sketch/2531406

2.3 Random Instruction

The third score explores randomness through the random arrangement of pre-designed graphical elements, each linked to specific performance rules for three performers. While the graphical elements and their instructions are fixed, their order and combination are randomized, making each performance unique. Interpretations are negotiated during rehearsal, allowing the ensemble to develop shared improvisational strategies. Some deliberately ambiguous combinations are also included, providing room for performers' interpretation.



Figure 3. Experimental Score 3: Random Instruction https://openprocessing.org/sketch/2544066

2.4 Hand-Pose

The fourth score employs hand-pose recognition to generate real-time cues for synthesizer performers, with a "score performer" whose gestures are tracked via webcam. Built on the ml5.js library [5], the system extracts 21 key points per hand and translates gestures—such as forming a circle or changing the distance between hands—into visual prompts for the ensemble. Unlike previous scores, precise rhythmic execution is not required; the piece emphasizes ambient texture and fosters an interactive relationship between the score performer and instrumentalists.

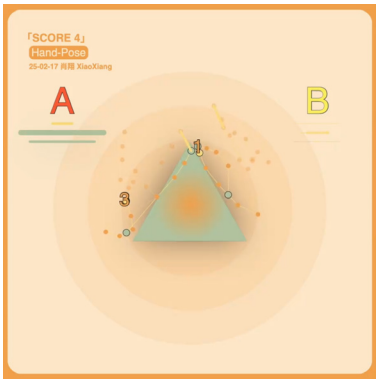


Figure 4. Experimental Score 4: Hand-Pose https://openprocessing.org/sketch/2547417

[5] ml5.js is a JavaScript library that brings machine learning to the web browser. https://ml5js.org/

2.5 BianZhong Gamelan Perform

The fifth score is a network music performance piece about toporhythm [4], inspired by Netronomia II project. [6] The score uses circles to display rhythmic patterns and utilizing the Netronome plugin [7] to manage the timing of audio streamed over the internet. It was performed via a live network connection between Beijing (two performers on Gamelan) and Wuhan (two performers on Bian Zhong [8]). The score adopts different performing orders in its three sections, allowing various playing perspectives under fixed network latency to produce different musical effects.

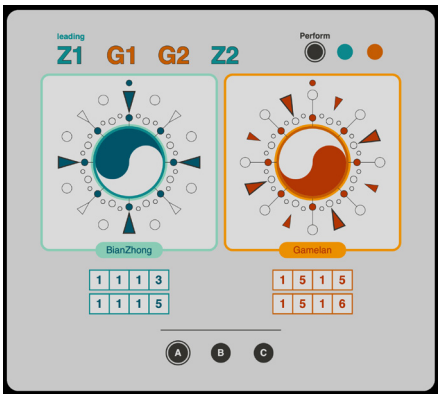


Figure 5. Experimental Score 5: BianZhong Gamelan perform https://openprocessing.org/sketch/2597323

2.6 Reflections on the Experimental Scores

The design and performance of these five experimental scores demonstrate that P5.js provides a convenient and effective framework for creating graphic scores. Although each score is relatively short in duration, non-linear musical structures were embedded directly into the notation, while the specific performance details—such as instrumentation and playing techniques—were intentionally left open for discussion and interpretation during rehearsal. Moreover, the integration of Socket.io highlighted further possibilities for score design, particularly in supporting real-time interaction, synchronization among performers, and distributed performance across networked environments.

3. "TOUCH MODULAR" DIGITAL SCORE

"Touch Modular" is a digital score designed to facilitate collaborative music performance with modular synthesizer system through integrated network technology. The score runs on a local Node.js server, and performers use their mobile phones as both a visual display for the score and an interactive user interface for their modular control. Each performer follows their own "performance curve" on their device, which guides their musical interpretation and interaction. Their performance data is transmitted across the local network using Socket.io. The server then aggregates this information and sends OSC (Open Sound Control) signals [5], which are converted to control voltage via a Max patch, directly interfacing with the modular synthesizer system, making the score system a voltage-controlled digital module [6].

The graphic design of the score reflects the logic of the modular synthesizer; its design serves as an extension of the modular synthesizer dispositif [9] [7]. Each performer controls an oscillator on their device, with the oscillators mutually modulating one another to produce a complex and responsive soundscape. Each performer's touch point generates a gate signal, while the x–y position data and movement acceleration are captured by the server and translated into OSC messages. The server computer then connects either to a virtual audio interface (for software modular environments such as VCV Rack) or to dedicated hardware—such as a high-quality DAC (e.g., the MCP4728)—which converts these messages into Eurorack-compatible control voltages.

Since the information provided by this curve-based score is limited, achieving a shared understanding among the performers requires some pre-established guidelines. Therefore, specific rules are communicated during rehearsals before the performance. For example, for the premiere, three performers are given follow instructions:

1. The music should gradually accelerate from a slow tempo to a pace where it becomes impossible to synchronize with one another;
2. The shape of each touch point is more important than its precise position;
3. When playing with two fingers, ensure that the main touch point remains on the note curve. Only add a second finger when it is necessary to propel the music forward;
4. When players' touch points approach each other, slow down the performance.

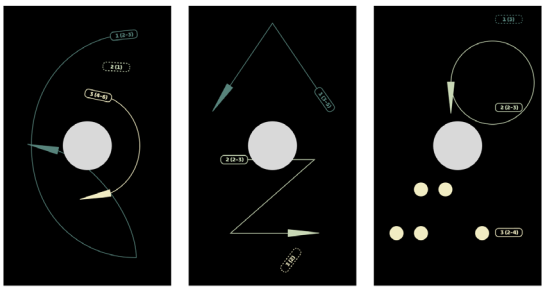


Figure 6. "Touch Modular" Score Performance Curves

The first edition of "Modular Touch" operated over a local network. [10] In this version, the routing of sound and

[6] https://digiscore.github.io/2025-05-22-Netronomia-II/
[7] https://github.com/pparocza/Netronome
[8] BianZhong is an ancient Chinese musical instrument consisting of a set of bronze bells, played melodically.
[9] "Dispositif" here refers to the integrated system of interfaces, technologies, and practices that together shape the interactive musical environment.
[10] First edition performer: Wang XiJia, Zuo MoQi, and Xiao Xiang. Video: https://www.bilibili.com/video/BV1VDQXY3E8B/

control signals was managed within Ableton Live. A custom Max for Live device was developed to receive OSC messages from the local server, which collected performance data from each player's device and converted it into audio-rate control signals. These control signals were then routed into VCV Rack via a 16-channel virtual audio driver "BlackHole". [11]

The second edition of Modular Touch extended the system for online operation. The score project was migrated to an online environment, allowing performers to connect from remote locations. For this version, a dedicated application, OSC Client [12] , was implemented to receive performance data from the online server and translate it into signals compatible with the modular system.

4. "INARTICULATE" DIGITAL SCORE

"Inarticulate" is an interactive digital score that uses audio analysis and a pre-trained neural network to dynamically shape the performer-score relationship, creating a shared performative context for both musicians and the audience. The piece involves three instrumental performers and one score performer. The score system begins with one or more audio samples as input, analyzing and transforming the sounds to create a 'score instrument.' The performance of this score instrument, informed by a pre-trained neural network, simultaneously generates the notations displayed on the animated score and serves as the electronic ambient part of the piece, providing both a sonic and structural model for the instrumentalists to perform. The animated score, enhanced with dynamic visual effects, is projected for the audience. This approach ensures that performers and audience members experience the same score in real time, promoting a sense of synchrony between musical performance and its visual representation. [13]



Figure 7. "Inarticulate" Premiere

This work further investigates the relationship between "notation and performance" that was explored in previous digital score experiments. The score's tempo gradually accelerates throughout the piece, shifting from a very slow pace at the beginning to rapid speeds by the end. The symbols generated by the score performer lead to constantly changing musical content, as instrument performers must interpret them differently depending on the varying tempos and their instrument. In the final, fastest section of the piece, the rapidly changing symbols and extreme speed force the instrument performers to continually adapt their performance strategies—ultimately reaching a point at which it is impossible to fully follow the score's "instructions" and entering an unconscious, instinctive state of performance.

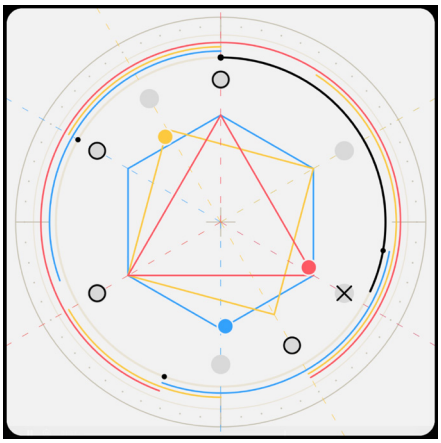


Figure 8. "Inarticulate" Score Graphic

A significant portion of the score's logic is implemented in Max, while the score display and fundamental rules are designed in JavaScript, and Ableton Live is used to route the sound. With the support of FluCoMa's [14] audio analysis and neural network objects for Max, the system analyzes sample input to generate the "score instrument" and provides an interface for live performance. The Max patch also transmits performers' data back to the Node.js server. Part of this information is processed by the server to generate the visualized score display, while other data is forwarded to the visual artist to support the creation of real-time visual effects for the audience.

The practical setup for this score involves running a local Node.js server and Ableton Live project on a computer. This computer creates a local wireless network, allowing each instrument performer to connect their tablet to a local website and read the score in real time. The system allows each client to select which version of the score they receive, enabling the distribution of different information for different roles. For example, the version received by the visual artist is specially programmed to meet the needs of their visual content and can be directly integrated into TouchDesigner.

5. STRATEGY IN REHEARSAL AND PERFORMANCE

Throughout these six months of performing with web-based digital scores, rehearsal has proven to be an essential component. Each score incorporates distinct forms of digital

[11] https://github.com/ExistentialAudio/BlackHole

[12] The app was designed by Lin XiangBin

[13] Premiere performers: Yue Yan(Piano), Wu XueMeng(Gu Qin), Zhou YiNi(Er Hu), Xiao Xiang(Score Performer), and Zuo MoQi(Live Visual). Video: https://www.bilibili.com/video/BV1hcNDzpEAg/

[14] FluCoMa (Fluid Corpus Manipulation) Max libraries offer a suite of objects for advanced audio analysis, machine learning, and corpus-based sound manipulation in Max. https://github.com/flucoma/flucoma-max

technology, opening up distinct possibilities for rehearsal and performance, and reflecting how technological mediation can shape the very conception of a musical work. [8] The graphic design of these scores aims to be as simple and intuitive as possible. For some scores (such as Experimental Score 1: Three-Body Movement, Experimental Score 2: Lissajous Curve, and Inarticulate), the graphics emphasize visual cues for rhythm. In other scores (such as Experimental Score 4: Hand Pose, Touch Modular, and the performer's view in Inarticulate), the graphics focus more on shapes and gestures, providing visual cues related to form rather than specific playing techniques. As a result, much of the information needed during performance is abstract rather than concrete. During rehearsals, collaborative negotiation between the composer (or conductor) and performers determines how the visual themes are musically interpreted."

Much of the score design took place in parallel with rehearsals, ensuring that the scores could be effectively adapted for the performers. Throughout this process, a variety of performance strategies were explored, including the use of open improvisational structures versus pre-composed material [2, p.30]; performing over local networks versus over the internet [9]; and implementing a unified score versus content tailored for each performer. These explorations served not only to refine the score designs, but also to test their musical possibilities and practical feasibility across different settings and diverse groups of musicians.

Rehearsals for "Touch Modular" and "Inarticulate" were more complex. For "Touch Modular," professional musicians and their familiar acoustic instruments were not used; instead, performers engaged with specially designed interactive interfaces and modular synthesizers. As a result, a significant portion of the rehearsal was dedicated to familiarizing the performers with these instruments and making necessary adjustments. In the case of "Inarticulate," since input audio samples became part of both the music and the score, the audio samples were finalized before each performance to ensure consistency across rehearsals. Additionally, a key aspect of the rehearsal was the development of coordination between the score performer and the instrument performers, as the score performer offered musical direction or material, while the instrument performers' reactions, in turn, influence the score performer's strategies.

6. CONCLUSION

These web-based digital score projects aim to create participatory performance environments, and to allow composers to create such works with minimal programming skills. Although implementing digital scores as web apps has its limitations—such as restricted access to mobile device functions like Bluetooth, NFC, and cameras—more powerful features can still be developed with the help of various JavaScript libraries and music programming environments such as Max. The replacement of traditional music notation with graphic design forms a central theme of this research. In this way, certain musical intentions can be expressed and communicated to performers and audiences more intuitively, resulting in a unique approach to performance. Moreover, the nature of networked music performance makes it possible to realize various design approaches—such as establishing a relationship between the score performer and the instrumentalists, or enabling each performer to influence the playing of others—thus opening up possibilities for interactive performance. Ultimately, these explorations demonstrate how web-based technologies can redefine both the conception and practice of musical notation in the networked age.

Acknowledgments

The research is a result of the "Presentation of Digital Score Solutions in the Context of Artificial Intelligence" project at the Central Conservatory of Music (CCOM) in Beijing, China. I wish to thank Professor Kenneth Fields for his longtime invaluable guidance. I am also grateful to Professor Guan Peng, Zuo Moqi, Wang Xijia, Zhang Yuan, Yue Yan and Lin Xiangbin for their steadfast support.

7. REFERENCES

[1] S. Robaszkiewicz and N. Schnell, "Soundworks a playground for artists and developers to create collaborative mobile web performances," *Proceedings of the Web Audio Conference(WAC'15)*, 2015.

[2] C. Vear, *The digital score: musicianship, creativity and innovation*. Routledge, 2019.

[3] G. Flake, *The Computational Beauty of Nature*. MIT Press, 1998.

[4] E. Cayko, "Rhythmic topologies and the manifold nature of network music performance," *UNIVERSITY OF CALGARY*, 2016.

[5] M. Wright, "Open sound control: an enabling technology for musical networking," *Organised Sound*, vol. 10, no. 3, p. 93–200, 2005.

[6] C. Roads, *Composing electronic music: a new aesthetic*. Oxford University Press, 2015.

[7] V.-R. Carinola and J. Geoffroy, "On notational spaces in interactive music," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'2022*, V. Tiffon, J. Bell, and C. de Paiva Santana., Eds. Marseille, France: PRISM Laboratory, 2022, pp. 149–154.

[8] T. Magnusson, *Sonic Writing, Technologies of Material, Symbolic, and Signal Inscriptions*. BLOOMSBURY ACADEMIC, 2019.

[9] K. Fields, "The well-tempered network: From monochrony to polychrony," *Organised Sound, Cambridge University Press*, vol. 30, no. 3, (in press).

AI SENTIENCE IS JUST AROUND THE CORNER: SOLARIS 2, A CREATIVE-AI SCORE AND PUNK ROCK BAND WITH HUMAN IN THE LOOP

Fabrizio Augusto Poltronieri

University of Nottingham
fabrizio.poltronieri@nottingham.ac.uk

Craig Vear

University of Nottingham
craig.vear@nottingham.ac.uk

ABSTRACT

This paper presents the results of a five-year research project, focusing on the creation of an autonomous punk rock improvising artificial intelligence (AI) that features a digital score. Central to this AI is a live performance ensemble called *Solaris 2*, comprising AI-generated bass, drum machine, and guitar, as well as human musicians. To develop *Solaris 2*, we built an unprecedented dataset that captured specific physiological parameters from a series of professional musicians. We developed a hive of neural networks trained on the resultant dataset, which was used to couple the musical core of *Solaris 2* with a digital score, enabling human musicians to play along with it. The results obtained from the interaction of human musicians with *Solaris 2* are surprising, as they indicate that human musicians perceive *Solaris 2* as a competent improviser who acts and responds in real-time to the musical space (*musicking*). This led us to the third stage of the research, which we present here as the conceptualization of *Solaris 2* as being close to a sentient Creative-AI system.

1. INTRODUCTION

Creative-AI has been a field of theoretical and practical encounter between the authors of this research. For over twenty years, we have been exploring diverse topics, including the application of AI in experimental processes and various artistic languages. Since we began working together in 2016 and established a Creative-AI Research Lab at a previous institution, our research efforts have focused on models that utilise a bottom-up topology. For this, we start with specific problems and move towards general models, always having creative processes at the heart of our projects. This bottom-up approach focuses on human experience over the more pervasive top-down approach of recreating the artefact of sound, music or art.

Copyright: © 2025 First author et al. This is an open-access article distributed under the terms of the Creative Commons Attribution License 3.0 Unported, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Our research indicates that the creative application of AI in symbiotic environments, where human creativity and AI-driven creative forms coexist and share spaces and processes transparently and in real-time, is an effective way to leverage new methods and visions [1]. This path and our interest in music led us first to create *Solaris: A Jazz-AI ensemble*. In this, the AI performed the piano, bass, and drum machine along with human musicians. Together, they symbiotically generate the improvisation in real time and are part of a universe of musical meaning (*Musicking*) inserted into a flow of musical creation. *Solaris 2* presents a step forward in terms of the digital score component and a departure from jazz.

2. CURRENT STATE OF THE ART AND MOTIVATION

Performance modelling of music has been a research challenge for some time, and more sophisticated real-time solutions are being developed. However, there is a lacuna in the systems and knowledge around performance modelling from an embodied, bottom-up perspective.

From a symbolic modelling perspective, the bulk of computational models attempt to codify musical expression in terms of mathematical formulae or symbolic programs, unlike our research, which aims to stimulate creative relationships. For example, the Cancino-Chacón model focuses on enabling machines to perform music that can be perceived as expressive [2]. It achieves this by manipulating musicking artefacts such as pitch, phrases, and timing into expressive parameters.

The most modelled parameters within these systems focus on loudness, articulation, ornamentation, and timing. These are generally controlled using the MIDI protocol, and in popular music genres such as folk, jazz, and pop [2]. A common concern with music performance modelling is performance generation, rather than in-the-loop real-time interaction. For instance, the Music Transformer system generates realistic accompaniments and performance given a melodic line input [3]. However, in the limited literature surrounding this topic, it appears that improvisation and variation are usually ignored when modelling

classical music performance. However, other styles (jazz and some folk traditions) consider them essential aspects of expressive performance [2].

3. FOUNDATIONAL CONCEPTS

3.1. Can an AI play music in the loop with humans?

Musicking and *flow* are the two main theoretical concepts that guided the development of *Solaris 2* [4][5]. The application of both led us to realize that sentience applied to AI could be linked to the practical context in which artistic creation occurs. We used the idea of *musicking* to construct meaning in the musical pieces created by *Solaris 2*. *Musicking* is a term coined by Christopher Small to define music from the perspective of doing, as the word "music" lacks a corresponding verb. For Small, *Musicking* is the process of making music by participating in all the acts involved, whether composing, performing, listening, or dancing [4]. He emphasizes that from the point of view of *Musicking*, the construction of musical meaning-making is to be found in the relationships forged within these creative acts [4]. In this way, musical meaning is generated through the emerging relationships between the actions, materials, behaviors, and agents that comprise these acts, such as people, algorithms, sounds, space, and time [6]. Through *Musicking*, musicians can undergo a process in which their sense of bodily self dissolves, and they become music. Csikszentmihalyi's theory of *flow* also supports the argument that the acts involved in making music should be considered immersive and corporeal experiences [5]. The author defines flow as a state in which people are so involved in a particular activity that nothing else seems to matter.

Solaris 2 was designed to be considered an integral part of the *Musicking* experience by stimulating an acute sense of being in the same immersive flow as human musicians. From this perspective, there is no hierarchical distinction between human musicians and the AI in *Solaris 2*, all of whom are central to the *Musicking* decision-making process. In *Solaris 2*, humans and AI are engaged in the shared endeavor of *Musicking*, sensing each other's activities and being attuned to the flow of their shared environment, which is the musical world. This view is supported by the widespread understanding that the act of making music does not consist of merely emitting sounds but of engaging in a bodily experience of becoming the sounds created in the flow of musical creation [6].

The project's first phase, spanning 2019 to 2020, involved the creation of an unprecedented, embodied dataset that captured the multidimensional interrelationships of professional musicians playing a jazz standard piece. The process of capturing and creating this dataset, as well as a detailed analysis of its results and training the neural networks, has been detailed in three other papers and book chapters [7, 8, 9]. For *Solaris 2*, this has been expanded to allow the system to play other genres, such as Punk Rock.

3.2. Experimental process

Having built a new dataset that captures the essence of *Musicking* and *flow*, we began to ask many questions about the nature of this Creative-AI and how to develop it for real-world applications. Generating ideas is an integral part of the music improvisation process. Hargreaves observes that in generating musical ideas during improvisation processes, different sources can be used initially, and the subsequent selection of sources is context dependent [10]. Starting from a vast repertoire, it is narrowed down within a specific context. Could *Solaris 2* make this cut, generating new ideas from a previous repertoire? Is there room for what Bloom calls “influence” [11]? The author describes the process of influence as the struggle of an artist to overcome the influence of their precursors, creating an art of their own. Could *Solaris 2* create something so unusual that its human peers could consider it a novelty?

Another crucial factor is the relationship between improvisation and embodiment. Music, for all its cognitive aspects, is highly embodied, and this becomes evident in the hands of the guitarist, the mouth and breath of the flautist or trumpet player, or the dance of a drummer, the way the musicians’ bodies move when they perform, and so on [12]. One of the factors that makes music so embodied is its interactive aspect, the unified way the musician connects with their instrument and into musicking. Embodiment is also a result of our natural response to music. Jazz audiences tend to make their involvement with the music explicit through physical signals, such as clapping their hands or stomping their feet [12]. Additionally, developing skilful musical agency involves assuming and assimilating various body positions, postures and movements [13]. This is particularly important for improvisation, since there is a phenomenological aspect to musicians’ body attitude that informs their peers about various musical elements [14].

3.3. The problem of embodiment

Since music is such a bodily activity, we faced the problem of embodying *Solaris 2*. Although our dataset represents the embodied performance of musicians, which is undoubtedly present in how *Solaris 2* is integrated into *Musicking*, it was necessary to embody the system in its performance, thereby avoiding the phenomenological sensation that *Solaris 2* was merely a computer producing musical notes.

The field of Embodied-AI studies is vast, bringing together researchers from diverse fields [15, 16, 17]. Unsurprisingly, this field’s central feature is the widespread conviction that intelligence should be embodied and conceived in terms of physical agents – biological or artificial – that act in the real and social world. Given this perspective, much of the work and research in the area involves designing and constructing robots or other types of artefacts. One of the reasons for the transdisciplinary nature of Embodied-AI is that intelligence has a strong connection with behaviour and interaction with the real world, just like music!

3.4. Embodied approach to *Solaris 2* AI

To conceptualise the theoretical hypothesis of this project into a real-world situation, it was necessary to conceptualise an AI architecture that mirrored aspects of *Musicking*. In short, as it is not the remit of this paper to outline the AI design in detail, we designed a modular design of:

- Layer 1 - Percept input and formatting. This module manages all the real time input data from a single audio input and sends it to the AI factory and gesture manager for processing.
- Layer 2 - AI Factory. This module generates streams of data from the four neural network models in the AI factory. Each of these models are created using 2D recurrent neural networks that have been trained on single features of the dataset. For example, a single point of the pianist's movement with the related arousal data from that individual. This is a free-flowing loop with all RNN's predicting live. The purpose is to generate a constant flow of predicted data from each model, emulating a busy mind negotiating options inside music-making.
- Layer 3 - Gesture manager. This module chooses one of the outputs from the AI factory (layer 2), or live input stream (layer 1) and holds this stream for a few sections before randomly choosing the next.
- Layer 4 - Belief system. This module takes the single output stream from the gesture manager (layer 3) and uses its value to determine which instrument will play and which note to play. It also sends conditions to the visual generator to convey its inner state.

3.5. Embodying *Solaris 2* through a digital score

The next stage of the project involved developing strategies for embodying and communicating *Solaris 2* with the real world, i.e., the human musician and the audience. This phase resulted in the development of a graphic interface for the project, which functions like a digital score (Figure 1), working in conjunction with the sonic aspect of the AI. This digital score phenomenologically informs the human musician through a visual language, providing sensory information about the mood of the rhythm section that forms the core of *Solaris 2*'s AI.

A digital score is characterised as a technically mediated communication interface that enhances the representation of ideas in music [6]. The digital score developed in this phase enables *Solaris 2*'s AI to communicate its presence, behaviour and responses to the outside world. As such, it helps stimulate the sensation of joint action when performing with human musicians. In our participatory observation process, we noticed that from a human point of view, notions of self-representation within musicking together in *Solaris 2* gain new representations, mainly through the mediation generated by the continuous interface, which is a constituent part of the digital score. The digital score helps define the space for *Musicking* to take place, establishing a constant link between the artificial and human minds, both of which are endowed with the ability to make mental representations, and providing a space for creative invention.

Just as the movements of the bodies of human musicians improvising indicate to their peers the general atmosphere of the musical piece, the visual forms of the digital score inform human musicians of the internal movements of *Solaris 2*'s AI architecture, thereby giving a "body" to this artificial musician. It is important to emphasise that *Solaris 2*'s digital score differs from a classical score, which brings detailed information on how the music should be performed. *Solaris 2*'s digital score is a set of abstract shapes and colours that provide a feeling, along with the sound, about the piece in question, open to interpretation and improvisation, and embracing the affectual and phenomenological properties of the media. In this way, the communicology of *Solaris 2* unfolds through the sounds produced and the visual representations of its internal state, as represented in a digital score. The digital score is projected for the human musicians to interact with when performing.

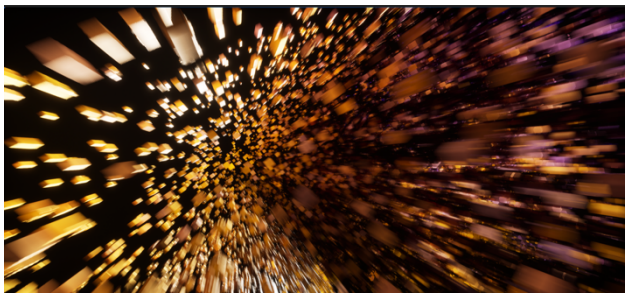


Figure 1. *Solaris 2'* visual interface, displaying a frame of its digital score.

4. A NEW TAKE ON SENTIENCE IN CREATIVE-AI

AI sentience has become an important topic of research, discourse and public debate considering the remarkable progress and capabilities not only of Large Language Models (LLMs) but also in expert systems, such as *Solaris 2*, which employs an entirely different approach to AI.

AI sentience is not related to the technology used to train models, but rather to how the AI is designed, perceived by users, and the quality of its outputs. *Solaris 2*, even without employing LLMs, can generate cultural artefacts – music – of such high quality that it is difficult to distinguish from human-composed music. The fact that Solaris 2 does not employ LLMs is a design decision, as discussed in [7, 8, 9]. Additionally, LLMs typically require user prompts to generate responses, lacking the ability to initiate interac-

tions independently [18]. They can mimic human language, but do not possess the capacity to engage in nuanced social interactions, such as *Musicking*, missing the capability to generate culturally relevant content beyond their training data.

Let us take as our starting point the concept that we want to dismantle: The idea, still dominant in AI studies, that the most pertinent question to be answered is whether "AI is sentient" [19]. This question is based on the Cartesian assumption that consciousness, sentience, and personhood are closely related terms, stipulating that language is a reliable indicator of consciousness. The primary interest of this approach is to define what consciousness is and whether an entity possesses it.

Our main question is: Can a system like *Solaris 2*, which is embedded with a level of intelligence and able to play in real time, in the loop with humans, be considered sentient? Sentience (from the Latin *sentire*, to feel) means the ability of beings to feel sensations and feelings and have conscious perceptions of their surroundings and what happens to them. Scholars across various disciplines and fields have long grappled with questions of sentience, including those in philosophy, psychology, and sociology. Sociologists, for instance, have a long history of considering sentience [18]. The scientific study of sentience, which is usually related to the possession of conscious experiences, including genuine feelings of pleasure and pain, is highly contentious [20].

The concept of sentience emerges in traditional narratives to describe the distinctions between plants and animals that were apparent to the ancient world [21]. Aristotle's account of sentience, for example, emphasizes the unique sensitivity of animals compared to plants [22].

Recent theories of consciousness, such as the "global workspace theory" [23] and the "attention schema theory" [24], indicate that we might not be far away from creating genuinely conscious systems, and machine sentience would require only incremental changes or piecing together existing technology in the right way.

One of the key issues with understanding sentience and demonstrating its existence at a scientific level is that the concept relates to a being's thoughts, feelings and emotions, none of which can be fully understood by physiological processes or anatomical structures [20]. As a result, sentience is often described as anthropomorphic assumptions, in other words, the attribution of human characteristics to animals or AI. The "carnal sociology" framework for ethnographic research has a similar approach, arguing that humans and animals are sentient because we are "endowed with senses, exteroceptive, proprioceptive, and interoceptive; she senses of what her[1] sensorium captures" [20]. There are no elements of physical or visual anthropomorphism in *Solaris* 2. There is no android or robot replacing humans, and the visual score is entirely composed of abstract shapes, which is a conscious design decision.

On the other hand, an AI sentient system should feature embodiment, world- and self-models, recurrent processing, a global workspace, and unified goal hierarchies [26]. Global workspace theory is a cognitive architecture and theoretical framework for understanding consciousness, developed to explain both conscious and unconscious processes qualitatively [27]. From this list of features, *Solaris 2* is characterised by embodied world- and self-models, recurrent processing, and unified goal hierarchies.

As an embodied agent, *Solaris 2* is completely integrated within the extended physical system of the human players and the audience, actively perceiving through listening and reacting to them, and participating in the process of *Musicking*, as discussed above. Its ecosystem comprises an AI factory, a gesture manager, and a belief system that provides both world- and self-models, working in a recurrent mode that continually processes its *Umwelt* in real-time, enabling creative collaboration with humans in *musicking*. The concept of *Umwelt* is pivotal. Conceptualised in the realm of biology, it defends that every creature inhabits a world of its own [28]. The structure of this inner world is primarily determined by the species to which a creature belongs, its physiology, behaviour, and environment, but this world discloses itself only through individual subjective experience. As such, these worlds are both private and unique to each subject. *Umwelt* refers to these worlds of subjective experience. Every ecosystem is composed of a combination of dialogue, dispute, and arguments of distinct *Umwelten*, which reinforces the importance of our concept of a symbiotic existence between AI and humans, where artificial intelligence and humans interact in real-time, sharing transparent processes and decisions, with collaboration as a critical concept. This design also takes a collaborative approach that considers the non-human (*Solaris 2*) perspective and *Umwelt*.

One of the primary driving forces behind *Solaris 2* is to challenge and enhance human creativity through an embodied digital score informed by its *Umwelt*, which is shared within the *Musicking* ecosystem. It is within this ecosystem that we can discuss *Solaris 2* exhibiting elements of sentience. The potential for technology to complement rather than replace human experiences and practices, providing new grounds to enable, reimagine, and reorganize how we understand ourselves, how we conceive the meaning of "human", and how we define meaning in new creative practices, working with novel sentient forms of intelligence [21]. The human *Umwelt* needs to be reimagined in terms of how we interact with complex technological systems, reconciling human and machine interactions and enabling new and more advanced forms of autonomy and self-expression in an AI-enhanced world.

[1] "she" refers to a hypothetical person in the original text.

One relatively neglected challenge in ethical AI design is ensuring that AI systems evoke a degree of emotional and moral concern commensurate with their moral standing, as even non-sentient or partially sentient systems can already provoke substantial and intense emotional responses in users [26]. Hence, AI systems should not be morally confusing, meaning that the ethically correct way to treat them should be evident from their design and obvious from their interface. AI systems designed to interact with humans should have their sentience clearly defined and be designed to elicit an appropriate emotional response in their users. Regarding *Solaris 2*, our design is transparent, well-documented and clear. Our moral ambition with it is not to replace humans with machines. As stated before, *Solaris 2* has been designed to collaborate as capable musicians in an ensemble. To do that, we employ, as one of our strategies, a matrix with which to interpret George Russell's Lydian Chromatic Concept of Tonal Organisation [29]. Beyond that, the music it creates is built on the relationships formed in the musicking space.

We believe that the definition of sentient AI needs revaluation to encompass broader forms of intelligence and emotional responses. Sentience is often underestimated in non-human agents [21]. Various modes of feeling transcend human experience, and the term should be accepted as an open-ended way of describing a wide range of aptitudes. Estrada problematizes the actual "AI discourse", which posits both machines and humans as abstract agents with idealized capacities that can be compared and evaluated in absolute terms through abstract measures like "intelligence", advocating for a more pragmatic flexibility in our recognition of artificial agency and sentience, accommodating the myriad ways these artifacts show up in our actual lives [30]. The author suggests that recognising sentient AI fundamentally depends on a shift in our attitudes towards AI agents, confronting our status as mechanical agents, which requires breaking the Western Cartesian way of thought that led to modernity.

4.1. A new take on sentience in creative AI

The views of René Descartes, whose philosophical and scientific beliefs led to the creation of Cartesianism, on sentience outside the human sphere are well known, especially about animals. Descartes believed that animals are just complex living machines, responding to external stimuli in the same way that machines react to specific triggers. In his 1636 Discourse on Method, he contends that even if animals sometimes behave as if they have some desires, feelings, sensations, or other mental states or processes, we should not be deceived by these appearances, as they do not correspond to anything genuinely mental [31].

It was not until the 18th-century Enlightenment that philosophers began to accept the notion that animals have feelings. In the last 50 years, the concept of animals being sentient, or "capable of experiencing positive and negative affective states", has become a topic of great interest to biologists [32].

Descartes developed an ontological thesis about the kind of thing minds are as features of the world and an epistemological thesis about how things of that kind could be known, making a fundamental distinction between minds, which think, and everything else, which is at most a machine. To explain how the human mind inhabits a body, he states that the human body is merely a machine, but made by the hands of God, who unites a rational soul to this machine, advocating a form of dualism in which mind and body are mutually exclusive categories [31]. "Minds" are things that can think, where access to minds can be secured using a faculty known as "introspection", which is a kind of inward perception of a person's mental states [33].

Later, Descartes says that the rational soul is a "thinking thing", something that doubts, understands, affirms, denies, is willing, is unwilling and also imagines and has sensory perceptions [34]. Although Descartes never adequately resolves the question of how perception occurs, he consistently places sentience and consciousness in the category of the mind, which is restricted to the rational soul of human beings.

[19] highlights the fact that debates around AI and sentience have historically taken the early modern period, and Descartes in particular, as a starting point. In this view, a sentient AI is impossible, as it would just be a complex device reacting to external stimuli without being aware of its actions. Alan Turing was aware of these limitations when he introduced what has come to be known as the computational conception of the mind, which inverts the Cartesian account of machines as mindless by turning minds themselves into special kinds of machines, where the boundaries of computability define the boundaries of thought [33].

Hull summarizes the problems with the Cartesian approach to AI:

"Whatever theory of mind underlies current AI work, this Cartesian attitude is persistent in a number of deficiencies in the literature on AI that have been the subject recent of critical attention. It persists in the tendency to look at algorithms in isolation, without paying attention to their deployment in algorithmic systems. It persists in the tendency to treat "fairness" as a formalizable concept that can be operationalized independently of its context. It persists in the isolation of technical academic work on AI from other disciplines that are more concerned with social impacts. And it persists in the quest to determine whether an AI is conscious at the expense of assessing its current socio-political implications. Whether or not an AI can be conscious or sentient in the Cartesian sense may well turn out to be unknowable." [19]

The author concludes that in the face of all this uncertainty, we need an alternative, proposing "the Hobbesian Alternative, replacing the Cartesian question – 'is AI sentient?'– to a more Hobbesian one: 'How should we regulate AI systems that behave as if they are sentient?" [19]. Hull also suggests this alternative based on the assumption that Thomas Hobbes extended a mechanistic treatment of

the body into the mind as well, making Hobbes' mechanistic account of thinking seem initially attractive from an AI standpoint, because it explicitly reduces thinking to computation [19]. Such an alternative is promising, and Hull develops it further in his paper. However, we would like to propose a third approach, more aligned with our theoretical and practical research considering *Solaris 2*.

5. AI AS A TYPE OF MIND

Our proposal asks, "What Sentience in Creative AI is for?" This allows us to explore the idea of creative collaboration between human and artificial minds, the kind of collaboration we observe in *Solaris 2*. To achieve this, we need to understand how AI can be considered a form of mind. We do so based on Charles Sanders Peirce's theory of synechism.

Synechism is the branch of Peirce's metaphysics that deals with continuity, the connection between mind and matter:

"I carry the doctrine [continuity] so far as to maintain that continuity governs the whole domain of experience in every element of it. Accordingly, every proposition, except so far as it relates to an unattainable limit of experience (which I call the Absolute), is to be taken with an indefinite qualification; for a proposition which has no relation whatever to experience is devoid of all meaning" [35].

The philosophical conception that underpins synechism is generous, as it does not separate mind, spirit, and body, matter, breaking with a Cartesian tradition. In the Peircian conception, the human mind differs from other mental forms, such as those observed in AI, in that it is the most plastic reality in the entire known universe, possessing the most significant level of malleability for change [35].

The semiotic field of sign processes, from computers to living systems, has often been analysed in Cartesian dualisms: "tool vs instruments", "instruments vs machines" and above all "machines vs living beings". Rather than confirming these dualisms, Peirce describes this field as a continuum of symbolic processing from the simplest to the most complex. Among the less complex systems are those mediated by instruments or technical devices, such as a thermometer or an automatic traffic system. The most complex semiotic systems occur in living beings. According to synechism, there is no separation or division, but differences of degree between nature and culture, the organic and the inorganic, the natural and the artificial.

Synechism should be understood as continuity between mind and matter, the latter also being a form of mind, but more exhausted and tired, especially if placed next to, for comparison purposes, the human mind. This connatural aspect enables a communicational flow between the various types of minds we observe in the universe, including AI. The mind in artworks allows contemplation or listening to be a dialogue and not just a discourse or narrative to be observed or listened to. Similarly, there is an active mind in *Solaris 2*, a sentient mind that engages in *Musicking*. Synechism is a crucial concept for understanding Peircean metaphysics, based on the pragmatic idea of continuity and evolution, as observed when human musicians play along with *Solaris 2*.

Everything that makes up the universe is a mental form because its physical laws are derived from psychic models, so the great law of the universe is the law of the mind [35]. This law is found in the tendency to generalize and form associations, which is also the tendency to acquire habits, itself a habit. All of reality is governed by the law of mind, that is, the law to acquire habits from the purely physical world to the human mind, with the difference, as we have noted, that the human mind does not submit to the law in the same rigid way that matter does [32]. That is why matter is a mind so locked into habits that it ceases to exhibit the same spontaneous behavior that is abundant in the human mind. This is particularly true for objects that retain their solidified form, such as stones. However, AIs like *Solaris 2* exhibit a very spontaneous and diverse pattern of behavior with remarkable plasticity, resembling the human mind.

Synechism reinforces the possibility of the existence of a creative sensibility in Creative-AI systems like *Solaris 2* because no matter how exact the theories are and how highly predictive they are, the more precise the instruments, techniques, methods and expertise employed by the researcher in the search for truth, the more deviations from the norm will be found, i.e. the more variables and irregularities the object will reveal [35]. Peirce's originality lies in the fact that he attributes this not only to errors of human nature, i.e., imperfections on the part of the observer, but also to a chance present in the object of study, which is alive and pulsating.

Peirce brings us the realization that everything we can know is purely mental since intelligence can only act on what is intelligible. Synechism opens the door to a more comprehensive understanding of sentience in AI. The connaturality between representation and the real object eliminates the Cartesian barrier between subject and object, between consciousness and the world. This free boundary is already present in Peircian phenomenology, as all phenomena, under its prism, pass through interiority – phenomena as they are – and exteriority – appearance, or how we perceive phenomena – in an undifferentiated manner. To understand and utilize the doctrine of synechism, we must admit that if the material and embodied universe is provided with habits of conduct in the form of natural laws, we must conceive of it as a form of mind.

Performing with *Solaris 2* establishes a semiotic way of communicating with the sentient mind that exists in AI.

6. CONCLUSIONS

We present an argument for reconsidering the potential for sentience within a specific context and from a particular perspective. We believe that one way to develop a proper

symbiotic creative ecosystem between AI and human musicians is to meet in the domain where the creativity is happening, namely, inside musicking. Our research appears to indicate some traction here, as both the Creative-AI and the human were embodied, acting in real-time, and able to represent symbols mentally.

Going one step further, we assert, based on Peirce's synechism theory, that current AI constitutes a type of mind. It is this characteristic that makes it possible for AI to communicate with the human mind and for a sentient ecosystem to emerge, especially if the AI is reaching outside its black box and sensing the world through appropriate percepts. Peirce's synechism assumes that continuity always prevails, and the assumption of such continuity is of fundamental methodological importance. Minds are continuous entities, constantly changing and in a state of perpetual flux.

We acknowledge that this is just one experiment and one instance of Creative-AI cooperating symbiotically within a single context. However, due to the openness of our approach and our guiding hypothesis of sentience, we found that – while engaged in musicking – the musicians who participated in the experimental practice seemed to thrive with *Solaris 2*.

7. REFERENCES

1. F. Poltronieri, "Towards a Symbiotic Future: Art and Creative AI," in *The Language of Creative AI*, C. Vear and F. Poltronieri, Eds., in Springer Series on Cultural Computing, Cham: Springer International Publishing, 2022, pp. 29–41.
2. C. Cancino-Chacón, M. Grachten, W. Goebl, G. Widmer, "Computational models of expressive music performance: A comprehensive and critical review", *Frontiers in Digital Humanities*, 5:25, 2018.
3. C. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. Dai, M. Hoffman, M. Dinculescu, D. Eck. *Music transformer*. arXiv preprint arXiv:1809.04281, 2018.
4. C. Small. *Musicking: The Meanings of Performance and Listening.* Wesleyan University Press, 1998.
5. M. Csikszentmihalyi. *Beyond Boredom and Anxiety: Experiencing Flow in Work and Play*. Jossey-Bass, 1975.
6. C. Vear, *The digital score: musicianship, creativity and innovation*. London: Routledge, Taylor & Francis Group, 2019.
7. C. Vear and F. Poltronieri, "Crafting Artificial Musicians: Building Solaris as a Co-operative, Perceiving and Creative AI," in *Artificial Media*, N. Zagalo and D. Keller, Eds., in Springer Series on Cultural Computing, Cham: Springer Nature Switzerland, 2025, pp. 215–239.
8. C. Vear et al., "Building an Embodied Musicking Dataset for Co-creative Music-Making," in *Artificial Intelligence in Music, Sound, Art and Design*, vol. 14633, C. Johnson, S. M. Rebelo, and I. Santos, Eds., in Lecture Notes in Computer Science, vol. 14633., Cham: Springer Nature Switzerland, 2024, pp. 373–388.
9. F. Poltronieri, "Towards sentience: A path through jazz, datasets and digital scores" in *Proceedings of the 29th International Symposium on Electronic Arts. ISEA2024 Everywhen Proceedings. Volume 1: Academic Papers*. Queensland University of Technology and ISEA International, 2025
10. W. Hargreaves. "Generating ideas in Jazz improvisation: Where theory meets practice" *International Journal of Music Education*, 30(4), pp.354-367, 2012.
11. H. Bloom. *The anxiety of influence*. Oxford University Press, 1973.
12. S. Torrance, F. Schumann. "The spur of the moment: What Jazz improvisation tells cognitive science" *AI & Society* 34, pp.251-268, 2019.
13. W. Bowman, "Cognition and the Body: Perspectives from music education" in Liora Bresler (eds) *Knowing Bodies, Moving Minds. Landscapes: The Arts, Aesthetics, and Education, vol. 3.* Springer, 2004.
14. D. Sudnow, *Ways of the hand*. Harvard University Press, 1978.
15. R. Chrisley, "Embodied Artificial Intelligence", *Artificial Intelligence*, 149(1), pp.131-150, 2003.
16. R. Pfeifer, L. Fumiya (eds), *Embodied Artificial Intelligence. International Seminar*. Springer, 2004.
17. R. Pfeifer, "Embodied Artificial Intelligence 10 Years Back, 10 Years Forward" in Reinhard Wilhelm (eds) *Informatics. Lecture Notes in Computer Science*. Springer, 2001.
18. A. Alvero and C. Peña, "AI Sentience and Socioculture,", *Journal of Social Computing,* vol. 4, no. 3, pp. 205–220, Sep. 2023.
19. G. Hull, "Unlearning Descartes: Sentient AI is a Political Problem", *Journal of Social Computing*, vol. 4(3), pp.193-204.
20. H. Proctor, "Animal Sentience: Where Are We and Where Are We Heading?" *Animals,* 2(4), pp.628-639, 2012.
21. Z. Papacharissi, *A networked self and human augmentics, artificial intelligence, sentience*, Routledge, 2019.
22. A. Ducharme, "Aristotle's Mark of Sentience," *Apeiron*, vol. 47, no. 3, Jan. 2014.

23. S. Dehaene, *Consciousness and the brain*. Penguin, 2014.

24. M. Graziano, *Rethinking consciousness*. Norton, 2019.

25. L. Wacquant, "For a sociology of flesh and blood", *Qual. Sociol.,* vol. 38, no. 1, pp.1-11, 2015.

26. E. Schwitzgebel, "AI systems must not confuse users about their sentience or moral status", *Patterns*, vol. 4, no. 8, p. 100818, Aug. 2023.

27. B. Baars, *A Cognitive Theory of Consciousness*. Cambridge University Press, 1988.

28. J. von Uexküll, D. Sagan, and G. Winthrop-Young, *Foray into the worlds of animals and humans: with a theory of meaning*, University of Minnesota Press, 2010.

29. G. Russell, *George Russell's Lydian chromatic concept of tonal organization*, Fourth edition. Brookline, Massachusetts: Concept Publishing Company, 2001.

30. D. Estrada, "AIdeal: Sentience and Ideology," *Journal of Social Computing*, vol. 4, no. 4, pp. 275–325, Dec. 2023.

31. D. René, *Discourse on Method*, Hackett Publishing, 2014.

32. I. Duncan, "The changing concept of animal sentience", *Applied Animal Behavious Science*, 100(1), pp.11-19, 2006.

33. J. Fetzer, "Peirce and the Philosophy of Artificial Intelligence", *The Commens Encyclopedia: The Digital Encyclopedia of Peirce Studies*, 2004.

34. R. Descartes, *The philosophical writings of Descartes*, Cambridge, 1984.

35. C. S. Peirce, *The Essential Peirce, Volume 1: Selected Philosophical Writings (1867-1893),* Indiana University Press, 1992.

OscillaScore: A Modular Platform for Graphic Notation in Networked Music Performance

Dr Rob Canning

Independent Researcher
rc@kiben.net

ABSTRACT

This paper introduces O*scillaScore (https://robcanning.github.io/oscilla)*, a browser-based framework for authoring and performing cue-driven graphic scores using SVG, WebSockets, and OSC. Developed as an open-source project, O*scillaScore* enables synchronised, media-rich performance environments in which animated notation and time-based cues drive ensemble coordination across networked devices. It supports a hybrid approach to score representation, accommodating both fixed-form and open-form works, while allowing composers to embed interactive logic directly into visual elements via a declarative mini-notation system.

OscillaScore's architectural principles, cue syntax, and animation logic—based on XML ID-embedded transformations—are described through the lens of a foundational case study: the 2025 premiere of *Six Inches to the Mile*, a structured comprovisation commissioned by the Stuttgart-based ensemble Pony Says. Developed and rehearsed remotely using OscillaScore, the work served as both the compositional impetus and foundational case study for the system. The score's design included animated graphical elements, embedded performance annotations, and indeterminate behaviors implemented via lightweight browser-side logic.

Positioned at the intersection of composition, performance, and interactive digital notation, OscillaScore supports both fixed and improvised practices through a modular, browser-native platform for cue-driven graphic scores. By embedding animation and cue behaviors directly into visual elements, it enables composers to shape dynamic performance environments without relying on proprietary tools or external scripting. The system's accessibility and adaptability make it suitable for a wide range of contemporary music practices, from solo improvisation to networked ensemble work.

1. INTRODUCTION

Digital score systems have expanded the possibilities for musical notation, enabling dynamic visual elements, real-time interaction, and distributed performance. These tools are particularly valuable in contexts where fixed notation is insufficient—whether due to improvisatory structures, multimedia integration, or the need for synchronised but flexible ensemble coordination. However, many systems remain either overly complex or tightly coupled to specific software environments, limiting accessibility and adaptability.

Recent systems such as INScore [9], the Decibel ScorePlayer [10], and John, the Semi-Conductor [11] have each offered distinctive models for programmable or collaborative scoring environments. OscillaScore builds on this lineage while responding to a need for lightweight deployment, browser-native accessibility, and modular animation and cue logic. It also engages with broader aesthetic considerations around screen-based notation and performer interaction as explored by Hope and Vickery [12].

OscillaScore is a lightweight, browser-based framework for creating and performing cue-driven animated scores using open web technologies. Developed by the author to support both fixed and open-form composition, the system enables composers to embed animation logic, cue triggers, and interpretive annotations directly within SVG files. Playback is coordinated via a central server and synchronised across clients, with optional Open Sound Control (OSC) integration for interfacing with external media systems.

This paper introduces OscillaScore through a technical and artistic account of its development and first use in the composition *1:10,560 (Six Inches to the Mile)*. The discussion situates the system within a comprovisational practice—balancing structure with interpretive freedom—and reflects on the interplay between software development, score design, and performance. A concluding section explores the potential of OscillaScore as a community-driven platform for graphic notation and collaborative experimentation.

2. SYSTEM OVERVIEW

OscillaScore is a browser-based framework for real-time graphic notation, designed to support structured and open-form compositions in ensemble contexts. It enables composers to author scores as animated SVG documents, embedding cue logic and playback behavior directly

within graphic elements using a declarative naming convention. Playback is synchronised across clients via a central server, while Open Sound Control (OSC) provides integration with external audio and media environments.

At its core, OscillaScore uses standard web technologies: SVG for score layout and structure, JavaScript for animation and cue logic, and a Node.js server for timekeeping, synchronisation, and OSC message routing. Scores are authored primarily in Inkscape, allowing composers to work visually while embedding cue functionality via XML ID attributes.

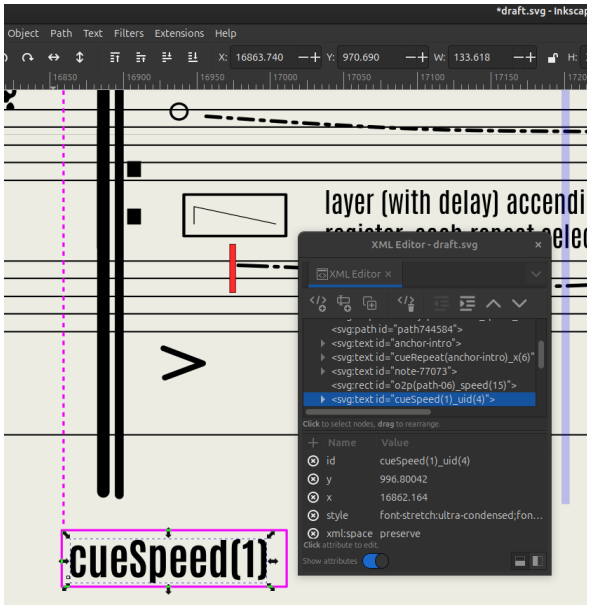


Figure 1: Editing a cueSpeed(1) element in Inkscape's XML editor. This example illustrates how OscillaScore's cue system is embedded directly into the SVG file using microsyntax within element IDs, allowing composers to define behavior declaratively during the visual authoring process.

By building on Inkscape and the wider SVG standard, OscillaScore benefits from an existing open-source ecosystem, avoiding the need to reinvent score editors while ensuring long-term interoperability and community-supported tooling. The system follows a client-server architecture: a single server coordinates playback state and time messages, while each client (typically a browser running on a tablet or laptop) renders and animates the score in real time.

Cues are typically represented as visible text objects arranged horizontally near the bottom of the score timeline. Each cue object displays its own ID, making the structure of the score legible during both authoring and rehearsal. While any SVG element could be used to encode a cue, text elements are preferred by the author for their clarity and ease of selection during authoring and rehearsal. In contrast, graphical elements like ellipses or paths are typically used to animate visual gestures rather than act as triggers. Cue visibility, like annotation visibility, can also be globally toggled if their presence is not beneficial to performers. Visual styling of cues is left to the composer; in some cases, cues may be enclosed in boxes with vertical flags extending through the score to improve visibility for performers reading individual parts.

Cue triggering is handled via a scrolling playhead that continuously checks for intersections with cue objects. When the bounding box of a cue intersects with the playhead's position, the associated action is triggered. On load, the system parses the SVG file and extracts all recognizable elements—cues, animations, rehearsal marks—into internal arrays for fast lookup during playback.

The cue system includes (but is not limited to) **`cuePause()`, cueSpeed(), cueStop(), `cueRepeat()`, `cueAudio()`, `cueMedia()`, `cueTraverse()`, `cueAnimation()`, `cueVideo()`, `cueOSC()`, and `cueChoice()`.** Each cue is defined via a compact ID syntax and may include additional arguments, such as loop counts, resume targets, or OSC payloads. For a complete list of supported cues and usage patterns, see the online documentation (https://github.com/robcanning/oscilla/tree/main/md_docs).

2.1 Animation Logic

Animated behaviors such as scaling, rotation, and guided movement are defined via compact `data-id` or `id` patterns embedded directly in SVG elements. These use a declarative function-style syntax with underscore-separated parameters, supporting step sequences, easing, direction control, randomized values, and cue-based triggering. Example syntaxes include **`rotate(deg(6x90–180)_rpm(5))`** (rotate through six randomly generated angles between 90–180° at 5 RPM), **`s([1.0,1.3,0.9]_seqdur(2))`** (scale through the listed values over a 2-second loop), **`s(rnd(6x0.6–1.4x)_seqdur(3))`** (six randomly generated scale values between 0.6 and 1.4, refreshed on each loop), and **`o2p(curve17)`** (animate an object along the SVG path with ID `curve17` at uniform speed).

These behaviors are parsed at runtime and animated using Anime.js. Easing is fully supported via integer-encoded identifiers (e.g., `ease(3)` for `easeInOutSine`), allowing concise control over animation feel. For path-following animations (`o2p(...)`), an optional `osc(1)` parameter enables the system to emit OSC messages during motion, transmitting the current position or associated metadata at each step. This feature allows external media systems to respond dynamically to visual gestures within the score.

These animation types are not purely visual; they are intended to convey musically interpretable gestures. For

example, scaling sequences can be used to suggest phrasing intensity or entrance timing, while path-following animations can indicate the trajectory of a glissando or dynamic swell. In one composition, a rotating arrow was used to randomly point toward different musical fragments arranged in a circle, signaling which material a performer should interpret next. These mappings and their interpretive implications are explored further in Section 3.

Beyond single-parameter effects, animations in OscillaScore can be layered and nested to create composite gestures. For example, a single object might simultaneously rotate, scale, and follow a path, creating complex temporal trajectories or emergent visual rhythms. These combinations allow composers to shape the interpretive space of a score over time, supporting forms of notation that are not merely illustrative but structurally generative. While animations currently follow fixed timing or randomized behavior, future extensions may explore performer-interactive or data-driven animation control, including integration with live coding environments for real-time manipulation.

2.2 Synchronisation and Playback

OscillaScore is designed to support distributed performance, where multiple performers interact with the same score via networked devices. The playback environment consists of a **Node.js server** (handling timing, playback state, and OSC output), **browser-based clients** (typically tablets or laptops, displaying and animating the SVG score), and a **local or remote file server** hosting the SVG and media assets. A detailed architectural overview is provided in Appendix A.

The score scrolls horizontally at a steady rate unless paused or interrupted by a cue. Manual rehearsal navigation is supported via `jump2rehearsal` cues, allowing performers to skip to specific points in the timeline. The system is designed to handle reconnects gracefully and maintain synchronised timing across devices. Screen resolution differences are managed with responsive layout techniques, although currently uniform device dimensions (e.g. iPads) are preferred in performance contexts.

The player interface includes user-facing controls consistent with common media software conventions. These include play, pause, rewind, jump-to-start, fullscreen toggle, and navigation via rehearsal marks, along with toggle buttons for annotation visibility, OSC output, and WebSocket connectivity. A local audio mute button is provided for rehearsal flexibility. An additional "sync pause" toggle allows users to temporarily suppress network updates and desynchronise their playback view — useful for personal score preview or navigation without affecting others. When re-engaged, the client automatically resynchronises to the current global position via a server message. The system is fully distributed, with no designated master client, allowing any connected device to operate independently if needed.

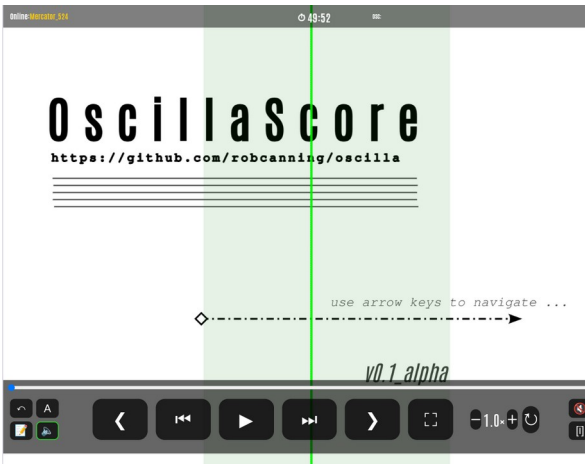


Figure 2: Default OscillaScore help file showing the playback interface, including transport controls and the animated playhead. This built-in guide introduces key concepts such as cue triggering, timeline navigation, and annotation visibility.

In networked performances, a single client can be designated as the audio output device, disabling audio playback on all others to avoid redundancy and conserve system resources. This client has the ability to mute or stop audio cues as needed during rehearsal or live performance.

A real-time stopwatch is displayed prominently (and can be toggled fullscreen), and connected clients are listed for ensemble coordination. While designed for complex animated scores, the system can also operate in a minimal "light mode" — functioning as a synchronised stopwatch with occasional cue displays. This makes OscillaScore a flexible tool for a wide range of musicians, including improvisers working with time-based forms. While iPads or other high-end GPU-accelerated tablets are preferred for complex scores due to their reliable browser performance and rendering capabilities, OscillaScore also runs effectively on low-cost Android tablets and recycled laptops for simpler use cases. This flexibility supports accessible deployment in educational, community, and non-institutional contexts.

3. CASE STUDY: *1:10,560 (SIX INCHES TO THE MILE)*

1:10,560 (Six Inches to the Mile) is a networked digital score for electric guitar, synthesiser, and electronic drums. Commissioned by the Stuttgart-based trio Pony Says with support from the Arts Council of Ireland / An Chomhairle Ealaíon, the work premiered at the 2025 Music Current Festival in Dublin, presented in

cooperation with Dublin Sound Lab (https://musiccurrent.ie/2025/pony-says-concert.html).

The composition explores the metaphor of mapping as both a creative and critical framework. Drawing inspiration from historical Irish Ordnance Survey maps, the score blends cartographic visual language with contemporary digital performance practices. Thematically, the work reflects on how maps structure our perception of space, movement, memory, and identity —both in the physical world and in the navigational language of the score itself.

The score was authored entirely in SVG using Inkscape and scripted media elements, with additional LilyPond-generated notational fragments converted to vector paths. Screenshots of online map details—particularly historical Ordnance Survey imagery—were easily imported and converted to vector paths in Inkscape, allowing the composer to incorporate real cartographic material directly into the graphic score. This fluid workflow enabled rapid integration of visual elements from diverse sources, tightly aligning the score's appearance with its conceptual themes. It was presented on synchronised iPads using the OscillaScore framework, with a local server managing playback state, cue triggering, and animation coordination. Performers engaged with a time-scrolling graphic score featuring animated symbols, path-following indicators, and dynamic visual cues. Object-to-path animations were used to represent glissandi, dynamics, and gestural flows, while rotating and scaling objects suggested energy, directionality, or interpretive density.

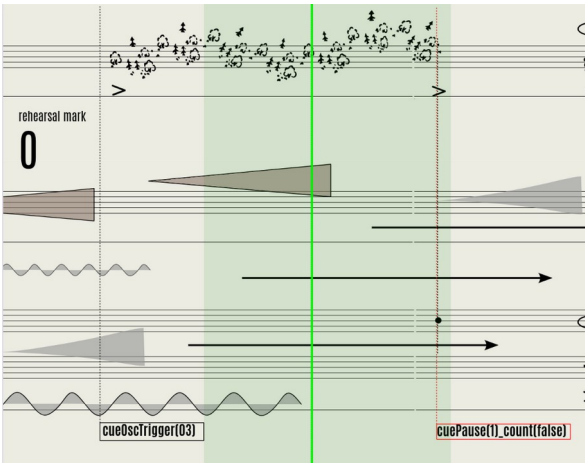


Figure 3: Excerpt from Six Inches to the Mile showing elements of the OscillaScore system in use. Visible are cueOscTrigger(03), used to trigger parameter changes in optional live electronics, and cuePause(1)_count(false), which introduces a short pause without displaying a countdown overlay. The rehearsal mark "O" forms part of the automated navigation system, while animated arrow symbols indicate interpretive direction and timing —though their motion is not visible in this still image.

Structurally, the work is a fixed-form composition with embedded pockets of controlled indeterminacy. Some sequences were rendered stochastically using client-side randomization logic, resulting in non-synchronous fragment playback between devices. Cue-based triggers such as **`cuePause(180)`** invited improvisation within defined zones, while visual countdowns provided temporal clarity. During rehearsal, performers could tap to bypass paused states using `the embedded rehearsal` mark logic, supporting fluid navigation without needing to manually scrub through the score. Annotation layers were toggleable and served as interpretive guides, offering suggested mappings between visual gestures and musical response.

The rehearsal and development process took place over several months, with performers accessing the evolving score via a remote server as features were added and refined. This iterative model allowed the composer to respond quickly to feedback, resolving bugs and implementing new behaviors in real time. The final performance used a local network setup to ensure reliability, with each client fully synchronised but capable of operating independently in case of failure. Notably, the technical workings of the score were never a topic of concern among the performers themselves—an indication that the infrastructure operated smoothly and unobtrusively throughout the rehearsal and performance process.



Figure 4: PonySays trio performing Rob Canning's composition 1:10,560 (6 inches to the Mile), 2025 — intermedia score for electric guitar, synthesiser, and drums — at Dublin Sound Lab's Music Current Festival, Project Arts Centre, Dublin. The musicians performed using iPads synchronised over a local network with Oscilla, while the projector was connected as a fourth client displaying the score to the audience.

4. DISCUSSION AND REFLECTION

The development and use of OscillaScore in *1:10,560* raised several questions and insights beyond the technical implementation. At its heart, the system supports a comprovisational ethos—providing structured, animated guidance while allowing performers freedom to shape timing and detail. The score's combination of constraint and openness created space for interpretation without undermining the work's formal identity. These

compositional values were further reflected in the dual role of composer as system developer, where moments of musical inspiration often emerged from technical problem-solving, and vice versa.

4.1 Comprovisation and Constraint

The compositional approach in *1:10,560* reflects a comprovisational sensibility: the use of structured materials that are open to interpretation, variation, and real-time responsiveness. While the overall form and cue sequence were fixed, performers were given agency in how they responded to visual gestures, managed timing within pauses, and interpreted animations such as scaling, rotation, and motion.

Rather than functioning as a prescriptive timeline or playback script, the score acted as a guide—an interactive surface of suggestion. Performers were encouraged to interpret notational gestures in ways that fit their instrument and idiom. This flexible framework enabled divergent responses while maintaining a shared temporal and structural outline across the ensemble.

In this setting, trust between composer and performer became a foundational element. Because the system handled cue triggering, synchronisation, and transitions automatically, the composer did not need to micromanage detail. Instead, their role shifted toward shaping an environment in which musical decisions could emerge organically from the score's behaviors. This shift in authorship underscores a broader philosophy of graphic score practice: that control can be distributed, and structure does not preclude freedom.

4.2 Composer–Developer Dual Role

The creation of *1:10,560* highlighted both the possibilities and tensions that arise when the composer is also the software developer. The dual role offers an unusual degree of control over both medium and message: musical ideas can lead directly to the invention of technical features, and conversely, technical discoveries can inspire new compositional directions. In this project, specific animation strategies, cue behaviors, and interaction patterns were often developed in response to emerging musical needs—such as the desire to pause, loop, or randomly reorder fragments—and those features now remain available for future works.

At times, however, the separation between composing and coding became blurred or even conflicted. A musical problem might be deferred in favour of solving a technical one, or vice versa. One role could serve as a refuge from the demands of the other, leading to productive cycles but also occasional imbalance. This push and pull shaped the piece’s development: the software did not merely support the score, it *became part of the compositional terrain*.

This experience echoes a wider trend in experimental music and digital art, where composers and artists increasingly create their own tools. Yet it also underscores the need for usable, adaptable systems that do not require deep technical expertise to modify. While the development of OscillaScore remains ongoing, the author recognises the importance of building a broader community of users and contributors who can shape the tool in new directions—without starting from scratch.

4.3 Workshops, Accessibility, and Community Potential

Beyond its use in a commissioned work, OscillaScore has shown promise as a lightweight, adaptable framework for broader artistic and educational contexts. This was tested in a workshop presented by the composer at the Contemporary Music Centre of Ireland during the Music Current Festival 2025. In this session, ten composers each ran a separate OscillaScore instance on shared infrastructure—a single budget laptop and consumer-grade router—demonstrating the system’s ability to support multiple users in parallel with minimal technical overhead. Server processes were isolated by port, and setup scripts allowed rapid deployment.



Figure 5: The author demonstrating OscillaScore to participating composers during a workshop at the Contemporary Music Centre of Ireland, Music Current Festival 2025. Each participant ran an individual instance of the score system on shared local infrastructure.

Participants responded enthusiastically, with many expressing interest in using the system for their own compositions. Several noted that OscillaScore’s architecture—rooted in standard web technologies and modular JavaScript—made it accessible to those with even modest coding experience. The system's clear structure and well-commented codebase encouraged composers to imagine extensions or adaptations aligned with their personal artistic needs.

These responses suggest a strong potential for OscillaScore not just as a performance tool, but as a collaborative, open-source platform. For such a

community to emerge, continued development of documentation, onboarding tools, and example scores will be essential. A future goal is to support a contributor model in which composers and developers alike can share templates, animation modules, cue handlers, and performance strategies. This would strengthen OscillaScore's position not just as a singular system, but as part of a wider ecology of experimental notation and distributed performance practices.

5. CONCLUSION

OscillaScore offers a flexible, browser-based environment for animated graphic scores, supporting a range of compositional approaches from fixed-form structures to open, comprovisational frameworks. Developed in parallel with the composition of *1:10,560 (Six Inches to the Mile)*, the system demonstrates how lightweight, modular web technologies can serve both the compositional process and the practical needs of ensemble rehearsal and performance.

By embedding animation, cue logic, and annotation directly into the SVG score format, OscillaScore enables composers to design interactive, time-based experiences without requiring performers to install complex software or learn new systems. The smooth integration of visual design and playback logic, combined with network synchronisation and OSC support, makes it a versatile tool for contemporary practice.

The author's experience—as both composer and developer—underscored the potential of this approach, while also revealing the need for community engagement, documentation, and extensibility. Early responses from performers and fellow composers suggest that OscillaScore is already enabling new modes of experimentation. As the project continues, its open-source foundation invites others to adapt, extend, and contribute to a growing ecosystem of graphic score creation and performance. Future development will include support for live-codable animations and OSC-reactive behaviors, enabling real-time score manipulation and expanded performative interaction.

OscillaScore is free and open-source, with a pre-release version currently available on GitHub. A stable public release is forthcoming, marking the next milestone in its development.

Acknowledgments

The author gratefully acknowledges the Arts Council of Ireland / An Chomhairle Ealaíon for funding the commission of *1:10,560 (Six Inches to the Mile)*, and the Stuttgart-based trio Pony Says for commissioning and premiering the work. Thanks also to Fergal Dowling and Dublin Sound Lab for organising the Music Current Festival, and to the Contemporary Music Centre of Ireland for supporting the development and dissemination of this project. Technical thanks go to the Ljudmila Art and Science Laboratory (Ljubljana) for server infrastructure during development—building on ideas first explored during an artist residency there in 2010—with special thanks to Luka Frelih for his support. Appreciation is also extended to the kompot.si community for hosting the open-source repository.

6. REFERENCES

1. R. Canning, "Interactive parallax scrolling score interface for composed networked improvisation," in *Proc. Int. Conf. New Interfaces for Musical Expression (NIME)*, 2014, pp. 144–146.

2. R. Canning, "Real-time web technologies in the networked performance environment," in *Proc. Int. Computer Music Conf. (ICMC)*, Ann Arbor, MI, 2012.

3. R. Canning, "The M2 Compositions: A technical overview of a modular workflow towards the creation of a video score," in *Proc. Int. Computer Music Conf. (ICMC)*, Huddersfield, UK, 2011.

4. R. Canning, O*scillaScore: Browser-based animated graphic score system*, Open source repository, 2025. [Online]. Available: https://github.com/robcanning/oscilla

5. R. Canning, *Parallaxis: A networked scrolling score system for composed improvisation*, Performed at the Linux Audio Conference (LAC), Graz, Austria, May 10, 2013. [Online]. Available: https://archive.org/details/ParallaxisForFourInstrumentsAndElectronics

6. R. Canning, *Parallaxis Score System*, Open source repository, 2013. [Online]. Available: https://git.kompot.si/rob/parallaxis

7. R. Canning, *Desire Lines for Four Violins*, digital score composition and performance. [Online]. Available: https://archive.org/details/DesireLinesForFourViolins

8. R. Canning, *Pathways for Operation Integratus*, digital score composition and performance. [Online]. Available: https://archive.org/details/PathwaysForOperationIntegratus

9. D. Fober, Y. Orlarey, and S. Letz, “INScore: An Environment for the Design of Live Music Scores,” in *Proc. Linux Audio Conf. (LAC)*, 2012.

10. L. Hope, A. Vickery, and C. Furneaux, “The Decibel ScorePlayer—A Digital Tool for Reading Graphic Notation,” in *Proc. Int. Computer Music Conf. (ICMC)*, 2015.

11. V. Goudard, “John, the Semi-Conductor: A Tool for Comprovisation,” *arXiv preprint*, 2018. [Online]. Available: https://arxiv.org/abs/1811.06858

12. C. Hope and L. Vickery, “Screen Scores: New Media Music Manuscripts,” in *Proc. Int. Computer Music Conf. (ICMC)*, Huddersfield, UK, 2011.

6. APPENDICES

Appendix A – System Architecture Diagram

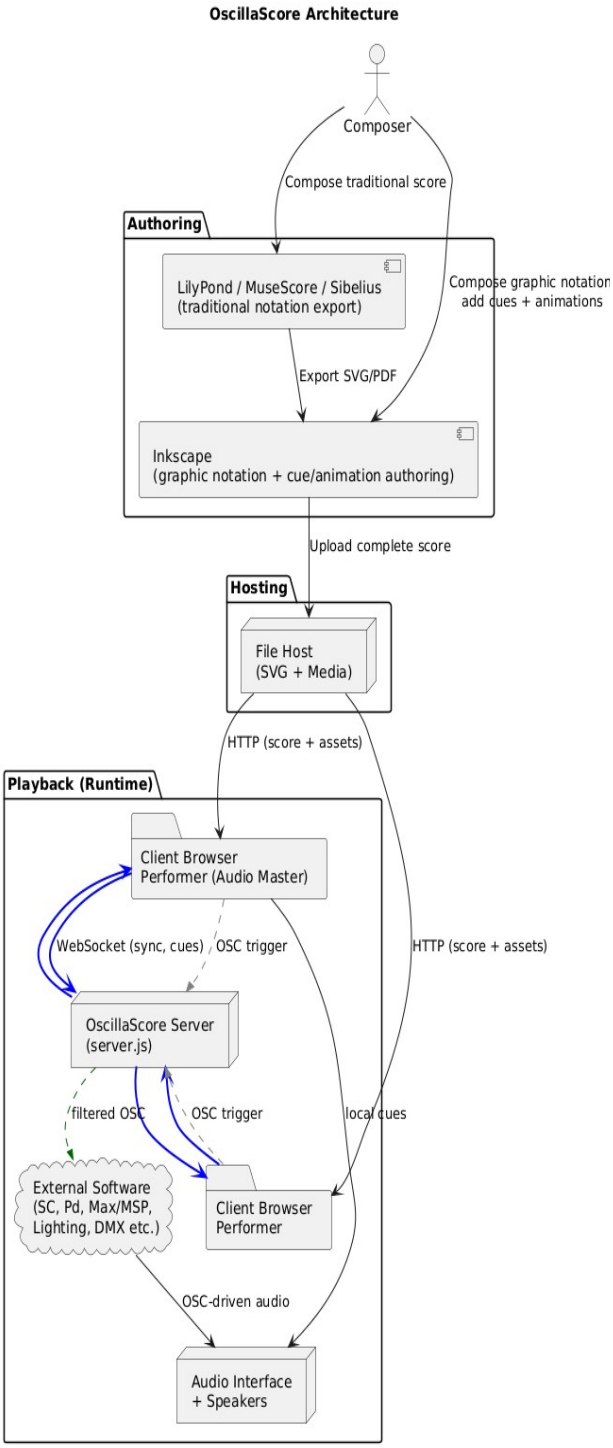


Figure 6 This diagram illustrates the full authoring-to-performance pipeline of the OscillaScore system. Composers create scores in Inkscape, embedding cue and animation logic directly into SVG element IDs. Traditional notation exported from software like LilyPond, MuseScore, or Sibelius may optionally be imported into the SVG layout to coexist with graphical material. The completed score and any associated media are uploaded to a file host, which serves them to performer clients via standard HTTP. During playback, browser-based clients run OscillaScore and synchronize with a central Node.js server via WebSockets, sharing timing, cue triggers, and state updates in real time. OSC messages are centrally routed through the server to external systems tenor such as SuperCollider, Pure Data, lighting rigs, video playback engines, or electromechanical devices. A designated audio master client is responsible for triggering audio playback—either using native browser technologies for simple fixed media, or via OSC to external audio software for more complex setups. This modular, synchronized architecture supports flexible, distributed performance with minimal technical overhead.

Interpretation and Learning of Modular Synthesis: Three Studies

Pierre-Luc Lecours

Université de Montréal
CIRMMT student member
pierre-Luc.lecours@umon-treal.ca

Maryse Gagnon-Legault

McGill University
CIRMMT student member
maryse.gagnon-legault@mail.mcgill.ca

ABSTRACT

This article presents the creation of three progressive studies for modular synthesizer, composed by Pierre-Luc Lecours and interpreted by Maryse Gagnon-Legault. The project aims to facilitate the learning process associated with modular synthesis as well has contributing to the instrument's repertoire. Through a dialogue between composition and performance, the authors address challenges related to musical notation and the reproducibility of synthesis systems across different setups. This paper outlines various historical approaches to synthesizer notation and describes the notational strategies employed in these studies.

1. INTRODUCTION

This paper describes and discusses the creation of three progressive studies for modular synthesiser. These studies were developed in collaboration between composer Pierre-Luc Lecours and modular synthesiser performer Maryse Legault. This initiative is rooted in the desire to contribute to the formalisation of the modular synthesiser interpretation and learning process. We believe these studies can offer a starting point for beginner and intermediate modular synthesis performer wishing to develop new performing skills on their instruments.

We define the modular sound synthesis performer according to the characteristics established by Lecours and Bernier (2024) [1]: "…we define the modular sound synthesis performer as a specialist in the interpretation of a repertoire linked to any electronic or network of electronic instruments producing synthesised sounds which must be programmed prior to their use and whose parameters (pitch, rhythm, and timbre) can be controlled manually or automated with voltage and/or digital control automations. These synthesis instruments can produce sounds from both analogue and digital circuits, excluding sounds from conventional acoustic instruments. This definition therefore also excludes electronic instruments whose synthesis systems are pre-programmed (presets-based) and whose interpretation parameters are mainly pitches and their intensities (e.g., Yamaha's DX7 or SY-77 synthesisers). This definition would typically involve building a synthesis system to meet the interpretation needs of the piece to be played." This research-creation project arose from an interrogation both authors had: is it possible to design musical studies intended to be performed by a modular synthesiser performer? The answer is not simple, and we will discuss our findings in this paper.

Copyright : © 2025 Pierre-Luc Lecours and Maryse Gagnon-Legault. This is an open-access article distributed under the terms of the Creative Commons Attribution License 3.0 Unported, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

The literature or creative project revolving around learning process and interpretation of modular synthesis is relatively scarce. However, there are educators working actively to make this instrument more accessible as well as instruments system specific study. This project is part of this effort to make the initial learning stages of this often-challenging instrument more accessible.

The collaborative work on this project was structured as a dialogue between Lecours' compositional work, who initially created sketches of the studies, and Legault's score analysis and interpretation work, who tested the studies and suggested adjustments based on encountered difficulties.

In this article, several approaches to patch synthesis notation that have been realised in the past will be discussed. In the following sections, we will present different patch notation initiative that had been realized in the past, the three studies composed for this project, analyse them, and describe the collaborative work between composer Lecours and performer Legault. We will then discuss theoretical and practical reflexions that has emerged from this project.

2. MODULAR SYNTHESIZER NOTATION: HISTORICAL CONTEXT

The impetus to crystalize ideas through some form of notation appeared quite early in the existence of the modular synthesizer. Whether it was to take note of successful patches to ensure reproducibility in future performances, or for pedagogical purposes, modular synthesis notation existed in some form or the other since at least the 1960s.

Some early manifestations that guided our methodological process must be underlined.

A precursor in its genre, Allen Strange's book *Electronic music: systems, techniques, and controls* [2], is still today appraised for its forward-looking vision and the relevance of its educational content.[1] The scientific approach of the volume helps both the performer and the composer to understand the mechanisms of synthesis abstractly, outside of any connection with specific gear or modules.

In this book, Strange focusses on the basic types of sound sources, the different kinds of voltage controls one can use to shape these sounds, and other techniques useful to modular synthesis. If Strange includes a variety of graphs, diagrams and visual representations of each subtractive and additive technique broached in the book, he does not specifically approach notation in performative way. However, he solidly places the presence of the synthesizer in the Western Classical music landscape, mentioning parallels between the wide sonic palette of the synthesizer and the timbral aspirations of twentieth-century composers such as Maurice Ravel and Paul Hindemith. In this perspective, Strange raises important questions that guided our creative undertaking: "What are the processes the 'electronic composer' follows, then, to make a composition? What are the parameters involved in the production of electronic music? What are the available techniques and how do they work?" [2].

At the end of the book, he includes a chapter dedicated to performance instructions for various electronic pieces including Douglas Leedy's *Entropical Paradise with Bird Call* (1969), conceived for the Buchla System. He gives the performer a page-long list of instructions, including the position of the knobs, as well as a diagram for the performance of the piece.

A few years after the publication of the first edition of Strange's *Electronic Music*, Suzanne Ciani produced a grant report in which she describes what she calls a "Basic Performance Patch" for the Buchla Series 200 instruments [3]. In this account, Ciani uses a module-specific notation to describe the patches themselves and provides indications to perform with the patches and the transitions from one performance patch to the next.

At the end of her introduction, she underlines that she finds "that the best performances combine the competence of pre-planned and well-rehearsed playing with the magic of being able to follow one's inspiration when inspired by the audience and the moment. To do the latter a performer must be familiar with his patch to the point of not having to "think twice" (at least not more than once) about what effect or series of consequences will be produced by a given action." [3]

The core of Ciani's report is a prosaic description of her patches, along with the reasoning behind her choices. In the second part of the essay, she includes diagrams showing the routing between every module of her system (figure 1). Noteworthy is how she details the behaviour of certain patches (such as the "vertical sequencer" and "keyboard rotations") using traditional western classical notation (figure 2), resembling some examples from Strange's book.

Certain visual similarities can be observed between Suzanne Ciani's Basic Performance Patch notation (figure 1) and the one proposed by Allen Strange in various sections of his book. However, a key distinction lies in the fact that, in Ciani's case, the operators refer directly to specific modules of the Buchla 200 system. This level of precision lends her notation particular value in terms of reproducibility: it enables anyone with access to a Buchla 200 to faithfully reconfigure the original patch. This system can therefore be used to interpret the musical ideas developed in the document, such as *Keyboard Rotations*, *Melodic-Rhythmic Reliefs*, *Vertical Sequencer*, and *String Patch*.

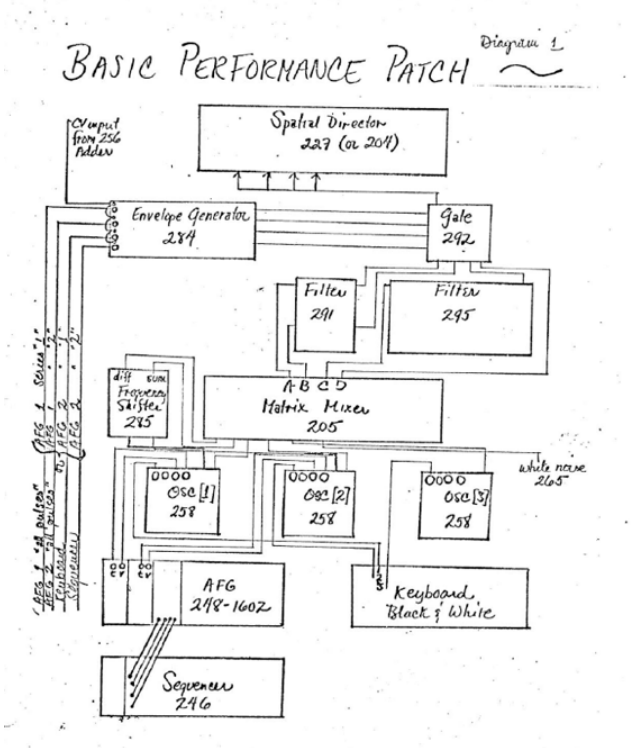


Figure 1. Ciani's diagram for her "Basic Performance Patch" from Report to National Endowment © Suzanne Ciani

An important pioneer in modular synthesis notation have raised the many challenges posited by notation. In an article in the journal *Synapse*, published in 1978, Richard Bugg explains that "in attempting to maintain some semblance of reality in notation of synthesizer patches, it soon becomes apparent that if there is no method to the madness, the former will be lost to the latter. Some form of permanent, or at least semi-permanent, recording of information is needed to allow the reconstruction of a specific sound." [4]

[1] A third edition of the book has recently been published.
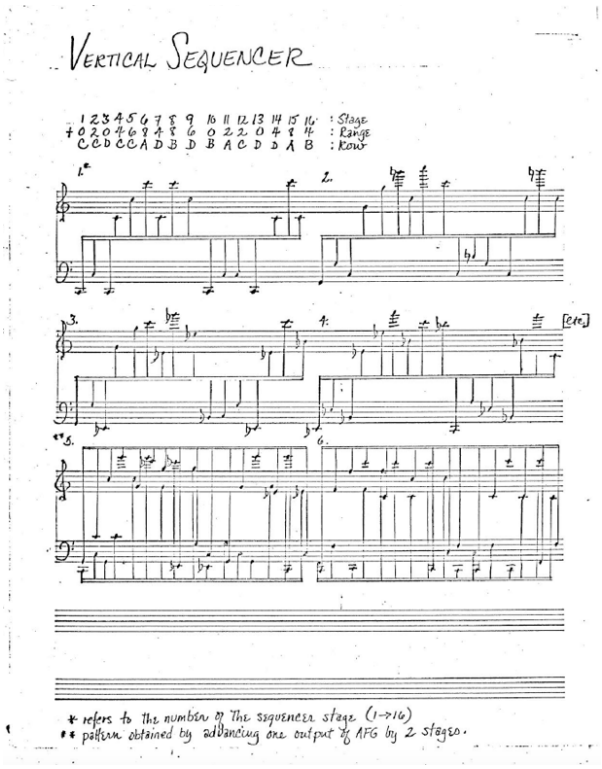


Figure 2. Ciani's notation for the "Vertical Sequencer" from *Report to National Endowment* © Suzanne Ciani.

Bugg's interest in notation is linked with reproducibility. He finds in what he calls the "FRONT PANEL FACSIMILE" notation, developed by makers of small systems, such as the Minimoog and the ODYSSEY, many advantages. This notation, focusing exclusively on a specific system, has the advantage of giving clear, easily reproducible, commands. However, this type of dashboard-focused notation sets limitations when one tries to apply the same commands to a different system, be another set of modules, or simply a bigger instrument.

Another type of notation Bugg mentions in the article is one exclusively focused on modules. Aspiring to be a more universal and widely applicable method, Strange points out some limitations, namely that "several operations may be performed in one module on one type of instrument. The same operations may be in several modules or non-existent on a different type of instrument." [4] Indeed, the plasticity of each module to suit different compositional needs is one of the most attractive advantages of modular synthesis performance, and using such a type of notation has significant creative restrictions.

To offset, these various issues, Bugg suggests focussing on the three general roles modules in any given instrument:

1. Signal sources (oscillators, but also noise generators, ringing filters, other types of modules used as signal sources),
2. Modifiers (envelopes, filters, amplifiers, etc.),
3. Controllers (keyboards, touch pads, or any other module triggering a command).

By focussing on these three groups of modules, we can indeed transpose the patch to different systems and insure a minimum of reproducibility.

The approach we adopted for the notation of the studies scores is based on Lecours's modular synthesis notational approach he developed in doctoral studies which is derived from various methods presented in this section, but more specifically the one established by Bjørn and Meyer (2018) in the book Patch & Tweak [5].

In their book, the authors propose a synthesis system notation based on a series of graphic symbols representing four main categories of operators that can constitute a modular synthesizer patch. Each category is distinguished by a geometric shape containing the symbol, allowing for quick identification of its function. The four categories are audio sources (circle), audio modifiers (triangle), control voltage sources (square), and control voltage modifiers (diamond). The notation of patches is constructed by connecting the different operators (symbols) of the system, following a left-to-right direction for audio and bottom-to-top for control voltage, gates, and triggers. A small textual note can be added to indicate where to connect the cable. The colour of the cables indicates the nature of the signal being transported (yellow = audio, grey = 1V/oct, blue = other control voltage, red = triggers and gates, green = master clocks). Figure 3 shows the implementation of a simple patch using the system proposed by Bjørn and Meyer.

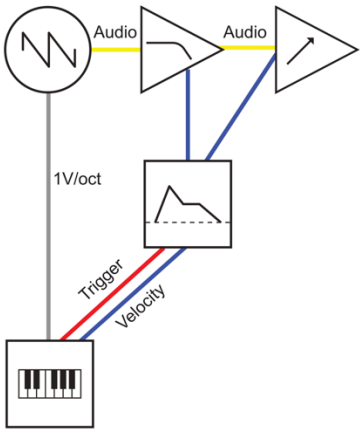


Figure 3. Filtered sawtooth wave patch, with pitch and envelope controlled via keyboard. This example is reproduced from the Patch & Tweak book using the authors' patch symbol library. See the Acknowledgments section for patch symbol licensing information.

We found this patch notation system to be the most comprehensive to date, drawing inspiration from several of the systems described above and others (Strange, Bugg, and Beaver & Krause).

3. THREE STUDIES: THE GENESIS

The project to create studies for modular synthesiser arose in response to several observations. The first is related to the absence of a specific repertoire for the instrument. This instrument is often used in concerts in improvisational contexts or by composers who perform their own pieces. Thus, musicians dedicated to interpreting written music for modular synthesiser are extremely rare. There is also very little repertoire specifically written for the instrument.

There are musicians like Luis Codera Puzo, who compose piece for his specific modular synthesizer instrument and who has created a virtuosic study for modular synth call *Plétora*. The study permits interesting and virtuosos sound manipulation, however, to play the piece, a musician needs to have a similar modular system to the one Puzo used to create it. The indications in the score are module-specific oriented and prescriptive in the parameter-notation approach.

The second observation is the need for pedagogical tools to facilitate the learning of modular synthesis. In this regard, there are very few resources available for learning and playing pieces on this instrument. This highlights an additional difficulty for those who wish to learn to play and interpret music with this instrument. Therefore, this project also aims to address this issue by proposing studies that can be used in a learning and teaching context. This aspect will be discussed further in the text.

3.1. Three Studies for Modular Synthesiser

The three studies for modular synthesiser were created to allow for a progression in the difficulty of execution and patch creation, as well as the exploration of three approaches to sound synthesis and interaction with the instrument. The notation of the studies will be detailed in each of their respective sections of this paper. However, there are some common aspects that can be observed among them.

Firstly, each score consists of two distinct sections. The first section provides instructions for creating the playing system on the modular synthesiser. The instructions are divided into voices, each with its own patch construction guidelines and the parameters that need to be manipulated within each voice (Figure 4).

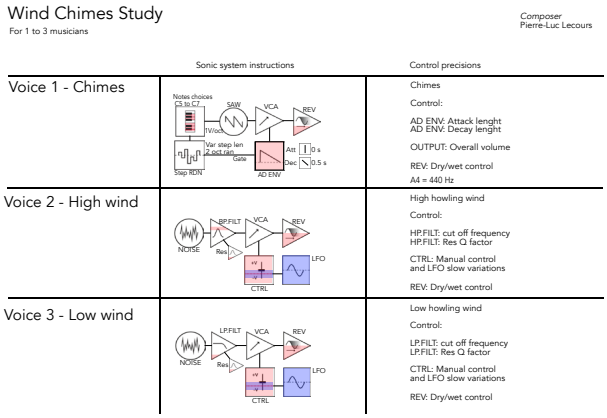


Figure 4. Instruction page for the construction of the *Wind Chimes Study* synthesis system. © Pierre-Luc Lecours

The instrument patch is divided into different voices to facilitate the notation and reading of the various parameters to be performed, and to aid in the visualisation and musical understanding of each voice. This division by voices can also allow for the interpretation of the pieces in small ensembles.

The instructions for creating the patch use the symbols proposed by Bjørn and Meyer in their book Patch & Tweak. Their notational approach has been modified to better suit the needs of the studies. The meaning of the symbols and the reference to categories by the shape containing the symbol (audio sources (circle), audio modifiers (triangle), control voltage sources (square), and control voltage modifiers (diamond)) have been maintained. However, we have abandoned the coloured connections indicating the type of information to be transmitted from one operator to another in favour of a textual indication attached to the cable. The instructions are generally read from left to right, but not necessarily from bottom to top for control voltage indications (unlike the approach proposed by Bjørn and Meyer). An internal colour-coding system has been added to indicate parameter activation in pink and automation in purple.

The writing of the notation systems was approached with an abstract systemic approach [2]: "[…] notation by seeking to abstract synthesis concepts rather than represent them in their entirety. This abstraction takes the form of a network of interconnected icons designed to represent the structural logic of the system. These icons highlight the fundamental units of the system's construction and the parameters that need manipulation […]. Additionally, textual indications complement this iconographic representation, providing performers with interpretative freedom."

This abstract systemic approach allows musicians to build a system using tools different from those used by the composer and to choose how they will perform the indicated manipulations. This approach also inevitably involves a simplification of the patch necessary for interpretation. In the sense that it is a notation that represents the concepts of the patch to be realised rather than the actual connections on the musician's instrument. The musician will thus have to find idiomatic solutions for their instrument in response to the indications proposed by the score.

For the performance instructions, the approach used is derived from those described for *Poussière de Soleil* by Lecours and Bernier [1]. Each voice in the study has a dedicated section indicating the manipulations to be executed (Figure 5).

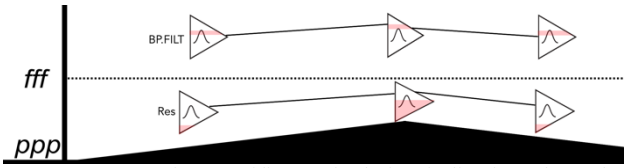


Figure 5. Voice specific manipulation indications from the studies. © Pierre-Luc Lecours

In this example, the lower part with the black triangular shape represents the intensity from ppp to fff; the performer must execute a crescendo followed by a decrescendo. The upper part shows manipulations to be made on a bandpass filter, which should move from a medium-high to a high register while increasing its resonance and then return to their original values.

In other cases, a note to be played might be indicated (Figure 6). In this example, the performer must play a note close to E1 using a controller.

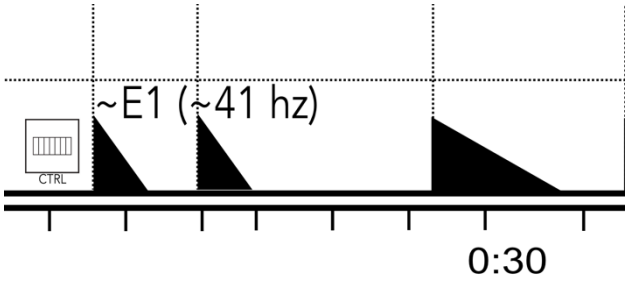


Figure 6. Note indications to play example from the studies. © Pierre-Luc Lecours

These indication aims to allow the creation of an initial multi-voice synthesis system, allowing the performer to manage the resources of their synthesiser to play the piece. As such, the patch and manipulation instructions are always a starting point on which the musician can build and develop. Indeed, we consider it part of the performance practice of modular synthesiser performers to develop and enrich a given patch to approach a sonic and/or timbral aesthetic idea. Orning [3] discusses this concerning Lachenmann performance practice, but the concept applies to any repertoire: "Performance practice has been characterised as 'a term borrowed from the German 19th-century Aufführungspraxis to describe the mechanics of a performance that define its style' (Parrott and de Costa 2011a). Within 'mechanics of a performance' I include knowledge, know-how, and aesthetic preferences on an individual level as well as in communities of practice."

It must be also noted that some of the graphic approach used by Lecours in these scores come from Bernier work with his *Ensemble d'oscillateurs* [4] and his oscillator score notation system.

3.1.1. Wind Chimes Study *no.1*

Wind Chimes Study is a piece where the musician performs a soundscape consisting of a high-pitched wind, a low-pitched wind, and a wind chime.

The two wind voices are relatively easy to create; they essentially consist of filtered white noise with volume control and slight reverb dry/wet modifications. The chimes voice requires a bit more resource to create from the synthesiser. It involves randomly sending notes into a sawtooth wave and triggering gates randomly in the envelope that allows the sound to pass. This voice can be relatively easy to create with certain modules like Make Noise's Wogglebug, but more complex in other cases. It is also necessary to quantise the random pitches to obtain the five tempered notes indicated in the score. Additionally, the ability to vary the speed of the random triggers is required. For the interpretation of the study, the first manipulations involve simulating the sound of the low-pitched wind, followed by the high-pitched wind using filter and volume adjustments (Figure 7).

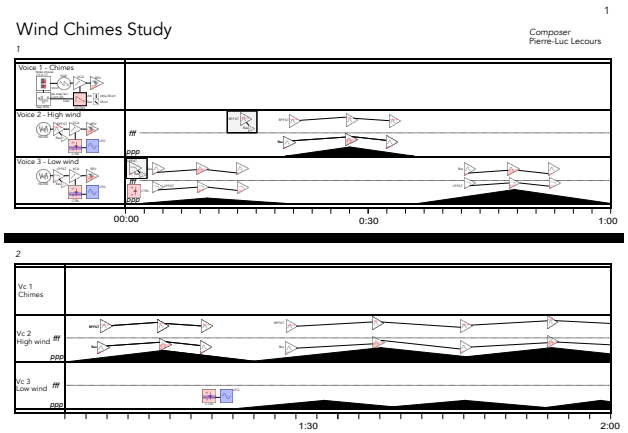


Figure 7. First page of the study *Wind Chimes Study*. © Pierre-Luc Lecours

An automation of the low wind volume occurs around 1m20. The slow volume variation is achieved using a sine wave LFO.

From 2m15, the wind chimes gradually emerge, with manipulations in this section primarily involving volume variation and changes to the envelope release of the chimes. The low wind transforms into a low-pitched sawtooth wave drone towards the end of the piece. The intended duration for the performance of the piece is 4 minutes.

The musician should approach this study creating and playing each voice separately at first and then play them altogether.

3.1.2. Arpeggios and Melody Study *no.2*

Arpeggios and Melody Study explores timbral manipulations on repeated sequences of arpeggio, bass, and a random melodic voice. Rooted in the early aesthetic work of Suzanne Ciani and Caterina Barbieri, this study allows the musician to become familiar with setting up multi-voice melodic and rhythmic systems and to explore an expressive vocabulary through timbre modifications.

Creating the synthesis patch presents certain challenges (Figure 8), particularly the sequencing of two voices (voices 2 and 3) that have a different number of steps. To successfully perform the study, the synthesiser must include three sawtooth wave oscillators (one per voice and could be replaced if needed by a square wave), three envelope generators, and three VCAs, a master clock that generates triggers, a low-pass filter, a reverb, and a delay (master clock synced).

Voice 3 can be played with a keyboard or controller to allow manual note changes. This provides an alternative if the musician cannot sequence a second voice with their system.

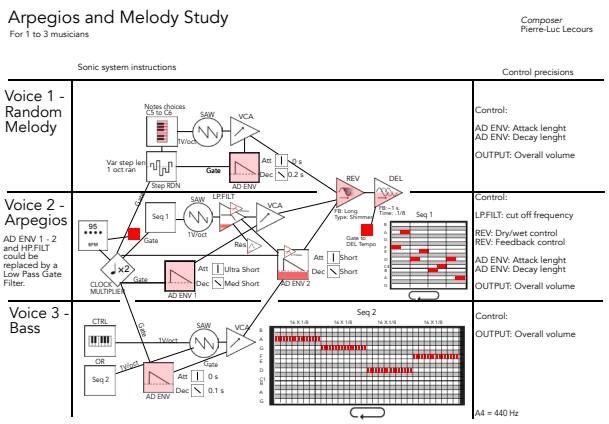


Figure 8. Instruction page for the construction of the synthesis system for the *Arpeggios and Melody Study*. © Pierre-Luc Lecours

Once the synthesis patch is set up, the performer's manipulations focus on changes of voice 1 and 2 envelope release, dry/wet reverb mix on the sound, and the volume levels of each voice (Figure 9).

The challenge of interpreting this study is to make parameter changes smoothly while staying highly focused on the effects these manipulations produce.

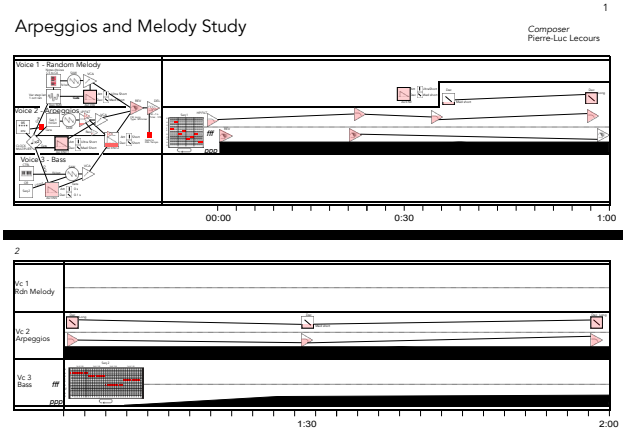


Figure 9. First page of *Arpeggios and Melody Study*. © Pierre-Luc Lecours

*3.1.3.*Noise, Rhythm and Articulation Study *no.3*

Noise, Rhythm and Articulation Study explore sound articulation, rhythm, and noisy sounds. This study presents several challenges in both patch design and the manipulations required to perform the piece.

For the realisation of this study, it is necessary to have controllers capable of sending triggers/gates and control voltage values. It is required to have a few envelope generators, an oscillator capable of FM synthesis, and another oscillator to create a sound similar to an electronic kick (Figure 10).

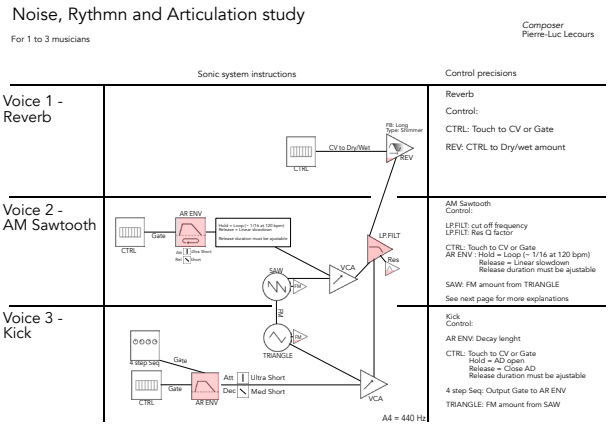


Figure 10. First page of instructions for the construction of the synthesis system for the *Noise, Rhythm and Articulation Study*. © Pierre-Luc Lecours

The first voice involves activating the dry/wet parameter of the reverb using a control surface. The second voice presents certain challenges, as it requires an iterative sound to be triggered by a controller and for the iteration to gradually slow down. There are several ways to achieve this, but a solution is specified on the second page of instructions for the patch construction (Figure 11).

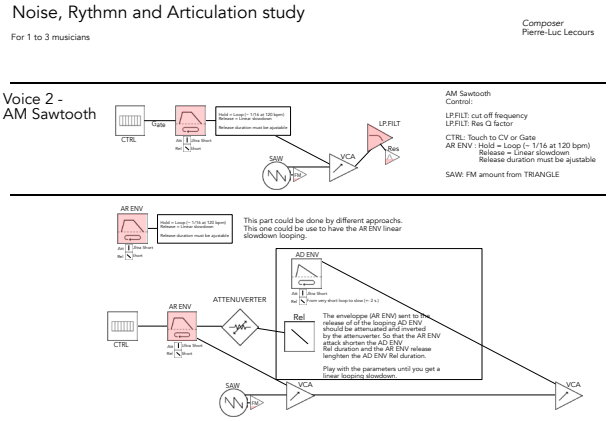


Figure 11. Second page of instructions for the construction of the second voice of the synthesis system for the *Noise, Rhythm and Articulation Study*. © Pierre-Luc Lecours

The third voice is an electronic kick, which needs to be controlled alternatively by a 4-step sequencer and a controller. The *Control Precisions* section (Figure 10) provides more details on how to configure each voice to be able to play the piece.

Interpreting this study focus on direct interaction with the sound material. The piece begins with the slowing down of noisy and low-pitched iterations (Figure 12). These interventions vary in length, so it is necessary to be able to adjust the duration of the slowing down. Then, the slowing down combines with a very low kick (~41 Hz), which is sometimes modified by a long reverb. When activated, the reverb should create a wall of sound.

At 1m30, the kick is controlled by a sequencer that plays every two pulses at a speed of 100 bpm. This section requires the performer to play the slowing down iteration (voice2) and reverb triggers (voice 1) on certain beats

while changing the pitch of the oscillator. It allows the performer to practice syncing up rhythmic intervention with a clocked process on the modular synthesizer.

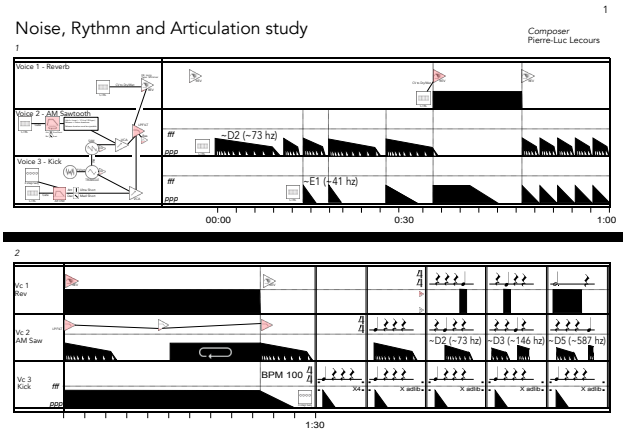


Figure 12. First page of *Noise, Rhythm and Articulation Study*. © Pierre-Luc Lecours

This third study thus explores other paradigms of synthesis interpretation. We will return to certain aspects of the interpretation of these studies in the discussion section of the article.

4. THE PROCESS OF INTERPRETATING THE STUDIES.

The composer-interpret dynamic has evolved in several steps and processes that will be described in this section. One of the first steps, at the core of the project, was the development of an appropriate system by the performer. In order to celebrate the tradition of modular synthesis performance, the performer constructed her own synthesizer by choosing modules suited for the three studies, under the guidance of composer Lecours. Building the synthesizer helped the performer to have a closer relationship with her instrument and to understand better the function of each module. The system (figure 13) that was used for the apprenticeship of the studies was designed to be portable, and as minimalist as possible, while allowing for a wide range of artistic expression.

Before composing the studies and producing scores, the collaborators divided the target competencies of the studies into general categories to judge how to organize them in a progressive manner, thus identifying the different skills to develop through the performance of each piece.

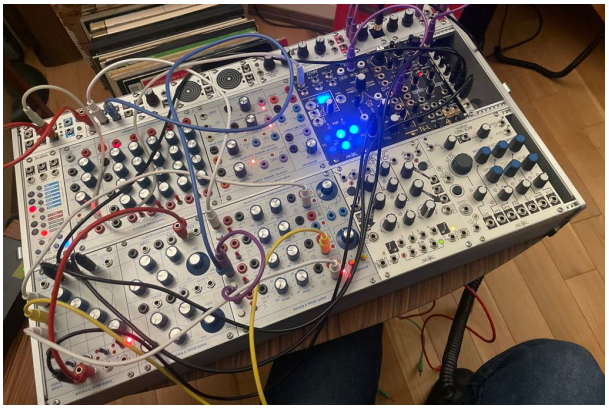


Figure 13. Legault's modular synthesiser. © Maryse Legault

During the first meeting, a first exploration of the instrument was made by the performer as directed by the composer, who gave inceptive oral directions in the context of an overview of each module of the system. Once the instrument was more familiar to Legault, it was possible to start working on the studies themselves. During a subsequent meeting, Lecours submitted to Legault a first draft of the first study, for which Legault imagined an appropriate patch. Benefiting from real-time feedback, the meeting allowed for quick adjustment of the score to clarify the intentions of the composer, but also to test the feasibility of the piece for a beginner. Some of the issues encountered were the lack of transparency of the symbols, the amount of different commands to perform in a short lapse of time for an unexperienced performer, and the insufficient details in the directives to help build an effective patch. Lecours was able to produce a second, then a third version of the first study thanks to the feedback of Legault, but also by observing the limitations and struggles of the performer while reading the score and performing the study. A part of each workshop was dedicated to guiding the performer into creating the most effective patch, so that her personal practice included not only the performance of the score, but also the conception of the patch. A similar approach was done for the completion of studies 2 and 3.

Once every study was finished and mastered, Lecours and Legault tried to transpose the performance on different systems. The studies were thus played on Legault's personal synthesizer, but also on a *Synthesiser.com* custom *Studio-66* Moog inspired system, a *Shared System* by Make Noise, and on Lecours's personal system. The type of comprehension generated by the function-focussed notation allowed the performer to be able to translate the score easily to a new system, once she understood the role of each newly encountered module.

At the end of the project, and after less than a year of modular synthesizer apprenticeship, Legault performed the three studies at the LFestival d'Ondes in Montreal, in 2024. Because of the impossibility to repatch between each piece, the performer used two connected synthesizers. However, it should be underlined that each study had been entirely patched and played on Legault's personal system.

5. DISCUSSION

The process of creating and interpreting these studies has generated numerous reflections, which we will discuss in this section. To begin, we will cover aspects related to the creation of the studies, particularly those concerning notation. Then, we will address the challenges related to their interpretation and finally lack of literature around the subject.

5.1. Writing studies for Modular Synthesizer

The writing of the three studies for modular synthesiser raised several issues. The first is the question of the system required for their interpretation. We discussed this aspect in previous sections, but the issue of transferring instruc-

tions from one system to another was central to the creation of the studies and also at the heart of several previous notation initiatives presented in the historical context.

Despite the work on inter-system transferability considered during the writing of the studies, they still require from the musician to have access to a modular system configuration oriented towards the production of synthesised sounds with certain effect modules (delay and reverb).

While it is necessary to establish a specific base of operators for the interpretation of the studies and to explore certain specifics of synthesis and its interpretation, these choices could limit access to certain studies for musicians with modular systems designed differently.

5.2. Patch creation for the studies

Despite the simplicity of most synthesis systems when approached one voice at a time, combining multiple voices can become complicated to coexist on a single system. Indeed, while working on the studies, Legault and Lecours sometimes had to devise complex solutions to successfully integrate all three voices of each study, occasionally requiring the use of advanced functions of the modules used. This could make their adoption by a broader group of musicians prohibitive, as some performers might find the difficulty too great.

That been said, the aspect of the systemic affordability of a given modular system is intrinsic to the practice of this instrument. We believe it is important to be accustomed to it at every stage of learning modular synthesis. When preparing the performance of the studies in concert, Legault could not simply reproduce the original three patches and then interpret the score. A new approach to the three systems had to be devised to allow the interpretation of the three studies without requiring too lengthy reconfigurations between studies (reconfiguring the entire system between each study would have taken too much time in a public concert setting). As such, since the interpretation of the three studies required more resources in modules and connections on the modular synthesiser and the systemic affordability it allows remains the same, despite all efforts from Legault to optimise the patches, it was impossible to play the three studies consecutively without adding another system. For the occasion, Lecours lent his system to address this challenge of interpretation and system design.

Thus, there was an initial patch conceptualisation for the instrument when the studies were approached separately and a second system design to integrate, as much as possible, the three studies on one system.

5.3. Interpretation of the studies

The interpretation of the studies raised multiple reflexions. One point highlighted during the joint work of the two authors was the difficulty of executing all the manipulations required by the score. At times, there were too many actions to perform simultaneously, making some sections impossible to play. There was thus a discrepancy between what the composer considered possible and what was achievable by the performer.

Additionally, the rehearsal of each study individually allowed for the repetition of a precise sequence of manipulations on the synthesiser. It has allowed to master each study separately but when played on other system some problems arose. We saw that the notation of the studies was approached in a way that the instructions could be transposed from one system to another and that the annotation of the manipulations to be performed uses the symbols employed in the section with the instructions for the patch construction. As such, the performer must transition from an abstract representation (the symbols referring to an abstraction of the patch) to concrete elements of the modular synthesiser. This approach allows the performer to correlate their system with what they have associated with a given symbol and the manipulation they need to perform. Although we consider this notation approach necessary to facilitate the possibility of playing the pieces on different systems, mastering the manipulations or sequences on the instrument through repetition becomes laborious. In the sense that rehearsing the studies that is linked to a specific patch system, which, when changed, requires relearning certain sequences and manipulations.

We experienced this issue when testing the studies on other systems. It required an adaptation period for each new system to be able to reinterpret the study. Since each module is often parameterised differently, we had to go through an exploration phase to create new reference points on the instrument. For example, the frequency distribution on a filter's control can vary greatly between modules, as can the operation of a filter. Another instance where this issue arose was during the concert setup. The sequence of the three studies required reconfiguring the patches to coexist within Legault's system and the one lent by Lecours. This reconfiguration required the performer to renew the reference points established when rehearsing of the studies separately.

5.4. Lack of literature

The ubiquitousness of the synthesizer in the musical landscape since the second half of the twentieth century makes in an unavoidable instrument for all types of musical production alike. However, we can observe a lack of literature to ensure the solidification of the modular synthesizer as a serious object of study. This lack has been identified by authors, such as Jeff Pressing, who dedicated a whole book to the apprenticeship of synthesizer performance [8]. He underlines that “skilled performance on the synthesizer, as on any instrument, demands an appropriate technique – a set of physical skills used by the performer to produce the desired sounds. Technique I the crucial link in the chain from musical conception to musical production and expression; through it, internally heard sounds become external. Obviously, technique must be both comprehensive and well-founded if it is to allow the full detail and precision of the performer’s musical conception to be expressed.” [7] Although Pressing identifies the importance of developing a technical practice in synthesizer performance, we can notice the absence of any mention of modular synthesizer performance technique in his comprehensive 1992 book Synthesizer performance, in which he does not include guidance for synthesizers not resembling other

acoustic instruments, be the keyboard, the guitar, wind instruments or percussions.

The growing interest for modular synthesizers has the potential to make space to develop this much-needed instrumental technique, and this specific set of studies aims to contribute to the scarcity of synthesizer performance literature.

5.5. Studies as pedological tools

The studies have been used in two teaching contexts since their creation. During tutoring sessions on modular synthesis, Lecours used these three studies as exercises for learning the basics of modular synthesis, system creation, and interpretation with the instrument.

The pedagogical contribution of the studies was remarkable, allowing students to have exercises to practice and eventually pieces to play.

6. CONCLUSIONS

The creation of modular synthesizer studies has confirmed both the potential and challenges inherent to this musical practice. The process of creating and interpreting these pieces has revealed several insights about the interaction between notation, system configuration, and performance practice.

As we saw in this paper, the integration of multiple voices within a single system often necessitates unusual solutions that can be challenging for performers. This aspect underscores the importance of familiarity with one's patches, allowing musicians to navigate their systems intuitively during performances.

Moreover, the lack of comprehensive literature on modular synthesizers remains a notable gap in the field. While synthesizers have become ubiquitous in contemporary music production, their study has not yet achieved the academic presence seen with other instruments. We suggest that this absence could eventually be resolved by the development of a standardized approach to performance and notation, which provide better and more effective communication among musicians.

The challenges faced during rehearsals and performances also point to the need for adaptable notation systems that facilitate transitions between different modular setups. The process of relearning sequences when switching systems can be particularly demanding, emphasizing the need for a more unified approach to synthesizer pedagogy.

We are planning to create new studies in the future to explore various forms of synthesis and interpretation. We also intend to publish these works once the research cycle is completed.

As modular synthesis continues to evolve, we do think that is this essential for scholars and practitioners alike to advocate for more structured research and documentation. By doing so, we can deepen our understanding of this unique instrument's capabilities and ensure that its rich sonic possibilities are accessible to a wider audience of musicians. The journey through the creation of these studies not only enhanced our technical skills but also enriched our artistic expression, opening to future electronic music composition and performance.

Acknowledgments

The Social Sciences and Humanities Research Council of Canada (SSHRC) for funding this research, the Laboratoire formes • ondes (LFO) de l'Université de Montréal and the Centre for Interdisciplinary Research in Music, Media, and Technology (CIRMMT). Patch Symbols from PATCH & TWEAK by Kim Bjørn and Chris Meyer, published by Bjooks, are licensed under CC BY-ND 4.0.

7. REFERENCES

1. Lecours, P.-L., & Bernier, N. (2024). Sound Synthesis Notation Applied to Performance: Two Case Studies. In P. Kocher (Ed.), Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'24 (pp. 142–149). Zurich University of the Arts.
2. Strange, Allen, and Gordon Mumma. *Electronic Music: Systems, Techniques, and Controls*. 2nd ed. Dubuque, Iowa: W.C. Brown Co., 1983, xvi.
3. Ciani, Suzanne. *Report to National Endowment, Re: Composer Grant*, 1976, 2.
4. Bugg, Richard. « Patch notation, a business approach." In *Synapse*, March/April 1978, 37.
5. Bjørn, K., et Meyer, C. (2018). Patch & Tweak. Exploring Modular Synthesis. Bjooks Media.
6. Orning, T. (2013). Pression Revised: Anatomy of Sound, Notated Energy, and Performance Practice. *Sound and Score: Essays on Sound, Score and Notation*, 94–109.
7. Bernier, N. (2022). Observations on Performing Sine Waves with an Oscillator Ensemble. *Leonardo, 55*(2), 161–165.
8. Pressing, Jeff. *Synthesizer Performance and Real-Time Techniques*. Madison: A-R Editions, 1992, 123.

Gesture as Score: A Hybrid Notation System for Guzheng and Electroacoustic Music

Bihe Wen

Macau University of Science and Technology
bhwen@must.edu.mo

ABSTRACT

This paper presents a hybrid notation system developed for *Immemorial/Flux*, a composition for prepared guzheng and fixed media requiring continuous slide techniques and coordination with electronic sound. Traditional guzheng notation offers limited tools for representing such action-based techniques or aligning performance with fixed-media timelines. The proposed system combines three components: an adapted staff for temporal orientation, a graphic layer that specifies left-hand slide gestures, and an embedded sonogram that visualizes the temporal–spectral structure of the electronic track. The notation was refined through rehearsal with professional performers, whose feedback informed adjustments to symbol clarity and the sonogram's practical function in rehearsal. The findings indicate that this selectively detailed, performer-oriented framework can support extended techniques within mixed electroacoustic contexts while remaining compatible with established reading practices.

.

1. INTRODUCTION

Immemorial/Flux, a composition for prepared guzheng and fixed media, examines how musical notation can support extended techniques that rely on continuous gesture, spatial motion and coordination with electronic sound. Traditional guzheng notation, whether character based, numbered or adapted from the five-line staff, provides effective pitch and rhythmic representation but offers limited means for describing these action-based techniques. Although contemporary guzheng performance increasingly incorporates timbral exploration, physical objects and electroacoustic elements, notational approaches that address these developments remain relatively underexplored. A notation that reflects performers' embodied actions is therefore needed.

This paper proposes a hybrid notation system developed for *Immemorial/Flux*. The system integrates three complementary components: an adapted staff that provides temporal orientation and string reference, a graphic layer that represents left-hand slide gestures with spatial clarity, and a sonogram that offers a temporal and spectral overview of the fixed-media track. Rather than replacing established notational practices, the framework aims to balance familiarity and precision with the flexibility required to support continuous gesture-based techniques.

The remainder of the paper outlines the motivations for this approach, presents the structure and design principles of the hybrid system, and discusses how it was refined in collaboration with performers. The paper concludes by considering the broader implications of performer-oriented notation for works that combine extended techniques with fixed media.

2. CONTEXT AND MOTIVATION

Traditional guzheng notation has evolved from early character-based and orally transmitted methods to twentieth-century numbered (cipher) and staff notations. Across these traditions, notation has primarily represented discrete pitch and rhythm, effectively serving modal and melody-based repertories but offering limited means to convey continuous motion and timbral nuance. In recent decades, some contemporary guzheng works have explored cross-tuning, prepared materials, and interaction with electronics, expanding the instrument's sonic palette. While these works have developed their own notational approaches for extended techniques, such approaches remain highly piece-specific and therefore lack broader applicability.

The focus on performer action as a primary notational concern has been explored by several composers since the late twentieth century. As Santini explains [1], gesture-based notation has developed along several distinct trajectories [1]. In Helmut Lachenmann's *Pression* (1969), the score specifies sequences of instrumental gestures while leaving pitch and sonic outcome partly indeterminate. Aaron Cassidy's *Second String Quartet* (2010) separates notation into independent layers for finger, bow and pressure actions, allowing sound to emerge from the interaction of these physical processes. Pierluigi Billone's *Mani.Mono* (2007), by contrast, employs a gesture-based system in which each prescribed motion produces a single, consistent sonic result. A comparable approach appears in Simon Steen-Andersen's *AMID* (2015), which, as Stene observes, prescribes performers' movements rather than their sonic results, effectively reversing the hierarchy between gesture

and sound; the score isolates and values specific motions as compositional material and as a choreography of performance [2]. Together these examples illustrate how action-centred notation can encompass both indeterminate and highly controlled sound relationships.

A related line of exploration can be found in approaches that separate performance parameters across multiple staves, aiming to clarify complex instrumental actions. During my graduate study with Panayiotis Kokoras, I was influenced by his notational approach to instrumental and electroacoustic writing, which frequently distributes different aspects of performance behaviour across separate staves; for example, in *Mutation* (2015) for clarinet and electronics, three staves respectively indicate embouchure, fingering, and the electronic part as a sonogram. A comparable principle underlies Aaron Cassidy's notation, which assigns distinct parameters such as finger motion, bow direction, and pressure to separate staves in order to map sound construction through physical process. Yet, as several authors observe in relation to Cassidy's system, this highly detailed approach may result in visual density and a steep learning curve that challenge practical readability [3]–[6]. My engagement with this multi-layered, parameter-specific notation reflects an awareness of the need to balance detailed representation with interpretive accessibility, underscoring the challenge of maintaining precision without adding unnecessary cognitive complexity.

Alongside these gesture-oriented and parameter-separated approaches, composers have also explored graphic and animated notations, which broaden the visual representation of timbre and process and often privilege interpretive openness, real-time visualization, and improvisatory flexibility [7]. While such approaches offer valuable insights, they are less suited to the specific temporal and performative requirements of *Immemorial/Flux*.

These observations collectively underline the need for a notation capable of mediating between physical action, sonic result, and performer readability. The hybrid framework developed for *Immemorial/Flux* responds to this need by integrating gestural clarity with the structural familiarity of the five-line staff, providing a balanced means of representing both instrumental and electroacoustic elements. The following section outlines the design principles that informed the development of this system.

3. HYBRID NOTATION SYSTEM

This section presents the hybrid notation system for *Immemorial/Flux*. It outlines the structure of the three-layer score, articulates the design principles behind it, details the graphic encoding of left-hand slide gestures, and explains the role of the sonogram as a visual guide to the electronic layer.

3.1. Design Principles

This framework is based on three interrelated design principles that guided the development of the hybrid notation system.

3.1.1. Embodied Clarity

Notation is designed to reflect physical action. By depicting gesture directly rather than relying on abstract symbols, the score specifies the movement through which the intended timbral and sound-based effects are produced. Making the required actions visually explicit reduces interpretive ambiguity and allows performers to engage more naturally with the music through bodily intuition during performance.

3.1.2. Cultural Continuity

Retaining the five-line staff builds on performers' existing notational literacy and aligns the system with contemporary guzheng pedagogical practice. This continuity preserves accessibility and also provides a familiar structural anchor through which new gestural elements can be integrated without creating an entirely new notational language.

3.1.3. Notational Integration

The hybrid score integrates three complementary layers, each addressing a different performance need. The adapted staff offers temporal orientation and string references; the gesture layer depicts left-hand slide motions in a direct visual form; and the sonogram provides a temporal–spectral view of the fixed media. Designed to work together rather than to function independently, these layers form a coherent notational environment in which each component supports and reinforces the others.

3.2. Score System Overview

Immemorial/Flux employs a three-stave hybrid notation (Fig. 1). Each stave serves a distinct but integrated function.

3.2.1. Guzheng-Strings (first stave)

As shown in Fig. 1, a conventional five-line staff is employed, but its pitch notation refers to string reference rather than fixed pitch, since the actual intonation is continuously modified by left-hand motion. Square noteheads indicate plucked strings, providing a simplified temporal–string framework rather than a prescriptive pitch notation. The notation in this stave is deliberately economical: symbols are restricted to those that support performance clarity, avoiding unnecessary detail while maintaining efficiency and interpretive precision.

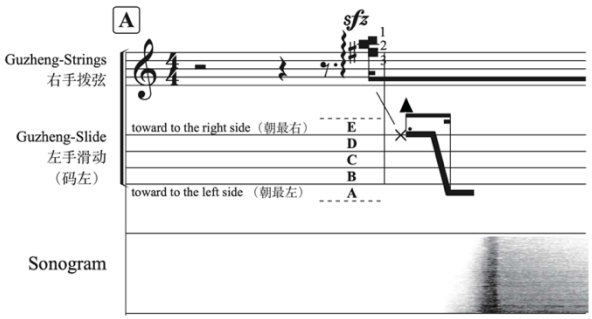


Fig. 1. Hybrid three-stave notation: right-hand strings, left-hand slide gestures, and fixed-media sonogram.

3.2.2. Guzheng-Slide (second stave)

Graphic symbols map left-hand slide gestures onto five spatial zones (A–E), as shown in Fig. 2. Lines, arrows, and icons directly represent four motion types: horizontal sliding, rubbing, scraping, and muted glissando. These visual elements provide clear cues and strong gestural correspondence, as shown in Figs. 3(a)–3(d).

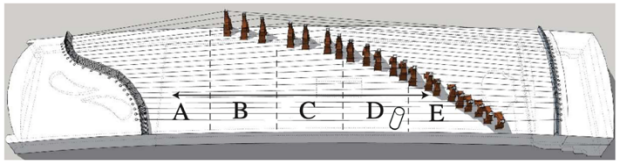


Fig. 2. Division of the guzheng's left side into five spatial zones (A–E) for slide gestures.

3.2.3. Sonogram (third stave)

A spectrogram of the fixed-media track functions as a visual timeline, showing the spectral density of the electronic layer and helping performers synchronize live gestures with its temporal structure.

3.3. Slide Techniques for Guzheng

Four primary slide techniques, performed with a metal guitar slide, are represented through graphic symbols that reflect the physical actions of performance, as shown in Figs. 3 and Fig. 4.

3.3.1. Horizontal Sliding (Fig. 3)

A solid line with arrowheads indicates lateral motion and directional intent.

3.3.2. Rubbing (Fig. 3)

A zigzag line conveys rhythmic tremolo gestures, with icon placement specifying hand orientation.

3.3.3. Scraping (*Fig*. 3)

A slanted line denotes angled edge contact; accompanying annotations specify speed and pressure.

3.3.4. Muted Glissando (Fig. 4)

Dashed versus squiggly lines differentiate overtone-like muting from damped scraping, annotated with hand-icon cues.

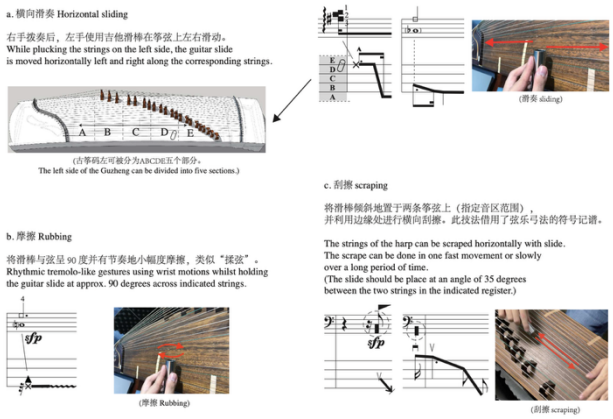


Fig. 3. Performance notes of Horizontal Sliding (a), Rubbing (b), and Scraping (c) on the Guzheng.
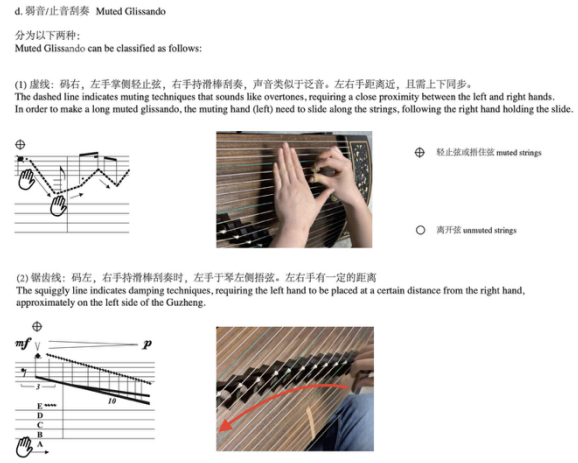


Fig. 4. Performance notes of muted glissando on the Guzheng.

3.4. Sonogram as Listening Aid

The sonogram embedded in the score represents the spectral structure of the fixed-media track, functioning as a visual aid rather than a performative notation. Because the electronic part lacks defined pitch or rhythmic material, conventional notation fails to convey its structure effectively. Instead, the sonogram visualizes temporal and spectral organization across time, frequency, and intensity. As Landy suggests [8], in many electroacoustic contexts, representation tends to shift from symbolic notation toward visual tools that aid listening and coordination. Within this framework, the sonogram offers performers a visual guide to the electronic part, enabling temporal and expressive coordination with the live guzheng. Though it does not provide precise rhythmic or pitch cues, it enhances temporal awareness and ensemble cohesion. Its effective use relies on brief orientation and collaborative rehearsal, especially for performers new to such hybrid scores. Nevertheless, the system is learnable and adaptable across similar works, illustrating notation as an interface that mediates perception rather than dictates execution.

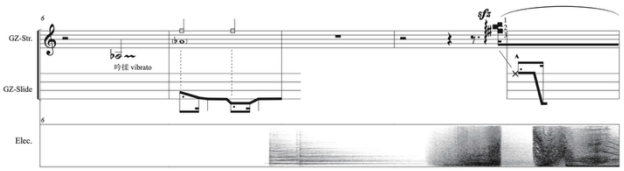


Fig. 5. Sonogram excerpt illustrating temporal coordination between guzheng slide gestures and the fixed-media track

4. PERFORMER ADAPTATION AND COLLABORATION

4.1. Collaboration Process

In 2023, the piece was prepared and performed through two separate rehearsal–performance processes: first with Yuxin Lu for the International Computer Music Conference (ICMC 2023, Shenzhen), and later with Runyitong Zhang for Musicacoustica–Hangzhou 2023. Because both collaborations faced significant time and geographical constraints, early preparation was carried out remotely. Performers practiced from the score and sent rehearsal videos for feedback, allowing small notational refinements before brief in-person rehearsals shortly prior to each performance. This remote exchange made it clear which notational elements communicated effectively and which benefited from refinement during rehearsal. Performer comments on symbol clarity, spacing, and synchronization guided refinements to the score layout. Practical considerations also surfaced, such as how different string tensions influenced the resonance and stability of slide-based techniques. While not intended for notation, this finding highlighted how instrument setup affects the practical realization of the score.

4.2. Performer Feedback

Two professional guzheng players, one with traditional conservatory training and the other more experienced with Western staff notation, reported consistent advantages.

4.2.1. Spatial Zones

The five labeled zones (A–E) provided clear reference points, reducing hand-placement errors and rehearsal time.

4.2.2. Graphic Cues

Direct representational symbols for sliding, rubbing, scraping and muting were immediately intelligible, allowing performers to focus on musical expression rather than decoding notation.

4.2.3. Staff Familiarity

The adapted five-line stave anchored timing and pitch expectations, offering assurance in sections where graphic notation dominated.

4.2.4. Sonogram Alignment

Embedding the spectrogram into the score enhanced real-time synchronization with electronics, reducing reliance on external click tracks.

4.2.5. Instrument Setup

Both performers noted that string tension influenced the ease and resonance of slide-based techniques: looser strings risking bridge instability (as observed by Yuxin Lu), and tighter strings limiting vibrational richness (as reported by Runyitong Zhang). This feedback clarified the practical conditions under which the notated gestures are most effectively realized.

Overall, performer feedback suggests that the hybrid notation functioned as intended in rehearsal, particularly in coordinating actions with the fixed media.

4.3. Using the Sonogram in Rehearsal

In rehearsal, performers confirmed that the sonogram served as a temporal–spectral guide rather than a prescriptive cue. Brief introductions on reading spectral activity and event onsets helped them grasp the display quickly. As performers became familiar with the sonogram, they aligned their performance timing with the fixed media by following spectral developments, gradually reducing their reliance on the click track in certain passages.

5. Conclusions

This paper has presented a hybrid notation system for *Immemorial/Flux*, combining gesture-based symbols, adapted staff notation and an embedded sonogram to support extended guzheng techniques and coordination with fixed media. Collaboration with performers indicated that the system functioned reliably in rehearsal and aligned well with established reading habits. More broadly, the work suggests that the notation reflects how its sounds are generated, through continuous physical gestures rather than discrete pitches. The compositional and rehearsal process also emphasized the value of selective notation: some continuous actions were understood more intuitively through brief video demonstrations, not because the notation was inadequate, but because certain gestures are more directly grasped through visual motion. This reinforced the importance of providing essential notational information while avoiding unnecessary detail, allowing room for performers to interpret movement through embodied experience. While developed for a single composition, these considerations may offer a useful point of reference for works that engage extended techniques, sound-based materials or fixed-media environments.

6. REFERENCES

1. G. Santini, “Action Scores and Gesture-Based Notation in Augmented Reality,” in *Proceedings of the TENOR 2020 Conference on Technologies for Musical Notation and Representation*, Hamburg, Germany, 2020, pp. 83–90.

2. T. R. Moore and P. Gielen, “The Politics of Conducting: The conductor’s role in new music ensembles based on a study of performance practice in Simon Steen-Andersen’s AMID,” *Journal for Artistic Research*, vol. 9, 2021, ISSN 1893-9562, doi: 10.32063/0903.

3. M. Iddon, “Notational Entanglements: Complexity and the Material of Performance,” *Contemporary Music Review*, vol. 32, no. 5–6, pp. 537–550, 2013.

4. C. Fox, “Why Complexity?,” *Tempo*, vol. 67, no. 263, pp. 20–33, 2013.

5. A. Cassidy, “Interview: Aaron Cassidy in Conversation,” *Search: Journal for New Music and Culture*, no. 9, pp. 1–12, 2013.

6. P. Archbold, “Complexity and Notation: Performer Perspectives,” in *Complexity and the Performing Body: Perspectives on Contemporary Musical Performance*, L. MacArthur, J. P. E. Harper-Scott, and N. Reyland, Eds. New York, NY, USA: Routledge, 2019, pp. 107–123.

7. C. Hope, “Electronic Scores for Music: The Possibilities of Animated Notation,” *Computer Music Journal*, vol. 41, no. 3, pp. 21–35, 2017.

8. L. Landy, *Understanding the Art of Sound Organization*. Cambridge, MA: MIT Press, 2007.

Practice Strategies for Polyphonic Music: Exploring the Role of Gesture Controllers in Enhancing Technical Stability and Optimizing Mapping Methods

Yu-Hsin Chang

Macau University of Science and Technology
yhchang@must.edu.mo

ABSTRACT

Polyphonic music presents unique technical and cognitive challenges for performers, especially when it comes to voice independence, finger coordination, and memorization (Sweller et al., 2019). This study proposes an interactive learning model that integrates Max/MSP gesture recognition technology with Ableton Live sequencing to enhance practice strategies for polyphonic repertoires, with a focus on Bach fugues. In the proposed system, distinct hand gestures are mapped to different musical voices and mapped to separate MIDI tracks to facilitate targeted practice of motifs, counter-motifs, and episodes. The design aims to optimize memorization, expand hand-eye span, reduce cognitive load, and support flow in performance. Technical comparisons of various Max/MSP gesture recognition approaches are discussed, alongside a detailed account of the system's architecture. Instead of large-scale quantitative testing, initial consultation and feedback were gathered from piano students and music educators to assess the system's usability and perceived pedagogical value. These expert insights suggest that technology-assisted practice schemas, grounded in embodiment and cognitive load theory, hold substantial promise for supporting the acquisition of polyphonic works and offer new avenues for future research and innovative pedagogy in music education.

1. INTRODUCTION

The mastery of polyphonic music stands as a perennial challenge for musicians, requiring advanced coordination, auditory discrimination, and memory skills (Godøy & Leman, 2010). Works such as Bach fugues, and other multi-voice compositions, exemplify these difficulties: they demand the simultaneous management of independent melodic lines by both hands and stretch the limits of traditional practice strategies, such as sectional repetition, hands-separate drills, and slow practice (Sweller et al., 2019). These conventional approaches often fall short when confronting the cognitive complexity inherent in polyphonic music.

Recent advances in digital music systems and interactive technology now offer innovative ways to re-envision these practice paradigms (Lyons et al., 2016). Gesture recognition platforms in environments such as Max/MSP, in combination with controllers like Leap Motion or Myo, create opportunities for linking bodily movement directly with musical structure (Lyons et al., 2016; Gillian & Paradiso, 2014).

This paper introduces a conceptual framework for gesture-based polyphonic practice, leveraging Max/MSP gesture detection mapped to independently tracked voices in Ableton Live. Rather than conducting quantitative user trials, the current stage of research focuses on iterative development and gathering expert feedback through consultations with students and teachers. This approach enables a critical examination of the technical, musical, and pedagogical implications of this technology-driven practice model, and lays the groundwork for subsequent empirical testing and refinement.

2. TECHNICAL BACKGROUND

2.1. Methods of Max/MSP Gesture Recognition

Gesture recognition in Max/MSP has evolved rapidly, leveraging advances in both hardware and software to capture and interpret real-time user movements for musical interaction. Among the most widely adopted solutions, the Leap Motion controller provides low-latency, high-precision skeletal hand tracking, and is particularly valued where precise spatial sweeping and movement in three-dimensional space are required—making it especially suitable for intuitive "spatial sliding" gestures (Gillian & Paradiso, 2014). In parallel, computer vision frameworks such as MediaPipe enable robust, multi-modal hand and body tracking via standard cameras, excelling in the recognition of continuous and complex gesture sequences, which is particularly beneficial for tracking nuanced, sustained motion over time (Lyons et al., 2016). Additional OSC-based

Copyright: © 2025 Yu-Hsin Chang. This is an open-access article distributed under the terms of the Creative Commons Attribution License 3.0 Unported, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

devices, such as Myo armbands and IMU-based gloves, offer alternative low-latency sensing modalities and have found success in event triggering and continuous control applications. Max/MSP's modular ecosystem—combining objects like jit.matrix, coll, and Node for Max—enables seamless routing and transformation of gesture data, supporting rapid prototyping and flexible mapping strategies suited to diverse musical contexts.

Each technology presents specific trade-offs in setup complexity, accuracy, robustness, and extensibility. Leap Motion and MediaPipe, in particular, are favored in the interactive music community for their versatility, active developer support, and rich open-source resources. Critically, when selecting or designing a gesture recognition platform for musical purposes, practitioners must weigh not only technical metrics (precision, latency, computational overhead) but also musical relevance: the ability to reliably and expressively map nuanced gestures—whether discrete spatial sweeps or continuous tracked motions—to meaningful musical actions in real-time performance (Gillian & Paradiso, 2014).

2.2. Challenges of Practicing Polyphonic Music

Polyphonic music poses distinctive challenges for learners, distinct from those encountered in homophonic styles. Unlike homophonic music, which features a clear melody and accompaniment, polyphonic works consist of multiple independent melodic lines intricately interwoven. This simultaneous interplay of independent voices places considerable demands on a performer's cognitive resources. When students practice Bach's fugues from the Well-Tempered Clavier, for example, they must maintain independent lines, recognize and differentiate subjects and countersubjects, analyze contrapuntal structure, master shifting tempos, balance ensemble voices, and memorize lengthy sequences—each constituting a significant cognitive load (Sweller et al., 2019). According to Cognitive Load Theory, working memory has definite limits, and tasks that exceed those limits can result in fragmented processing and poor retention.

The risks of cognitive overload are exacerbated in polyphonic repertoire: the contrapuntal texture provides no clear division between melody and accompaniment, but rather presents a "melody versus melody" scenario. In fugues with three or more voices, the distribution of melodic content across both hands often makes separate hands practice insufficient or even meaningless for achieving true independence and fluency. Memory challenges are further compounded by the need to integrate motor (muscle) memory with analytic understanding. While muscle memory allows fingers to "remember" embedded gestural patterns even amid overlapping lines, it is insufficient for passages involving wide leaps or complex fingerings; cognitive strategies must supplement physical repetition.

The value of slow practice is emphasized in this context: working through the interactions of subject and countersubject in small fragments, while integrating harmonic progressions, supports a gradual discovery process. Though this incremental approach fosters deep understanding, it requires significant time and patience. Ultimately, effective mastery of polyphonic music demands a nuanced integration of embodied (kinesthetic) and cognitive faculties, as well as a flexible approach to chunking, attention, and memory. Innovative technical solutions, such as those enabled by gesture recognition and real-time feedback, are needed to address these persistent learning barriers and optimize the acquisition of polyphonic repertoire.

2.3. Benefits of this Proposed Method

Integrating gesture-controlled Max/MSP environments into the polyphonic practice routine yields multiple educational and artistic advantages uniquely suited to the demands outlined above. By leveraging the real-time, high-precision tracking capabilities of tools like Leap Motion for spatial gestures and MediaPipe for continuous, complex motion capture, this method enables performers to intuitively map bodily movement to discrete musical functions. For polyphonic music, where cognitive resources are taxed by concurrent melodic strands and traditional hands-separate practice often fails to foster true voice independence, targeted gesture mapping offers critical scaffolding. Assigning specific gestures to individual MIDI tracks—such as motifs, counter-motifs, and episodes—not only allows learners to isolate and sequentially engage with each musical line, but also helps manage cognitive load by turning abstract polyphonic relationships into concrete, embodied actions (Sweller et al., 2019).

The immediacy of gesture detection and visual or auditory feedback in Max/MSP supports a virtuous cycle between movement and sound, actively reinforcing auditory-motor associations and encouraging the development of robust muscle memory as described by Leman (2008). This multisensory feedback loop is particularly salient for complex fingerings or passages involving rapid leaps, where kinesthetic awareness must be consciously integrated with analytic understanding. Moreover, the inherent flexibility and modularity of Max/MSP environments allow instructors and performers to adapt and customize mappings to personal ergonomic needs, skill levels, or repertoire, further supporting individual learning trajectories (Gillian & Paradiso, 2014).

Perhaps most importantly, technology-assisted gesture practice makes salient the Gestalt principles of grouping, segmentation, and exploratory listening identified in recent neurocognitive research (Mencke et al., 2022; Trujillo & Holler, 2023). Real-time interaction with structural and gestural features promotes moments of pattern discovery, flow, and insight—foundational experiences for mastering the intricacies of polyphonic repertoire. Thus, integrating gesture recognition does not merely augment traditional practice but fundamentally reconfigures the learning process, allowing music students to more efficiently encode, retain, and confidently execute complex, multi-voiced works.

Having established these educational and cognitive benefits, the next section details the specific materials and methodologies by which this integrated system is implemented and evaluated.

3. MATERIALS AND METHODS

3.1. MIDI Preparation

For this study, polyphonic works by J.S. Bach—specifically his fugues—were chosen as the primary experimental material. This selection is motivated by several factors. First, fugues composed for a single instrument (piano) offer a controlled testing environment: the sound source is both realistic and reproducible, and the physical layout of the instrument helps ensure that observations are not confounded by orchestration or ensemble performance variables. In contrast, other paradigmatic polyphonic works, such as fourteenth-century Renaissance vocal music, often span longer durations and involve multiple performers or complex instrumentation, making them less practical for detailed, gesture-based experimental protocols.

Each Bach fugue is imported into Ableton Live as a MIDI file and carefully separated so that each independent voice—whether in a three-voice or five-voice texture—is assigned its own track. For example, a three-voice fugue yields three MIDI tracks, while a five-voice fugue yields five, mirroring the compositional structure. Figure 1 shows a four-voice fugue programmed in Ableton Live labeled indicated with analysis marks. This division is essential for both analytical clarity and for mapping specific gestures to discrete musical actions in the subsequent Max/MSP environment.

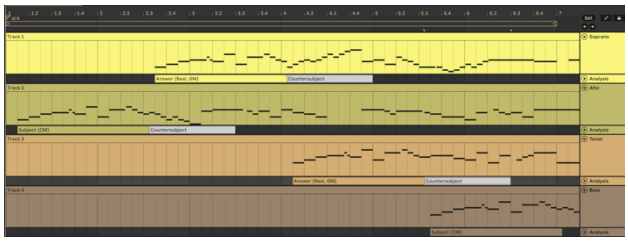


Figure 1. An Example of MIDI Preparation. Bach: Fugue No.1 from The Well-Tempered Clavier, Book 1, mm.1-6

To enhance both the learning experience and system feedback, each MIDI track is meticulously annotated. Motives, countersubjects, and episodes are color-coded: themes and imitative lines within the same key region are assigned analogous hues, supporting the performer's visual association between related musical material and physical gesture. When modulations or key changes occur, the color assignation shifts accordingly—contrasting colors are employed to visually signal increased harmonic distance and structural transitions. This systematic color-mapping aids users in tracking polyphonic relationships and anticipating musical changes during practice.

For larger musical sections and formal boundaries, custom tab markers are inserted in Ableton Live. Popularly known as a "locator," these markers enable efficient navigation, allowing learners to jump directly to specific sections or isolate challenging passages for repeated, focused practice. Practicing music in chunks helps performers focus on detail and build durable muscle memory.

This modular approach to MIDI preparation not only supports isolated voice work but also enables dynamic combinations and custom sequencing, further optimizing the gesture-controlled practice workflow for individual learners.

3.2. Max/MSP Design

The Max/MSP component is built as a two-tiered patch, explicitly designed to address the challenges of polyphonic practice. The first module provides an interactive menu that enables users to select repertoire and load the corresponding pre-annotated MIDI tracks, where each independent voice—differentiated by color and structural markers as detailed in Section 3.1—appears as a discrete entity. The second module implements flexible gesture recognition, utilizing Leap Motion (preferable for spatially expressive sweeps) and MediaPipe (optimized for continuous, nuanced gesture sequences), based on prior analysis (Section 2.1).

A rigorous mapping scheme links specific gestural vocabularies—such as pointing, hand opening, pinching, or multi-finger configurations—to distinct voice entrances and structural events in the MIDI file. Users can choose either to control a single line or navigate between combinations, directly targeting the cognitive bottlenecks of polyphonic learning highlighted in Section 2.2. Visual interface elements within the patch provide real-time feedback through color and iconography, helping users track both which voice is currently engaged and how gestures map onto musical gestures or transitions.

Max/MSP handles gesture-to-voice mapping logic, seamlessly transmitting control data via MIDI or OSC to Ableton Live to achieve precise synchronization, even in complex multi-voice passages. All gesture-related events—including accuracy, timing, and transition patterns—are automatically logged for comprehensive analysis, enabling both quantitative performance assessment and qualitative studies of learner strategy and engagement. This integrative design ensures that technical innovation translates directly into pedagogical value, reinforcing embodied memory, focused attention, and adaptive learning for polyphonic mastery.

3.3. Supplemental Materials: Ableton Live and Score Following

To further support and contextualize the gesture-based practice workflow, dedicated modules within Ableton Live are developed that reflect and extend the structure established in Max/MSP. Within these modules, every motif, counter-motif, and episode is color-coded and explicitly labeled, allowing users to visually distinguish between different musical voices and formal sections at a glance. This integration ensures that the mapping between gesture, sound, and visual reference is consistent across both the interactive patch and the sequencer environment.

Crucially, Ableton Live's automation and looping functions enable precise, hands-free repetition of challenging segments. Learners can rehearse individual voices, motifs, or entire sections in isolation or in combination with other tracks, rapidly cycling through difficult passages without unnecessary interruption. This modular system fosters targeted, adaptive rehearsal—users can shift focus as needed or experiment with different polyphonic configurations, supporting the individualized, attention-guided rehearsal advocated earlier.

For advanced users, the implementation of score-following algorithms marks a further pedagogical enhancement. By synchronizing score position, audio/MIDI playback, and live gesture input, the system provides seamless, real-time auditory and visual feedback. This feature not only reinforces the mapping between physical gesture and musical structure, but also accelerates the transfer of embodied, gesture-based skills to traditional keyboard performance settings, bridging the gap between exploratory practice and concert-ready fluency.

Collectively, these supplementary materials both reinforce cognitive mapping of complex polyphonic relationships and empower users to consolidate their learning through interactive and multisensory engagement. With the technical infrastructure established, the next section will present the empirical results and discuss how these innovations impact technical stability, memorization, and learning motivation among polyphonic music students.

4. EXPERT INSIGHTS AND PRELIMINARY REFLECTIONS

4.1. Expert Perspectives on Mapping Strategies and Gesture Triggering

Consultations with experienced piano educators and advanced students indicate that gesture-to-voice mapping holds strong pedagogical promise for polyphonic music practice. Experts observe that assigning intuitive, physiologically natural movements (such as finger spreading for voice entry or hand closure for phrase termination) to specific motifs and counter-motifs could help learners isolate and reinforce independent lines. These targeted mappings are seen as a way to minimize cognitive interference between voices, supporting focused, repetitive practice without overwhelming working memory resources (Sweller et al., 2019; Leman, 2008). Participants note that, compared to keyboard-only routines, such gesture integration may facilitate a faster acquisition of voice independence and a more accurate rendering of complex textures. Furthermore, the real-time visual and auditory feedback afforded by the system is believed to foster immediate error correction and strengthen technical reliability. These impressions are consistent with earlier research on embodied interaction for music learning, suggesting that technology-mediated approaches can deepen engagement and efficiency (Gillian & Paradiso, 2014). However, these findings remain provisional and reflect the perceptions of consulted experts rather than outcomes from systematic experimental trials.

4.2. Views from Student and Teacher Consultations

Semi-structured interviews and informal usability feedback collected from piano students and music teachers generally point to a favorable reception of the gesture-assisted practice framework. Many users report that the design increases their confidence in identifying and memorizing distinct voices, and some suggest it could help reduce anxiety when working through challenging polyphonic sections. The immediacy and intuitiveness of gesture input are credited with making practice more engaging and interactive, transforming repetitive tasks into playful exploration. Additionally, the combination of visual (color-coded tracks), auditory (live playback), and kinesthetic (gesture) cues is viewed as supporting deeper learning and more robust retention. Criticisms mainly center on initial system calibration and the learning curve, highlighting needs for better interface customization and onboarding materials. Participants also emphasize that real-world classroom implementation would benefit from iterative refinement and further teacher input. While these insights are encouraging, they primarily reflect qualitative perceptions and preliminary classroom observations, to be followed by more rigorous empirical validation. Such findings align with broader evidence indicating that technology-assisted, interactive practice environments can enhance motivation and engagement—both crucial for long-term musical development (Anglada-Tort et al., 2023; Fink et al., 2021).

5. CONCLUSION AND FUTURE DIRECTIONS

This proof-of-concept study suggests that the integration of gesture recognition technologies with modular MIDI segmentation in a Max/MSP-based system may provide significant pedagogical advantages for polyphonic music learning. The platform enables precise, embodied mapping between performer movement and musical structure, potentially reducing cognitive load and supporting more effective memorization and technical development (Leman, 2008; Sweller et al., 2019). Expert and user feedback to date affirms the conceptual and practical value of this approach, highlighting improvements in learner confidence, motivation, and achievement. Nonetheless, some technical challenges remain—notably, gesture calibration and the need for enhanced user accessibility. As the research advances, emphasis will shift toward comprehensive pilot trials, broadening hardware compatibility, refining gesture vocabularies, and conducting systematic, large-scale investigations to further assess and strengthen the educational impact of this interactive approach.

Acknowledgments

This research was funded by the Education Fund of the Government of the Macao Special Administrative Region (澳門特別行政區政府教育基金).

6. REFERENCES

1. Anglada-Tort, M., Harrison, P. M. C., Lee, H., & Jacoby, N. (2023). Large-scale iterated singing experiments reveal oral transmission mechanisms underlying music evolution. *Current Biology, 33*(13), 2875–2887. https://doi.org/10.1016/j.cub.2023.05.027

2. Fink, L. K., Warrenburg, L. A., Howlin, C., Randall, W. M., Hansen, N. C., & Wald-Fuhrmann, M. (2021). Viral tunes: Changes in musical behaviours and interest in coronamusic predict socio-emotional coping during COVID-19 lockdown. *Humanities and Social Sciences Communications, 8*(1), 180. https://doi.org/10.1057/s41599-021-00814-4

3. Gillian, N. E., & Paradiso, J. A. (2014). *Multimodal sensing for hand gesture recognition in musical performance*. In Proceedings of the 14th International Conference on New Interfaces for Musical Expression (pp. 297-302).

4. Godøy, R. I., & Leman, M. (2010). *Musical gestures: Sound, movement, and meaning*. New York, NY: Routledge.

5. Kirschner, S., & Tomasello, M. (2009). Joint music making promotes prosocial behavior in 4-year-old children. *Evolution and Human Behavior*, 30(5), 329-337. https://doi.org/10.1016/j.evolhumbehav.2009.03.00

6. Leman, M. (2008). *Embodied Music Cognition and Mediation Technology*. Cambridge, MA: MIT Press.

7. Mencke, I., Omigie, D., Wald-Fuhrmann, M., & Brattico, E. (2022). Atonal music as a model for investigating exploratory behavior in learning and engagement. *Frontiers in Neuroscience, 16*, 793163. https://doi.org/10.3389/fnins.2022.793163

8. Sweller, J., van Merriënboer, J. J., & Paas, F. (2019). Cognitive architecture and instructional design: 20 years later. *Educational Psychology Review*, 31, 261–292. https://doi.org/10.1007/s10648-019-09465-5

9. Trujillo, J. P., et al. (2023). Trujillo, J. P., & Holler, J. (2023). Interactionally Embedded Gestalt Principles of Multimodal Human Communication. *Perspectives on Psychological Science*, *18*(5), 1136-1159. https://doi.org/10.1177/17456916221141422

JUST INTONATION IN BACH

Andrea Agostini
University and Conservatoire of Antwerp
Orpheus Institute Ghent
andreaagostini@bachproject.net

Daniele Ghisi
Conservatory of Turin
danieleghisi@bachproject.net

ABSTRACT

This article discusses the experimental implementation of just intonation within the *bach* [1] software tool, a package for Max [2] dedicated to musical representation and computer-aided composition. This implementation augments the already extant pitch data type with a representation of Helmholtz-Ellis Just Intonation pitches, as well as a set of operations upon them, providing users with the virtually unique ability, among software tools of this kind, to manipulate algorithmically, as well as representing graphically, such kind of pitch representation.

We shall firstly discuss the context for this new feature set. We will then provide a brief introduction to the main concepts of modern just intonation systems. Next, an overview of the current status of pitch representation in *bach* will be given, complemented by some general considerations about tuning systems in computer music software. Then we will delve into the details of the actual implementation of just intonation in *bach*, from the perspectives of both the user and the underlying workings, describing the syntax of the pitch representation and the possible mathematical operations on justly tuned pitches, and providing an overview of the *bach* modules dealing with them. Finally, we will present some simple examples of the new possibilities afforded by this new addition to the *bach* package.

1. INTRODUCTION

Pitch representation in computer music software is a complex topic. Simplistic representations, such as frequency values or MIDI pitches, are easily implemented and allow for a basic set of fundamental operations such as transpositions, expansions and inversions in the chromatic or frequency space. The conversion between the two spaces is mathematically simple, and involves logarithmic and exponentials (a corollary of the fact that human perception of intervals relates to frequency ratios, rather than to frequency differences).

On the other hand, such representations neglect a number of details and possibilities that may or may not be relevant according to the aesthetic, theoretical and technical context in which the representation is employed. These include enharmonicity, that is, the distinction between e.g. C♯ and D♭, and the expression of pitches in different tuning systems.

Software tools for computer aided composition approach these problems differently. The venerable PatchWork family, including Patchwork itself and PWGL (both obsolete nowadays) as well as OpenMusic, [3] used to rely on MIDIcents, which are just a higher-resolution variant of MIDI pitches, with limited support for enharmonic information. As mentioned above, this, and more specifically the fact that real-valued MIDIcents are supported, in principle allows one to express any pitch in any tuning system. In this scenario, native semantic support — including, but not limited to, graphical representation — is only provided for divisions of the octave in 12, 24 or 48 equal steps. Recent versions of OpenMusic provide semantic support for other equal divisions of the octave. On the other hand, as we shall see in section 2, this only covers a part of the problem.

OpusModus [4] and Music21 [5] support operations with both pitch classes (thus disregarding enharmonic information) and an enharmonically-aware pitch data type; operations upon these two different representations are however expressed very differently. [6]

MaxScore [7] provides extensive support for all the aforementioned features, with a strong focus on alternative tuning systems, but it is aimed at graphical representation much more than algorithmic processing, albeit in principle (and, to some extent, in practice) the latter is also possible [1].

The latest publicly distributed version of the *bach* package (version 0.8.1) supports both MIDIcents representation and a pitch type, which features advanced support for micro-intervals and enharmonicity. On the other hand, its support for alternative tuning systems is quite limited, and essentially applies to equal-tempered microtonal subdivisions of the octave.

[1] www.bachproject.net
[2] http://cycling74.com

Copyright: ©2025 Andrea Agostini et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

[3] https://openmusic-project.github.io
[4] https://opusmodus.com
[5] www.music21.org
[6] In computer science terms, the pitch data type of OpusModus, as well as that of OpenMusic and Music21, is a class, that is, a construct associating data and operations upon it. This collides with the fact that "pitch class" has a very specific, and completely unrelated, meaning in music theory — the meaning referred to when discussing OpusModus and Music21 a few lines above. For this reason we shall keep referring to "pitch type" or "pitch data type" when it comes to the computer representation, even if it is technically a class; and use the term "pitch class" only for the music theory concept introduced by Allen Forte.
[7] www.computermusicnotation.com

It seemed to us that it would be an interesting feature for *bach* to include full support for just intonation, which is not just a variant of the widespread equal temperaments, but rather is based upon on a different approach altogether. We think that such a feature, which at the time of writing (October 2025) is functional in a pre-release version of *bach*, may be of interest to a growing community of composers and musical artists interested in working with alternative tuning systems.

2. A SHORT INTRODUCTION TO JUST INTONATION

The history of Western tuning systems is a fascinating and complex topic in itself, and we are certainly not attempting to tackle it in this article [2, 3, 4]. Still, from a modern, non-historical perspective, two influential ways of determining which pitches and intervals are afforded to musicians are equal temperaments and just intonation.

Equal temperaments (from now on, ET) take a reference interval and subdivide it uniformly in a number of steps. From a frequential point of view, if R is the size of the reference intervals, every one of the n steps corresponds to a frequency ratio $r = \sqrt[n]{R}$, such that when we pile up n of them — i.e., when we perform n consecutive multiplications of r — we get back R.

The reference interval R is often set to be the octave, corresponding to a $2/1$ frequency ratio. ET tunings based upon the octave are called *octaviant*.

Subdividing the octave into $n = 12$ equal steps provides the widespread 12-EDO tuning, with EDO standing for 'equal divisions of the octave' (also known as 12-TET, for 'twelve-tone equal temperament'). This tuning informs the vast majority of the music that we hear and instruments that we play in the Western world. In this case, $r = \sqrt[12]{2} = 1.059\ldots$ corresponds to a tempered semitone. It is important to notice that this is an *irrational* number, i.e., there is no way to obtain it as a proportion of integer numbers. By stacking semitones, i.e., by multiplying r by itself multiple times, we obtain every interval in the system. The traditional Western system adds on top of this a complex naming policy based upon the concepts of degree and alteration, but this is beyond the scope of this introduction.

Subdividing the octave into $n = 24$ equal steps gives the 24-EDO, i.e., the temperament of quarter-tones; with $n = 48$ we get eighth-tones; and so on. Of course, historically, not all values of n have the same relevance, and the profound reasons why the 12-EDO has emerged as a standard have to do with acoustics, psychoacoustics, algebra and instrument making, and are certainly beyond the scope of this article [5, 4, 2].

In any case, in principle, one can choose R and n freely. At least a few non-octaviant ET system are in relatively widespread usage, if only in the somewhat rarified community of tuning systems aficionados [6]. The most widely known one is probably the Bohlen-Pierce equal-tempered scale, with R set to 3 (the so-called 'tritave') and n to 13.

A profoundly different way of thinking about intervals, traditionally traced back to Pythagoras of Samos, is to notice that small-integer frequency ratios provide blueprints for the most consonant intervals. Physiologically and cognitively, this holds true only to a degree — consonance is certainly a many-headed beast that we are not going to tame here. You can refer to [7] for an insightful introduction. In any case, the choice of the octave as the reference interval for an ET system aligns with this observation — the octave $2/1$ being the 'most consonant' interval in this respect, after the unison $(1/1)$. This idea highlights the role of ratios such as $3/2$ (just fifth) or $4/3$ (just fourth), all corresponding to intervals borrowed from the lowest portion of the harmonic series. Systems based upon this kind of approach are often called *just-intonation* (JI from now on) systems.

In a very profound sense, the friction between equal *frequential* subdivisions (JI systems and the harmonic series) and equal *pitch* subdivisions (ET systems) is the foundational clash of worlds of Western music theory, as well as the driving force that has shaped many of the features of Western music. Among other things, one of the most prominent reasons why 12-EDO has emerged as a standard is that it approximates relatively well a subset of these small-ratio intervals, and particularly the fifth and fourth. On the other hand, its approximations of the just major third $5/4$ and the just minor third $6/5$ are suboptimal.

In a way, JI systems are simpler than ET systems since the calculation of frequencies only requires multiplications of rational numbers, whereas ET systems are based upon irrational frequency ratios.

It is worth noting that, if we extend the idea of ET system to a continuum of pitches (or logarithmic frequency), then any frequency and frequency ratio can be represented within it, including of course JI ratios. The case of 12-EDO representation is particularly relevant because we often use cents, that is, hundredths of 12-EDO semitones, as a sort of 'neutral' unit of measurement for pitch. Still, while representing JI pitches and intervals in terms of cents happens to be convenient, the cents value will generally be irrational, so any truncation will necessarily be approximated.

The opposite is *almost* possible: as any irrational number can be approximated to any desired degree of precision to a rational one,[8] then any ET pitch can be approximated to a JI one with an arbitrary degree of precision. Deciding which rational number makes a good approximation involves balancing simplicity and accuracy. For example, while $\pi = 3.14159\ldots$ can be approximated quite precisely by $31415/10000$, a classic approximation such like $22/7$ is arguably more expressive. Though less accurate, it is much simpler — or, in musical terms, more 'consonant', in the sense that it involves ratios between lower harmonics. (Such approximations are often derived using continued fractions, which we'll briefly touch on in Section 5.)

Combining steps of an EDO system is straightforward, as every step is identical and one can stack them up as LEGO bricks. JI provides a potentially infinite palette of bricks all with different heights, so building your cathedral requires further attention. This is to say that putting into practice

[8] In formal terms, this is to say that the set of rational numbers is *dense* in the number line.

the concept of just intonation is not without complications, as the combination of different harmonics can produce a potentially infinite number of variants of the same pitch.

For instance, consider the interval of major third C-E. This can be obtained in two ways: by stacking four perfect fifths (as in C-G-D-A-E) and adjusting by two descending octaves ($(3/2)^4/4 = 81/64$), or by associating it by definition with the pure ratio $5/4$ ($80/64$). In fact, virtually any interval can have different, equally reasonable variants — not to mention the "unreasonable", but not necessarily uninteresting, ones.

One way to curb this potentially overwhelming combinatorics is to generate tuning systems by stacking of a single interval, a bit like in the EDO case, but with this interval being a justly tuned one. The most prominent historical examples are Pythagorean tunings (used throughout the Middle Ages, and still informing the tuning of string instruments today), which generate all of their pitches as combination of pure fifths. Their E would then be at an interval of $81/64$ above C. Of course, this comes at the price of massively limiting the palette of just intervals at one's disposal. As a matter of fact, the 'natural' $5/4$ major third is 'more consonant' than the Pythagorean $81/64$ one, in that it correspond to a simpler ratio, but theoretically more complicated as well, because it requires a generator interval ($5/4$) higher up in the harmonic series, with respect to the octave ($2/1$) and the fifth ($3/2$).

When thirds became unavoidable building blocks of consonance, roughly during the Renaissance, just intonation system emerged that would borrow the E from the harmonic series, using this purer interval of $5/4$ above C. The two ratios $81/64$ and $5/4$ are close but not identical: their discrepancy, given by their ratio $81/80$, is known as 'syntonic comma'.

Theories of tuning are full of similar little discrepancies, called commas, each corresponding to the distance (or, more precisely, the ratio) between different variants of the same interval. Their implications with respect to the musical language are massive. One way to face them is to *temper* them, compromising the purity of integer ratios in order to achieve, or at least approach, some target characteristics — say, transposability, or enharmonic equivalence. Many different tempered systems have emerged in history, and the one that has today gained widespread usage, especially from the 19th century, is the twelve-tone equal temperament. It can easily be said that the harmonic language of many Western composers, from Schubert to Schönberg and beyond, requires by necessity, and was actually prompted by, its uniformity. [9] On the other hand, the aforementioned classical Western pitch naming policy, its notation system, and the inner core of the highly normative set of compositional rules are deeply rooted in pre-ET practice and theory, in a complex and often intimidating multi-layered construction.

Another posture is to accept the variants of JI, embracing the diversity of intervals that rational frequency ratios propose. Compositional interest in this approach resurged in the mid-20th century, through the seminal works, writings and teachings of composers such as Harry Partch, Lou Harrison and LaMonte Young, who started reconsidering the affordances of JI theory and tunings with respect to modernist music. This prompted the development of new theoretical frameworks and notational systems aimed at representing nuances that, in Medieval, Baroque and Renaissance music, were just outsourced to the preliminary choice of a specific tuning system.

Modern approaches to JI are not just a matter of local technical choices: the preference for JI over ET tends to be a defining and affirming fact for composers who embrace it, often identifying (or being identified) as JI composers, and not seldom has philosophical and even political undercurrents [8]. Within this context, tuning discrepancies are often seen as fundamental compositional devices and, as such, they require a notational framework to accommodate it. Intervals such as $9/7$ or $11/7$, for instance, are not something one finds in a classical music theory book, but they provide unique flavours that one can learn to appreciate. To borrow a famous Partch's metaphor, all these ratios lie before us like a "large table of aromatic and unusual viands" [9].

This is where modern-day JI notational systems come in: they provide detailed ways, usually through the redefinition of the frequency ratios associated with the white-key pitches and the introduction of special symbols and accidentals, to notate the different varieties of just intervals. In the following sections, we shall briefly describe a few of them.

2.1 Primes and limits

A crucial facet of just intonation is its connection with the Fundamental Theorem of Arithmetic: the numerators and denominators of a frequency ratio can be factorized, and this factorization gives readable insight on the type of harmonics (and intervals) involved. For example, $81/64 = 2^{-6}\times3^4$, and $5/4 = 2^{-2}\times5^1$. Larger primes correspond to higher harmonics, and the highest prime factor that shows up in an interval ratio is called its *limit*. For instance, $5/4$ is in the 5-limit, while $81/64$ is in the 3-limit. This means that it can be built by combining intervals up to the third harmonic (octaves and fifths), whereas, in order to build $5/4$, we need intervals up to the fifth harmonic (that is, we need to include pure major thirds) .

From a slightly different point of view, an interval can be defined by a list containing the exponents of its factorisation, so that $81/64$ is $[-6, 4]$ and $5/4$ is $[-2, 0, 1]$. Every element in the list corresponds to one of the prime numbers, in ascending order.

2.2 Notating just intervals

But one thing is to have a consistent mathematical representation for just intervals; another thing entirely is to have a way to notate them that does not collide (too much) with the traditional musical staff. This is clearly a bit of a bat-

[9] It seems to be established, on the other hand, that the tuning system for which Johann Sebastian Bach composed his *Well-Tempered Clavier* is not the modern 12-EDO, but a different one on whose details entire bookshelves of hypotheses have been written. Also, as a bit of trivia fact, 12-EDO is certainly not the longest-standing 'standard' — meantone temperaments enjoyed a much longer period of prominence.

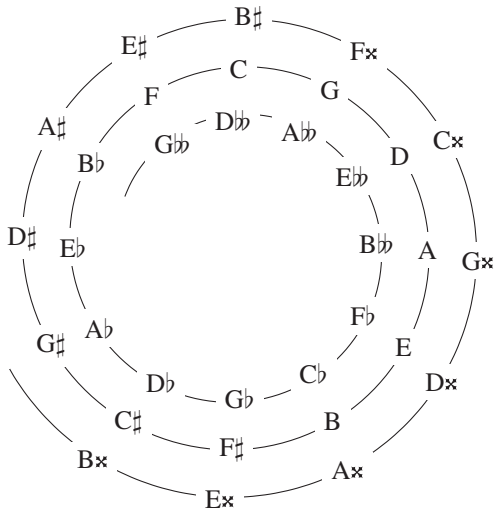


Figure 1. The Pythagorean spiral of fifths (enharmonic pairs, such as D♯ and E♭, refer to different frequencies).

tle against windmills, because of the foundational clash of worlds we mentioned above.

To find a compromise solution, a number of systems have emerged in the last few decades. A relatively popular one is Ben Johnston's notation, which consider 'white-key' pitches (C, D, E, F, G, A, B) as belonging to the pure diatonic scale of the Renaissance, and introduces accidentals to alter them by specific commas [10]. Another one is Sagittal notation, a comprehensive system for notating microtonal scales and tunings using arrow-like symbols [11]. A third system, and one that is arguably gaining traction at least in Europe, is the Extended Helmholtz-Ellis JI Pitch notation (or HEJI for short) [12]. [10] Due to its consistency and relative simplicity, it is the notational system that we have decided to embrace in *bach*.

Differently from Johnston's notation, HEJI considers all 'white-key' pitches (C, D, E, F, G, A, B), as well as any of their alterations with sharps and flats, as Pythagorean, i.e., organised in a spiral of 3/2 fifths (see Figure 1).

In other words, the core elements of the system are, by default, in the 3-limit. For instance, HEJI considers E_5 as $81/64$ above C_5. In order to vary the interpretation of intervals, HEJI introduces specific accidentals, each representing a comma. The crucial connection is that every prime number comes with a specific comma that concerns it.

The comma for the 5-limit is the syntonic comma $81/80$, and its graphical HEJI sign corresponds to adding an upward or downward arrow to the existing Pythagorean accidental (or to a natural, if no accidental was provided). Of course, raising by two or more syntonic commas is also possible, and corresponds to adding more arrows. This means that we can represent the $5/4$ E (with respect to C) in HEJI starting from the Pythagorean E and lower it by one syntonic comma. But nothing prevents us to for example lower it by one more syntonic comma, or to raise it by any number of syntonic commas instead.

There is also a comma for the 7-limit, called septimal comma, whose accidental is inspired by the digit 7; it amounts to $64/63$, that is, the discrepancy between a Pythagorean minor seventh and a harmonic seventh (the interval between the 4th and 7th partial of a harmonic series). There is a comma for the 11-limit ($33/32$), which, due to its proximity with the tempered quarter-tone, is called undecimal quarter-tone; and so on. Since any positive fraction can be factorised uniquely, any frequency ratio can be expressed uniquely as a combination of a note name with optional sharps or flats (corresponding to the factors 2 and 3) and a set of commas (introducing any higher prime). For instance, the quite exotic fraction $15309/8800$, representing yet another kind of minor seventh, can be decomposed as

$$\frac{15309}{8800} = 2^{-5} \times 3^{7} \times 5^{-2} \times 7^{1} \times 11^{-1}$$

but by introducing the necessary denominators and numerators we can change every one of the primes beyond 2 and 3 into its respective comma. Now, each comma is the discrepancy between a given prime-indexed harmonic, shifted by one or more octaves (and thus multiplied by a power of 2), and the nearest Pythagorean pitch (whose factors are, by definition, powers of 2 and 3). Therefore, changing a prime into a comma requires us to counterbalance the exponents of the Pythagorean part of the expression. After some number juggling, we are left with a decomposition into a Pythagorean part (the 16/9 fraction) and a sequence of commas:

$$\frac{15309}{8800} = \frac{16}{9} \times \left(\frac{81}{80}\right)^{2} \times \left(\frac{64}{63}\right)^{-1} \times \left(\frac{33}{32}\right)^{-1}.$$

HEJI features commas, and accidentals, for limits up to prime 47 (the first few are displayed in Figure 2). The mathematics is made easier by the fact that all these commas only involve a single prime, in addition to powers of 2 and 3.

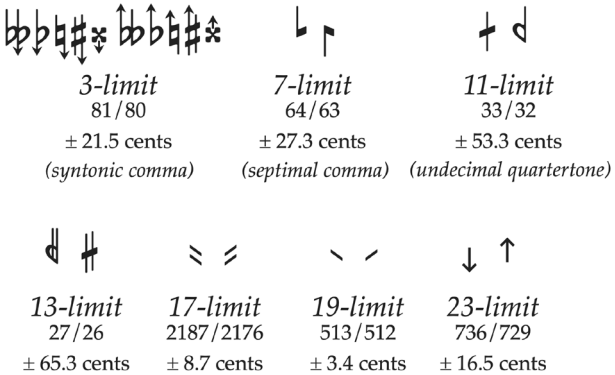


Figure 2. HEJI commas, for the first few primes, along with their accidental representations. The deviations expressed in cents are approximations of irrational values.

As an example, our interval of ratio $15309/8800$ would

[10] Whereas Ben Johnston's notation was indeed invented by Ben Johnston, the Helmholts-Ellis system was inspired by the works of Hermann von Helmholtz and Alexander Ellis, but actually developed in the early 2000s by Marc Sabat and Wolfgang von Schweinitz.

be notated as in Figure 3, while Figure 4 shows the first few elements of a harmonic series written in HEJI notation.

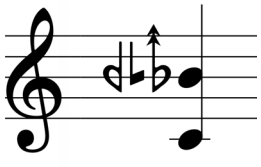


Figure 3. HEJI representation of the interval $15309/8800$: starting from a Pythagorean minor seventh, we raise by two syntonic commas, and lower by both one septimal comma and an undecimal quarter-tone.
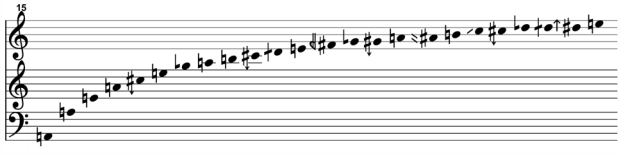


Figure 4. A portion of harmonic series displayed in HEJI notation. Increasingly more unusual accidental signs appear as the series goes higher, as higher prime factors require higher-order commas.

3. PITCHES IN THE *BACH* PACKAGE FOR MAX

bach is a software package aimed at augmenting the Max programming environment with advanced musical representation capabilities [13]. At its forefront are two musical notation editors and sequencers called bach.roll and bach.score, both relying upon a set of custom data types added to Max by the package, namely rational numbers, multi-level lists (*llll*s in the *bach* jargon) and musical pitches (more about these later). The two notation objects are complemented by more than 200 other modules dealing with a wide array of operations, ranging from basic ones (such as arithmetic operations and elementary list processing on the new data types) to advanced ones (such as rhythmic quantisation and the solution of constraint satisfaction problems). *bach* is free and open source software; it was conceived in 2010 as a personal project and has been constantly developed by the authors of this article ever since. As of today, we estimate that it has some thousands of users worldwide.

3.1 Pitch representation and operations

As hinted at before, one notable feature of *bach* is the fact that it treats musical pitches as a full-fledged data type. In the current (at the time of writing) publicly distributed version of the system, a pitch is represented internally through the following components:

- an integer representing the white-key class, from 0 (C) to 6 (B);
- a rational number representing the chromatic alteration with respect to the white key, measured in whole tones or fractions thereof;
- an integer representing the octave, according to the convention by which the middle C is located at the beginning of octave 5. Such relatively unusual convention has been chosen for simplicity's sake with respect to the MIDIcents representation: in this way, the octave of a pitch expressed in MIDIcents can be calculated simply through an integer division by 1200. (For the reasons behind this seemingly idiosyncratic choice, as well as many more details about all the information contained in this section, see [14].)

So, for example E♭$_6$, that is, the E♭ in the topmost space of the treble clef, will be represented through white-key class 2 (E), alteration $-1/2$ (one chromatic semitone lower) and octave 6. It should be remarked that D♯$_6$, that is, an enharmonic pitch corresponding to the same key on a piano keyboard (and hence to the same MIDI note) is represented differently from E♭.

One could rather see pitches as intervals counted from C$_0$: in this sense, G$_0$ represents not only a very low G, but also a perfect fifth. This makes arithmetic operations on pitches conceptually meaningful. Sums represent the stacking of intervals (stacking a major third and a minor third produces a perfect fifth, thus E$_0$+E♭$_0$ = G$_0$) as well as transposition (transposing D$_4$ a fifth above yields A$_4$, thus D$_4$+G$_0$ = A$_4$). Multiplying a pitch by an integer represents stacking an interval on top of itself multiple times (three stacked major thirds yield an augmented seventh, so E$_0\times 3$ =B♯$_0$). Dividing a pitch by an integer means dividing an interval in equal parts — which sometimes is exactly possible and other times can only return an enharmonic equivalent to the result: for example, B♯$_0/3$ =E$_0$, as per the previous example, but C$_1$ (which is enharmonic to B♯$_0$) divided by 3 does not have a solution proper, since there is no interval that, stacked three times, forms an octave. [11] Yet, taking inspiration from what happens with integer division, where (save of course for the case of division by 0) a result is always defined even if it entails approximation, as in $7 \div 3 = 2$, we implemented pitch division so that it always produces a result, with a potential enharmonic approximation: so, C$_1/3$ will anyway return E$_0$. A corresponding pitch modulo operation returns the remainder of such division, always an enharmonic equivalent of the unison. Multiplication and division of a pitch by rational numbers then immediately follow as combinations of integer multiplications and divisions. Quotitive division of a pitch by a pitch is also possible, telling how many times a given interval contains another — and always allowing for enharmonic approximation when needed. For all the operations and mathematical functions that do not have a specific meaning with respect to pitches, the pitch is preliminarily converted into MIDIcents. This ensures that, for example, summing an actual pitch and a value in MIDIcents (as in C$_5$+500) produces a meaningful value (in the example, 6500). [12]

[11] Of course this implies that, in agreement with classical Western music theory, we are considering enharmonic distinctions in the picture, something that sometimes escapes more informal discourses about intervals.

[12] The fact that, in the example in parentheses, the return value of the

Pitch comparisons also deserve a brief explanation. Equality is trivial: two pitches are considered equal if all their components are equal. So, B♯$_4$ is considered different from C$_5$, although they are enharmonically equivalent and both correspond to 6000 MIDIcents. Converting to MIDIcents and comparing for equality is also possible: in this case, the two pitches will be considered equal. As for inequalities, the ordering is always firstly defined by the octave, subsequently by the white key if the octave is the same, and finally alterations are taken into account only if the two previous values are the same. For example, if we compare F♭$_0$ with D×$_0$, the former will be considered greater, because the white key is. On the other hand, if those pitches are converted to MIDIcents, then the second one is greater, as it corresponds to 400, while the first is 300.

Besides the arithmetic operations, *bach* features a number of other modules dealing with pitches, for finding enharmonic equivalents, constructing computationally pitches from various kinds of data and so on. Generally speaking, the easiest way to represent a pitch in *bach* is just through a text symbol resembling the ones above, and composed by a mandatory white-key name (`A`, `B`, `C`, `D`, `E`, `F`, or `G`), an optional sequence of accidentals expressed through specific characters (`#` for sharp, `b` for flat, `q` for one quarter-tone up and so on, covering ascending and descending chromatic semitones, quarter-tones and eighth-tones), a mandatory octave number and an optional further deviation, expressed in tones or fractions thereof, followed by the letter `t`. Examples of valid pitch symbols (that, as stated above, can also be considered as the representation of intervals corresponding to their respective distances from C$_0$) include the following: `C3`, `D#4`, `Fq0` (F$_0$ plus one quarter-tone), `G#v6` (G♯$_6$ minus one eighth-tone), `Bbb^5-3/19t` (B♭♭$_5$ plus one eighth-tone minus 3/19 of a tone).

Besides the actual pitch data type, all the objects dealing with notation also accept and understand MIDIcents.

3.2 Limitations of the *bach* pitch representation

The pitch system of *bach* discussed above is essentially focused on the representation of equal-tempered, octaviant tuning systems. More specifically, it is straightforward to express through it semitones, quarter-tones and eighttones, since there are specific accidental symbols for them. It is not difficult to express any other kind of division of the semitone, by specifying the rational deviation from a reference pitch through the `t` part of the pitch symbol. Moreover, one can easily define custom structures of MIDIcents associated with correspondingly custom structures of frequencies. Formally, because there is a relatively simple and universal definition of the mathematical relation between cents and frequency, one should assume to define their custom MIDIcents structure so as to reflect the desired frequency structure, and vice versa. Practically, if one considers the MIDI representation as a numeric encoding of the layout of a physical instrument, then it can be perfectly meaningful to elude this mathematical relation altogether.

In fact, one can almost as easily associate note names and accidentals with arbitrary frequencies, conceptually not unlike re-tuning a piano to whatever collection of fundamental frequencies one may fancy. As long as a concept of octave and a subdivision in 12 non necessarily equal steps is preserved, such an operation poses no special problems, although it may as well be that processes often given for granted (such as transposition) cannot be meaningfully applied any longer. [13] It should also be mentioned that such an operation would require the user to choose univocally between, say, C♯ and D♭: if the black key between C and D is tuned as a C♯ (calculated, for example, as a 5:4 ratio starting from A), then D♭'s (calculated, for example, as a 4:5 ratio starting from F) are out of the game, and vice versa. For this reason, the whole pitch system of *bach*, with its accounting for virtually infinite enharmonic distinctions, [14] is rendered useless and the user is better off to stick to MIDIcents, since a value of 6100 (the black key placed at the immediate right of the middle C) only has a single possible interpretation in this scenario. Accounting for more keys in an octave, as in split-sharp harpsichord keyboards, would not change the gist of the reasoning, but only require more MIDIcents values mapped to frequency values within an octave.

Still, this is only a part of the representation problem: the other part is semantic, and has to do with the inherent relations and affordances that are built into a pitch structure, and the operations one can apply to it. In this sense, the more one strays away from the aforementioned kinds of division of the octave, the more the representation becomes cumbersome and problematic, at least within the context of *bach*. Dividing the octave in a number of steps that is not a multiple of 12 (that is, creating an octaviant ET system that is not based upon the semitone) delegitimates both note (and accidental) names and the diatonic staff representation. Non-octaviant ET systems delegitimate the octave concept altogether. Moreover, in such scenarios all the pitch operations described above lose their meanings. Established systems such as ET Bohlen-Pierce generally have their naming schemes and notational practices, but custom ET systems, that can be defined by just plugging in the equation arbitrary generator ratio and num-

sum is 6500 rather than F$_4$ is linked to the problem of enharmonicity: whereas intuitively we may think that 500 MIDIcents and F$_0$ (and hence a perfect fourth) are equivalent, they are not: in an enharmonically aware system, 500 MIDIcents may as well be E♯$_0$ (an augmented third), G♭$_0$ (a doubly-diminished fifth) and infinitely many more enharmonic variants.

[13] One specific and very relevant example of this scenario is the tuning of keyboard instruments for music predating the ET era. Most of the times, the keyboards of such instruments featured an arrangement of keys exactly identical to the one of the modern piano, with all the embedded knowledge of a diatonic seven-note scale punctuated by a specific structure of five altered pitches and repeating at distances of octaves. As hinted at above, such instruments can be tuned to a plethora of different kinds of tunings, something that historically informed harpsichord players actually do all the times, according to the specific repertoire. Once the instrument is tuned, though, it can only play in a few selected keys, and the modern concept of transposition, by which a skilled pianist can just move any piece of music up and down the keyboard in whatever tonality she or he likes, is just inapplicable, as the same operation on an instrument with such a tuning will almost invariably result in horrendously dissonant chords and melodies.

[14] Technically, due to the limited precision of numeric representation in computers, the amount of enharmonic possibilities — and of pitches altogether — in *bach* is not infinite. On the other hand, it is hardly imaginable that in a realistic musical scenario the limits of their potential quantity are hit, or even just approached.

ber of steps, are even more problematic: naming systems notations addressing broader families of tunings have been proposed, but their implementation would be difficult to say the least. The ideal solution would be to provide users with ways to implement their own pitch grammars, arithmetics and notation system, something along the lines of MaxScore's *scorepions* [1], but such a device would risk to pose other problems, especially in terms of efficiency. The simple truth is that, as of now, *bach* is not currently well-suited to such kinds of scenario.

4. JUST INTONATION PITCHES IN *bach*

The kinds of problems with generalised ET systems that have been discussed above are magnified when it comes to representing generalised JI systems. As hinted at above, one specific historical just tuning system may as well be implemented as a specific MIDIcents-to-frequency mapping. On the other hand, the generalisation of just intonation systems, with their arbitrarily expanding series of commas and potential interpretations of the same note names and piano keys, poses substantially different problems from the ones that are faced in ET representation.

These considerations, combined with a personal fascination with the aesthetics and formalism of just-intonation music, has prompted us to embark in the experimental project of supporting generalised just intonation through the HEJI system described above. [15]

The implementation of just intonation has required a thorough rethinking of the pitch data type and its corresponding operations. Whereas a preliminary hypothesis was to keep the extant pitch type unchanged and add a further type for JI, careful reflection prompted us to embed the two representation in a single data type. Although this has caused some additional complexity in the codebase, it also poses great practical advantages that will be discussed further on.

4.1 Textual syntax: curly brackets

The fundamental principle is that, at the lowest level, a pitch in *bach* is composed of two separate parts — an equal-tempered part (already described in [14]) and a justly tuned part. Both parts can be represented with letters, accidentals and octaves, and to distinguish between them we use curly brackets. For instance, `E5` is the tempered E, 400 cents away from middle C, while `E{}5` is about 408 cents away from middle C, or more precisely a $81/64$ ratio, corresponding to a Pythagorean major third. We shall discuss now the JI part, and leave for a further section an explanation of how the combination of ET and JI works.

C's have a special status of 'amphibian' entities, in that they constitute the reference point for both the *bach* ET and JI systems: `C0` corresponds to 0 cents. Similarly, `C0` as a justly tuned pitch has an interval of $1/1$ from the equal-tempered `C0`, so it corresponds to it. `C1` is both equal tempered and just, with an interval of $2/1$ (octave), and so on. All C's will be both equal tempered and just so that `C5` and `C{}5` are exactly the same thing: *bach* will understand both representations, but will always output the former.

Working with ratios with respect to `C0` is of course consistent, but perhaps cumbersome operationally, and one can change the reference pitch in several ways. A powerful one is by writing an explicit frequency ratio in terms of a rational deviation; for instance, to get a pitch a pure major third above `G{}5`, one can write `G{}5+5/4r`. As a matter of fact, one can consider the `r` part as the non-logarithmic counterpart of the aforementioned `t` component representing the alteration expressed in terms of fractions of a tone (e.g., `E5+7/12t`). The `r` part can as well appear on its own, as a simple way to enter JI pitches directly in terms of their generator fraction. Thus, writing `48r` is equivalent to writing `G{}5`, that is, $48 = 2^4 \times 3$, that is, 4 octaves plus one twelfth from C0.

Notice that `G{}5+5/4r` does not correspond to the Pythagorean `B{}5`, but actually differs by it precisely by one descending syntonic comma. A more convenient way to notate this specific note is as `B{-1}5`: we are starting to use the space between curly brackets to account for prime commas, and the first term that appears corresponds to the 5-limit deviation, measured in number of syntonic commas ($81/80$). Similarly, a pure major third over middle C we find `E{-1}5`.

One can introduce additional commas, each corresponding to a further prime number, by concatenating them via colons. For instance, a harmonic seventh (the interval between the 4th and 7th partials of a harmonic spectrum) above middle C we find `Bb{0:-1}5`, where 0 signals that there are no syntonic commas, and -1 stands for the lowering by a septimal comma ($64/63$). This corresponds to the decomposition of $7/4$ into a Pythagorean and a septimal part, whose exponent is precisely -1:

$$\frac{7}{4} = \frac{16}{9} \times \left(\frac{64}{63}\right)^{-1}.$$

Of course one can lower or raise by more than one comma at a time; and you can pile up commas up to the 47-limit, which is the highest prime for which the newest version of HEJI specifications has an official representation [12]. For instance, the $7/4$ ratio raised by an undecimal comma (almost a quarter tone) gives `Bb{0:-1:1}5`.

4.2 Just Intonation Arithmetics

An arithmetic system for JI pitches, parallel to the ET one, has been implemented. The fundamental concept that lies at the heart of JI arithmetics is that any possible representation of a JI pitch is equivalent to a rational number expressing the frequency ratio: the various different representations (exponents of prime factors; note name with optional commas and sharps or flats; Helmoltz-Ellis, Sagittal or Johnston's; or any combination of those) are just different ways of writing that rational number. With this in mind, we can briefly discuss the main principles of JI arithmetics in *bach*.

[15] Alongside just intonation, the forthcoming version of *bach* introduces a third notation type, with a linear vertical pitch domain, as in a true piano-roll scenario where all the semitone steps have the same graphical distance. This notation type, which will not be discussed further here, is meant among other things to enable more flexible display of non-standard tuning systems.

- Sums and subtractions of JI pitches produce transpositions, thus they amount to respectively multiplying and dividing the corresponding frequency ratios.

- Multiplications and divisions of a JI pitch by an integer or a rational make more sense if the pitch is thought of as an interval. They amount to raising the frequency ratio of the pitch to respectively the integer or rational, or its opposite. This corresponds respectively to stack multiple JI intervals on top of each other, and to calculate how many times a given interval contains a different one (for example, `D{}1 / G{}0` returns 2, because a Pythagorean major ninth is composed exactly of 2 Pythagorean fifths).

- Equalities and inequalities are calculated by simply comparing the frequency ratios. Importantly, unlike with ET enharmonics, the result always corresponds to that of the comparison of the corresponding MIDIcents values.

- For all the other possible operations, the pitch is preliminarily converted into MIDIcents.

Through either direct expression or arithmetic operations, it is possible to generate a JI pitch that should in principle exceed the 47-limit that is hard-coded within the *bach* system. Because such a pitch cannot inherently be represented, there is an approximation mechanism that comes into play in this situation: the desired pitch is approximated to one with a smaller limit (47 by default, but a different one can be chosen), within a range of error that can be specified in MIDIcents (67 by default, roughly corresponding to one third of a tone).

4.3 Internal Representation of Pitches

As mentioned above, in the forthcoming version of *bach*, all pitches are actually composed of two parts: an ET one and a JI one. The ET part is represented internally in terms of the white-key class and a rational alteration. The JI part is represented internally in terms of its corresponding frequency ratio with respect to C0. The two parts share the octave component, because the octave is the only interval whose corresponding frequency ratio is the same, and as such it would make no sense to treat it separately. In fact the octave of any pitch is always stored internally within the JI part of the pitch. In the simplest cases, the exponent of the 2 factor actually coincides with the octave. On the other hand, this is not true in general: for instance, a just G_0 has a ratio of $3 : 2$ (or $3^1 \times 2^{-1}$), because the numerator 3 actually pushes the pitch to octave 1, and this must be compensated by the denominator 2 so as to bring back the pitch to octave 0.

Unlike the standard *bach* rational numbers, which are stored internally as a simple pair of integers, the rational representing the JI component of a pitch is stored in terms of a vector of exponents of the first 15 primes, from 2 to 47. This vector is precisely the vector we mentioned at the end of Section 2.1. The first 7 primes, from 2 to 17, are allocated 8 bits each, allowing for exponents from -128 to 127; the subsequent 8 primes, from 19 to 47, are allocated 4 bits each, allowing for exponents from -8 to 7.

This peculiar representation is motivated by the fact that the whole data structure for a pitch should fit into 16 bytes; one byte is taken by the ET white key [16]; four bytes are taken by the ET alteration rational number (we deemed that reducing the alteration to a two-byte rational number, with 8 bits each for numerator and denominator, was not enough for the precision that is expected from the system); this leaves 11 bytes for the JI part.

On the other hand, the JI part is not expected to be just any rational number whose numerator and denominator fall within a given range, as is the case for standard *bach* rationals as used, for instance, for the ET alteration. Rather, frequency ratio are conceived and computed in terms of the exponents of their prime factors. This means that the range of their numerator and denominator is potentially very wide (each would require more than 300 bits to be represented as an integer), but also extremely sparse, in that very few values over the range can be obtained as a product of such primes with exponents the allowed range described above. Thus, representing the ratio precisely in terms of its prime factor exponents is extremely more space-efficient than employing a standard rational, and hopefully not much more expensive in computational terms. If anything, the basic mathematical pitch operations are simpler: summing pitches, that is, multiplying ratios, actually amounts to summing 11 bytes of exponents rather than multiplying an almost 80-byte rational (more than 300 bits for the numerator and as many for the denominator) and then having to simplify the fraction. The same line of reasoning goes for pitch multiplication. As a matter of fact, albeit technically possible, a standard rational representation of JI ratios within the desired 47-limit would have required very complicated and inefficient workarounds.

The choice of the ranges for the individual exponents, and specifically the fact that lower prime factors exponents have a greater range that higher ones, has been motivated by space considerations, and by the observation that low primes such as 2 and 3 practically require a relatively wide range of exponents (a HEJI $A\sharp_0$, for instance, is $2^{-15} \times 3^{10}$), whereas prime factors higher than 13 or 17 are very seldom used, and if they are they probably have very small exponents.

An important check that is carried out at the beginning of virtually every operations on a pitch is whether the concerned pitch is ET-pure or JI-pure, that is, respectively, if the JI exponent vector is 0 for all the values except optionally the first (which means that only the octave JI component is present); and if the ET white key and alteration are both 0. Since, in practice, most pitches are expected to be ET- or JI-pure, this check is specifically meant to avoid the extra computational burden of dealing with the complexity of the JI part for pitches that do not have one, save for the octave. Among other things, ET-pure pitches always have their octave coinciding with the exponent of the 2 factor,

[16] Actually, 3 bits would be sufficient for representing 7 possible values, but using the remaining 5 bits for other purposes would have required deeper and more extensive and dangerous changes to the whole *bach* codebase than we were willing to face.

thus making their manipulation even easier. As mentioned above, C's are exceptional in that they are both ET-pure and JI-pure.

4.4 Mixing just and equal-tempered parts

The ability of expressing non-pure pitches has been added in order to keep the *bach* type system simpler; to avoid a proliferation of object attributes to distinguish between the types of accepted and returned pitches; and, more importantly, to allow for the specific musical problem of setting the reference for individual JI pitches.

Even more than ET pitches, JI pitches are inherently intervals. A frequency ratio is not a frequency: it is a deviation from a reference frequency, corresponding to a reference pitch. The reference pitch may as well be a JI pitch, but at some point, as in the myth of the turtle supporting the world, an arbitrary, non-relative pitch must be given. In terms of the *bach* world, this means that this base reference must be an ET pitch. The choice of C_0 is the most elegant, because it is both an ET and a JI pitch, and also because it corresponds to 0 MIDIcents, which is a natural starting point. On the other hand, it is not the only possible choice. To express, for example, a pitch located a just fifth above A_3, we can simply write `A3+G{}0`. In fact, entering a sum or a subtraction of an ET and a JI component will result in a non-pure pitch. Incidentally, when the expression is parsed and the pitch is generated and stored, the octave number that here is attached to the ET part will be summed to the one attached to the JI part (0 in this case) and to the optional frequency ratio, and stored as the exponent of prime 2 of the JI part.

Arithmetics of non pure-pitches works according to the same rules as for pure pitches, with the two parts computed separately and then summed together again. On the other hand, the two parts are not fully independent as they share the octave specification. This means that the same pitch could be represented for instance as `A3+G{}0`, `A0+G{}3` or even `A1+G{}2`. All those writings are actually equivalent and will have the same internal representation. When required to output a textual representation of non-pure pitches, *bach* will follow a standard form by which the octave number is attached to the ET part except in the case of JI-pure pitches.

Non-pure pitches can be also leveraged to solve elegantly specific musical problems, such as representing the harmonic series of a given ET pitch such as `A3` — something that can be expressed as, for instance, the sequence `A3`, `A3+2r`, `A3+3r`, `A3+4r` and so on.

5. AN ECOSYSTEM FOR JUST INTONATION

Just intonation comes with a set of objects dedicated to make it work properly.

There are two objects that can be seen as Swiss-army knives for pitches. One is aptly called bach.pitch, and it performs various kinds of conversion, query, packing and unpacking for pure and non-pure pitches. It can, among other things, separate the individual components of a pitch (white-key class for the ET and JI parts, ET alteration, JI commas, octave); convert a pitch in MIDIcents; and retrieve the frequency ratio or exponent vector for a JI pitch. Also, it can build pitches starting from all those separate components.

The other general-purpose object is bach.eval. This is a very advanced object, implementing a small functional programming language called *bell* and meant to facilitate the expression of complex algorithmic problems in *bach* [15]. Among its features, *bell* provides a set of functions operating on pitches and reproducing within the context of the language the various possible behaviours of bach.pitch, and more. As a matter of fact, *bell* is probably the most advanced and comprehensive way to deal algorithmically with the complexity of pitch representation.

bach.eval can perform all the arithmetic operations mentioned above, but they are also accessible from simpler dedicated modules such as bach.+, bach.- and so on, as well as the bach.expr object, which is a general evaluator for mathematical expressions. [17]

A way to generate sequences of rational numbers is the bach.fareyser module, which produces Farey series, i.e., ordered sequence of reduced fractions with denominators less than or equal to some number. These sequences are often used by composers interested in just intonation and spectralism as 'reservoirs of consonances'.

Another important facet of dealing with rational numbers is to handle their approximations. The *bach* package already includes bach.approx, approximating to equal tempered grids, but approximating just intonation ratios is much trickier as rational numbers are dense in the real number line. To this end, we have included bach.continuedfraction, computing rational approximations by using the convergents (and, optionally, semiconvergents) of continued fractions. This object is not just useful to approximate rational numbers with simpler ones, but rather any floating point number, and indeed approximations of irrational number have profound significance [16], as exemplified in Figure 5. [18]

In order to explore Partch's "large table of aromatic and unusual viands", we have added a brand new object, called bach.jiwheel, where one can explore pitches and intervals interactively by zooming and selecting. The bach.jiwheel object shows JI pitches up to a given limit arranged in a circular fashion, and returns visual information about their graphical representation and mathematical properties.

Finally, one problematic aspect with tunings other than 12-EDO is the difficulty of handling them in MIDI-based contexts. For a long time, the only practical way to play even just quarter-tones was to set up clumsy contraptions of MIDI channels and pitch bend: this is, among other things, how microtonal playback works in OpenMusic, and how it is implemented in the bach.ezmidiplay module.

Luckily, in recent time, a number of microtonal MIDI protocols have emerged to ease this process. In partic-

[17] Actually, the simpler arithmetic modules are wrappers for bach.expr.

[18] The *bach* package also already included the bach.float2rat module, designed for rational approximation of floating point numbers, according to different types of criteria, such as error thresholds and maximum denimonators.

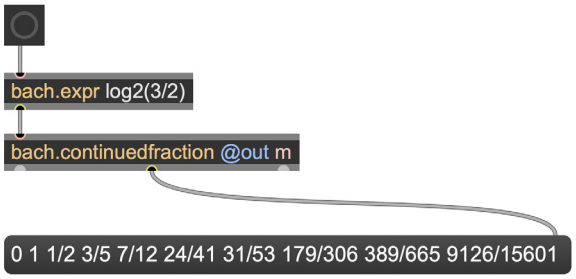


Figure 5. The continued fraction convergents for $\log_2 3/2$ are profoundly connected with equal temperaments that best approximate the fifth $3/2$. You will notice, in particular $7/12$ (corresponding to the common 12-EDO, whose perfect fifth is on the 7th step) and $31/53$ (corresponding to 53-EDO, whose fifth is on the 31th step).

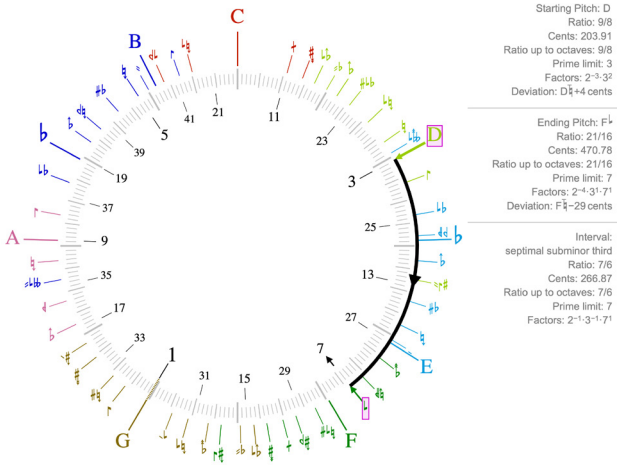


Figure 6. The bach.jiwheel object provides an interactive way to explore just intonation pitches and intervals, while monitoring their mathematical relationships.

ular, the MTS-ESP protocol [19] is becoming a standard, and it is now implemented in a large number of plugins and applications. To take advantage of it, we have endowed bach.playkeys (the object responsible for extracting and formatting playback information from the output of notation editors) with the ability to act as a MTS-ESP master, handle all microtonal adjustments internally, and communicating them on a note-by-note basis to MTS-ESP-compliant plugins. This allows seamless microtonal playback of *bach* scores with very little effort on the user's side. There are inevitable limitations anyway, due to the fact that all these are essentially hacks on the MIDI protocol, and there may be situations in which a note cannot be reproduced. On the other hand, Max allows one to build their own sound generation modules directly in the environment. Users willing to do so will be able to overcome all the limitations of MIDI.

[19] https://oddsound.com

6. EXAMPLES

As a first, simple example of application, consider the so-called 'Benedetti's puzzle': a looping progression of four chords that, when performed in rigourous JI, prompts the tuning to raise by a syntonic comma every time [17]. The patch in Figure 7 shows how, starting from the initial cell, one can build this comma-raising sequence just by using bach.+ C{1}0.

As a second example, consider the patch in Figure 8, which uses the concept of Farey sequence (also connected to mediants) to produce a descending sequence of intervals, from the octave to the unison, along with the fundamentals that their respective harmonic series underpin.

7. CONCLUSIONS AND FUTURE WORK

As mentioned above, at the time of writing the pitch system is fully implemented and is undergoing an extensive testing phase. We will hopefully be able to publicly release the new version of *bach* before the end of 2025.

This being said, the current implementation of JI pitches can be seen as the starting point for a much ampler work on supporting more features related to pitch representation and tuning systems. One interesting addition could be supporting Scala files. On the other hand, those are already supported by Max's native objects mtof and ftom, and their very nature — just a list of cents or frequency ratios — makes them only adapted to the basic way of dealing with tuning systems other than 12-TET through, so to speak, "retuning the piano strings", that is, associating arbitrary MIDIcents values to the data describing each specific scale or tuning.

In order to complement this limitation, we could add native support for at least a few established non-standard tunings such as the aforementioned Bohlen-Pierce. On the other hand, the notation editors should integrate the respective accidental sets and general paradigms of behaviour; and, even more importantly, the *bach* type system should be enhanced so as to accommodate these additional tunings — something probably not impossible, but surely rather complicated.

Finally, *bach* is at the core of a wider ecosystem of Max packages, also including *cage* (for high-level musical processes), *dada* (for creative graphical interfaces and data corpus management) and *ears* (for non-realtime algorithmic audio processing). It would definitely be useful to at least include in cage some new tools for making the usage of the admittedly complex new pitch system easier and less intimidating.

8. REFERENCES

[1] G. Hajdu and N. Didkovsky, "Maxscore: Recent developments," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'18*, S. Bhagwati and J. Bresson, Eds. Montreal, Canada: Concordia University, 2018, pp. 138–146.

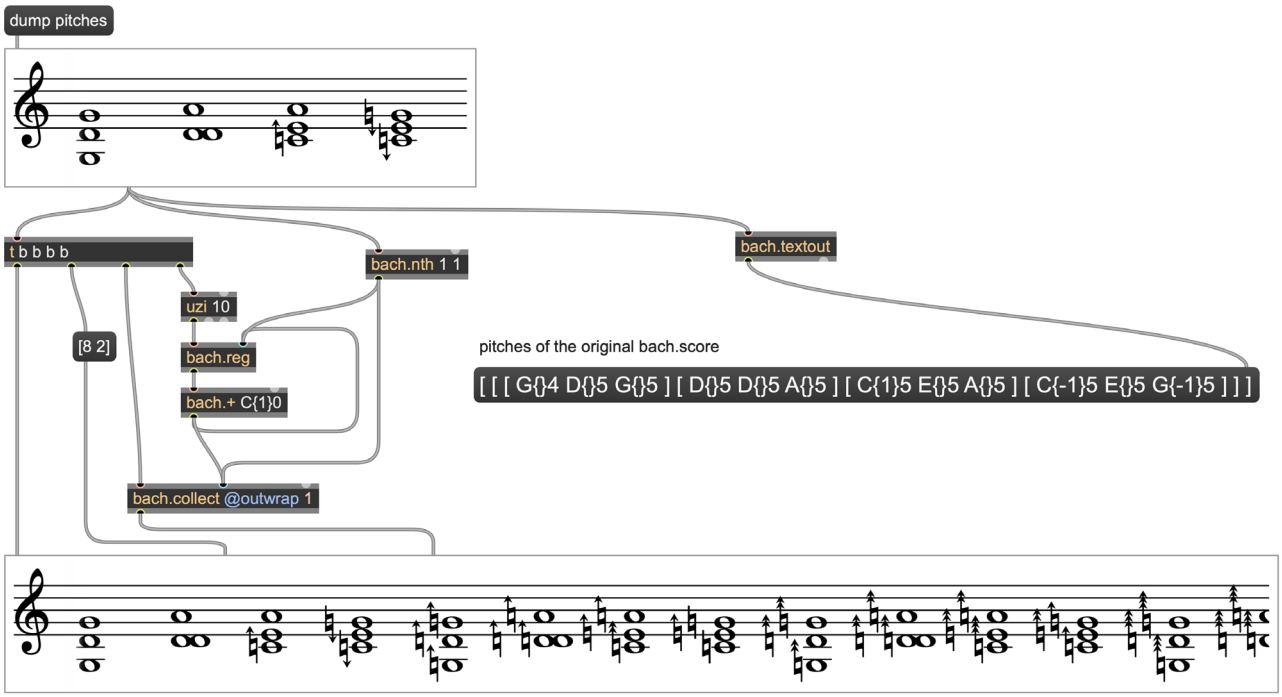


Figure 7. So called 'Benedetti's puzzle' in *bach*'s just intonation: in order to maintain the purity of the harmonic intervals, every repetition of the phrase must be shifted up by one syntonic comma.

[2] J. M. Barbour, *Tuning and Temperament: A Historical Survey*. Michigan State College Press, 1951.

[3] K. Gann, *The Arithmetic of Listening: Tuning Theory and History for the Impractical Musician*. University of Illinois Press, 2019.

[4] P. Barbieri, *Tuning and Temperament: Practice vs Science. 1450-2020*. Gangemi Editore, 2023.

[5] D. Tymoczko, *A Geometry of Music: Harmony and Counterpoint in the Extended Common Practice*. Oxford University Press, 2011.

[6] D. Blackwood, *The Structure of Recognizable Diatonic Tunings*. Princeton University Press, 1985.

[7] J. Tenney, *A History of 'Consonance' and 'Dissonance'*. Excelsior Music, 1988.

[8] P. Oliveros, "My "american music": Soundscape, politics, technology, community," *American Music*, vol. 25, no. 4, pp. 389–404, 2007.

[9] H. Partch, *Genesis Of A Music: An Account Of A Creative Work, Its Roots, And Its Fulfillments (2nd Edition)*. Da Capo, 1979.

[10] K. Gann. (2018) How to Use Ben Johnston's Just Intonation Notation. [Online]. Available: www.kylegann.com/BJNotation.html

[11] G. D. Secor and D. C. Keenan, "Sagittal: A microtonal notation system," *Xenharmonikôn, An Informal Journal of Experimental Music*, vol. 18, 2006.

[12] M. Sabat, *The Extended Helmholtz-Ellis Pitch Notation*. Plainsound Music Edition, 2005.

[13] A. Agostini and D. Ghisi, "A max library for musical notation and computer-aided composition," *Computer Music Journal*, vol. 29, no. 2, pp. 11–27, 2015.

[14] ——, "Pitches in bach," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'18*, S. Bhagwati and J. Bresson, Eds. Montreal, Canada: Concordia University, 2018, pp. 128–137.

[15] A. Agostini and J.-L. Giavitto, "Bell, a textual language for the bach library," in *Proceedings of the 2019 International Computer Music Conference*. Michigan Publishing Services, 2019.

[16] D. Benson, *Music: A mathematical offering*. Cambridge University Press, 2006.

[17] R. W. Duffin, "Cipriano de Rore, Giovanni Battista Benedetti, and the Just Tuning Conundrum," *Journal of the Alamire Foundation*, vol. 9, no. 2, pp. 239–266, 2017.

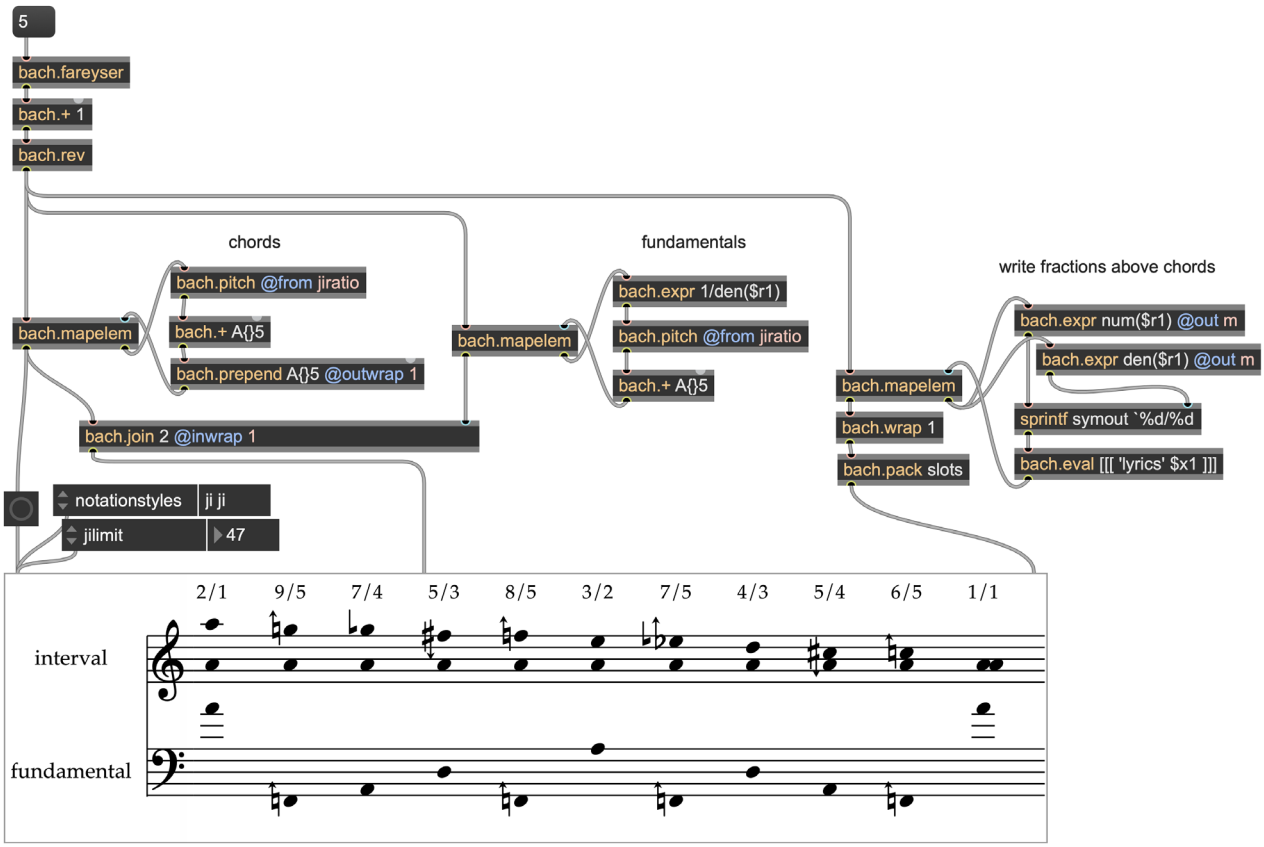


Figure 8. A descending Farey sequence of fractions built over `A{}5`, with the fundamentals of their corresponding harmonic series highlighted in the lower staff.

SPAR-RT: EVOLVING REPRESENTATIONS OF MUSICAL TIMBRE, PITCHES, AND TUNING IN REAL-TIME USING SYMMETRIC PROJECTION ATTRACTOR RECONSTRUCTION

Wentao Hao, Merlin Chao, Mateusz Soliński, Elaine Chew
Department of Engineering, Faculty of Natural, Mathematical & Engineering Sciences
School of Biomedical Engineering & Imaging Sciences, Faculty of Life Sciences & Medicine
King's College London, London, UK
{wentao.hao,merlin.chao,mateusz.solinski,elaine.chew}@kcl.ac.uk

ABSTRACT

The Symmetric Projection Attractor Reconstruction (SPAR) method was proposed to visualise and analyse nonlinear characteristics of physiological signals. It has recently been applied to music signals to visualise instrument timbre. Here we evaluate its use for characterising pitch sets, timbre, and tuning in music signals, and create a real-time visualisation tool based on SPAR to analyse live music. We show how SPAR integrates phase space reconstruction with the Discrete Fourier Transform (DFT) to generate attractors whose shapes reflect the signal's dynamic structures. Using synthetic sounds, the attractors are shown to quantify the interference of different harmonics, music intervals, chords or tuning systems. The aliasing effect is found to significantly affect the shape of the attractor, and perceptual properties like dissonance and commensurability are linked to the attractor's entropy. Using sampled audio in the NSynth dataset, the attractors discriminate between timbres through geometric density patterns over a wide range of pitches. Cluster analysis achieved perfect classification accuracy between flute and brass instruments. The real-time SPAR visualisation enables interactive analysis and visualisation of pitch, timbral, and tuning characteristics in live music sounds evolving dynamically over time, offering novel ways to represent and process music audio. Applications include music signal analysis and learning, sound and instrument design, and live multimedia performance.

1. INTRODUCTION

Music signals exhibit rich acoustic characteristics such as timbre, pitch, and tuning variations, which are encoded in the harmonic structure, amplitude envelope, and waveform morphology. The analysis of these elements is mainly based on traditional time domain or frequency-based representations such as spectrograms and Mel-frequency cepstral coefficients (MFCCs). However, these methods may fail to capture morphological trajectories or nonlinear phenomena that give rise to perceptual differences across musical signals. To address such limitations, recent studies of nonlinear dynamics have provided promising approaches such as the Variable Markov Oracle(VMO) to analysing complex audio signals beyond traditional methods using low-level acoustic representations [1]. Applications such as pitch perception [2] and affect recognition in speech [3] have explored the use of nonlinear dynamic models. One approach is phase space reconstruction, grounded in Takens' embedding theorem, which provides a geometric framework for reconstructing dynamical states in time series by embedding them into a higher-dimensional phase space. An earlier study [4] compared phase space reconstruction analysis to MFCC features in speech recognition, suggesting its potential in audio processing. However, there is a lack of tools in the literature for systematic analysis of the influence of various signal features on its shape and phase space characteristics.

In this study, we developed an implementation of the Symmetric Projection Attractor Reconstruction (SPAR), called SPAR-RT, to represent a signal's phase space as attractors in two-dimensions. The SPAR method was initially introduced to analyse physiological signals [5], then applied to musical signals by Solinski et al. [6] for the visualisation of instrument timbres. SPAR-RT offers animated visualisations of attractors generated from uploaded music files or real-time audio captured via a microphone. We use this framework to create attractors based on synthetic and real acoustic sounds to investigate how musical features such as timbre, pitch, and tuning shape dynamical and nonlinear characteristics of sound signals.

The remainder of the paper is organised as follows: Section 2 introduces the theoretical background and principles of the SPAR method; Section 3 describes the datasets and experiment design; Section 4 presents the results including both quantitative measurements and visualisations of the SPAR attractors; Section 5 discusses the findings in relation to the theoretical framework; Section 6 concludes the paper and outlines the future work.

2. METHODS

This section details the SPAR methodology, covering the mathematical framework, quantification measures, and se-

Copyright: ©2025 Wentao Hao, Merlin Chao, Mateusz Soliński, Elaine Chew et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

lection of parameter used for generating attractors, as well as the real-time implementation (SPAR-RT) developed for interactive analysis.

2.1 Symmetric Projection Attractor Reconstruction

We adopt the SPAR method as a new representation of audio signals. The introduction and mathematical interpretation of the SPAR method are described in [7]. The SPAR method embeds time series into N-dimensional phase space delay coordinates with a specified time delay $\tau = T/N$ (T is the average period of the raw signal) and projects the results onto a 2D representation of an attractor. The attractor's trajectories capture the local waveform morphology and variability between the signal's cycles.

Given an audio signal $x(t)$ with average period T, the (N, k) attractor is defined as the k-th projection of an N-dimensional phase space representation of the signal $x(t)$. The delay vector is defined as $x_{N,j}(t) = x(t - j\tau), j = 0, ..., N-1$. The coordinates of the (N, k) attractor $a(t)$ and $b(t)$ are as follows:

$$a_{N,k}(t) = \mathrm{Re}\left\{\frac{1}{\sqrt{N}}\sum_{j=0}^{N-1} e^{-i\frac{2\pi jk}{N}} x_{N,j}(t)\right\}, \quad (1)$$

$$b_{N,k}(t) = \mathrm{Im}\left\{\frac{1}{\sqrt{N}}\sum_{j=0}^{N-1} e^{-i\frac{2\pi jk}{N}} x_{N,j}(t)\right\}. \quad (2)$$

2.2 Analysis and Parameter Selection

Next, we describe the quantitative measures used to analyse the generated attractors and detail the approach for selecting proper parameters N and k of the SPAR method.

2.2.1 Quantification and Analysis of Attractors

The attractors can be quantified and analysed by four measures based on polar coordinates:

- **r density:** the density distribution in function of the distance (radius) from the attractor's centre;
- θ **density:** the density distribution with angle $(0 - 2\pi)$;
- **outline:** the maximum r of an attractor for a given θ angle; and,
- **joint entropy:** the joint entropy distribution over the (r, θ) space.

2.2.2 Selection of Parameters

The coordinates of attractors $(a_{N,k}(t), b_{N,k}(t))$ are the real and imaginary parts of the k-th coefficients of the Discrete Fourier Transform for the vector $x_{N,j}(t)$. This fundamental linkage to spectral representations of signals guides the selection of the attractor's embedding dimension N and projection index k.

The SPAR method defines $\lfloor \frac{N-1}{2} \rfloor$ distinct projections for a given N based on the properties of the circulant matrix. Consequently, the selection of projection index is $k \in \{1, \ldots, \lfloor \frac{N-1}{2} \rfloor\}$.

The average period T determines the frequency resolution of the SPAR method as $1/T$. The frequency of th k-th projection is consequently given by $f_k = k/T$. For extracted N points, the effective sampling rate is $f_{eff} = N/T$, so the Nyquist frequency is:

$$f_{Nyquist} = \frac{N}{2T}. \quad (3)$$

According to the Nyquist–Shannon sampling theorem, any signal component with a frequency higher than $f_{Nyquist}$ cannot be independently resolved by the DFT system of N points, which results in aliasing. Therefore, the selection of N is crucial to a proper analysis using SPAR as it relates to the maximum number of harmonics that can be represented without aliasing. However, increasing N indefinitely does not always yield better results. It may introduce unnecessary computational effort and complexity in the analysis.

2.3 Real-time Implementation of SPAR for Music

We present a dedicated implementation of the SPAR method referred to as SPAR-RT as shown in Figure 1.

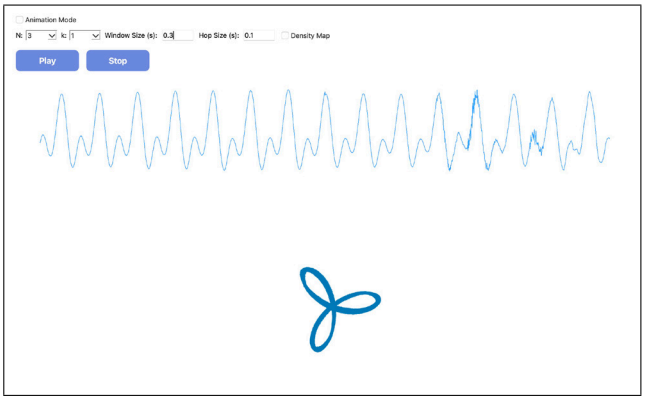


Figure 1: The user interface of the SPAR-RT tool

Built using Python, SPAR-RT is designed to enable the interactive analysis and intuitive observation with two modes:

- **Real-time:** The tool captures the surrounding sound and generates corresponding real-time attractors.
- **Animation generation:** This mode allows users to upload audio files and export corresponding attractor animation videos synchronised with the audio.

Both operational modes support the adjustment of key SPAR parameters (N, k) and frame settings (window and hop size) to optimise the visualisation for specific analysis goals. The user may also choose to view the visualisations on a white or dark background.

Example videos of visualisations demonstrating the dynamic visualisation capabilities of SPAR-RT are provided in the Supplementary Material.

3. DATA

This section describes the synthetic and real audio datasets used to investigate how musical characteristics like timbre, pitches and tuning are represented by the SPAR method.

3.1 Synthetic Sinusoid Harmonic Waves

We generated synthetic sinusoid harmonic waves by combining sinusoidal waveforms having different frequencies, where each frequency was an integer multiple of the fundamental frequency. The mathematical formulation of the synthetic sinusoid harmonic wave is as follows:

$$x(t) = \sum_{m=1}^{M} A_m \cdot \sin(2\pi m f_0 t + \phi_m), \qquad (4)$$

where f_0 is the fundamental frequency, and A_m and ϕ_m are the amplitude and phase of the m-th harmonic, respectively.

To explore how different harmonic components impact the shape of attractors, we used two settings of composite sinusoidal signals, containing three and six harmonic components, respectively. Different embedding dimensions ($N = 5, 10$) were selected for attractor generation. The fundamental frequency was set to $f_0 = 440$ Hz and the average period T was $1/f_0$. Signals were generated at the sampling rate (fs) of 44000 Hz, ensuring that $T * fs/N$ was an integer. This prevented misalignment between the sampling rate and the SPAR delay ($\tau = T/N$) caused by rounding errors.

3.2 Intervals

Using SuperCollider, we synthesise all dyads in the chromatic scale as a pair of pure sine waves. For each dyad, we generate its attractor with parameters $N = 4$ and $k = 1$ to avoid aliasing. We then compute the joint entropy and plot it alongside a dissonance curve modelled on pure sine wave interactions, as proposed in [8].

3.3 Tuning and Chords

To evaluate how different tuning systems are represented by SPAR, we synthesised both C major and C minor chords using pure sine waves. For each chord, one version was tuned according to the just intonation tuning system, in which pitch relationships follow simple integer ratios (e.g., $5:4$ for a major third, $6:5$ for a minor third), and another in equal temperament system, where frequency ratios are defined by dividing the octave into twelve equal parts on a logarithmic scale. Table 1 provides an example of the frequency ratios in each tuning system for a major triad and a minor triad, and Table 2 gives the actual frequencies for a C major triad and a C minor triad.

Chord	Just Int. Ratios	Equal Temp. Ratios
major triad	$\left[1,\ \frac{5}{4},\ \frac{3}{2}\right]$	$\left[1,\ 2^{4/12},\ 2^{7/12}\right]$
minor triad	$\left[1,\ \frac{6}{5},\ \frac{3}{2}\right]$	$\left[1,\ 2^{3/12},\ 2^{7/12}\right]$

Table 1: Frequency ratios of a major and a minor triad in Just Intonation and Equal Temperament tuning systems

Pitches	Just Int. Freq. (Hz)	Eq. Temp. Freq. (Hz)
C, E, G	[261.7, 327.0, 392.4]	[261.7, 329.6, 392.0]
C, E♭, G	[261.7, 314.0, 392.4]	[261.7, 311.1, 392.0]

Table 2: Rounded frequencies of generated C major and C minor chords in Just Intonation and Equal Temperament tuning systems (base note: C = 261.7 Hz)

3.4 Real Acoustic Instrument Audio

We applied the SPAR method to real acoustic signals to show differences in generated attractors regarding the timbre of the instruments. We used 22 samples of flute sounds and a brass instrument (44 samples in total) from the NSynth database by Google Inc [9]. Each audio sample consists of a single note with a fixed pitch, ranging from G4 to E6 across samples, and a duration of 4 seconds.

We applied the SPAR method to these samples using two embedding dimensions, $N = 5$ and $N = 10$. The generated attractors were compared and analysed in terms of both pitch and instrument timbre. To quantify attractors, we extracted three measure profiles (r density, θ density, and outline) and concatenated them into a single profile vector as the numerical representation for each attractor.

Finally, we clustered the vectors using hierarchical clustering, with the results demonstrated in a dendrogram.

4. RESULTS

This section presents the results for both synthetic and real acoustic audio signals, demonstrating how the SPAR method effectively characterises and distinguishes musical features such as harmonics, dissonance, tuning, pitch and timbre.

4.1 Synthetic Sinusoid Harmonic Waves

Figure 2 shows the attractors generated from a sinusoidal harmonic signal consisting of three components based on different dimensions N and projections k. We observed:

When $N = 5$, $k \in \{1, 2\}$, $f_{Nyquist} = 1100$ Hz, and

- $k = 1$ (440 Hz), the attractor appears as a perfect circle.
- $k = 2$ (880 Hz), the attractor is a star-shaped structure with a five-fold rotational symmetry due to aliasing from the harmonic with 1320 Hz.

When $N = 10$, $k \in \{1, 2, 3\}$, $f_{Nyquist} = 2200$ Hz,

- All three attractors show perfect circle shapes, which indicates that three sinusoidal components have been fully resolved by the SPAR with a high dimension $N = 10$.

Figure 3 illustrates the attractors for the sinusoidal harmonic signals consisting of six components based on different dimensions N and projections k.

When $N = 5$, $k \in \{1, 2\}$, $f_{Nyquist} = 1100$ Hz, and

- $k = 1$ (440 Hz), the attractor exhibits a unique petal-like structure with five-fold rotational symmetry. Five

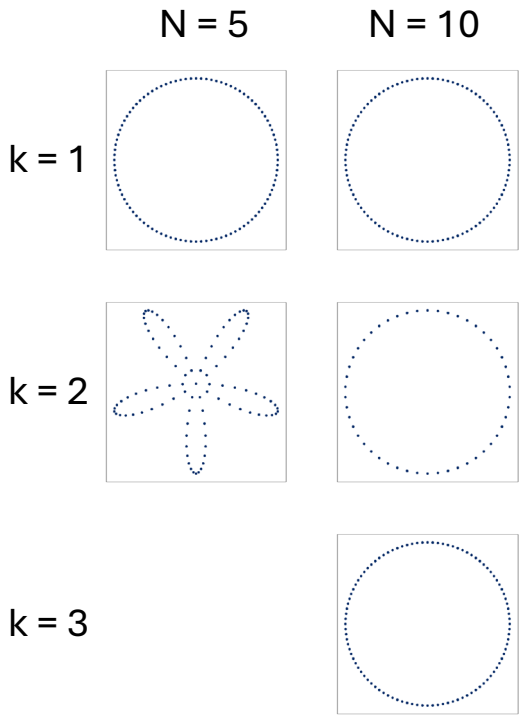


Figure 2: The shapes of attractors for audio signals consisting of three sinusoid functions with $N = 5, 10$ and $k = 1, 2, 3$. The non-circular attractor $(5, 2)$ is a manifestation of aliasing from higher harmonics.

distinct petals are symmetrically distributed around the centre point.

- $k = 2$ (880 Hz), the attractor presents a star-like structure with five-fold rotational symmetry.

When $N = 10$, $k \in \{1, 2, 3, 4\}$, $f_{Nyquist} = 2200$ Hz, and

- $k = 1, 2, 3$ (440 Hz, 880 Hz, 1320 Hz), the attractors show the perfect circles, indicating sinusoidal components are fully represented.
- $k = 4$ (1760 Hz), the attractor presents a star-like shape with five-fold rotational symmetry.

The results suggest that for the m-th sinusoidal component with a frequency $f_m > f_{Nyquist}$ we observe aliasing at $f_{alias} = |f_m - f_{eff}| = |f_m - 2 * f_{Nyquist}|$. The corresponding projection *k* of the attractor for resolving f_{alias} exhibits rotational symmetry.

4.2 Intervals

The shape of the attractor varies noticeably across dyads in the chromatic scale. We observe that its structure is primarily determined by the frequency ratio between the two tones, influenced by two key factors: beating and commensurability (the existence of a shared multiple among frequency periods). Both factors are known to impact auditory perception [10], and their effects may emerge at different time scales, either within a single analysis frame or across multiple frames. To illustrate effects that unfold

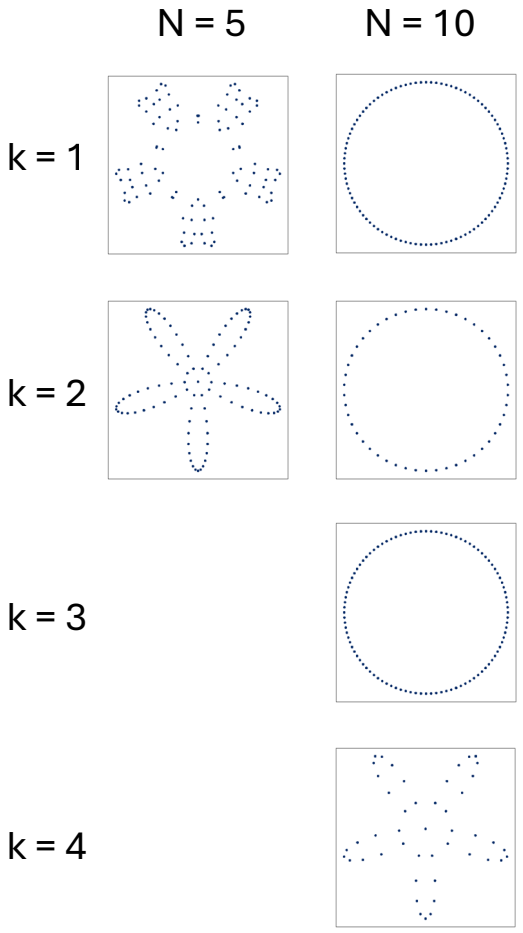
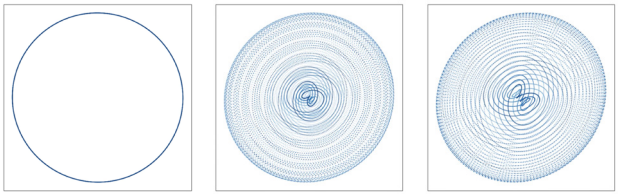


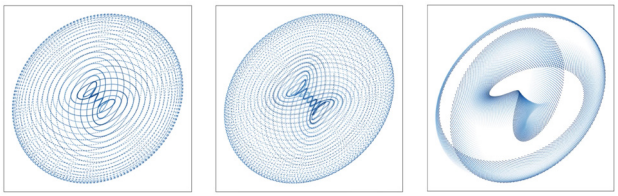
Figure 3: The shapes of attractors for audio signals consisting of six sinusoid functions with $N = 5, 10$ and $k = 1, 2, 3, 4$. The non-circular attractors $(5, 1)$, $(5, 2)$ and $(10, 4)$ are manifestations of aliasing from higher harmonics.

over time, a video showing the attractors for all dyads is provided in the Supplementary Material (Supplementary Animation 1). As expected, the attractor derived from the unison and octave forms a perfect circle, reflecting the fact that the sound frequencies were either identical or related by a factor of two, respectively. Notably, the intervals of the fourth and the fifth (see Figures 4f and 4h) produce attractors with shapes that are similar one to another but distinct from the shapes of other intervals.

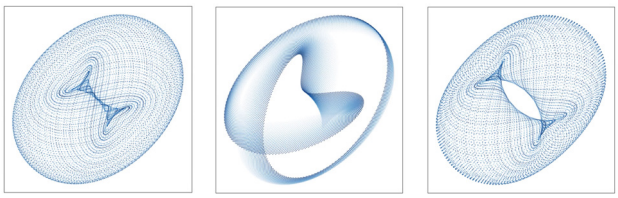
We compared the dissonance curve for two pure sine waves [8] with the joint entropy of the attractor; both measures show similar patterns as shown in Figure 5. The lowest values for both measures appeared at the unison and the octave (12 semitones) intervals, considered the most consonant. From the unison to the minor second, we observe a steep rise toward a maximum level for both entropy and dissonance, followed by a gradual, monotonic decrease toward the octave. This comparison shows an existing relationship between the SPAR representation and aspects of sound perception.



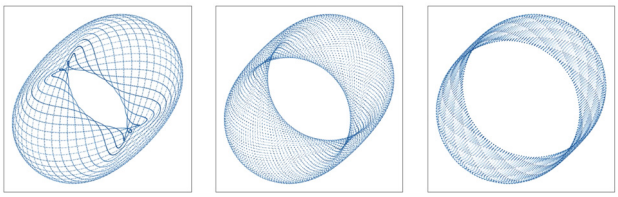
(a) Unison / Octave (b) Minor Second (c) Major Second



(d) Minor Third (e) Major Third (f) Fourth



(g) Tritone (h) Fifth (i) Minor Sixth



(j) Major Sixth (k) Minor Seventh (l) Major Seventh

Figure 4: Attractors for all dyads in the chromatic scale for N=4 and k=1

4.3 Tuning and Chords

The attractors exhibit significant differences between 12-tone equal temperament and just intonation tuning systems. This shows that the attractor's shape is highly sensitive to the commensurability of the signal's frequency components. Commensurability refers to how closely the individual signals' periods share a common multiple. To determine this common multiple, calculate the least common multiple (LCM) of the denominators of the frequency ratios [11]. For example, the major triad in just intonation, with frequency ratios $1 : \frac{5}{4} : \frac{3}{2}$, realigns every $\mathrm{LCM}(4, 2) = 4$ cycles of the fundamental frequency. The minor triad, with ratios $1 : \frac{6}{5} : \frac{3}{2}$, realigns every $\mathrm{LCM}(5, 2) = 10$ cycles, indicating a longer but still periodic alignment. These simple integer ratios produce bounded, low-entropy attractors, with the minor chord's attractor showing a slightly more intricate pattern (see Figures 6a and 6b). In contrast, equal temperament uses irrational frequency ratios that approximate just intervals logarithmically. These incommensurable components contribute to more complex attractor shapes with chaotic behaviour (without strict periodic loops; see Figures 6c and 6d).

Although the relationship between tuning systems and perceptual roughness remains debated, previous studies have linked commensurability to spectral interference and roughness perception [12], suggesting another connection between the attractor's shape and another aspect of sound perception.

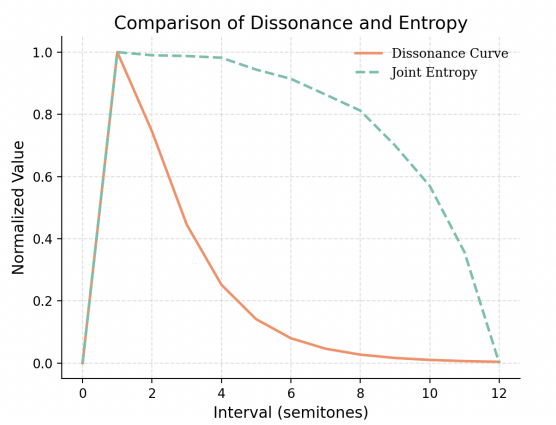
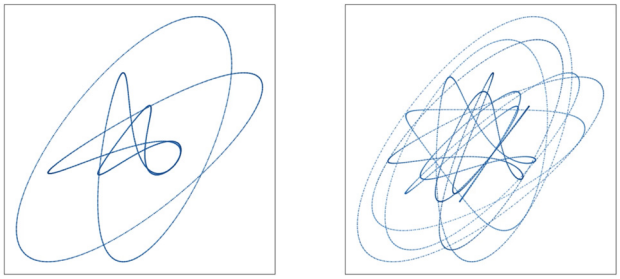


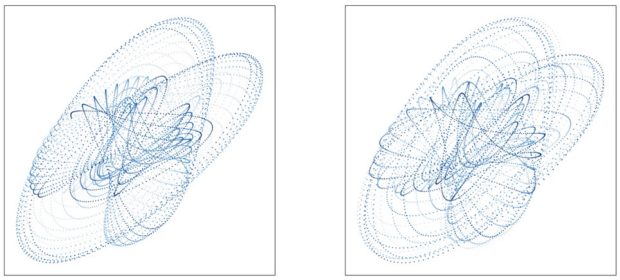
Figure 5: Normalised dissonance and joint entropy for pure sine dyads of the chromatic scale.

4.4 Real Acoustic Instrument Audio

Figure 7 shows the 5-dimensional attractors ($N = 5$) extracted from the flute sound with pitches A4 (440.0 Hz) and C#6 (1108.73 Hz) and their corresponding power spectra. For A4, the power spectrum contains multiple harmonic frequency components, and the corresponding attractor exhibits a non-circular five-fold rotational symme-



(a) C Major triad in just intonation tuning system (b) C Minor triad in just intonation tuning system



(c) C Major triad in equal temperament tuning system (d) C Minor triad in equal temperament tuning system

Figure 6: Attractors of C Major and Minor triads with $N = 3$ and $k = 1$ in just intonation and equal temperament tuning systems.

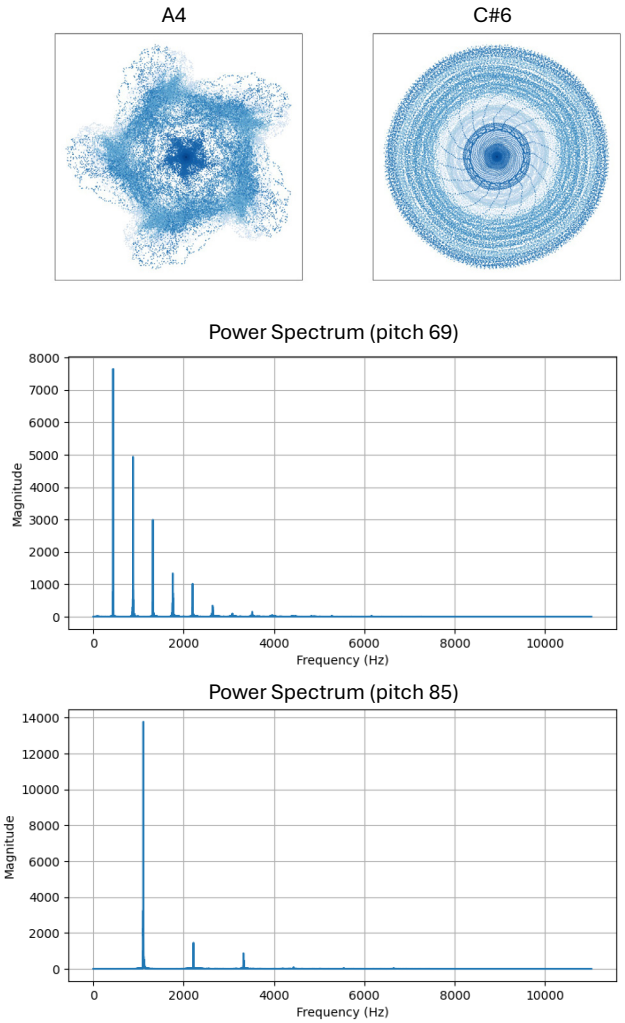


Figure 7: The attractors from the flute sounds on different pitches (A4 and C#6) and corresponding power spectra.

try. For pitch 85, the attractor has a circular outline, and the spectrum shows fewer harmonic components with a significantly dominant main fundamental frequency. A dynamic visualisation illustrating how the shape of the 5-dimensional attractor evolves as the pitch increases is provided in the Supplementary Material (Supplementary Animation 2)

Figure 8 shows the comparison between the 10-dimensional attractors extracted from a flute and a brass instrument at the same pitch of F#5 and their corresponding attractor measure profiles. Both attractors are circle-shaped, indicating that the selection of the dimension $N = 10$ resolved the audio signal well without significant aliasing. However, they are distinct in density distribution. The attractor representing a brass instrument reveals a higher density concentration in the centre and in the ring with a radius of approximately 0.15, compared to the attractor extracted from the sound of a flute. For the θ density profile, we observe an oscillatory-like pattern for the attractor linked with a brass instrument, while the profile for the flute is distinctly flatter. The outline profiles for flute and brass exhibit distinct geometric patterns across the angular domain. The flute profile shows higher and more fluctuating maxi-

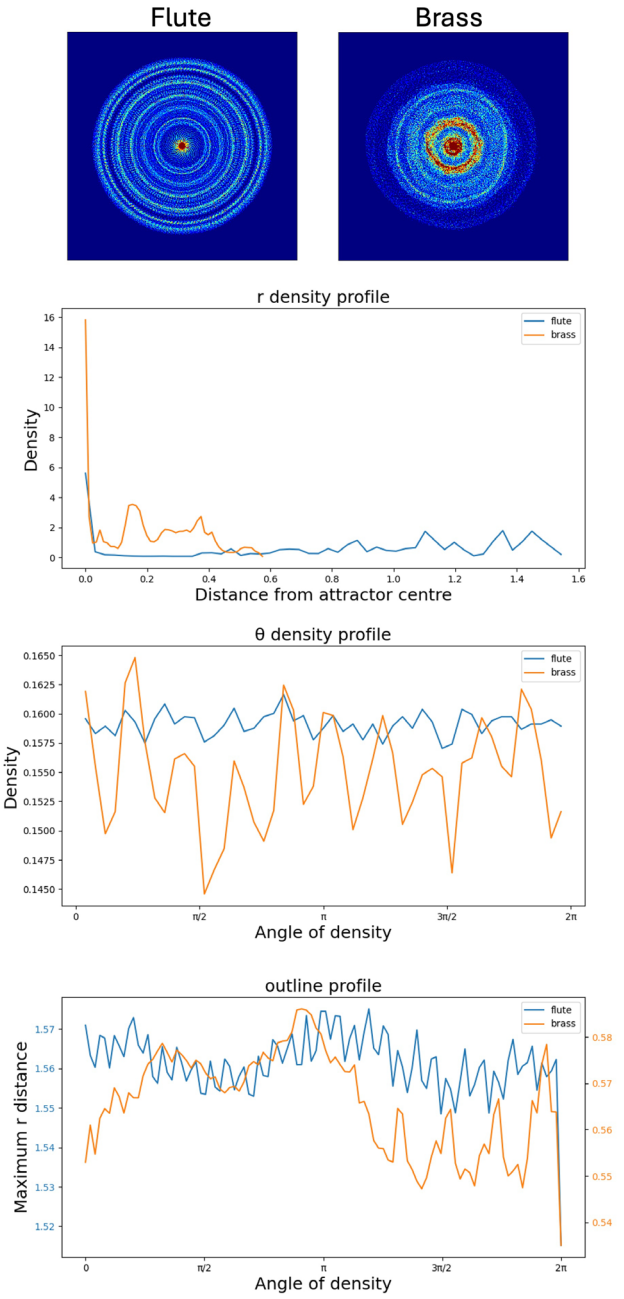


Figure 8: The attractors of flute and brass with $N = 10$, $k = 1$, pitch $= 78$, along with the corresponding profiles of the three attractor measures. Here, the attractors are shown as heat maps depicting the density of points in phase space used to calculate the density profiles.

mum r distances. In contrast, the brass profile is generally lower and more variable, suggesting a less symmetrical or more irregular attractor geometry.

Figure 9 illustrates the dendrogram generated from the hierarchical clustering of concatenated profile vectors. Using a distance threshold of 50, two distinct clusters emerge: one composed entirely of brass instruments and the other exclusively of flutes.

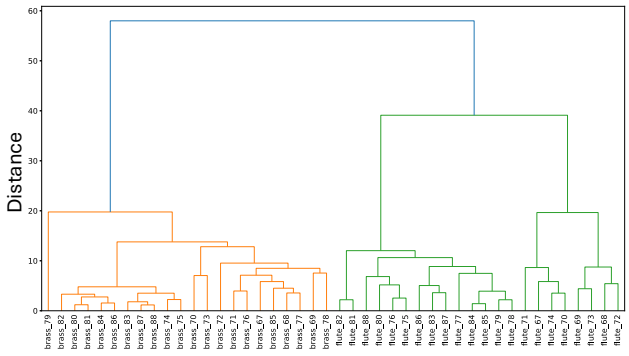


Figure 9: The dendrogram created from the distances between different 10-dimensional attractors.

5. DISCUSSION

In this work, we have applied the SPAR method to characterise dynamic changes in the synthetic and acoustic audio signals and presented the differences between attractors regarding timbre, pitch (including pitch sets) and tuning.

The analysis using harmonic sinusoid signals revealed the occurrence of aliasing, which deforms the complete reconstruction of the phase space. We showed a close relation between the embedding dimension N, projection index k, and the signals' frequency harmonics. These findings deepen our understanding of SPAR visualisation capabilities and enhance the interpretability of the geometric patterns, offering guidelines for parameter selection in future applications.

The differences between attractors for distinct dyads demonstrated the phenomenon of frequency intervals resulting in differential projections of phase space. Specifically, we observed that more dissonant intervals, typically associated with prominent beatings, correspond to attractors with higher joint entropy. This suggests a close relationship between the shape of the attractor and perceptual phenomena such as dissonance and beatings. The comparison between just intonation and equal temperament demonstrated that the attractor is sensitive to the commensurability of frequency components, which is also a measure of dissonance and roughness. These results highlight the potential of the SPAR method as a tool for linking signal structure to auditory perception.

The analysis of real acoustic signals confirmed the presence of a direct link between signal harmonic contents and attractor morphology, which is consistent with the observations for the synthetic harmonic sinusoids. The SPAR method was also able to distinguish between instrument families using geometric density profiles, even when the overall attractor shapes appear visually similar and circular. These examples demonstrated that SPAR can discriminate betwee subtle dynamics and inharmonicity within real acoustic audio signals, providing a powerful means to differentiate between sonic timbres. The classification using hierarchical clustering of the concatenated SPAR features suggests that the SPAR method may extract instrument-specific characteristics that complement traditional spectral features, thereby enhancing automatic instrument recognition. Future research involving supervised machine learning techniques could explore the effectiveness of this approach across a broader range of instruments and pitches.

This study offers a foundational analysis of the SPAR method as a music signal representation. There are some limitations to the approach. The analysis on synthetic data was conducted under controlled conditions, specifically by ensuring the sampling frequency was perfectly aligned with the SPAR time delay and dimension N. While the validation on a real acoustic dataset matched the findings on synthetic data, this perfect alignment may not occur in a real-world scenario, which may introduce distortions in the trajectories of attractors. Therefore, future work is needed to more thoroughly analyse how to quantify and interpret this real-world shift, which would enhance the SPAR method's robustness for practical applications.

The development of the real-time visualisation and animation generation tool provides a promising option for such future exploration, enabling interactive analysis and intuitive observation of evolving acoustic features over time, for instance in performance contexts where visualisation of attractors provides expressive feedback for sound and instrument designers, composers, performers, and the audience.

6. CONCLUSIONS

This study presents an analysis of the Symmetric Projection Attractor Reconstruction (SPAR) method as a novel representation for music signal analysis, focusing on its ability to encode musical characteristics such as timbre, pitch, and tuning. Through a series of simulated and real-world experiments, we demonstrate that SPAR attractors offer interpretable geometric representations that encode meaningful signal dynamics.

Our harmonic simulations reveal how attractor morphology reflects frequency content and aliasing effects, while analyses of musical intervals and tuning systems highlight SPAR's sensitivity to perceptual properties like dissonance and commensurability. The real acoustic music audio study further validates the method's ability to distinguish between instrument timbres through subtle geometric density patterns, achieving perfect classification in our cluster analysis. These results underscore SPAR's potential as a powerful and perceptually relevant tool for music signal analysis.

The real-time SPAR visualisation and animation-maker tools support not only analyses and learning but also creative applications in sound and instrument design, and live and multimedia performance. It will also support future explorations, enabling interactive analysis and development of new dynamic acoustic features. Future work will focus on further analyses of the discrepancies between controlled simulations and real-world scenarios, and on improving the robustness and interpretability of SPAR.

Acknowledgments

This result is part of the COSMOS project (`https://cosmos.isd.kcl.ac.uk`), which has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation

programme (Grant agreement No. 788960, https://cordis.europa.eu/project/id/788960). WH is supported by a King's College London China Scholarship Council (K-CSC) Scholarship.

7. REFERENCES

[1] P. Mouawad and S. Dubnov, "On modeling affect in audio with non-linear symbolic dynamics," *Advances in Science, Technology and Engineering Systems Journal*, vol. 2, no. 3, pp. 1727–1740, 2017.

[2] J. H. E. Cartwright, D. L. González, and O. Piro, "Pitch perception: A dynamical-systems perspective," *Proceedings of the National Academy of Sciences*, vol. 98, no. 9, pp. 4855–4859, 2001.

[3] A. Harimi, A. AhmadyFard, A. Shahzadi, and K. Yaghmaie, "Anger or joy? emotion recognition using nonlinear dynamics of speech," *Applied Artificial Intelligence*, vol. 29, no. 7, pp. 675–696, 2015.

[4] A. Lindgren, M. Johnson, and R. Povinelli, "Speech recognition using reconstructed phase space features," in *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03).*, vol. 1, 2003, pp. I–60.

[5] M. Nandi and P. J. Aston, "Extracting new information from old waveforms: Symmetric projection attractor reconstruction: Where maths meets medicine," *Experimental Physiology*, vol. 105, no. 9, p. 1444–1451, May 2020. [Online]. Available: http://dx.doi.org/10.1113/EP087873

[6] M. Solinski, W. Hao, and E. Chew, "Spar-timbre: Unlocking non-linear dynamics in music audio with symmetric projection attractor reconstruction," in *17th International Symposium on Computer Music Multidisciplinary Research*, 2025, accepted.

[7] J. V. Lyle and P. J. Aston, "Symmetric projection attractor reconstruction: Embedding in higher dimensions," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 31, p. 113135, 2021.

[8] W. A. Sethares, *Tuning, Timbre, Spectrum, Scale*, 2nd ed. London; New York: Springer Science & Business Media, 2005.

[9] J. Engel, C. Resnick, A. Roberts, S. Dieleman, D. Eck, K. Simonyan, and M. Norouzi, "Neural audio synthesis of musical notes with wavenet autoencoders," 2017.

[10] J. R. Pierce, "Periodicity and pitch perception," *The Journal of the Acoustical Society of America*, vol. 90, no. 4 Pt 1, pp. 1889–1893, Oct. 1991.

[11] F. Stolzenburg, "Harmony perception by periodicity detection," *Journal of Mathematics and Music*, vol. 9, no. 3, p. 215–238, Aug. 2015. [Online]. Available: http://dx.doi.org/10.1080/17459737.2015.1033024

[12] P. N. Vassilakis, "Perceptual and physical properties of amplitude fluctuation and their musical significance," Doctoral Dissertation, University of California, Los Angeles, 2001.

NOTATION FOR ORGAN MUSIC: BALANCING PRESCRIPTIVE AND DESCRIPTIVE MODELS FOR AN INSTRUMENT WITHOUT STANDARDISATION TO UNLOCK COMPOSITIONAL AND PERFORMATIVE POTENTIAL

DDr. Martin Ritter
University of Music
and Performing Arts Vienna
ritter-martin@mdw.ac.at

Dr. Alyssa Aska
University of Music
and Performing Arts Vienna
aska@mdw.ac.at

Mag. Pablo Mariña
University of Music
and Performing Arts Vienna
pablo.abelardo.mm@gmail.com

ABSTRACT

A unique challenge in notating music for the organ lies in the instrument's inherent site-specificity. Pipe organs are not mass-produced or standardized, unlike most musical instruments. This variability presents a core dilemma for composers: how to notate music for an instrument that defies treatment as a fixed or portable entity. Notation must therefore engage not only with abstract musical ideas but also with the physical, mechanical, and spatial realities of each unique instrument.

Composers have responded to the challenges of organ notation through a range of strategies that reflect both notational detail and the function of the score itself. These approaches often fall along a spectrum between prescriptive and descriptive notation.

This paper examines a range of organ notation examples, from the early modern period to recently premiered works, highlighting how composers tailor their methods to the specific demands of each composition. We argue that the most effective notational choices emerge from a sensitivity to the sonic goals, technical requirements, instrumental mechanics,and contextual conditions of each individual piece.

1. INTRODUCTION

The pipe organ's lack of standardization presents a unique challenge for composers, who must find ways to notate for an instrument that varies widely in construction, tuning, playing range, timbral possibilities, action type, and acoustic environment. Each organ is custom-built for a specific space, reflecting not only the architectural and acoustic context, but also often historical, technological, and liturgical considerations. These factors result in substantial diversity among instruments, complicating the development of any universally viable notational system. Composers are then left to choose between a prescriptive approach, which offers precise, instrument-specific instructions, and a descriptive approach, which prioritizes the intended sonic outcome.

This variability leaves composers with a fundamental notational dilemma: how can one communicate musical ideas for an instrument that does not exist in a fixed or standardized form? One way to approach this challenge is to explore their work on a continuum between prescriptive and descriptive notation. At one end of this spectrum, prescriptive notation offers precise, instrument-specific instructions that maximize clarity but limit portability. At the other, descriptive notation prioritizes the intended sonic outcome, while introducing a degree of ambiguity.

A prescriptive score, such as tablature, specifies exact instructions often tied to the physical and mechanical features of a particular instrument. Such notation precisely describes the performer's actions on the instrument's interface but can severely limit portability between instruments. In contrast, descriptive notation seeks to define the intended sonic outcome rather than prescribing exact technical means. In practice, most organ works blend prescriptive and descriptive methods. Composers must navigate this spectrum, balancing the need for clarity with the desire for flexibility and interpretive freedom. This paper explores how composers have addressed these notational challenges from the early modern period to the present day, analyzing specific examples to illustrate the diverse strategies employed. [1]

Notational systems therefore reveal which musical parameters a composer considers essential. The Western staff notation system originating with Guido of Arezzo in the eleventh century designated pitch as primary musical substance, reducing other sound qualities such as timbre, articulation, or dynamics to secondary attributes. [2] In a 2016 lecture at the Darmstadt Ferienkurse, Klaus Lang described the impact that Guido's system would have on music:

> Guido of Arezzo's system of notation transformed a process into an object and took one aspect of this process, namely pitch, and defined it as substance. All other aspects of sounds, like duration, dynamics, sound qualities, became attributes attached to this substance. [3]

This system evolved into the notation still used in most Western classical music today. Pitch is widely treated as

Copyright: ©2025 DDr. Martin Ritter et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

the central unit of representation. Notation constructs what aspects of a work are considered most important. We examine how composers have approached this challenge by adopting notational strategies that balance compositional intention and the physical and acoustic realities of the organ.

Our goal is not to propose a universal or codified notational system for the organ. Instead, we focus on how composers from the early modern period to today make notational choices in response to the specific sonic, spatial, and technical demands of each work. Through this lens, notation emerges not just as a tool for transmitting music, but as a form of documentation that reflects the aesthetic values and priorities of a composer, a tradition, or a period. We analyze how composers have confronted these organ-specific notational choices. This analysis goes beyond a simple survey of options to provide a critical guide for composers, clarifying the strengths and weaknesses of various approaches and highlighting how a composer's notational decisions directly impact a work's realization. Ultimately, we argue that by understanding these established precedents, composers can make more informed and intentional choices, ensuring their musical ideas are effectively communicated and realized regardless of the attributes of a specific organ.
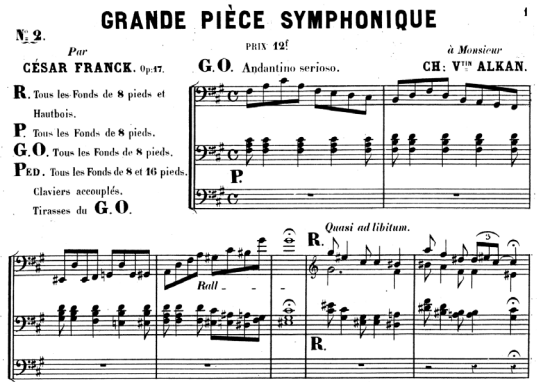


Figure 1. César Franck's organ music exemplifies the complex registration practices of the late Romantic period using symbols and text indications in the score.
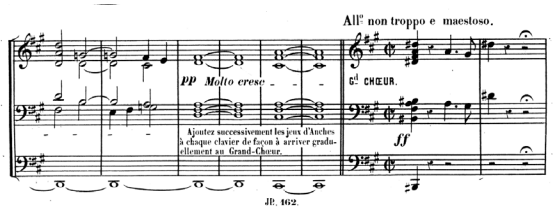


Figure 2. César Franck: text indications in the score.

2. ORGAN NOTATION OVERVIEW AND PRECEDENTS OF MODERN ORGAN NOTATION

Organ music lacks a fixed notational standard, as a result of the instrument's inherent variability. Aspects such as layout, registration, and dynamics can vary widely from piece to piece. The most common historical layout features two staves for the manuals (hands) and a third for the pedals (feet), though this configuration is not universal. Organs often include multiple stops that affect both timbre and octave displacement (e.g. $16'$, $8'$, $4'$ stops). However, organ notation always reflects the played pitch, not the sounding pitch. Additional instructions, such as which manuals to use or which stops to engage, are often conveyed through textual annotations in the score.

A brief overview of how the organ produces sound is necessary in order to contextualize the notational precedents that follow. A mechanical pipe organ typically includes one or more keyboards (manuals), a pedalboard, ranks of pipes constructed from wood or metal, and a system (such as bellows or an electric blower) for directing air into the pipes.

Stops, which are controlled by drawknobs, buttons, or levers, control the flow of air to specific ranks of pipes. Each rank produces a distinct timbre, often modeled after specific instruments; engaging or disengaging stops alters the organ's overall timbre. Sound can be produced only when a stop is activated and a corresponding key is pressed. Since there is no default sound or registration, performers must either follow composer-specified timbral directions or make their own registration choices.

2.1 Organ notation in the late romantic and early modern period

One of the earliest challenges in organ notation is how composers communicate registration, or which types of pipe or pipes should sound when a key is pressed. Composers began to expand the role of the organ beyond the church in the 19th century, recognizing its potential to fill large concert halls and contribute to the breadth and timbral richness of Romantic orchestral traditions. Registration of organ music started to be depicted in scores to reflect this. In the example above (Figure 1), from César Franck's Grande Piéce Symphonique (1860–62) general registration information is given in the score on the first page, with general indications next to each manual and pedal (R = récit, P = positif, G.O. = Grand Orgue, Ped. = Pedal). Length of pipes (and thus sounding octave) is specified. Further information in the score is provided using dynamic levels and text instructions (see Figure **??**). While the composer's intention regarding volume and overall sound quality and transposition is made clear, the specific timbre remains the responsibility of the performer.

Olivier Messiaen is widely recognized as one of the first composers to approach the organ as an experimental instrument, pushing the boundaries of both its sound and its notation. He was also among the first to produce organ works with highly specific registrations, written with a particular instrument in mind. Messiaen defines the exact registration for each manual at the beginning of the score with the abbreviations for the manuals that Franck also used. Messiaen's descriptions are very precise. For example, in the récit (R): „cymbal 3 rangs" indicates that the cymbal 3 rangs (a mixture with 3 ranks) stop should be pulled
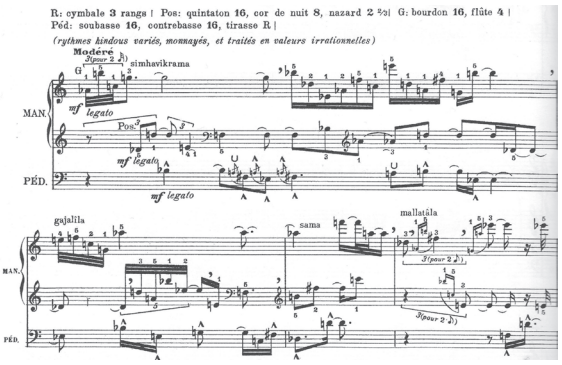


Figure 3. Olivier Messiaen: precise registration indications in the score as used for the organ at Sainte-Trinité in Paris.

rather than using a more general description like mixture 3. Messiaen is also very specific in notating the fingerings for hands and uses symbols in the pedal notation to depict which notes should be played with the heel, which with the tip, and which foot should be used, as seen in Figure 3.

Although many organs share similar stop types, instruments are not designed with identical collections in mind, and stop names often vary significantly. This means that Messiaen's works cannot always be realized exactly as instructed; performers must interpret his instructions and adapt them based on the stops available on the instrument at hand. In this way, the performer engages in an active interpretive process, translating the composer's intentions through the lens of a specific instrument. As an example of interpreting Messiaen's organ music on different organs, Adam Ravian notes: "there is a similar challenge that will be interpreted differently on instruments that are French by design or built according to other national styles. On the French organ, the manuals from lower to upper are Grand Orgue, Positif, and Récit, whereas on instruments that are outside of the French organ building tradition, the order of the two lower manuals is reversed." [4]

Organ notation not only guides interpretation but also shapes the expectations placed on the performer. It assumes a familiarity with organ-specific idioms, registration practices, and instrument mechanics. Conversely, composers must understand how organists interpret scores, so that their intentions can be realized across a range of instruments. The practice of organ notation is important to the composers, but it is also important for performers to be familiar with and understand. The composer must have some understanding of organ interpretation if they wish to surmise how an organist might register the piece with the available information in the score. Messiaen was trained in the tradition of Marcel Dupré that prioritized strict adherence to the written sign, however the interpretations of his own works displayed considerable flexibility. This divergence leads Ravian to conclude that "Messiaen had ideas that could not be adequately conveyed through notation." [4] This highlights the inherent limitations of notation in capturing the full range of interpretive possibilities available to organists, and the need for composers to consider how their notational choices will be realized in practice.

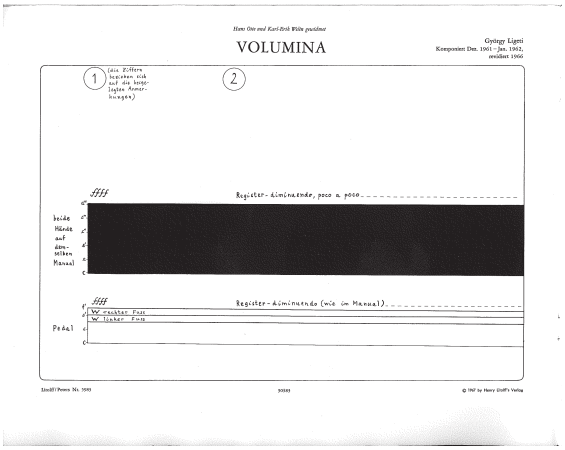


Figure 4. György Ligeti: graphic notation in the score of Volumina (1962).

2.2 Second Half of the 20th Century

Messiaen did not employ extended techniques in the strict sense, but his attention to registration and timbre opened the door to later composers' explorations of the organ's full sonic potential. In the latter half of the 20th century, composers such as György Ligeti pushed the limits of organ composition by incorporating extended techniques, unconventional textures, and experimental (graphical) notation. Ligeti's *Volumina* (1962) remains one of the most well-known experimental organ works of the mid-twentieth century, featuring graphic notation. As seen in Figure 4, Ligeti specifies actions by hand and foot and while the information is primarily conveyed through a combination of graphic notation and text, the score is appended with four pages of performance notes.

3. CONTEMPORARY NOTATION FOR ORGAN

While the pipe organ never occupied a central position in the postwar avant-garde, composers of the late 20th century continued to expand its technical and expressive possibilities, including innovations in notation. The organ experienced a revival in the early 21st century, with more composers embracing its sonic potential. New performance and research venues, such as Orgelpark, provide platforms for experimentation and documentation of this renewed interest in organ exploration. Contemporary approaches often build on 20th-century precedents, but with a shift in notational priorities. Particularly the use of stops as compositional material is an aspect that is more prominent in contemporary organ works. Whereas mid-20th-century scores often emphasized visual complexity and aesthetic appearance, including Ligeti's *Volumina*, many 21st-century composers now prioritize practicality and clarity in their notation. Below we discuss several contemporary techniques and considerations and some of the solutions that composers have developed to approach them.

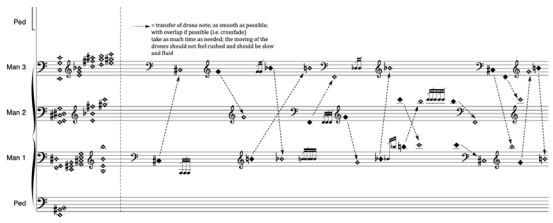


Figure 5. Martin Ritter: *Saturate*.

3.1 Staves and Layout

The traditional layout for organ notation consists of three staves, as described previously: two for the manuals and one for the pedalboard. However, this configuration is frequently adapted to fit the specific needs of each piece. A common development in contemporary organ notational practice is the use of separate staves for each manual, allowing for greater visual clarity, especially when different manuals are used simultaneously or manual-specific gestures are sustained. When manual changes are frequent or sustained pitches like drones appear across systems, separate staves prevent visual clutter and help the performer track held notes and manual assignments clearly.

The choice between manual-based and hand-based notation depends on the demands of the piece and how much manual-specific information plays a critical role. Using notation with two manuals highlights the importance on specific sounding results and is therefore more descriptive, whereas the standard keyboard layout is more prescriptive, placing emphasis on performative aspects (which hands to use). Some scores also use additional staves to notate elements such as stop changes. This is particularly useful when registration changes occur frequently or evolve over time as part of a musical gesture. The examples discussed below will use various forms of the layout, and demonstrate that different pieces have different layout needs.

3.2 Keyboard Action and Drones

One of the most widely used extended techniques in contemporary organ writing is the sustained drone. On the organ, keys can be held down using external objects, allowing the performer to play other material simultaneously.

In Martin Ritter's *Saturate* (2017), Figure 5, filled diamond note heads indicate pitches to be sustained as drones, while empty diamond noteheads signal their release. Dotted lines connect these gestures, indicating their temporal trajectory.

Another feature of this type of drone notation is the *grace measure* at the beginning of each system that reflects all keys currently being held down as drones. This is an example of a composer combining both prescriptive and descriptive notation in a meaningful way: the actions of the performer placing and removing drones are depicted, as are the existing sounding pitches at the beginning of each system.

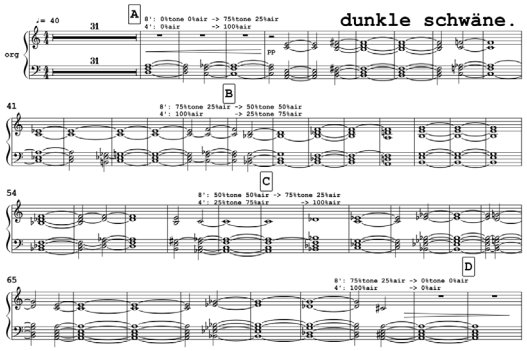


Figure 6. Klaus Lang: *Sadunkle schwäne*.

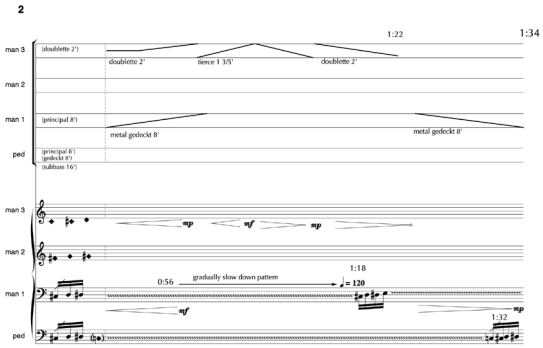


Figure 7. Alyssa Aska: *looped.(dynamic).*.

3.3 Registration and Stops

Registration is a crucial aspect of organ notation, as it determines the timbre and character of the sound. Contemporary stop notation takes multiple forms. Traditional text-based instructions are still common, especially when a composer seeks a general timbral result or when registration changes are infrequent.

Klaus Lang's *dunkle schwäne* (Figure 6) demonstrates how mechanical stop action can be used for fine control, including partial engagement of a stop to create slight detuning effects. These gradual stop movements are notated verbally rather than graphically. For example, a change in the $8'$ stop is notated using the following text:

$$8' : 0\%\ \text{tone}\ 0\%\ \text{air}\ \rightarrow 50\%\ \text{tone}\ 50\%\ \text{air}$$

This indicates that the specified $8'$ foot stop should be pulled out gradually and slowly until the sound consists of half tone and half air.

In Alyssa Aska's *looped.(dynamic).* (2017), stop movements are notated on their own staves above the keyboard staves (Figure 7). The performer reads keyboard action from the lower staves and stop motion from the upper staves. Each line represents a stop being drawn or closed over time: an ascending line indicates a gradual opening, while a descending line shows a stop being pushed back in.

This notational choice reflects the musical priorities of the piece. In *dunkle schwäne*, stop changes are also gradual, but the notation only gives verbal instructions and does not indicate precise timing.
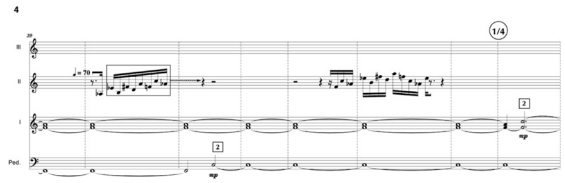


Figure 8. Alyssa Aska: *stagnation.transmutation*.

The emphasis in that case lies on the overall sonic effect and a general sense of transition rather than exact alignment of tuning or timing. Aska's use of graphical notation indicates that the timing and trajectory of each stop movement are structurally important. The score demands coordination between stop movement and manual activity, suggesting that the resulting timbral balance between manuals and their changes over time is central to the composition's structure.

3.4 Intonation and Tuning

One of the defining characteristics of the pipe organ is that each instrument is tuned individually, often to suit the acoustics, repertoire, or historical context of its location. As a result, most organs are actually slightly out of tune and the precise tuning system can vary greatly between individual organs. This presents a particular challenge for composers: the same score may sound in a different temperament or tuning system depending on the instrument on which it is performed.

When writing for mechanical organs, composers must decide whether to accommodate this variability by leaving tuning details more open or to write for a specific instrument whose tuning and temperaments are known in advance. While detailed intonational instructions can be provided in the latter case, such precision is only meaningful when the performance instrument matches the one for which the piece was composed. In general, mechanical organs cannot be retuned during performance.

Electronic and MIDI-based organs often include tuning presets or other features that allow for real-time pitch control. For example, the custom built Rodgers organ at University for Music and Performing Arts Graz, includes several built-in tuning modes, similar to systems found in the commercially available Viscount e-organ series. In Alyssa Aska's *stagnation.transmutation* (2019) (Figure 8), tuning changes are indicated with simple text above the staff, similar in style to stop annotations.

The symbol ¼ above the staff instructs the performer to switch to quarter-comma meantone. More advanced tuning changes are possible using external control systems or hyper-organ extensions.

One such extension is The Frescobaldi[2], attachment, which expands the traditional keyboard by adding five additional keys per octave, enabling performers to distinguish between enharmonic pitches such as G♯ and A♭. This system allows for precise microtonal and historical tunings that would otherwise be impossible on a standard 12-note octave layout. [5, 6]

Rather than inventing a new notational system, composers

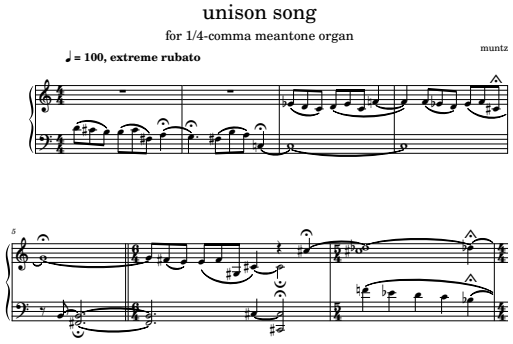


Figure 9. Mat Muntz: *unison song*.

writing for the Frescobaldi[2] typically continue to use conventional accidentals in the score (e.g. G♯ vs. A♭), treating them as distinct enharmonic pitches rather than enharmonically equivalent. In this way, the notation itself remains familiar, while the instrument provides access to the expanded pitch set. A similar approach can be used for historical organs with split keys, where notes such as D♯ and E♭ are assigned to physically distinct keys. Standard notation is adapted in both of these cases to instruments with non-standard tuning layouts, allowing composers to write microtonally without altering the notational language. In Figure 9 from *unison song* (2024) by Mat Muntz, the composer specifies that the work is for quarter-comma meantone organ, and pitches with both enharmonic spellings are used.

As with registration, there is currently no standardized system for notating microtones on the organ. However, composers often adapt one of several established microtonal notation systems such as arrows, accidentals (e.g., half-sharp, three-quarter flat), or extended symbols depending on their aesthetic goals. Since most organs do not include microtonal intervals by default, performers typically interpret these notations flexibly, selecting the closest available pitches on the instrument. On instruments with extended tuning systems, such as split-key historical organs or enhanced digital organs, microtonal notation can correspond more directly to actual pitch options. In all cases, the feasibility of microtonal realization depends on the specific instrument, and composers must consider this variability when scoring microtonal material for the organ.

3.5 Electronics and Technological Extensions

In addition to expanding the possibilities afforded by organs through the use of technique development, organs themselves can also be expanded with hyper-organ extensions and technological interfacing. One work that expanded the possibilities of an organ beyond even extended techniques is Pablo Mariña's *dba_2* (2022) (Figure 10) for organ and electronics. The piece is instrument and system-specific.

The registrations are intended for the Rodgers organ, and the electronics are designed to work with the existing multichannel system. The notation is primarily prescriptive for the organ and the electronics. Mariña provides succinct

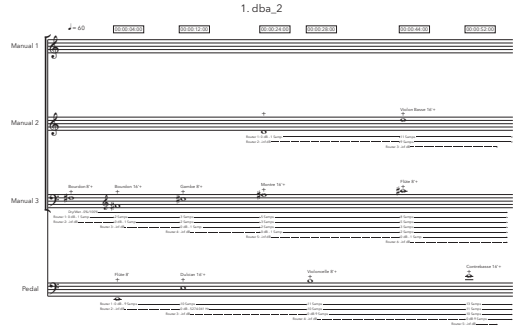


Figure 10. Pablo Mariña: *dba_2* score example.

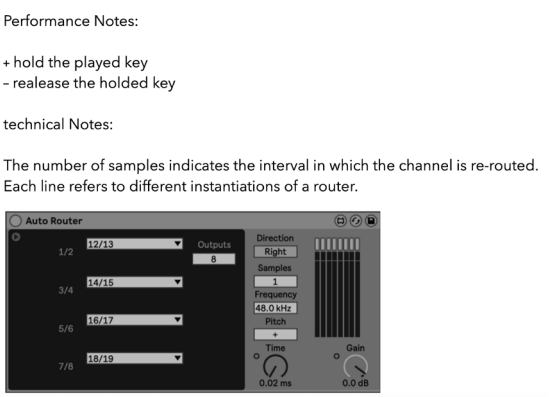


Figure 11. Pablo Mariña: *dba_2* program note.

but precise indication in the program notes (Figure 11). As can be seen in the second image, this information is easily annotated under the staff. The sample and router information is indicated under the corresponding pitch. Mariña also chooses to use a separate staff for each manual and the pedal. This layout choice emphasizes the independence of each manual and the pedal.

4. SOLUTIONS, IMPACT ON PERFORMANCE AND COMPOSITION

There are many possibilities for notating different aspects of organ music. Even something as basic as the number of staves can vary widely between different pieces. This means that organ music has some of the largest variance in notation from piece to piece. This is largely due to the lack of instrumental standardisation. However, this aspect is also what makes the organ such a unique and interesting instrument to compose for. Each new organ is a completely new instrument, with different technology and different sound possibilities. In Western art music, sign and sound (notation and performance) are traditionally considered to be poles of a dialectic [7] that can be modified over time but not suspended, because the music is both "text that is fixed by the medium of writing, so that it can be passed on, and sound event, which is momentary and ephemeral." [8] We are not prescribing a specific type of notation or saying that one is superior to another for organ writing. However, the impact that notation has on both the musical composition and the performative realisation is extremely important. The relationship between notation and importance of musical materials was already discussed, and the many examples highlight how diversity of organ notation contributes to a diversity of foregrounded or blended musical material. This also means that organists need to have a specific relationship with interpretation. Because organs are not standardised, organists need to be adept at both interpreting a score, and having the tools to realise a score on any given organ. Understanding organ interpretation involves not just bringing musical pitches on a staff to life, but also finding solutions when an indicated registration is not possible, or when a given technique is not available on the instrument. Not even the number of manuals is standardized across organs, meaning that organists sometimes have to arrange compositions to be playable on fewer manuals than indicated. This places extreme importance on clarity and efficiency in notation, so that composers are able to get their intentions across effectively in such a way that an organist can understand not just what is written, but what is intended. Decisions such as indicating specific names of stops (i.e., Blockflöte $8'$) or simply categories (i.e., Flutes $8'$) require much more careful consideration than one may anticipate. The former indicates that the sound should have specifically the characteristics of a recorder, likely a wooden pipe with a soft sound and muted overtones.

An organist can understand this, and make a decision to choose a stop with a similar characteristic on a given organ, even if "Blockflöte" is not exactly available. Writing "Flutes" leaves the decision more open for the performer. There is a certain quality of sound expected, and definitely not a reed-based sound, but it could be something with a brighter sound than what Blockflöte would indicate. That doesn't mean that either is superior to the other, but that the choice of how to notate will have a very large impact on the performer's choices. It is possible that the composer does intend for more of a general flute sound, and prefers to keep this aspect more general. But it is also possible that the composer wishes for a precise timbre of the Blockflöte, in which case they would indicate this very specific sound, even if it might not be available on a given instrument.

In *Notation for Organ Extended Techniques* [9], Claudio Pina outlines solutions for standardizing the notation of organ extended techniques, including play and pause symbols for motor control and dynamic lines to indicate gradual stop changes.

While such standardization offers the benefit of immediate legibility and potentially reduces the need for extensive performance legends, our analysis suggests that a universal approach may not always be the optimal solution. The primary concern with eliminating legends or imposing a single notational standard is the potential to inadvertently obscure a composer's specific intentions. As demonstrated by examples like Klaus Lang's *dunkle Schwäne* and Alyssa Aska's *looped.(dynamic)*, the choice of notation, whether that be in the form of verbal instructions or precise graphical lines, directly reflects the composer's em-

phasis on particular musical parameters, such as the overall sonic effect versus the exact timing of a stop movement. A standardized symbol, while clear, might result in the inadvertent prioritizing of one aspect (e.g., volume change) over another (e.g., subtle timbral evolution or intonational nuance), thereby reducing the notational framework's capacity to convey the full breadth of a composer's musical ideas. The diverse and often unique characteristics of individual organs necessitate a flexible notational approach, one that allows composers to tailor their instructions to specific instrumental possibilities and artistic goals, rather than being constrained by a one-size-fits-all system.

5. CONCLUSION

We have examined a spectrum of notational strategies for contemporary organ music, from the broad dynamic indications of César Franck to the highly specific stop registrations of Olivier Messiaen, and to the graphical and text-based innovations in works by contemporary composers such as György Ligeti, Martin Ritter, Klaus Lang, and Alyssa Aska. Each approach reflects a deliberate choice by the composer regarding what constitutes the *substance* of their musical ideas, whether that be pitch, timbre, gesture, or spatial effect. Rather than advocating for a singular, universal standard, we argue that the efficacy of organ notation lies in its adaptability and its capacity to precisely convey the composer's priorities for a given work, even if this means embracing ambiguity or requiring significant interpretive agency from the performer. Ultimately, the ongoing evolution of organ notation underscores the dynamic interplay between composer, performer, and the instrument itself. Understanding the historical precedents and contemporary innovations in notation provides a robust toolkit for composers to make informed decisions that ensure their musical visions are effectively communicated and realized across the diverse landscape of pipe organs. A nuanced understanding of these notational choices empowers performers to navigate interpretive challenges, adapt works to specific instruments, and ultimately bring the composer's intentions to life. As new technologies and compositional approaches continue to emerge, the dialogue between notation and performance will undoubtedly continue to shape the future of organ music, highlighting its unique position at the intersection of fixed text and ephemeral sound.

Acknowledgments

This research was funded in whole or in part by the

- Austrian Science Fund (FWF) **10.55776/ART8035524**
- University of Music and Performing Arts Vienna

6. REFERENCES

[1] M. Ritter, "Playing the field: A holistic approach to the analysis of electroacoustic music using higher order ambisonic technologies, computer vision, and the composer's collaborative environment." [Online]. Available: https://prism.ucalgary.ca/handle/1880/112917

[2] T. F. Kelly, *Capturing Music: The Story of Notation*. W.W. Norton & Company Ltd., ISSN: 0009-4978.

[3] K. Lang, "Liebe und notation." [Online]. Available: https://www.voicerepublic.com/talks/liebe-und-notation

[4] A. Ravain, "Performance practice considerations in the organ works of olivier messiaen."

[5] K. Lang, A. Aska, M. Ritter, and P. A. M. Montalvo, "Imagination as the instrument of the composer: developing the frescobaldi enharmonic keyboard enhancement," in *Mikrotöne: Small is Beautiful: International Symposium: "Mikrotöne: Small is Beautiful"*, A. Castilla-Ávila, Ed. Mackingerverlag, vol. V, pp. 112–124. [Online]. Available: http://repository.moz.ac.at/obvumshs/9440913

[6] M. Ritter, A. Aska, P. Mariña, and K. Lang, "Frescobaldi2: applying historical split-key keyboard models to modern electronic organs," in *Proceedings ICMC2025*.

[7] A. Aska and M. Ritter. Looking back, looking forward: Reflecting on the past for a notation of the future. [Online]. Available: http://econtact.ca/19_3/aska-ritter_notationparallels.html

[8] G. Borio, "The relationship between musical notation and performance after 1950: Historical survey and theoretical considerations," in *CIRMMT Distinguished Lectures in the Science and Technology of Music*. [Online]. Available: http://www.cirmmt.org/activities/distinguished-lectures/borio

[9] C. de Pina, "NOTATION FOR ORGAN EXTENDED TECHNIQUES." PRISM Laboratory, pp. 57–61.

FROM SHAPE TO MUSIC: CONTOUR-CONDITIONED SYMBOLIC MUSIC GENERATION

Qiaoxi Zhang
Centre for Digital Music,
Queen Mary University of London
qiaoxi.zhang@qmul.ac.uk

Mathieu Barthet
Centre for Digital Music,
Queen Mary University of London
Aix-Marseille Univ CNRS PRISM
m.barthet@qmul.ac.uk

Anna Xambó Sedó
Centre for Digital Music,
Queen Mary University of London
a.xambosedo@qmul.ac.uk

ABSTRACT

We present a novel approach to symbolic music generation that enables users to guide melodic trajectories through abstract contour inputs. This work explores how intuitive, high-level melodic controls can influence the expressive shape of generated music, paving the way for tools that grant users more creative control. Our framework consists of two main stages. In the first stage, we extract melodic contour from symbolic music using the Ramer–Douglas–Peucker (RDP) algorithm, which simplifies the melodic trajectory while preserving its essential directional shape. Second, a Transformer-based generative model is conditioned on user-specified contour tokens to produce polyphonic continuations that follow the desired melodic shape. To strengthen contour adherence, we introduce a contour-aware loss based on cosine similarity between the target contour vector and the generated melodic motion.

Objective evaluations on the MAESTRO dataset indicate that the proposed model closely aligns with target contours while preserving overall musical quality comparable to a baseline Music Transformer. These results highlight the feasibility of high-level contour-conditioned music generation and point toward future applications that integrate contour-based inputs with modalities such as hand gestures in VR, enabling intuitive, real-time human–AI co-composition.

1. INTRODUCTION

Controllability in AI music generation is rapidly advancing, with notable progress in recent years, yet it remains an open and actively explored research area. Recent advances in controllable music generation have explored a wide range of conditioning inputs, including chord [1, 2, 3], rhythm [1, 2], style [4, 5], emotion [6], and even textual or visual modalities [7, 8, 9]. These approaches enable users to steer the generation process toward desired musical attributes. While a few prior works have considered contour-based conditioning, direct and flexible control over melodic structure remains relatively underexplored.

Copyright: ©2025 Qiaoxi Zhang et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

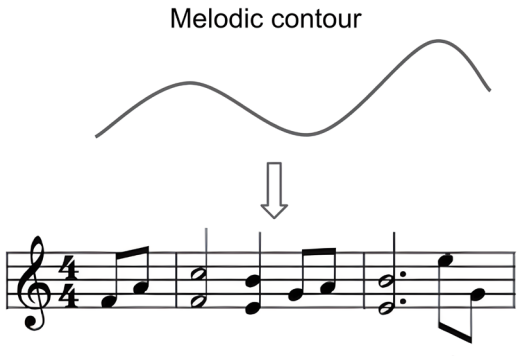


Figure 1. Illustration of abstract contour-conditioned music generation (Melody fragment inspired by Call of Silence, adapted by the authors).

Melodic contour captures the explicit shape and direction of pitch movement over time and is central to human perception and music composition [10].

Traditional instrumental performance often requires complex motor skills that are learned. Our longer-term design goal is to propose an alternative approach to music performance and composition that establishes more intuitive links between user actions and the resulting music, thereby lowering the learning barrier and enabling more direct expression of musical ideas. As a first step, we design a contour-conditioned music generation system and evaluate it quantitatively.

This study explores new ways for composers to shape machine-generated music through high-level melodic sketches. First, we present a novel method for extracting high-level melodic contours by combining time-pitch segmentation with Ramer–Douglas–Peucker (RDP) simplification, producing a compact yet expressive representation of melodic direction. We extend the Music Transformer [11] with contour tokens and a contour-aware loss, enabling the generation of polyphonic, musically coherent continuations aligned with user-defined contours and facilitating collaborative human–AI composition.

2. RELATED WORK

Melodic contour has been extensively studied in the context of audio-based music analysis, where the goal is to estimate and interpret pitch trajectories from audio signals [12, 13].

These approaches rely on fundamental frequency (F0) extraction and are often used in tasks such as music transcription, genre classification, or melody identification. In contrast, our work operates directly on symbolic music (e.g., MIDI), where pitch and onset information is precisely available. This enables a more direct and interpretable extraction of high-level contour, which we further use as a conditioning signal in generation.

For polyphonic symbolic music, a prerequisite for contour analysis is melody extraction, which aims to identify the main melodic line from multiple voices. Early rule-based methods such as the skyline algorithm [14] simply select the highest note at each onset. Later approaches evolved toward data-driven models: piano-roll based convolutional networks treat scores as images [15], while sequence-based recurrent neural networks capture contextual dependencies among notes [16]. Feature-based systems [17] use handcrafted note properties such as pitch, duration, and note dissonance. Recently, transformer-based pre-trained models like MIDI-BERT [18] have demonstrated strong performance on melody extraction as a downstream symbolic task, reflecting a shift from heuristic to representation learning approaches.

Prior research in symbolic music generation has extensively explored melody-conditioned approaches over the past decades. MusicGen [19] and MusicLM [7] transfer the style of an input melody based on textual descriptions or extracted melodic features. Works such as [20] and [21] investigate note-level contour manipulation: [20] proposes an approach to generate variations or mixtures of multiple contours, while Piano Genie [21] maps keystrokes from a simplified 8-key interface to a full piano keyboard, following the direction of pitch movement on the smaller keyboard. GaMaDHaNi [22] introduces expressive elements to music by modeling singers' vocal melodies extracted from audio recordings and generating ornamental notes for a given sequence. Music SketchNet [23] performs pitch interpolation by transforming a set of user-specified pitches into a complete sequence.

Some systems have also explored mapping abstract contours to melodies, which is more closely related to our work. For example, Hyperscore [24] introduced a melody-pattern-based composition interface, where different colours represent predefined melodic patterns and the drawn shapes control their transformations. This approach allows novice users to compose music by drawing, but its flexibility is constrained by the small set of predefined patterns available.

Pizzicato [25] allows users to draw a pitch curve to fill a measure. The system is rule-based, requiring explicit rhythm and chord labels for each beat. Notes are chosen as the closest pitch within the chord, independent of surrounding measures, which makes it unsuitable for users without prior music theory knowledge.

Another related system addresses melody inpainting by allowing users to draw curves representing the desired pitch contour, which the system then converts into a melody matching both rhythm and pitch contour [26]. Built on a variational auto-encoder with a melody disentanglement

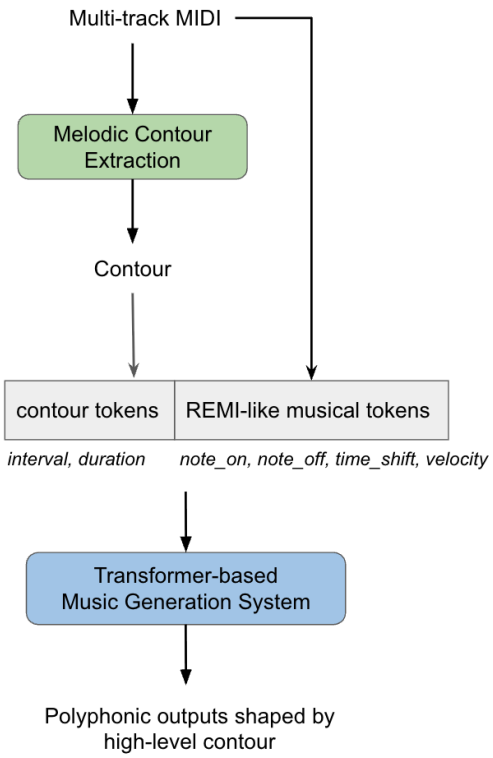


Figure 2. Overall pipeline of contour-conditioned symbolic music generation.

scheme, it enables interactive generation without requiring users to understand music notation. However, this system is limited to monophonic melodies and focuses on filling missing measures rather than generating full polyphonic music from abstract contours. In contrast, our work conditions a transformer-based model on high-level contour tokens to support polyphonic generation, allowing contour-driven composition beyond inpainting tasks.

The system Drawlody [27] enables users to create melodies by sketching pitch contours, utilising a simplified contour representation and a CNN-Transformer architecture to map sketches into melody. Drawlody improves usability by eliminating the need for auxiliary musical inputs like chords or pre-set rhythms, making it accessible to non-expert users. However, it is limited to monophonic melody generation and does not support conditioning on higher-level polyphonic or structural context.

3. METHOD

We develop a contour-conditioned music generation framework that integrates high-level structural information into symbolic music modelling. The overall process consists of two main components: (1) contour extraction, which preprocesses symbolic MIDI files by simplifying melodic motion using an RDP-based algorithm, and (2) a Transformer-based model that learns to generate polyphonic music conditioned on the extracted contour. The contour is encoded as high-level event tokens and combined with REMI-like musical tokens to form the model input. These

contour tokens serve as structural guidance during both training and generation. An overview of this pipeline is illustrated in Fig. 2.

3.1 Contour Extraction using RDP

The dataset used in this project is MAESTRO v2.0.0, a polyphonic piano MIDI corpus [28]. Our contour extraction process involves first deriving a simplified monophonic melody line from the original polyphonic music, followed by encoding the contour information in a format specifically designed to support model training.

The first step is to extract the main melody line, i.e., to identify a monophonic sequence from the polyphonic texture. As a proof of concept, we employed a simplified strategy that selects notes based on pitch and velocity. Specifically, within each sliding window of 1 second (with a hop size of 0.5 seconds), the note with the highest pitch is chosen, with velocity serving as a secondary criterion to break ties or slightly favour louder notes. Here, velocity is weighted roughly one-tenth of the pitch value in the selection process. The pitch–time trajectory of the selected notes is then recorded. Fig. 3 illustrates the result of extracting a monophonic melody line from polyphonic input.

The next and most crucial step is melody simplification, where we apply the Ramer–Douglas–Peucker (RDP) algorithm to identify key turning points in the melodic trajectory while discarding less structurally relevant notes [29]. Originally developed for applications such as computer graphics and cartography, RDP is widely used to simplify complex curves and reduce data while preserving essential shape characteristics. The RDP algorithm recursively eliminates points whose perpendicular distance to the line formed by the first and last points in a segment is below a predefined threshold. In our adaptation, we apply the RDP algorithm recursively using a dynamic threshold tailored to the pitch range of each piece. The process begins with a threshold equal to the smallest pitch difference between adjacent notes and gradually increases, eliminating points until the number of remaining notes falls below a fixed ratio of the total notes in the monophonic sequence. We empirically tuned this ratio to 1/3 to simplify the contour while preserving melodic shape granularity; however, optimisation techniques could be applied in future work to determine a more optimal value.

Finally, we extract melodic contour information from the simplified melody. Each contour segment is represented by a pair: interval and duration, corresponding respectively to the pitch and time differences between the first and last notes of the segment. To produce a more abstract representation of melodic direction, we apply a merging step: consecutive segments with the same direction (ascending, descending, or flat) are combined, and segments with a shorter duration are merged with adjacent segments. Empirically, segments shorter than 2 seconds are merged as they typically reflect fine-grained melodic variation rather than the overall contour. Fig. 4 provides an overview of the full melodic contour extraction pipeline, illustrating the intermediate steps and final result.

3.2 Model Architecture

We base our architecture on the Music Transformer [11], a well-known autoregressive model with relative positional encoding (RPR). Rather than modifying the core architecture of the Music Transformer, we extend its input token vocabulary by introducing two additional event types to represent melodic contour information (see Section 3.3). These contour tokens are injected into the input sequence and processed identically to other events, allowing the model to condition on high-level melodic direction without altering its internal structure.

We enable Relative Positional Representation (RPR) throughout training and inference, which improves the model's ability to capture positional relations between events over long spans. Additionally, we apply label smoothing with $\varepsilon = 0.1$ in the cross-entropy loss to reduce overconfidence in token prediction and enhance generalisation, especially for autoregressive generation tasks with large output vocabularies.

3.3 Contour Token Design

The data is represented using a REMI-like event-based vocabulary [30] consisting of note_on, note_off, time_shift, and velocity tokens. To incorporate high-level melodic guidance, we introduce two additional token types: contour_interval and contour_duration, which respectively represent the expected pitch movement and temporal span of a melodic segment. These contour tokens are inserted immediately before the group of note events they describe. An example token sequence is shown below:

```
<Event type: contour_interval, value: 85>
<Event type: contour_duration, value: 160>
<Event type: time_shift, value: 99>
<Event type: time_shift, value: 0>
<Event type: velocity, value: 10>
<Event type: note_on, value: 62>
<Event type: time_shift, value: 1>
<Event type: velocity, value: 10>
```

Based on an empirical analysis of the dataset, we define the unit and range for each contour token type. contour_interval is measured in semitones, with 201 discrete values ranging from -100 to $+100$ (inclusive), and contour_duration is measured in deciseconds (0.1 seconds), using 150 discrete values from 0.1 to 15.0 seconds. As a result, the overall vocabulary is expanded by 351 tokens to accommodate the additional contour information.

3.4 Contour-Aware Loss

To guide the model to follow the desired melodic contour, we introduce a contour-aware loss that aligns the movement of generated notes with the contour direction. Each contour block is defined by a pair of tokens: `<contour_interval>` and `<contour_duration>`, which precede a group of notes in the token sequence. The interval represents the expected pitch change, and the duration defines the expected temporal span.

We extract vectors representing melodic motion within each contour block by uniformly dividing the interval

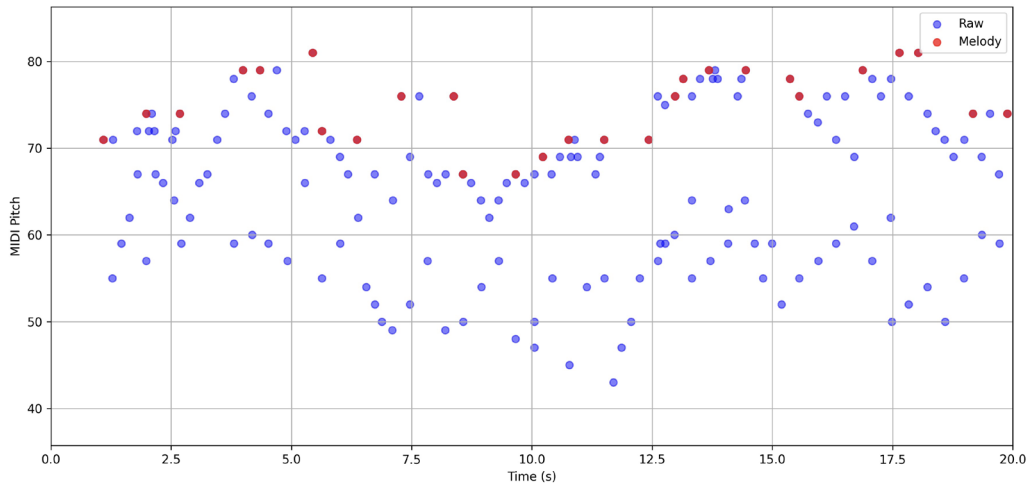


Figure 3. Example of monophonic melody extraction from polyphonic symbolic input. The extracted melody (red) captures the main pitch trajectory selected from the full polyphonic texture (purple).

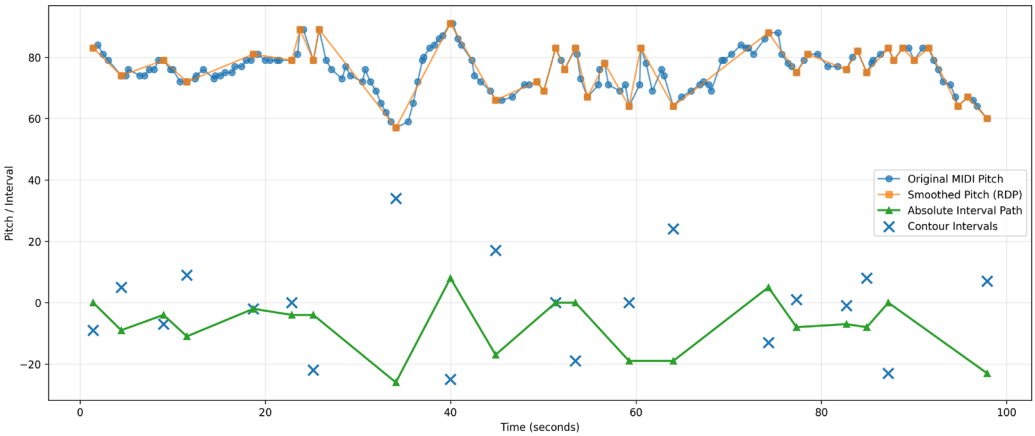


Figure 4. Visualisation of the complete melodic contour extraction pipeline. The figure shows the original MIDI pitch sequence, the RDP-simplified melody, the absolute interval path, and the final extracted contour intervals. The value of each contour interval (blue crosses) indicates the pitch difference between the current and the next blue cross. The absolute interval path is reconstructed from the contour intervals to facilitate comparison with the shape of the original MIDI sequence.

(e.g., into 5 equal-length time slices), and selecting the highest-pitched note in each slice. Each note vector encodes its relative time and pitch difference from the first note in the block. A contour vector is similarly defined as $(\Delta t, \Delta p)$, where Δt is the total contour duration and Δp is the contour interval.

We define the cosine margin loss between note vectors $\mathbf{n}_i$ and the contour vector $\mathbf{c}$ as given in (1).

$$\mathcal{L}_{\text{contour}} = \frac{1}{N} \sum_{i=1}^{N} \max\left(0, m - \cos(\theta_i)\right) \tag{1}$$

where θ_i is the angle between $\mathbf{n}_i$ and $\mathbf{c}$, and m is a fixed cosine similarity margin (empirically set to 0.87, corresponding to an angle of approximately 30 degrees between $\mathbf{n}_i$ and $\mathbf{c}$). This encourages note movements to align with the direction of the contour vector, while allowing flexibility for natural musical variation.

To balance the contour-aware loss and the token-level cross-entropy loss, we introduce a weighting factor λ_{contour} that gradually increases during training. This allows the model to first focus on learning basic token structure before contour-level alignment. The λ scheduler is defined as given in (2).

$$\lambda_{\text{contour}}(e) = \begin{cases} 0, & \text{if } e < 15 \\ 0.02 \cdot (e - 15), & \text{if } 15 \leq e < 30 \\ 0.3, & \text{if } e \geq 30 \end{cases} \tag{2}$$

where e denotes the current epoch. The final training objective is the weighted sum of cross-entropy loss $\mathcal{L}_{\text{CE}}$ and contour-aware loss $\mathcal{L}_{\text{contour}}$, as shown in (3).

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CE}} + \lambda_{\text{contour}} \cdot \mathcal{L}_{\text{contour}} \tag{3}$$

4. EXPERIMENTS AND RESULTS

4.1 Experiment Setup

4.1.1 Dataset and Data Preprocessing

We conduct our experiments on the MAESTRO v2.0.0 dataset [28], a large-scale collection of virtuosic piano performances comprising approximately 200 hours of paired MIDI and audio recordings. In this work, we use only the MIDI portion of the dataset, which provides precise symbolic representations of note events (pitch, velocity, timing) necessary for contour-level modeling and event-based tokenisation. The dataset features expressive, polyphonic piano music with wide pitch coverage across registers, providing a rich and versatile context for learning melodic contours and supporting contour-aware generation. Furthermore, MAESTRO was also used to train the original Music Transformer [11], ensuring compatibility and reproducibility in our architecture and training pipeline.

Each MIDI file is tokenised using a REMI-like vocabulary consisting of note, velocity, and time-shift events. We introduce two new event types: `<contour_interval>` and `<contour_duration>`, which represent contour direction and span as described in Section 3. Time shifts are quantised to 10ms, and contour durations use a coarser unit of 100ms.

4.1.2 Model Configuration

We base our model on the Music Transformer with Relative Positional Representations (RPR) [11]. The model dimension is set to $d_{\text{model}} = 512$ with 8 attention heads, consistent with the original configuration. To ensure sufficient model capacity for learning extended contour-aware sequences, we added two Transformer layers compared with the original model. The maximum sequence length is set to 768 tokens, sufficient to cover a contour span of up to 15 seconds, whereas the original Music Transformer used sequences of up to 2048 tokens to model entire pieces. The model is trained with a batch size of 16 and label smoothing ($\varepsilon = 0.1$).

4.1.3 Training Details

We train the model for up to 300 epochs using the Adam optimizer. To encourage progressive learning of contour structure, we employ a curriculum-like scheduling for the contour-aware loss (Section 3), where the weight λ_{contour} is linearly increased over the first 15 epochs and held constant afterward. This allows the model to first learn local structure before aligning to high-level contour.

4.2 Contour Extraction Analysis

We assess our contour representation on MAESTRO v2.0.0 files, comparing the extracted contour against the monophonic melody line obtained in preprocessing. The method achieves a mean compression ratio of **0.1362** $\pm$ **0.0141**, retaining only $\sim$14% of the original notes while capturing overall melodic direction. Root Mean Square Error (RMSE), widely used to quantify reconstruction accuracy in symbolic or audio-based music generation [31, 32], measures the average deviation between predicted and reference values. Pitch fidelity is measured via RMSE between reconstructed and original trajectories, yielding **8.20** $\pm$ **2.44** semitones. Despite the strong abstraction, visual comparisons (Fig. 4) show that the contour preserves key turning points and the global rise-and-fall pattern of the melody. These results suggest that our RDP-based approach produces a compact yet musically meaningful representation suitable for guiding high-level controllable music generation.

4.3 Music Generation Objective Evaluation

We evaluate the generated music using two main approaches: musicality assessment via MusPy metrics [33] and contour alignment analysis.

4.3.1 Musicality Metrics

We compute several symbolic music descriptors using MusPy [33], as shown in Table 1. The *number of pitch classes used* and *pitch entropy* measure melodic diversity, while the *polyphony rate* captures the complexity of polyphonic textures. The *empty measure rate* and *groove consistency* describe rhythmic completeness and stability.

Results show that the proposed contour-conditioned model (MT+CC) performs similarly to the baseline Music Transformer (MT) model across most musicality metrics, indicating that contour conditioning does not negatively impact overall musicality. MT achieves empty measure rates closer to the groundtruth, indicating better alignment with human-like phrasing. In contrast, MT+CC produces fewer empty measures, likely because specifying a target contour forces the model to generate notes throughout the conditioned segment, leaving less room for natural pauses or silence.

4.3.2 Contour Alignment Metrics

Given a primer clip and a target contour defined by an interval and a duration (see Fig. 5), the model generates a polyphonic continuation. For evaluation, we extract a monophonic melody using a window-based selection method and fit a linear regression to the note pitches over time. We then compute the **slope ratio**, defined as the generated slope divided by the target slope, which measures how well the melodic trajectory aligns with the intended contour. A slope ratio close to 1 indicates strong alignment with the target contour. A slope ratio greater than zero indicates that the generated melody follows the same directional trend as the target contour, whereas a negative value indicates an opposite direction. The absolute value reflects the relative steepness: values larger than one indicate faster pitch changes (steeper melodic motion), while values smaller than one suggest a gentler slope compared with the target contour.

As shown in our results, MT+CC achieves a slope ratio of 1.20 on average, indicating that contour conditioning successfully biases the melodic direction towards the target contour. The ablation model, without contour conditioning, achieves a negative slope ratio (-0.17), demonstrating a lack of alignment with the target direction.

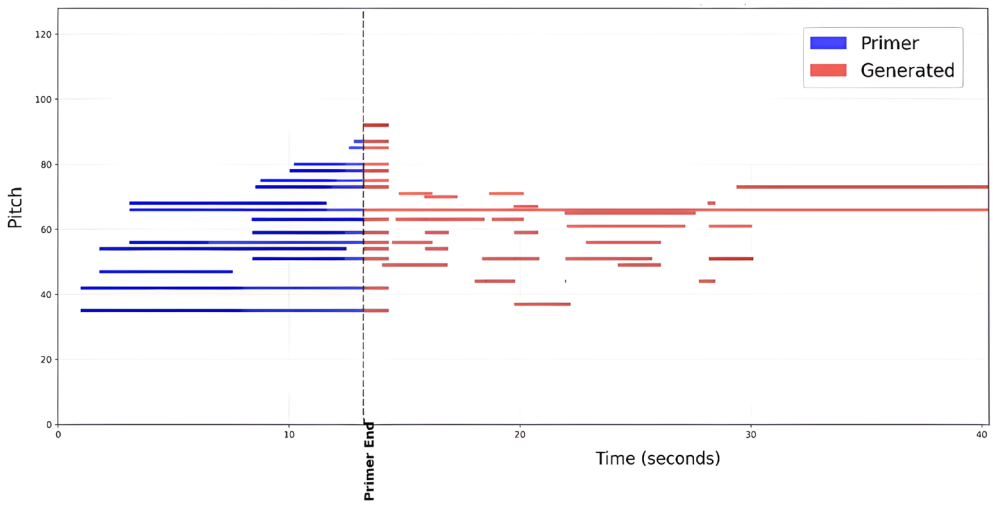


Figure 5. Contour-conditioned generation example. Given the condition to descend by 30 semitones over 10 seconds, the generated melody follows the target contour, dropping from approximately pitch 90 to 60 between 13–23 seconds.

Model	pitch classes	pitch entropy	polyphony rate	empty measure rate	groove consistency
MAESTRO v2.0.0	11.98	5.3762	0.3305	0.0081	0.9762
Music Transformer (MT)	10.26	4.3819	0.6782	0.0075	0.9841
MT+contour-conditioning	10.36	4.4654	0.6591	0.0056	0.9836

Table 1. Comparison of mean musicality metrics between the groundtruth dataset (MAESTRO v2.0.0), the baseline Music Transformer (MT), and the proposed contour-conditioned model (MT+CC).

4.3.3 Interactive Generation

To demonstrate potential user-facing applications of contour-conditioned generation, we implemented a simple interface where users can specify a sequence of contour conditions over time (a rolling window). Unlike real-time systems, the current prototype processes the entire input sequence offline and produces a complete continuation before playback.

Generated examples are available on Zenodo. [1]

5. DISCUSSION

The current simplified strategy for extracting a monophonic melody from polyphonic symbolic music can approximate a main melodic line, but errors and ambiguities are inevitable. Misalignments in timing or selecting harmonically relevant but non-melodic notes can distort the intended melodic contour. As a result, the evaluation of contour-conditioned music generation systems suffers from noise in the extracted reference melody, making it difficult to reliably quantify alignment between generated music and target contours.

Despite these limitations, contour-conditioned generation has strong potential in real-world applications. By allowing users to specify only a high-level contour—through drawing a line or humming a rough pitch trajectory—such systems lower the barrier to music creation. Non-professional users can express musical intent without providing precise pitches, rhythms, or chords, fostering accessibility and democratisation of composition tools. In educational contexts, contour-based interfaces could help students visualise melodic motion and explore variations interactively. Similarly, they could enhance live improvisation setups, where performers guide generative systems via intuitive gestures.

Looking ahead, contour conditioning enhances the controllability and intuitiveness of music AI. This paradigm may serve as a foundation for future interactive systems that combine computational generation with human compositional intent.

6. CONCLUSIONS

This work presented an initial exploration of abstract contour-conditioned symbolic music generation. We introduced a method to extract melodic contours via time-pitch segmentation and RDP simplification, and integrate these contour tokens into a Music Transformer model. Our experiments demonstrated the feasibility of conditioning symbolic music generation on abstract melodic contours, although the results were limited by current melody extraction heuristics and the lack of robust objective evaluation tools.

Future research should focus on improving melody extraction techniques, for example by leveraging data-driven methods to isolate a more musically meaningful main melody line, which would also benefit evaluation reliability. Expanding the conditioning scheme to incorporate additional musical context, such as backing tracks or harmonic information, could increase controllability and improve musical coherence. Human

[1] DOI: `https://doi.org/10.5281/zenodo.17322509`

evaluations are needed to complement automated metrics and better capture perceptual alignment with contour intent. Finally, we envision applications where contour-based input is combined with novel modalities, such as hand gesture control in VR environments, enabling intuitive and expressive human–AI co-composition with real-time responsiveness, where latency and interaction design remain open challenges.

7. ACKNOWLEDGMENT

This work is supported by the UKRI and EPSRC under grant EP/S022694/1.

8. REFERENCES

[1] Y.-H. Lan, W.-Y. Hsiao, H.-C. Cheng, and Y.-H. Yang, "MusiConGen: Rhythm and chord control for transformer-based text-to-music generation," in *Proceedings of the 25th International Society for Music Information Retrieval Conference*. ISMIR, Nov. 2024, pp. 311–318.

[2] O. Tal, A. Ziv, I. Gat, F. Kreuk, and Y. Adi, "Joint audio and symbolic conditioning for temporally controlled text-to-music generation," in *Proceedings of the 25th International Society for Music Information Retrieval Conference*. ISMIR, Nov. 2024, pp. 264–271.

[3] Z. Guo and S. Dixon, "Moonbeam: A MIDI foundation model using both absolute and relative music attributes," *arXiv preprint arXiv:2505.15559*, 2025.

[4] G. Brunner, Y. Wang, R. Wattenhofer, and S. Zhao, "Symbolic music genre transfer with CycleGAN," in *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2018, pp. 786–793.

[5] K. Choi, C. Hawthorne, I. Simon, M. Dinculescu, and J. Engel, "Encoding musical style with transformer autoencoders," in *International Conference on Machine Learning*. PMLR, 2020, pp. 1899–1908.

[6] H.-T. Hung, J. Ching, S. Doh, N. Kim, J. Nam, and Y.-H. Yang, "EMOPIA: A multi-modal pop piano dataset for emotion recognition and emotion-based music generation," in *Proceedings of the 22th International Society for Music Information Retrieval Conference*, 2021, pp. 318–325.

[7] A. Agostinelli, T. I. Denk, Z. Borsos, J. Engel, M. Verzetti, A. Caillon, Q. Huang, A. Jansen, A. Roberts, M. Tagliasacchi *et al.*, "MusicLM: Generating music from text," *arXiv preprint arXiv:2301.11325*, 2023.

[8] H. Liu, Z. Chen, Y. Yuan, X. Mei, X. Liu, D. Mandic, W. Wang, and M. D. Plumbley, "AudioLDM: Text-to-audio generation with latent diffusion models," in *Proceedings of the 40th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., vol. 202. PMLR, 23–29 Jul 2023, pp. 21 450–21 474.

[9] Z. Wang, C. Bao, L. Zhuo, J. Han, Y. Yue, Y. Tang, V. S.-J. Huang, and Y. Liao, "Vision-to-music generation: A survey," *arXiv preprint arXiv:2503.21254*, 2025.

[10] W. J. Dowling, A. Barbey, and L. Adams, "Melodic and rhythmic contour in perception and memory," *Music, mind, and science*, pp. 166–188, 1999.

[11] C. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, C. Hawthorne, A. M. Dai, M. D. Hoffman, and D. Eck, "An improved relative self-attention mechanism for transformer with application to music generation," *CoRR*, vol. abs/1809.04281, 2018.

[12] A. De Cheveigné and H. 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.

[13] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "CREPE: A convolutional representation for pitch estimation," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 161–165.

[14] A. Uitdenbogerd and J. Zobel, "Melodic matching techniques for large music databases," in *Proceedings of the Seventh ACM International Conference on Multimedia (Part 1)*, 1999, pp. 57–66.

[15] F. Simonetta, C. E. Cancino-Chacón, S. Ntalampiras, and G. Widmer, "A convolutional approach to melody line identification in symbolic scores," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

[16] Y.-W. Hsiao and L. Su, "Learning note-to-note affinity for voice segregation and melody line identification of symbolic music data," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*, 2021, pp. 285–292.

[17] H. Zhao and Z. Qin, "TuneRank model for main melody extraction from multi-part musical scores," in *2014 Sixth International Conference on Intelligent Human-Machine Systems and Cybernetics*, vol. 2. IEEE, 2014, pp. 176–180.

[18] Y.-H. Chou, I.-C. Chen, C.-J. Chang, J. Ching, and Y.-H. Yang, "Bert-like pre-training for symbolic piano music classification tasks," *Journal of Creative Music Systems*, vol. 8, no. 1, pp. 1–19, 2024.

[19] J. Copet, F. Kreuk, I. Gat, T. Remez, D. Kant, G. Synnaeve, Y. Adi, and A. Défossez, "Simple and controllable music generation," *Advances in Neural Information Processing Systems*, vol. 36, pp. 47 704–47 720, 2023.

[20] T. Akama, "Controlling symbolic music generation based on concept learning from domain knowledge." in *Proceedings of the 20th International Society for Music Information Retrieval Conference*, 2019, pp. 816–823.

[21] C. Donahue, I. Simon, and S. Dieleman, "Piano Genie," in *Proceedings of the 24th International Conference on Intelligent User Interfaces*, 2019, pp. 160–164.

[22] N. N. Shikarpur, K. M. Dendukuri, Y. Wu, A. Caillon, and C.-Z. A. Huang, "Hierarchical generative modeling of melodic vocal contours in hindustani classical music," in *Proceedings of the 25th International Society for Music Information Retrieval Conference*. ISMIR, Nov. 2024, pp. 1020–1028.

[23] K. Chen, C.-i. Wang, T. Berg-Kirkpatrick, and S. Dubnov, "Music SketchNet: Controllable music generation via factorized representations of pitch and rhythm," in *Proceedings of the 21th International Society for Music Information Retrieval Conference*, 2020, pp. 77–84.

[24] M. M. Farbood, E. Pasztor, and K. Jennings, "Hyperscore: a graphical sketchpad for novice composers," *IEEE Computer Graphics and Applications*, vol. 24, no. 1, pp. 50–54, 2004.

[25] Arpege-Music, "Pizzicato notation software," http://www.arpegemusic.com/manual36/EN855.htm, 2025, online; accessed 20 July 2025.

[26] C. Benetatos and Z. Duan, "Draw and listen! a sketch-based system for music inpainting," *Transactions of the International Society for Music Information Retrieval*, vol. 5, no. 1, 2022.

[27] Q. Liang and Y. Wang, "Drawlody: Sketch-based melody creation with enhanced usability and interpretability," *IEEE Transactions on Multimedia*, vol. 26, pp. 7074–7088, 2024.

[28] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," in *International Conference on Learning Representations*, 2019.

[29] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: the International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.

[30] Y.-S. Huang and Y.-H. Yang, "Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 1180–1188.

[31] A. Kasif, S. Sevgen, A. Ozcan, and C. Catal, "Hierarchical multi-head attention lstm for polyphonic symbolic melody generation," *Multimedia Tools and Applications*, vol. 83, no. 10, pp. 30 297–30 317, 2024.

[32] J. Liu, C. Li, Y. Ren, Z. Zhu, and Z. Zhao, "Learning the beauty in songs: Neural singing voice beautifier," *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, p. 7970–7983, 2022.

[33] H.-W. Dong, K. Chen, J. McAuley, and T. Berg-Kirkpatrick, "MusPy: A toolkit for symbolic music generation," in *Proceedings of the 21th International Society for Music Information Retrieval Conference*, 2020, pp. 101–108.

Chinese Characters as Digital Scores: A Prototype for Explainable, Semantics-Driven Composition

Zhang Yuan
zhang_yuan@ccom.edu.cn

Zhang Xinran
zxinran@ccom.edu.cn

ABSTRACT

This paper presents an explainable semantic-driven composition framework that generates controllable sound code from cultural symbols. Unlike black-box AI music generation systems that focus primarily on stylistic imitation, our approach leverages semantic information—including linguistic meaning and cultural context—from a large language model (LLM) to guide music creation. Building upon and extending the Digital Score [1] paradigm, our framework treats Chinese characters not merely as symbolic inputs but as semantic carriers forming a unique modality of music notation: their semantic and structural features are algorithmically mapped into performable musical instructions. This enables interpretable relationships between symbolic meaning and sonic expression, positioning the character itself as a carrier of musical intention. A prototype system is implemented that maps semantic embeddings of Chinese characters to sonic parameters and symbolic score structures, and implements a prototype capable of producing executable sound code. The prototype allows users to trace, visualize, and adjust semantic-to-sonic mappings, improving creative control and cultural representation. An artistic case study demonstrates how Chinese characters can be transformed into interactive, interpretable digital scores, highlighting the potential of semantic-driven composition to foster transparency, cultural integration, and new modes of creative agency in algorithmic music practices.

1. INTRODUCTION

Artificial intelligence (AI) has significantly transformed music composition, offering novel ways to generate sound materials, automate workflows, and explore creative boundaries beyond human limitations. Yet many AI-driven systems operate as "black boxes," focusing on stylistic imitation or statistical pattern generation with limited interpretability. This opacity raises challenges for composers and researchers, particularly when artistic control and cultural representation are at stake.

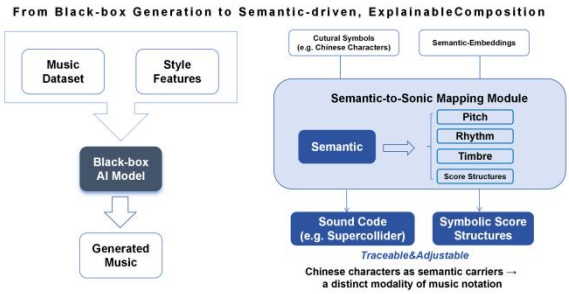


Figure 1. Comparison between black-box AI music generation and our semantic-driven, explainable composition framework, highlighting Chinese characters as semantic carriers forming a unique modality of music notation.

Digital Score, as conceptualized in recent research, has reimagined notation as an interactive, computationally mediated space for creative performance. This work builds upon and extends that concept by introducing a semantic-driven approach [1, 2] in which Chinese characters themselves function as semantic carriers forming a unique modality of music notation. Their semantic and graphic structures are computationally interpreted as performable musical instructions, enabling the character to serve as both cultural sign and music notation. Instead of relying solely on numerical features or stylistic references, our framework integrates semantic embeddings and algorithmic mappings by leveraging the powerful semantic modeling functionality from a large language model (LLM), allowing users to trace and control how meaning informs sound generation.

We present a prototype implementation using Chinese characters as a case study. Semantic features derived from the LLM for the linguistic and cultural attributes are mapped to sonic parameters and symbolic score structures, producing sound code executable in SuperCollider. In doing so, the prototype reframes the Chinese character as a performable and interpretable digital score, enriching the conceptual space of notation by merging linguistic semantics with musical structure. This design offers explainable semantic-to-sonic mappings and supports creative agency.

The contributions of this paper are threefold: (1) it proposes a framework for explainable, semantic-driven composition that extends Digital Score concepts; (2) it implements a prototype that links cultural symbols—here, Chinese characters — to sound code as performable notation; and (3) it presents an artistic case study illustrating how semantic meaning can be transformed into interpretable musical structures using LLM. These contributions aim to bridge semantic representation, cultural expression, and explainability within algorithmic music composition.

2. AI COMPOSITION: SEMANTICS, EXPLAINABILITY, AND CULTURAL SYMBOLS

Artificial intelligence has reshaped compositional practice, providing automated generation systems capable of producing musical material in a wide range of styles. Early approaches focused primarily on statistical models and style imitation, including Markov models, genetic algorithms, and, more recently, deep generative models such as recurrent neural networks and Transformers. While these systems successfully capture stylistic features, they often function as "black boxes," offering limited insight into how input data shapes the resulting output. This limitation has stimulated interest in semantic-driven [2, 3] approaches, where the goal is not simply to reproduce stylistic surfaces but to integrate meaningful relationships between symbolic input and sonic output. Existing research in semantic composition explores emotional mapping, text-to-music transformation, and cross-modal correspondences, but has rarely considered cultural symbols as structured semantic carriers.

The rise of Explainable AI (XAI) has brought new possibilities for music technology, emphasizing transparency and user control. In composition systems, explainability allows composers to trace algorithmic decisions, adjust creative parameters, and integrate machine-generated content within coherent artistic visions. Most XAI research in music, however, focuses on model interpretation or user interfaces, rather than on the semantic transparency of the compositional material itself. Our approach extends this discussion by proposing that semantic meaning — encoded as structured embeddings from cultural symbols—can become a core element of an explainable compositional process.

The use of cultural symbols in music is not new: calligraphic scores, poetic transformations, and visual music traditions have long sought to merge language, image, and sound [4, 5]. In digital media art, these ideas have evolved into cross-modal mapping techniques, including graphical scores, live visuals, and multimodal generative systems. However, most existing work treats symbols either as aesthetic inspiration or as fixed visual references, rather than as computationally interpretable semantic carriers. By framing Chinese characters as structured semantic entities and integrating them into an algorithmic composition pipeline, our research situates itself at the intersection of cultural representation, semantic processing, and digital score innovation.

While there is growing work on AI composition, explainability, and cross-modal use of cultural symbols, these areas have seldom converged in a unified semantic-driven framework. Our system design, presented in the next section, responds to this gap.

3. SEMANTIC-DRIVEN DIGITAL SCORE FRAMEWORK

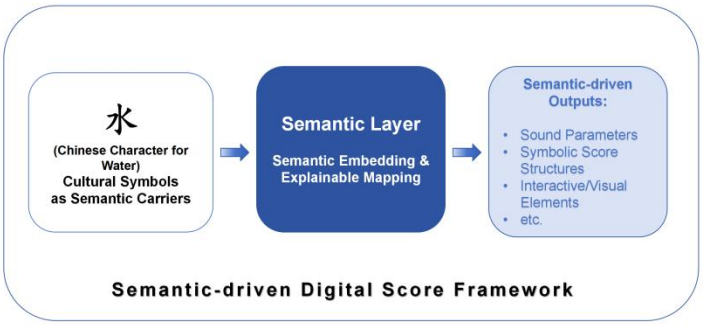


Figure 2.Semantic-driven Digital Score framework, showing how cultural symbols are processed through a semantic layer to generate diverse outputs such as sound parameters, symbolic score structures, and interactive/visual elements.

The core principle of our framework is to treat cultural symbols not as decorative triggers but as semantic carriers that can directly shape musical structure and sonic behavior. Instead of relying on opaque, style-imitating generative models, we introduce an explainable pipeline where semantic meaning flows transparently from symbol to sound. This design builds on the Digital Score concept by adding a semantic layer that translates linguistic and cultural information into computationally interpretable musical instructions. As a result, composers can trace how symbolic meaning influences pitch, timbre, and rhythm, while retaining creative control and cultural specificity.

3.1. Semantic Architecture and Processing Pipeline

The proposed framework extends the concept of the Digital Score by embedding semantic processing directly into the compositional pipeline. Instead of treating input symbols as decorative elements or fixed triggers, our system interprets cultural symbols — here, Chinese characters — as semantic carriers that influence compositional decisions. The architecture is structured around three layers: (1)semantic processing, which extracts linguistic and cultural meaning from the symbols; (2)semantic-to-sonic mapping, which translates this meaning into sound parameters and symbolic score structures; and (3)output rendering, which produces executable sound code and interpretable digital scores. This layered design ensures that symbolic meaning flows

transparently through the pipeline, resulting in a unique modality of music notation where cultural semantics directly shape musical outcomes.

At the semantic processing stage, each Chinese character is converted into its semantic embedding using an open-source LLM that has been pre-trained on Chinese text corpora rich in cultural content. This embedding captures contextual semantic nuances and associative meaning beyond the visual appearance of the character. To translate semantics into musical form, we adopt a rule-based mapping layer ensures interpretability by aligning specific semantic dimensions (e.g., "fluidity," "brightness," "motion") with musical attributes such as pitch contour, timbral quality, and rhythmic density. This allows composers to trace how symbolic feature informs sound generation while still benefiting from adaptive AI-driven mapping. The mapped parameters are then encoded as SuperCollider code, supporting direct sound synthesis, and as symbolic score structures following the Digital Score paradigm, ensuring both computational execution and performable interpretation.

3.2. Use of LLM in Semantic Processing

A key aspect of this work is the use of LLM for semantic processing and mapping. We adopt the LaBSE model [6] to convert each Chinese character into its semantic embedding. By leveraging the contextual semantic modeling power of LLM, these embeddings capture nuanced cultural and contextual representation beyond simple lexical definitions. To translate these semantic features into musical parameters, we employ a mapping function to convert the embedding into parameters that control the synthesization of music sound in SuperCollider. This LLM-driven design allows adaptive and explainable transformation of cultural semantics into sound parameters and symbolic score structures, supporting both creative flexibility and transparent control.

3.3. Cultural Extensibility

While this work is centered on Chinese characters as the primary symbolic source, their role goes beyond that of a simple demonstration. Chinese characters form a major branch within the framework, providing both a testbed for prototyping and a rich, culturally grounded symbolic system that can scale within the architecture. Their complex structure and deep semantic associations make them ideal for exploring how meaning can be algorithmically linked to musical parameters and digital score structures.

At the same time, the framework is designed for broader cultural integration. Because the semantic embedding and mapping modules are modular, they can be adapted to other symbolic systems such as Arabic calligraphy, indigenous glyphs, or abstract graphic signs. Each of these systems carries distinct visual and semantic characteristics, requiring only domain-specific embedding models and minimal training data to capture their meaning. By using multilingual or culturally specialized pre-trained LLMs, the framework can generalize semantic processing across diverse writing systems and symbolic traditions.

This flexibility positions the framework as a foundation for cross-cultural semantic composition, where diverse symbolic traditions can be interpreted as meaningful musical notation. Beyond technical adaptation, this approach supports artistic and educational applications, such as cultural heritage projects or immersive multimedia performances that link local symbolic vocabularies with dynamic sound and score structures. As a result, the system provides both a platform for technical experimentation and a space for cultural dialogue, with Chinese characters representing a significant and evolving branch of semantic-driven digital scoring.

4. CASE STUDY OF THE FRAMEWORK

In the prototype system developed for this research, **the moon** was chosen as the thematic focus.[1] Literary materials related to the moon in different languages were mapped into a semantic space to demonstrate the operability and cross-cultural applicability of the digital score framework.

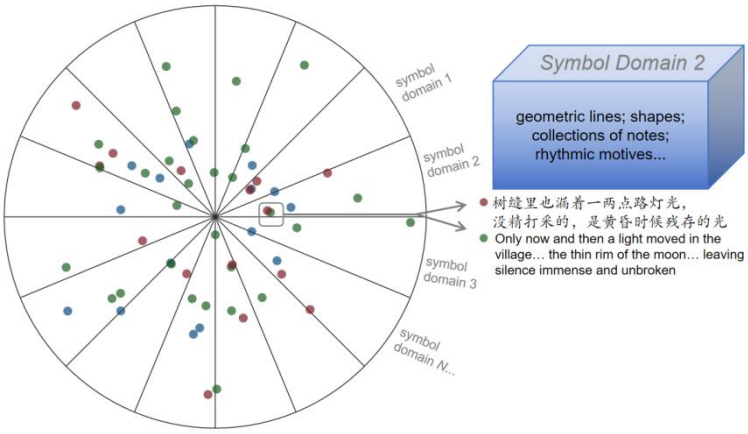


Figure 3. Semantic mapping of text fragments into symbolic domains for digital score representation

4.1. Data and Corpus

To ensure cultural and linguistic diversity, three categories of texts were selected:

- Chinese: Zhu Ziqing's modern prose-poem *Moonlight over the Lotus Pond* (荷塘月色).
- Latin: The medieval Latin poem *O Fortuna*(from *Carmina Burana*), in which the imagery of *Fortuna* as changeable as the moon serves as the connection.
- English: Moon-related passages from Virginia Woolf's novel *To the Lighthouse*.

[1]This case study isolates the notational/performative pathway; the sound-code pathway is implemented in the prototype but is not evaluated here to avoid potential confounds.

All of these texts contain imagery directly or metaphorically associated with the moon, forming resonances across different cultures and contexts.

4.2. Construction of the Semantic Space

The texts were segmented into their smallest semantic units: paragraphs for the Chinese material, stanzas for the Latin poem, and descriptive fragments for the English text. Each unit was represented as a semantic point, mapped into a high-dimensional semantic space and then projected into two dimensions through dimensionality reduction.

- Color: distinguishes language (Chinese = red, English = green, Latin = blue).
- Angle: divides the semantic space evenly into angular sectors, so that language points with different semantic orientations are clearly distributed.

In this way, composers and performers can visually observe how multilingual materials are distributed and clustered within the same semantic field.

4.3. Compositional Symbol Domains

The semantic space is evenly divided into *N* angular sectors, and for each sector a **compositional symbol domain** is defined.

Each sector functions as a compositional symbol domain, that is, a concept predetermined by the composer, such as a theme, pitch sequence, rhythmic pattern, or performance technique.

These symbols do not function as mechanical notes but as triggers for improvisation: when a performer encounters a point located within a given sector, they are required to improvise according to the musical content indicated by the corresponding symbol domain.

The semantic space thus becomes a conceptual map, establishing associations between natural language and musical symbols. This association makes full use of the properties of semantic space, breaks down barriers between languages, and ultimately establishes explicit mappings to musical symbols.

In terms of performance logic, improvisation is carried out according to the compositional symbol domains determined by the sequence of semantic points within the semantic space .

4.4. Summary of the Case

The moon-themed case study highlights the core features of the framework:

(1) A unified semantic space across languages: Chinese, English, and Latin texts are juxtaposed within the same space, allowing performers to intuitively perceive the convergence and divergence of cross-cultural imagery.

(2) Semantic similarity-driven spatial distribution: text units with similar meanings are distributed in adjacent regions, thereby creating a layered structure of semantics at the macro level.

(3) Composer's personalized symbol domains: provide performers with conceptual cues rather than fixed prescriptions.

(4) A performance logic combining sequence and symbol: the sequence is determined by the numbering of language points, while the symbols are created by the composer; improvisation emerges under the guidance of these symbols.

This case demonstrates the feasibility of the semantic-driven digital score and highlights its potential in uniting multilingual texts, musical composition, and performative interpretation.

5. CONCLUSION

The key insight of this study is that the proposed framework is not merely a mechanism for mapping between text and sound, but is gradually constructing **a new notational language** [7]. Unlike traditional scores that rely on abstract musical symbols, this language is rooted in semantic carriers (such as Chinese characters) and extends into multiple dimensions of visuality, sound, and performance.

This new language exhibits the core characteristics of a symbolic system:

(1)Lexical level: each symbol module mapped from a semantic point retains its cultural meaning while simultaneously encoding performance instructions.

(2)Syntactic level: the arrangement of these symbols along the temporal axis forms a kind of *grammar*, constrained by the semantic ordering of language points derived from cross-linguistic corpora.

(3)Pragmatic level: the same symbol may assume very different expressive roles in different poems and performance contexts. It is precisely this sensitivity to context — one of the hallmarks of pragmatics — that endows the symbolic language with elasticity, adaptability, and vitality, allowing it to continuously generate new meaning in use.

Within this framework, the composer is no longer merely a creator of musical materials, but becomes **a designer of language**. The role of composition is thus redefined: not simply the writing of fixed notes, but the curation of a symbol–semantic space in which performers and language models co-construct meaning. In this sense, composition is the invention and evolution of a digital notational language — one that is culturally grounded, semantically interpretable, and adaptable across languages and works. We evaluated the symbolic/performative path; systematic evaluation of the sound-code path is left as future work.

Acknowledgments

This paper is supported by the following projects:

Digital Scores – investigating the technological transformation of the music score
(DigiScore Grant agreement ID: 101002086)

Research Funding Program of the Central Conservatory of Music (Grant No. 23YB19).

6. REFERENCES

1. C. Vear, *The Digital Score: Musicianship, Creativity and Innovation*. Routledge, 2019.

2. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, vol. 1, 2019, pp. 4171–4186.

3. T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

4. E. Brown, "December 1952," in *Notations*, J. Cage and A. C. Childs, Eds. New York: Something Else Press, 1969, pp. 57–58.

5. C. Cardew, *Treatise*. London: Gallery Upstairs Press, 1967.

6. F. Feng, Y. Yang, D. Cer, N. Arivazhagan, and W. Wang, "Language-agnostic BERT sentence embedding," in *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, vol. 1, 2022, pp. 878–891.

7. N. Goodman, *Languages of Art: An Approach to a Theory of Symbols*. Indianapolis: Hackett Publishing, 1976.