Designing Percussive Timbre Remappings: Negotiating Audio Representations and Evolving Parameter Spaces

Jordie Shier
Centre for Digital Music
Queen Mary University of London
London, UK
j.m.shier@qmul.ac.uk

Rodrigo Constanzo
Royal Northern College of Music
Manchester, UK
rodrigo.constanzo@rncm.ac.uk

Charalampos Saitis
Centre for Digital Music
Queen Mary University of London
London, UK

Andrew Robertson
Ableton AG
Berlin, Germany

Andrew McPherson
Dyson School of Design Engineering
Imperial College London
London, UK

Abstract

Timbre remapping is an approach to audio-to-synthesizer parameter mapping that aims to transfer timbral expressions from a source instrument onto synthesizer controls. This process is complicated by the ill-defined nature of timbre and the complex relationship between synthesizer parameters and their sonic output. In this work, we focus on real-time timbre remapping with percussion instruments, combining technical development with practice-based methods to address these challenges. As a technical contribution, we introduce a genetic algorithm—applicable to black-box synthesizers including VSTs and modular synthesizers—to generate datasets of synthesizer presets that vary according to target timbres. Additionally, we propose a neural network-based approach to predict control features from short onset windows, enabling low-latency performance and feature-based control. Our technical development is grounded in musical practice, demonstrating how iterative and collaborative processes can yield insights into open-ended challenges in DMI design. Experiments on various audio representations uncover meaningful insights into timbre remapping by coupling data-driven design with practice-based reflection. This work is accompanied by an annotated portfolio, presenting a series of musical performances and experiments with reflections.

Keywords

Synthesizer Parameter Mapping, Augmented Instruments, Timbre Mapping, Machine Learning, Practice-Based Research

1 Introduction

Digital musical instruments (DMIs) often involve mapping sensor inputs to synthesizer parameters—a well studied topic [1, 27, 28]. One compelling approach to mapping is to use audio features captured from an instrumental performance as the primary input. Such a mapping enables musicians to leverage proficiency on a familiar instrument to control new sounds, effectively "recycling virtuosity" [52].

In this work, we explore a method for audio-to-synthesizer mapping through timbre remapping. The core assumption of timbre remapping is that relative differences in timbre should be preserved during mappings—that is, that timbral differences observed between successive notes in a musical phrase are reflected in the sound of the synthesizer. This draws on the concept of timbre analogies [15, 37, 54], or the idea that timbral phrases, like melodic phrases, can be transposed. In effect, we seek to transpose timbral phrases from our input instrument onto our synthesizers. Such a mapping enables deliberate navigation of the timbre space of a synthesizer from an instrument, emphasizing the role of timbre as a structuring force in the creation of musical phrases [35]—particularly salient for percussion [13].

In previous work [45], we designed a timbre remapping system for percussion using differentiable digital signal processing (DDSP) [16], which implemented a differentiable 808 drum synthesizer controlled by a neural network for real-time performance. DDSP enables DSP synthesizers to be integrated with neural networks and trained on audio loss functions using gradient-based optimization; however, restricts synthesizers to differentiable implementations. In this work, we replace gradient descent with a genetic algorithm, which can be used with arbitrary non-differentiable synthesizers such as Virtual Studio Technology[1] (VST) plugins and hardware synthesizers. Audio features, computed on synthesizer outputs, are used by our genetic algorithm to search for parameter settings that match timbral differences observed in recorded drum performances. Offline searches result in a corpus of synthesizer presets and their associated audio features. Real-time, low-latency performance is enabled via neural networks that are trained to predict audio features captured over longer temporal windows (i.e., 250ms)—which were used during search—from features computed on short windows (i.e., 5ms) at a detected onset.

During the design of this timbre remapping system, other questions emerged with less obvious solutions. The most pressing of these questions relates the use of audio features to represent sounds and the impact they have on mappings. Often, audio features do not map to perception in clearly defined ways, which leads to uncertainty regarding which features are most salient to include for timbre remapping. Taking inspiration from previous work that has turned to practice-based methods for navigating complex and open-ended problems in DMI design [10], we engaged in a collaborative design process with a practicing percussionist and researcher—the second author of this paper—to explore this question. We conducted data-driven and practice-based experiments to investigate audio features as they relate to timbre remapping and to explore the musical affordances of our proposed system. Situating our design within the practice

[1] https://steinbergmedia.github.io/vst3_dev_portal/pages/index.html

NIME '25, June 24–27, 2025, Canberra, Australia

of the second author, which involves augmenting drums with machine learning, enabled us to reflect on data-driven results within practice and resulted in meaningful design interventions and insight into the nature of percussive timbre remapping. An annotated portfolio [21] of mapping variations and performances recordings is presented alongside this work on a supplementary website[2], which additionally contains an appendix and code links.

2 Background

2.1 Timbre, Representations, and Analogies

Musical timbre, a concept that has resisted precise definition [31], has been referred to as the "psychoacoustician's multidimensional waste-basket category for everything that cannot be labelled pitch or loudness" [36]. Studies investigating the perceptual differences between instrumental tones have been used to construct a multi-dimensional representations of timbre, referred to as a timbre space [23], and acoustic correlates for each dimension identified [6, 24]. Many audio features that describe timbre have been proposed [39, 40], and have found broad application, including in music information retrieval tasks related to percussion instruments [25, 51] and computational musicology studies on drum kit performances [13].

Research into timbre analogies seeks to identify whether transpositions of timbre sequences, similar to how one might transpose a melody, might be a perceptually viable operation. This concept was first explored by Ehresman and Wessel [15] and was subsequently proposed as a method for DMI control [55]. Later studies verified the perceptual viability of timbre analogies [37]. Given a pair of sounds $\mathbf{x}_a$ and $\mathbf{x}_b$, they asked participants to select a sound $\mathbf{x}_d$ (from a set of choices) that differed from $\mathbf{x}_c$ by the same amount as $\mathbf{x}_b$ differed from $\mathbf{x}_a$. Results showed that $\mathbf{x}_d$ could be predicted by a parallelogram model of similarity within a multi-dimensional timbre space. We use timbre analogies as the basis for performing timbre remapping in this work.

2.2 Synthesizer Programming and Mapping

Programming synthesizers to achieve desired sonic outcomes is a challenging process that has drawn considerable research attention [7, 44, 56]. A common approach is *sound matching*, an inverse problem that involves determining synthesizer parameter settings to reproduce a target sound. Previous work has largely employed genetic algorithms [34, 50] and deep learning [5, 17, 32, 56]. Genetic algorithms iteratively explore large parameter spaces with minimal constraints; however, they tend to be slow and typically solve a single target at a time. Deep learning allows fast inference after training, but their effectiveness can be limited by suboptimal parameter loss functions [17].

Our goal is related to, but distinct from, sound matching: rather than reproducing a sound, we seek variants of a preset that differ in timbre by a specified amount. We propose a formulation of a genetic algorithm to generate datasets of synthesizer parameters for non-differentiable synthesizers. These datasets can either be queried directly or used as training data for a neural network. Evolutionary algorithms have previously been employed to navigate synthesizer parameter spaces; however, primarily focus on an interactive sound design context [11, 46]. For background on evolutionary algorithms within a musical context, we refer the reader to Dahlstedt's comprehensive overview of the topic [12].

Real-time, audio-driven control of synthesizers based on timbral features has been explored with corpus-based concatenative synthesis [43] and vocal-driven mapping to VSTs [20]. Although not focused on audio-driven control, Fasciani proposed a systematic method for analyzing timbre features in relation to synthesis parameters to support timbre-to-parameter mappings [19]. A timbre remapping approach was proposed by Stowell and Plumbley [49], which identified the challenge in mapping between distinct timbral distributions, and proposed an unsupervised regression to tree to support this mapping. A key difference of our timbre remapping approach is the use of timbre analogies for audio-driven control.

2.3 Practice-Based DMI Design

Dahl describes the "problem" of DMI design as a "wicked problem"—a problem that is complex, lacks clear guidelines, and has ambiguous success criteria [9]. Practice-based approaches offer a method for navigating these problems and involve an "iterative conversation with materials through which problem setting and problem solving co-occur [10]." A practical example includes the development of the TANC DMI [57], which involved iterative cycles of practice and redesign. Practice has also been proposed as a method for evaluation in DMI design [29], where examination of user experience in musical context supports reflection on designs and generation of new ideas.

Technical research can also be practice; Pelinski et al. proposed the concept of *technical practice research* as an alternative mode of knowledge production, prioritizing first-person and real-time insights [41]. Green et al. use the term *practice-led design* to describe an iterative methodology incorporating practice research and technical research [22]. In contrast with a traditional design process, they identify a process that involves repeated episodes of practice, reflection on theories, refinement, and re-application.

3 Design Overview

We approach the design of real-time timbre remapping through a process of collaborative, iterative prototyping that is grounded in the musical practice of the second author. In doing so, we aim to address both practical challenges, such as navigating synthesizer parameter spaces and implementing a low-latency performance system, and open-ended problems related to the use of audio features for timbre remapping. We first outline the main design problems and then describe our collaborative design process.

3.1 Problems

3.1.1 Navigating Synthesizer Parameter Spaces. We seek synthesizer parameter settings to match specific sonic attributes. Our initial mappings leveraged DDSP to achieve this task; however, this approach restricts us to synthesizers that are differentiable, limiting generalizability. A key question arises: How can we create datasets of synthesizer presets for non-differentiable, black-box[3] synthesizers—both software and hardware—that possess the desired sonic characteristics?

To state this problem more precisely, given a feature extraction function $f(\cdot)$ that produces a k-dimensional feature vector $\mathbf{x} \in \mathbb{R}^k$ from a window of audio, we can produce a dataset of M audio feature vectors $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\}$ from a set of M individual drum hits extracted from a recording. Timbre trajectories are then

[2] https://jordieshier.com/projects/nime2025/

[3] A system where the internal workings are opaque. Only the inputs and outputs can be observed.

computed by selecting an anchor sound $\mathbf{x}_\alpha \in \mathbf{X}$ and computing the feature difference for all items in the dataset:

$$\Delta\mathbf{X} = \mathbf{X} - 1\mathbf{x}_\alpha^T \tag{1}$$

Then, given a synthesizer, which can be described as a function $g(\phi)$ that maps from synthesizer parameters ϕ to audio, we can compute synthesized timbre trajectories:

$$\Delta\mathbf{y} = f(g(\phi)) - f(g(\phi_\alpha)) \tag{2}$$

where ϕ_α is an anchor synthesizer preset, which we manually select in this work. We seek a set of synthesizer parameters $\mathbf{\Phi} = \{\phi_1, \phi_2, \ldots, \phi_M\}$ such that the resulting synthesized timbre trajectories match those in our input dataset, i.e., $\Delta\mathbf{Y} \approx \Delta\mathbf{X}$, where $\Delta\mathbf{Y} = \{\Delta\mathbf{y}_1, \Delta\mathbf{y}_2, \ldots, \Delta\mathbf{y}_M\}$ is generated by applying equation (2) above to all synth parameters in $\mathbf{\Phi}$.

3.1.2 Audio Representations for Timbre Remapping. Selecting audio features for constructing timbre analogies is another challenge. Although numerous audio features exist and some correlate with perception [6], they often do not map directly or consistently to human experience. Multi-dimensional representations encode a notion that perception is segmented along independent axes. For example, features like spectral centroid and lower MFCCs (minus the 0th coefficient) are chosen for their invariance properties, such as amplitude-invariance under linear scaling, which help isolate timbre characteristics independent of loudness or pitch. These properties are attractive for constructing timbre analogies because they support mathematical operations where sounds can be translated along certain dimensions while being held constant along others. However, research shows that even "basic" sound dimensions interact in complex ways with timbre [42]. Furthermore, perceptual evidence for these analogies is limited to certain translation magnitudes [37]. The entirety of auditory perception does not neatly conform to models. This leads us to ask: How do we make decisions in light of "perceptual incorrectness" of our representations when designing mappings?

3.2 Process

The design and experimentation were conducted collaboratively by the first two authors. Author A is a doctoral researcher studying machine learning and music and has a background in performance with live electronics. Author B is an active performer, improviser, and researcher experienced with augmented percussion and machine learning in Max/MSP.

Our collaboration began with email exchanges and Zoom calls in spring 2024, during which a mutual interest in exploring timbre remapping within Author B's musical-technical practice was established. Early experiments involved exploring our initial DDSP-based system and developing a prototype system using a genetic algorithm. Initial design questions, as outlined above, began to emerge at this stage. During a second phase of development we engaged with the design space of our timbre remapping, particularly as it relates to the use of audio features. Over the course of several months in fall 2024 we developed the timbre remapping system and performed data-driven and practice-based experiments in an iterative process. Remote dialogue between authors was facilitated by sharing video recordings of patches, audio examples, and reflections over email. The final design phase was an intensive, three-day in-person studio session at QMedia Open Studios at Queen Mary University of London. We engaged in practice-based experimentation and reflection through critical listening and performance, and intervened with our designs.

Figure 1: Snare drum with crotales. Drum trigger and DPA microphones attached at the top of the left of drum.

Table 1: Selection of synthesizers used in experiments

Synthesizer	Format	Method	Params
808 Snare[1]	Max	Sines + filtered noise	14
Quantussy[2]	Max	Chaotic	9
Derailer[3]	VST	Physical modelling	91
FM2[4]	Max4Live	FM	9
Eurorack[5]	Hardware	FM / subtractive	7

[1] 808-inspired snare drum from [45]
[2] Digital emulation of the "central analog brain" of the CocoQuantus Ciat-Lombarde; see https://ciat-lonbarde.net/ciat-lonbarde/cocoquantus/index.html
[3] https://physicalaudio.co.uk/products/derailer/
[4] FM2 from Ableton's Max for Live Essentials Drum Synth collection
[5] See supplemental website for description

Drawing from technical practice research [41], we kept audio and written journals, regularly pausing to reflect and discuss new insights.

3.3 Context and Materials

During remote collaboration the second author utilized a snare drum that was optionally extended with crotales (small, tuned cymbals) placed on the top head, and in-person we used a different snare drum and a cowbell to mimic the crotales. Two different acoustic sensors were used. The first is an open-source drum trigger[4] based on the Sensory Percussion[5] triggers and the second a DPA 4099[6]. The two sensors can be used separately for both onset detection and audio feature extraction, or combined, where the drum trigger is used for onset detection and the DPA for feature extraction. A photo of the snare drum used during remote collaboration is shown in Figure 1. We selected a set of software and hardware synthesizers, outlined in Table 1, aimed

[4] https://www.cycfi.com/2023/04/nu-drum-sensor/
[5] https://sunhou.se/
[6] https://www.dpamicrophones.com/microphones/instrument/4099?variant=29

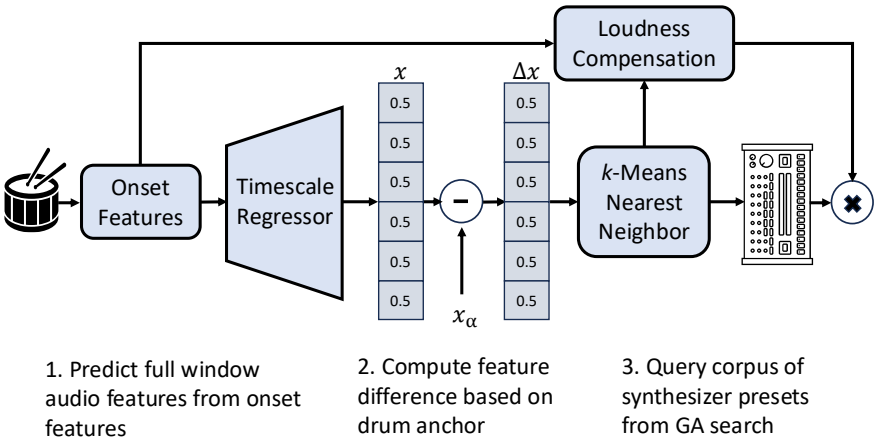


Figure 2: Overview of real-time mapping system.

at covering a range of synthesis techniques with varying number and complexity of parameters—which include time-varying modulations such as envelopes and low-frequency oscillators. A complete list of all employed synthesizer parameters can be found in the appendix. Synthesizers that we regularly use in our own musical practice include: the Eurorack modular synthesizer (Author A) and Quantussy (Author B).

4 Timbre Remapping Design

Here we outline the proposed timbre remapping system. The primary steps involved include:

(1) recording an input dataset for feature extraction;
(2) compute feature differences on dataset;
(3) perform genetic algorithm search with a synthesizer to match feature differences resulting in a corpus of synthesizer sounds;
(4) train neural networks (timescale regressors) to predict audio features from onset features;
(5) real-time performance by querying corpus using feature differences computed on timescale regressor output.

The final playable system is reminiscent of the two-layer mapping model proposed by Hunt et al. [28], although the middle parameter layer in our system represents audio features that can be manipulated to effect meaningful timbral adjustments, an affordance we found useful in practice. A diagram of our resulting system is shown in Figure 2. We developed primarily in Max/MSP using the SP-Tools[7] and FluCoMa[8] packages, with the GA developed in JavaScript (ECMAScript 6+) using the v8[9] package available in Max 9. Neural network training and hyperparameter tuning was conducted in Python.

4.1 Audio Feature Extraction

The first step is to record a small dataset (a few hundred drum strokes) of material from the input instrument, which defines the range of timbre variance for mapping. Onset detection is performed using FluCoMa's *AmpSlice* object, which selects onsets based on a thresholded difference between two time-domain envelope followers—a fast envelope and a slow envelope. A short window is extracted following a detected onset for feature extraction. Three different window sizes were considered: 256 samples ($\approx$ 5.8ms at 44.1kHz), which supports low-latency operation, 4410 samples (100ms) and 11025 samples (250ms), which are used during the offline search phase and were selected to capture low-frequency content and the *morphology* of percussive sounds at different time scales. Here, we refer to morphology as the time-varying sonic characteristics of a sound including the amplitude and spectral evolution.

4.1.1 Audio Features. A selection of audio descriptors were selected based on the second author's practice and those available in FluCoMa. These form four sets of multidimensional audio descriptors, which are outlined in Table 2. All features are computed using frame-based processing. The first-order derivative over time-varying frames is optionally computed and appended. The temporal dimension is then reduced using loudness-weighted summary statistics. Once feature vectors were computed for all samples in the dataset, we optionally normalized along each dimension using the observed minimum and maximum. In practice, we found normalization improved results for hybrid descriptors and spectral shape features, but not MFCCs or Mel-bands. See the appendix for full details of feature extraction.

Each feature set encodes different sonic attributes for comparison. The hybrid descriptor set encodes loudness, pitch, and timbral dependencies. Mel-bands encode loudness dependencies specific to certain frequency ranges. MFCCs without the 0th coefficient have been noted as being relevant to timbre [8], and are less-sensitive to both pitch and amplitude. The spectral shape features provide an alternative perspective on timbre representation and are also less-sensitive to amplitude.

4.1.2 Feature Difference Vectors. Once a dataset of audio features $\mathbf{X}$ has been created, we compute feature differences with respect to an anchor sound $\mathbf{x}_\alpha$, which is selected as the median within the dataset. Concretely, we compute a central sample m by taking the median of each feature independently across all samples in $\mathbf{X}$. We then select $\mathbf{x}_\alpha$ as the sample closest to m using Euclidean distance. We then create a feature difference dataset $\Delta\mathbf{X}$ by subtracting the anchor feature from each item in the input dataset, as described in Equation (1).

[7] https://github.com/rconstanzo/SP-tools
[8] https://www.flucoma.org/
[9] https://docs.cycling74.com/reference/v8/

Table 2: Sets of audio descriptors used

Name	Features	Dimensions	Derivative	Summarization
Hybrid Descriptors	loudness, spectral centroid, spectral flatness, pitch, pitch confidence	8	yes (except pitch)	mean
Mel Bands	40 mel-bands	40	no	mean
MFCCs	coefficients 1-13	104	yes	mean, std, min, max
Spectral Shape	centroid, spread, skewness, kurtosis, rolloff, flatness, crest	14	yes	mean

4.2 Genetic Algorithm Corpus Generation

Here we address the problem of creating datasets of synthesizer presets as described in the problem formulation in Section 3.1.1.

4.2.1 Genetic Algorithm. We use a GA to search for synthesizer presets, which has no constraint on differentiability. Our initial efforts followed previous work on synthesizer sound matching [26, 34] and involved independent searches for each $\Delta\mathbf{x} \in \Delta\mathbf{X}$. We briefly describe the genetic search process for a single target.

The search starts with a population of individuals, i.e., a set of synthesizer presets $\mathbf{\Phi} = \{\phi_1, \phi_2, \ldots, \phi_N\}$ where ϕ_i corresponds to a single candidate synthesizer preset and N is the population size. Each individual is uniformly randomly initialized $\phi_i \sim \mathcal{U}(0, 1)$ (all synthesizer parameters are normalized). We also include the anchor preset ϕ_α in the initial population. Each individual is then evaluated by first synthesizing the new preset and computing the synthesized feature difference $\Delta\hat{\mathbf{y}}$ using Equation (2). The *fitness* is then calculated using a fitness function $\mathcal{L}$, defined as the mean absolute error (MAE) between the target feature difference $\Delta\mathbf{x}$ and the synthesized feature difference $\Delta\hat{\mathbf{y}}$:

$$\mathcal{L}(\Delta\mathbf{x}, \Delta\hat{\mathbf{y}}) = \frac{1}{k}\sum_{i=1}^{k} |\Delta x_i - \Delta\hat{y}_i| \tag{3}$$

where k is the dimensionality of the feature space. A fitness value is computed for each individual in the population.

Following this, each individual is evolved using a set of genetic operators, which are inspired by biological processes including genetic mutation and crossover. These operators are applied to the current population to generate new *offspring*. Mutation applies a random modulation to each parameter during each iteration to discover variations of current solutions, whereas crossover randomly swaps genes (i.e., individual parameter values) between two individuals to generate a new offspring. We found that using a single operator, polynomial mutation [14], provided the best results in terms of discovering timbre analogies. Synthesis parameters are in a normalized range $[0, 1]$, leading to a slightly simplified version of polynomial mutation, which is as follows:

$$\delta(x_p, r) = \begin{cases} \delta_l(x_p, r) & r \leq 0.5, \\ \delta_h(x_p, r) & \text{otherwise.} \end{cases} \tag{4}$$

$$\delta_l(x_p, r) = \left[\, 2r + (1-2r)\,(1-x_p)^{\eta+1}\right]^{\frac{1}{\eta+1}} - 1 \tag{5}$$

$$\delta_h(x_p, r) = 1 - \left[2(1-r) + 2(r-0.5)(x_p)^{\eta+1}\right]^{\frac{1}{\eta+1}} \tag{6}$$

The final mutated value is then computed as follows:

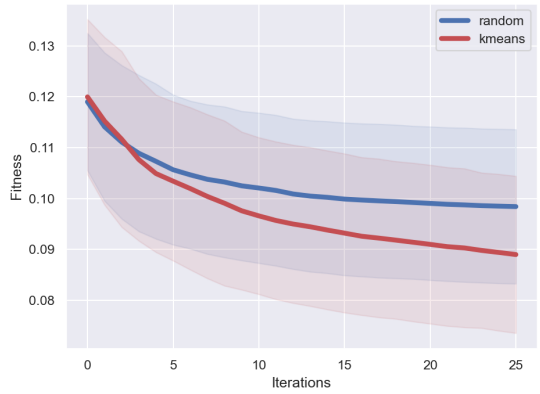
$$x_o = x_p + \delta(x_p, r) \tag{7}$$

where x_p is the current (parent) parameter value, x_o is the mutated (offspring) parameter value, r is a random number $r \sim \mathcal{U}(0, 1)$, and η is the polynomial mutation index, which determines the variance of the mutation. This process is repeated for a set number of iterations.

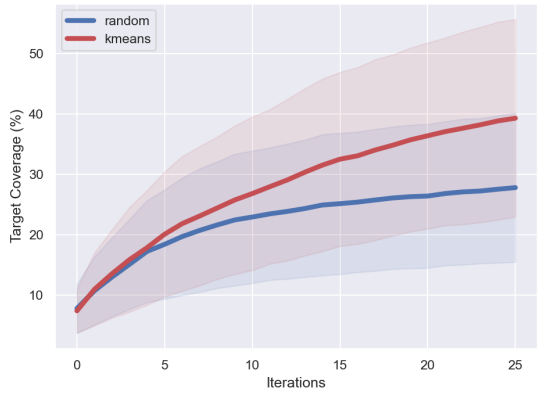
4.2.2 Single-Search Multi-Target Corpus Generation. Running this process on each target from the input dataset is time consuming as it requires M separate searches. Based on the fact that our synthesized sounds should be variations of a preset, we hypothesized that it may be possible to generate solutions for *all* targets in a single run using variations within an evolving population.

The main problem we faced in single search corpus generation was how to select individuals for the next population. Elitism is a common tactic to maintain the best performing individual in a population without modification—but this breaks down when the number of targets is greater than the population size, i.e., $M > N$. To ensure adequate coverage of our target dataset, we devised a method for selecting the next population using k-Means clustering [33]. The target dataset of size M is partitioned into N (population size) clusters using k-Means and representative samples are selected as those closest to the cluster centroids using Euclidean distance. During a search, the fitness of each candidate is evaluated against these representative samples, and the new candidate is kept if it improves upon the previous best solution for that representative sample. In parallel, the best solutions for all targets in the input dataset $\Delta\mathbf{X}$ are stored and updated during the search. We also maintain an archive of all candidates evaluated during a search, which provides a much larger and diverse corpus of synthesizer presets for nearest neighbour mapping (described in Section 4.4), while still containing solutions based on the targeted search.

4.2.3 Evaluation. Our evaluation goal was to validate our approach within the constraints of our practice rather than to provide a comprehensive evaluation in relation to existing methods. We used two metrics to assess the GA: average fitness value and *target coverage* (the percentage of targets with unique solutions). These reflect our aim to generate distinct, high-quality solutions for each target. To validate our k-Means selection approach we compared with an alternative method inspired by MAP-Elites [38], which selects individuals for the next population through randomly sampling from the set of optimal solutions assigned to each target. We ran multiple GA runs with the 808 snare synthesizer using a dataset of 267 drum hits. Each run lasted for 25 iterations with a population size of 100 (~ 21 minutes). The k-Means selection approach on average outperformed the random selection method, as shown in Figure 3. These results show that on average we are able to find unique solutions for about 40% of the target dataset, although with some variance.



(a) Fitness Value (smaller values are better)



(b) Target Coverage (larger values are better)

Figure 3: Fitness and target coverage across genetic search iterations, comparing our k-Means selection to random target-based selection. Bold lines show averages; shaded areas indicate standard deviation.

4.3 Timescale Regression

Following GA corpus generation, we have a dataset of synthesizer presets paired with their feature differences. However, feature extraction during the GA search may be computed using windows of 100ms or 250ms, complicating the process of querying presets in real-time with low-latency from our input audio. To overcome this issue we train neural networks to predict audio features computed over longer windows from short windows of 256 samples. It is worth noting that the dimensionality of the features remains consistent across different window sizes due to temporal summarization statistics.

Timescale regressors are trained once per instrument, for each set of audio features, and can be reused across different synthesizers and mappings. Multi-layer Perceptrons (MLP) networks are trained using a hyperparameter search [3][10]. Input and output data is normalized prior to training. A validation set of 20% is withheld from the training dataset and training is halted if the validation loss doesn't improve for 20 epochs. The best resulting neural network with respect to validation loss during hyperparameter search is saved for inference within FluCoMa.

[10]For details on the hyperparameter search please refer to our supplemental website. We release a Python tool alongside this paper to support FluCoMa practitioners with neural network training.

Table 3: Timescale regressor errors (MAES) for predicting 100ms and 250ms features from 256-sample inputs. Baseline shows error without regression.

	100ms		250ms	
Features	Baseline	Predict	Baseline	Predict
Hybrid Descriptors	0.48	0.12	0.56	0.15
Mel-Bands	0.20	0.08	0.26	0.12
MFCCs	0.38	0.09	0.56	0.10
Spectral Shape	0.36	0.10	0.40	0.12

4.3.1 Evaluation. To evaluate timescale regressors we trained regressors for all features mapping from onset features to features computed with 100ms or 250ms windows. To verify that timescale regressors offer a benefit, we benchmark against using the onset features themselves in place of the longer term features, that is, we compute the error between the onset features and the longer windows. Timescale regressors for each feature were trained using a hyperparameter search, and the best resulting network based on validation loss was selected for evaluation. Training was conducted on a dataset of 1233 drum hits, and evaluated on a separate evaluation dataset of 650 drum hits recorded on the same drum. Error was computed using the mean absolute error on normalized feature values, averaged across all hits in the evaluation datasets. Results are show in Table 3, and highlight that timescale regressors offer clear improvements.

In addition to numerically validating these regressors, we also performed qualitative evaluation in practice. We found it useful to visualize the outputs produced by the timescale regressors on real-time plots to visually confirm that predicted features follow contours that matched our perceptual expectations.

4.4 Real-time Timbre Remapping

We now have a corpus of synthesizer parameters, paired with audio feature difference vectors, and a method to compute longer term audio features in real-time with low-latency from onset features. The process of real-time timbre remapping is visually outlined in Figure 2 and is performed as follows: 1) an onset is detected from an input stream; 2) audio features are computed on a window of 256 samples; 3) longer term features, the same used during the GA search, are predicted using a timescale regressor; 4) the feature difference with respect to the anchor drum $\mathbf{x}_\alpha$ is computed; 5) synthesizer presets are queried from the generated corpus and synthesized; 6) (optionally) loudness compensation is applied (described below). We used a nearest neighbour search with the k-d tree algorithm [2] in FluCoMa. Training neural networks to predict synthesizer parameters from features is also an option; however, we found the nearest neighbour search to be sufficient for prototyping, supporting faster design iterations.

4.4.1 Loudness Compensation. In parallel with timbre remapping, we built in functionality to adjust the loudness of the synthesized output to map the input. We use loudness, k-weighted, relative to full-scale (LKFS) [53], implemented in FluCoMa. Loudness for synthesizer presets are stored alongside the feature difference vectors during evolutionary search. The difference between the loudness detected during the onset frame of an input and the loudness of the synthesizer preset is used to adjust the amplitude of the synthesizer in real-time.

5 Experiments

We present a set of technical and practice-based experiments aimed at investigating the nature of audio representations, based on our inquiry outlined in Section 3.1.2, and to explore the musical affordances of our timbre remapping system.

5.1 Data-Driven Experiments

In work focused on timbre remapping audio features extracted from vocal signals Stowell [48] defined a selection criteria for audio features: robustness, relevancy, and independence. We focused on robustness in these data-driven experiments and investigate relevancy through practice. Independence between features is a useful quality for selecting a minimal set of features for classification-based machine learning tasks; however, in the case of timbre remapping it is not immediately clear that independence is desirable. Audio features that are highly correlated on a particular acoustic instrument may not necessarily be correlated on a synthesizer – encoding this relationship across multiple features may be important for successful timbre remapping.

Robustness measures the stability of audio features across input gestures that are deemed to be similar. We evaluated our representations on both acoustic drum signals and synthesizer sounds using a method outlined by Fasciani et al. [18, 19]. The metric is the relative mean difference (RMD), which measures the statistical dispersion of a signal. RMD is defined as:

$$RMD = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} \left| x_i - x_j \right|}{\left| \sum_{i=1}^{n} x_i \right| (n-1)} \tag{8}$$

where n is the number of samples and x is the individual feature being investigated. For drums, we recorded 32 hits with consistent loudness and strike location with 10 different snare drum hit locations and 6 different crotale hit locations. We also recorded hits with variable loudness and consistent strike location to query the loudness sensitivity of features. We evaluated different timescales and tested the three acoustic sensor combinations. For each synth, we generated 100 random presets and recorded each preset 100 times with the same note and velocity (if the synth accepted MIDI). Fasciani et al. select individual audio features if the RMD value is below a threshold. RMD is scale-invariant, so all RMD values are in a normalized range. We follow Fasciani et al. in using a threshold of 0.5.

Overall, results were remarkably similar across the drum and synthesizer sounds. The main takeaways from this analysis is that derivative features included in the hybrid descriptors and spectral shape features are not robust, even for repeated synthesizer sounds. The non-derivative features for spectral shape on the other hand were robust for both loudness consistent and loudness variable datasets, confirming the loudness insensitivity of these features. Looking at results for different microphone combinations, the combination of the drum sensor and DPA performed the best, with the DPA by itself also performing well.

Based on these observations, we decided to remove the derivative features from the spectral shape extractor. We kept derivatives in the hybrid descriptor extractor as a counter example to investigate whether morphology encoded by these features would be useful for timbre remapping, despite noisiness.

5.2 Practice-Based Experiments

Practice-based experiments were conducted throughout the design process. We conducted listening-based reflections using pre-recorded musical phrases with different variations and performed with our mappings in-studio. Reactions and ideas that emerged through dialogue were captured via audio recording and periods of journalling. Reflection on this data surfaced several key themes, which are summarized here.

5.2.1 Defining Success. Ultimately we evaluated the success of a mapping based on aesthetic judgements and broad experience during listening and playing; however, several concrete elements emerged: 1) **Consistency** (but not too consistent): noisiness in mappings was undesirable, however too little variance was also bad; 2) **Repeatability**: we were able to reproduce sounds with intention; 3) **Perceptual Alignment**: timbral contours matched our expectations; 4) **Plausible**; timbral analogies were being solved through reasonable synthesizer parameterizations.

5.2.2 Inter-Modulations vs Intra-Modulations. We distinguished between inter-modulations (e.g., hitting the drum vs. hitting the rim) and intra-instrument modulations (e.g., hitting different locations on the same drum head). Certain features captured inter-modulations well but produced overly varied intra-modulations (hybrid descriptors and MFCCs) and vice versa for others (Mel-Bands). Overall we found the spectral shape feature to produce the most balanced mappings.

5.2.3 Direction of Timbre Trajectories. We noticed that for certain features, particularly with the MFCCs, timbre analogies were mapped in reverse to our expectations. For example, rim-clicks sometimes mapped to darker sounding presets instead of brighter ones. This directional misalignment highlights both our sensitivity to the directionality of timbral changes and to a potential limitation of MFCCs to represent these timbral translations.

5.2.4 Synthesizer and Performance Dependencies. Each synthesizer exhibited distinct timbre remapping behaviours. The 808 snare often produced unexpected results due to pitch volatility and excessive noise whereas the chaotic Quantussy synthesizer responded well to timbral remappings, likely because its parameters are less directly tied to loudness and pitch. Finally, the performance context mattered: while the physical modelling synthesizer Derailer produced sonically smeared results in pre-recorded material, live performance allowed natural adaptations to its long resonances, result in a rich performance experience.

5.2.5 Importance of Loudness Mapping. Applying loudness compensation to the synthesized results was a simple extension that we found broadly beneficial. Although some feature extractors (i.e., hybrid descriptor and Mel-bands) encode loudness, we generally found that tying loudness to the input was favourable.

5.2.6 Lack of Morphology. In general, we found that morphological aspects of our input sound, both timbral and amplitude-based, were poorly mapped onto parameters associated with time-varying characteristics. Morphology was encoded to a certain extent through derivative features, however our GA search was unable to leverage this information effectively.

5.3 Exploration of Musical Affordances

We implemented several mapping variations that explored timbre remapping within a performance context during our collaborative in-studio sessions. These patches either sought to mitigate a perceived short-coming of timbre remapping, or explored creative extensions of it. These patches, with reflections, are shared on

our supplementary website in the form of an annotated portfolio with videos: https://jordieshier.com/projects/nime2025/.

6 Discussion

6.1 Evolving Parameter Spaces

The genetic algorithm provided a method to explore diverse synthesizer spaces with respect to audio features. This method supported rapid prototyping of mappings with alternative audio representations over multiple non-differentiable synthesizers, including hardware synthesizers like Eurorack modulars. Being agnostic to the target synthesizer enabled swapping synthesizers with minor configuration—we developed a Max wrapper for VSTs and hardware synthesizers to support this. However, not all synthesizers are built the same, and the internal mapping of parameters to sound generators had a strong bearing on results. Some of our best results were achieved through thoughtful tuning of parameter ranges, which we did for Quantussy, allowing for more explicit focus on timbre variations with less-sensitivity to pitch and loudness.

Speed is a factor with evolutionary algorithms. Most of our GA runs lasted for about 20 minutes, which involved the evaluation of 2500 different synthesizer parameters using windows of 250ms for audio feature extraction. The speed of the GA and reliance on real-time synthesis (as opposed to offline accelerated synthesis) is an obvious downside of this approach. Offline and parallelized synthesis with VSTs is supported using Python VST wrappers like DawDreamer [4], although this complicates the process of quickly generating playable mappings in a creative coding environment like Max/MSP.

Although evolutionary algorithms are by no means a novel approach for synthesizer parameter search, we think that a deeper exploration of how they can be integrated with other machine learning approaches is an exciting direction, either for the generation of datasets, like we explored here, or by combining learned representations to steer evolution [47].

6.2 Negotiating Audio Representations

One of the main goals of our work was to develop a deeper understanding of timbre remapping with audio features. Comparing data-driven insights with practice-based reflections provided us with a method to better understand the impact of feature selection for timbre remapping in our musical context. Maximizing robustness with the spectral shape was effective and generally resulted in mappings that responded in a balanced way to a wide range of timbral variations. Extending this work to explore the relationship between audio features and synthesizer parameters would be particularly valuable—for example, by systematically identifying features that both characterize acoustic percussion instruments *and* capture meaningful variations on the target synthesizer. Techniques introduced by Fasciani [19] could likely be adapted to support this investigation.

Practice-based interventions upon our timbre remappings, particularly when they were aimed at overcoming shortcomings, provided valuable insights into what worked well and what didn't. For instance, morphological control was a particularly salient shortcoming and we found ourselves trying to address it by directly mapping audio features from the input to controls for envelopes to effect more dynamic control over amplitude shape. A patch involving multiple mapping systems operating on different timescales highlighted the potential to integrate information from distinct temporal segments to enable fine-grained control of a sound over time. In general, exploration of richer representations of sonic morphology—either through exploration of pre-processing techniques for derivative features or alternative techniques —and how those map to synthesis parameters could be a valuable line of future work. One theme that emerged during our more experimental patching sessions was that of "queryable" features versus "transferable" features. That is, the question of what features should be applied to the process of timbre mapping (queryable) and what features should be directly mapping (transferable) to the output. Loudness compensation is an example of a feature that worked well for transferring. Others, such a timbral transfer or pitch transfer may also be interesting.

7 Conclusions

Timbre remapping is a difficult task. The designer must grapple with the many possible meanings of the word "timbre", the multitude of numerical methods for defining or measuring it, the necessary temporal windows and resulting latency needed to perform those measurements, and the always-imperfect alignment between audio features and human perception. The idea that a technical specification could encompass all of these factors sufficiently thoroughly to make timbre remapping into a purely scientific problem is unlikely; as Jordà writes about digital instrument design more generally, timbre remapping proceeds "as a sort of craftsmanship, that may sometimes produce – in very exceptional cases – a work of art; no less than music" [30].

This paper has presented a combination of technical and practice-based contributions toward timbre remapping from acoustic to synthetic percussion. On the technical side, the paper demonstrates how genetic algorithms can address some persistent challenges of discovering relationships between synthesiser parameter spaces and audio features, and it also proposes a way to sidestep some of the latency issues of real-time audio feature analysis by using a neural network to predict features from a substantially shortened audio window. On the practice-based side, we show how exploration and iteration led to discoveries and opportunities that a pure informatics-based approach might have missed, and we catalogue some of these discoveries in an annotated portfolio.

8 Ethical Standards

This paper describes a practice-based exploration between the authors and did not involve experiments with other human participants, hence no institutional ethics board review was required. This work incorporates machine learning techniques, where all data used in training is generated by the authors themselves.

Acknowledgments

This research is supported by the UKRI Centre for Doctoral Training in Artificial Intelligence and Music (EP/S022694/1), a UKRI Frontier Research (Consolidator) Grant (EP/X023478/1, "RUDIMENTS"), and by the Royal Academy of Engineering under the Research Chairs and Senior Research Fellowships scheme. Thank you to Physical Audio for supporting this research and contributing research licenses for their audio plugins. Thank you to the all the reviewers for their thoughtful feedback, which helped improve the quality of this paper.

References

[1] D. Arfib, J. M. Couturier, L. Kessous, and V. Verfaille. 2002. Strategies of Mapping Between Gesture Data and Synthesis Model Parameters Using Perceptual Spaces. *Organised Sound* 7, 2 (2002), 127–144. https://doi.org/10.1017/S1355771802002054

[2] Jon Louis Bentley. 1975. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM* 18, 9 (1975), 509–517. https://doi.org/10.1145/361002.361007

[3] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems 24*.

[4] David Braun. 2021. DawDreamer: Bridging the Gap Between Digital Audio Workstations and Python Interfaces. In *Extended Abstracts for the Late-Breaking Demo Session of the 22nd Int. Society for Music Information Retrieval Conf.*

[5] Fred Bruford, Frederik Blang, and Shahan Nercessian. 2024. Synthesizer Sound Matching Using Audio Spectrogram Transformers. In *Proceedings of the 27th International Conference on Digital Audio Effects (DAFx24)*.

[6] Anne Caclin, Stephen McAdams, Bennett K. Smith, and Suzanne Winsberg. 2005. Acoustic correlates of timbre space dimensions: A confirmatory study using synthetic tones. *The Journal of the Acoustical Society of America* 118, 1 (2005), 471–482. https://doi.org/10.1121/1.1929229

[7] Mark Cartwright and Bryan Pardo. 2014. SynthAssist: Querying an Audio Synthesizer by Vocal Imitation. In *Proceedings of the International Conference on New Interfaces for Musical Expression*.

[8] Ondřej Cífka, Alexey Ozerov, Umut Şimşekli, and Gaël Richard. 2021. Self-Supervised VQ-VAE for One-Shot Music Style Transfer. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing*. 96–100. https://doi.org/10.1109/ICASSP39728.2021.9414235

[9] Luke Dahl. 2012. Wicked Problems and Design Considerations in Composing for Laptop Orchestra. In *Proceedings of the International Conference on New Interfaces for Musical Expression*.

[10] Luke Dahl. 2016. Designing New Musical Interfaces as Research: What's the Problem? *Leonardo* 49, 1 (2016), 76–77. https://www.jstor.org/stable/43834328

[11] Palle Dahlstedt. 2001. Creating and Exploring Huge Parameter Spaces: Interactive Evolution as a Tool for Sound Generation. In *International Computer Music Conference*.

[12] Palle Dahlstedt. 2004. *Sounds Unheard of Evolutionary Algorithms as Creative Tools for the Contemporary Composer*. Ph. D. Dissertation. Chalmers University of Technology.

[13] Anne Danielsen, Carl Haakon Waadeland, Henrik G. Sundt, and Maria A. G. Witek. 2015. Effects of instructed timing and tempo on snare drum sound in drum kit performance. *The Journal of the Acoustical Society of America* 138, 4 (2015), 2301–2316. https://doi.org/10.1121/1.4930950

[14] Kalyanmoy Deb and Santosh Tiwari. 2008. Omni-optimizer: A generic evolutionary algorithm for single and multi-objective optimization. *European Journal of Operational Research* 185, 3 (March 2008), 1062–1087. https://doi.org/10.1016/j.ejor.2006.06.042

[15] D Ehresman and David L. Wessel. 1978. *Perception of Timbral Analogies*. Technical Report 13. IRCAM.

[16] Jesse Engel, Lamtharn (Hanoi) Hantrakul, Chenjie Gu, and Adam Roberts. 2020. DDSP: Differentiable Digital Signal Processing. In *8th International Conference on Learning Representations*.

[17] Philippe Esling, Naotake Masuda, Adrien Bardet, Romeo Despres, and Axel Chemla-Romeu-Santos. 2020. Flow Synthesizer: Universal Audio Synthesizer Control with Normalizing Flows. *Applied Sciences* 10, 1 (2020), 302. https://doi.org/10.3390/app10010302

[18] Stefano Fasciani. 2012. Voice Features for Control: A Vocalist Dependent Method for Noise Measurement and Independent Signals Computation. In *Proceedings of the 15th Int. Conference on Digital Audio Effects (DAFx-12)*.

[19] Stefano Fasciani. 2016. TSAM: a tool for analyzing, modeling, and mapping the timbre of sound synthesizers. In *Proceedings of 13th Sound and Music Computing Conference*.

[20] Stefano Fasciani and Lonce Wyse. 2018. Vocal Control of Sound Synthesis Personalized by Unsupervised Machine Listening and Learning. *Computer Music Journal* 42, 1 (2018), 37–59. https://doi.org/10.1162/comj_a_00450

[21] Bill Gaver and John Bowers. 2012. Annotated Portfolios. *Interactions* 19, 4 (2012), 40–49. https://doi.org/10.1145/2212877.2212889

[22] Owen Green, Pierre Alexandre Tremblay, and Gerard Roma. 2018. Interdisciplinary Research as Musical Experimentation: A case study in musicianly approaches to sound corpora. In *Proceedings of the Electroacoustic Music Studies Network Conference*.

[23] John M. Grey. 1977. Multidimensional perceptual scaling of musical timbres. *The Journal of the Acoustical Society of America* 61, 5 (1977), 1270–1277. https://doi.org/10.1121/1.381428

[24] John M. Grey and John W. Gordon. 1978. Perceptual effects of spectral modifications on musical timbres. *The Journal of the Acoustical Society of America* 63, 5 (1978), 1493–1500. https://doi.org/10.1121/1.381843

[25] Perfecto Herrera, Amaury Dehamel, and Fabien Gouyon. 2003. Automatic Labeling of Unpitched Percussion Sounds. In *Audio Engineering Society Convention 114*.

[26] Andrew Horner, James Beauchamp, and Lippold Haken. 1993. Machine Tongues XVI. Genetic Algorithms and Their Application to FM Matching Synthesis. *Computer Music Journal* 17, 4 (1993), 17–29. https://doi.org/10.2307/3680541

[27] Andy Hunt and Ross Kirk. 2000. Mapping Strategies for Musical Performance. In *Trends in Gestural Control of Music*. IRCAM, 231–235.

[28] Andy Hunt, Marcelo M. Wanderley, and Matthew Paradis. 2003. The Importance of Parameter Mapping in Electronic Instrument Design. *Journal of New Music Research* 32, 4 (2003), 429–440. https://doi.org/10.1076/jnmr.32.4.429.18853

[29] Andrew Johnston. 2011. Beyond Evaluation: Linking Practice and Theory in New Musical Interface Design. In *Proceedings of the International Conference on New Interfaces for Musical Expression*.

[30] Sergi Jordà. 2005. *Digital Lutherie Crafting musical computers for new musics' performance and improvisation*. Ph. D. Dissertation. Universitat Pompeu Fabra.

[31] Carol L Krumhansl. 1989. Why Is Musical Timbre So Hard to Understand?. In *Structure and Perception of Electroacoustic Sound and Music*. 43–55.

[32] Gwendal Le Vaillant, Thierry Dutoit, and Sébastien Dekeyser. 2021. Improving Synthesizer Programming From Variational Autoencoders Latent Space. In *Proceedings of the 24th International Conference on Digital Audio Effects (DAFx20in21)*. 276–283. https://doi.org/10.23919/DAFx51585.2021.9768218

[33] Stuart Lloyd. 1982. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory* 28, 2 (1982), 129–137. https://doi.org/10.1109/TIT.1982.1056489

[34] Naotake Masuda and Daisuke Saito. 2021. Quality Diversity for Synthesizer Sound Matching. In *Proceedings of the 24th International Conference on Digital Audio Effects (DAFx20in21)*. 300–307. https://doi.org/10.23919/DAFx51585.2021.9768271

[35] Stephen McAdams. 2019. Timbre as a Structuring Force in Music. In *Timbre: Acoustics, Perception, and Cognition*. Springer Handbook of Auditory Research, Vol. 69. 211–243.

[36] Stephen McAdams and Albert Bregman. 1979. Hearing Musical Streams. *Computer Music Journal* 3, 4 (1979), 26–60. https://www.jstor.org/stable/4617866

[37] Stephen McAdams and Jean-Christophe Cunible. 1992. Perception of Timbral Analogies. *Philosophical Transactions: Biological Sciences* 336, 1278 (1992), 383–389. http://www.jstor.org/stable/55908

[38] Jean-Baptiste Mouret and Jeff Clune. 2015. Illuminating Search Spaces by Mapping Elites. (2015). https://doi.org/10.48550/arXiv.1504.04909 [Preprint].

[39] Geoffroy Peeters. 2004. *A Large Set of Audio Features for Sound Description (Similarity and Classification) in the CUIDADO Project*. Technical Report. IRCAM.

[40] Geoffroy Peeters, Bruno L. Giordano, Patrick Susini, Nicolas Misdariis, and Stephen McAdams. 2011. The Timbre Toolbox: Extracting audio descriptors from musical signals. *The Journal of the Acoustical Society of America* 130, 5 (2011), 2902–2916. https://doi.org/10.1121/1.3642604

[41] Teresa Pelinski, Andrew McPherson, and Rebecca Fiebrink. 2025. Ways of knowing, ways of writing: technical practice research in new musical instrument design. *Journal of New Music Research* (2025), 1–14. https://doi.org/10.1080/09298215.2024.2442348

[42] Charalampos Saitis and Zachary Wallmark. 2024. Timbral brightness perception investigated through multimodal interference. *Attention, Perception, & Psychophysics* 86, 6 (2024), 1835–1845. https://doi.org/10.3758/s13414-024-02934-2

[43] Diemo Schwarz, Grégory Beller, Bruno Verbrugghe, and Sam Britton. 2006. Real-Time Corpus-Based Concatenative Synthesis with CataRT. In *Proceedings of the 9th International Conference on Digital Audio Effects (DAFx-06)*.

[44] Jordie Shier. 2021. *The Synthesizer Programming Problem: Improving the Usability of Sound Synthesizers*. Master's thesis. University of Victoria.

[45] Jordie Shier, Charalampos Saitis, Andrew Robertson, and Andrew McPherson. 2024. Real-time Timbre Remapping with Differentiable DSP. In *Proceedings of the International Conference on New Interfaces for Musical Expression*.

[46] Zefan Sramek, Arissa J. Sato, Zhongyi Zhou, Simo Hosio, and Koji Yatani. 2023. Soundtraveller: Exploring Abstraction and Entanglement in Timbre Creation Interfaces for Synthesizers. In *Proceedings of the 2023 ACM Designing Interactive Systems Conference*. 95–114. https://doi.org/10.1145/3563657.3596089

[47] Christian J. Steinmetz, Shubhr Singh, Marco Comunità, Ilias Ibnyahya, Shanxin Yuan, Emmanouil Benetos, and Joshua D. Reiss. 2024. ST-ITO: Controlling Audio Effects for Style Transfer with Inference-Time Optimization. In *Proceedings of the 25th International Society for Music Information Retrieval Conference*.

[48] Dan Stowell. 2010. *Making music through real-time voice timbre analysis: machine learning and timbral control*. PhD Thesis. Queen Mary University of London.

[49] Dan Stowell and Mark D Plumbley. 2010. Timbre remapping through a regression-tree technique. In *Proceedings of the 7th Sound and Music Computing Conference*.

[50] Kivanç Tatar, Matthieu Macret, and Philippe Pasquier. 2016. Automatic Synthesizer Preset Generation with PresetGen. *Journal of New Music Research* 45, 2 (2016), 124–144. https://doi.org/10.1080/09298215.2016.1175481

[51] Adam Tindale, Ajay Kapur, George Tzanetakis, and Ichiro Fujinaga. 2004. Retrieval of Percussion Gestures Using Timbre Classification Techniques. In *Proceedings of the 5th International Conference on Music Information Retrieval*.

[52] Pierre Alexandre Tremblay and Diemo Schwarz. 2010. Surfing the Waves: Live Audio Mosaicing of an Electric Bass Performance as a Corpus Browsing Interface. In *Proceedings of the International Conference on New Interfaces for Musical Expression*.

[53] International Telecommunication Union. 2006. Algorithms to Measure Audio Programme Loudness and True-Peak Audio Level. *ITU-R BS.1770* (2006).
[54] David Wessel, David Bristow, and Zack Settel. 1987. Control of Phrasing and Articulation in Synthesis. In *Proceedings of the International Computer Music Conference*.
[55] David L. Wessel. 1979. Timbre Space as a Musical Control Structure. *Computer Music Journal* 3, 2 (1979), 45. https://doi.org/10.2307/3680283
[56] Matthew John Yee-King, Leon Fedden, and Mark d'Inverno. 2018. Automatic Programming of VST Sound Synthesizers Using Deep Networks and Other Techniques. *IEEE Transactions on Emerging Topics in Computational Intelligence* 2, 2 (2018), 150–159. https://doi.org/10.1109/TETCI.2017.2783885
[57] Eoghan Ó Néill and Miguel Ortiz. 2024. From Prototype to Performance Practice: Reflections on Iterative Instrument Design. In *Proceedings of the 19th International Audio Mostly Conference: Explorations in Sonic Cultures*. 439–444. https://doi.org/10.1145/3678299.3678360