Digging it: Programmatic Data Mining as Musicking

Pierre Alexandre Tremblay **Gerard Roma** **Owen Green**
CeReNeM – University of Huddersfield
{ p.a.tremblay, g.roma, o.green } @ hud.ac.uk

ABSTRACT

This paper presents a new software toolbox to enable programmatic sound bank mining for musicking and musicking-driven research. The toolbox is available for popular creative coding environments currently used by techno-fluent musicians. The paper describes the design rationale and functionality, then the methodology: several versions of the toolbox have been seeded to early adopters who have, in turn, contributed to the design. Based on this feedback, we describe some observed musical affordances of the proposed approach to music corpus exploration and manipulation, as well as the main roadblocks. We finally reflect on a few emerging themes for the next steps in programmatic sound bank mining.

1. INTRODUCTION

Developments in general computing continue to be inspired by the possibilities of exploiting large amounts of data. For musicking [1], taken as music making in its widest, most diverse and inclusive sense, new data-driven paradigms are promising but still challenging due to the diversity and size of sound collections amassed by musicians, as well as the context-dependant nature of questions that determine the paths of musical creativity.

Notably, whilst there has been much research around the potential of isolated technologies or techniques in a musical context, there has been much less detailing or comparing musical strategies and tactics arising from sustained practice. The Fluid Corpus Manipulation project (FluCoMa) aims to help bridge this gap by making available cross-platform technologies in tandem with supporting resources and community, all focused on discovering and developing complete creative workflows for audio corpora with data-driven techniques.

In previous work [2] we introduced tools for decomposing and analysing sounds to facilitate creating audio corpora and extracting audio descriptors from within the most popular musical creative coding environments (CCEs) such as Max[1], PureData[2] (Pd) and SuperCollider[3] (SC). This leaves open the challenge of enabling musicking and musicking-driven research using the resulting corpora and data.

In this paper we present the second iteration of our toolbox, where we focus on exploring, interacting with, and manipulating audio corpora with a framework of tools for organising and learning from data. Our aim is to provide this combined functionality in a single package for each CCE with enough commonality in syntax and concepts to enable discussion and exchange between a range of musicians, in the interests of supporting a broad community for future research.

In the next section we review existing tools for music creation based on audio corpora and note some limitations with respect to the musical workflows they afford. Section 3 then describes the aims and content of our toolbox. In Section 4 we describe examples of workflows enabled by the toolbox and, in Section 5, offer some insights into our design process and the initial work of a community of early adopters. As described in [3], our methodology is rooted in a pluralist and cross-disciplinary view of techno-fluent musicking. Here, we focus on some emerging themes around knowledge transfer, interface granularity and points of entanglement.

2. RELATED WORK

There is an extensive body of prior work that makes available some of the tools and techniques in which we are interested. Although none of this work aligns precisely with our focus on developing a framework aimed squarely at flexible music making with audio corpora, it has been a rich source of learning and motivation. Our design has also been influenced by the wealth of libraries and learning materials in data science communities, notably around the Python language and the scikit-learn library [4] and their reflections on interface [5].

One category of existing work focuses on specific forms of musical interaction. AudioGuide [6] and Orchidea [7] are geared towards computer assisted orchestration. Meanwhile, CataRT [8] and the more recent AudioStellar [9] offer an intuitive interface for corpus exploration through snippets arranged in a 2D space. In these cases, the very specific focus means that the system presents as more of a black box than we aim for in the current work, insofar as certain decisions about analysis, presentation and general interface are already taken and fixed. This limits their creative coding affordances: it makes exploration of interaction, customisation and integration within the CCE difficult.

A second category is work that makes available particular algorithms or tools in isolation, but not as part of a framework that also supports gathering, exploring and refining data as part of an overall musical project. Here we could include the Wekinator application [10], the Max packages ml.lib [11] and ml.star [12], and certain SC

[1] https://cycling74.com/products/max
[2] http://msp.ucsd.edu/software.html
[3] https://github.com/supercollider/supercollider

extensions like KDTree and KMeans. Most of these solutions are focused more on gestural data streams, and do not offer facilities for exploring larger corpora, nor dynamic creative coding between various complementary algorithms. Conversely, toolsets that enable handling and exploring data within CCEs have been proposed, but without a focus on audio. For instance, the Bach [13] family of extensions for Max has recently been enriched with a suite of objects [14] that provides alternatives to Bach's core focus on Western common notation.

Finally, two prior contributions align most closely with the proposed work, and warrant more extensive discussion: the SCMIR package for SC [15], and the MuBu & Friends package for Max [16].

SCMIR provides a complete pipeline from audio feature extraction to a range of algorithms and representations, along with some supporting machinery for conditioning data. The lack of a native plugin API for sclang[4] is worked around by using external programs as 'pseudo-plugins'. Whilst the initial focus for SCMIR was analysis, it also enables creative work with analysis data and algorithms. Our toolbox similarly enables complete pipelines, but all the computation is implemented in server-side SC plug-ins. This provides a more natural and granular integration with the SC environment.

MuBu & Friends likewise offers a framework with pipelines from audio feature extraction (via PiPo [17]), through to a suite of algorithms for exploring and learning from data. Its focus is more general than our proposed toolbox, insofar as sensor data is a more explicit concern, whilst less attention is given to the framework's place in an overall creative workflow. At the heart of framework is the MuBu (multi-buffer) object itself, which extends the familiar paradigm of the audio buffer with labels, different datatypes, time tags and other useful facilities. However, our experience has been that this richness can be difficult to grasp, especially for newcomers, and that inter-operation with the wider Max environment is not always simple. Our approach is different in that we aim to provide more open and modular interfaces, consistent across all the supported CCEs.

3. THE TOOLKIT

The tools added in the new iteration of the Fluid Corpus Manipulation toolbox [2] are well established in data science and most have been previously used in audio signal processing research. Our aim is to enable programmatic interaction with sound collections and other signal corpora within CCEs to enable future research around sustained musical practice with these sorts of tools and concepts. In the toolbox's previous iteration, we pursued this aim with a focus on musical approaches to signal decomposition and description[5]. The new additions comprise a suite of 33 objects available for each of the most popular CCEs (Max and SC; PD and command-line interface are under development), based on a common C++ architecture. All code is open source, under a BSD-3 license. In this section, we discuss the design for this new suite of tools, and then describe its main functionality[6].

3.1 Aims and Priorities

The priorities for the toolkit stem from wishing it to be useful as a general framework in its own right as well as from broader goals of the FluCoMa project overall.

The following aims and priorities inform our design decisions and trade-offs, and draw on our experiences with the existing work in this area [18], lessons from developing the first iteration of our toolbox, and continuous feedback from a pool of musicians working with the tools from early on:

- **Native integration:** Our tools should respect as closely as possible established idioms in the host CCE and allow easy transfer of data between the framework and native data structures.
- **Consistency:** Objects should fit together neatly and offer a consistent way of working. We have opted for an analogue of the scikit learn [4] naming conventions for our algorithms.
- **Learnability:** The overall framework and the granularity of its objects should afford early, easy exploration yet offer rich scope for deeper experimentation.
- **Configurability:** Objects should offer fine adjustment and tweaking where the algorithm supports it.
- **Scalability:** Algorithms should be able to cope with reasonably large collections of data within the confines of CPU-based personal computing in currently available hardware.
- **Breadth:** An initial offering of well-known and understood algorithms, touching on a breadth of possible approaches helps us draw on existing knowledge and practice, and assess the kinds of extension that might be of interest to the community.
- **Completeness:** We aim to enable complete workflows from corpus-building, curation, learning, to sound making.

Additionally, because of the project's overarching aim to catalyse more and better musicking research in this area, close attention has been paid to interface consistency across different CCEs, to enable researchers working in different environments to exchange ideas and insights. Variations on similar ideas can be explored iteratively within the distinct idiomatic contexts that different CCEs afford [19].

Finally (and perhaps most elusively), is a commitment to providing tools that are 'artist spec' [20]: as well as being robust and performant enough for use in artistic projects, the code base and supporting resources need to be sustainable well beyond the lifetime of the research project.

Clearly some of these goals are in tension with each other; however, we believe the resulting toolkit strikes a reasonable balance and makes a valuable contribution to the goal of helping to animate current and future research. The following sub-sections describe briefly the different

[4] The language side of SuperCollider

[5] https://www.flucoma.org/commissions/

[6] For source code and binaries, visit https://flucoma.org

object categories in the framework. Section 4 will outline some of the workflows these make possible.

3.2 DataSets

The DataSet object is the core component for all data processing algorithms. It is a key-value store with unique string identifiers as keys and numerical vectors as values. The purpose is mainly to store feature vectors, but it aims to be as generic and versatile as possible within that scope. Other objects in the toolkit consume and produce DataSets as inputs and outputs.

DataSet entries are mostly introduced via the CCE's audio buffer objects, which is the main interface in our audio feature extraction objects. Buffers are familiar to creative coders and the only open-ended memory access available in all music CCEs. Many of our objects copy data to new DataSets, rather than making many objects depending on one or few monolithic DataSets. We found this approach works well for practical amounts of data using current computers and is typically easier to reason about than shared access, especially in multithreaded environments.

DataSet instances have a JSON interface, as do all other objects in the toolkit that hold significant amounts of data and variables in their state. This allows a widely supported way to store, back up and transfer data into different environments, as well as loading to and from native CCE dictionary structures. A sibling object is the LabelSet, which holds labels instead of feature vectors, and is used particularly in classification tasks.

3.3 Nearest Neighbours

Similarity queries are perhaps the most well-established mechanism in descriptor-driven corpus-based music making, such as concatenative synthesis or automated orchestration. Usually implemented via nearest-neighbour algorithms, they allow the association of sounds by proximity according to some acoustic feature or derived metric. Following established practice, we provide a KD-tree algorithm to perform efficient queries. Our implementation allows querying within a given radius around a point and can report the distances to the returned points to allow intuitive use of this feature. This algorithm is often significantly better than brute-force search, although limited with respect to data dimensionality and dynamically adding or removing tree entries.

3.4 Supervised Machine Learning

Supervised machine learning models a relationship between input and output data, both of which are supplied by the user, and that are related by some common set of labels. It is well established in creative musical contexts, as has been shown in longitudinal research examining sustained practice [21], as well as in many disciplines involved with music and audio analysis. This type of machine learning can be used both for classification (categorizing inputs into discrete classes), and for regression (devising a continuous mapping between a given set of inputs and outputs). These kinds of task can be quite intuitive for musicians, especially if they have experimented with machine listening, and tried to handcraft code for such purposes.

The toolbox provides both K-nearest neighbours (KNN) and multi-layer perceptron (MLP) regression and classification. KNN models are lightweight and very quick to train, query and experiment with, yet limited in the complexity of relationship it can model. MLP models are more appropriate for more difficult problems. Our implementation allows flexible specification of the architecture and a choice of activation functions, which affords a great number of configurations and combinations. They also allow both input and output access to any layer, so an MLP can be used as a feature extractor or to learn complex mappings: for example, an autoencoder [22] setup can be used to learn hidden representations from data. While ever more possibilities and layer types keep appearing in the deep learning community, our estimation was that a tailored implementation has better chances of providing a good balance of flexibility and complexity within the computing power available to most of the creative coding community.

3.5 Unsupervised Machine Learning

In unsupervised learning schemes, only input data are provided, and the algorithm attempts to model it directly. Although this means that the resulting outputs may be more abstract, this approach requires less preparation and can be very useful for learning general features or spaces that relate items in a DataSet. Given the arduousness of obtaining supervised training data for arbitrary creative endeavours, we expect unsupervised machine learning to be particularly useful in data-driven musicking.

While recent developments in general machine learning have not been as spectacular as in the supervised case, in our view there is still much promise for establishing existing unsupervised ML tools and paradigms in creative domains. Our main focus is data clustering and dimensionality reduction.

With respect to data clustering, the toolbox includes an implementation of the classic K-means algorithm, which has a long tradition in audio signal processing [23]. While the output of clustering may seem a bit abstract for a creative workflow, it can be used as a building block for complex interfaces. We plan to add other clustering algorithms to overcome the classic limitations of K-means (globular shapes and a pre-defined number of clusters) as well as metrics for assessing clustering quality.

The use of dimensionality reduction is perhaps more straightforward as it has direct applications for visualization and parameter mapping, as well as pre-processing data for other ML tasks. In previous research [24] we demonstrated the creative potential for adaptive interfaces based on the results of different dimensionality reduction algorithms. Based on the intuitions gained, we provide a selection in the toolset: Principal Components Analysis (PCA), which is also generally useful for linear mapping and preprocessing of high-dimensional data; Multi-Dimensional Scaling (MDS), which allows easy experimentation with different distance measures between data points; and Uniform Manifold Approximation and Projection (UMAP) which can be seen as the state-of-the-art algorithm for dimensionality reduction. As mentioned above, MLP autoencoders can also be used for non-linear dimensionality reduction.

3.6 Data scaling and normalization

Data scaling and normalization are generally necessary when using any of the described ML algorithms, as imbalances in the scale of the different dimensions can make it very hard to converge in multi-dimensional spaces, and some models expect inputs to be centred and / or scaled. The toolbox provides a set of common pre-processing objects for normalization, standardization and robust scaling, alongside other housekeeping processes such as thresholding and outlier removal.

3.7 Querying

A trend of feedback from many of the early adopters was a keen interest in manipulating the numerical data (typically from audio features) in DataSets. SQL-like interfaces have been used in some cases to interact with audio corpora [8]. The toolbox provides an object to accommodate this kind of interaction, allowing manual filtering of the rows and columns of datasets as well as compositing before further processing.

3.8 Abstractions and Utilities

Finally, utilities are provided for accommodating common workflows and dealing with specific affordances of each CCE, including buffer processing and real-time querying operations. We provide some flexible abstractions that speed up the process of assembling a DataSet from a starting collection of audio samples/files, allowing parameters for batch analyses to be expressed concisely and abstracting away boilerplate code.

4. USAGE EXAMPLES

We believe that the value of the proposed framework rests in large part on the affordances it yields for musicians to experiment with these technologies in familiar environments, so as to stimulate more and better-documented practical research on creative strategies and tactics around their use in practice. A key methodological commitment of the project is that our designs are informed and interrogated at the earliest opportunity through intense use by musicians outside the development team. We are lucky to benefit from the input of a group of commissioned artists as well as a growing pool of enthusiastic and engaged voluntary alpha-users. As previously stated, we hope that the resulting discussions will go beyond implementation and engage across the various communities around each respective CCEs. We will now describe examples of creative workflows enabled by the toolbox, as well as a more detailed example, to provide an idea of how objects can be applied to musical practice.

4.1 Data-driven workflows

We have requested that our early co-investigators contribute to a series of online presentations[7] discussing their projects and experiences. Some common patterns of workflows we have observed can in these early entanglements be grouped as follows:

- **Corpus building:** The tools for statistical processing (scaling, clustering, dimensionality reduction) have been used to clean and filter sound corpora obtained in different ways (e.g. mass generation, sonification, informal recording). This includes removal of outliers or near duplicates; finding good references and ranges for intuitive descriptor values; and sanitising data across dimensions.
- **Dynamic querying:** As opposed to fixed workflows based on similarity queries in concatenative synthesis or automatic orchestration tools, the modular interface of our toolbox allows a more nuanced approach to complex material. For example, sounds can be modelled into different segment decompositions (e.g. stages of the energy envelope) which are analysed separately, then either merged or indexed into multiple descriptor spaces. Similarity queries can also be *forked:* different indices can be queried depending on the value of a given descriptor. A typical example is using pitch features only for voiced segments of a certain duration, otherwise reverting to spectral centroid.
- **Exploring parameter spaces:** Machine learning can be used to create non-linear mappings of parameter spaces of complex synthesizers and processors. For example, MLP objects are configured and trained to expand from a few dimensions (which can be more easily mapped to input devices and interfaces) to many parameters; control can also be regressed to the descriptor space of its audio output.
- **Adaptive interfaces:** As shown in [24], interactive scatterplots for playing sound corpora can be extended using dimensionality reduction and visualization (which can also represent clusterings and classifications) to make playable interfaces that adapt to the data distribution. Examples include interaction with a touchscreen or other controllers, or sequencing of spatial trajectories and patterns.

4.2 Tuning Nearest Neighbours

Below is an example of dynamic querying in more detail. This comes from a studio composition by the first author where similarity-based queries were used to concatenate new utterances from a corpus of spoken voice material.

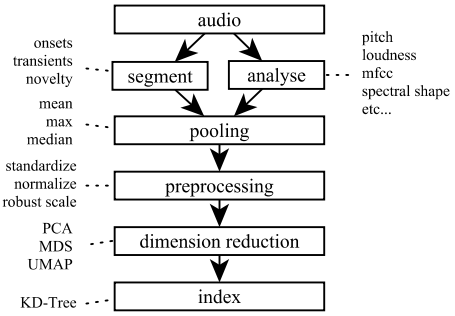

Figure 1: A typical workflow for similarity matching.

[7] Available at www.flucoma.org/plenaries/

For context, Figure 1 shows a typical workflow for similarity matching, alongside options available in our toolkit for each stage. For training, each stage will normally be performed in batch. Collections of host buffers are converted to a DataSet, and each modelling stage produces a new DataSet that serves as input to the next stage. At query time, new data points (in CCE buffers) are passed through the pipeline to the KD-tree, which returns the nearest sounds from the corpus.

In practice, a great deal of experimentation is required. The granularity of segmentation, choice of features, pooling mechanism, and so forth, will all affect the behaviour of the similarity query at the end of chain. Moreover, in a creative context, often the desired behaviour is not known and fixed but forms through the process of exploration.

The first author found that he was able to enrich the workflow above so that options could be explored interactively and fluidly during composition. The ability to filter and merge different DataSets using the DataSetQuery object meant it was possible to quickly trial different combinations of descriptors, scaling schemes, and dimension reduction models. Additionally, the segmentation could be adjusted *ex post* by inserting a clustering model into the chain prior to pooling: starting with a purposely fine-grained segmentation, K-means was used to identify adjunct segments that could be re-joined, enabling an effective way to discover a timescale that worked well.

Crucially, new ideas and possibilities suggested themselves during exploration, in a way that would have been much less likely in a less interactive, more disjointed setting. Moreover, this workflow was able to fit well into the first author's distinctive practice, which bridges working in CCEs and DAWs in *ad hoc* ways (through audio sends, MIDI events, or EDLs), easily hacked together in context to best fit the need of the musicking moment.

5. PRELIMINARY ASSESSMENTS

The first version of the new extensions was released to our community of creative coders in November 2019, at that point comprising eleven participants plus the authors, augmented by a small group of eager people along the way. Throughout the process, we released nine private updates and we are now in public beta. Whilst it is still too soon to reflect in detail on how well the proposed framework supports sustained musical research, and notwithstanding the encouraging range of possible uses described above, we have noted some early trends in people's encounter with the tools that will guide our future work.

5.1 Framework Aims

The variety and sophistication of the workflows described in Section 4 provide encouraging signals that our aims for the toolkit are being fulfilled. Our early adopters have been able to integrate the tools into their diverse range of broader working patterns and been able to complete a variety of new work that would have been otherwise much harder. This suggests that the tools themselves are robust, performant and usable enough to support serious creative work, and that the coverage of algorithm types provides enough breadth for a range of ideas to be explored and brought to fruition. From a framework development perspective, the C++ architecture has made it easy to add and adapt objects to the toolbox.

At the time of writing, some avenues for enhancement are already clear, based on feedback from our contributing artists and the authors' own experiences of creative work with the tools. Most prominent among these are some rough edges around idiomatic integration into CCEs.

5.2 Project Aims

The workflows in Section 4 took shape in a context of continual exchange and discussion, on the project forum and through regular online group meetings where early adopters shared work in progress on their respective projects. Such communication on musical mining has been helped by the consistency of concepts and interface between the Max and SC implementations. This communal approach has been significant not just in showing what can be done, but in helping form shared understandings and research ideas, and augurs well for the longer work of stimulating research more widely.

Early experiences with the framework have also suggested some possible additions to the toolkit to consider. One avenue would be to look at making available some algorithms geared at analysing temporal evolution of features, which is not always easy to account for in the current suite yet is so integral to musicking. Another area is to enlarge the range of tools for validating learnt models in supervised learning workflows, to help streamline their use.

Further optimisations of the code base will be enabled, by observing the toolkit's performance in practice, and what sorts of size and complexity of collection people end up gravitating towards, with a particular perspective on more investigation of the real-time possibilities and hurdles.

Finally, early experiences have also signposted areas in which the broader project could help musicians become fluent with, and critical of, these technologies. We have noticed, in particular, that for those with little prior experience in the language and techniques of data science, the conceptual jumps can seem quite daunting: people are confronted with abstract features and procedures far removed from musical intuitions, and with the task of relating objective validation to subjective expectations.

Moreover, the hype around the general and musical potential of artificial intelligence and machine listening requires careful calibration of expectations whilst still showing their creative potential when embraced divergently.

6. CONCLUSIONS / FUTURE WORK

This paper presented the second iteration of the Fluid Corpus Manipulation toolbox, a new software framework for programmatic soundbank mining in CCEs. Combined with its previous iteration, it offers a broad system that enables programmatic data-driven musicking, for interaction with audio and other data corpora within popular creative coding environments. Early usage and feedback from a community of users suggest that through several iterations, the toolbox has fulfilled its design goals with respect to native integration, consistency, learnability and configurability of

interface, scalability and breadth, while enabling complete mining-as-musicking workflows. We have shown a range of possibilities that emerged from commissioned works, and we hope that further interest in the toolkit will be stimulated by the premieres of these works, as well as their documentation.

In the final year of the FluCoMa project, we intend to continue growing the toolbox with new objects or new categories of object. Several signal processing tools for hybridizing sounds and interpolating gestures are already available [25]. We are also developing a platform of supporting learning resources that respond to the kinds of need identified in Section 5. In addition, we will run workshops and publish workshop materials and essays, with the hope that we continue to benefit and learn from diverse and enthusiastic feedback and discussion that can inform future technical and critical work in this area.

Our long-term hope is that engagement from a broad cross-section of the creative coding music community will help to turn this into a collaborative effort that supports and informs sustained, collaborative musicking research.

Acknowledgments

We would like to thank the creative coders engaging in the alpha community (James Bradbury, Rodrigo Constanzo, Richard Devine, Alice Eldridge, Daniele Ghisi, Leafcutter John, Lauren Hayes, Ted Moore, Olivier Pasquet, Sam Pluta, Hans Tutschku), the CeReNeM and its Creative Coding Lab, and the European Research Council, since this research was made possible thanks to a project funded under the European Union's Horizon 2020 research and innovation programme (grant agreement No 725899).

7. REFERENCES

[1] C. Small, *Musicking: The Meanings of Performing and Listening*. Wesleyan University Press, 1998

[2] P. A. Tremblay, O. Green, G. Roma, and A. Harker, 'From Collections to Corpora: Exploring Sounds through Fluid Decomposition', in *Proceedings of ICMC, 2019*

[3] O. Green, P. A. Tremblay, and G. Roma, 'Interdisciplinary Research as Musical Experimentation: A case study in musicianly approaches to sound corpora', in *Proceedings of EMS*, 2018

[4] F. Pedregosa *et al.*, 'Scikit-learn: Machine Learning in Python', *Journal of Machine Learning Resources*, vol. 12, pp.2825–2830, 2011

[5] L. Buitinck *et al.*, 'API design for machine learning software: experiences from the scikit-learn project', in *ECML PKDD workshop: Languages for data mining and machine learning*, 2013, pp. 108–122

[6] B. Hackbarth, N. Schnell, and D. Schwarz, 'AudioGuide: A Framework for Creative Exploration of Concatenative Sound Synthesis', IRCAM, IRCAM Research Report, 2010

[7] G. Carpentier, E. Daubresse, M. G. Vitoria, K. Sakai, and F. V. Carratero, 'Automatic Orchestration in Practice', *Computer Music Journal*, vol. 36, no. 3, pp. 24–42, 2012

[8] D. Schwarz, G. Beller, B. Verbrugghe, and S. Britton, 'Real-Time Corpus-Based Concatenative Synthesis with CataRT', in *Proceedings of DAFx,* 2006

[9] L. Garber, M. A. y Ciencia, T. Ciccola, and J. C. Amusategui, 'AudioStellar, an Open Source Corpus-Based Musical Instrument for Latent Sound Structure Discovery and Sonic Experimentation', in *Proceedings of ICMC,* 2020

[10] R. Fiebrink and P. R. Cook, 'The Wekinator: A System for Real-Time, Interactive Machine Learning in Music',in *Proceedings of ISMIR,* 2010

[11] J. Bullock and A. Momeni, 'ml.lib: Robust, Cross-Platform, Open-Source Machine Learning for Max and Pure Data.', in in *Proceedings of NIME,* 2015

[12] B. D. Smith and W. S. Deal, 'ml.* Machine Learning Library as a Musical Partner in the Computer-Acoustic Composition Flight', in *Proceedings of ICMC*, 2014

[13] A. Agostini and D. Ghisi, 'Bach: An Environment for Computer-Aided Composition in Max', *Practical Applications*, no. 23, p. 6, 2012

[14] D. Ghisi and C. Agon, 'Real-Time Corpus-Based Concatenative Synthesis for Symbolic Notation', in *Proceedings of the TENOR*, 2016

[15] N. Collins, 'SCMIR: A SuperCollider Music Information Retrieval Library', in *Proceedings of ICMC,* 2011

[16] N. Schnell, A. Röbel, D. Schwarz, G. Peeters, R. Borghesi, and I. C. Pompidou, 'Mubu & Friends-Assembling Tools for Content Based Real-Time Interactive Audio Processing in Max/Msp', in *Proceedings of ICMC*, 2009

[17] N. Schnell, D. Schwarz, J. Larralde, and R. Borghesi, 'PiPo, A Plugin Interface for Afferent Data Stream Processing Modules', in *Proceedings of ISMIR*, 2017

[18] A. Harker and P. A. Tremblay, 'The HISSTools Impulse Response Toolbox: Convolution for the Masses', in *Proceedings of the ICMC*, 2012

[19] A. McPherson and K. Tahıroğlu, 'Idiomatic Patterns and Aesthetic Influence in Computer Music Languages', *Organised Sound*, vol. 25, no. 1, pp. 53–63, Apr. 2020

[20] B. Buxton, 'Artists and the art of the luthier', *ACM SIGGRAPH Computer Graphics*, vol. 31, no. 1, pp. 10–11, Feb. 1997

[21] R. Fiebrink and L. Sonami, 'Reflections on Eight Years of Instrument Creation with Machine Learning', in *Proceedings of NIME*, 2020

[22] H. Bourlard and Y. Kamp, 'Auto-Association by Multilayer Perceptrons and Singular Value Decomposition', *Biol. Cybern.*, vol. 59, no. 4–5, pp. 291–294, Sep. 1988

[23] S. Lloyd, 'Least Squares Quantization in PCM', *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–137, Mar. 1982

[24] G. Roma, O. Green, and P. A. Tremblay, 'Adaptive Mapping of Sound Collections for Data-driven Musical Interfaces', in *Proceedings of NIME*, Jun. 2019

[25] G. Roma, O. Green, and P. A. Tremblay, 'Audio Morphing Using Matrix Decomposition and Optimal Transport', in *Proceedings of DAFx*, 2020