From Collections to Corpora: Exploring Sounds through Fluid Decomposition

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

ABSTRACT

This paper introduces the Fluid Decomposition Toolbox*, a set of extensions to enable programmatic signal decomposition for the most widely used musical creative coding environments. As part of the Fluid Corpus Manipulation project, the toolbox is aimed at helping to make musical sense of sound recordings, which we frame as a transition from a collection to a corpus. We present tools for decomposing signals in terms of 'slices', 'layers' and 'objects', and discuss the larger aims of the project and its toolkits.*

1. INTRODUCTION

Much computer music revolves around working with some form of recorded sound. Despite this, and despite lowering storage costs making it easier to build up very large collections of recorded material, there is a surprisingly limited palette of options for working with such large collections in musical creative coding environments. Meanwhile, there are promising techniques from DSP, MIR and machine learning research that could be of musical value for working with large collections that have either never appeared in the ecosystems that these musicking environments support, or have appeared but without sufficient documentation, or have failed to become embedded for some other reason.

This paper presents early results from a longer-term project on *Fluid Corpus Manipulation*[1], that aspires to address such a gap between available technologies and the means to explore them musically. The overarching aim of the project is to seed a musically diverse community of practice around this broad topic, by making available technologies, accompanied by musically-focused learning resources. The musical impulses driving this work are to provide scaffolding that enables creative coders to explore and embed ways of querying corpora, hybridising and interpolating sounds, and playing with sound-symbol relationships.

The project also aims to make a methodological contribution, by taking the opportunity to explore the prospects for a practice-led cross-disciplinary collaboration between technological and music-practice researchers. Whilst these collaborations are quite common in practice, they seem seldom documented or theorised in such terms. Drawing on the insights in [1] we hope to develop both methodological and practical tools that might help nurture a productive cross-disciplinary agonism between computer musicians and colleagues in allied technical disciplines, with the aim that each finds better ways of understanding their shared object of study. For further discussion, see [2].

In the present work, we are focused on an initial offering of tools concerned with *making* a corpus from a collection of recordings or real-time streams, through various techniques of signal decomposition. The result of this has been the first of two planned toolkits which, so far, offers a suite of Max[2] and Pure Data[3] (Pd) externals and SuperCollider[4] (SC) plugins to perform decompositions both in real-time and non-real-time, as well as a command-line interface.

The following section discusses the specific aims and questions of this initial phase of work. We then go on to discuss the methodology that we have arrived at, before presenting the toolkit itself and discussing the musical affordances that have been found so far.

2. AIMS AND QUESTIONS

2.1 What is a Corpus?

Within the planning stages of the project, it became clear that our desire to support diverse musical practices meant trying not to foreclose aesthetic possibilities too much with our own musical assumptions, and that thinking through what constitutes a *corpus* required some care. What, if anything, distinguishes a corpus from any old collection of recorded sound?

Our position is that the distinction *could* be as small as just a shift in one's intentful orientation towards a collection; that is, corpus-hood connotes something that musicians have settled down to explore, with some project in mind. Clearly, though, such explorations could take a wide variety of forms. Consider, non-exhaustively:

- 'Atomic' sounds being queried by feature, as in concatenative synthesis [3] or concatenative orchestration [4].
- Sections sampled from other recordings, to be explored for potential combinations, as in Hip-Hop.
- Snippets built up in live performance, perhaps by some kind of automatic sampler.
- Long recordings, for instance of soundscapes, being explored for moments of particular interest.

Each of these scenarios implies a musician asking *different questions* of a body of recorded materials, looking for different things, some of which a computer can help with, others not. Also, they imply different steps in preparing a

[1] http://www.flucoma.org
[2] http://www.cycling74.com
[3] https://puredata.info/
[4] https://supercollider.github.io/

collection in the first place–some methodical and labour-intensive, others more exploratory–as well as distinct *modes of listening*, at varied time-scales. This variety isn't just limited to distinct scenarios: from moment to moment, musicians will be interested in exploring various levels and types of detail. For our purposes here, *signal decomposition* becomes something of an umbrella to collect together different manoeuvres we may perform in preparing a corpus. Through our toolkit, we aim to facilitate programmatic exploration of this process for creative coders.

2.2 Taxonomy of Decompositions

A key question is how to reconcile the open-endedness of the things that people may listen for with the available technical possibilities, to yield a taxonomy for guiding the design of both software and learning resources. Moreover, because we are interested not only in addressing problems that are already known, but want also to open up a space for communally exploring and learning musical possibilities, the question becomes one of how to provide mappings between the kinds of thing a musician might ask, and the technical affordances now available.

We have arrived at a simple, three-term taxonomy that describes the different kinds of entity that a musician might split a sound into:

- *Slices*: A simple segmentation in a single dimension – most usually in time. At a minimum, isolating gestures is a foundational part of preparing a recording for future use. We may also want to split at key moments, or try to locate longer-term gestures. In principle, one could slice on another basis, e.g. in frequency.
- *Layers*: Approaching a sound as being an additive combination of some sort, and decomposing along those lines. This could be separating into archetypes, such as sinusoids and transients, or exploring source separation techniques that model signals as additive mixtures.
- *Objects*: A more open-ended category, where one might be looking for a *type of thing* in a recording, with more or less specificity. One could look for instances of certain specific time-frequency patterns, or for more loosely specified morphological characteristics.

2.3 Design Criteria

For the first incarnation of the toolbox, we set out to provide a suite of extensions for creative coding hosts, with each of the areas above served by at least one object that encapsulates some published algorithm (see §4). As well as ensuring that we fulfilled foundational design criteria for artistically-useful software, as established in prior work [5], we try to balance the following considerations.

2.3.1 Target Audience

The toolkit is aimed at people who, we assume, already have a degree of fluency in their chosen host environment, and are trying to push its possibilities further. At the same time, a certain amount of thought has gone into trying to avoid taking for granted that such fluency implies an appetite for, or familiarity with, technical detail or mathematical explanation. Rather, we treat these as a matter of *disposition* rather than ability, and try to design our interfaces and learning resources (§2.4) on the basis that different registers of technicity will suit different people, differently, at different points in their 'production cycle' [2]

2.3.2 Encapsulated and Idiomatic

Each object should be useable in isolation. In contrast to powerful suites, such as FTM [6], MuBu [7], FrameLib [8], and Bach [9], we are not making a *framework* and do not want to make people learn a new grammar. It follows that our objects should be *uncoupled*, i.e. should not rely on communicating with each other, and should rely on the data structures and idioms of their host environment, to also enable a wider range of unforeseen usages.

2.3.3 Granular

The toolbox must balance the need to be approachable, yet enable and reward progressive, divergent exploration. This is a fundamentally ill-posed problem. On the one hand, simply providing an implementation of some abstract algorithm, with all its parameters exposed, is unlikely to be musically suggestive or inviting. On the other hand, black-box encapsulations geared at curated, high-level tasks can shut down possibilities to explore, and will imprint our assumptions about what is musically useful or desirable.

In practice, the solution is to aim for multiple ways in: to provide both encapsulations of operations that are frequently called for, as well as making lower-level objects available for more experimental play.

2.3.4 Back-end Reuse and Rapid Development

This is an open-source toolkit that we hope will attract enough interest and involvement to outlive the project. As such, making the code re-useable and maintainable is a key priority. Our implementations of algorithms should not be tightly-coupled to the host environments, but should be available for future use in other contexts.

Meanwhile, to support flexible development that can respond swiftly to feedback from practitioners, we need to enable a rapid path from developing an algorithm to producing a working host plugin. This means being able to specify the behavior of encapsulated objects in a way that is independent of host-specific issues, and bundle the details of the respective APIs in a generic, yet flexible way.

2.4 Learning Resources

Because we aim to make available techniques and tools that have not yet received a great deal of musical attention in creative coding contexts, and because we are more interested in fostering a community around exploratory usage than in providing a pre-packaged suite of 'solutions', it is critical that the tools are supported by extensive learning resources, pitched at appropriate levels.

In addition to a rich set of help files, tutorials and examples within each host, we aim to provide two online resources. First, a forum in which the aspired-for community can come together to exchange ideas and support. Second, a website that provides musically-orientated explanations of the algorithms and techniques available.

As with the solution of providing objects at various granularities (§2.3.3), these resources enable multiple points of entry, to offer ways in to people with a variety of dispositions towards technical details. However, we consider the level and clarity of explanations offered by

Wishart in [10] to embody a useful basis from which to start, enriched by the imaginative use of visualisation and interactive display that can be seen in contemporary online resources, such as the Distill journal[5].

2.5 Practice-Led Design

As noted in §1, we aim for the overall project to make a methodological contribution by paying particular attention to the ways in which practice-led and technological research can enrich each other's understanding. Our strategy here is to try and make artistic interrogation of our tools and assumptions as integrated and continual as possible, and to expose the toolkit to a progressively widening cast of practitioners, in the hope that new and surprising perspectives will present themselves. We contend that, to be effective, practical exploration needs to be as thoroughgoing as possible. To that end, two waves of professionals, covering a range of musical proclivities, are given fully-compensated commissions, using very early incarnations of the toolkits, culminating in public performances at the end of a year's investigation. In this way, we hope that a variety of tactics for forming and playing with corpora get to be rigorously worked-through, and that the artists' continual engagement and feedback supports our aim for a flexible, open-ended set of exploration-orientated tools.

3. METHODOLOGY

To give an impression of how our software-development, learning resource and artistic-integration aims intermingle, we offer a brief overview of our process. We proceed through a process of internal prototyping and development, through to our commissioned artists' first encounter with the toolkit, to preparations for public release.

3.1 Prototyping and early tests

Much of our prototyping was done in Python. In addition to having excellent facilities for signal processing, linear algebra, and visualization, one of the authors had led the development of a robust library that implements a number of the algorithms we were interested in exploring [11].

To explore our vision for artistic interrogation and knowledge exchange at a local, team level, we established a pattern of work that has largely continued into later development. The authors broadly embody a gradient of familiarity with the algorithms we are investigating. This enabled us to adopt a workflow where one person's 'naïveté' is deliberately preserved, by avoiding implementation details and focusing on musical questions, whilst the other team members, with their technical insight, are responsible for developing useful explanations and examples.

In parallel, we also needed to start to explore things in a more interactive setting. For this we were able to use Framelib [8], and to start developing prototype objects for Max and SC.

3.2 Version 0.1

From our prototyping, we arrived at an architectural scheme to support the requirements outlined above (§2.3), encapsulating functionality into three layers: one that provides base algorithms, one for composing algorithms into task-specific clients, and a layer for host-specific wrappers that embody interface conventions idiomatic to the respective environments (e.g. using attributes in Max).

To facilitate rapid development, we use the Eigen library [12] for vector and matrix computing, and the FFT used in [5], [8], but encapsulate these dependencies into the algorithmic layer to build in some resilience to future change. At this stage many implementation details were still to be discovered, mainly for the client and host-wrapper layers, so a certain amount of code duplication had to be tolerated to produce a workable suite of objects.

Wherever possible, the toolkit made available real-time and offline versions of algorithms, with the latter using the native buffers of the host environment. In Max, offline processing took place on the main thread, and in SC, we used the asynchronous command facilities to process buffers on the server's non-real-time thread [13] .

3.3 The Artists' First Encounter

The artists were introduced to the first version of the toolkit objects at a project plenary in September 2018. As we will be working with this group of eight practitioners for a total span of two years, in two waves, part of the purpose of the plenary was to explore the possible boundaries and hopes for the project, before introducing the concrete tools on the final day (of three), in concert with a first sketch of the learning resources, using explanations of algorithms developed during prototyping. At this point, our online forum, using Discourse[6], was also established, albeit on an initially private basis.

Whilst the feedback we got was encouraging, it was also very clear that the learning resources needed considerable further development to really fulfil our goal of providing an inviting and inspiring way into the toolkit. Even for a group of very experienced creative coders, well-practised in devising divergent techniques, the consensus was that a gentler run-up, more focused on foregrounding some of the musical possibilities would be appreciated. One take-away from this is that, despite our efforts, the team as a whole was still too embedded in the project's details to fully anticipate a newcomer's experience, which signals the usefulness of getting such feedback sooner rather than later.

Fortunately, the engaged and thoughtful feedback we got from the artists has continued on our forum in the period since. As well as much useful critical comment on issues of interface and performance, we have–as hoped–been able to watch and support the musicians delve deeply into the tools as they prepare their commissions.

3.4 Towards Public Release

Much of the work between the alpha version and a public-beta has focused on integrating feedback from our musicians, refactoring code, and adding a few new objects. At

[5] https://distill.pub/about/

[6] https://www.discourse.org/

the time of the plenary, we were quite aware that some of our interface decisions were still cumbersome, but that engaged feedback would be needed to help choose between alternative improvements. For instance, many objects offer too many options, and that these need reorganizing; also, some objects will benefit from being encapsulated with some high-level abstractions for new users.

Meanwhile, having tolerated a certain amount of code duplication for the alpha version, this didn't serve our goal of rapid development. As such, the client and host-wrapper layers were refactored to take better advantage of modern C++'s facilities for generic programming, and to condense the host-layer down to a single template wrapper per target environment, making integration into further hosts in the future a simpler matter. Feedback from our musicians has also helped us to embark on a thorough re-working of the learning resources. Whilst much of this work consists simply in fleshing out our initial sketches, we have also sought to provide more, and richer ways in.

The next process, in progress at the time of writing, is an extensive beta testing of the revised toolbox with creative coders with a wide breadth of experiences and interests, as well as supporting more operating systems and polishing the wrappers for Pd and the command-line.

4. LIBRARY CONTENT

We provide a brief overview of the algorithms that the toolbox provides access to, grouped along the lines of the taxonomy from §2.2. Some of these clearly address a single aspect of our *slices-layers-objects* model, whereas others can be plausibly addressed to any of the three. We focus here more on the musical affordances so far discovered, than on technical detail.

All internal processing happens using 64-bit floating point arithmetic, although the native buffers in Max, Pd and SC, and the real-time audio in SC, use 32-bit floating point. All of our spectral decompositions achieve resynthesis through time-frequency masking [14], which allows for almost perfect reconstruction.

4.1 Slices: Temporal Segmentation

We provide a range of options for slicing signals in time. Real-time variants produce an output signal of 0s and 1s, and offline variants produce a buffer of slice indices. As well as more familiar onset-based segmentation, we are interested in the possibilities of segmentation at longer timescales, as well as the affordances of a programmatic interface, such as being able to *tune* the segmentation on the basis of a desired density of slices.

4.1.1 AmpSlice and BufAmpSlice[7]

The simplest way to slice is by detecting significant amplitude changes. Our approach uses two stages of thresholding, one fixed and one adaptive. Both have time and threshold hysteresis, use separate attack and release settings, and can request a minimum duration for each state (on or off). This scheme offers great flexibility at low computational cost. Its uses range from classic silence removal, to identifying and subdividing loud segments in musical passages, to very tight time-based event detection.

4.1.2 OnsetSlice and BufOnsetSlice

SC has enjoyed a suite of spectral onset detection algorithms [15] for some time, whilst Max and Pd have more limited options. We provide an expanded range of choices for the musician to select from [16]. These allow slicing a signal along lines that are possibly independent of level, such as changes in pitch or timbre, which can be useful for signals that don't have pronounced amplitude profiles.

4.1.3 TransientSlice and BufTransientSlice

Using the transient modelling algorithm described below (§4.2.3), a recording can be segmented at impulsive moments. Whilst this method is quite CPU intensive, it allows for very sensitive and precise slicing.

4.1.4 NoveltySlice and BufNoveltySlice

To explore the possibility of slicing at a range of timescales, we provide an implementation of the algorithm developed by Foote [17], which has a simple mechanism for segmenting based on the convolution of a 2D kernel with an arbitrary feature representation. Because of the simple mechanism for adjusting the temporal scope of the algorithm, it affords creating segments that exhibit cohesion in relation to the features of interest.

4.2 Layers: Separation using Additive Models

These algorithms model signals as being an additive combination of components of some sort. By using masking in resynthesis, obtaining a residual is simple, and make the objects simple to compose. For instance, one could build a sines-transients-residual model by combining the objects from §4.2.2 and §4.2.3.

4.2.1 HPSS and BufHPSS

The harmonic-percussive source separation algorithm (HPSS), first proposed in [18], provides a computationally cheap way to separate a sound into more tonal and more transient layers, using median filtering of the spectrum. We implement both the original version, and the extension proposed in [19], which allows one to produce an additional residual layer. We also add our own extension that refines the latter version by introducing frequency-dependent thresholding. This can reduce artefacts and allow for a more uniform separation across the spectrum. The simplest use for this process is to adjust the balance between percussive and tonal elements of a complex signal, relatively transparently. Using larger adjustments, or layers in isolation offers an exploratory way to derive new materials from composite sounds.

4.2.2 Sines and BufSines

Sinusoidal modelling has a long pedigree in musical signal analysis [20], but we lack implementations in creative coding environments that offer both high-quality analysis and

[7] As algorithms will adapt their name to respect each environment's idiomatic naming conventions, we present here the common base. For instance, *AmpSlice* becomes `fluid.ampslice~` in Max and Pd, and `FluidAmpSlice` in SC. Also, the Buf prefix refers to the non-real-time version of the algorithm (e.g *BufAmpSlice*). Finally, the command-line interface implements only these latter versions, without a prefix and all lowercase (i.e. `ampslice`).

resynthesis. Our implementation uses peak-classification [21] and peak-tracking to tune the process for different complexities of material, and produces a residual that can be sent for further decomposition. Having this control at a programmatic level brings new affordances, as we can now *play* with the decomposition on the basis of other musical or analytical parameters; for instance, adaptively 'de-voicing' material in real-time.

4.2.3 Transient and BufTransient

Whilst transient modelling algorithms have been an area of active research for some time [22] and are used in audio coding and restoration, they have perhaps been hitherto too CPU intensive for creative coding use. We employ an algorithm from the restoration literature [23], again providing a residual signal that can be further decomposed. Despite the heavy computational cost, this algorithm has offered fascinating sound design possibilities. For instance, the very short duration of the decomposed transients allows for creative use of convolution with complex kernels, to produce new, hybrid materials.

4.3 Objects: Finding Shapes

Extracting and detecting time-frequency morphologies from signals is the most open-ended and challenging area of our taxonomy. Our starting point has been to implement and explore the well-established Nonnegative Matrix Factorisation (NMF) algorithm [24], which can be used as a flexible device for both supervised and unsupervised learning of signal components.

4.3.1 BufNMF, NMFMatch, NMFFilter

NMF has enjoyed a long history as the basis of much research in audio source separation [25], and provides the most flexible of our algorithms so far. One way of imagining how NMF works is as an adaptive vocoder: the algorithm attempts to iteratively decompose a magnitude spectrogram into a set of sub-spectrograms that sum together to the original. These components are expressed as a set of discovered 'filter shapes' (bases) and temporal envelopes (activations).

We offer a flexible offline object, *BufNMF*, that can be used to decompose a signal into a given number of layers. By default, the decomposition is unsupervised, but can also be used in a supervised mode, using either pre-discovered dictionaries or activations. This added flexibility means that the process can also be used for temporal slicing, or finding 'objects' in a signal. This latter task is also available in a real-time variant, *NMFMatch*, that produces activations against pre-made bases, and an extension, *NMFFilter*, that will resynthesise detected components.

Uses found so far include creating composite sounds from sub-components; generating variations by purposely 'over-decomposing' material; and selectively processing sub-components of complex signals. The offline object can be used to train dictionaries for use by the online variants. These can then be used for detecting desired spectral profiles in real-time streams, and for doing content-sensitive routing in real-time.

4.4 Utilities

As we started creative coding with the toolbox, we stumbled upon tasks where the tools built into the host environments were lacking some specific functionality.

4.4.1 BufCompose

Facilities for compositing offline buffers are surprisingly limited in both Max and SC. We provide an object, *BufCompose*, that enables buffers to be combined with enough flexibility to enable mixing, concatenating, stacking and a variety of other operations.

4.4.2 Curated Descriptors

Whilst there are a variety of available libraries for generating descriptors available in Max and SC, we provide access to a small range. Not only does this help provide consistency between our target environments, but also our segmentation algorithms need these, and they serve as a bridge to the second toolkit that our project will produce, which will concern querying and exploring corpora. The current selection of objects computes SpectralShape, MFCCs, MelBands, Pitch, Loudness, in both real-time and non-real-time. A BufStats object provides various statistics on the acquired time series, and their time-derivatives.

5. EARLY RESULTS AND FURTHER WORK

The early outcomes of this work are promising. As well as already featuring in the latest music of the first author, becoming acquainted with the affordances and quirks of these algorithms constitutes ongoing ear-training and opportunities to think and listen differently. At the time of writing, the first wave of users is also exploring these possibilities, and helping us to refine and reimagine future directions. For instance, HPSS has turned out to be useful as a pre-processing step before further analysis on otherwise challenging signals. Our research has also generated a range of other materials that document our various experiments, sub-projects, and discoveries[8].

Our next steps on this toolbox revolve around continuing to refine its codebase and its learning resources. There are extensions to some algorithms, particularly NMF, that we would like to explore, as well as integrating the descriptors more flexibly into our segmentation schemes. There are also limitations we would like to address, in particular, for longer offline processes, we will investigate the option of running these in a dedicated thread to avoid locking-up the host whilst they run.

In parallel to this work on the first toolbox, we will apply and refine this methodology and infrastructure in the design of the second one, focused on 'fluid manipulation'. In this next wave we will be investigating the scope of recent developments in machine learning to help musicians explore, play with and reshape their corpora.

6. CONCLUSION

We have presented the *Fluid Decomposition Toolkit*, as the first component in a project that aims to extend the scope

[8] See www.flucoma.org/publications.html for a full list of outputs.

for musicking with large corpora in creative coding environments. By distinguishing between a collection and a corpus as being a matter of musicianly intent, we position this toolkit as helping the transition from one to the other, in a flexible and contextually-sensitive manner. We also discuss the importance of rich learning resources to our broader project goal of developing a firmer basis for cross-disciplinary research driven by musical practice. Our strategy of integrating thoroughgoing artistic engagement from the outset has been helpful in making our early designs tessellate with the practices of a wider pool of musicians. It has also underlined the value of expanding engagement sooner rather than later in the development process, and is why an effort for proactively seeking out interested artists, and their insights, forms a major plank of our future plans.

Acknowledgments

We would like to thank the creative coders working with the alpha versions (James Bradbury, Rodrigo Constanzo, Nicolas d'Alessandro, Richard Devine, Daniele Ghisi, Leafcutter John, Lauren Hayes, 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] G. Born and A. Barry, "ART-SCIENCE: From public understanding to public experiment," *Journal of Cultural Economy*, vol. 3, no. 1, pp. 103–119, 2010.

[2] O. Green, P. A. Tremblay, and G. Roma, "Interdisciplinary Research as Musical Experimentation: A case study in musicianly approaches to sound corpora," presented at the Electroacoustic Studies Network Conference, 2018.

[3] D. Schwarz, R. Cahen, and S. Britton, "Principles and Applications of Interactive Corpus-Based Concatenative Synthesis," in *Journées d'Informatique Musicale (JIM)*, Albi, France, 2008, pp. 1–1.

[4] B. Hackbarth, N. Schnell, P. Esling, and D. Schwarz, "Composing Morphology: Concatenative Synthesis as an Intuitive Medium for Prescribing Sound in Time," *Contemporary Music Review*, vol. 32, no. 1, pp. 49–59, 2013.

[5] A. Harker and P. A. Tremblay, "The HISSTools Impulse Response Toolbox: Convolution for the Masses," in *Proc. of the ICMC*, M. Marolt, M. Kaltenbrunner, and M. Ciglar, Eds. 2012, pp. 148–155.

[6] N. Schnell, R. Borghesi, D. Schwarz, F. Bevilacqua, R. Müller, and I. C. Pompidou, "FTM—Complex Data Structures for Max," in *In Proc. ICMC*, 2005.

[7] 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," presented at the ICMC, 2009.

[8] A. Harker, "FrameLib: Audio DSP using frames of arbitrary length and timing," in *2017 ICMC/EMW*, 2017, pp. 271–278.

[9] A. Agostini and D. Ghisi, "A Max Library for Musical Notation and Computer-Aided Composition," *Computer Music Journal*, vol. 39, no. 2, pp. 11–27, Jun. 2015.

[10] T. Wishart, *Audible design: a plain and easy introduction to practical sound composition*. sine loco: Orpheus the Pantomime, 1994.

[11] G. Roma, E. M. Grais, A. J. Simpson, I. Sobieraj, and M. D. Plumbley, "Untwist: A new toolbox for audio source separation," in *Ext. abstr., Late-Breaking Demo Session of the 17th ISMIR Conf.*, 2016.

[12] G. Guennebaud, B. Jacob, and others, "Eigen v3," 2010.

[13] R. Bencina, "Inside scsynth," in *The SuperCollider Book*, 2011, pp. 721–740.

[14] Y. Wang, A. Narayanan, and D. Wang, "On Training Targets for Supervised Speech Separation," *IEEE/ACM Trans Audio Speech Lang Process*, vol. 22, no. 12, pp. 1849–1858, Dec. 2014.

[15] D. Stowell and M. Plumbley, "Adaptive whitening for improved real-time audio onset detection," in *International Computer Music Conference (ICMC) 2007*, 2007, pp. 312 – 319.

[16] S. Hainsworth, and S. Macleod, and Malcolm, "Onset Detection in Musical Audio Signals," in *In Proc. Int. Computer Music Conference*, 2003.

[17] J. Foote, "Automatic audio segmentation using a measure of audio novelty," in *2000 IEEE International Conference on Multimedia and Expo. ICME2000. Proceedings. Latest Advances in the Fast Changing World of Multimedia (Cat. No.00TH8532)*, 2000, vol. 1, pp. 452–455 vol.1.

[18] D. Fitzgerald, "Harmonic/Percussive Separation Using Median Filtering," in *Proceedings DaFx 10*, 2010.

[19] J. Driedger, M. Uller, and S. Disch, "Extending Harmonic-Percussive Separation of Audio Signals," in *Proc. ISMIR*, 2014.

[20] X. Serra and J. Smith, "Spectral Modeling Synthesis: A Sound Analysis/Synthesis System Based on a Deterministic Plus Stochastic Decomposition," *Computer Music Journal*, vol. 14, no. 4, pp. 12–24, 1990.

[21] D. Griffin and J. Lim, "A new model-based speech analysis/Synthesis system," in *ICASSP '85. IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1985, vol. 10, pp. 513–516.

[22] S. N. Levine and J. O. Smith III, "A Compact and Malleable Sines+ Transients+ Noise Model for Sound," in *Analysis, Synthesis, and Perception of Musical Sounds*, Springer, 2007, pp. 145–174.

[23] S. Godsill, P. Rayner, and O. Cappé, "Digital audio restoration," in *Applications of digital signal processing to audio and acoustics*, Springer, 2002, pp. 133–194.

[24] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, Oct. 1999.

[25] P. Smaragdis, "About this non-negative business," in *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2013, pp. 1–1.