NAVIGATING SAMPLE-BASED MUSIC: IMMEDIACY AND MUSICAL CONTROL IN RECENT ELECTRONIC WORKS

Alexander J. Harker

CeReNeM
University of Huddersfield
Huddersfield, HD1 3DH, United Kingdom
ajharker@gmail.com

ABSTRACT

This article outlines some of the technical processes used to exert musical control over sampled material in my recent compositional work. The pieces in question make use of custom software that facilitates a more musically meaningful interaction with computing technology. As well as addressing issues of appropriate musical representation, these processes also afford the composer an increased degree of immediacy than when working solely within the familiar context of the Digital Audio Workstation.

The article first addresses the context within which the work has taken place, as well as the issues that motivated the development of new technical tools. Techniques of sample selection using audio descriptor matching and systems of gestural representation are then discussed, along with less novel methods of musical control. Finally, the compositional implications of the various proposed approaches are explored.

1. CONTEXT

My background is as a composer of instrumental and electronic works, and in recent years an increasing number of works combining instrumental and electronic media. Much of my earlier electronic work centred around dense and intricate layerings of large sets of samples, and employed a Digital Audio Workstation (DAW) as the primary tool for composition.

In this article I discuss my use of custom software tools designed to allow more desirable methods of interacting with and controlling music based on recorded samples.

1.1. Programming and Creative Work

Programming has been a part of my creative practice for the last nine years. Initially, my interest in programming, and particularly digital signal processing, was borne out of a desire to support my own creative work. Alongside the necessary software to realise realtime pieces, I also began to feel that the possibilities of newer more complex technologies could, in my case at least, be best harnessed and customised for my artistic purposes by taking responsibility for programming lower level tools. Whilst such tools are rooted in my own creative aims, I also aim to make tools that might serve the needs of other practitioners. Thus, the underlying technologies for the pieces presented here (a set of externals for *MaxMSP*) is now available for download [2].

1.2. Pieces Discussed

This article discusses the approaches taken in two more recent electronic works. *Fluence* (2010) for clarinet and *MaxMSP* utilises a large bank of around 1600 dry clarinet samples to construct a realtime accompaniment to the composed instrumental part. This electronic part is to a large extent pre-determined in terms of musical shaping, but varies in exact realisation between performances. This is analogous to the manner in which the score for the clarinet pre-determines the sequence of notes and rhythmic events, but not exact details of phrasing, timbre, etc.

Fractures (2011) for fixed media employs custom processing tools (in *MaxMSP*) in conjunction with recordings of electric guitars and various small household objects.

The technology developed for both pieces was designed to enable more immediate and musically relevant levels of control than I had previously been able to achieve within the context of working within a DAW. Recordings of both pieces discussed are available for streaming and download online [3].

2. PROBLEMS AND MOTIVATIONS

2.1. Problems

Three main issues with working with a DAW motivated the development of custom software for use in both *Fluence* and *Fractures*.

Firstly, the process of constructing fixed media pieces from large sets of samples is overly time-consuming and non-immediate. Navigating long lists of sample names is unwieldy, and locating an appropriate sample to realise a given musical gesture often requires a lengthy process of trial-and-error auditioning. In extreme cases, the process of dealing with the placement and choice of individual samples in dense textures can result in such a slow rate of generating material (perhaps less than 15 seconds

in a week) that constructing longer pieces in this manner is totally impractical. More importantly, this process is a misdirection of time resources, and one that takes the composer away from dealing with more important concerns of shaping larger-scale musical structures. A lack of immediacy affects not only the practical speed of composition, but also the ability of the composer to effectively deal with the musical flow, always drawing attention towards the fixing (or non-realtime 'performance' of) details within the music.

Secondly, fixed realisations are not always desirable. When combining instrumental and electronic media it is often preferable to allow the performer(s) a level of flexibility that cannot be afforded by a static electronic part. In a piece such as *Fluence*, in which the rhythmic materials are generally fluid and flexible in nature, it makes little sense to make the performer suffer the inflexibility of fixed media.

Thirdly, and finally, existing software tools often make use of modes of interaction and representation ill-suited to the task of musical composition. This can be partially attributed to the fact that commercially available DAWs are most commonly based on a tape-recorder/mixing desk paradigm, two tools which are suited well to the task of recording, but not necessarily the representation and manipulation of musical ideas. This issue is perhaps more complex than the preceding two and requires a more detailed explanation.

2.1.1. Appropriate Representation of Musical Ideas

In *The C++ Programming Language* [6], Bjarne Stroustrup states that a programming language 'provides a set of concepts for the programmer to use when thinking about what can be done'. In order to achieve this, it should 'ideally [be] ...'close to the problem to be solved" so that the concepts of a solution can be expressed directly and concisely.' Whilst his discussion concerns programming, this idea can be generalised to any human-computer interaction in which the goal is to instruct the computer to perform a specific task. In a compositional context, this means expressing ideas in a manner 'close to the music'. Ideally, I wish to interact with the computer using representations that map closely to my ways of thinking about the important aspects of the music, rather than through representations that are simply convenient or appropriate.

To illustrate how this issue manifests itself in the context of a DAW, **adlucemshot** shows a screenshot of part of a earlier fixed media composition (*Ad Lucem* (2006)). Here we see that the graphical representation of blocks of audio and audio waveforms conveys something about when blocks of sound are present (or not), and perhaps some information about the dynamic envelope of each sound, but nothing (or almost nothing) about pitch-content, timbre, frequency range or countless other potentially crucial musical concerns.

Figure 1. Screenshot from *Ad Lucem* (2006)

2.2. Aims

In response to these issues four main aims emerge:

1. to replace manual auditioning and sample selection with an automated, but musically relevant sample selection procedure.
2. to exert meaningful control over the musical shapes created though abstract perceptual elements of sound.
3. to achieve variability and flexibility in performance.
4. to allow immediacy in the studio.

These aims imply certain musical values and ways of thinking. Unsurprisingly, they encapsulate my personal preoccupations as a composer, and as such the solutions proposed will be of most relevance to composers with a similar set of interests.

The idea that details of sample choices may be automated within a given framework implies that, given a large set of samples, these samples may be musically, or perceptually equivalent in a given context. This is to say that a sample would be chosen in order to fulfil given criteria and any sample fulfilling this set of criteria would be equally valid. This rejects the idea that it the absolute and concrete nature of a sample that is of most interest. Rather, it places an emphasis on the perceptual qualities that might be abstracted from a sound. Clearly, the second stated aim reveals that my interest is in creating musical structure through the manipulation of such perceptual qualities, either by forming suitable sequences of samples, or by processing samples such that these qualities can be directly controlled.

3. PROPOSED APPROACHES

3.1. Audio Descriptor Matching

In seeking to replace manual auditioning, it is necessary for samples to be accompanied by relevant metadata regarding their contents, such that an algorithm can be devised to select samples in a musically meaningful manner. Such metadata is known as an audio descriptor. In part, such metadata can be human-assigned as part of the editing process. This does not remove the initial need for audition, but after suitable labels have been assigned,

samples can be found through a search procedure, without needing to repeatedly reaudtion.

However, whilst this technologically simple approach has strong potential to allow selection based on the most obvious categorisations of samples, or descriptive tags, it is impractical to tag samples according to a large number of categorisations. This approach is also most suited to sonic characteristics that are easily divided into discrete categories. Where perceptual qualities are more continuous (such as the qualities of brightness, or loudness) it is preferable to use an algorithmic machine-listening approach. In this field, a large number of computable audio descriptors are available with which to analyse audio content [4]. In the *AHarker Externals* package, the *descriptors* ~ object allows for samples to be analysed using over twenty different numerical descriptors. Typically, values are calculated for small blocks of samples, and then averaged over the duration of the sample to produce a final value. However, it is also possible to derive values from the range, standard deviation, or to locate the frames with the largest (or smallest) values[1], in order to derive useful perceptual data about the behaviour of a sample over time. For example, one might locate the highest estimated fundamental frequency over the duration of a sample, or look at the rate of change in the loudness of a sample.

Ideally, for a compositional approach aiming to control perceptual qualities of sound, the most useful descriptors are those that map well to human perception. Pitch and loudness (meaning the perceptual loudness, rather than amplitude) are two obvious and tangible examples. However, there are other effective descriptors that deal with less traditional aspects of sound quality. The spectral centroid is a measure of the central tendency of energy within the frequency spectrum. This maps well to the perceived relative frequency levels of unpitched sounds. In the case of *Fluence* this descriptor is used throughout to select appropriate transient sounds (key clicks, etc.) to fit a particular frequency contour. The spectral flatness measure maps well to the perceived noisiness of a sound, and can be used to select more or less noisy samples. In future we can expect to see new, more musically orientated descriptors, such as Thomas Grill's recent work to derive a set of descriptors relating to perceptual qualities of textural sounds [1].

3.1.1. Sample Selection by Matching

Once a set of samples has been analysed and/or labelled to form a related set of audio descriptors it is then possible to perform a matching procedure to select an appropriate sample for use. This involves specifying a set of target values that the sample should match, as well as an optional required level of closeness to the desired values (outside of which a sample will not be deemed to have matched). This is essentially a search for any samples within a specified portion of an N-dimensional euclidean space, in which the dimensions represent each descriptor used in the matching procedure. A wide variety of criteria are possible. If the search is well-constructed and appropriate descriptors employed, the resulting chosen sample will match well to a set of desired perceptual qualities. This approach is inspired the CataRT software by Diemo Schwarz [5].

In *Fluence* no audio samples are specified directly, rather they are always selected using a matching procedure according to the musical needs of the moment. Thus, in a given texture, parameters may be specified so as to match high, noisy sounds, or perhaps short sounds with a high central spectral tendency. Whilst these textual descriptions are relatively vague, in practice a high degree of precision is possible when specifying the exact levels of loudness, spectral centroid, noisiness or any other parameter. Matching is near enough instantaneous, so realtime realisation is possible.

The matching objects in the *AHarker Externals* package are specifically designed to suit the purpose of making sample selections based on the direct audio descriptors, rather than those systems that employ normalised values. This means that the user (and in this case composer) can think in familiar and tangible units such as frequency and dBFS. Each descriptor can be matched in an independent manner with a search (e.g. loudness *below* -10dBFS and with a fundamental frequency *within* an octave of 400Hz). Importantly, the closest N matches may also be returned and the result chosen using a pseudo-random process, thus allowing for the notion that any of the closest matches may be equally valid as a selection. This is notably different to a system where the closest match is always returned, and thus the same input parameters always produces the same choice.

3.1.2. Auto-Segmentation

Audio descriptors can also be used to perform automatic segmentation of audio, breaking it into salient chunks for further organisation. In the main processing patch for *Fractures* it is possible to segment samples using several schemes, including segmentation at amplitude peaks, or at moments of increased spectral change (moments that indicate a change of timbre, or underlying fundamental frequency). After the segmentation is performed, each chunk is analysed with a further set of descriptors that can be used to select appropriate samples over time.

3.2. Gestural Representation and Realisation in *Fluence*

Having described a method of automated sample selection, the question remains as how to shape musical gestures effectively over time.

Fluence employs a custom system of gestural representation and realisation to allow for the creation and control of meaningful control curves. As this system is one that creates generic numerical outputs, it can be used to drive any given parameter over time (be it a filter centre frequency, playback amplitudes, or the distance between

[1] To detail but a few possibilities.

sample onsets). In the piece this is used in a variety of ways, including as the input to a descriptor matching procedure. Thus, short samples (such as recordings of individual key clicks) can be selected in realtime so as to have a perceptible organisation clearly related to a specified abstract shape. The two systems together remove the need for lengthy processes of trial-and-error auditioning, and place the focus entirely on creating the desired gestural shaping.

3.2.1. Design Criteria

The system employed in *Fluence* for dealing with gesture embodies four main design criteria:

Control in Terms of Shape Parameters
: As my primary interest is in controlling the *shape* of musical gestures, the parameters of the gestural system are designed to relate directly and logically to the resultant shape.

Wide Range of Output Shapes / Low Parameter Count
: As a generalised system, it should ideally be possible to create almost any conceivable output, with a relatively small number of parameters. A compact representation is efficient in offering only a few variables for the composer to specify in order to achieve the desired result.

Controllable Variability of Realisation
: From both performance and compositional viewpoints, it is desirable to be able to realise a specified gesture in a number of different ways. In the former case this is to emulate the variation between instrumental performances of the same material. In a compositional context it is a tool for creating variation on a single idea. This criterion demands that low levels of variation maintain the overall contour of the specified shape, whereas higher levels of variability allow more dramatic changes. Thus, the level of variability becomes a parameter of the system.

Hierarchical Stacking of Shapes
: Musical gestures are often complex/multi-layered in their shaping. Instead of requiring the user to chain many curves together to create more elaborate shapes, it makes sense to introduce at least two levels, one for the large scale shape, the other for internal inflections.

3.2.2. System Outline

Full details of the gestural system are outside of the scope of this article. However, a rough outline of important aspects for the system is given below. More detail can be found in the helpfile for the *gesture_maker* object [2].

- Named linear kernel shapes can be specified (see **kernels**)

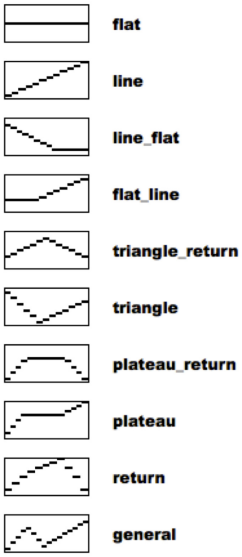

Figure 2. Named Linear Kernel Shapes

- The timing and target values of the segments is specifiable
- Parameter specification is numerical (and reasonably compact)
- Linear segments can be curved to achieve
  - variable steepness of curve
  - s-curvature in a variety of rotations
- All numerical parameters can be specified in one of three ways
  - exactly (the final value)
  - as a numbered band (representing a predetermined range of values) with randomisation
  - as a range of possible bands (again randomised)
- There are two hierarchical layers
  - A main layer (with one kernel)
  - An inflections layer (with multiple kernels - either similar or heterogenous as desired)

variable shows a visual example of three realisations of the same gestures specification. Here the parameters have been specified in such a way as to preserve the general contour and behaviour between realisations, but with noticeable variation in timing and the amplitude of inflections.

3.3. Further Methods of Control

3.3.1. Controlled Randomisation

Tightly bounded and limited pseudo-random processes are a useful tool for creating variability whilst maintaining some sense of predictability. This is already built into the system of gestural realisation for *Fluence*, in terms of the final choice of shape parameters when realising a gesture. The processing patches for *Fractures* also utilise controllable probability distributions to allow a finer level

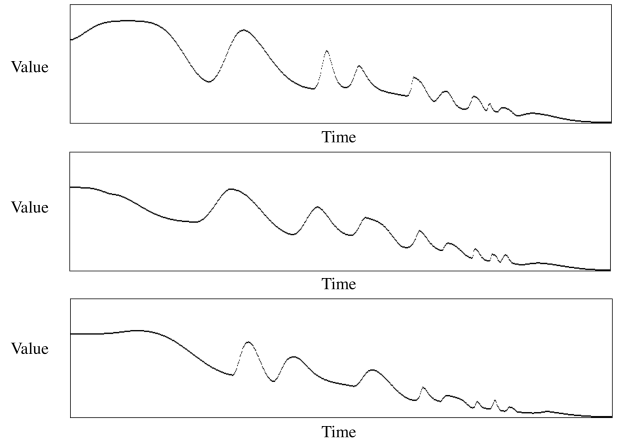

Figure 3. Gestural Variability

of specification than a simple equal probability distribution will allow. The chosen method here is to select between multiple windowed Gaussian distributions with a weighted probability such that probability distributions of varying levels of complexity are possible given a relatively simple interface. This is more clearly understood by looking at **gauss**, in which three windowed Gaussian distributions are combined to produce a final probability distribution (the green curve). Each of these is controllable via a graphical user interface in terms of its mean, deviation (or width), and weighting (or height). More or less complex distributions are possible by adding or removing windowed gaussian curves.

This system allows for engaging complex textures to be created with a fine degree of control. For instance, one patch may be used to granulate a sound such that each grain is filtered at a different frequency. Using bandpass filters of reasonably narrow width (high Q), it is possible to randomise the cutoff frequency parameter using such a distribution. Thus, different frequency areas of the input sound may be emphasised or selected by placing windowed gaussians at the desired frequencies, and these elements can be balanced both spectrally and temporally by the weighting of these curves. This is quite a different effect to simply EQ'ing the output of a granular process.

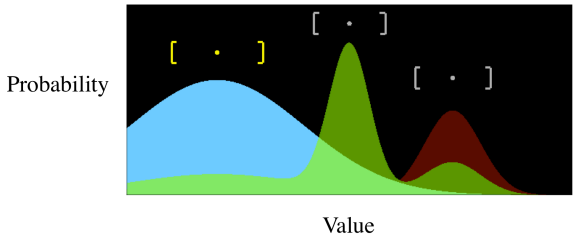

Figure 4. Windowed Gaussian Probability Distribution

3.3.2. Control using Hardware / Audio Analysis

In order to achieve immediacy in the studio, the processing patches for *Fractures* are designed to allow realtime control of all parameters (alongside options for randomisation). Two options for control are available, the first of which is via a *wacom* graphics tablet. This offers 3 dimensions of simultaneous control (x, y and pressure) and maps well to the idea of visual shapes as a metaphor for musical shapes (as in the gestural system in *Fluence*). In addition to this, parameters can be controlled via audio analysis. Here, an audio input (typically from a microphone) is analysed in realtime to yield a range of audio descriptors. These can be mapped to any parameter of the processing. Thus, particularly with more direct mappings (such as mapping input loudness to gain) it is possible to simply vocalise the gestural shapes one wishes to impose on the material, without the need for the composer to deal with an intermediate representation (such as a visual shape, or set of numbers). Here, the idea was very much to create a playable 'instrument' for use in the studio.

4. COMPOSITIONAL IMPLICATIONS

4.1. Focus on Higher-Level Control and Representation

Dealing with the formalisation of sample selection procedures, or gestural shapes, means that time and attention is moved away from details of realisation, and onto higher-level concerns of content and behaviour. Of course, this has both advantages and disadvantages. Consideration of large-scale timings and flow becomes easier. However, if one has a strong desire to place samples in specific time relation to one another, then the extremely fine audio placement and editing possible in a DAW are likely to be more appropriate. This is an aesthetic question, in that it depends on the music one wishes to write, and how much it can be shaped by over-arching processes, rather than very fine manipulation of materials. Importantly, all musical shapes and intentions must be expressed formally. If one cannot derive a set of principles, or criteria for creating a texture or gesture, then automating the process is impossible. In my case, this suits my predisposition towards strong compositional frameworks, but doubtless some composers would find this requirement musically limiting.

4.2. Improved Representation of Musical Ideas

Whilst the systems of gestural representation and descriptor matching presented here are by no means perfect, in terms of offering an easier and more transparent interaction with technology, they do place the composer closer to more musical ways of thinking about sound. Addressing sample content through matching means that it is trivial to perform tasks such as constructing specific harmonic structures (simply search for samples with the desired pitches), or to create a texture of only breath sounds, given

that the samples are appropriately labelled. Changing the entire texture to include only loud, or noisy samples is again, merely a matter of changing the matching criteria, and all the time the descriptions are in terms very close to perceptual, musical qualities.

Likewise, the processes of controlled randomisation and realtime control used in *Fractures* place one as a composer directly in connection with issues of prime musical concern. In order to create a musical gesture that moves jerkily between high and low frequency content it is simply necessary to either map a suitable parameter (perhaps transposition level or filter centre frequency) to the graphics tablet and perform the desired, jerky behaviour in realtime. Alternatively, a suitable probability distribution can be rapidly constructed graphically (assuming that the temporal behaviour is relatively static). In either scenario the logistics of realising musical ideas are significantly reduced.

4.3. Rapid Prototyping and Realisation

Sample matching criteria can be generated very quickly, and audio realisation is immediate. Thus, it becomes trivial to prototype complex textures built from many tens or hundreds of samples. In a DAW, constructing such textures would take a considerable time to produce. With the proposed approaches multiple different textures, or gestures (or variations on these) can be designed and auditioned in a relatively short space of time. This allows for a higher rate of rejection and more time for fine tuning the behaviour of materials to an appropriate degree.

4.4. Implied Notion of Perceptual Equivalence

The notion of perceptual equivalence applies not only to the idea of one or more samples being considered equivalent in a specific context (as discussed earlier), but also to the idea that variability of control shapes can be allowed without altering the fundamental musical structure. In other words, that this variability is simply a matter of flexible performance realisation[2]. This is the case throughout *Fluence* where I consider the gestural specifications in the piece to be analogous to the musical score, in that they determine the important shapes of the piece. Different realisations of each gesture perform the same function within the piece, even though the details vary.

It is necessary to carefully consider which perceptual qualities of a sound will be most important within a particular musical texture or gesture, especially when one considers that , often it is practical to match only a handful of audio descriptors. Whilst controlling some qualities of the sound, others remain completely uncontrolled.

4.5. Appropriate Levels of Variability

Different musical contexts will allow different amounts of variability whilst maintaining perceptual equivalence. Alternatively, there may be times where it is appropriate to re-use gestural specifications so that the variation is perceivable and important to creating musical structures. Thus, the level of variability must be chosen carefully as appropriate to the context and musical intention. For example, strongly directional/gestural material may require tighter levels of control in order to maintain important aspects of the musical shape with each realisation. Textural material may be generated by repeatedly re-using a gestural specification with a much higher level of variability, as a means of creating complexity through a straightforward procedure.

[2] I will refrain from using the word *interpretation*, as this implies a level of musical consideration not built into the system.

4.6. Descriptor Matching Requires Large Samples Sets

When the set of available samples becomes small the audio descriptor space becomes quite sparse. This means that any given part of the descriptor space (the area in which a match might be sought for instance) is more likely to be empty. As it therefore becomes increasingly likely that a given set of target parameters for matching will return either few or no samples, descriptor matching becomes less and less practical as a powerful tool for selecting and structuring sampled materials.. Of course, auditioning fewer samples is conversely more practical, but there may well be a mid-ground in which neither algorithmic matching, nor manual audition is particularly efficient.

It is also necessary to be wary of sparse areas in the descriptor space. Matching in these areas is likely to return many repeated samples, which may be highly undesirable, especially where the intention is to create dynamic gestural or textural materials.

5. REFERENCES

[1] T. Grill, "Constructing high-level perceptual audio descriptors for textural sounds," in *Proceedings of the 9th Sound and Music Computing Conference (SMC 2012), Copenhagen, Denmark*, 2012.

[2] A. Harker, "AHarker Externals," 2011. [Online]. Available: http://alexanderjharker.co.uk/Software.html

[3] ——, "Fluence," Digital Release, 2011. [Online]. Available: http://ergodos.bandcamp.com/album/fluence

[4] G. Peeters, "A Large Set of Audio Features for Sound Description (Similarity and Classification) in the CUIDADO Project," IRCAM, Technical Report, 2004.

[5] D. Schwarz, "Catart (version 1.2.3 for ftm 2.6)," 2010. [Online]. Available: http://imtr.ircam.fr/imtr/CataRT

[6] B. Stroustrup, *The C++ Programming Language*. Addison-Wesley, 2010.