AN APPROACH TO REAL-TIME CONTENT-BASED AUDIO SYNTHESIS

Jan Erik Hagevold

Submitted in partial fulfillment of the requirements for the
Master of Music in Music Technology
in the Department of Music and Performing Arts Professions
in The Steinhardt School
New York University
Advisor: Dr. Juan P. Bello
[DATE:2008/12/09]

"Research into sound synthesis is governed by aesthetic goals as much as by scientific curiosity. Some of the most interesting synthesis techniques have resulted from applied practice, rather than from formal theory. Sound design requires taste and skill and at the experimentation stage, musical intuition is the primary guide" (Roads, 2004)

Acknowledgements

This thesis was written while studying at the graduate Music Technology program at New York University. I would like to thank my advisor, Juan P. Bello, for his continuing support in my studies, providing valuable advice, guidance, and enthusiasm. His involvement goes beyond the work of this thesis. Besides conducting excellent teaching, he has provided opportunities to get involved with the research community, and encouraged collaboration between students in the program. I would further like to thank Agnieszka Roginska for advice in the process. The colloquy class designed to prepare for writing this document and preparing for the presentation was a great experience. Her teaching has also been invaluable to me during the course of the program. A special thanks goes to the members of the Music Technology research group, which I have been grateful to be part of this past year; Andy Sarroff, Zeeshan Lakhani, Tae-Min Cho, Aron Glennon, Peter McCullough. Thanks to the director of the Music Technology program, Kenneth Peacock, and everyone else that has contributed to my development during my studies at NYU; Anton Vishio, Robert Rowe, Jean-Luc Cohen, Nick Didkovsky, Barry Greenhut, Leigh Newsome, Anna Stephan-Robinson.

Finally, I would like to thank family and friends.

TABLE OF CONTENTS

Abstract

This thesis explores an approach to interactive content-based audio synthesis by using granulation techniques of recorded audio in combination with analysis. By obtaining a perceptually meaningful *representation* for short audio segments, these may be organized to reflect different *facets* of the source material that they are derived from. A visual representation is suggested that makes way for navigating the segmented audio with control over the acoustic result. The implementation is meant to function as a tool that allows for intuitive exploration and manipulation of recorded sound.

The thesis is inspired by pervious work in content-based sound synthesis, but focuses on the mapping of complex features, not necessarily representable on a subjective scale, to a navigable space. The acoustic parameter used in the representation of audio segments is *timbre.* The mapping is based on a distance measure for high-dimensional multivariate data, which is further arranged in a 2-dimensional space using a force-directed model for multidimensional scaling.

1.0 Introduction

1.1 Motivation

In recent years, much attention has been attracted to the analysis of musical signals in order to reveal information related to acoustic or musical parameters. Some of the applications are automatic transcription of music, genre recognition, visualization, machine listening, playlist generation, and general content-based management systems for large music collections. In these areas, a range of computational methods from diverse research fields have been applied and developed. My motivation has been to exploit some of the available methods to develop a tool that can be used in creative musical settings. The proposed implementation is an extension of granulation, a form of granular synthesis that allows one to harness the richness of detail in recorded sound. This particular synthesis technique requires a large amount of control parameters to shape the final results. Knowledge of the temporal evolution of the selected audio material is often necessary to obtain a particular composition of sound, unless one chooses to go with a random process or prefers exploring the content on a trial and error basis. Implementations of granular synthesis usually include a range of parameters that have an impact on the acoustic result. Amongst these are grain durations, grain density, manipulation of playback rate and start position, and choice of source material. Still, this is to some extent an abstract way of working with recorded sound. It is perhaps reflected in such terms as *time-scrambling*, or *cloud generation,* often used to describe the various implementations. By analyzing the source material and providing a useful organization of the resulting audio segments, another dimension of control becomes available that may be useful in certain applications. The contribution of this work is to define an approach to content-based sound synthesis that is founded on a combination of well defined computational models.

1.2 Overview of this thesis

This report is structured into four parts. In this introductory section, I have already discussed my motivation and goals for the work, and I will also discuss the background of granular synthesis, some perceptual concerns, and related works. The following section on computational models will discuss in detail the methods used in the analysis and in producing the layout of the navigable space featured in the final application. The third section outlines the overall structure of the application, the parameters used for analysis, and the choice of controls for interaction. The final section contains a discussion about some of the issues that I have encountered in the process, along with suggested solutions and future work.

1.3 Perception

1.3.1 What is timbre?

In obtaining a perceptually meaningful representation of audio segments, one would naturally be interested in capturing the essence of *how it sounds.* The acoustic parameter dealt with in this thesis is 'timbre'. Timbre is described by the American Standards Association (A.S.A, 1994) as "[...] that attribute of sensation in terms of which a listener can judge that two sounds having the same loudness and pitch are dissimilar". In other words, timbre is basically everything else besides pitch and loudness. It is associated with the quality, or color, of a sound, but has no real subjective rating scale in terms of "high" or "low" values. Timbre depends on the spectral content and how it changes over time, but it is also affected by the temporal envelope, transients, and the overall brightness of a sound. One approach is to define timbre as a complex high-level feature that may be

explained through a battery of low-level audio descriptors (Jehan, 2005). These may be organized into various categories which include, temporal descriptors, energy descriptors, spectral descriptors, harmonic descriptors, and perceptual descriptors. Martin (Martin, 1999) points out that timbre is a "nebulous word for a perceptual quality in addition to loudness, pitch, and duration".

1.3.2 The spectrogram

A common approach in investigating timbre is to look at the frequency content of a sound. This can be achieved by means of a discrete Fourier transform (DFT). The DFT analyzes a signal of finite length and determines the sinusoidal and phase content. The DFT for signal *x(n)* can be calculated as:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-\frac{j2\pi nk}{N}} \quad k = 0,1,...,N-1 \tag{1.1}$$

The result, X(k), is complex-valued. The magnitude spectrum, |X(*k*)|, can be extracted as follows:

$$|X(k)| = \sqrt{X_R^2(k) + X_I^2(k)} \tag{1.2}$$

Where X_R is the real part and X_I is the imaginary part. To investigate the frequency content of a signal over time, a short-time Fourier transform (STFT) is used. The STFT calculates the DFT on short segments of a signal at set time intervals. The frequency resolution, Δf, of a DFT is

determined by the sampling rate, *fs,* and the length of the signal, *N,* specifically $\Delta f = fs/N$. In order to increase frequency resolution, one needs to increase *N*. This, however, results in loss of temporal resolution. The time intervals, or the hop size, used by the STFT is often set so that the segments are overlapping, to compensate for temporal loss. Alternatively, the segments can be zero-padded to reach a desired *N* value. Zero-padding translates to interpolation in the frequency domain. A final thing to mention concerning the DFT is that in practice, when segmenting a signal, there will be sharp changes at the signal boundaries that introduce noise. This noise appears as a smearing of the spectrum and is known as *spectral leakage*. This can be avoided by performing *f0*-synchronous analysis, or more commonly, applying a window function to the signal that minimizes the effects of spectral leakage. Most window functions smoothly reduce the signal to zero at the signal boundaries. The transform of a window function will reveal its side effects, as multiplication in the time-domain can be translated to convolution in the frequency-domain. In practical applications, a fast version of the DFT is used, namely the fast Fourier transform (FFT). For the FFT algorithm to run optimally, the chosen number of samples, *N,* should be a number that is a power of two.

Although the spectrogram produced by means of an STFT is very useful in investigating timbral features, Tzanetakis points out some problems with this representation in relation to computer audition; "1) a lot of information contained in the spectrum is not important 2) machine learning algorithms work better with feature vectors of small dimensionality that are as informative as possible" (Tzanetakis, 2002). Considering these observations, one can conclude that a smaller set of features approximating the spectral shape is therefore a better option in many applications. The approach taken in this implementation will be discussed in detail in section 2.0 of this paper.

1.4 Granular Synthesis

1.4.1 History

Roads gives a thorough survey of the history and origins of particle-based synthesis techniques in his book *Microsounds* (Roads, 2004). He puts the matter in context with the debate concerning the "wave" versus "particle" views in optics and acoustics which were central in forming what we now call modern science. He points out that "the notion that apparently continuous phenomena can be subdivided into particles can be traced back to the atomistic philosophers of Greek antiquity". The science of acoustics has to a large extent been dominated by the wave theory of sound. However, Roads makes reference back to the well known theoretical physicist Albert Einstein and his ideas concerning the acoustical quantum in relation to a threshold in human hearing. This idea was further development by Hungarian scientist Dennis Gabor, which led to an understanding of these theories as somewhat complimentary.

In the field of musical sound synthesis, the particle based approach was introduced through Gabor's initial experiments on changing the pitch and duration of sounds using granulation techniques in the 1940s. Previous experiments in sound synthesis were mainly 'wave-oriented', but Gabor's research opened up a different approach, one that has been continuously developed by acousticians, physicists, mathematicians, and composers of music. Iannis Xenakis coined the term 'grains of sounds' and was the first composer to develop a compositional theory using this taxonomy (Roads, 2004). A well known quote from his book *Formalized Music* explains his idea that "All sound is an integration of grains, of elementary sonic particles, of sonic quanta...All sound, even continuous musical variation, is conceived as an assemblage of a large number of elementary sounds adequately disposed in time" (Xenakis, 1971). Other pioneering composers, such as Stockhausen and Koenig, also applied temporal theories and the notion of acoustic quantum into

music, initiating debate as well as stimulating new compositional ideas. The early approaches to micro-sound composition were realized by manipulating tape recordings, or even explored through orchestral and instrumental techniques. With the introduction of digital technology in the 1970s, experiments with granular synthesis became more feasible by allowing exact control over sound events on a small timescale. A number of implementations have been realized since then, and today, granular synthesis is found and used in numerous compositions and has made its way into media and music production environments.

1.4.2 Effects of granular synthesis

Granular synthesis can be described as a technique based on streaming brief segments of audio in order to build a sonic texture, or sound objects. The segments are referred to as *grains*, and they consist of a waveform shaped by an amplitude envelope, see Figure 1.1. The duration, waveform, envelope, density, and synchronicity of the stream greatly affect both our perception of individual grains, as well as the spectrum of the overall cloud texture.

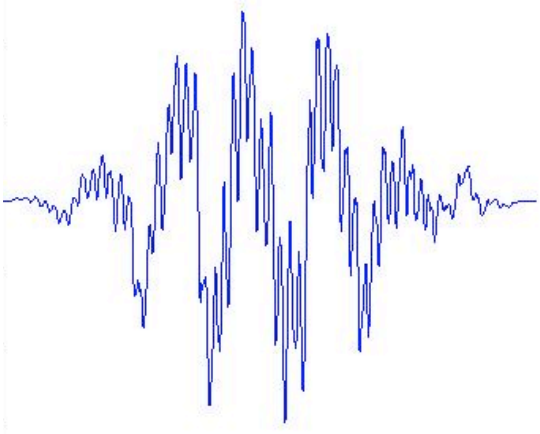

Figure 1.1 Example of a 'grain' of sound

Roads talks about micro-temporal perception with references to literature, i.e. (Buser, 1992; Roads, 1996), giving an overview of our perceptual limits in this domain. Although pointing out that "... human hearing mechanisms, however, intertwine with brain functions, cognition, and emotion, and are not completely understood", Roads brings forward a number of scientific observations relating to intensity, fusion/fission, silence, pitch perception, auditory acuity, etc. I will not discuss these aspects in detail, but in this context, I will point out some matters that have been important concerns in the design of this project.

Our limitations in regard to perceiving pitch and timbre are related to the duration of events. Longer duration allows for more detail in the grain waveform to be perceived. Although the limit for timbre perception is estimated to some hundred microseconds, the true impression of the waveform's timbre in the context of granulation synthesis, that is, the actual content of a grain from some particular sound, emerges fully only after 40-50 ms (Truax, 1988). Truax indicates a duration of 50 ms to be a rough dividing line between different psychoacoustic effects. While around and above this threshold, the timbral qualities of the waveform are perceivable, shorter grains are dominated by audio rate fusion and broadband effects.

Recognizing pitch and identifying the source of the waveform may be desirable particularly in exploratory sound synthesis. Pitch perception in regards to the duration of sound events is frequency dependent, (Roads, 2004) with reference to studies by Meyer-Eppler, in such terms that lower pitched tones require longer durations for recognition. This is a general property that also applies to perception of different timbres. An extreme example is when the duration of an audio segment is shorter than a period of the waveform, which results in misrepresentation. Instrument, or sound source, recognition is more ambiguous and varies depending on the source. Longer events generally provide more information, hence more specific clues to the origins of sound. See i.e. (Martin, 1999) for details on sound-source recognition.

The shape of the envelope applied to the waveform also affects the resulting sound. Relating back to the discussion of timbre, the envelope really defines the overall attack and decay of individual grains. In the context of dense granular streams, one is also concerned with the modulatory effects of the envelope function. For synchronous streams, the envelope becomes a periodic signal which has the effect of amplitude modulation. The waveform can be seen as the carrier signal, being modulated by the envelope function. As a result, a number of sidebands can be observed in the final spectrum. The number, amplitude weights, and spacing of sidebands are directly related to the shape and duration of the envelope function, as well as to the rate of the granular stream. In the case of an asynchronous granular stream, that is, when the time gap between the triggering of individual grains is irregular, the sideband effect is randomized and results in more of a widening of the spectrum.

1.5 Related work

This thesis is heavily inspired by Schwarz's concatenative real-time synthesis system *CataRT,* (Schwarz, 2006). CataRT uses a large corpus of segmented and descriptor-analyzed sounds. The model is a multidimensional space defined by descriptors. Sound units are positioned in the space according to their characteristics. The application allows for interactive exploration of the model through a graphical user interface, shown in Figure 1.2. Control is also possible by means of a target sequencer, or a target audio file, and even live input that is analyzed in real-time. The application features a 2-dimensional visualization space where grains are projected along the x and y axis according to two selectable descriptors.

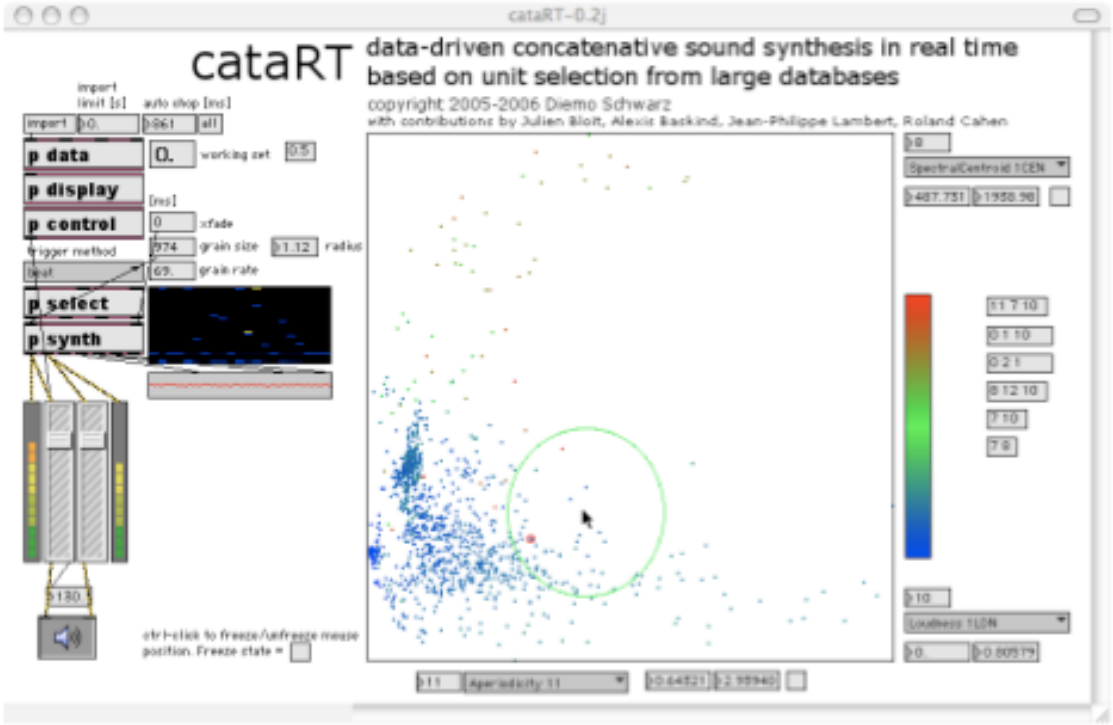

Figure 1.2: The CataRT interface

The particular descriptors used are either pre-calculated MPEG-7 low-level descriptors, see (Schwarz, 2004) for an overview, or descriptors calculated in the application. These include fundamental frequency, aperiodicity, loudness, and a wide variety of spectral features.

The descriptors are calculated at the FFT-frame rate over each sound unit. Each time-varying feature is represented by fixed scalar values, namely the mean, variance, slope, curve, min, max, and range. Unit selection is achieved by evaluating the square Mahanalobis distance between a target position in the descriptor space and the units of the corpus. The efficiency of the selection operation is improved by a constructed search tree. The tree is created by repeatedly splitting the descriptor space along the hyper-plane perpendicular to the principal components vector until only a

few units are left at the leaf nodes (see (D'haes et al., 2003) for details). The application offers many features for interaction and playback. Along with different interpretations of mouse gestures, control is made possible through MIDI and other input devices for advanced gestural control. The possible applications mentioned include explorative granular synthesis, gesture-controlled synthesis, data-driven drumbox, and expressive speech synthesis.

Another source of inspiration, leading to the choice of methods used in creating the layout of the navigable space was found in (Gasser et al., 2008). *Streamcatcher* was presented at the International Conference on Music Information Retrieval 2008. The application explores a content-based visualization of online audio streams and takes the form of "a simple interactive visualization approach that incorporates the user's musical vocabulary into the definition of semantic spaces for music similarity judgement." The system uses a number of *prototype instances* consisting of different music clips from the user's own music library. These define the visualization space by means of a content-based music similarity measure and a user-feedback based *concept labeling*. The prototype instances are processed offline and are represented by the distribution of Mel Frequency Cepstral Coefficients (MFCCs). The similarity measure is calculated by the symmetric Kullback-Leibler divergence. Music clips are positioned in a 2-dimensional space by means of a force-based multidimensional scaling technique, attempting to preserve the high dimensional distances in the lower dimensional space. An online stream is analyzed in real-time and placed nearby similar prototypes. Computation of music similarity between the online audio stream and the prototypes, and the visualization algorithm is optimized for real-time performance. The process is visualized by an attractive animation. The interface of the *Streamcatcher* application can be seen in Figure 1.3.

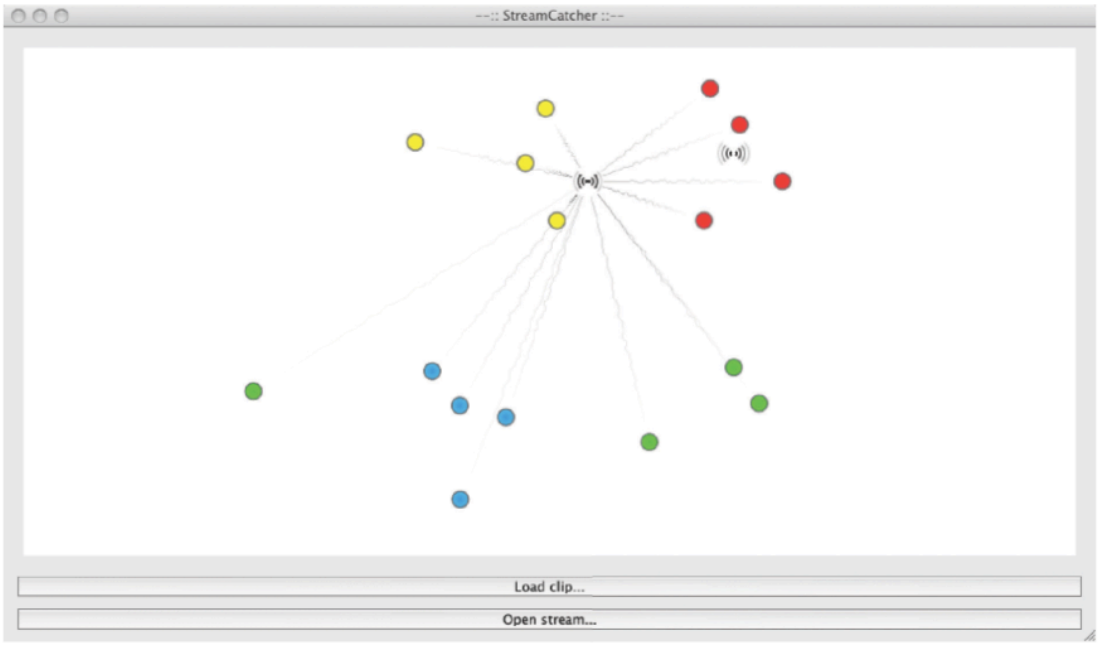

Figure 1.3: The Streamcatcher interface

The approach described in this thesis is really a combination of these two works. The idea of extending granular synthesis with content-based unit selection, as described in (Schwarz, 2006), is combined with the methods for feature extraction and similarity calculation, and aspects of the mapping process described in (Gasser et al., 2008). The goal has been to create a navigable layout where entities are positioned based on relational data derived from a statistical method of multivariate data analysis. The data characterize timbre and can not easily be projected along an axis representing high and low values.

2.0 Computational Models

2.1 Feature Extraction

2.1.2 Mel-Frequency Cepstral Coefficients

In describing the audio content, the concern has been to capture the timbral aspects of the sounds. Mel Frequency Cepstral Coefficients (MFCC) has been widely used in speech recognition, see i.e. (Oppenheim, 1969; Rabiner & Juang, 1993), and has been shown to provide good results in modeling music and audio signals, see i.e. (Foote, 1997; Logan, 2000). The coefficients represent a rough approximation of the spectral shape of a given magnitude spectrum where both frequency and amplitude are scaled logarithmically. The representation hereby conveys general characteristics of the human hearing mechanism. The process of calculating MFCCs can be summarized as in Figure 2.1.

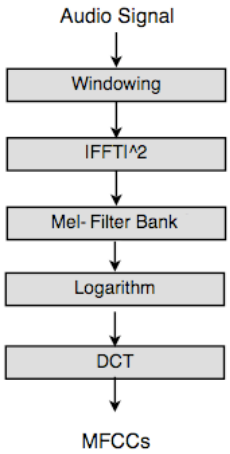

Figure 2.1: Flowchart of the MFCC calculation process

In the conversion process, the magnitude spectrum is warped according to the *Mel* frequency scale, defined in (2.1). The scale takes into account the nonlinear nature of how we perceive pitch. It roughly coincides with the standard frequency scale up to about 500 Hz. After this point, increasingly larger intervals are perceived as being equal increments in pitch. In fact, the word *Mel* is derived from the root of the word melody (Stevens & Volkmann, 1937).

$$mel(f) = 1127.01048 \log\left(1 + \frac{f}{700}\right) \tag{2.1}$$

Where *f* is frequency in hertz, and *mel* is the associated *Mel* frequency. The implementation provided in (Pampalk, 2006a), creates a selected number of overlapping triangular filters spaced linearly on the *Mel* frequency scale. Figure 2.0 shows the distribution of filters along linearly spaced FFT bins. The spacing is arranged so that each triangular filter starts at the center frequency of the preceding one. The height of each triangular filter is calculated so that the sum of weights, or the integral, is equal for all channels.

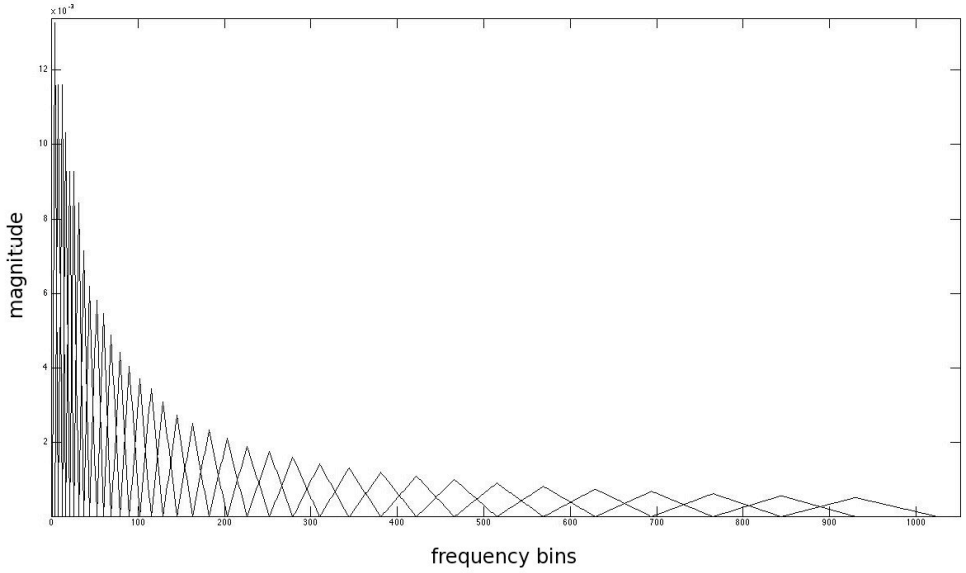

Figure 2.2: Mel filter bank

Applying the filters to the magnitude spectrum results in a certain emphasis on lower frequencies which are considered perceptually more meaningful in terms of discerning the timbral aspect of sounds. Our auditory system is limited in its ability to resolve inputs whose frequency difference is smaller than what is called a *critical band.* The increasing width of the filters corresponds with the characteristics of critical bands in human hearing. Applying the *Mel* filters can be seen as coarsely imitating this feature, while the further conversion into decibel values imitates our perception of loudness.

The final step is applying the discrete cosine transform (DCT) to the *Mel* power spectrum. The DCT compresses the number of frequency bands into a defined number of coefficients. These coefficients represents the weights of a number of kernels defined in a transformation matrix (see Figure 2.3). The kernels are cosine functions with frequencies increasing by half integers, 0, 0.5, 1, 1.5, etc. The *Mel* power spectrum is represented by a combination of these cosine functions. The first few of the coefficients serve to represent the main features of the spectrum. The DCT is a good fit for many types of signals, as one expects that "the higher frequency coefficients are small in magnitude and can be more crudely quantized than the low frequency coefficients" (Blinn, 1993). The compression results in a simplification and smoothing of the spectrum.

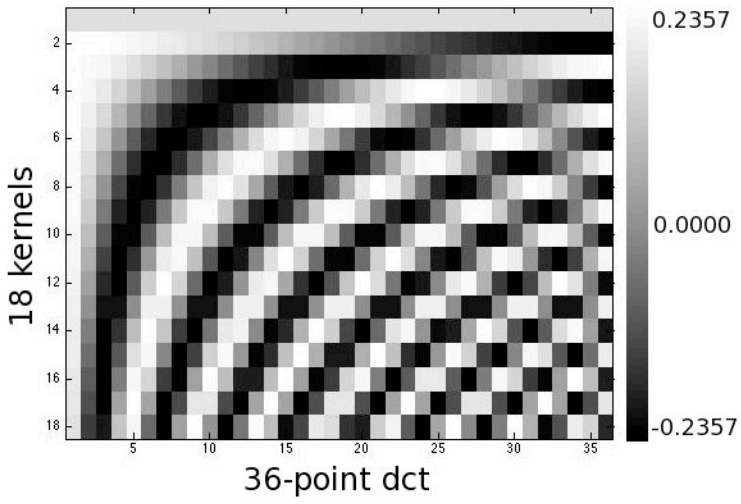

Figure 2.3: DCT transformation matrix

Clearly the MFCC feature coefficients offer a significant reduction in the amount of data needed to represent the spectral features. As an example, imagine a case where the STFT frame length is set to 1024 samples, hop-size is set to 512 samples, and the segment length is 366.9 ms (16184 samples), so that each segments is represented by 29 overlapping frames. By using only 18 coefficients, the amount of data is reduced by a factor of ~56.8:1, from 1024x29 to 18x29. The resulting feature matrix is properly called a *multivariate time series*.

From the discussions above, it might be more appropriate to talk about spectral shape features rather than using the term timbre, as the representation does not explicitly take characteristics such as attack and decay into account. However, previous studies in speech processing and music information retrieval (see citations above) support using spectral shape as a viable means to characterize timbre. MFCCs are widely used and achieve this approximately and efficiently. Pampalk puts it like this: “Spectral similarity relates to the timbre of a piece of music as much as color histograms relate to the color composition of a painting” (Pampalk, 2006a).

An alternative to using the *Mel* frequency scale is the Bark scale, or even uniformly spaced filters. In a comparative study focused on speech recognition (Shannon & Paliwal, 2003), these alternatives were found to yield similar results. When the goal is to characterize the timbre of sounds, there might be benefits of using a logarithmic scale, especially due to the better resolution at lower frequencies. Logan (Logan, 2000) concludes that in the context of speech/music discrimination, use of the Mel frequency scale is at least not harmful. She suggested further experimentation to verify that this is the optimal solution in modeling music spectra in the general case.

2.2 Similarity

2.2.1 Single Gaussian (G1)

Calculating similarity between different multivariate time series may be a computationally expensive task. Most approaches summarize the data by clustering the frames and then calculating a distance measure between the cluster models. These models are statistical in that they represent a set of probability distributions on the sample space.

This implementation uses a single Gaussian (G1) representation of each segment (Pampalk, 2006a). The Gaussian is represented by a vector, μ, containing the mean values (2.2) for each frame, and the full covariance matrix Σ (2.3).

$$\mu = \frac{1}{N}\sum_{n=1}^{N} x_n \tag{2.2}$$

$$\Sigma = \frac{1}{N\text{-}1}\sum_{n=1}^{N} (x_n - \mu)(x_n - \mu)^T \tag{2.3}$$

Where N represents the number of observations in the feature set x. The superscript ^{T} denotes the transpose of a vector.

2.2.2 Kullback-Leibler Divergence

The distance measure between different Gaussian models is calculated by the *symmetric* Kullback-Leibler divergence. A detailed description can be found in (Moreno et al., 2003), and a

Matlab[1] implementation is provided in (Pampalk, 2006a). The symmetric Kullback-Leibler divergence is defined in (2.4). The analytical solution, shown in (2.5), is used in the implementation.

$$D(p(x\mid\theta_i), p(x\mid\theta_j)) = \int_{-\infty}^{\infty} p(x\mid\theta_i)\log\left(\frac{p(x\mid\theta_i)}{p(x\mid\theta_j)}\right)\delta x + \int_{-\infty}^{\infty} p(x\mid\theta_j)\log\left(\frac{p(x\mid\theta_j)}{p(x\mid\theta_i)}\right)\delta x \tag{2.4}$$

$$D(p(x\mid\theta_i), p(x\mid\theta_j)) = tr\left(\Sigma_i\,\Sigma_j^{-1}\right) + tr\left(\Sigma_j\,\Sigma_i^{-1}\right) + tr\left(\left(\Sigma_i^{-1}+\Sigma_j^{-1}\right)\left(\mu_i-\mu_j\right)\left(\mu_i-\mu_j\right)^T\right) \tag{2.5}$$

Where the θ terms represents the mean vector, μ,and the full covariance matrix, Σ, of the statistical model $p(x|\theta)$ of a data set. The i and the j terms indicate the two different Gaussians, which the divergence, denoted by $D(p(x|\theta_i), p(x|\theta_j))$, is calculated between. The *tr* term in (2.5) represents the trace of the resulting matrices, or the sum along the diagonal. The term Σ_i, represents the full covariance matrix, and the inverse of the covariance matrix is denoted by $\Sigma^{-1}i$. The mean vectors are represented by μ_i, and the superscript ^{T} represents the transpose.

This solution is symmetric in the sense that $D_{ij} = D_{ji}$, meaning that the divergence is equal regardless of which of the two gaussians, i or j, is compared against the other. The self similarity is nonzero ($D_{ii} \neq 0$), see (Flexer, et al., 2008) for details. The distances calculated between each segment are collected in a distance matrix. To save computation time, the calculations were only performed 'one way', that is, only the upper triangle of the matrix was calculated and simply

[1] http://www.mathworks.com/

mirrored over the diagonal. An additional scaling of the distances, shown in (2.6), is also applied. The diagonal representing the self similarity is simply set to zero for convenience. The scaling factor affects the overall distance distribution, in terms of changing "the ratio between the maximum value and the median value" (Pampalk, 2006a). The distribution of distances has a great effect on the resulting layout discussed in the next section. The distance matrix is finally normalized to values between 0 and 1.

$$D_{ij} = e^{\left(\frac{-1}{fact \cdot D_{ij}}\right)} \qquad (2.6)$$

Where *fact* represents the scaling factor. This method for similarity measurement was chosen, as it has shown to provide good results in the context of audio-based similarity and retrieval tasks based on MFCC data, e.g. (Pampalk, 2006b).

2.3 Multidimensional scaling

2.3.1 Force-directed model

Multidimensional scaling in the context of this work refers to the technique for the mapping of high-dimensional data onto a low-dimensional space. In other words, this technique attempts to reduce the dimensionality of the data while maintaining an optimal layout. Ideal solutions should lower the error in terms of minimizing the difference between the distances in the low-dimensional and high-dimensional space, preserving the similarity relationships of the data. The particular algorithm used was proposed in (Chalmers, 1996). It builds upon force-directed methods using a physical analogy to drawing graphs. The data set may be referred to as *proximity data*, within which one can assess the similarity between entities. In this case, the data set is represented by the distance matrix. The similarity measures are seen as forces acting between entities. A common terminology used in graph drawing applications is to refer to entities as *vertices* and to their relationships as *edges*. Eades, (Eades, 1984), described the force-directed method in terms of steel rings connected by springs. A mechanical system where steel rings represent vertices and springs represent the edges, see Figure 2.4. The length of the relaxed strings reflects the high-dimensional distance between entities. The positions of entities, or steel rings, is initially random, and when the model is released, the attractive and repulsive forces of the strings will move entities into an equilibrium state.

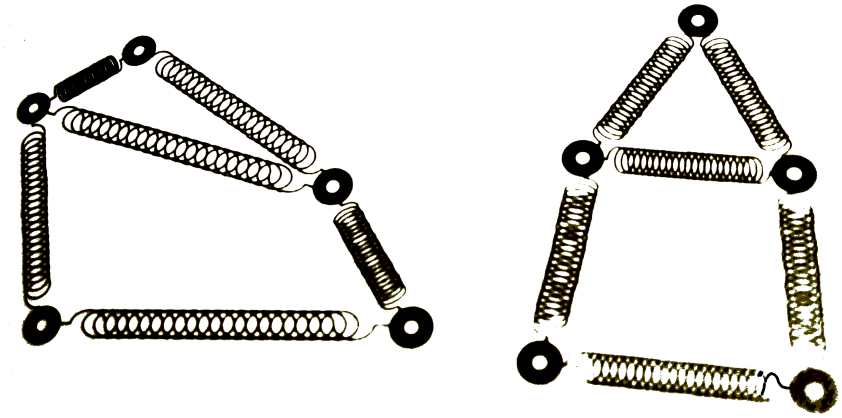

Figure 2.4: Spring Model (from Di Battista et al. 1998)

The algorithm is iterative. It maintains three properties for each entity: *position*, *force, and velocity.* On each iteration, positions are updated by calculating a force vector, *F,* acting upon each entity, which is added to a velocity vector that is stored. The magnitude of the force between individual objects is proportional to the absolute value of the difference between the high-dimensional, $d_{ij,}$ and low-dimensional, g_{ij}, distance. A loss-function can be defined as the overall *stress* exerted on the model (2.6) (Di Battista, et al., 1998). The problem with the basic algorithm is the increased number of required operations with the size of the data set to produce a stable layout.

$$\text{Stress} = \frac{\sum_{i<j} (d_{ij} - g_{ij})^2}{\sum_{i<j} g_{ij}^2} \qquad (2.6)$$

Chalmer's algorithm from 1996 improves efficiency by means of stochastic sampling and neighboring sets. Rather than performing all possible pairwise force calculations, two distinct sets

are used for each entity. A 'neighbor set', V_i, is defined as a list of entities which are found to have the lowest high-dimensional distance so far in relation to the current entity. The set is initially empty with a stored threshold value that will qualify entities from a 'stochastic set', S_i, to be inserted, or as it is constructed, replace entities in the set V_i. The stochastic set is recreated on every iteration. It consists of randomly selected entities that are not members of the neighbor set. Each member is tested to see if they qualify as members of the neighbor set, that is, if the high-dimensional distance is lower than the threshold value, or lower than that of any current member of the neighbor set. The force calculation for each iteration is only performed between the current entity and the members of the two sets (2.7). This reduces the number of calculations from *N(N-1)* to *N(Smax + Vmax)*, where *Smax* is the size of the stochastic set, and *Vmax* is the size of the neighbor set, making the computational cost linear with respect to the size of the data set.

$$F(i) = \sum_{v \in V_i} F_{iv} + \sum_{s \in S_i} F_{is} \qquad (2.7)$$

A additional damping factor is applied to the velocity of each entity to stabilize the system, and the stochastic sampling adds 'jitter', which helps the breaking out of local minima.
In this implementation the values *Vmax*=4 and *Smax*=6 has been used. Typical values indicated in (Chalmers, 1996) were *Vmax=5* and *Smax=10.* Chalmers also experimented with higher values. He suggested that layout quality is still good with lower values "because of indirect linkage via neighbors". He also found that using purely random sampling, *Vmax=0* and *Smax=15,* provided good initial speed, leading to an idea of a hybrid method bringing in neighbor usage only after the initial stage for refinement. I found no significant difference in layout quality using *Vmax=5* and

Smax=10, compared with using *Vmax=4* and *Smax=6,* so the values were cut back for efficiency. As the value that indicates the *stress* of the model varies greatly with data sets of different size and character, I left it for the user to decide when to stop the iterative process based on providing an animated display of the progress. Alternatives on deciding when to terminate are to decide a threshold for the stress measurement in (2.6), or to base it on little variance over successive stress measurements.

Checking for either does add to the computational cost of the algorithm. In (Chalmers, 2003), an algorithm described as a hybrid approach based on stochastic sampling, interpolation and spring models was presented, allowing efficient visualization of very large data sets. In the same paper, two arguments are made for adopting a spring model for multidimensional scaling over principal component analysis (PCA), a statistical dimension reduction technique. First, in PCA, a linear combination of the original data is used in deriving the lower dimensional representation. This results in “an additional constraint on the system and renders it less free to converge to an optimal solution with regard to stress”. It is especially an issue with data exhibiting nonlinear clusters. The other benefit of the spring model approach, is that the iterative nature of the algorithm allows one to observe the process and easily make adjustments that would alter the resulting layout. Additional data may also be appended to a completed model without having to rerun the process in its entirety.

2.4 Selection

2.4.1 K-Dimensional Tree

The result of the multidimensional scaling algorithm is a set of points scattered in a two-dimensional space. Each point represents an entity, or a grain of sound. The goal is to be able to efficiently make selections in this space in real-time. A naive way to go about this would be to scan through the entire data set to find the nearest-neighbor to a target selection point. This, however, is not very efficient. The number of operations, *O*, including distance calculation and comparison, per selection point is equal to the size, *N*, of the data set. The time complexity of the algorithm is thus *O(N)*. The data structure described in this section offers a way to arrange a finite data set in an intelligent manner that in most cases makes it unnecessary to calculate the distance for every point.

A k-dimensional tree is an extension of a binary search tree, working with k-dimensional data. It allows for fast searching and nearest-neighbor queries. An informative tutorial can be found in (Moore, 1991).

In a k-dimensional tree, a data set is represented as nodes (see Figure 2.5 for a simple example), where each node represents a single entity of the set. In this case we are working with 2-dimensions. The tree is constructed by first finding the median point, and the space is split into two subspaces along one dimension.

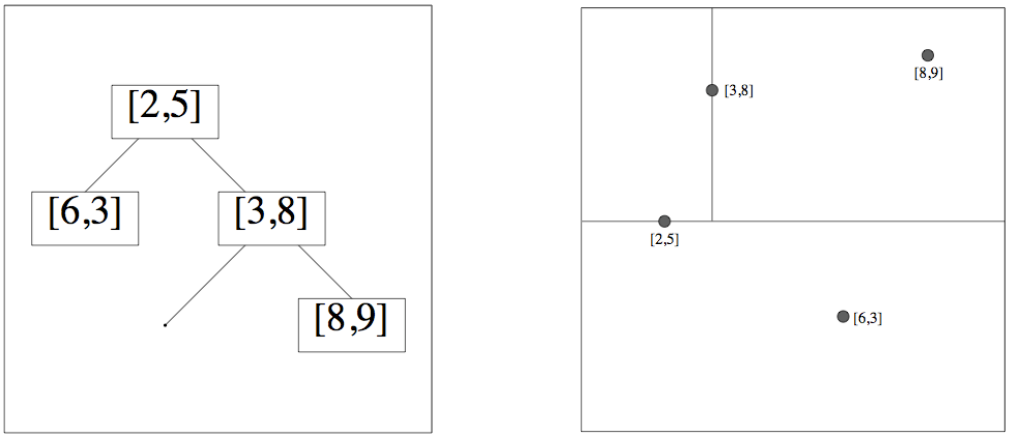

Figure 2.5: A simple tree structure and its representation in the xy plane (from Moore, 1991)

In choosing the splitting plane, Moore explains two desirable attributes; the tree should be balanced, and the shapes of the subspaces corresponding to leaf nodes should be equally proportioned. This is recommended based on the fact that unbalanced trees could lead to accessing behavior more similar to *O(N),* rendering the tree structure almost useless. Also having equally proportioned (more square) subregions maximizes the chances for discarding sections during search. These attributes may be more or less forced by i.e. selecting the splitting dimension in regard to the amount of variance (PCA), or by looking at the longest dimension of the current subspace, emphasizing more squareness.

The median point becomes the *root node* of the tree, and the resulting subspaces are referred to as the *left* and *right* subtrees. Points within each space will belong to the consequent subtree based on whether its value perpendicular to the split is larger or smaller than that of the median point. The splitting process continues until a node is found not to have any *children*.

In performing a nearest neighbor search, the *leaf nodes* of the tree are first examined. The leaf node of the subspace containing the target point is the first approximation. This node might not

necessarily be the nearest neighbor, but it has been determined that the nearest neighbor must lie within a circle centered on the target point and passing through this node. The following step is to back up to the *parent* of the leaf node to check whether there might be a closer node in this node's other child. This is only a possibility if the circle intersects with the subspace occupied by the parent's other child. In the case where this is true, the child is recursively searched. If it does not intersect, the algorithm may move on to the next parent to check whether the circle intersects its space. Range search is possible by changing the nearest neighbor search to make sure that the initial desired distance is not changed as closer points are discovered, and that all discovered points within this distance are stored.

Moore points out that calculating the expected number of node inspections is difficult, as the analysis really depends both on the expected distribution of points in the kd-tree as well as the positions of a target points used to test the algorithm. "It is clear that on average, at least *O(log N)* inspections are necessary, because any nearest neighbor search requires traversal to at least one leaf of the tree. It is also clear that no more than *N* nodes are searched: the algorithm visits each node at most once", he remarks. His research focussed on empirical behavior indicates that the speed of search varies marginally with tree size, very quickly with the dimensionality of the distribution of the data points, and linearly with the number of components in the kd-tree's domain given a fixed distribution dimension. From this, it is acceptable to draw the conclusion that a real-time search in 2-dimensional space should be practical.

3.0 Implementation

The implementation has been executed using SuperCollider[2], an environment and programming language for real-time sound synthesis and algorithmic composition, in combination with Octave[3], a high-level language with a primary focus on numerical computations. SuperCollider is built around a client-server model, meaning that it is actually two separate applications. Communication is done using Open Sound Control (OSC), see (Wright, et al., 1997). SuperCollider was originally written by James McCarthy, and is now an open source project maintained and developed by a large group of people from various backgrounds. The implementation is packaged as a 'stand-alone' SuperCollider application in the form of a Mac OS X bundle. A binary version of GNU Octave is included in the bundle. OctaveSC[4], a SuperCollider class implementing FIFO pipe-based binary communication, is used for function calls and data transfer between the SuperCollider client and Octave. The overall application is controlled by the SuperCollider client-side application. The client handles the graphical user interface, opens the connection and communicates with Octave, and sends messages to the SuperCollider server when audio processing is required. An overview of the application can be seen in Figure 3.1.

[2] supercollider.sourceforge.net

[3] www.gnu.org/software/octave

[4] www.sonification.de/projects/sc3/index.shtml

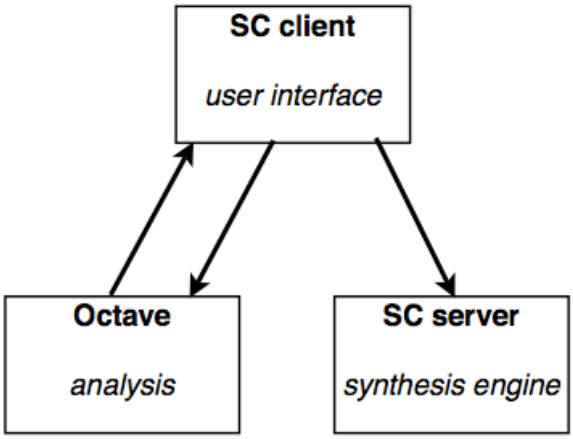

Figure 3.1: The application workflow

3.1 Analysis

The analysis process handled by Octave is summarized in Figure 3.2:

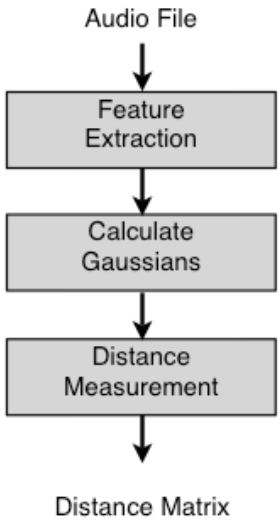

Figure 3.2: Flowchart showing the analysis process

Upon receiving a request to analyze a selected file, Octave reads the audio data and receives parameters indicating the selected analysis frame length, hop size, grain length, grain hop size, and the range for the final distance matrix. The analysis is separated into three functions, *octfeatures, octgauss,* and, *octklsymm,* which are called one after the other.

In *octfeatures,* feature extraction is first performed at the frame level for the entire audio data. The default frame length is set to 1024 samples, with a hop size of 512 samples. Mel frequency Cepstral Coefficients are calculated following the procedures described in section 2.1. The selected number of coefficients is 18. The default values can be changed in the application in order to experiment with different time or frequency resolution for the analysis.

In *octgauss,* the entire feature data set is segmented according to the grain length and the corresponding grain hop size. These values are restricted so that the length will encompass a certain number of frames, and the hop size will be a multiple of the analysis hop size. For each grain, the mean values, covariance matrix, and the inverse covariance matrix are calculated and stored in an Octave structure. The output is an array of structures representing each grain.

Finally, the distances are calculated in *octklsymm.* The symmetric KL divergence is found between all grains and stored in an nxn matrix, the values are scaled according to the selected scaling factor, and finally normalized between 0 and 1. As mentioned earlier, since the distance $D_{ij} = D_{ji}$, the calculation is only performed 'one way'. The half that is below the main diagonal is simply mirrored to fill the square matrix. The diagonal representing self-similarity is set to 0. The whole reason for actually representing distances in a square matrix is that it is convenient in the algorithm used for multidimensional scaling.

In Octave, the variable that eventually will hold the distance matrix is initially set to the value 0. While analysis is in progress, a routine running in the SuperCollider client application checks

every 3 seconds for valid data. Once the process is completed, the data is sent and the connection between the two applications is automatically terminated.

3.2 Creating the layout

Upon reception of the distance matrix, the client is set up for starting the multidimensional scaling algorithm. The algorithm starts by initializing random coordinates for the entities in the data set, and creates subsets used in force calculations. The neighboring set has initial threshold values set to 0.25. A routine is executed that on each iteration generates a stochastic set, then, it calls a function to update the velocity vector along with the position of each entity. The neighboring set is normally defined after the first couple of iterations. The entities are displayed in a 2-dimensional scatterplot which is also updated on each iteration of the routine. To keep all entities within the view, the coordinates are normalized before being displayed. When the user decides to terminate the process, the k-dimensional tree is constructed based on the current positions of the entities in the plot.

3.3 Graphical Interface

The user interface is split into two sections; the visualization/navigation space, and a control section, see Figure 3.3. The user can select different tabs in the control section to (1) load an audio file and control the multidimensional scaling process, (2) control the real-time granulation and selection parameters, and (3) set preferences for the analysis, see Figure 3.4.

The setup allows the user to load an audio file, and call for it to be processed in Octave. When the application receives valid data, the option to start the MDS process is activated. The process can be

terminated by the user at any time. If the layout or the sounding results are not satisfactory, the preferences allow for the selecting of a specific portion of the audio file, adjusting the analysis frame length and hop size, the grain length and grain hop, and the scaling factor applied to the distance matrix. The grain length and grain hopsize determines the segmentation of the audio file, while the scaling factor adjusts the distribution in the distance matrix, which, again, has an impact on the layout.

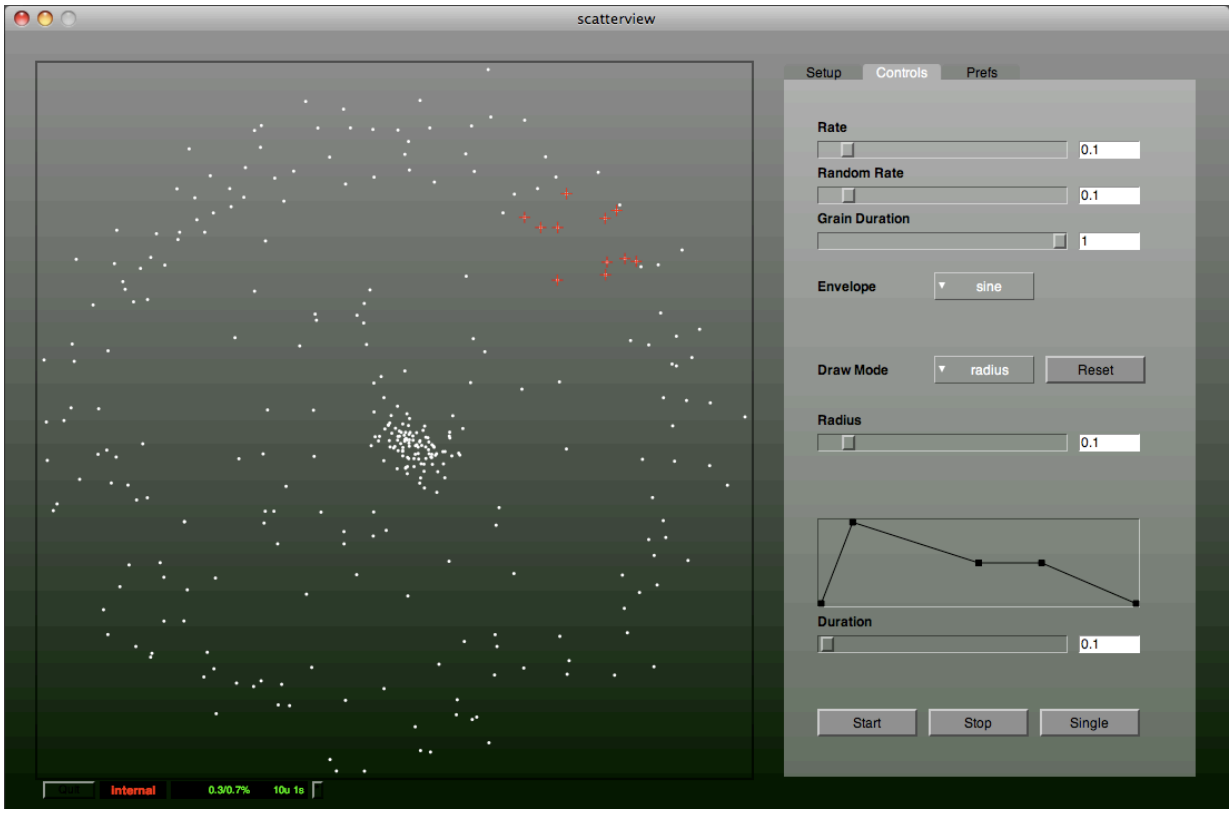

Figure 3.3: The user interface

3.4 Control

The control tab for real-time performance offers some standard parameters in relation to granulation synthesis. Sliders allows for the grain rate to be set and to introduce randomness for asynchronous playback. A slider allows for decreasing the grain duration. The grain envelope function can be set by the options available in a pop-up menu. Currently the different types included are sine, triangle, and a percussive shape featuring a sharp attack and exponential decay. Buttons are provided for starting and stopping the stream of grains, and an option to trigger a single sound object with an adjustable ADSR envelope and duration is offered. This is useful in creating separate sound objects.

Interaction with the layout is achieved by tracking mouse movement in the visualization space. Options are provided for selection based on nearest neighbor, selection within a radius around the target position, and free drawing in the space. When several entities are selected, these are stored in an array in which random indices are selected for playback.

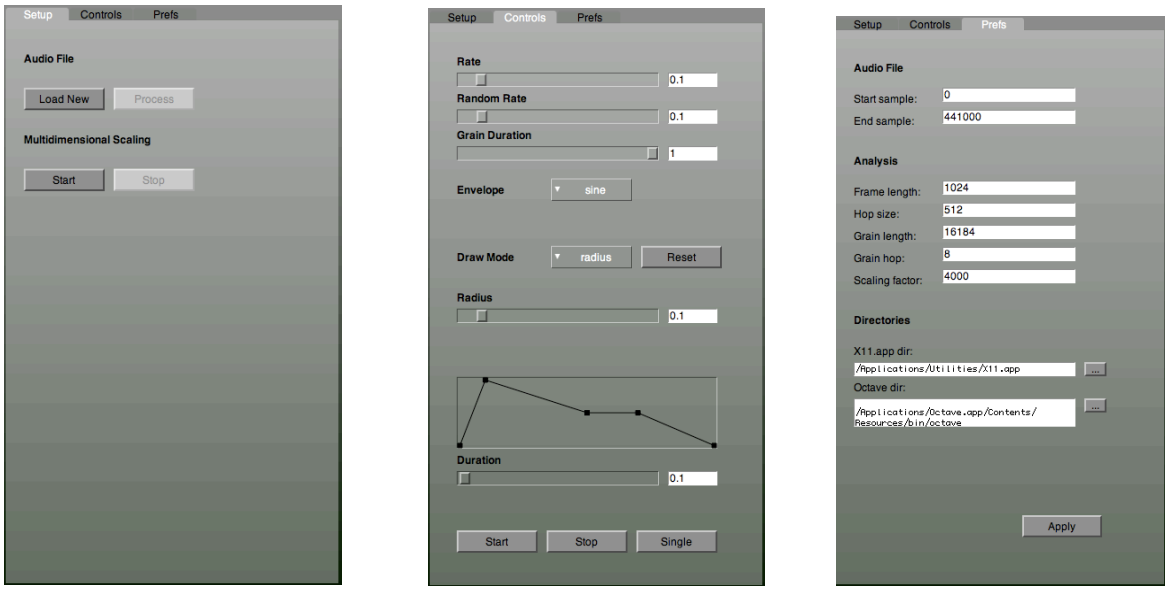

Figure 3.4: The setup, control and preference panel

3.5 Synthesis Engine

The loaded audio data is stored in a buffer on the server. A routine takes care of sending messages from the client to the server application at specified intervals. A message results in the creation of a synth node. The specific synth node is defined by a synth definition and is a container for one or more unit generators that execute together. A unit generator is an object that processes or generates sound on the server. An individual synth node is created for each message received. It takes the following input parameters: a reference to the buffer, start position, duration, and an envelope shape. The synth node ceases to exits when it has finished playing. The routine is controlled by a ‘wait’ time that is set by the user.

4.0 Results

4.1 Discussion

The focus of this thesis has been on developing a utility for manipulating sound, and not necessarily do extensive testing on the methods used. I assume that users may have different creative processes in mind when using the application, so the performance may be better, or worse, for particular needs. The sounding results really depend on the sound material being manipulated, and the context of how it is used. However, there are many aspects that allow for discussion. In the following paragraphs, I will bring forward issues that I have encountered, and make suggestions for solutions.

Analysis

The analysis is the foundation of the application. The goal is to have a distance measure that reflects the timbral similarity between grains. Assuming the multidimensional scaling process is adequate, the results can be evaluated by observing the distribution of entities in the layout, and by listening to grains played back from different parts of the space. In my experience, the separation between differently sounding grains, and clustering of similar grains, is good for the cases where the underlying audio material is somewhat coherent over its duration. I have identified some specific cases where the results are poor.

Grains that are not coherent, or homogenous, over their duration represent a challenge. Examples would be a sustained texture that is interrupted by a transient, or grains featuring dramatic variations in timbre. I refer to these instances as *ambiguous grains.* Due to fast, or

complex, changes in the spectral content, these instances often do not find a clear relationship with other grains and, therefore, do not form meaningful clusters. In the layout, they tend to just fill the space between other more homogenous groups of grains and, in many cases, do not reflect a natural transition between different parts.

A possible solution to this issue would be to introduce informed segmenting of the audio material. Currently, segmenting is naively restricted to a set grain length and hop size. The informed approach could be implemented by performing detection of onsets or transients, separating potentially problematic parts of the signal from more sustained homogenous parts. Shorter events and rapidly changing sections may benefit from shorter grain lengths, compared with more sustained textures. Varying grain lengths could easily be accommodated by the similarity algorithm, but would require extra information, specifying lengths for individual grains, to be attached along with the distance matrix passed to the client application. A side note to this is that there might always be cases where these '*ambiguous grains'* are a welcome part of the layout. They may introduce an aspect of chaos into an otherwise controlled environment.

At the time of writing, it is left to the user to remove silence from the audio files used for the layout. Grains that are too quiet have no function but to appear as gaps in the stream of grains on playback. If this is desirable, it can easily be introduced by adjusting the rate at which grains are triggered. Silent grains may also distort the layout, as they may be found severely dissimilar to the rest of the content, consequently making the remainder appear more similar than necessary. I have also observed that certain grains with low energy tend to cluster near noisy grains that feature a broad and relatively flat spectrum. Perhaps this is explained by the more prominent background noise in these quiet segments.

Another issue that I came across during development, also noted in (Pampalk, 2006a), was calculating the inverse covariance matrix, used in the distance calculation, for certain sets of

features. These particular segments feature minimal variance in their MFCC representation. The solution used in this implementation is to calculate a pseudo-inverse covariance matrix in these cases, using Octave's *pinv* function, where any singular values less than a set tolerance are treated as zero (Moore, E. H., 1920; Penrose, 1955). This does not appear to have detrimental effects on the distance calculation, but should be investigated if accuracy is of the essence. An alternative would be to completely leave these segments out of the algorithm.

In the analysis, no concern has been given to the fact that an envelope will be applied to the audio segments. I ignored this issue, as I saw the ability to change the grain envelope type in the application as a nice feature. Applying the envelope means removing, or attenuating, parts of the audio that have been analyzed, and in some cases perhaps compromise the validity of the representation. Allowing for adjustment of the grain length after the analysis stage also brings up the same issue. A possible solution to this could be putting more emphasis on the center part of the audio segment that is analyzed. As, in most cases, the envelope will mostly affect the boundaries.

With regards to computation time, what slows down the analysis process is the distance measurement. Although implementing the simple optimization steps outlined previously, specifically ignoring self-similarity and only performing a 'one way' distance calculation between grains, this step takes a significant amount of time in the presence of a large number of grains. As analysis is performed offline, this does not really impose a problem. If extensions were to be made to the application that would rely on faster performance, perhaps an alternative similarity measurement should be investigated, or possible optimizations for calculating the symmetric Kullback-Leibler divergence should be looked at.

Layout

The layout produced by the force-directed multidimensional scaling algorithm seems to be ideal for the purpose of this application. However, I think that a useful feature would be to provide a visual display of the effects of applying the final scaling to the distance matrix before initiating the process. Plotting the matrix itself may not necessarily be the ideal solution, but perhaps indicating the distribution of small, median, and large distances would be possible. The distribution is well reflected in the final layout, and may in some cases save the user some time in selecting the right values.

I would also like to note a particular characteristic with the observed layouts. Prominent clusters will, in most cases, eventually travel to the edges of the visualization space. The behavior is a result of the cumulative force that closely related entities exert on the remainder of the set. This feature can be useful as an indicator that the layout is reaching a settled state, but may also reflect an inherent weakness in the chosen method. In some cases, all the underlying timbral relationships will not be reflected in the layout, i.e. a smaller number of close connections lose out to the larger cluster groups. Experiments with the scaling of the distance values would perhaps reveal different characteristics, and perhaps a more ideal solution.

Interaction

In terms of interaction, I have only implemented what I considered a minimum for making the application useful. The visualization space is the main feature in this application, and I believe mouse tracking is a limiting way of interacting with the layout. Perlin, Rosenberg, et al.'s *UnMousePad,* a pressure sensitive multi-touch surface, would be a great candidate to implement support for in this application. It would allow a user to simultaneously access different parts of the

space, and in addition, either amplitude, or the radius of a larger selection, could be mapped to pressure.

4.2 Future Work

Along with the improvements outlined in the previous section, I would like to propose some extensions to the application.

In terms of application performance, the programming environments used were chosen for their abilities in offering fast prototyping and implementation. For optimizing and extending the application, I would suggest moving the application to a more consistent framework. This could make way for optimizing both the visualization and the analysis process, as well as providing better data handling.

A feature that would benefit the application greatly would be to make way for adding and removing audio content to an existing layout. Currently the application only accommodates one audio file at a time. The removal could simply include an option to delete an entity from the visualization space, thereby removing any reference to the audio segment. Introducing new audio would require analysis and adding additional distance measurements to the current distance matrix. Due to the iterative nature of the multidimensional scaling algorithm, new entities can easily and quickly be appended to an existing layout without rerunning the entire process.

In terms of the analysis, I would suggest experimenting with different feature sets. Chroma features would perhaps be a useful representation for musical material. Clusters would then presumably reflect different harmonies used in the composition. Using a battery of low level

features, as mentioned in (Jehan, 2005), may possibly provide a more accurate description of audio segments.

There would have been benefits by extending the visualization space to 3 dimensions. The extra dimension would allow the high-dimensional relationships to be better reflected in the layout. However, this would also bring up many new issues related to interaction, which I consider to be beyond the scope of this thesis.

4.3 Conclusions

In this thesis, an approach to content-based audio synthesis has been proposed. The approach has been realized in an application that performs offline analysis of a given audio file and creates a navigable layout based on timbral relationships between segments of the chosen material. The synthesis is performed in real-time and allows for interaction through mouse input.

Starting this project, I had an idea, or more of a desire, to gain a sense of unlimited control in manipulating recorded sound material. The idea sparked from being introduced to the various methods used in music information retrieval, including procedures for feature extraction and classification of audio signals. It formed into a more feasible definition of moulding sounds according to some higher level understanding of the sound characteristics through analysis and organization. The initial experiments included a simple implementation of content-based concatenative synthesis along the lines of i.e. (Zils, et al., 2001; Schwarz, 2000), performing selection of audio segments from a corpus according to a target audio file. Although very interesting, it seemed like a natural progression to leave the control to the user, rather than performing selection based on already existing material. The *CataRT* interface, described in the

related work section, drew my attention to the idea of projecting multidimensional features onto a simple navigable surface. The application started taking shape from there on in.

During the process, I have become aware of a number of possibilities within this area of sound synthesis, but I have also noticed limitations. By adding the extra dimension of control, as discussed in this thesis, one also loses out on other attractive aspects of the synthesis. An example of a use could be to carefully scan over a selected sound event that features interesting temporal evolution. In other words, by focussing control on a particular feature, one imposes a number of limitations in other areas. That said, it seems more accurate to see this application as adding more constraints on the user. In composition, constraints are often used in terms of working out of a musical style, setting up rules regarding the overall structure of a piece, or making specific choices for smaller sections, such as key changes or development a composed theme. Setting these constraints on a small scale, as in focussing control on a specific characteristic of a sound may lead to interesting compositional results.

5.0 Bibliography

A. S. Associations (1994). Acoustic Terminology. S1.1-1994 (R2004).

Blinn, J. F. (1993). What's that deal with the DCT?. *Computer Graphics and Applications, IEEE, 13(4), 78–83.*

Buser, P. A. (1992). Audition. *Massachusets, USA: MIT Press*.

Chalmers, M. (1996). A linear iteration time layout algorithm for visualising high-dimensional data. *In VIS '96: Proceedings of the 7th conference on Visualization '96 127-133.*

Di Battista, G., Tollis, I. G., G., Eades, P., and Tamassia, R. (1998). Graph Drawing: Algorithms for the Visualization of Graphs. *New Jersey, US: Prentice Hall.*

D'haes, W., van Dyck, D., and Rodet, X. (2003). PCA-based branch and bound search algorithms for computing K nearest neighbors. *Pattern Recognition Letters, 24(9-10), 1437–1451.*

Eades, P. (1984). A Heuristic for Graph Drawing. *Congressus Numerantium, 42, 149–160.*

Flexer, A., Schnitzer, D., Gasser, M., Widmer, G. (2008). Playlist Generation using Start and End Songs. *In Proc. International Symposium on Music Information Retrieval (ISMIR'08).*

Foote, J. T. (1997). Content-based retrieval of music and audio. *In Multimedia Storage and Archiving Systems II, Proc. of SPIE, 138-147.*

Gasser, M., Flexer, A., and Widmer, G. (2008). Streamcatcher: Integrated Visualization of Music Clips and Online Music Streams. *In Proc. International Symposium on Music Information Retrieval (ISMIR'08).*

Jehan, T. (2005). Creating Music by Listening. *PhD Thesis, Massachusetts Institute of Technology.*

Logan, B. (2000). Mel frequency cepstral coefficients for music modeling. *In Proc. International Symposium on Music Information Retrieval (ISMIR'00).*

Martin, K. (1999). Sound Source Recognition: A Theory and Computational Model. *PhD Thesis, Massachusetts Institute of Technology.*

Moore, A. (1991). A tutorial on kd-trees. *Extract from PhD Thesis, University of Cambridge Computer Laboratory Technical Report No. 209.*

Moore, E. H. (1920). On the reciprocal of the general algebra matrix (Abstract). *Bulletin from the American Mathematical Society, 26, 394-395.*

Moreno, P. J., Ho, P. P., and Vasconcelos, N. (2003). A Kullback-Leibler divergence based kernel for SVM classification in multimedia applications. *In Advances in Neural Information Processing Systems 16.*

Morrison, A., Ross, G., and Chalmers, M. (2003). Fast multidimensional scaling through sampling, springs and interpolation. *Information Visualization, 2(1), 68–77.*

Oppenheim, A. V. (1969). Speech Analysis-Synthesis System Based on Homomorphic Filtering. *The Journal of the Acoustical Society of America, 45(1), 309.*

Pampalk, E. (2006b). Audio-Based Music Similarity and Retrieval: Combining a Spectral Similarity Model with Information Extracted from Fluctuation Patterns. *In Proceedings of the International Symposium on Music Information Retrieval (ISMIR'06).*

Pampalk, E. (2006a). Computational Models of Music Similarity and their Application to Music Information Retrieval. *PhD Thesis, Vienna University of Technology.*

Penrose, R. A. (1955). A generalized Inverse for Matrices. *Proceedings of the Cambridge Philosophical Society, 51, 406-413.*

Rabiner, L. and Juang, B. (1993) Fundamentals of Speech Recognition, *New Jersey, US: Prentice Hall.*

Roads, C. (2004). Microsound. *Massachusets, USA: MIT Press.*

Schwarz, D. (2000). A System for Data-Driven Concatenative Sound Synthesis. *In proc. COST G-6 Conf. on Digital Audio Effects (DAFx-00), 97–102.*

Schwarz, D. (2004). Data-Driven Concatenative Sound Synthesis. *PhD Thesis, University of Paris 6 - Pierre et Marie Curie.*

Schwarz, D., Beller, G., Verbugghe, B., and Britton, S. (2006). Real-Time Corpus-Based Concatenative Synthesis with CataRT. *In proc. Digital Audio Effects (DAFx-06), 279-282.*

Schwarz, D. (2007). Musical Applications of Real-Time Corpus-Based Concatenative Synthesis. *In proc. International Computer Music Conference (ICMC'07).*

Shannon, B. J. and Paliwal, K. K. (2003). A comparative Study of Filter Bank Spacing for Speech Recognition. *In proc. Microelectronic Engineering Research Conference.*

Truax, B. (1988). Real-Time Granular Synthesis with a Digital Signal Processor. *Computer Music Journal, 12(2), 14–26.*

Tzanetakis, G. (2002, June). Manipulation, Analysis and Retrieval Systems for Audio Signals. *PhD Thesis, Princeton University*.

Wright, M. and Freed, A. (1997). Open Sound Control: A New Protocol for Communicating with Sound Synthesizers. *In proc. International Computer Music Conference (ICMC'97).*

Xenakis, I. (1971). Formalized Music: Thought and Mathematics in Composition. *Bloomington: Indiana University Press.*

Zils, A. and Pachet, F. (2001). Musical Mosaicing. *In proc. COST G-6 Conf. on Digital Audio Effects (DaFx-01).*