Manipulation and Resynthesis of Environmental Sounds with Natural Wavelet Grains

by

Reynald Hoskinson

B.A. (English with Computer Science Minor)

McGill University, 1996

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES

(Department of Computer Science)

We accept this thesis as conforming
to the required standard

The University of British Columbia

March 2002

Department of Computer Science

The University of British Columbia
Vancouver, Canada

Date March 27 2002

Abstract

A technique is presented to facilitate the creation of constantly changing, randomized audio streams from samples of source material. A core motivation is to make it easier to quickly create soundscapes for virtual environments and other scenarios where long streams of audio are used. While mostly in the background, these streams are vital for the creation of mood and realism in these types of applications.

Our approach is to extract the component parts of sampled audio signals, and use them to synthesize a continuous audio stream of indeterminate length. An automatic speech recognition algorithm involving wavelets is used to split up the input signal into syllable-like audio segments. The segments are taken from the original sample and are not transformed in any way.

For each segment, a table of similarity between it and all the other segments is constructed. The segments are then output in a continuous stream, with the next segment being chosen from among those other segments which best follow from it. In this way, we can construct an infinite number of variations on the original signal with a minimum amount of interaction. An interface for the manipulation and playback of several of these streams is provided to facilitate building complex audio environments.

Contents

List of Figures

Acknowledgements

Thank you to my parents Michael and Gisele, the adroit guidance of Dinesh K. Pai, and Holger Hoos. I also appreciate the efforts of Antoine Maloney, who has always given me good advice, although I haven't always followed it.

REYNALD HOSKINSON

The University of British Columbia
March 2002

Chapter 1

Introduction

1.1 Problem and Motivation

Natural sounds are an infinite source of material for anyone working with audio. Although the source may be infinite, there are many situations where one sample has to be used repeatedly. Electro-acoustic music composers often use samples as motifs that reappear again and again over the course of a piece. Acoustic installations sometimes stretch pre-obtained source material over the entire life of an exhibit. Video games can use the same sample ad infinitum during game play. Simple repetition is not effective for long, so we often create variations of a sample by manipulating one or more of its properties.

There is a long tradition in the electro-acoustic music community of splitting audio samples into portions and manipulating them to create new sounds. Curtis Roads [Roa78] and Barry Truax [Tru94] pioneered granular synthesis, in which small grains are combined to form complicated sounds. Grains can be constructed from scratch, or obtained by splitting an audio sample into small segments. More recently, Bar-Joseph [BJDEY+99] proposed a variant of granular synthesis using wavelets where the separation and re-combination of grains is done in the time-frequency representation of an audio sample. Similar work is also being done on images to produce variations of tiles or textures [WL00, SSSE00].

When what is desired is simply a variation on the original source that still bears a strong resemblance to the original, the above audio techniques have critical problems. Granular synthesis is a technique to create new sounds, not recognizable variations of the original except in a very abstract sense. A long audio sample is not even required; it suffices to specify the shape of the grain and its envelope. When an audio sample is used, a grain is an arbitrary slice chosen independently of the sound's inherent structure.

Attempts at better preserving the original structure of the sound have been made. Bar-Joseph [BJDEY+99] uses a comparison step where wavelet coefficients representing parts of the sample are swapped only when they are similar. The authors employ "statistical learning", which produces a different sound statistically similar to the original. In this algorithm, only the local neighbours in the multi-resolution tree are considered when calculating similarity, and the swapping is very fine-grained. This means that large-scale changes over time will not be taken into account. On almost any signal, this results in a "chattering" effect.

To address the limitations of the above methods, we have developed an algorithm for segmenting sound samples that focuses on determining natural transition points. The sound in between these transition points is considered atomic and not broken up any further or transformed in any way. We refer to the sound between transition points as "natural grains".

Once we have the grains, creating new sounds becomes a problem of how best to string them together. We do this by constructing a first-order Markov chain with each state of the chain corresponding to a natural grain. The transition probabilities from one state to others are estimated based on the smoothness of transition between it and all other grains. The natural grains are thus output in a continuous stream, with the next grain being chosen at random from among those other grains which best follow from it. In this way, we can construct an arbitrarily large number of variations on the original signal with a minimum amount of user input.

1.1.1 Natural Grains

Segmenting an audio sample into natural grains involves some understanding of the process of how the acoustic waves that are detected by our ears are transformed into the sounds we perceive. What cues do we use to distinguish one sound from another? More specifically, what are the clues that our brains pick up to distinguish where one sound ends and the next begins?

From the time Helmholtz published "On the sensations of tone as a psychological basis for the theory of music" in 1885 [Hel54], it was generally held that the steady-state components in a sound were the most important factor in human recognition. Risset and Matthews [RM69] wrote a seminal study of the time-varying spectra of trumpet tones which invalidated this hypothesis. Their work showed that the primacy of steady-state components was not realistic from the perspective of synthesizing realistic musical instrument sounds.

Instead they proposed that the *dynamic* components of a spectrum were primary, and steady-state components did not help very much at all for instrument classification and tone-quality assessment. Risset's hypothesis has now become the dominant view of sound structure in the psychological literature, and claims to represent the perceptibly important dynamic structures that comprise auditory phenomena from the perspective of musical instrument sound structures.

Handel [Han95] defines *timbre* as perceptual qualities of objects and events, or "what it sounds like." The sense of timbre comes from the emergent, interactive properties of the vibration pattern. Clearly, any segmentation algorithm that purports to preserve the perceptual properties of the sound must segment on a larger scale than the local changes that make up the timbre of a sound event.

Drawing on the work of Hemholtz, Michael Casey [Cas98] enumerates the types of change in a sonic structure by examining the constraints of the human auditory system. **Fourier Persistence** refers to how the cochlear mechanics of the ear are sensitive to changes on the order of 50ms and shorter. The ear represents

these changes as a static quality in log frequency space. In other words, when our ears sense regular changes in air pressure at rates greater than 20Hz, we perceive one pitch rather than each individual change in air pressure. 20Hz is the frequency perception threshold of the cochlear mechanism.

We are, however, able to perceive changes occurring at rates less than 20Hz as actual change. Those that are continuous in terms of the underlying Fourier components are classified as **short time** changes in the static frequency spectrum. For example, when I drop a coin, there is a Fourier persistence due to the physical characteristics of a small metallic object. The short-time change reflects the individual impacts.

The above information leads us to focus on the changes on scales greater than 20Hz for our segmentation algorithm. Considering that we are looking at samples recorded at 44.1kHz, our windows should be at least 2205 samples long.

1.1.2 Ecological Perception

There is a significant amount of literature arguing that the atomicity of human perception of sound is more on the level of what we have defined as a grain than that of an individual sound wave. J. J. Gibson [Gib79] originally introduced the term **Ecological Perception** to denote the idea that what an organism needs from a stimulus, for the purposes of its normal everyday life, is often obtained directly from invariant structures in the environment. Some types of complex stimuli may be considered as elemental from the perspective of an organism's perceptual apparatus, unmediated by higher-level mechanisms such as memory and inference.

While Gibson was primarily referring to the vision system, there are analogous patterns in hearing. Perception is not simply the integration of low-level stimuli, such as single pixels in the retina or narrow-band frequency channels in the cochlea, but instead directly perceivable groups of features.

Ecological perception was further explored for the auditory domain in William

Gaver's pioneering work on everyday listening [Gav88]. Everyday listening involves perceiving the source of the sound and its material properties such as size and weight. Take, for instance, the sound of a door slowly closing on rusty hinges. In everyday listening, attention is focused on the door itself, the force with which it is being closed, the size of the room it is being closed in, and other material properties of the origin of the sound. This type of listening is differentiated from another type of auditory experience, musical listening, in which musical parameters such as pitch, duration and loudness are most important. In the door example, musical listening would involve hearing the change in pitch as the door opens, the particular timbre of the hinges, and the band-limited impulsive noise as the door hits the frame. Everyday listening instead involves distinguishing the individual events which produce the sounds that we hear.

From the perspective of everyday listening, the perceptual world is one where sounds have clear beginnings and endings, even continuous sounds such as wind that have no onsets or offsets. In this way, a grain could be defined by its temporal boundaries. However, the beginning and endings of sounds are not necessarily due to the structure of the acoustic wave; often they are not physically marked by actual silent intervals. Our main task, then, is to find points in the acoustic wave which best approximate the beginning and endings that we can perceive by listening to the sounds ourselves.

However, there is as yet no accepted mathematical framework within which to use this theory of perception in a systematic manner. Casey [Cas98] does provide a mathematical framework using group theory, but he is primarily concerned with extracting the structure of larger-scale sounds, such as the timing and spectral changes between bounces as a ball bounces a number of times before settling on the ground.

While any sound consists of a time-varying pattern of harmonics, when sounds from different sources overlap, all of the harmonic components are mixed

in time and frequency. A listener can use the timing, harmonic, and amplitude and frequency modulation relationships among the components to parse the sound wave into discrete component sources. These component sources often are happening contemporaneously, and are called "streams" in the literature of acoustic perception. In this thesis, however, will limit ourselves to splitting sound solely in the time domain, with one stream only per sample.

1.1.3 Objectives

There will never be a lack of natural world sounds to record and feed into a computer system. In an application which uses empirically recorded samples, playing them back blindly is inefficient in terms of resources and often less than optimum in terms of the desired effect the sound has on a user. The more the information the application has about the sound, the more it can tailor its output to its situation.

Our implementation aims to provides users with a tool to automatically manipulate sound sources and soundscapes. Along with the core segmentation/resynthesis tool, we provide a higher level interface which allows for multiple randomized sounds to be played at once, each with a number of controls that effect how it appears in the soundscape. There are controls for automating how often a sound stream is triggered and how long an instance lasts. There are pan and gain controls for controlling stereo amplitude over the course of the instance. This allows a user to assemble a permanently changing auditory soundscape from just a few representative samples. Such a tool is useful for immersive virtual environments, video games, soundtracks for film, auditory displays, and even music composition.

1.2 Thesis Organization

This thesis is divided into six chapters. Chapter 1 introduced the problem, and stated the objectives of the work. In chapter 2, we provide a general background on the signal processing tools used. Chapter 3 details our earlier, different approach to

creating sound textures. The segmentation and resynthesis algorithm is detailed in chapter 4. Chapter 5 shows the results of the research, including some user testing to demonstrate utility. Finally, chapter 6 summarizes goals and results, and offers some potential future research areas.

Chapter 2

Background and Related Work

2.1 Overview

This chapter will review related research into the representation and analysis of audio signals for the purpose of segmentation. The wavelet transform will then be introduced, and we will review some of the methods which use wavelets for signal segmentation. We then review some of the related work which uses wavelet packets. Finally, techniques from the electro-acoustic community such as granular synthesis and concatenative synthesis are briefly touched upon.

2.2 Representation

The sound samples we use for this algorithm are input in pulse-code modulation (PCM) format. PCM means that each signal sample is interpreted as a "pulse" at a particular amplitude. To determine where to segment an input audio sample, it is useful to change representations from the original PCM format to something which more compactly expresses the information we are interested in.

Ultimately, we need a representation that can aid us in determining when an audio signal changes, and by how much. With the location of change, we can segment the signal into natural grains. With a metric of how much the signal has

changed, we can offer a threshold value that increases or decreases the coarseness of the grains, and also compare them to each other to estimate how they fit together.

There are a multitude of ways to represent a sound signal. Using an appropriately multi-resolutional approach, [SY98] identifies three main schemes for classifying audio signals:

1. **Signal statistics** such as

   (a) mean

   (b) variance

   (c) zero-crossing

   (d) auto-correlation

   (e) histograms of samples/difference of samples, either on the whole data or blocks of data.

   The main problem with low-level statistics is that they are vulnerable to alterations in the original signal, making them fragile in the presence of noise.

2. **Acoustical attributes.** Another general category of audio signal classification tools are acoustical attributes such as pitch, loudness, brightness and harmonicity. Statistical analysis is then applied to these attributes to derive feature vectors. Because we hope to preserve as many of the acoustical attributes as possible in our resynthesized sound, this appears to be a much better alternative.

   Most of these measurements, however, are directed towards music where the sound is already relatively structured. For less structured environmental sounds, where possibly many different events are happening simultaneously, these measurements are less effective. Calculating acoustic attributes is also much more expensive computationally overall, and suffers from the same lack of robustness to noise as traditional signal statistics.

3. **Transform-based Schemes.** Here the coefficients of a transform of the signal are used for classification to reduce the susceptibility to noise. There are many advantages to using signal transforms in analysis, such as the potential for compression, and the ability to tailor the transform used to bring out the characteristics of the signal that are most important to the task at hand. The canonical transform used in audio processing is the Fourier transform, in part because of the efficiency of the Fast-Fourier transform, and its utility over a broad range of tasks. For reasons explained below, we instead chose the Discrete Wavelet Transform, which has the same algorithmic complexity as the FFT.

2.3 Signal Transforms

A *spectrum* can be loosely described as "a measure of the distribution of signal energy as a function of frequency" [Roa96]. We must define this term loosely because according to Heisenberg's Uncertainty Principle, any attempt to improve time resolution of a signal will degrade frequency resolution [Wic94]. Both the time waveform and frequency spectrum cannot be made arbitrarily small simultaneously [Sko80]. The product of these two resolutions, the time-bandwidth product, remains constant for any system. So any representation of a signal's spectrum is necessarily a trade-off between these competing concerns.

Long a staple of digital signal processing, the Fourier transform is one way of calculating a spectrum. It was originally formulated by Jean Baptiste Joseph Fourier (1768-1830). Its main principle is that all complex periodic waveforms can be modeled by a set of harmonically related sine waves added together. The Fourier transform for a continuous time signal $x(t)$ can be defined as:

$$X(f) = \int_{-\infty}^{+\infty} x(t) e^{j2\pi ft} dt \tag{2.1}$$

The results of evaluating $X(f)$ are analysis coefficients which define the global

frequency f in a signal. As shown in Figure 2.1, the coefficients are computed as inner products of the signal with sine wave basis functions of infinite duration. This means abrupt changes in time in a non-stationary signal are spread out over the whole frequency axis in $X(f)$.

To obtain a localized view of the frequency spectrum, there is the Short-Time Fourier transform, (STFT) in which one divides the sample into short frames, then puts each through the FFT. As shown in Figure 2.1 below, the smaller the frame, the better the time resolution, but employing frames raises a whole new set of problems.

To begin with, it is impossible to resolve frequencies lower than the length of the frame. Using frames also has the side effect of distorting the spectrum measurement. This is because we are measuring not purely the input signal, but instead the product of the input signal and the frame itself. The spectrum that results is the convolution of the spectra of the input and the frame signals.

For each frame, we can think of the STFT as applying a bank of filters at equally spaced frequency intervals. The frequencies are spaced at integer multiples of the sampling frequency, divided by the frame length. Artifacts of frame analysis arise from the fact that the samples analyzed do not always contain an integer number of periods of the frequencies they contain. There are a number of strategies to curb the effect of this "leakage", such as employing a envelope on each frame that accentuates the middle of the frame at the expense of the sides, where most of the leaking is [Roa96].

For audio purposes, the FFT also has the drawback that it divides the frequency spectrum up into equal linear segments. The user is put into an inescapable quandary: narrow frames provide good time resolution but poor frequency resolution, while wide frames provide good frequency resolution but poor time resolution. Moreover, if the frames are too wide, the signal within them cannot be assumed to be stationary, which is something the FFT depends on.

The problem with using linear segments is that humans perceive pitch on a scale closer to logarithmic [War99]. We are relatively good at the resolution of low-frequency sounds, but as the frequency increases, our ability to recognize differences decreases.

2.4 The Wavelet Transform

A wavelet is a waveform with very specific properties, such as an average value of zero, and an effectively limited duration. Analysis with wavelets involves breaking up a signal into shifted and scaled versions of the original (or *mother*) wavelet. Wavelet analysis uses a time-*scale* region rather than a time-frequency region. Since only artificial tones are purely sinusoidal, this is not in itself a drawback.

The wavelet transform is capable of revealing aspects of data that other signal analysis techniques miss, such as trends, breakdown points, discontinuities in higher derivatives, and self-similarity. Intuitively, the wavelet decomposition is calculating a "resemblance index" between the signal and the wavelet. A large index means the resemblance is strong, otherwise it is slight. The indices are the wavelet coefficients.

In contrast to the linear spacing of channels on the frequency axis in the STFT, the wavelet transform uses a logarithmic division of bandwidths. This implies that the channel frequency interval (bandwidth) Δ f/f is constant for the wavelet transform, while in the STFT, the frame duration is fixed.

To define the continuous wavelet transform (CWT), we start by confining the impulse responses of a particular filter bank to be scaled versions of the same prototype $\psi(t)$:

$$\psi_a(t) = \frac{1}{\sqrt{|a|}}\psi\left(\frac{t}{a}\right) \tag{2.2}$$

where a is a *scale factor*, and the constant $\frac{1}{\sqrt{|a|}}$ is used for energy normalization. $\psi(t)$ is often referred to as the mother wavelet. With the mother wavelet,

we can define the Continuous Wavelet transform (CWT) as

$$CWT_x(a,b) = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{+\infty} x(t)\, \psi\left(\frac{t-b}{a}\right) dt \tag{2.3}$$

Here * refers to the convolution operator, and $x(t)$ is the original signal. As the scale a increases, the scaled wavelet $\psi\left(\frac{t}{a}\right)$ (the filter impulse response) becomes spread out in time and thus takes only longer durations into account. Both global and very local variations are measured by using different scales a. b controls the translation of the wavelet along the signal.

We will limit our discussion to the discrete wavelet transform, which involves choosing dyadic scales and positions (powers of two). In the discrete wavelet transform, a decomposition into wavelet bases requires only the values of the transform at the dyadic scales:

$$a = 2^j$$

and

$$b = k2^j.$$

The analysis is more efficient and just as accurate for our purposes as the continuous wavelet transform. The discrete wavelet transform approach was first developed by Mallat [Mal89]. It is based on a classical scheme known as the two-channel sub-band coder.

Unlike the FFT, which uses sinusoids of infinite duration, wavelets are localized in time. Leakage is also a different concern with the wavelet transform. If we confine the length of our signal to be a power of two, the wavelet transform will analyze the signal in an integer number of steps, so leakage is not an issue. It doesn't matter that the frequencies present in the signal don't line up with power-of-two boundaries, since we are not measuring frequencies per se, but scales.

The effectiveness of the Discrete Wavelet Transform (DWT) for a particular application could depend on the choice of the wavelet function. For example, Mallat [MZ92] has shown that if a wavelet function which is the first derivative of a

smoothing function is chosen, then the local maxima of the DWT indicate the sharp variations in the signal, whereas the local minima indicate slow variations.

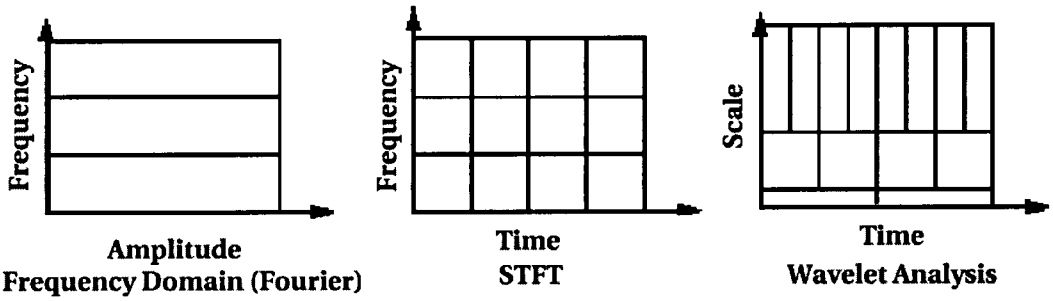

Figure 2.1: Contrast between frequency-based, STFT-based, and wavelet views of the signal

As figure 2.1 shows, the wavelet transform gives better scale resolution for lower frequencies, but worse time resolution. Higher frequencies, on the other hand, have less resolution in scale, but better in time.

One notable aspect of wavelet transforms as they pertain to audio processing is that they are not shift-independent. For this reason, we use the energies of the coefficients rather than the raw coefficients themselves for our metrics, as in [PK99].

2.4.1 Implementation

For an implementation-centred point of view, we can think of the wavelet transform as a set of filterbanks, as in figure 2.2.

Here the original signal, S, passes through two complementary filters and emerges as two signals. The low-and high-pass decomposition filters (L and H), together with their associated reconstruction filters (L' and H'), form a system of quadrature mirror filters.

The filtering process is implemented by convolving the signal with a filter. Initially, we end up with twice as many samples. Throwing away every second data point (downsampling) solves this problem. We are still able to regenerate the entire

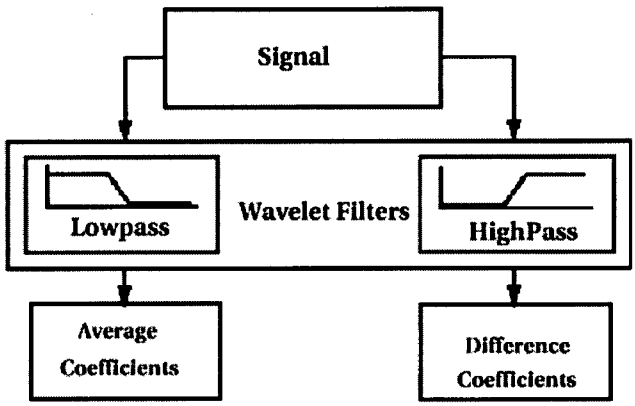

Figure 2.2: The wavelet filtering process

original sample with the downsampled signal.

The decomposition process can be iterated, with successive approximations being decomposed in turn, so that one signal is broken down into many lower-resolution components. This is the wavelet-decomposition tree, or otherwise known as the Mallat tree. The average coefficients are the high-scale, low frequency components of the signal. Difference coefficients are the low-scale, high frequency components.

To reconstruct the original signal from the wavelet analysis, we employ the inverse discrete wavelet transform. It consists of upsampling and filtering. Upsampling is the process of lengthening a signal component by inserting zeros between samples.

For a more in-depth introduction to the wavelet transform, the reader is directed to articles such as [SN96, KM88, Mal89, MMOP96] for the theory, and [Wic94, Cod92] for ideas on implementation.

2.5 Audio Segmentation using Wavelets

Once we have expressed the signal using the wavelet representation, we use it to identify the potential points to split into grains. There have been many attempts at

segmenting sound using the wavelet transform, most of those we looked at geared towards speech analysis.

Attempts at using signal statistics on wavelet transform coefficients have also been made. However, a statistical prior model for wavelet coefficients is complicated because wavelet coefficients do not have a Gaussian distribution [Mal89]. Though wavelets coefficients are decorrelated, their values are not statistically independent, another limitation to take into consideration when using statistical properties.

An important step for classification techniques such as [LKS+98, SG97, TLS+94] is to characterize the signal in as small a number of descriptors as possible, without throwing away any information that would help classification. Reducing the feature set has several advantages: primarily, it is computationally more cost effective, but it also aids in generalization. Our task is to figure out where it is best to segment the signal into grains. We must then define what happens in the signal at the start or end of an event before looking for these features.

Most speech recognition algorithms include a segmentation step to separate the phonemes for later recognition. However, segmentation in speech recognition has important differences from our goals. Smoothness of transition is not an issue for recognition, because in human speech phonemes usually blend into each other to such an extent that any splits are usually in areas with a very high degree of frequency change, too much so for our purposes. This is acceptable for speech analysis because the results are only needed for the purposes of recognition, not resynthesis.

2.5.1 Speech Classification Techniques

A paper by Sarikaya and Gowdy [SG97], on identifying normal versus stressed speech, details a scheme that showed some promise as a signal segmentation technique applicable to our needs. In their algorithm, an 8KHz sample is segmented into frames of 128 samples. A lookahead and history of 64 samples is added, to make it

256, with skip rate of 64 samples. This representation is the base of a classification algorithm that uses a two-dimensional separability distance measure between two speech parametrization classes.

Their separability measure uses Scale Energy, which represents the distribution of energy among frequency bands, defined as:

$$SE^{(k)}(s_i) = \frac{\sum_{m \in s_i} \left[(W_\psi x)(s_i, m) \right]^2}{sup(SE^{(n)}(s_i))} \tag{2.4}$$

where $W_\psi x$ is the wavelet transform of x, k is the frame number, i the scale number, s_i is the ith scale, and n spans all available frames. The denominator is a normalizing constant. They use the scale energy for an autocorrelation computation that measures the separability of two signals. The ACS, Autocorrelation of Scale Energies, measures how correlated adjacent frames are. It can be defined as:

$$ACS_{s_i}^{(l)}(k) = \frac{\sum_{n=k}^{k+L} \left[SE^{(n)}(s_i) * SE^{(n+l)}(s_i) \right]^2}{sup\left(ACS_{s_i}^{(l)}(j) \right)} \tag{2.5}$$

Here j is an index which spans all correlation coefficients at a given scale. l is the correlation lag, which is fixed in this paper at 1. If we would have set $l = 0$, we would look at only one frame at a time, so the ACS would model the normalized power in scale i. For $l > 0$, ACS models changes in the frame-to-frame correlation variation of SE parameters. Sarikaya also fixes the correlation frame length L at 6. This means the ACS parameters are measures of how correlated six adjacent frames are.

Used in this way, the autocorrelation of scale energies is a comparison between levels to bring out hidden identifying features. On a test of the phoneme /o/ in "go," they achieved satisfactory scores with both the SE and the ACS parameters, although ACS, which takes into account the change between frames, was significantly higher. Readers can refer to the paper [SG97] for further information on testing methodology and complete results.

This work is more tuned toward recognition and distinction between classes than we are interested in. The Scale Energy parameter is useful for identifying features of individual frames, and we adopt a similar approach using wavelet coefficients, as does the paper by Alani [AD99] discussed in the next section. ACS, however, tends to smooth out local changes because it is measured over a number of frames. We are more interested in the locations and magnitude of these local changes, and the *differences* between frames, and so do not adopt this technique.

2.5.2 Using Differences between Coefficients

A paper by Alani and Deriche [AD99] details another approach to segmenting speech into phonemes. The signal is broken into small frame of 1024 samples each, with an overlap of 256 samples. Sound samples are input as CD-quality 44.1 kHz, so each frame is approximately 23.2ms long. To provide metrics for what is happening during the length of each frame, they are each analyzed with the wavelet transform. The energies of each of the first six levels of difference coefficients are calculated for each frame.

Their next step is to segment the signal based on the differences in energy between each level of difference coefficients in consecutive frames. A Euclidean distance function over four frames is used. As an example, we calculate the strength of transition between frames 2 and 3:

$$D(f_2, f_3) = \sum_{i=1}^{2} \sum_{j=3}^{4} \sum_{k=1}^{6} (X_{i,k} - X_{j,k})^2 \tag{2.6}$$

Here k refers to the wavelet level, i and j are frame numbers, $X_{i,k}$ and $X_{j,k}$ are the energies of wavelet difference coefficient levels. Only like levels are compared, and the differences between them are added up to obtain a overall difference between frames.

Alani and Deriche [AD99] use the algorithm to isolate phonemes which are then fed into a separate speech-recognition system. In normal speech, vowels have

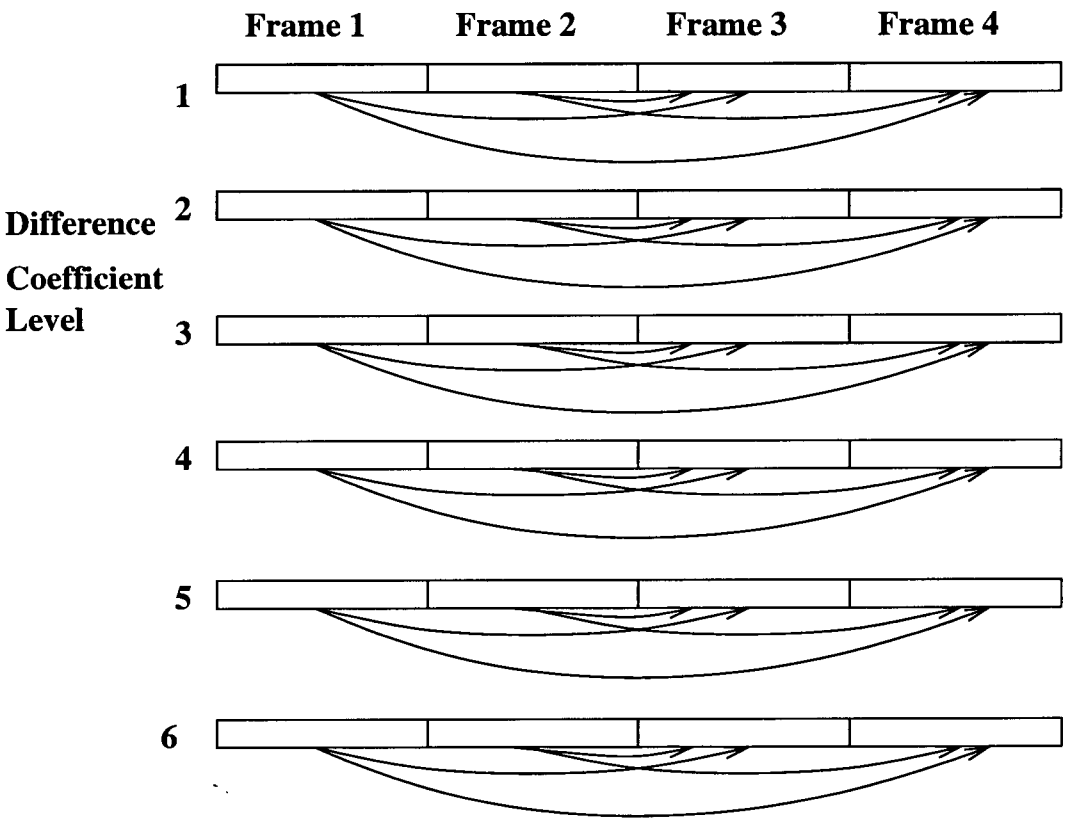

Figure 2.3: Distance measure for transition between frames 2 and 3. The arrows represent difference calculations.

pitches which are relatively constant over time, whereas consonants are not pitched at all, and so have frequencies that change considerably over the course of the phoneme. Additionally, in human speech phonemes meld into each other, making isolation an even more difficult task, one that can really only be successfully achieved using context-sensitive information. Taking this into account, the authors pick the points where the distance measure is highest, reasoning that this is where the speakers are going from one phoneme to another.

Isolating phonemes, however, is quite a different task than trying to isolate grains. We would like something more on the lines of syllables, where there are clearer demarcations to latch on to. When segmenting, we also have to take into

account how the grains will fit back together again, something Alani and Deriche did not consider. As described in the next chapter, we adopt a modified version of this algorithm which is more suitable for a coarser level of detail (on the level of syllables rather than phonemes), and our differing requirements.

2.6 Segmenting in the Wavelet Domain

Not only can we do the analysis in the wavelet domain, but it is also an option to do the separation and re-connection of grains as well. The inverse wavelet transform would then be performed to obtain the new, modified signal.

This is the approach taken by Bar-Joseph et al. [BJDEY+99]. The authors use a comparison step where wavelet coefficients representing part of the sample are swapped only when they are similar, and call their approach "statistical learning". The aim is to produce a different sound statistically similar to the original. Satisfactory results are reported, with "almost no artifacts due to the random granular recombination of different segments of the original input sound." Unfortunately, no sound samples are available to support these claims. [1]

To find out ourselves, we implemented this algorithm as it is described in the paper. More details about the implementation are given in the next chapter. The results were perceivably different, enough to make it inappropriate for our use. With any signal, we found that there was a characteristic "chattering" effect as parts of the signal were repeated quickly right after each other.

The problems stem from the way the algorithm produces new variations. Only the local neighbours in the multi-resolution tree are taken into account when calculating similarity, and the swapping is very fine-grained. Because swapping only takes place when the coefficients are similar, much of the large scale patterns are preserved, resulting in a sound that still has much of the same order of events. The

[1] At the 2001 International Computer Music Conference, I asked a number of people who had attended the conference in 1999 when Bar-Joseph's work was presented, but nobody could remember what it sounded like.

events themselves are changed to a degree, but also muddied because of convolution artifacts.

These convolution artifacts arise because switching coefficients of the wavelet transform has unpredictable results. Unless the changes are on the dyadic boundaries, it is really changing the timbre of the input sound rather than switching the sound events. These changes cannot be easily predicted; they have to do with the choices of wavelet filter, the filter length, and the position of the coefficients. The convolution involved in reconstruction makes this process virtually impossible do without introducing unwanted artifacts.

Extending the sample to an arbitrary length is also non-trivial. Unless all that is needed is extending it by a power of two, the inverse wavelet transform becomes much more complex. Extending the sample by a power of two is also unsatisfactory. The result is very similar to looping the original sample, but with a lot of added artifacts, which are the very things we would like to avoid.

2.7 Wavelet Packets

Wavelet Packets were seriously considered as a method for representing natural grains. They differ from wavelets in that at every decomposition step, the difference *and* average coefficients are further broken down. The results are particular linear combinations or superpositions of wavelets. They form bases which retain many of the orthogonality, smoothness, and localization properties of their parent wavelets [Wic94].

Wavelet packets seem to have a lot of potential: depending on the strategy used to find a suitable basis from the over-complete set of packets in a full wavelet packet transform, you can find the basis which represents the signal with the fewest non-zero coefficients. See Wickerhauser [Wic94], for example, for a discussion of various basis-finding algorithms.

A wavelet packet library was written in Java expressly to see if something

similar to Bar-Joseph's work [BJDEY+99] could be done with wavelet packets. Instead of interchanging bald wavelet coefficients, we would interchange the wavelet packets, which ostensibly would hold information about whole events, rather than sample-level information.

While efficiency of representation is important, there are some key problems that cannot be easily be overcome. First of all, normal wavelet packets are not shift-invariant. This makes comparison between different regions of the wavelet packet transform extremely difficult.

Another difficulty with comparison has to do with packet levels. Every packet is denoted by the order in which the average or difference coefficients have been further broken down, as shown in Figure 2.4. However, there is no guarantee that all portions of the signal will be represented on the same packet level. If we have a packet that is denoted by ADADDDAD in Figure 2.4, it is not trivial to change it with another packet is denoted by DADD. They are different sizes, and have to interact with different packets in order to be properly put through the inverse wavelet packet transform. This makes switching positions of packets very difficult.

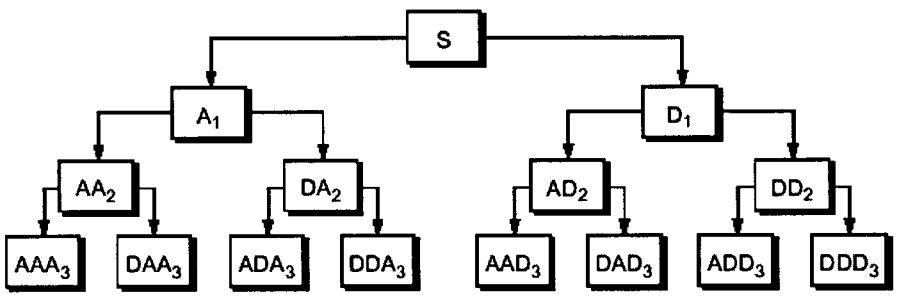

Figure 2.4: Wavelet packet decomposition

Some artificial means of constraining the representation could be taken, such as limiting the result to be all on one level of the wavelet packet tree. However, this seriously undermines the whole point of finding the best basis, as the number of coefficients increases by a power of two each level.

Related literature supports the problems listed above. In [WW99] Wickerhauser notes that the best basis algorithm is not well suited for isolating phonemes, since there is no reason for phonemes to even "begin" and "end" at dyadic points. Packets are thus not guaranteed to represent entire, re-arrangeable events.

Wickerhauser instead uses a segmentation algorithm to split up the time axis of the unprocessed signal. The segmentation algorithm measures the instantaneous frequency at discrete points, and places segmentation points at places where this changes. [2]

Despite the problems raised above, there have been attempts to use wavelet packets to aid in signal classification. However, surmounting the above concerns seems to take up any advantage over regular wavelets. For example, Sarikaya [SG98] has proposed an alternate version of his paper [SG97] discussed in 2.5.1, but using wavelet packets instead of wavelet difference coefficients. This involves characterizing each window by subband features derived from the energy of wavelet packets. There results, however, were not different enough from their earlier, wavelet-based approach for us to change our algorithm.

Delfs and Jondral [DJ97] use the best-basis algorithm for wavelet packets to characterize piano tones. They use a specialized, shift-invariant discrete wavelet packet transform to improve classification. The packet coefficients in the best-basis whose energies exceed a threshold are normalized, then compared to piano tones in a database which have been similarly analyzed. The euclidean distance determines the success of the match. This system was used for identification only, not resynthesis. Their results indicate that these specialized wavelet packets do not seem to offer any advantage over simple discrete Fourier transform features.

[2]The algorithm was supposed to published in another paper, but unfortunately, it was not. To my knowledge, unpublished copies are not available either.

2.8 Granular Synthesis

There is a long history in the electro-acoustic music community of arranging small segments of sound to create larger textures. Granular synthesis, pioneered by Curtis Roads [Roa88] and Barry Truax [Tru88] [Tru94], is a method of sound generation that uses a rapid succession of short sound bursts, called *Granules* that together form larger sound structures. Granular synthesis is particularly good at generating textured sounds such as a waterfall, rain, or wind. The grains in this case are taken as portions of a larger sound sample which can be specified to the algorithm. The sound sample itself has a large influence on the result of the granular synthesis, but since it can be specified by the user from anywhere, it is impossible to facilitate easier interaction with this parameter.

Curtis Roads describes granular synthesis as involving "generating thousands of very short *sonic grains* to form larger acoustic events"[Roa88]. A grain is defined as a signal with an amplitude envelope in the shape of a bell curve. The duration of a grain typically falls into the range of 1-50 msec. This definition puts granular synthesis entirely into the realm of new sound creation rather than manipulation which preserves the original perceptible properties of the sound. He likens granular synthesis to particle synthesis in computer graphics, used to create effects such as smoke, clouds, and grass.

Samples from the natural world have been used in granular synthesis, most notably by Barry Truax [Tru94]. He creates rich sound textures from extremely small fragments of source material. The scheme relies on a grain attack and delay envelopes to eliminate clicks and transients. The primary goal is time-shifting: drawing out the length of the sample to reveal its spectral components as a compositional technique.

There has been work done on extracting grains from natural signals and recombining them with *phase alignment* [JP88]. Phase alignment is used because with such small grains, the method of joining them has a large effect on the resulting

sound. This strategy helps avoid discontinuities in the waveforms of reconnected grains. Phase alignment works for both periodic and noisy signals. However, it is primarily a way to alter the original signal, for instance for time-stretching, by combining parts of the signal in various ways.

Gerhard Behles uses another method [BSR98] with *pitch markers*, which reference each pitch period (the inverse of the local fundamental frequency.) The onset of the source sound excerpt is quantized to the closest pitch marker. This method is less computationally expensive than the one outlined by Jones. It has trouble, however, with inharmonic signals.

Granular synthesis with natural signals deals with arbitrary extraction of grains from the original sample, without regard to what is going on in the signal. The majority of granular synthesis literature refers to it as a compositional approach with no intention of being perceptibly similar to the original sound.

2.9 Concatenative Sound Synthesis

Also primarily working in the electro-acoustic music community, Diemo Schwarz has developed the CATERPILLAR system [Sch00], which uses a large database of source sounds, and a selection algorithm that data-mines these sounds to find those that best match the sound or phrase to be synthesized.

The first step of audio segmentation is not discussed in Scharz' paper, readers are instead directed to a thesis [Ros00] available only in French. Segments are characterized by acoustical attributes, such as pitch, energy, spectrum, spectral tilt, spectral centroid, spectral flux, inharmonicity, and voicing coefficients. For each feature, low-level signal statistics are calculated, which are then stored in the database as keys to the segment.

Because of the complexity and number of the features measured for each sound segment, this system is not meant to be real time. Neither is it intended to be automatic: every segment is individually chosen by the user. This makes it inap-

plicable for our goals, but does show the breadth of applications that concatenation of audio segments can address.

2.10 Physically-Based Synthesis

Kees van den Doel's method for audio synthesis [vdDKP01] via modal resonance models can be viewed as a kind of resynthesis, where the sound's physical properties are estimated and used for resynthesis. It has been used to create a diverse number of sounds, such impacts, scraping, sliding and rolling.

The modal model $M = (f, d, A)$ consists of modal frequencies represented by a vector f of length N, decay rates which are specified as a vector d of length N, and an $N \times K$ matrix A, whose elements a_{nk} are the gains for each mode at different locations. The modeled response for an impulse at location k is given by

$$y_k(t) = \sum_{n=1}^{N} a_{nk} e^{-d_n t} \sin(2\pi f_n t) \tag{2.7}$$

with $t \geq 0$ and $y_k(t)$ is zero for $t < 0$. Geometry and material properties, such as elasticity and texture, determine the frequencies and damping of the oscillators. The gains of the modes are dependent on the location of contact on the object.

For simple objects geometries, parameters for the modal model can be derived, but for most realistic objects, where derivation becomes untenable, we can directly estimate location-dependent sound parameters. Josh Richmond [RP00] has developed a method to do this using the telerobotic system ACME [PLLW99]. The object in question is poked with a sound effector over various locations, creating a map of sounds from which it is possible, using an algorithm developed by van den Doel, to estimate the sample's dominant frequency modes. These modes are fed into the modal model algorithm to produce resynthesized sounds.

This technique is very effective for creating and manipulating models of the sound properties of everyday objects. However, so far it has primarily been applied

to contact sounds, where the objects making the noise can be modeled or measured satisfactorily. For background sounds, such as wind, animal cries, and traffic noises, it is possible to use recorded samples to estimate the modal model parameters. However, tweaking these parameters for these types of sounds, which usually have a high degree of variation, is time-consuming and difficult. Producing a sufficiently randomized stream with this technique is also non-trivial, and the method to achieve this for one type of environmental sound wouldn't necessarily be transferable to others. So while this technique is very adept at synthesizing contact sounds, we think there is room for a resynthesis system specifically for background, environmental audio.

Chapter 3

Our Early Attempts at Segmenting Audio Samples

Part of the genesis of this thesis was the paper by Bar-Joseph et al. [BJDEY+99] detailing their attempt at automatic granulation of a sample using wavelets. A description has been given in Section 2.6. Interested by its promise, we implemented the algorithm detailed in the paper. Matlab was a convenient language to use in this case because the wavelet toolbox [MMOP96] has all of the wavelet functionality we needed to implement Bar-Joseph's algorithm.

Although the paper claims that they achieved "a high quality resynthesized sound, with almost no artifacts due to the random granular recombination of different segments of the original input sound", our results were disappointing. Some implementation details were omitted from paper, so some of the algorithm had to be guessed and re-invented. There is also no samples available on the web to verify their claims.

A note about terminology, taken from the Bar-Joseph paper: if we consider a Mallat tree as defined in Section 2.4.1 turned upside-down, we end up with a binary tree with the first level of difference coefficients of the wavelet transform on the very bottom. In this view, *predecessors* refer to the adjacent wavelet coefficients to its

left on the same level, and *ancestors* are coefficients in higher levels that have, in the binary tree sense of the word, this coefficient as a *child.*

The object is to mix up the wavelet coefficients in this binary tree representation in a judicious manner, so that when the inverse transform is performed, the result sounds like a variation of the original.

To replace a wavelet coefficient on a given level of the binary tree representation, we examine the node's predecessors and ancestors. Other wavelet coefficients from the same level are considered as candidates to replace it if they have similar predecessors and ancestors. This similarity is measured within a threshold, which is specified by the user.

In the naïve Bar-Joseph algorithm, all the neighbours of a coefficient are taken into account when deciding what to switch. Doing this over the whole transform results in a quadratic number of checks. This makes the algorithm impractically slow, so the authors suggest to limit the search space to the children of the candidate set of nodes of the parent, which would greatly decrease the search space.

Our implementation found that the candidates were almost always only the immediate neighbours, because they had the most ancestors and neighbours in common with the node to be replaced. The node adjacent to the one to be replaced has all the same neighbours except itself, and all of the same ancestors as well.

Almost never were there any nodes other than the immediate neighbours being considered as candidates, and almost never were the immediate neighbours *not* considered. This was the case no matter what the threshold was set to. It was either only the immediate neighbours considered, or all of nodes in the entire level. This explains some of the effects we observed in our results, which tended to sound similar to the input sample, but with slight artifacts from the wavelet resynthesis. There were no large-scale reorganizations of the sample, only local changes.

To promote more large-scale changes, we only allowed shuffling of the coefficients in the higher levels of the inverted Mallat tree, ones that represented more

than 10 ms of audio. The rest of the tree was re-organized according to the last level scrambled. For any coefficient on the last level scrambled, not only is that coefficient taken from somewhere else on the same level, but also that coefficient's children, and the children's children, and so on until the end of the tree. Thus for any coefficient on the last level processed, it and all of its descendants are moved en masse.

However, allowing coefficient shuffling only at higher levels is only a partial solution, because the inverse wavelet transform will work on at least the number of samples of the length of the filter at one time. There is really no direct analogy to the *parents* and *children* of a binary tree, because the even the smallest filter has more than two coefficients. Because of the convolution step that happens between the filter and the coefficients in the inverse wavelet transform, the effect of any one coefficient is spread over a number of coefficients equal to twice the filter length. This almost invariably leads to artifacts, because there is no guarantee that the inverse wavelet transform will produce a smooth signal from these altered coefficients. For applications such as computer music, perhaps these artifacts are desirable, although they won't be predictable in any useful way. For applications such as environmental sound production, it is a definite drawback.

Despite our concerns, the results did have some promise. While there were artifacts, the sound was recognizable. The artifacts were mostly due to "chattering," with the same portion of the sound repeated a few times without enough of a decay envelope. At this point, we thought there was potential to solve these problems, so the decision was made to try a Java implementation that streamed audio in real time. This required a Java version of the wavelet transform, which was then implemented, and is discussed in 4.5.

3.1 A Streaming Granular Synthesis Engine

To achieve real-time performance, we computed possible candidate for replacement for every coefficient beforehand, so that when generating audio, all that had to be done was to construct a Mallat tree from the pre-computed candidate sets, then do an inverse wavelet transform. The word "streaming" is used loosely – the smallest unit was the whole wavelet tree, which was the same length as the input sound.

Streaming audio in this way gave mixed results. Although we managed to get the audio out in real-time, because of the local nature of the granulation, the sound wasn't adequately changed to allow for seamless combination of finished versions in the way the paper described. Although events were mixed, there was no guarantee that the end of one and the start of another would flow well.

The root of the problem was still the granularity of the coefficients. Their interdependence made it impossible to move audio events around cleanly. We needed a better way to characterize the events of the signal. Wavelet Packets were briefly considered, then rejected for reasons detailed in Section 2.7.

After abandoning the wavelet packet transform, we hit upon the idea of using a speech recognition algorithm that used the wavelet transform in the analysis step. Because we valued the fidelity and similarity of the input sound to the output, using the wavelet transform for analysis only gave us much more satisfactory results.

Chapter 4

Segmentation and Resynthesis

In this chapter, we describe the steps towards an implementation of a resynthesis engine based on natural grains. First is the segmentation algorithm, which analyzes the input sound signal and outputs a series of graded points in the sample that are most appropriate to segment around. The user can then fine-tune the default threshold to determine the total number of segments in the sample. Next is the method to grade how segments fit together with each other for the purposes of playback. We then describe our implementation.

4.1 Segmentation

The core of our segmentation algorithm is a modified version of the method described in the paper by Alani and Deriche [AD99] described in Section 2.5.2, in which a signal is divided into frames and analyzed with the wavelet transform. An input audio signal is broken into small frames of 1024 samples each, with an overlap of 256 samples. Sound samples are input as CD-quality 44.1 kHz, so each frame is approximately 23.2ms long. To provide metrics for what is happening during the length of each frame, six levels of the wavelet transform are computed for each frame.

An additive information cost function is then computed on each level of difference coefficients. The function currently used is the sum of $u^2 * log(u^2)$ for all

non-zero values of u, where u ranges over all difference coefficients in one level of the wavelet transform. This function is a measure of concentration, i.e. the result is large when the elements are roughly the same size and small when all but a few elements are negligible. A simpler function that sums the absolute values of the sequence has also been tried, with similar results.

A measure of correlation between corresponding wavelet levels across adjacent frames is then used to map the local changes in the signal. For this we use the same function as 2.6. Equation 4.1 gives a slightly more general version, giving the strength of transition between frames a and b:

$$D(f_a, f_b) = \sum_{i=a-1}^{a} \sum_{j=b}^{b+1} \sum_{k=1}^{6} (X_{i,k} - X_{j,k})^2 \tag{4.1}$$

Again, k refers to the wavelet level, i and j are frame numbers, $X_{i,k}$ and $X_{j,k}$ are the energies of wavelet difference coefficient levels. Only like levels are compared, and the differences between them are added up to obtain an overall difference between frames.

When this calculation is done for each frame (minus the first and last two) in the signal, the result is one number per frame which represents the degree of change between a frame and its immediate neighbours. The numbers are represented as an array, and only need to be calculated once per sound sample.

Alani and Deriche used this method to find the points where the distance measure is highest, in order to separate phonemes. We are not trying to 'understand speech', however. Isolating phonemes is not critical to our application. Rather, we would like to segment on the granularity of a syllable, where transitions are much more pronounced and the boundaries more amenable to shuffling.

A simple alteration to their algorithm that makes it more suitable for our goals is to look for the points which have the *least* difference between frames, instead of the greatest. With respect to the amplitude envelope, these points are more likely to be in the troughs of the signal between relevant portions, rather than in

the middle, or in the attack portion.

Another change is that we normalize the energies before calculating correlation. This is done by dividing each energy in a frame by the sum of energies in that frame. This focuses the correlation on the differences in strength of bandwidths between frames. This was not done in the Alani's version, but we consistently get smoother transitions after normalization.

For every frame boundary, we now have a number representing how similar its neighbours are on either side. To segment the sound into grains, we compare each of these numbers to a threshold. Those lower than the threshold are taken as new grain boundaries. We thus favour splitting the signal up at the points where there is little change, and keeping together parts of the signal where there is a relatively large amount of change.

We need to ensure a minimum grain size of more than 40ms so that we do not have grains occurring at a rate of more than 20Hz, the limit of frequency perception. So we only consider the point with the minimum distance measure compared to its two neighbours, which means that if two grain boundaries occur in two adjacent frames, one of them is ignored.

4.2 Grading the Transitions

The segments boundaries derived from the above approach represent the locations in the signal where it changes least abruptly. The degree of change is given by the result of the difference algorithm in Equation 4.1. Our final aim is to re-create randomized versions of the signal that retain as many of the original characteristics as possible. The next task, then, is to determine which of the grains flow most naturally from any given grain.

To enumerate the most natural transitions between grains, the grains are compared against each other and graded on their similarity. This is done in the same way as we calculated the original grains from the sample. To calculate the

transition between grains A and B, the last two frames of A are fed into the four-frame Euclidean distance algorithm of Equation 4.1, along with the first two frames of B. The lower in magnitude the result, the smoother the transition will be between the two grains.

4.3 Resynthesis

By taking the last two frames of each grain, and comparing them with the first two of all other grains, the similarity metric allows us to construct probabilities of transition between each and every grain. These probabilities are used to construct a first-order Markov chain, with each state corresponding to a natural grain. The next to be played is chosen by a random sample of the probabilities that have been constructed from the measure of how well the end of the current grain matches with the beginnings of all the other grains.

Probabilities are constructed by employing an inverse transform technique which uses the match scores for each grain as observed values for a probability density function (pdf). The smaller the result of Equation 4.1, the smoother the transition between the two windows on either side. We would like higher probabilities for smoother transitions, so we take the inverse of each transition score to orient the weights in the favour of the smaller scores.

We don't always want the probabilities of choosing the next grain to be dependent entirely on the smoothness of transition. Randomness is sometimes as important a consideration as smoothness. Some probabilities might be much greater that all of the others, and so picked often enough for the repetition to be noticed. To allow for the control over the weighting differences, we add a noise variable C which will help even out the grain weightings.

Let $P_{ij} = 1/D(i,j)$, indicate the likelihood that grain i is followed by grain j . We can convert this to a probability p_{ij} by normalizing as follows:

$$p_{ij} = \frac{P_{ij} + C}{\sum_{j=0}^{n} P_{ij} + nC} \tag{4.2}$$

where n is the number of grains. C denotes the constant noise we want to add to the distribution to give those with smaller similarities more of a chance to be selected. This number can be changed interactively to alter the randomness of the output signal.

We now construct a cumulative density function (cdf) from the pdf which gives us a stepwise function from 0 to 1. This is sampled by taking a random number from 0 to 1 and using it as the index to the function. The desired index can then be found using a binary search for the interval the random number lies between, and using that step of the cdf to index our map.

Once we have the transition probabilities, resynthesis is as simple as choosing what grain will be played next by random sampling from the empirical distribution p_{ij}. In this way, the smoother the transition between current grain and the next, the higher the probability that this grain is chosen as the successor. A high noise variable C flattens this preference, but never eliminates it.

4.3.1 Cross-fading

Our algorithm works to match energies of wavelet bands between grains as best as possible. Since the energies are normalized before comparison, boundary amplitude changes are not given much weight in our resynthesis choices. Normally, this is not much of an issue because the algorithm prefers the "troughs" of the signal in which the amplitude is near its minimum.. For sounds samples where there are no such troughs, and to give an overall cohesiveness to the output, we cross-fade between each successive grain. A linear cross-fade of 2 frames (approximately 5ms) is used.

4.3.2 Thresholding

The user has control over how many grains a signal is split up into. A slider in the graphical interface changes the value of the threshold below which a frame boundary is considered a grain boundary. The threshold extremities are determined by the maximum and minimum values of all frame boundaries. Between them, the threshold slider sacrifices control at the high end to obtain fine-tuning over the first few grains. This is important because there is a perceptually much bigger difference between changing the value when there is only 5 grains compared to 50 over the same sample. An exponential function is thus used to control the values of the threshold slider. There is a default threshold value which is currently set at 25% of the total possible number of grains.

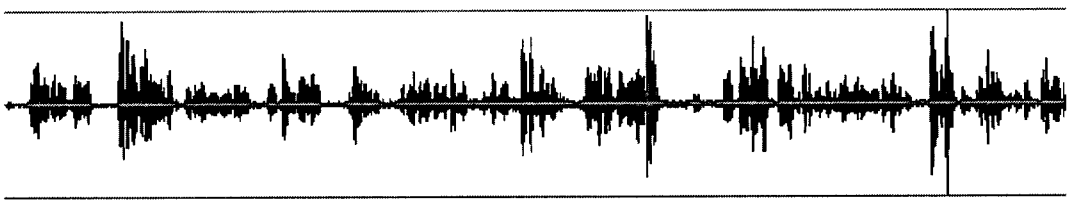

Figure 4.1: Input waveform

Figure 4.2: Portion of output waveform

Figures 4.1 and 4.2 show an example of the transformation which is the result of resynthesis. Although the output waveform in 4.2 looks significantly different than that of 4.1, both use the exact same samples (except for the small cross-fades on grain boundaries). The samples in the output sound are just re-arranged, and sometimes repeated.

4.4 Implementation

Our resynthesis system is implemented in Java, and features a graphical interface to facilitate real-time interaction. By manipulating sliders, users can change the segmentation threshold, and the noise parameter, C of the grain selection process. Using JavaSound on any platform that supports it, such as Linux and Windows, all of this can be done in real-time with no signal interruptions on systems with a Pentium II processor.

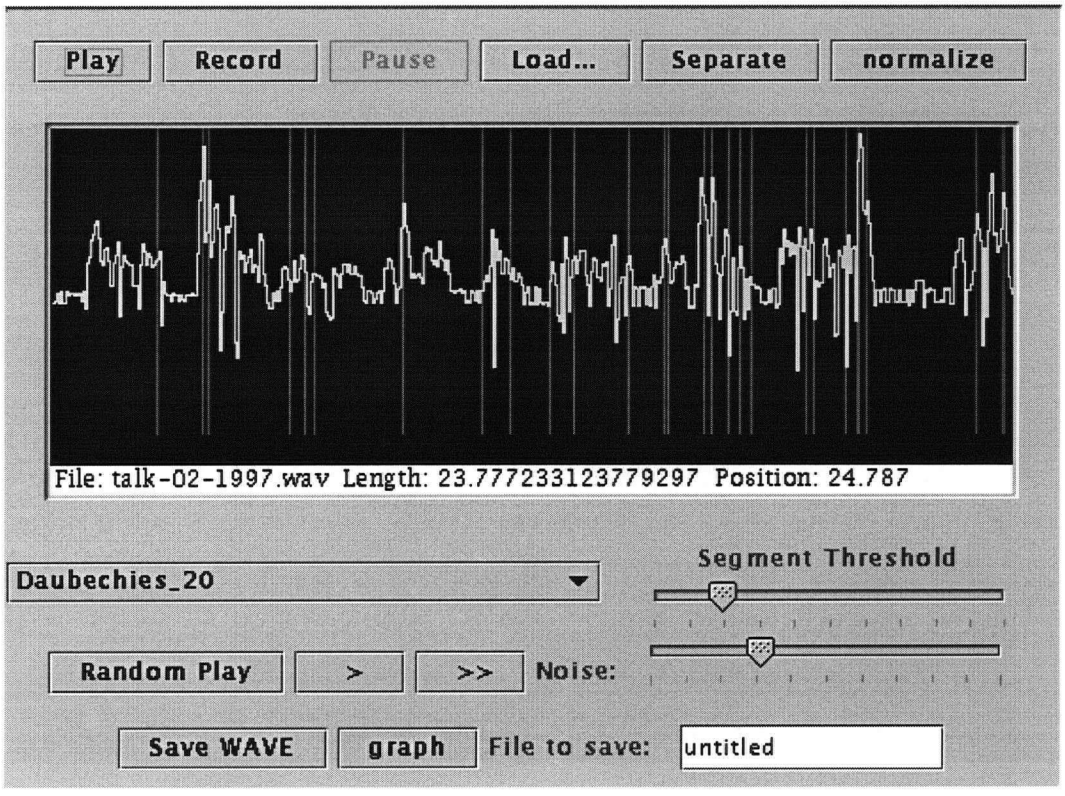

Figure 4.3: Segmented waveform and interface

Figure 4.3 shows the interface we have built to assist with segmentation. The top row of buttons, from left to right, are to play the original sample, to record a new sample, pause playback, load a new sound sample, and to separate the sample using the algorithm described above.

In the center is the current segmented waveform, with vertical lines slicing through it at the locations of the grain boundaries. Moving the threshold slider left or right causes more or less segmentation lines to appear, giving instant feedback as to the segmentation threshold. The noise slider affects the weights giving to next possible grains to be played. It is C in Equation 4.2.

4.5 Implementing the Wavelet Transform

To our knowledge, no publicly available version of the wavelet library exists for Java, so we decided to write our own. We used techniques described by Wickerhauser [Wic94], in the UBC Imager wavelet library wvlt [(or95] and in Dr. Dobbs [Cod92, Cod94]. All of these contained partial code examples written in C. To verify our implementation, we compared our results against those using the compiled Dr. Dobbs version. We also verified that the inverse transform gave the same results as the input data.

The wavelet filters available were all ported from the imager wvlt library. The filters ported include:

- The Adelson, Simoncelli, and Hingorani filter [ASH87]
- Filters by Antonini, Barlaud, Mathieu and Daubechies [ABMD92]
- The Battle-Lemarie filter [Mal89]
- The Burt-Adelson filter [Dau92]
- Coiflet filters [BCR91]
- Daubechie filters [Dau92]
- The Haar filter [Dau92]
- Pseudocoiflet Filters [Rei93]

- Spline Filters [Dau92]

For our purposes, usually the Daubechies 10 wavelet filter is used, although the other filters also work well. In general, there has been little work done on which particular filters to use for sound. More rigorous study is needed in this area. In our application, because the wavelet coefficients are summed into energies per level then normalized, the result is not very sensitive to the features of individual wavelet filters.

The wavelet library is designed to be a separate module from the rest of the implementation so that it can be used for other tasks. A wavelet packet library is also included in the Java package, which we plan to release to the community.

4.6 Real-time Considerations

To make our implementation adequate for real-time use, the segmentation step is done before playback. This is done by computing the differences between every frame, so that setting the threshold is just a matter of checking which frame scores fall below the selected threshold value. Those that do become grain boundaries. Segmentation data can be saved for a later date, so this step only has to be done once per sound sample.

This allows us to synthesize our audio output in real time with very little computation overhead, leaving plenty of extra computation power for other tasks on even the most average of desktop computers.

4.7 Segmentation/Resynthesis Control Interface

To facilitate construction of larger scale sound ecologies, a higher-level interfaces have been implemented. Per sound sample, there are controls for gain (amplitude) and pan (stereo left-right amplitude). There are three controls for each: start value, middle, and end, allowing for the sound to change over time. Because not every

sound should be played continuously, there are also controls for trigger (how often the sound should be played, given in seconds. For instance, a value of 10 would mean the sound would be activated every 10 seconds). Duration controls how long each of these higher-level segments are. Trigger and Duration both have associated values for controlling random variability. The Trigger value and its associated random element are used as the mean and standard deviation in a normal distribution random number generator. The pan and gain envelopes affect each of these higher-level segments individually.

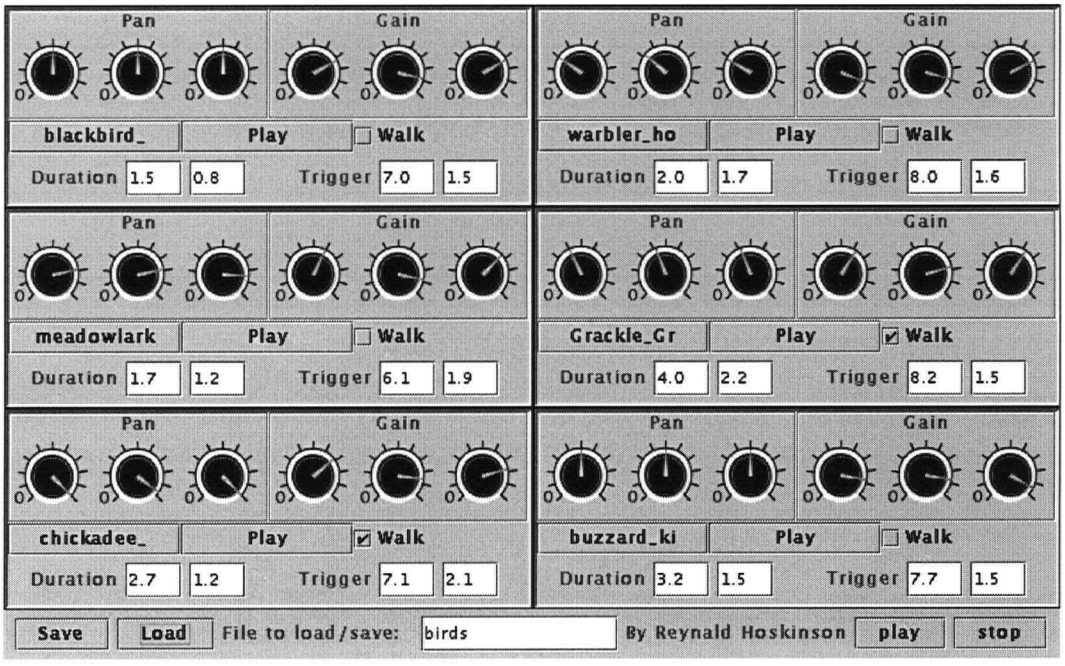

Figure 4.4: Segmented stream management interface

The sample-segmentation interface shown in figure 4.4 which allows for the control of how many natural grains the sample is chopped up into is still accessible by a button for each control. A "Random Walk" button places the sound in a continuous random walk through stereo space. "Play All" and "Stop All" buttons allow for overall control of all samples loaded at one time.

4.8 Preset Mechanism

Because the initial time to analyze the signal for segments can be relatively substantial, the interface also has the capability of saving presets for sounds that have already been analyzed.

Currently, all of the window transition values are saved, so that when the preset is loaded again, none of the wavelet transforms have to be done again. The threshold, noise and all of the interface settings such as pan, gain, etc. are also all saved. When a preset is loaded, all the user has to do is push "play" to hear the sounds at the same settings they were when the presets were last saved.

4.9 Discouraging Repetition

Favourable transitions between segments are given a higher probability of being chosen, so for segments that have very good compatibility with just a few others, and very low for the rest, the same sequence of samples can occur. For certain samples, this repetition of sections was recognizable, especially if the input signal was relatively short in duration. A high noise parameter helps by evening out the probabilities of the next candidate segments. To further prevent repetition, a transition that has just occurred has its weight reduced for further picks from this transition.

Discouraging repetition is accomplished as follows. For each segment s_1 that is played, we pick the next segment from weighted probability list of s_1. We then reduce the possibility this transition from s_1 will be picked again in the near future. This is done by associating with each segment a list of the 10 most recently chosen transitions from that segment. These 10 are themselves weighted with a higher value. The normal weights (the weights given by signal analysis) of these 10 recent-most transitions are then divided by the repetition weights, yielding new weights that are a fraction of their originals. Using this scheme, after enough time these last

10 played will revert to their original weights.

We chose to discourage the transitions rather than the actual segments because it kept our overall system of choosing segments intact. If we had discouraged actual segments instead, there are cases where our system would no longer be random, and would develop a recognizable repeating pattern. Consider the extreme case of a sample with less segments than the list of recently played segments that we do not play again. We would end up just playing a loop, with the next segment to be played almost always being the one we played least recently.

Chapter 5

Results and Evaluation

As example sound inputs to our algorithm, we have used samples taken from the Vancouver Soundscape [Tru99], and other natural recordings from a number of different environments. This provided us with a wealth of real-world samples, both pitched and unpitched, which were ideal for this algorithm.

5.1 User Study

To test the utility of our natural grain technique, we carried out a pilot user study on subjects taken from around the UBC Computer Science Department. We tested the hypothesis that a sound generated from our algorithm will be indistinguishable from the real audio sample it is taken from.

The seven samples used for the tests were:

- a series of car horns
- the sounds of some tree frogs
- crickets
- the sound of crumpling and tearing paper
- ambient sounds from a fish market

- the sound of a bubbling brook
- birds chirps in the forest.

Each sample was gone through by hand to pick an appropriate segmentation threshold. We chose threshold values that would ensure there was more than one segment in any snippet we played as tests.

There is a possibility of subjects comparing the sound events in the real sample versus the resynthesized sound, instead of evaluating whether the resynthesized sound is plausibly realistic. To minimize this, we didn't just use a snippet of a real sound and that same snipped resynthesized, because then the same events would always happen in both samples. Instead, the snippets were taken from two larger pools, constructed from original and resynthesized samples of much larger duration. The location of the snippet within the larger sample pools was chosen at random, the only condition being the start had to leave enough room in the pool to play the 4 second duration of the test snippet itself. New locations were chosen each time a snippet from a sound was played. This precaution preserves the intent of the algorithm: to produce sounds that appear to have been recorded from the same source as the original, but at different times.

5.1.1 Participants

Ten members of our department (9 males, 1 female) participated in the user study. All reported normal hearing. The participants were not paid for their time.

5.1.2 Experimental Procedure

The experiment used a two-alternative forced-choice design. Subjects sat on a chair in front of two speakers in an enclosed room. We told them that we would play a series of various environmental sound samples in pairs. Each pair would consist of a random section of the original sample, and a random section of a resynthesized

sound. They would be in no particular order, and there would be a number of different types of sounds. For each pair the subjects were instructed to identify the 'real' sample. They were told they could only listen to the samples once; no repetition was allowed. They were then given a practice sample different than those used in the test. Finally, the test began. There were three iterations of both orders real-synthesized and synthesized-real for each sample, for a total of 42 tests for each subject. The order of tests was random.

Subjects were not told that the tests would be symmetrical, that is, there would be as many with the resynthesized sound first as there was with the real sound first. Nor were they told the number of repetitions in the experiment, nor the nature of the resynthesis algorithm.

There were some potential issues with the approach we took to demonstrate the utility of our algorithm. For one thing, the subject always has the original sound to listen to next to the resynthesized version, and thus might be able to pick out idiosyncrasies with the resynthesized sound that might not be recognizable if the real sound was not played. However, on the flip side, if the users cannot statistically tell between the real sound and the resynthesized one in this test, it is a strong endorsement that the algorithm can produce realistic sounds.

5.1.3 Results

Figure 5.1 displays the correct scores per subject tested. There was substantial variation between subjects. Three out of the 10 scored above 70 %, 2 others above 60 %, 1 above 50 % and the other 4 below chance, as shown in figure 5.2.

We tested the null hypothesis that the subjects perform at the chance level (each response is a pure guess) for the 10 subjects. By hypothesis, the mean number of correct responses $\mu = 21$ and the standard deviation $\sigma = 3.24$. Using the normal

Correct Scores per Subject and Sample (each out of 6)							
Subject number	Car horns	Crickets	Paper	Market	Frogs	Stream	Birds
1	3	2	2	2	2	3	0
2	5	3	2	3	4	6	3
3	6	4	3	4	5	5	5
4	1	2	2	2	3	3	5
5	2	5	3	4	3	2	5
6	2	2	5	3	4	2	1
7	6	2	4	4	5	5	4
8	5	5	1	5	5	5	5
9	5	3	2	2	2	2	2
10	6	2	4	3	3	3	5

Figure 5.1: Correct scores per subject and sample. Each score is out of 6.

Subject	Total Correct	Percentage Correct	Standard Deviation
1	14	0.33333	1
2	26	0.61905	1.38013
3	32	0.76190	0.9759
4	18	0.42857	1.272418
5	24	0.56251	1.272418
6	19	0.45238	1.380131
7	30	0.71428	1.253566
8	31	0.73809	1.511858
9	18	0.42857	1.133893
10	26	0.61904	1.380131

Figure 5.2: Percentage correct answers per subject, over all samples.

approximation to the binomial distribution we conclude that we can reject the hypothesis with a two-tailed test at the significance level $\alpha = 0.05$ only if the sample mean is outside the interval $\mu \pm 1.96\sigma = [14.65, 27.35]$. Two subjects scored above this range (32, 31), and one below (14). However, the mean, 23.8, falls solidly within this range, so overall we cannot reject the null hypothesis.

5.1.4 Discussion

These tests showed that samples resynthesized with this algorithm are virtually indistinguishable from the originals. This demonstrates the utility of this process to

	Correct Responses
mean	23.8
max	32
min	14
std	6.268

Figure 5.3: Statistics for the number of correct responses

Percentage Correct, Per Sample	
car horns	0.683
brook	0.600
frogs	0.583
birds	0.583
market	0.533
crickets	0.500
crumpling paper	0.483

Figure 5.4: Percentage correct per sample, compiled over all subjects

create audio streams of indefinite length from samples of fixed length. It can also be used to create a number of variations of a fixed-duration sound sample.

The results also reveal something about which sounds are most effectively rendered with our algorithm. Figure 5.4 shows the fraction of correct answers per sample. There is substantial variation between samples, with the car horns being the most identifiable and the crumpling paper sounds the least.

In the case of the car horns, we can partly attribute the high success rate to the nature of the sample. A car horn is a pitched sound with a distinct attack, middle and end. Because it is so plain and in the foreground, it is easier to pick out idiosyncrasies when the horn is altered, which in turn makes it easier to identify the resynthesized sound. In this particular case, some subjects remarked the horn stopped or started too quickly to be considered normal. The segmentation/resynthesis process sometimes changed the original envelope of the horn enough to be noticeable. They were comparing not only the two samples played for them, but also each against their own personal idea of what a car horn should sound like.

Sounds of the market, on the other hand, depict the bustle and commotion of many people going about their routines. It is chaotic and layered, displaying much less temporal structure than the regular sound of a car horn. Without the larger temporal clues to give them away, the resynthesized market sounds were thus more difficult to identify.

Another low-scorer, the crumpling paper also does not contain as much temporal information as the car horns. Like the market, it is unpitched and irregular, which provides a much richer set of possibilities for segmentation. Pitch does not provide an inherent difficulty for our algorithm, but these types of sounds are usually accompanied by amplitude envelopes, which are a problem because of their temporal structure. This shows that our algorithm performs best on the sorts of sounds it was designed for: unstructured environmental sounds suitable as 'background' noises.

As it stands, sounds with temporal structures can only be handled by starting with an input sound that contains a number of the discrete structures, and then only segmenting between them. For instance, to achieve better car horn sounds, we could have only segmented between each toot of the horn. Another solution which would involve adding temporal information to our algorithm is detailed in the next chapter.

One confound that might exist for these tests is the segmentation threshold we chose for the individual samples. Each sample was gone through by hand, to ensure there was more than one segment in any snippet we played as tests. The threshold could not be too large, either, because then the segments would be to small, causing the resultant sound to differ perceptibly in timbre from the original. These two constraints still leave considerable to maneuver, so the notion of an 'ideal' threshold is a loose one.

Our design decision to give final say on the threshold to the sound designer allows for maximum flexibility. However, it also makes tests that limit the threshold to one value per sample to be necessarily limited in scope relative to the choices the implementation offers.

Overall, the subject's reaction to the resynthesized sounds was uniformly positive. They invariably commented that it was very difficult to distinguish our sounds from the originals, irrespective of their final scores.

Chapter 6

Conclusions and Future Work

6.1 Overview

This chapter will summarize the goals and results of this thesis, and will also outline some directions for further work and improvements.

6.2 Goals and Results

Our goal in developing the natural grain resynthesizer was to complement existing methods for generating audio through physical simulation of sounds with one that focuses on manipulating existing samples. Background sounds, such as birds in a forest, chatter in a café, or street sounds are currently beyond the scope of physical simulation, but are just as necessary to virtual environments, film, and other disciplines that value realistic sound environments.

In this thesis we extended the utility of sample-based audio resynthesis by providing a method to create randomized versions of samples that preserve perceptual qualities of the original. This allows users to create samples of indeterminate length from an input sound of fixed length, without having to resort to simple, deterministic looping. In a situation where the same sound sample is triggered in response to an event very often, one could create a number of variations of similar

duration to a sample, instead of repeating the original again and again.

Our implementation also provides an interface that makes it possible to easily mix together multiple streams of resynthesized audio to quickly create an integrated acoustic environment from separate elements.

The implementation can also be used for other purposes besides sound extension. Interesting effects can be achieved when we set the granularity to be very fine, in which case we achieve a sparse form of granular synthesis. Intriguing combinations and textures can also be generated by using a sample with many different heterogeneous sound sources present. The algorithm mixes them in unexpected and interesting ways, often in short bursts that are connected seamlessly with other short bursts from elsewhere in the signal to create new macroscopic structures.

Our main challenge was to preserve the perceptual characteristics of the original sound. There are many ways of creating new sounds with new timbres from input samples, such as granular synthesis, but much less work has been done on creating new samples that sound similar to those input. This is the key achievement of our work.

6.3 Future Work

While the system works well right now for a variety of applications, there are are many interesting extensions to the dynamic scrambling algorithm. In this section, we explore a few future directions for work.

1. More work could be done to automatically set the threshold to a reasonable value depending on the sample. Determination of automatic threshold values is tricky, because the optimum number of segments for a sample varies with the size of the sample and its inherent separability.

2. It would be useful not only to extract the components in the time domain, but also split the signal up into multiple simultaneous streams. For instance,

in a sample where there is simultaneously traffic noise and birds singing, we could extract the bird sound from the background and re-synthesize the two separately. Independent Component Analysis (ICA) [Cas98] could be a viable method of separating signals from one source.

3. Modifying the underlying signal through manipulation of the time-frequency domain can produce desirable and predictable changes in the perceived objects involved in the sound production. For instance, Miner and Cadell [MC97] have developed methods for altering the wavelet coefficients in the representation of a rain sample to change the perceived surface the rain was falling on. They could also change the perceived size of the drops, or their density. Since our algorithm already computes a version of the signal in the wavelet domain, this would be relatively efficient to implement.

4. Although it is not needed for the purposes of this thesis, adjusting the threshold slider in real time to change the grain size is also currently possible, but with more computational overhead. This functionality is useful for musicians and sound designers to create effects by continuously changing the grain size in real time.

   The generation of the first-order Markov chain is relatively expensive, and must be re-done after ever change in threshold. This is because until we know the threshold, we do not even know where the grains will be, so it is difficult to determine how well they match together. In practice, on a moderate length sound sample (less than 5 minutes), this is not a problem. However, longer samples could lead to thousands of grains. This becomes a problem because the time complexity of the algorithm is $O(n^2)$ to recompute the Markov chains, where n is the number of grains.

   For the purposes set out in this thesis, the existing implementation is adequate, but re-writing the code to pre-calculate segment relations is possible. We

would have to calculate every possible transition between segments for every possible threshold value, and be able to only consider those that are below the current threshold.

5. One of the most interesting of the possible extensions to this algorithm would be the incorporation of time information. This would allow us to produce new sounds with the same rhythmic structures as the originals. A possible way of accomplishing this would be to analyze amplitude, and use the resulting information in a hierarchical Markov chain. Entire sounds or just localized sections could be analyzed to obtain their large-scale amplitude envelopes. This information would be used to cut down the number of segments considered when choosing the next segment to play. The subset could be based on the criteria that the next segment must have an amplitude that matches the current portion of the larger pattern. Finer-scale choices between individual segments within the subset would still be done using the algorithm outlined above. Or, if subsets remove too much randomness, we could instead modify the entire weighting system used in resynthesis to reflect this new information.

   This approach would allow us to successfully segment a much broader class of sounds than is currently possible. Sounds with a high degree of temporal information would be particularly better handled. This approach would also create additional creative opportunities. For instance we could analyze one sound for its amplitude envelope, then use this information on a completely different sound. Since local decisions affecting sound quality would still be made normally as we have described above using local information, the sound would still be of very good quality, but with completely different (but recognizable) temporal characteristics.

6. In addition to the amplitude-following describe above, we might be able to implement something close to the beat-tracking interfaces offered in some

commercial packages for musicians. By analyzing the amplitude maps, we could estimate the rhythm of the sample, and allow the user to change it. The change would be brought about by exploiting an artifact of our algorithm. For rhythmic sounds, decreasing the segment size often increases the tempo of the sound. The exact cause of this is not known at present, but it could be exploited by offering it as a control to the user. With judicious use of amplitude maps, we could intentionally shorten the envelope of a particular sound by not picking as many segments in the sustain part of the sound, so the envelope becomes shorter. Since it is the attack that gives a sound most of its character, as described in a paper by Risset and Matthews [RM69], this would hopefully not change the perceptible characterization of the sound drastically.

7. For particular tasks that demand a high degree of control over which segments are combined, it would be useful to let the user manually combine segments. We could provide a sound-builder interface, where the user starts with one segment, then is provided with a menu with every other segment ordered according to goodness of fit. Or there could be other criteria besides goodness of fit that the user chooses to order the segments. The selection of segments offered could also be more sophisticated. For instance, the user could select segments from a whole database of sounds, called up according to some criteria. This takes us closer to the objectives of the CATERPILLAR system [Sch00].

8. Our aim of creating a perceptibly similar sound breaks down with extremely small samples (under 1-3 seconds, depending on the sample characteristics). On such a small amount of data, there is little possibility of having enough segments to produce quality output with an adequate number of segments: the segments become too small, and the result sounds more like granular synthesis. We could provide a system which analyzes the sound, and changes perceptible qualities like pitch, amplitude, and colour by small amounts to create the illusion of change or variation without radically changing the perception of

the sound. We would also have to choose the segment and resynthesis points carefully, using something like the amplitude-following technique described above.

This algorithm and its proposed extensions all aim to give the user an intuitive interface to sound design. We believe it is not enough to provide novel physical or graphical interfaces to sound synthesis engines; what is often needed, and rarely found in practice, are interfaces that reflect the underlying physical properties of a sound. When dealing with samples from the real world, as we have done, this involves developing methods for signal understanding, and manipulation techniques that preserve important aural properties.

Bibliography

[ABMD92] Marc Antonini, Michel Barlaud, Pierre Mathieu, and Ingrid Daubechies. Image coding using wavelet transform. *IEEE Transactions on Image Processing*, 2(1):205–220, 1992.

[AD99] Ahmed Alani and Mohamed Deriche. A novel approach to speech segmentation using the wavelet transform. In *Fifth Internation Symposium on Signal Processing and Its Applications*, 1999.

[AFG99] Maria Grazia Albanesi, Marco Ferretti, and Alessandro Giancane. Time-frequency decomposition for analysis and retrieval of 1-d signals. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, volume 2, pages 974–978, 1999.

[ASH87] E. H. Adelson, E. Simoncelli, and R. Hingorani. Orthogonal pyramid transforms for image coding. In *Visual Communications and Image Processing II*, pages 50–58, 1987.

[BCR91] G. Beylkin, R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms, 1991.

[BIN96] Jerry Banks, John S. Carson II, and Barry L. Nelson. *Discrete-Event System Simulation.* Prentice-Hall, 1996.

[BJDEY+99] Ziv Bar-Joseph, Shlomo Dubnov, Ran El-Yaniv, Dani Lischinski, and Michael Werman. Granular synthesis of sound textures using statistical learning. In *Proceedings of the International Computer Music Conference*, pages 178–181, 1999.

[BSR98] Gerhard Behles, Sascha Starke, and Axel Robel. Quasi-synchronous and pitch-synchronous granular sound processing with stampede ii. *Computer Music Journal*, 22:44–51, Summer 1998.

[Cas98] Michael Anthony Casey. *Auditory Group Theory with Applications to Statistical Basis Methods for Structured Audio.* PhD thesis, Massachusetts Institute of Technology Media Laboratory, 1998.

[CMW92] R. Coifman, Y. Meyer, and M. Wickerhauser. Wavelet analysis and signal processing, 1992.

[Cod92] Mac A. Cody. The fast wavelet transform: Beyond fast fourier transforms. *Dr. Dobbs Journal of Software Tools*, 17(4):16–18, 20, 24, 26, 28, 100–101, April 1992.

[Cod94] Mac A. Cody. The wavelet packet transform: Extending the wavelet transform. *Dr. Dobbs Journal*, April 1994.

[Dau92] Ingrid Daubechies. *Ten Lectures on Wavelets*, volume 61. Society for Industrial and Applied Mathematics, Philadelphia, 1992.

[DJ97] Christoph Delfs and Friedrich Jondral. Classification of piano sounds using time-frequency signal analysis. In *Proceedings of IEEE ICASSP '97*, pages 2093 – 2096, 1997.

[Dry96] Andrzej Drygajlo. New fast wavelet packet transform algorithms for frame synchronized speech processing. In *Proc. of the 4th International Conference on Spoken Language Processing*, pages 410–413, 1996.

[EC95] Karmran Etemad and Rama Chellappa. Dimensionality reduction of multi-scale feature spaces using a separability criterion. In *Inter. Conf. on Acoustics Speech and Signal Processing*, 1995.

[Gav88] W. W. Gaver. *Everyday listening and auditory icons.* PhD thesis, University of California in San Diego, 1988.

[Gib79] James Jerome Gibson. *The ecological approach to visual perception.* Houghton Mifflin, 1979.

[Han89] Stephen Handel. *Listening: An Introduction to the Perception of Auditory Events.* MIT Press, 1989.

[Han95] S. Handel. Timbre perception and auditory object identification. *Hearing (Handbook of Perception and Cognition 2nd Edition)*, pages 425–461, 1995.

[Hel54] H. L. F Helmholtz. *On the sensations of tone as a psychological basis for the teory of music.* Dover, New York, 1954.

[JP88] Douglas L. Jones and Thomas W. Parks. Generation and combination of grains for music synthesis. *Computer Music Journal*, 12:27–34, Summer 1988.

[KM88] Richard Kronland-Martinet. The wavelet transform for analysis, synthesis, and proceessing of speech and music sounds. *Computer Music Journal*, 12(4):11–19, Winter 1988.

[KT98] Dami'an Keller and Barry Truax. Ecologically-based granular synthesis. In *ICMC*, pages 117–120, 1998.

[LKS+98] T. Lambrou, P. Kudumakis, R. Speller, M. Sandler, and A. Linney. Classification of audio signals using statistical features on the time and wavelet transform domains. In *Proceedings of the IEEE 1998 International Conference on Acoustics, Speech and Signal Processing (ICASSP'98)*, volume 6, pages 3621–3624, Seattle (WA), May 12 - 15 1998.

[Mal89] Stephane G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-11(7):674–693, 1989.

[MC97] Nadine E. Miner and Thomas P. Caudell. Using wavelets to synthesize stochastic-based sounds for immersive virtual environments. In *Proceedings of the International Conference on Auditory Display*, 1997.

[MMOP96] Michel Misiti, Yves Misiti, Georges Oppenheim, and Jean-Michel Poggi. *Wavelet Toolbox User's Guide.* The MathWorks, 1996.

[MZ92] Stephane G. Mallat and S. Zhong. Characterization of signals from multiscale edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(7):710–732, July 1992.

[(or95] Alain Fournier (organizer). Wavelets and their applications in computer graphics. In *Siggraph 1995 course notes*, 1995.

[PK99] Stefan Pittner and Sagar V. Kamarthi. Feature extraction from wavelet coefficients for pattern recognition tasks. In *EEE Trans. On PAMI*, volume 21, pages 83–88, 1999.

[PLLW99] D. K. Pai, J. Lang, J. E. Lloyd, and R. J. Woodham. Acme, a telerobotic active measurement facility. In *Experimental Robots VI, vol. 250 of Lecture Notes in Control and Information Sciences*, pages 391–400, 1999.

[Rei93] L. M. Reissell. Multiresolution geometric algorithms using wavelets: Representation for parametric curves and surfaces, 1993.

[RM69] J. Risset and M. Mathews. Analysis of musical instrument tones. *Physics Today*, 22(2):23–30, 1969.

[Roa78] Curtis Roads. Automated granular synthesis of sound. *Computer Music Journal*, 2(2):61–62, 1978.

[Roa88] Curtis Roads. Introduction to granular synthesis. *Computer Music Journal*, 12:11–13, Summer 1988.

[Roa96] Curtis Roads. *The Computer Music Tutorial.* MIT Press, Cambridge, MA, 1996.

[Ros00] S. Rossignol. *Segmentation et indexation des signaux sonores musicaux.* PhD thesis, University of Paris VI, July 2000.

[RP00] J. L. Richmond and D. K. Pai. Active measurement and modeling of contact sounds. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, pages 2146–2152, 2000.

[Sch00] Diemo Schwarz. A system for data-driven concatenative sound synthesis. In *Proceedings of the COSI G-6 Conference on Digital Audio Effects (DAFX-00)*, Verona, Italy, December 2000.

[SG97] Ruhi Sarikaya and John N. Gowdy. Wavelet based analysis of speech under stress. In *IEEE Southeastcon*, volume 1, pages 92–96, Blacksburg, Virginia, 1997.

[SG98] Ruhi Sarikaya and John N. Gowdy. Wavelet based analysis of speech under stress. In *Proceedings of the 1998 IEEE Confrence on Acoustics, Speech and Signal Processing*, volume 1, pages 569–572, 1998.

[Sko80] M. Skolnik. *Introduction to Radar Systems.* McGraw-Hill Book Co., 1980.

[SN96] F. Strang and T. Nguyen. *Wavelets and Filter Banks.* Wellesley-Cambridge Press, Wellesley, Massachusetts, 1996.

[SSSE00] Arno Schodl, Richard Szeliski, David H. Salesin, and Ifran Essa. Video textures. In *Siggraph*, 2000.

[SY98] S.R. Subramanya and Abdou Youssef. Wavelet-based indexing of audio data in audio/multimedia databases. In *Proceedings of the International Workshop on Multimedia Databases Management Systems*, pages 46–53, 1998.

[TLS+94] B. Tan, R. Lang, H. Schroder, A. Spray, and P. Dermody. Applying wavelet analysis to speech segmentation and classification. In H. H. Szu, editor, *Wavelet Applications Proc. SPIE 2242*, pages 750–761, 1994.

[Tru88] Barry Truax. Real-time granular synthesis with a digital signal processor. *Computer Music Journal*, 12:14–26, 1988.

[Tru94] Barry Truax. Discovering inner complexity - time shifting and transposition with a real-time granulation technique. In *Computer Music Journal*, volume 2, pages 38–48, Summer 1994.

[Tru99] Barry Truax. *Handbook for Acoustic Ecology.* ARC Publications, 1978. CD-ROM version Cambridge Street Publishing 1999, 1999.

[vdDKP01] K. van den Doel, P. G. Kry, and D. K. Pai. Foleyautomatic: Physically-based sound effects for interactive simulation and animation. In *Computer Graphics (ACM SIGGRAPH 2001 Conference Proceedings)*, 2001.

[War99] Richard M. Warren. *Auditory Perception: A New Analysis and Synthesis.* Cambridge University Press, 1999.

[Wic92] Mladen Victor Wickerhauser. Acoustic signal compression with wavelet packets. In Charles K. Chui, editor, *Wavelets–A Tutorial in Theory and Applications*, pages 679–700. Academic Press, Boston, 1992.

[Wic94] Mladen Victor Wickerhauser. *Adapted Wavelet Analysis from Theory to Software.* AK Peters, Ltd., Wellesley, Massachusetts, 1994.

[WL00] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *Siggraph*, 2000.

[WS85] W. H. Warren and R.E. Shaw. Events and encounters as units of analysis for ecological psychology. In W.H. Warren and R. E. Shaw, editors, *Persistence and change: Proceedings of the First International Conference on Event Perception*, pages 1–27, 1985.

[WW99] Eva Wesfreid and Mladen Victor Wickerhauser. Vocal command signal segmentation and phoneme classification. In Alberto A. Ochoa., editor, *Proceedings of the II Artificial Intelligence Symposium at CIMAF 99*, page 10. Institute of Cybernetics, Mathematics and Physics (ICIMAF), Habana, Cuba, 1999.