SOUND ANALYSIS, MODIFICATION, AND RESYNTHESIS WITH WAVELET PACKETS

by

Ronald Heinz Gerhards

B.A.Sc., University of British Columbia, 1986

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF APPLIED SCIENCE in the School of Engineering Science

SIMON FRASER UNIVERSITY

November 2002

APPROVAL

Name: Ronald Heinz Gerhards

Degree: Master of Applied Science

Title of thesis: Sound Analysis, Modification, and Resynthesis with Wavelet Packets

Examining Committee:

Chair: Dr. Shawn Stapleton

_____

Dr. Jacques Vaisey, Associate Professor
School of Engineering Science, SFU
Senior Supervisor

_____

Mr. Barry Truax, Professor
School for the Contemporary Arts / School of Communication, SFU
Supervisor

_____

Dr. James Cavers, Professor
School of Engineering Science, SFU
Examiner

Date Approved: _____

ABSTRACT

The *Wavelet Packet Transform* (WPT) has many features that suggest its potential usefulness in the *computer music* field. Such transforms permit us to work with sound simultaneously from both time and frequency perspectives. We describe our research into applying the WPT to sound processing for the purposes of characterization, adding aesthetically interesting modifications, and granular synthesis. We work primarily with *environmental sounds* obtained through recordings of the world around us. These sounds do not necessarily originate from the "natural world", but are distinct from *music sounds*. This orientation means we are less burdened by the need to model the highly tonal aspects of most music sounds, and can focus our work on the more percussive, broadband nature of most environmental sounds. We use a fast multiresolution filter bank implementation of the WPT, and experiment with a range of sound modification and basis selection approaches.

Sound modifications that we have tried include time stretching, coefficient thresholding and filtering, cross synthesis, and equalization. They all exploit the somewhat separate control over the time and frequency dimensions we get through projecting the signal on to the time-frequency plane. We have found that our modifications produce aesthetically interesting effects, and further that a granular synthesis approach within the WPT framework is also of interest.

Since a WPT of significant depth creates many equal width subbands, we can often benefit from the derivation of a new basis that has less subbands of varying widths, and is better aligned with either the time-frequency content of the signal, or the human hearing process. Such "basis selection" produces a representation that in some way enhances our ability to modify the sound effectively. We have investigated several approaches, including our *Bi-rate Distortion Best Basis Algorithm* (BDBBA), where the deletion or quantization of coefficients and even entire subbands is permitted in order to obtain a suitable basis. Having a reduced number of subbands can be especially useful in real-time work, since our two hands need not be spread over too many subbands when imparting modifications on the sound. Several other basis selection methods are also considered, as well as a fixed basis partitioning that approximates the band spacings of *third-octave* graphical equalizers.

ACKNOWLEDGEMENTS

I am deeply grateful to both of my supervisors, Dr. Jacques Vaisey (senior supervisor) and Mr. Barry Truax, for all their help during the course of my research at SFU. The support I received from Dr. Vaisey in the signal processing area and from Mr. Truax in the computer music area have been instrumental in this research effort. I also thank the School of Engineering Science staff and administration for their flexibility and support in allowing the pursuit of such interdisciplinary work.

Finally, on a personal note, I thank my family for all the patience and support they have shown during this time.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

AGS	Asynchronous Granular Synthesis
BBA	Best Basis Algorithm
BDBBA	Bi-Rate Distortion BBA
BW	Bandwidth
CWT	Continuous Wavelet Transform
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DWPT	Discrete Wavelet Packet Transform
DWT	Discrete Wavelet Transform
EBBA	Entropy based BBA
FFT	Fast Fourier Transform
FIR	Finite Impulse Response (Filter)
FOF	Fonction d'Onde Formantique (formant wave-function synthesis)
FOG	FOF - Granular
FT	Fourier Transform
FWPT	Fast Wavelet Packet Transform
FWT	Fast Wavelet Transform
GSAMX	Granular Sampling (synthesis method)
GSX	Granular Synthesis (synthesis method)
GUI	Graphical User Interface
KLT	Karhunen-Loeve Transform
MSE	Mean Squared Error
OA	Overlap-Add (resynthesis)
OB	Oscillator Bank (resynthesis)
PR	Perfect Reconstruction
PTOB	Pseudo Third Octave Basis
PV	Phase Vocoder
QSGS	Quasi-Synchronous Granular Synthesis
SNR	Signal to Noise Ratio
STFT	Short Time Fourier Transform
TF	Time-Frequency
TPV	Tracking Phase Vocoder
VOSIM	Voice Simulation (synthesis method)
VQ	Vector Quantization
WPT	Wavelet Packet Transform
WT	Wavelet Transform

Chapter 1
Introduction

1.1 Overview

Time-frequency based transforms play an important role in the field of computer music. They are what allow us to work with a sound signal from both time and frequency perspectives simultaneously. Such transforms have traditionally been useful in studying the nature and makeup of sounds, and in facilitating the application of aesthetically interesting and novel modifications to specific sound signals. We are interested in a transform that is useful in working with environmentally based sounds, and in this thesis we look at the application of the *Wavelet Packet Transform* (WPT) to the analysis, modification and resynthesis of such sounds. The more familiar *Wavelet Transform* (WT) can be viewed as a special case of the WPT.

We are motivated to use the WPT for several reasons. The transform is inherently multiresolution, making it more suited to human psychoacoustics than such fixed resolution transforms as the *Short Time Fourier Transform* (STFT). It can be easily reconfigured to allocate time-frequency resolution in different ways through various basis selection approaches. Furthermore, efficient discrete time algorithms are available, and the transform basis functions are inherently time-localized without the introduction of a separate window function. In relation to granular synthesis approaches, the WPT can provide a unified framework within which the computer musician can operate. Signals may be transformed, modified and resynthesized, or, the resynthesis operation can be used stand-alone as a granular synthesis engine. For example the user could work by modifying pre-existing environmental sounds while also overlaying freshly synthesized sounds using the same wavelet packets as grains.

In our work we investigate several aspects of the WPT as it relates to computer music applications. We investigate it applicability as a transform in general, look at a series of sound modifications that can be performed as a result of the transform, and consider what bases can be derived from the complete tree of the initial WPT transform that might in some way be better suited to working with sounds.

To allow for flexible and convenient experimentation with the WPT, modification schemes, and basis selection approaches that we have researched, as well as to prototype how a user might work with them, we have developed a *Graphical User Interface* (GUI) based application to serve as a test-bed for this experimentation. This GUI proved highly useful, especially given the large number of potential parameters

and modifications associated with the transformation and manipulation of a given sound. A command line interface would have been impractical. Furthermore, the GUI does present the user with a useful general purpose environment in which to not just test different approaches, but to actually work with and modify sounds for its own sake.

1.2 Research Questions and the State-of-the-Art

In one sense, our research has been quite broad in scope. We have experimented with many aspects of the WPT. We have looked at several options in the application of the WPT from filter types to transform depths to boundary extension methods. We have developed and experimented with many modification approaches to sounds that take advantage in one way or another of the WPT. We have looked at a wide range of basis selection approaches to tailor the subband decomposition in some way to the potential needs of the user. And, we have experimented with the user interface, and how a user might practically work with these capabilities.

However, amongst these various avenues of research, two fundamental questions have driven the work, one general, and one more specific. First, we wanted to learn how useful the discrete WPT is when applied to the analysis, modification and resynthesis of environmental sounds for aesthetic purposes. We are not aware of any other research in this area thus far. Some work was done with the WT roughly ten years ago by Kronland-Martinet et. al. (Kronland-Martinet 1988, Boyer and Kronland-Martinet 1989, Kronland-Martinet and Grossman 1991, Kronland-Martinet, Guillemain and Ystad 1997), and later Evangelista released a series of papers involving a filter bank based *Discrete Wavelet Transform* (DWT) (Evangelista 1991, Evangelista 1997, Polotti and Evangelista 2001). However, neither appear to have worked with the more general WPT. Also, Wickerhauser (1991, 1994) claimed wavelet packets could be useful when applied to sound synthesis and modification but has not apparently followed up.

We feel research in this area is worthwhile because the WPT possesses characteristics that appear to make it well suited to the intended application. We listed these earlier: its multiresolution nature, availability of discrete efficient algorithms, inherent time localized properties, and its potential to act as a unified framework for signal modification and granular synthesis. Furthermore, since the WPT makes no assumptions about the nature of the sound being transformed, the framework is *non-parametric* and hence quite broadly applicable.

Second, we would like to know what useful bases can be derived from the WPT complete tree, why they

are useful, and how we can obtain them. Again we do not believe that this topic has been researched to any extent thus far. Regarding *best basis* schemes for the WPT, where a basis is derived based on the sound itself, only the scheme by Wickerhauser and Coifman (1992) is discussed in relation to sound, and then only in theory. Ramchandran and Vetterli (1993) look at an approach motivated by the application of data compression, however, it was not applied to finding a particular basis, nor was it motivated by the application of sound manipulation.

Regarding the derivation of *fixed bases* for the WPT, which are structured based on psychoacoustic considerations rather than the makeup of any particular sound, the research that has been conducted is focussed primarily on the data compression application. The WT basis is one possible fixed basis that can be derived from the complete tree of the WPT, and this basis is of course prominent in its own right. Regarding other possible fixed bases, critical band theory motivates the derivation of a basis that maps to the critical bands, and we have derived such a mapping in our work.

We believe this second question regarding basis selection is worthwhile because an automated approach to finding a useful basis that reconciles the users needs for simplicity, representational accuracy, and suitability for manipulation would be a powerful tool. It would quickly put the sound in a form that allocates time-frequency precision in a way suited to the sound and/or to the manipulations applied to it, thereby making it more workable, analyze-able, storable and quickly resynthesize-able. This process is in contrast to finding such bases via a manual search over the space of possibilities. For example, even for a *depth-4* transform, there are over 65000 possible bases obtainable simply by merging various combinations of the nodes of the complete tree.

1.3 Document Outline

The remainder of our thesis is structured as follows. In *Chapter 2 Time-Frequency Methods in Computer Music* we provide a brief history and overview of time-frequency related methods in the computer music field. This will include reviewing such approaches as the *Phase Vocoder* (PV), which uses the *Short Time Fourier Transform* (STFT), the *Wavelet Transform* (WT), and *Granular Synthesis*.

In *Chapter 3 Wavelet Packet Transform Fundamentals* we provide necessary background on the wavelet packet transform, including its discrete implementation using multiresolution filter banks. This chapter also includes a description of two prevalent methods of basis selection, one of which acted as a foundation for the development of our own method.

The next two chapters describe the specifics of our own work. In *Chapter 4 Wavelet Packets Applied to Sound* we describe our methods of sound modification using the WPT, followed by a description of the basis selection approaches developed. This chapter starts with a discussion of the types of sounds we have targeted in our work, namely environmental sounds. *Chapter 5 GUI Description and Testing Results* then describes the results of our experimentation with the ideas in *Chapter 4*, including a description of the GUI. We also cross reference a series of example audio clips generated to demonstrate the various test results.

Finally, *Chapter 6 Conclusions* describes our conclusions, key contributions, and suggestions for possible future work in this area.

Chapter 2
Time-Frequency Methods in Computer Music

The *wavelet packet transform* (WPT) is part of a large class of signal processing approaches that we refer to as *time-frequency* (TF) methods. TF methods have been successfully applied in the field of *computer music* for many years. In this section we provide some background on TF methods to set the stage for our own application of the WPT. Specifically, we list the reasons for using TF methods, discuss some important physical and psychoacoustic constraints related to their use, and provide a brief overview of some of the more well-known TF approaches.

In our overview of TF techniques we have included two categories. The first category we refer to as *analysis based*. This set includes approaches like the *phase vocoder* (PV) and *wavelet transform* (WT), where underlying the entire scheme is a transform or filter bank structure that allows for the transformation of a signal into the TF domain, followed by some form of processing or parameter extraction in that domain, and then the resynthesis of the TF domain result.

The second category we refer to as *granular* based methods. These generally have a strong time-frequency orientation, but are *synthesis-only* techniques that do not rely on an analysis phase or any transform/filter bank framework.

Our approach with the WPT shares similarities with both categories. Like the analysis based methods, it is based on a transform that allows for the analysis and resynthesis of a pre-existing signal. Like the *grains* in granular approaches, the *basis functions* of the WPT are time limited entities that have both a *time* (length, shape) and a *frequency* (cycles per function, shape) aspect to them, giving us control over both the time and frequency aspects of the synthesized signal. As such, our approach can serve as a *unified framework* within which to do not only analysis-resynthesis, but also synthesis in a more granular vein. As De Poli and Piccialli (1991, page 187) state, “Analysis and synthesis systems such as wavelets or the short-time Fourier transform (STFT) supply a local representation by means of waveforms or grains multiplied by coefficients that are independent from one another. This provides a theoretical foundation for granular synthesis”. We have chosen to use wavelet packets instead, which provide us with a much more flexible partitioning of the time-frequency plane than that offered by these other transforms.

2.1 Transforms versus Models

Here we attempt to clarify some concepts and terminology surrounding the notions of *transforms* and *models.* This process will help our understanding in subsequent sections.

The notions of *transform* and *model* should be kept distinct. For example, the *Fourier Transform* (FT) has been used with great success as a *transform* to aid in the *modeling* of sounds as collections of harmonically related sinusoids. This model of sound is generally attributed to Helmholtz (1863). The model is the basis for a widely used synthesis approach called *additive synthesis*. There are many versions of additive synthesis, the more elaborate of which allow for continuous amplitude and frequency modulations in each of the sinusoids, such as the *Phase Vocoder* (PV).

Our work looks at the application of a different transform, the WPT. The degree to which this *transform* is also a good *model* of sound will depend largely on the particular sound it operates on. For the sounds we are primarily interested in, which we refer to as *environmental sounds*, we believe the WPT is indeed a reasonable model. Even when it is not, however, the WPT can still provide a valuable way of partitioning a sound so that we can work with it and resynthesize it. In other words, it is still in this case valuable as a transform. In Sayood (2000) they state ‘we can see that decomposing a signal can lead to different ways of looking at the problem of compression’. We believe that the same holds for the problem of sound *analysis*, *modification*, and *resynthesis*.

Note also that when viewed as a model, the WPT is *non-parametric*. Its form is not *parameter* based, and therefore it is not designed to map to a particular type of sound. Non-parametric models tend to be more widely applicable, while parametric models are effective primarily at modeling the sounds they were designed for. Typical *vocoders*, for example, which are designed to efficiently model and compress voice signals, do not model music based signals very well.

We can also model the human hearing process. A given transform, by virtue of how it partitions a sound, may coincide more or less closely with our model of human hearing. As such, hearing can be a consideration in transform choice. *Lossy* music compression exploits our knowledge of the hearing process. Unlike voice compression, where an effective model of sound production exists, there is no single source model in the case of music. But since most listeners are equipped with highly similar hearing responses, music compression seeks to compress sound in places where the compression will not be heard.

In our descriptions later in this chapter of some of the prevalent time-frequency based methods, we will

find that some are nearly identical with a given transform, such as the *Short Time Fourier Transform*, while others are more related to a particular model of the composition of sound, such as the *Sonogram* or *Phase Vocoder*. In the latter category, a transform may not even be used for the analysis and/or synthesis of the sound.

One point of clarification regarding the term *analysis*. It will be used to signify either the act of transforming the sound signal into the TF domain, or the act of further analyzing the result of the transform for the purposes of modeling or parameter extraction. The intended meaning should be clear from context.

2.2 Motivation for Time-Frequency Methods

Most music and sound signals of interest possess frequency characteristics that change over time. Therefore the representation of such signals simultaneously as functions of *time* and *frequency* will provide us with an intuitive and usable format that coincides well with those aspects of the signal we are interested in. This approach has advantages over working with a signal either solely in the time or frequency domain. The *time-frequency* (TF) representation facilitates analysis and study of a signal by showing its evolution over time. The TF representation also allows for targeted manipulation of the signal at chosen time and frequency locations, thereby giving the computer musician a high degree of control in modifying sounds[†].

An early application of a TF method is the *sonogram*, which is used for speech analysis. Plotted in two spatial dimensions, the energy of the speech signal is represented by the color or grey-scale intensity level at a given point in the plane. Time is usually represented along the horizontal axis, and frequency along the vertical.

Fig.*2-1* shows an example of a signal in all three domains: *time*, *frequency*, and *time-frequency*. The time domain representation of the signal is given at the top of the figure. The spectral energy plot at the left of the figure is based on the squared magnitudes of the *Fourier Transform* coefficients of the signal. The time-frequency plot occupying the rest of the figure shows the squared magnitudes of the *Short Time Fourier Transform* (STFT) coefficients over time and frequency. This latter plot is an example of a *spectrogram*, which is analogous to the *sonogram* mentioned earlier[††].

[†] Note that a time-frequency based view is used at the level of the *musical score*. Frequency, or pitch, is charted as a function of time as the score is traversed.

[††] Methods other than the STFT can also be used to obtain the time-frequency information displayed in *spectrograms*.

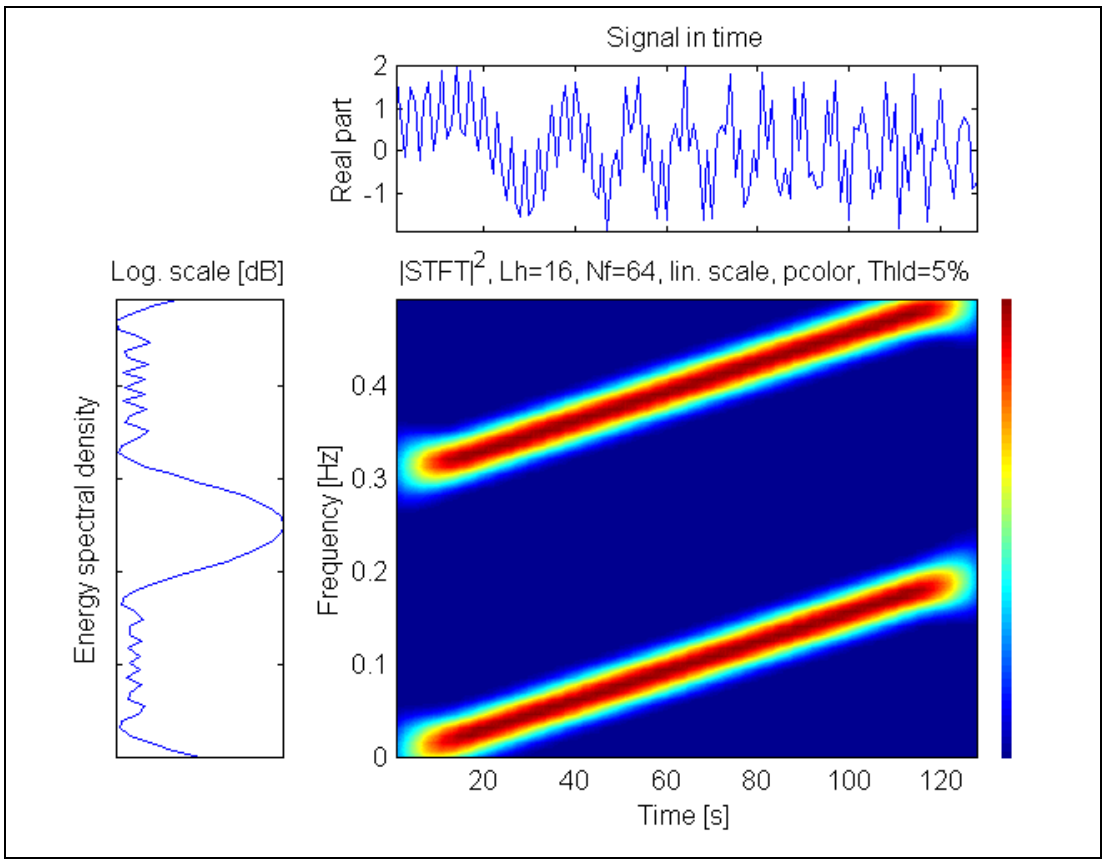

Figure 2-1: Different Representations of a Signal
TOP: time domain signal. LEFT: spectral plot. BOTTOM RIGHT: Energy versus time and frequency

We have stated that *sound analysis* and *sound modification* are two motivations for a TF approach. As well, a TF framework lends itself to synthesis approaches that do not rely on starting with a pre-existing transformed signal. In this synthesis case, we create samples in the TF domain and subsequently synthesize them to obtain time domain sample streams. These perspectives mean that a suitable TF approach can provide a *unified framework* for both *analysis-modification-resynthesis* and *granular synthesis* work. We elaborate on each of these applications in the following sections.

2.2.1 Analysis[†]

Several motivations exist for wanting to *analyze* a sound. First, we may desire to better understand the nature of sound in general, or the make-up of a particular sound. By transforming to the TF domain, we decompose the signal and see what it is comprised of relative to the basis functions of that transform.

Second, an *analysis* of a signal in the TF domain is a useful precursor to synthesis methods, such as in so-called *analysis-synthesis* approaches (Risset 1991). In that case, a particular model of the sound is assumed and the TF transform coefficients are analyzed to extract the necessary model parameters. The *Phase Vocoder* (PV) can be viewed as an example of this, modeling sound as a collection of slowly varying amplitude and frequency modulated sinusoids. Since this particular model has been studied and perfected over many years, the processes of analysis and synthesis are generally tied together in one automated application. In the study and application of newer models, the mapping of analysis features to a synthesis model may be more manual and experimental.

Third, for interactive performance with computers, accurate analysis is often necessary as an ear for the computer (Roads 1996).

Finally, in the case of the WPT, we optionally also perform a type of analysis when we perform a best basis search. We are effectively attempting to model the sound with a given subset of the possible basis vectors available. Our desire is to have an automated way of choosing a basis that allows for effective manipulation of the signal.

2.2.2 Sound Modification and Resynthesis

Having the sound in the TF domain creates many opportunities for modifications of the sound transform coefficients prior to and during re-synthesis. These modifications occur after analysis with the particular transform employed. The changes will generally be motivated by aesthetic considerations, but can also be due to practical considerations like changing the length of a sound segment to align it with some other event(s), or denoising a sound.

Some of the aesthetically motivated variants we might want to impart on a signal include time stretching and contraction, cross synthesis operations, frequency selective level adjustment (i.e., *equalization*), and

---

[†] We use the term *analysis* here to refer to the act of using the transform coefficients in some way, such as studying them, modeling them, or extracting parameters for a separate synthesis model from them.

transform coefficient filtering. These example modifications all rely on the somewhat separate control of time and frequency we get through projecting the signal on to the time-frequency plane. Some of the approaches have been used extensively in the computer music field (Roads 1996), though not in combination with the WPT.

2.2.3 Synthesis

A TF framework can be used even if no original signal exists. In this case, transform coefficients are created "from scratch", and only the structure associated with the resynthesis portion of the TF analysis-resynthesis scheme is used. For example, one could simply create transform coefficients in the TF domain associated with the wavelet transform (WT), and then submit these to the WT resynthesis process.

Since the TF view emphasizes the time-frequency duality of sound, it provides insight for the user into the nature of sound while composing. Having a view of and control over both time and frequency aspects of the sound promotes a different compositional paradigm (Clarke 1996).

2.3 Constraints

There are two constraints or realities that we should be aware of when considering a TF approach. The first concerns the nature of human hearing, which we will discuss under the heading of *psychoacoustics*. The second concerns the limits of time and frequency resolution that are possible with a TF transform, which is referred to as *time-frequency uncertainty.* We discuss each in the following.

2.3.1 Psychoacoustics

Several aspects of human hearing that we should be aware of in our subsequent work are now described. An understanding of the frequency *selectivity* of the ear is important in allocating time and frequency resolution in a way that we can best utilize it. Such understanding is especially important when we are attempting to find a fixed basis that best matches how the ear works. When attempting to find a *best basis*, on the other hand, knowledge of the *sensitivity* of human hearing as a function of frequency is beneficial, allowing us to weight the relative importance of the subbands that comprise the basis.

Human hearing exhibits a logarithmic characteristic with respect to frequency. In pitch perception this results in our music scales being octave based. For example, *A* above *middle C* is at 440 Hz, *A* one octave above that is at 880 Hz, and the *A* another octave up is at 1,760 Hz.

Critical bands also exhibit a logarithmic characteristic, increasing in width roughly linearly with frequency. A *critical band* for audio can be defined as a "Psychoacoustic measure in the spectral domain which corresponds to the frequency selectivity of the human ear" (Joint Technical Committee 1, 1993). For a given frequency, the *critical bandwidth* coincides with the smallest band of frequencies around it which activate the same part of the *basilar membrane* in the human auditory system. This quantity can be measured by taking a tone barely *masked* by a band of white noise around it, and then narrowing the noise bandwidth until the tone becomes audible. The bandwidth of the white noise at this point is the critical bandwidth of the frequency corresponding to the tone used. *Masking* is exploited in some audio compression codecs by determining the maximum amount of quantization noise that can be added without being heard at each critical band of human hearing, and then quantizing such that the noise is never greater than that amount.

Fig.*2-2* is taken from Roederer (1975). It shows the logarithmic relationship between pitch, or centre frequency, and critical bandwidth. The second upper curve shows critical bandwidth. Pitch relationships are also shown for intervals of a *Minor Third*†, *Whole Tone* and *Half Tone*.

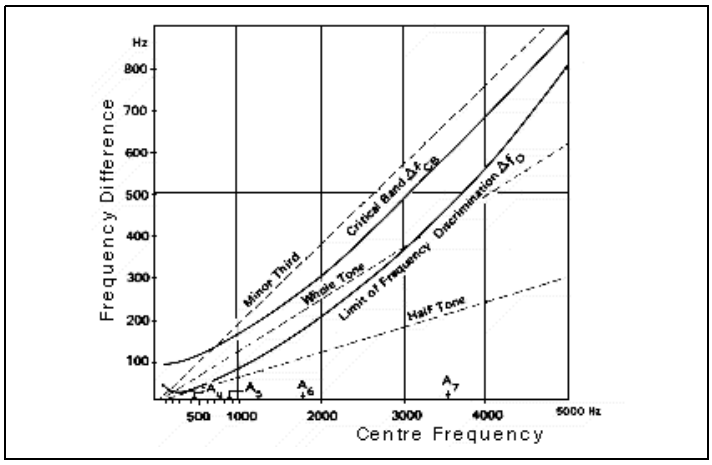

Figure 2-2: Critical Bandwidth and Music Intervals versus Frequency **(Roederer 1975)**

Zwicker and Fastl (1990) have approximated the continuous critical bandwidth curve as

$$BW_{critical} = 25 + 75(1 + 1.4(f/1000)^2)^{0.69} \quad \textbf{(1)}$$

It is also sometimes convenient to model the critical band response as a discrete set of bandpass filters. If

†. An example of a *minor third* interval in the distance from the note *A* to the note *C* within the same octave. In the same way that the octave grows with frequency, so do all other *intervals*, including the minor third example given here.

we start with a critical band centred at 100 Hz and continue up in frequency and band number we can create a full set of bandpass filters based on the known relationship between frequency and critical bandwidth. The critical bands in this approach are often referred to as *Barks*, and there are 25 of them stretching up to 16 kHz. The relationship is shown in Fig.*2-3*.

Later, we design a fixed basis that approximates the *one-third octave* spacing found in many equalizers (Roads 1996). The result is slightly coarser than the spacing of the critical bands, and consequently its *bandwidth-versus-frequency* curve would lie slightly above the *Critical Band* curve shown earlier in the Fig.*2-2*.

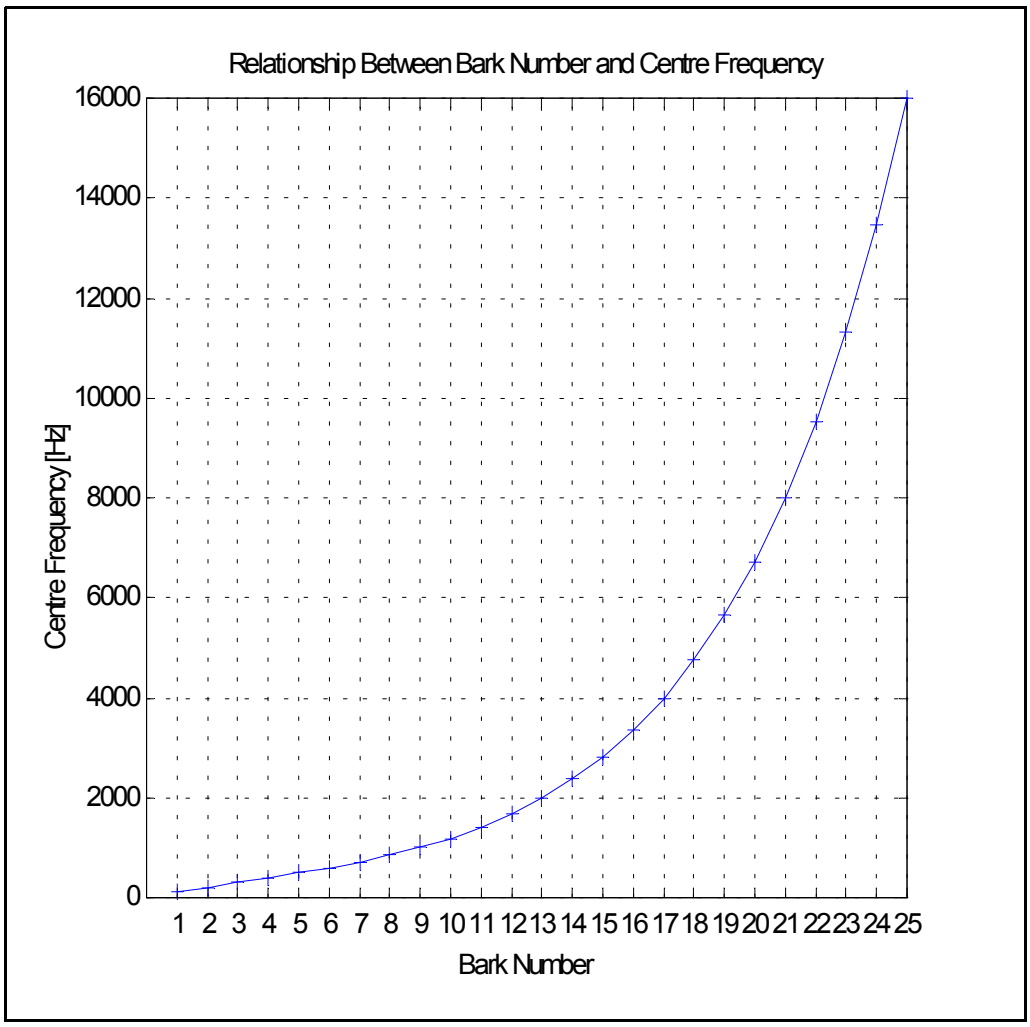

Figure 2-3: Example of Critical Bands (Barks) and their Centre Frequencies

We perceive sounds of equal magnitude at different frequencies with different loudness. The Fletcher-

Munson or *equal loudness* contours shown in Fig.*2-4* (Truax 1999) reflect this, showing the response of an average listener across different reference power levels. As can be seen from that figure, roll-off occurs at both ends of the spectrum. We can optionally include this psychoacoustic information in our calculations when doing best basis selection.

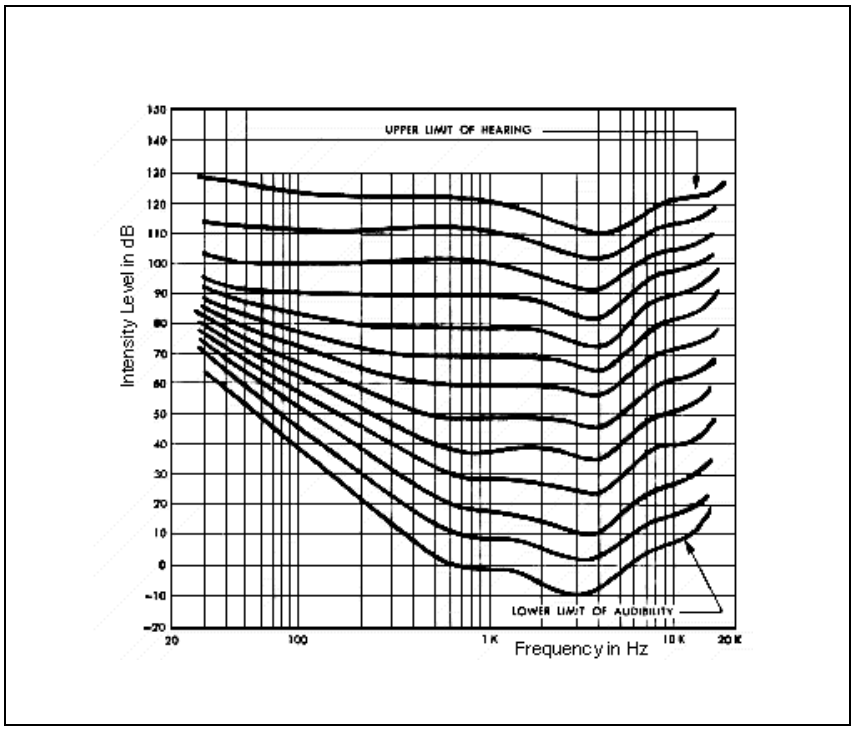

Figure 2-4: Equal Loudness Contours **(Truax 1999)**

2.3.2 Time-Frequency Uncertainty

As much as we might like to, it is not possible to model our sound signal with arbitrarily high time and frequency resolution. The name given to this fact is the *uncertainty principle*. We cannot know both the precise time onset and frequency content of a signal, no matter how good the measuring instrument. And, once we have reached a certain maximum precision, increasing the precision in one results in a decrease in precision in the other.

Gabor (1946) first formalized this concept. We repeat the analogous formulation from Rioul and Vetterli (1991) here:

$$\text{Time - Bandwidth Product} = \Delta t \Delta f \geq \frac{1}{4\pi} \quad \textbf{(2)}$$

where Δt is a measure of the time duration or time spread over which we look at the signal, and Δf is the bandwidth or frequency resolution of the measurement. If we use a "window function" $g(t)$ to extract the portion of the signal we want to look at, Δt and Δf can be formally defined as:

$$\Delta t = \sqrt{(\int t^2 |g(t)|^2 dt) / (\int |g(t)|^2 dt)} \quad \textbf{(3)}$$

and,

$$\Delta f = \sqrt{(\int f^2 |G(f)|^2 df) / (\int |G(f)|^2 df)} \quad \textbf{(4)}$$

Incidentally, Gabor (1946) pointed out that the *Gaussian* window function meets the bound in (2) with equality, which has made it a historically popular choice in signal analysis†. However, other considerations such as the main lobe width and sidelobe roll-off of the window's spectral response $G(f)$ will often favour different window function choices.

We saw in *Section 2.3.1* that the frequency selectivity of the human ear decreases roughly logarithmically with increasing frequency. By utilizing a fixed window size, *fixed resolution* transforms like the *short time Fourier transform* (STFT) also exhibit fixed frequency resolution, which means that the chosen window size will always result in a compromise. Either there will be not enough frequency resolution at low frequencies due to too small a window, not enough time resolution to accurately capture the attacks associated with high frequencies due to too long a window, or a combination of both problems due to a window size chosen for the middle frequency ranges.

With a *multiresolution transform* like the *wavelet transform* (WT), we get good frequency resolution at low frequencies, and good time resolution at high frequencies, which trades-off the resolution resource in a way that better matches how our ear can make use it. Recall, for example, the 25 *critical bands* depicted in Fig.*2-3*. The bandwidths of these critical bands increase with increasing frequency, analogous (but not identical) to the increasing bandwidths of the WT. The *wavelet packet transform* (WPT) improves on the

---

† The Gaussian function can be represented in the time domain as $g(t) = e^{-t^2/2}$.

WT, allowing for the allocation of time and frequency resolution in a highly flexible way. The distribution of time-frequency resolution corresponding to the WT is only one of a great many possible resolutions offered by the WPT.

In the context of our particular requirements of the transform, what are the ramifications of poor time or poor frequency resolution? If we are simply transforming the signal into the TF domain and then back into the time domain, as long as the transform offers *perfect reconstruction* (PR), there are no ramifications. However, we generally want to do something in the transform domain, such as modify coefficient values, or extract some parameters to feed a different synthesis model. In the latter case, use of too long a window will blur the onsets and endings of events such as attack transients, and any signal synthesized from a model based on parameters extracted from this transform data will necessarily include this blurring. Too short a window on the other hand may mean missing important spectral details, resulting in a blurring in the frequency dimension.

Another consideration in the allocation of time and frequency resolution is the potential introduction of aliasing distortion. For various reasons we may wish to eliminate certain subbands that are less important in the overall makeup of the signal. Alternatively, we may choose to modify certain transform coefficient values. However, these changes can result in the introduction of aliasing distortion with certain transforms, where the alias cancellation mechanism of the transform will be partially defeated. In this case we want a transform that minimizes aliasing across subbands in the first place so that alias cancellation is not needed as much. Such alias minimization suggests the use of long windows with sharp filter responses, and correspondingly poorer time resolution[†].

Any best basis selection could benefit from such longer filters, so that choices are made based on the actual energy in a band, not that due to the aliasing of energy picked up by spurious sidelobes. However, again the poorer time precision will cost us. For example, we may want to *threshold* low amplitude coefficients for noise reduction purposes, or perhaps equalize the signal energy across subbands in some way. In these and other cases, our changes will be based on somewhat blurred time-domain information. If we want to do some granular synthesis within the same transform configuration, we will also be limited to these long basis functions, and may not be able to generate shorter impulse-like events.

[†] For example, a perfect 'brick-wall' filter response appears in the time domain as the *sync* function, which is of theoretically infinite time duration.

In all these cases, having a flexible framework within which to trade-off time and frequency resolution is helpful, and perhaps the best we can do. When we study the WPT in more detail, we'll see that two factors affect the transform resolution, the length of the *prototype filter* impulse responses, and the depth of the transform at a given frequency. For example, if our prototype lowpass and highpass filters use eight coefficients, then all subbands at a depth of three will have basis functions with impulse responses of length $8 \times 2^3 = 64$. At a depth of two their length will be half that, or 32 samples. The effective filter length doubles with each new level due to downsampling by two. By allowing subbands of different depths for a given signal, we can often allocate the time-frequency resolution of the transform in a way that corresponds to the energy distribution of the signal, either manually, through the use of a best basis method, or a combination of both. Having such correspondence between the signal energy distribution and the transform structure can make our subsequent sound modifications more effective.

2.4 General Form

The *time-frequency* (TF) methods we discuss share a common structure. Whether we transform a pre-existing signal into the TF plane as in the case of analysis based methods, or whether we create new coefficients afresh as in the case of granular synthesis, the ability to work with time and frequency localized cells (or grains) is what unites all these approaches. We now describe the general form of the transform, for both analysis and synthesis.

With analysis based methods, at periodic time intervals we project the signal onto each of a set of *basis functions* to determine their respective correlations. The result of each correlation corresponds to a *transform coefficient*, which is a measure of the similarity between the signal and each of the basis functions. This inner product operation can be expressed as:

$$c(f, \tau) = \int s(t)\hat{b}(f, \tau, t)dt \qquad \textbf{(5)}$$

where $c(f, \tau)$ denotes the transform coefficient, $s(t)$ the signal being analyzed, and $\hat{b}(f, \tau, t)$ the basis function. In the general case, $\hat{b}(f, \tau, t)$ can be complex valued, in which case $c(f, \tau)$ is too. The variable τ is the point in time where the basis function is centered, and therefore specifies the time around which the signal is being analyzed. The variable f specifies the frequency of the basis function. Integrating over time results in a continuous function in τ and f, $c(f, \tau)$.

Resynthesis proceeds by inverting the transform. In the case of many transforms, such the STFT, the WT, and the WPT, the resynthesis basis function is simply the complex conjugate of the analysis function. In the continuous time case, the only requirements for this complex conjugate relationship to hold are that, for the WT and WPT, the basis function b be of finite energy and *band pass,* and for the STFT, that the window function portion of the basis function (see below) be of finite energy. Orthogonality is not required (Rioul and Vetterli 1991). The reconstructed signal is defined as

$$s(t) = K\int\int c(f, t)b(f, \tau, t)dfdt \quad \textbf{(6)}$$

where K is a constant that depends on the basis function b (in some cases, K is simply equal to 1).

We now discretize the transforms, replacing the continuous offset τ with the discrete offset n, and the continuous frequency variable f with the discrete frequency variable k. This results in us performing the analysis transform at discrete time offsets and for discrete frequencies, so that a grid or matrix of coefficients in the time-frequency plane is produced. Furthermore, for any practical implementation on a computer we require a discrete time variable, so we also discretize time t, replacing it with the discrete time variable m.

The analysis transform can then be represented as

$$c[k, n] = \sum_m s[m]\hat{b}[k, n, m] \quad \textbf{(7)}$$

In the *fixed resolution* case where the bandwidths of the basis functions are the same for all frequencies, the time-frequency resolution attained is constant over all time and frequency. In the *multiresolution* case where bandwidth is a function of the basis function frequency, a less uniform pattern results. Wider subbands have better time resolution, and vice versa. These two scenarios are shown in Fig.*2-5*. We note also that in most practical transforms, the coefficient sample rate of a given subband is proportional to the relative bandwidth of the subband. Wider subbands must use a higher sample rate to accurately capture the information, while narrower subbands can use a proportionally lower sample rate. When the sample rate of a given subband is reduced in direct proportion to its relative bandwidth, the transform is referred to as being *critically sampled.* For such a transform, the cells in Fig.*2-5* would also depict the relative incidence of transform coefficients for the two types of transforms.

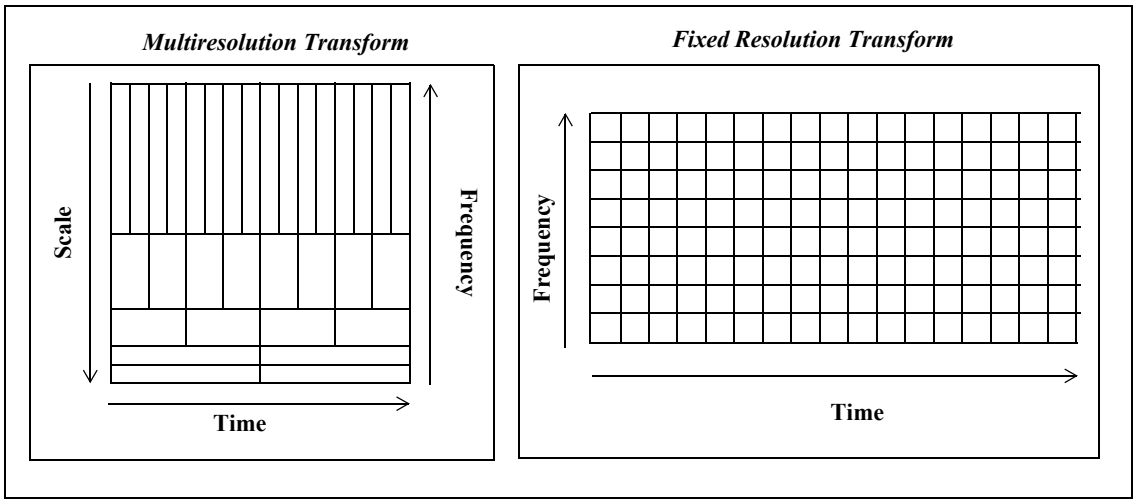

Figure 2-5: Time-Frequency Resolution of Fixed and Multiresolution Trans-

An example of a set of discretized basis functions is given in Fig.*2-6*, which depicts the *Haar* basis functions required for a *depth-2 wavelet transform* (WT). The functions labelled $b_{k,n}$ in the figure correspond to the basis function $\hat{b}[k, n, m]$ in (7). Since the functions are real, the conjugation symbol on the basis function is dropped. The sample number m is given along the horizontal axis. Since the basis functions are discrete in time, scale, and time offset, they are used in the *discrete wavelet transform* (DWT). These *depth-2* WT *Haar* basis functions correspond in time-frequency resolution and transform coefficient locations† to the bottom three rows of cells in the multiresolution grid of Fig.*2-5*. Specifically, $b_{0,n}$ corresponds to the bottom row of cells, $b_{1,n}$ to the row above that, $b_{2,n}$ to the first and third cells in the third from bottom row, and $b_{2,n+1}$ to the second and fourth cells of that same row. While the *Haar* basis is not often used in practice due to the poor isolation it provides between subbands, it does meet all the requirements of a basis function for the WT and WPT. Because it does represent a valid basis function, and given its simplicity, we will use it as an example in several places throughout this document. Later we will show an example transform with it, show its relationship to the filters used in a fast filter bank implementation, and also show how the filter and basis function (or *packet*) are related via the so-called *dilation* and *wavelet equations*.

---

† In the *critically sampled* case.

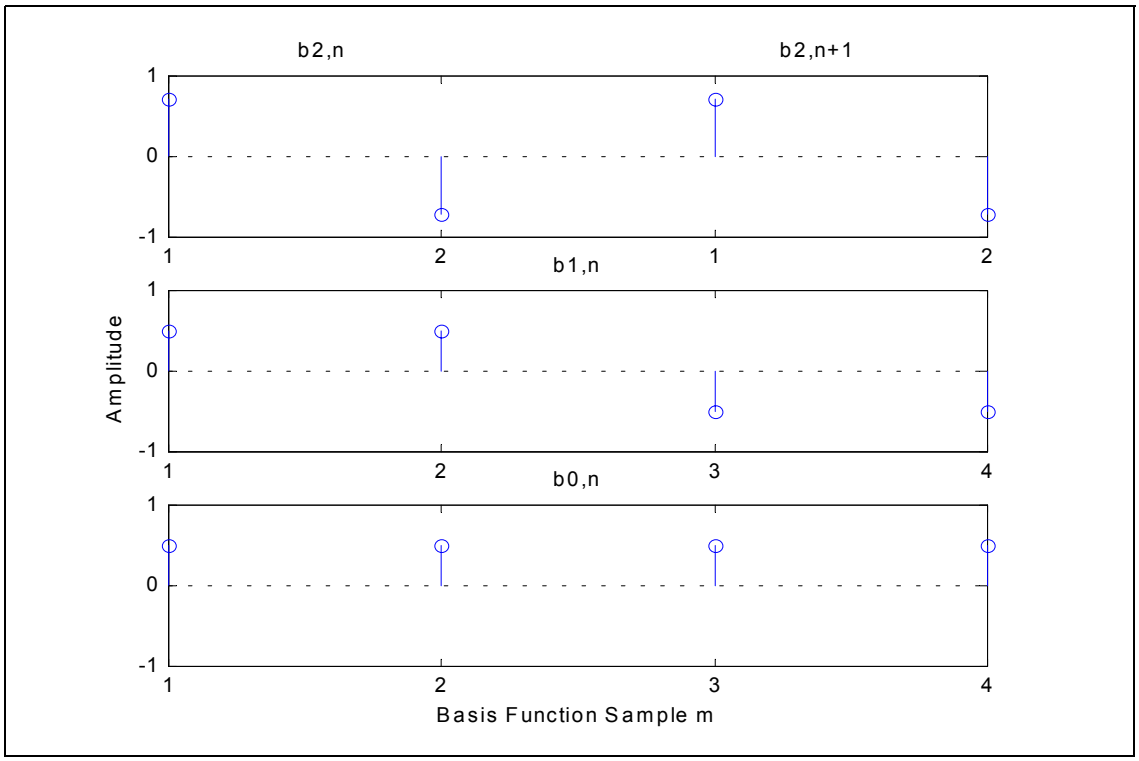

Figure 2-6: Example Basis Functions: Depth-2 Discretized Haar

Resynthesis in the discretized case generally involves more restrictions on the basis functions than in the continuous time case, and the less redundant the set of basis functions, the greater the restrictions. In the case of the WT and WPT, we will rely on either orthogonal or bi-orthogonal bases to enable perfect reconstruction. In the latter case, the resynthesis basis function is not simply the complex conjugate of the analysis function[†].

Given that the resynthesis basis function is the complex conjugate of the analysis function, such as with orthogonal wavelets or the STFT, the reconstructed signal is defined as

$$s[m] = K\sum_{k}\sum_{n} c[k,n]b[k,n,m] \qquad \textbf{(8)}$$

In the case of synthesis-only schemes such as the *granular* approaches, we can think of (8) as specifying

[†] We have primarily used orthogonal wavelet filters, since some of the cost functions used in best basis selection require such orthogonality.

the synthesis operation. In that case, the basis functions are the *grains* used, and the $c[k, n]$ are the time and frequency dependent values that trigger grain generation.

Sometimes the basis function is separable into a windowing part, and an oscillatory part. Then we can express the basis function as

$$b[k, n, m] = W[k, n, m]o[k, m] \quad \textbf{(9)}$$

where $W[k, n, m]$ is a window function, and $o[k, m]$ is an oscillatory function.

For the STFT, the window function will be one from a handful of possibilities, such as the *Hanning* window, the *Blackmann* window, or the *Gaussian* window of the *Gabor Transform*. The window function is real-valued, and is not dependent on the frequency variable k, resulting in the *fixed resolution* nature of the transform. The oscillatory function is the complex exponential of the *Discrete Fourier Transform* (DFT).

In the WT and WPT cases that we consider, the basis functions are all real. As such, they are identical for analysis and resynthesis transforms when we choose to use orthogonal functions (which we usually do). While the windowing and oscillatory aspects of the wavelet basis functions are not generally neatly separable, we can think of them that way. From that perspective, the window function length is dependent on frequency, since higher frequency wavelet basis functions are shorter than lower frequency ones. This scaling in length is what makes the transforms *multiresolution*.

2.5 Analysis based TF Methods

Precursors to present day *analysis based* TF methods include the *pitch synchronous* analysis effort led by Max Mathews and Jean-Claude Risset in the 1960's, and the *heterodyne filtering* attempts of Freedman, Beauchamp and Moorer (Roads 1996).

Pitch synchronous analysis was used to create time-varying amplitudes for the fundamental and harmonics of brass instruments. The basic idea is to extract segments of the signal whose lengths are based on the period of a partial in the signal, and then perform Fourier analysis on that segment as if it were periodic[†]. This information was then used for the so-called *analysis-synthesis* techniques employed by Mathews and Risset (Risset 1969), and later by Moorer and Grey. In 1977 the latter team published a *"lexicon of analyzed tones"* which provides data for instrument resynthesis (Moorer and Grey 1977). This

[†] This idea of windowing a signal segment and replicating it is used in the *STFT* too.

pitch synchronous technique was very time consuming, and because it relied on the identification of periodic segments, lacked the generality required for the analysis of all sounds.

Heterodyning analysis employed a bank of filters at the known frequencies of a signal fundamental and harmonics. The idea was to multiply the signal with sine waves at the fundamental and harmonic frequencies being analyzed, and integrate the result of each multiplication with a lowpass filter. This gave a time varying measure of the similarity between the signal and each partial frequency. While Beauchamp later added a tracking capability to allow for the following of pitch changes and sharper attacks, the technique was superseded by other approaches.

The other historically significant analysis based *TF* method is the *sonogram*, which we mentioned earlier. It has traditionally been used for speech analysis, where speech energy is plotted on a grey scale against time on the horizontal axis and frequency on the vertical axis. The first sonogram dates back to 1932. Early versions consisted of a bank of bandpass filters whose outputs drove a recording device, such as ink on a roll of paper. Today sonograms usually use the *STFT* to generate their input (Roads 1996).

2.5.1 STFT

The *Short Time Fourier Transform* (STFT) involves the windowing of a signal over time combined with the application of a *Discrete Fourier Transform* (DFT) to the samples produced in each of the windows. This modification of the DFT adds a time component to the result, thereby adapting it to the study of time varying signals (Allen and Rabiner 1977). Given this ability to provide a time-frequency view of a signal, combined with the fact that the highly efficient *Fast Fourier Transform* (FFT) can be used to implement the DFT, the STFT has enjoyed great popularity in the fields of audio processing and computer music over the past 30 years. We now describe the STFT in some detail. The description is more in-depth than that of other approaches because use of the STFT is so prevalent, because it is the foundation for many other techniques, and because other time-frequency approaches are often compared to it.

Gabor (1946) first proposed a time-frequency model for sound. The *Gabor Transform* can be viewed as an STFT with a *Gaussian* function for the window. Fig.*2-7* shows a series of Gaussian windowed oscillations at increasing frequencies from left to right†. The *STFT* generalizes this by allowing a variety of windows.

---

† The waves look similar to wavelets, except the number of cycles here varies with frequency and the time duration of the wave envelope is kept constant. We will see the opposite is true of wavelets.

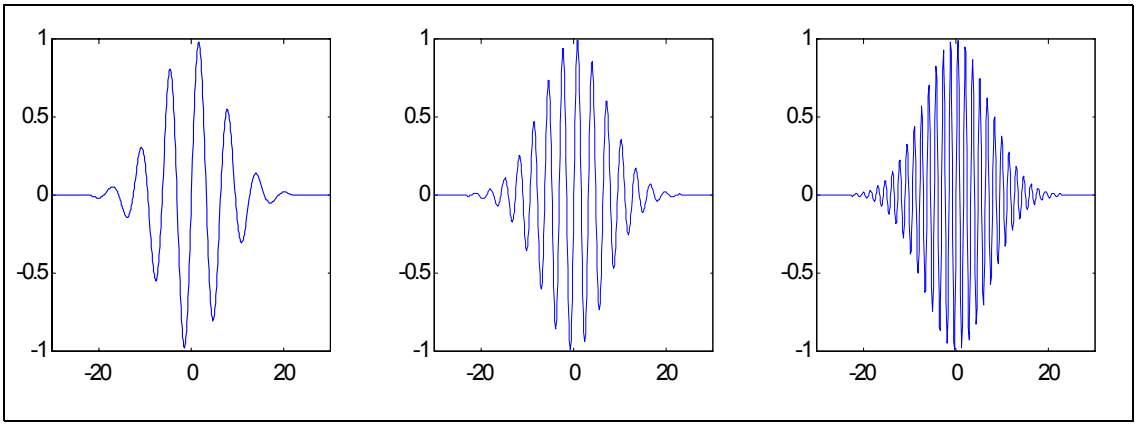

Figure 2-7: The Gaussian windowed Sinusoids used by Gabor
(frequency doubles moving from left to right)

The *STFT* determines the correlation of a signal over time with a set of harmonically related frequencies. As we saw in *Section 2.3.2*, there are limits to the time and frequency resolution possible with any transform. By windowing the signal over time, the time resolution over the basic DFT is improved. The price paid for this increased time resolution, however, is a decrease in frequency resolution. By multiplying the time domain signal by the window, the resulting spectrum will be the convolution of the window spectrum with that of the signal spectrum. If, for example, our signal were a pure tone and the window a rectangular function (*rect)*, the spectrum of the windowed tone would appear as a *sinc* function centered at the tone frequency†.

The *STFT* of a signal $s[m]$ can be described as

$$c[k, n] = \sum_{m} s[m]W[n-m]e^{\frac{-j2\pi}{M}km} \qquad \textbf{(10)}$$

where $c[n, k]$ denoted the transform coefficients, $W[n-m]$ is the window function, n is the time offset where the window is centered, m is the discrete time variable, and M is the size of the DFT. While (10) provides a general representation of the STFT, we should note that it is usually implemented with the FFT. In that case, after windowing the signal, the projection onto the complex exponentials is reorganized to exploit similarities at different frequencies. Something analogous is done with the WT, where the highly efficient *fast wavelet transform* (FWT) can be used.

---

† Since the *sinc* function is the *Fourier Transform* of the *rect* function.

There are many parameter choices required in specifying the STFT, such as the window type and size. Any window chosen should have time-frequency resolution properties that come close to the *time-frequency uncertainty* limit given in (2). Then the choice between different windows involves trading off the main lobe width of the window's spectrum, with its sidelobe behaviour. For example, the *rect* function's spectrum offers a narrow main lobe, but poor side lobe roll-off. More popular windows generally use a wider main lobe with much more aggressive sidelobe roll-off. Window lengths are typically in the 0.001 to 1 second range (Roads 1996). The particular value chosen will depend on sample rate, and the relative importance of time versus frequency resolution to the application at hand.

Another important choice is the spacing between successive windows. This will determine the time difference between successive values of n in (10). Typically an overlap of at least 50% is used, and even more when time stretching of the signal with *overlap-add* (OA) resynthesis is employed (see below). In order to achieve *perfect reconstruction* with OA resynthesis, the overlap value and window shape must be such that the overlapped windows sum to a constant over all time values m.

We can see from (10) that the STFT result is complex due to the complex exponential of the DFT. Generally the result is used in a *magnitude-phase* representation. The squared magnitude can be plotted as a spectrogram, and the (unwrapped) phase spectrum is also often of interest. We saw an example of such a spectrogram based on the STFT in Fig.*2-1*.

Two approaches to resynthesis are used, *Overlap-Add* (OA) and *Oscillator Bank* (OB) resynthesis. In the case of OA resynthesis, we simply apply the inverse of the forward STFT process. First the inverse DFT is applied to the spectral components at each time, and then the result is windowed and overlap-added with the results from other time offsets. In theory, this results in perfect reconstruction of the signal, though in practice some loss results in most applications (Roads 1996). For example, if the overlapped window functions do not sum exactly to a constant, some distortion will result.

In the case of *Oscillator Bank* (OB) resynthesis, a bank of oscillators centered at each of the DFT frequencies is used. Amplitude and frequency deviation envelopes are constructed from the frame-based DFT analysis data and used to drive the oscillators. The amplitude envelope information is derived from the magnitudes of the complex DFT coefficients, and the frequency deviation envelope is derived from the changing phase values of the coefficients. Converting phase to frequency involves *unwrapping* the phase prior to calculating the change in phase over time.

By employing these amplitude and frequency envelopes, the OB scheme lends itself well to certain modifications, especially time stretching and contraction when the source signal is highly tonal. If the sound is well modeled by one or more of the oscillators in the resynthesis bank, it can be stretched or contracted in time without changing its pitch. Such time changing with preserved pitch is not possible to the same extent with OA resynthesis, since the separation between time and frequency within the model is not as complete.

Instead, OA resynthesis utilizes a high degree of window overlap when time stretching is anticipated, allowing the user to trade-off the inefficiencies that come with a high degree of overlap and the benefits attained in time-stretching. During stretching, the overlapped windows are pulled apart prior to resynthesis, and the phases adjusted accordingly[†].

A disadvantage of the OB scheme versus OA is that it is less computationally efficient. Also, the OB approach is more susceptible to the limitations of the time-frequency resolution of the analysis transform since resynthesis is not achieved by simple inversion of the analysis process, but rather via model-based parameter extraction.

The primary drawback of the STFT as a whole is its fixed resolution nature.

2.5.2 Phase Vocoder

The *Phase Vocoder* (PV) was initially developed for the purpose of efficiently representing digitized voice for communications. It was first described by Flanagan and Golden (1966), and has seen a good deal of use in computer music over the past two decades. In fact, the PV is available in many general purpose sound analysis/synthesis programs.

The PV tends to be built around the STFT, though it can also be implemented with filter banks. In the case of the STFT, it can take advantage of the highly efficient FFT algorithm, and uses the OB resynthesis approach of the STFT described in the previous section.

Fig.*2-8* shows a block diagram of a single frequency band of the PV analysis stage (Dolson 1986). While the figure indicates a filter bank implementation, the STFT implementation would simply involve replacing everything inside the dashed box of the figure with an STFT.

[†] Note that the set of windowed complex sinusoids of the overlapped STFT is not a basis, but rather an *over-complete* set of functions that is not linearly independent. The advantage is that the time stretching operation can utilize this over-complete information, conveniently revealing the details of the overlapped regions during stretching.

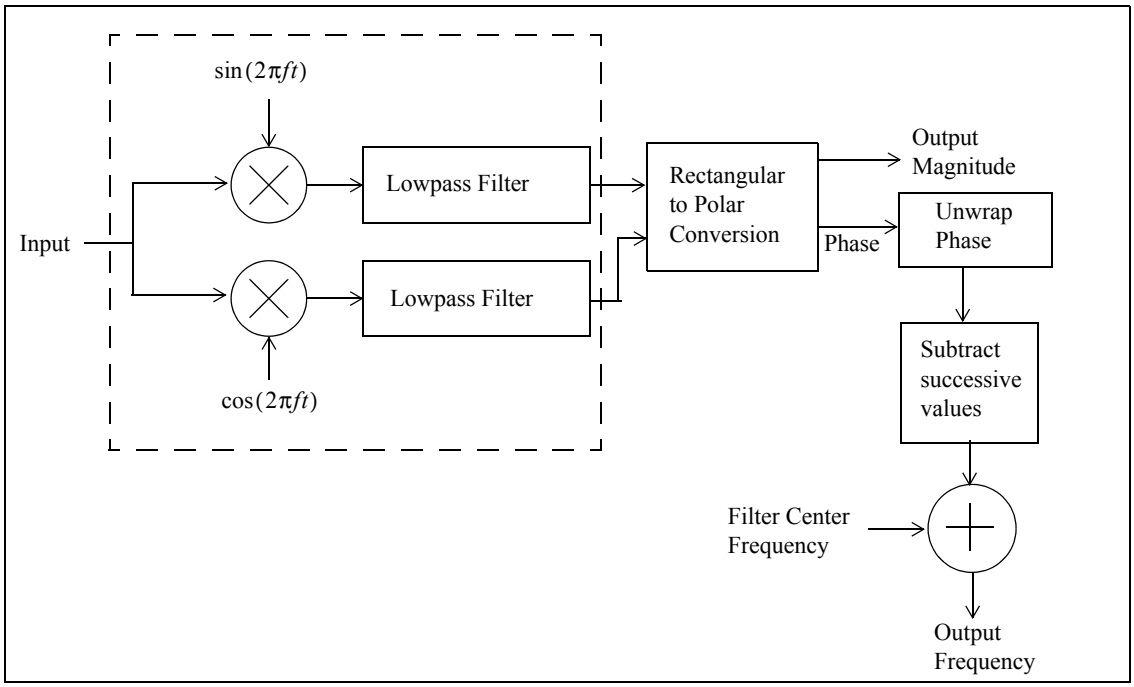

Figure 2-8: A Single Band of the Phase Vocoder

A particularly interesting version of the PV, called the *Tracking Phase Vocoder* (TPV), uses a sophisticated peak identification and tracking algorithm to allow for the tracking of changing tonal content over time. However, by constructing tracks, it also discards all data that does not match these tracks, eliminating much of the noise and transient data associated with the original sound (Roads 1996). This discarding of data has in turn motivated the addition of a noise-like component to the resynthesis process, such as is done in the *spectral modeling synthesis* approach of Xavier Serra (1997).

2.5.3 Wavelet Transform

The *wavelet transform* (WT) is inherently *multiresolution*. The particular partitioning of the WT is such that frequency resolution decreases with increasing frequency in the same proportion that time resolution increases. Such a multiresolution offers the user a time-frequency partitioning that corresponds better with the human auditory system then do fixed resolution transforms.

Recall from *Section 2.3.1* that we hear pitches with greater precision as they decrease in frequency. Also, we need to preserve the precise timing of attack transients, which are generally comprised of high

frequencies, if we are to successfully characterize and reproduce sounds without noticeable time-based distortions. We hear in a *multiresolution* way, and so it makes some sense to do our transform in this way. Whereas the STFT uses a constant window size for all frequencies, which in turn produce a constant bandwidth for all frequencies, the WT utilizes window lengths that are inversely proportional to frequency. The notion of *frequency* is in fact replaced with that of *scale*, where *scale* is directly related to window length, and inversely related to frequency. As a result of the changing window length (or scale), low frequencies are localized well in frequency but not so well in time, and higher frequencies are localized well in time but not so well in frequency. The number of cycles in all the wavelets is the same, so they get progressively shorter in duration as frequency increases. Recall Fig.*2-5*, which showed the time-frequency resolutions of a fixed and multiresolution transform, such as the STFT and WT respectively. In the case of the multiresolution WT, we can the frequency bands get narrower and longer in time duration as we move from top to bottom. The *scale* of the wavelets is increasing in this case. For the fixed resolution STFT, bandwidths and time durations are fixed over all time and frequency.

Many wavelets can serve as the *basis function* in the WT. While the window and sinusoid that comprise the basis function in the STFT are separate, the wavelet wraps up both aspects in one. The discretized version of the WT can be expressed mathematically as the following inner product operation

$$c[a, n] = (1/(\sqrt{a})) \sum_{m=-\infty}^{\infty} s[m]\hat{w}[\frac{m-n}{a}] \qquad \textbf{(11)}$$

where a is the *scale* parameter related to wavelet duration, n is a time shift parameter, m is the discrete time variable, and $\hat{w}$ is the complex conjugate of the *mother wavelet* w. In the general case, w is complex valued so that magnitude and phase are included in the transform.

Since (11) represents the discretized version of the transform, a finite set of discrete wavelet frequencies of specified exponentially based separation are used (a), along with a discrete set of time offsets (n). Furthermore, the transform is discretized in time, so that the signal s is comprised of a series of samples m, and the wavelet basis functions are also represented by a discrete set of values†. This transform is referred to

† Note that the continuous time transform that uses discrete scales or frequencies and discrete time offsets is referred to as the *discrete time wavelet transform* (DTWT) or *wavelet series* (WS). This is analogous to the *Fourier Series* (FS). When scale and time offset are also continuous, the transform is called the *continuous wavelet transform* (CWT). The CWT is analogous to the *Fourier transform* (FT). We will not use the CWT or DTWT in our work.

as the *Discrete Wavelet Transform* (DWT). When the DWT is implemented with a filter bank, very efficient algorithms are possible, such as the critically sampled *fast wavelet transform* (FWT) of S. Mallat (Strang and Nguyen 1997). In our work we focus on the DWT (and DWPT), and use an efficient filter bank implementation. We describe details of our approach later in *Section 3.2*.

We can compare the expression for the DWT in (11) with the general form of the discretized transform given earlier in (7). The discrete time variable *m*, and the time offset variable *n* are the same in both expressions. However, the scale variable *a* in (11) has replaced the frequency variable *k* in (7). One can see from (11) that as scale increases, the wavelet gets longer, and that to compensate for this, the amplitude drops by the $1/(\sqrt{a})$ factor.

Early application of the WT to computer music was centred in Marseilles France where the transform itself was developed for physics and acoustics. Richard Kronland-Martinet and his colleagues released a series of papers in the late 1980's and early 1990's (Kronland-Martinet 1988, Boyer and Kronland-Martinet 1989, Kronland-Martinet and Grossman 1991, Kronland-Martinet, Guillemain and Ystad 1997). Also Gianpaolo Evangelista (1991, 1997) released a series of papers in the early to mid-90's that use the wavelet transform in computer music applications.

In the case of Kronland-Martinet and his colleagues, much of their work centred on use of either the *continuous wavelet transform* (CWT), or a highly redundant DWT, in both cases using a complex valued wavelet. In the DWT case, they used a highly redundant set of wavelet functions and very small steps in the time offset to produce smooth time-frequency domain representations[†]. The approach is both computationally intensive and creates an enormous amount of data, especially when compared to the *dyadic* DWT used in the *fast wavelet transform* (FWT)[††]. The advantage for them was that by employing a continuous or pseudo-continuous approach, along with a smooth complex valued wavelet (they used a Gaussian windowed complex sinusoid), they were able to produce very smooth data and plots that lent themselves to further analysis and modeling. In one application, for example, they use their transform data for parameter extraction in the modeling of sounds.

This group also looked at using the DWT with larger scale and time offset steps, including the *dyadic*

[†] This would correspond to a small difference in the parameter *a* in (11) between two consecutive wavelets.

[††] *Dyadic* spacing denotes a power of two relationship between scales or levels within the WT. For example, a wavelet at depth *j* in the transform is twice as long as a wavelet at depth *j-1*, and covers one half the relative bandwidth.

spacing case (Boyer and Kronland-Martinet 1989)†. Again they used a complex wavelet. They reported having achieved interesting results, including performing various sound modifications such as time stretching and cross synthesis. Their success with large scale step sizes including the *dyadic* case suggested that the dyadically spaced DWT and DWPT used in our research might also be successfully applied. An advantage to their use of the DWT rather than the filter bank based FWT, which only supports dyadic spacing, is that the frequencies at which their wavelets were centred was easily adjusted (by the parameter a), and not tied to the sample frequency. They appear to have exploited this in experiments with lining up the wavelet frequencies with the source material, such as in choosing frequencies that match the fundamental and partials associated with a musical chord.

Also of note in the paper of Boyer and Kronland-Martinet (1989) is their recognition of the relationship between the WT and *granular synthesis*. In particular, they present a formula for resynthesis of the WT and state that the formula is related to the granular synthesis methods implemented by Roads (1985) and Truax (1988), underscoring the similarity between TF resynthesis techniques such as the WT resynthesis described in that paper and the granular synthesis approaches. They also suggest that the resynthesis engine of a given TF approach, with an appropriate *front-end* to feed it input data that would normally have come from the TF transform of a sound source, could be used in a synthesis mode.

Evangelista (1991) has experimented with the DWT, including dyadic spaced filter bank versions, and reports on how it might be used. He also describes a wavelet based granular synthesis scheme with user controlled impulse trains driving dyadically spaced wavelet basis functions. Later, Evangelista (1997) shows the application of the WT to amplitude envelope estimation and describes a recent extension of the WT that he calls the *pitch-synchronous wavelet transform*. Like so many other TF methods in computer music, this is largely motivated by a requirement to effectively model the tonal characteristics of many musical instruments; a requirement that is not of primary importance to us since we are focused on environmental sounds.

---

† Their research would have taken place at around the same time as the discovery of the FWT by Mallat, though it likely took some time to become broadly known. Also, the equivalence of filter banks and wavelets was apparently recognized later.

2.6 Granular based TF Methods

Granular methods are all based on building up complex sounds with small *grains.* In the terminology of the transforms in the last section, these grains are like the *basis functions* we used there. However, while the term *basis function* suggests certain mathematically rigorous properties, a *grain* is more loosely defined†. Since almost by definition granular synthesis techniques are not involved in analysis, the looser definition can be seen as providing more freedom. Also, the term *grain* emphasizes its limited time and frequency scope. *Grains* combine time and frequency domain information.

Granular based methods all share a common synthesis structure, differing primarily in the characteristics of the grains used, their rate, variation, and the number and synchronization of grain streams used. The grain itself is generally comprised of some waveform that is windowed. If the grain is derived from a pre-existing sound, we refer to the technique as *granulation*.

Iannis Xenakis was the first to compose with grains of sound. In the late 1950's and early 1960's he started creating granular sounds with analog tone generators and tape splicing. Curtis Roads created the first computer-based *granular synthesis* implementation in the mid-1970's. Since then several approaches have appeared, including the *Quasi-synchronous Granular Synthesis (QSGS)* techniques of *GSX* (Truax 1988), *VOSIM (*Kaegi and Tempelaars 1978), and *FOF* (Rodet 1980), and *Asynchronous Granular Synthesis* (Roads 1991, 2002).

Granulation goes back at least to the mid 1940's, when Dennis Gabor built an electromechanical time/pitch changer. Since then there have been several granulation based devices, including the magnetic tape based *Tempophon*, and various subsequent digital based implementations. Two of the more well known current granulation methods are *GSAMX* (Truax 1994) and *FOG* (Clarke 1996). Both include a large number of parameters and options that make them highly versatile granulation platforms.

Before proceeding to descriptions of particular granular methods, we describe the general relationship between the time domain aspects of the grain streams produced, and the frequency domain properties that should correspond. In our description of this relationship, we look at the spectrum and repetition rate of the

---

† A collection of basis functions forms a *basis*, which necessarily must span the space for which it serves as a basis. It should also be linearly independent, though we have used the term basis function loosely to also include the functions used in the continuous versions of the WT and STFT, which are certainly not linearly independent. More accurately, those functions form a *frame*. Though we loosely refer to the grains in granular synthesis as basis functions, they needn't necessarily meet either the spanning or independence requirements.

grain window and the spectrum of the grain waveform separately, and then predict the result of their combination.

Imagine that the grain waveform is a simple sinusoid at some frequency f_o, and that this waveform is windowed in time to limit its duration. The resulting grain is then repeated at some lower frequency f_m to produce the grain stream. Windowing involves multiplication in the time domain between the waveform and the window, and this is equivalent to convolving their spectra in the frequency domain. The spectrum of the sinusoid is a single line located at frequency f_o. The spectrum of the window will depend on its shape and duration. The longer and smoother the window, the narrower its spectrum, and vice-versa. Imagine that we window with the familiar rectangular (*rect*) function, a function that is decidedly not smooth. Since the *rect* function's spectrum is the *sinc* function, the resulting overall spectrum at the time the window is active will be a *sinc* function centred at the sinusoid frequency. Essentially the window has caused a smearing of the sinusoid in frequency. In the case where formant regions need to be synthesized, this smearing can be used as a way to create them.

The rate at which the grains are produced also effects the overall spectrum. Depending on this rate and the resulting duty cycle, different audible results occur. If the grains occur frequently enough and with sufficient duty cycle, the process can be viewed as a type of amplitude modulation (as in *quasi-synchronous granular synthesis*, or QSGS), and the frequency of this modulation causes audible side bands. In the case of a sparser pulse train, the frequency of the pulse train itself may emerge as a fundamental, along with its harmonics. These pulse train related frequencies are also used in the case of formant schemes which for example simulate the voice by creating both a fundamental frequency and a number of formant regions.

2.6.1 Granular Synthesis

2.6.1.1 GSX

GSX (Truax 1988) is an example of a granular synthesis scheme. One of its key features, like its granulation counterpart *GSAMX,* is that it operates in real-time.

In GSX, grain waveforms are derived either from a simple oscillator or an FM (frequency modulation) based waveform. Each grain is windowed with a three part linear envelope whose relative attack and decay portions can be configured. Multiple streams are used in parallel and are each configurable. By using different grain waveform frequencies and window durations, each of these streams can be mapped to

different frequency regions. In particular, the relative rise and fall times of the window and its overall duration determine the formant characteristics. In listing these parameters of a granular synthesis system, we note how well the basic structure of the WPT maps to the granular case. The wavelet packet basis functions can be viewed as the windowed waveforms that comprise the grains in granular synthesis, and there are a number of parallel streams of these waveforms, one corresponding to each subband of the transform.

A block diagram representation of the GSX instrument configuration is given in Fig.*2-9*. The basic structure is typical of many granular synthesis implementations.

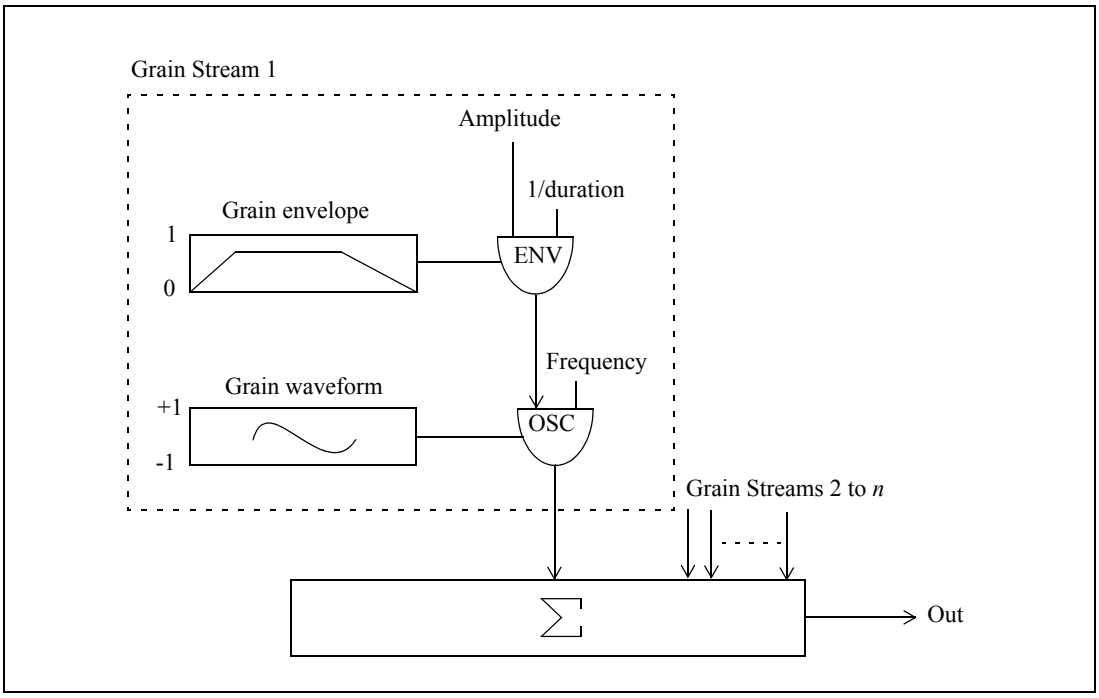

Figure 2-9: Granular Synthesis Instrument Configuration
(adapted from Roads (1996))

GSX can be categorized as a *quasi-synchronous granular synthesis* (QSGS) approach. It is *synchronous* in the sense that a regular flow of grains occurs within each grain stream, with a constant average grain rate. It is *quasi* (*-synchronous*) in the sense that while the average grain rates are fixed, the actual delay between grains is randomized over some configurable range. This randomizing of inter-grain onsets, along with varying the nature of the grain itself over time, leads to a thickening of the sound texture through a 'blurring' of the formant structure (Truax 1987, 1988).

2.6.1.2 FOF and VOSIM

The synthesis methods FOF and VOSIM can also be categorized as QSGS schemes (Roads 1996). They are grouped together because they have a lot in common. In particular, both were originally developed to model vowel sounds, which requires the creation and control of formant regions in the synthesized spectrum.

FOF is an abbreviation of the French *fonction d'onde formantique*, or formant wave-function synthesis (Rodet 1980). It looks structurally very much like GSX, with a bank of parallel grain stream generators. The FOF grain is a smoothly windowed sinusoid, and the frequency of this sinusoid for a given stream determines its formant peak frequency. Window attack and decay related parameters are configurable and affect the width and shape of the formant regions.

FOF aims for high precision control of the synthesized spectrum, in particular the fundamental frequency pitch. Such precision requires tight control over the timing of the FOF grains, with a resolution finer than the sample rate. This precision is achieved by taking into account the required timing of a grain, and compensating for sub-sample time offsets by adjusting grain phase.

VOSIM stands for *Voice Simulation* and was developed by Kaegi and Tempelaars (1978). In VOSIM each grain is manifested as a pulse train of decaying squared sine pulses, modeled loosely on the waveform produced by the human voice. These pulse trains are strung together to form a stream. The fundamental frequency of the stream corresponds to the repetition rate of the pulse trains, and the formant peak corresponds to the width of the individual squared sine pulses within the pulse train.

2.6.1.3 Asynchronous Granular Synthesis (AGS)

Roads (1991) refers to AGS as giving the composer a 'precision spray jet for sound, where each dot in the spray is a sonic grain'. As its name suggests, there is no synchronization between grains. Rather they are scattered in a statistical manner based on high level user input concerning their location in the time-frequency plane. Therefore while there may be precision in terms of the general shape of the grain distributions, which he calls *clouds*, the individual occurrences of particular grains are purely statistical.

The user has control over parameters associated with the cloud as a whole, and the grains that comprise it. Parameters that control the cloud include the cloud start time, duration, bandwidth, and amplitude distribution, and parameters that control the grains include the grain duration, density and wave shape. Waveforms can be synthetic or sampled.

2.6.2 Granulation

2.6.2.1 GSAMX

Like GSX, the GSAMX granulation scheme (Truax 1994) is designed to operate in real-time. It can perform continuous real-time granulation of a source sound and optionally play it back at a variable time rate.

Again a three part linear envelope is used for windowing, and each of up to twenty parallel streams can be configured. The grain waveform used is either a sampled sound with specifiable offset or a continuous real-time input stream. Truax (1994) states that the pitch and timbre† of the resulting sound are determined by the source material unless the grain duration is too short, in which case a broadband spectrum will result. Grain durations of 40-50 milliseconds were most successful.

GSAMX has most other parameters in common with GSX, such as number of streams, grain envelope shape and grain offset. One unique feature is the variable rate output capability, where a delay line holds the currently active samples of the source material. By specifying a rate, the speed at which the source material is traversed can be selected, allowing for the selective stretching of the source material over time.

2.6.2.2 FOG

FOG stands for FOF *Granular* (Clarke 1996), since it is based on its granular synthesis oriented predecessor FOF. It differs from FOF primarily in that it uses a sound file as input for its processing, rather than using a fixed sine wave. At the expense of added processing and complexity, FOG also attempts to gain tighter control over the spectral aspects of the sound processed. It achieves this tighter control by using smoother continuous grain windowing, and much finer resolution in the timing of grains.

Clarke (1996) makes the distinction between texture and timbre, pointing out that traditional granular approaches synthesize rich and interesting textures, but because of their less synchronous approach to grain generation, tend to have less control over the timbre. With FOG an explicit effort was made to achieve the required synchronization across grains necessary to achieve this timbral control.

---

† Timbre is determined by the behaviour in time of the frequency content of a sound, including its transients, which are extremely important for the identification of timbre. The presence and distribution of these frequency components, whether harmonic or inharmonic, and their onset, growth and decay in time, together with the phase relations between them, combine to give every sound its distinctive tonal quality or timbre. Often qualities of timbre are described by analogy to colour or texture (e.g. bright, dark, rough, smooth), since timbre is perceived and understood as a 'gestalt' impression reflective of the entire sound, seldom as a function of its analytic components (Truax 1999).

2.7 Summary

In this chapter we have provided an overview of the application of time-frequency methods in the computer music field. After clarifying the distinction between transform and model, we looked at the main reasons for working within a time-frequency transform framework: to provide a new view of a sound signal for the purpose of its analysis, to enable new approaches to modifying a sound signal in aesthetically interesting ways, and, to enable the synthesis of new sounds.

Two categories of constraints that impinge on all time-frequency methods were considered. First we reviewed some of the known facts and their implications in the area of *psychoacoustics*, and then reviewed the concept of *time-frequency uncertainty*, which imposes limits on the combination of time and frequency precision that we can obtain within a given transform structure.

We described a *general form* of the time-frequency transform to help us see the relationships between many of the specific examples of time-frequency methods that we reviewed. In that review we divided the methods into those that are *analysis based* and those that are *granular based*. We mentioned early examples of analysis based methods such as the *pitch synchronous* and *heterodyne filtering* methods, as well as the *sonogram*, which was used extensively for voice analysis as early as the 1930's. We then described the *Short Time Fourier Transform* (STFT), *Phase Vocoder* (PV), and *Wavelet Transform* (WT), since they are perhaps the most prominent examples of contemporary analysis based time-frequency methods. The STFT in particular has played a central role, and is often a reference point against which other methods are described and measured. It is also the foundation for other approaches, such as the PV and modern sonogram. In our description of the WT, we focused on its general description, and deferred our description of the highly efficient *fast wavelet transform* (FWT) to the next chapter.

Finally, we reviewed the *granular based* time-frequency methods. We touched on the rich history of these methods, and then described examples of some of the more prominent ones. First, we described several *quasi-synchronous granular synthesis* (QSGS) approaches, such as GSX, which has the feature of operating in real-time, and the techniques of FOF and VOSIM, which were both originally developed as voice synthesis methods. After describing the statistical approach of *Asynchronous Granular Synthesis* (AGS), we ended with several granulation approaches, which all base their synthesis on a pre-existing sound. In that category we reviewed GSAMX, and FOG, which are descendants of GSX and FOF respectively.

In the next chapter we describe the fundamentals of the *wavelet packet transform* (WPT).

Chapter 3
Wavelet Packet Transform Fundamentals

3.1 Overview

In this chapter we provide a review of the fundamentals of the *wavelet packet transform* (WPT). This background is essential since all our work uses the WPT as its foundation. Our approach is to first describe the *wavelet transform* (WT), and extend from that to the WPT as required. Since we focus on fast filter bank implementations of the WT and WPT, we make the necessary connections between wavelets and filter banks. The filter bank implementations of the WT and WPT are also referred to as the *fast* WT (FWT) and *fast* WPT (FWPT), respectively.

We also introduce the topic of *best basis* selection, describing two existing schemes in the process. Basis selection goes hand-in-hand with the WPT, since the transform result lends itself well to modifications of the initial complete tree basis.

Wavelets, Wavelet Packets, and Filter Banks

First we should clarify some terminology. Wavelets are basis functions in continuous time, or discretized versions thereof. They have the property that the entire family of wavelets used in a transform are derived from a so-called *mother wavelet* and (often) a *mother scaling function*. With the appropriate wavelet types, wavelet filtering can be implemented efficiently with *filter banks*. In that case, there is a mapping between a given wavelet and the filter that realizes it in the filter bank. As such, there is a mapping between the *mother scaling function* and *lowpass filter* and the *mother wavelet* and *highpass filter*, and consequently we will often use the terms *wavelet*, *basis function* and *filter* synonymously. Their exact relationships will be made clear later. *Wavelet packets* are basis functions that also are derived from the mother wavelet and scaling function, and so wavelet packet filters come from the mother lowpass and highpass filters as well. Since these wavelet packets and their associated filters come from a basic mother wavelet or filter, we will find it convenient at times to refer to wavelet packets as wavelets as well, especially when it is not important to distinguish between them, or when the relationship to basic wavelets is to be emphasized.

Wavelets have been with us and enjoyed great success for well over a decade now, and have been applied in a diverse range of applications. Today a solid theoretical foundation exists tying all aspects of wavelet theory and multiresolution filter bank theory together. These two disciplines developed somewhat

independently until Daubechies (1988) and Mallat (1989) determined the relationship between them.

Wavelets were developed in the mid 1980's by a group of researchers in France. They named them "*ondelettes"*, which translated to English became "*wavelets*". The research group was comprised of a geophysicist, a theoretical physicist, and a mathematician, named Morlet, Grossman, Meyer, respectively (Grossmann and Morlet 1984, Meyer 1990). *Multiresolution filter banks*, the precursors to discrete wavelets, originated in *subband coding* (Crosier Esteban, and Galand 1976, Crochiere Weber and Flanagan 1976, Esteban and Galand 1977) and *multiresolution signal analysis* (Burt and Adelson 1983).

Wavelets, whether continuous or discrete time, are used in a vast range of applications. In the broad field of signal processing they are used extensively. They have been applied particularly successfully in the field of *data compression*, especially where the data is *image*, *video* or *audio* related. For example, in image compression, whereas the *Discrete Cosine Transform* (DCT) is used in the initial version of the JPEG image compression specification, the most recent version of JPEG, called JPEG 2000 ((Joint Technical Committee 1 2000), uses the wavelet transform. Wavelets were chosen because blocking artifacts are lessened, coding gain is high (with no DC leakage to other subbands, opportunities for coding zeros increases), and the approach provides good opportunities for efficient *entropy coding*, such as with the popular *zerotree algorithm* (Shapiro 1993). One of the most well known early applications of wavelets was in the compression of FBI fingerprint files. The blocking of the DCT was found to be unacceptable, so a linear phase wavelet filter was chosen with symmetric boundary extension. For audio compression, the fact that the human auditory system responds to frequency in a logarithmic fashion (*Section 2.3.1*) has made wavelets a hot topic of research in that area as well, such as in the work of Srinivasan and Jamieson (1998) and Sinha and Tewfik (1993).

Continuing with signal processing applications, the WT have been successfully applied in the area of discontinuity detection. This success is because the transform offers information at several different time-frequency resolutions. For example, the highest frequency output (i.e., *node(1,1)* of Fig.*3-4*), which has the best time resolution and poorest frequency resolution in the transform, can capture subtle discontinuities of short duration in a signal that would not be captured by fixed resolution transforms of any practical block length. For the analysis of signals that possess a superimposed noise component, the WT outputs of several nodes can be tracked, and combined in more sophisticated discontinuity detection algorithms.

In the field of communications, filterbanks can be used as *transmultiplexers* to combine several input

signals into one wideband channel for transmission. This multiplexing is achieved by performing the synthesis part of the filterbank operation first. Low bandwidth signals are upsampled, filtered and combined in this operation, and the reverse is done at the receiver to separate the signals again. Perfect reconstruction can be achieved via a design approach closely related to that used to achieve a perfect reconstruction filterbank (Strang and Nguyen 1997).

Another area of application is in adaptive filters, which are used for such purposes as telephone line echo cancellation, and communications channel equalization. A significant challenge with these systems is that to accurately adapt the filter to the channel, a long filter is required. However, often this results in very slow convergence of the adaptation process. One solution is to divide the channel into several bands via a wavelet filter based filterbank, and adapt each of the subbands independently. Since the bandwidth of the subbands is less, the filters can be shorter, thereby speeding up convergence. The downside is the added complexity.

Finally, the WT has been useful in the analysis of processes that exhibit *fractal* characteristics. This is because the WT provides information at different *scales*, allowing for the recognition of patterns across those scales. An example of their use is in communications network traffic analysis, where packet inter-arrival times exhibit certain fractal characteristics (Gilbert, Willinger and Feldman 1999).

Basis Selection

We can combine and split subbands of the original complete tree associated with the WPT analysis transform to produce new bases. We may want a particular fixed arrangement of the subbands, or we may want to find a basis that is optimal in some way for the signal at hand. We refer to the former case as a *fixed basis*, and the latter as a *best basis*. We will explore both approaches.

A primary motivation for use of either a fixed basis or an automated basis selection process is that the manual selection of a basis of any significant depth that is in some way optimal will be prohibitively time consuming. The reason for this is that the space will generally be too large to search manually. Even when aided with a powerful processor, efficient algorithms are required. For example, the number of possible wavelet packet bases derivable from a *depth-d tree* is roughly 2^{2^d} (Ramchandran and Vetterli, 1993). For a *depth-4* transform, this leads to over 65 thousand possible bases. We discuss this issue of complexity more extensively in *Section 4.5.2*.

We can delineate various categories of bases, such as:

- *fixed versus sound dependent*
- *lossless versus lossy*
- *chosen based on the sound versus the nature of hearing*

The desired basis may be more psychoacoustically relevant. It may have a better mapping of time-frequency resolution to the characteristics of the signal. It should strike a balance between the desire to divide up the signal to learn more about it and be able to better manipulate it, and the need to not confront the user with too much complexity or too many subbands. For example, an ideal scenario would be to break down a signal into many subbands and find that most are empty, such as might occur when a sine wave is broken down with the DFT (especially if the DFT size is aligned with the wave's fundamental frequency). The basis should have a reduced number of subbands to allow for some combination of easier manipulation, quicker/better abstraction (i.e, extracting in some sense the more essential parts of a sound), lower storage requirements, lower resynthesis processing requirements, and more impact per modification for a given subband. We may also find that the distortion added via a *lossy* form of basis selection is occasionally desireable from an aesthetic perspective.

3.2 Transform Description

We now provide a more detailed mathematical description of the WT and WPT. Much of the terminology and concepts are based on the exposition in Strang and Nguyen (1997).

3.2.1 Wavelet Transform

In *Section 2.4* we gave a general form for all transform operations, having quickly proceeded to the discretized case. Later, in our overview of time-frequency methods in the field of *computer music*, we provided the general form of the discrete wavelet transform in Section *2.5.3 Wavelet Transform*. For convenience, we repeat it here.

$$c[a, n] = (1/(\sqrt{a})) \sum_{m=-\infty}^{\infty} s[m]\hat{w}[\frac{m-n}{a}] \quad \textbf{(12)}$$

This expression of the DWT simply represents an inner product operation between the signal s and the complex conjugate of the basis function, which is a scaled (by a) and shifted (by n) version of the *mother wavelet* w used in resynthesis. While orthonormality is not a prerequisite for a wavelet basis, we have

focused primarily on bases that are orthonormal so that we can perform best basis searches that take advantage of this†. Also note that if w is real, then the analysis and resynthesis wavelets are identical. To achieve the many requirements of *compact support*, orthogonality, *perfect reconstruction*, and fast implementations using discrete time filter bank representations, we limit ourselves to real valued wavelets.

We mentioned that the wavelets we use have the property of *compact support*, which means that they are highly localized in time, taking on non-zero values only over a finite, relatively short sequence of values. Certain continuous wavelets such as the *Morlet* wavelet have theoretically infinite support††. Given the compact support of our wavelets, the infinite sum in (12) will be comprised mostly of zeroes.

The notion of *scale* a is central to the WT and refers to the time domain support of the wavelet basis function. In the case of *dyadically* spaced wavelets, scale is doubled with each new depth in the transform, and consequently the signal is "analyzed" at twice the size. We are interested primarily in dyadically spaced wavelets and wavelet packets since they possess fast filter bank based implementations using the *fast wavelet transform* (FWT*)*. For the dyadic case, we define the following mappings for a and n in (12):

$$a = 2^j \quad , \quad n = p2^j = pa$$

Given these mappings, we now rewrite (12), also using a more compact form by indexing rather than using brackets where we can, and dropping the complex conjugation due to our wavelets being real-valued.

$$c_{jp} = \sum_m s[m]2^{-j/2}w[2^{-j}m - p] = \sum_m s[m]w_{jp}[m] \qquad \textbf{(13)}$$

where w_{jp}, the scaled and translated mother wavelet is defined as

$$w_{jp}[m] = 2^{-j/2}w[2^{-j}m - p] \qquad \textbf{(14)}$$

For w_{jp} to form a basis in $l^2(Z)$, we would require an infinite number of wavelets, letting $j \to \infty$ †††. In

---

†. An important class of wavelet basis functions are not orthonormal. These are referred to as *biorthogonal* wavelets since, while they do possess orthogonality in time at a given scale (called *double shift orthogonality*), orthogonality across scales for the same time segment does not exist. These wavelets are popular in image compression because they are symmetric and therefore exhibit linear phase. Some are nearly orthogonal and could also be used in our best basis searches.

††. The *Morlet* wavelet is not orthogonal (or biorthogonal), nor can it be implemented with the filter bank based DWT. It is defined as *morl(x) = exp(-x^2/2) * cos(5x)*. A complex version of this was used by Kronland-Martinet (1988). Note also that the discrete wavelets we work with do not possess such explicit definitions. Rather they are defined by their filter coefficient values.

practice we do not extend scale to infinity. Instead, we use a *scaling function* ϕ to represent the lowest frequency part of the signal.

Analogous to the wavelet case in (14), the scaling function at a given transform depth is also a scaled and translated version of a mother function, in this case the mother *scaling function*. The relationship is

$$\phi_{jp}[m] = 2^{-j/2}\phi[2^{-j}m - p] \quad \textbf{(15)}$$

The component of s that is correlated with ϕ_{jp} at a given level j in the wavelet transform is

$$d_{jp} = \sum_m s[m]\phi_{jp}[m] \quad \textbf{(16)}$$

So then s can be represented by its projection onto w_{jp} for $j = 1, 2, \ldots, j_{\max}$, plus its projection onto $\phi_{j_{\max} p}$, where $j_{\max} = \log_2(N)$ and N is the block size of the WT. As such, we can resynthesize s as

$$s[m] = \sum_{j,p} c_{jp} w_{jp}[m] + d_{j_{\max} p}\phi_{j_{\max} p}[m] \quad , \quad j = 1, 2, \ldots, j_{\max} \quad \textbf{(17)}$$

where p ranges over all time offsets at scale j for which the signal s is defined.

To help clarify the foregoing description, we again use the *Haar* basis functions for an example. Recall that the *depth-2 Haar* basis functions were introduced in an example in *Section 2.4*. We use those same basis functions now to calculate the wavelet transform coefficients for a simple input signal. The basis is dyadically spaced, which corresponds to the case we focused on in the preceding equations. The example input signal we use in the transform is shown in Fig.*3-1*.

The basis functions and transform coefficients that result in mapping the example input signal to these basis functions are shown in Fig.*3-2*. When we described the basis functions earlier, they were labelled according to the naming scheme employed to delineate the general form of a time-frequency transform. Here we label them according to our wavelet transform equations. Since the transform is of *depth-2*, we have three basis functions; one scaling function at *depth-2* (ϕ_2 in the equations, and phi2 in the figure), one wavelet at

---

†††By $l^2(Z)$, we mean the space of sequences s that can be represented by a linear combination of the w_{jp} such that the square of the difference between s and this linear combination of the w_{jp} approaches 0 as the number of basis functions used in the representation approaches infinity (i.e., $j \to \infty$).

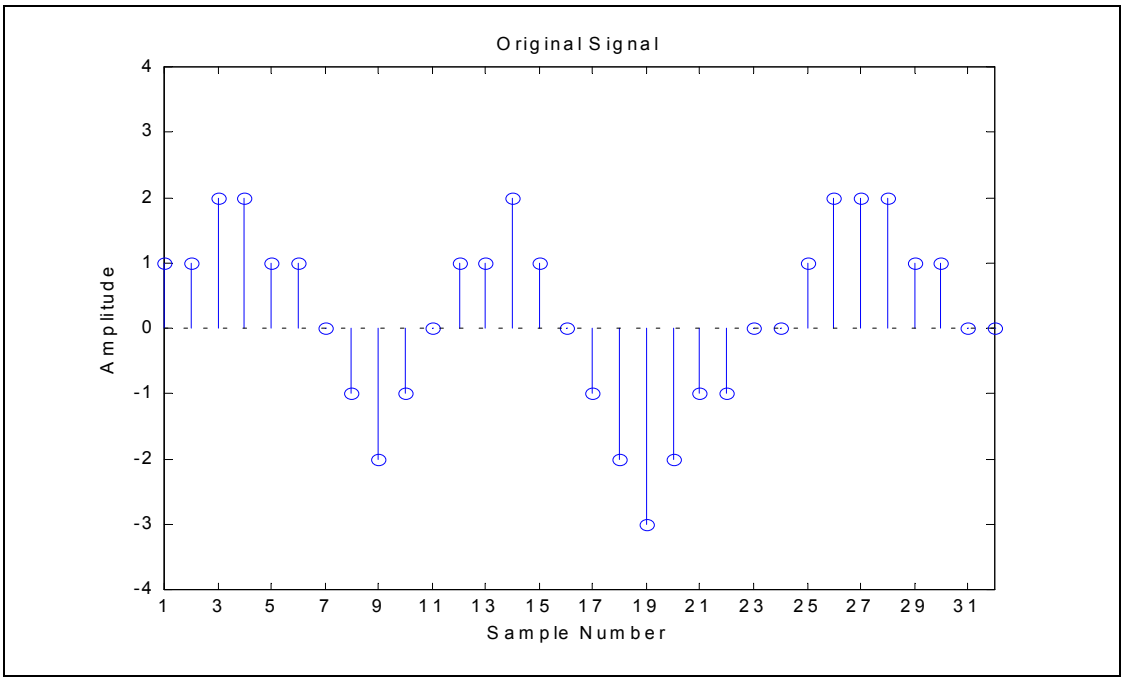

Figure 3-1: Signal used in Example WT

depth-2 (w_2), and one wavelet at *depth-1* (w_1). These correspond to the plots on the left side of the figure, from bottom to top, respectively.

The resulting transform coefficients are shown at the right of the plot, lined up horizontally with the corresponding basis function. Note that we have downsampled the coefficients in accordance with the relative bandwidth of the subband within which the coefficients reside. We can confirm that the result is *critically sampled*, since the total number of transform coefficients (8+8+16) is the same as the number of samples in the original signal. Furthermore, we can verify that the transform is correct by taking the inner products of the transform coefficients and basis functions in the plot for each subband and summing the results to re-create the original signal shown in Fig.*3-1*.

As we have stated, the fast DWT implementation of the WT relies on a filter bank based approach. We want to work with the filters that correspond to the wavelet and scaling function directly, and in the next section, show the relationship between the two[†].

[†] In the case of our *Haar* example, the discretized wavelet and scaling functions (which we have been working with thus far) look the same as their corresponding high and low pass filters. With other wavelet families the correspondence is not so obvious. We will see that it is specified via the *wavelet* and *dilation equations*.

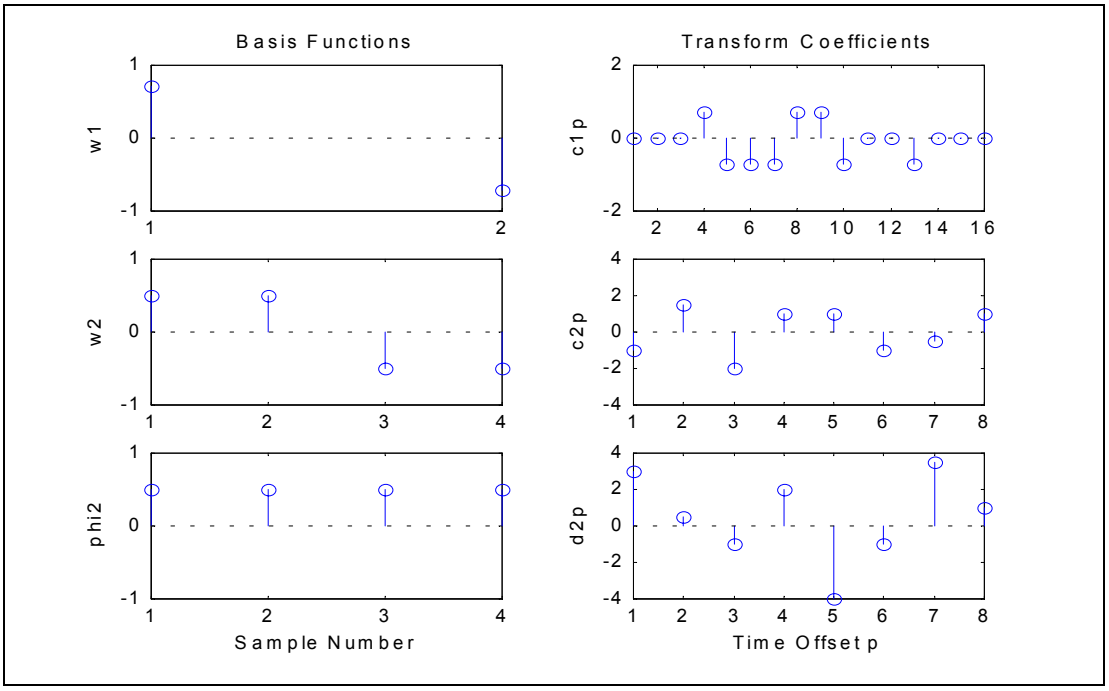

Figure 3-2: Example DWT Basis Functions and Transform Coefficients

3.2.2 Linking Wavelets to Filters

The scaling function $\phi(t)$ is the solution to the following *multi-scale equation*

$$\phi(t) = \sqrt{2}\sum_{m=0}^{M} h_0[m]\phi(2t-m) \qquad \textbf{(18)}$$

where h_0 is the M coefficient lowpass filter, which along with a highpass filter, is used to construct the hierarchical filter bank. The filter coefficients are normalized to unit energy so that the basis is *orthonormal*. Equation (18) is referred to as the *dilation equation*. It includes a function in continuous time $\phi(t)$ and a set of coefficients from discrete time $h_0[m]$. The resulting $\phi(t)$ that is derived from the lowpass filter h_0 via (18) can then be discretized to produce the discrete time *mother scaling function* used in (15).

A corresponding multi-scale equation defines the wavelet $w(t)$ as

$$w(t) = \sqrt{2}\sum_{m=0}^{M} h_1[m]\phi(2t-m) \quad \textbf{(19)}$$

where h_1 defines the M coefficient highpass filter of the filter bank, also with coefficients normalized to unit energy. This equation is referred to as the *wavelet equation*. As with the scaling function case, the discrete time *mother wavelet* is easily obtained.

The dilation and wavelet equations link filter banks to wavelets. All the wavelets we use in our work are derived from filter coefficients. We design the filters imposing certain requirements such as *perfect reconstruction, alias cancellation,* and possibly also *orthogonality*[†]. With the dilation and wavelet equations, we can see what the corresponding wavelets look like. Solving these equations is not always easy and there are several approaches. Later we show one such approach, called the *cascade algorithm*.

Thus far we have shown the relationship between the mother scaling function and mother wavelet function and their corresponding lowpass and highpass filters respectively (*Equations* (18) and (19)). We have shown how these mother functions generate the wavelets and scaling functions at the different transform depths necessary to perform the WT (*Equations* (14) and (15)). We have also shown how these functions produce the wavelet and scale function transform coefficients (*Equations* (13) and (16)) of the analysis operation. Finally, we have shown how the original signal is resynthesized via (17).

What we still need to show is how the filters themselves are designed, how they relate to the filter bank as a whole, and why this filter bank approach is more efficient than using the wavelets and scaling function approach just described. We defer the design of particular filters to Section *3.3 Filter Design*, for now using again the *Haar* wavelet and associated filter bank to help us describe the tree structured filter bank as a whole. In the process, we also describe the efficient *fast wavelet transform* (FWT) implementation of the DWT.

3.2.3 Filter Banks and the FWT

Wavelet and wavelet packet based filter banks are hierarchical. They are constructed using only two filters, a lowpass filter and a highpass filter. In the case of the WT, the lowpass filter output from the initial or first

---

[†] We would also like linear phase, but other than the simple *Haar* case, it is not possible to achieve orthogonality and linear phase with the same filter. We can, however, achieve near linear phase filters (i.e., symmetric FIR) that are orthogonal, such as the *sym* wavelets used in most of our testing.

level filtering of the signal is applied to the same lowpass and highpass filters. The output of this last lowpass filter is again applied to the same low and highpass filters, and so on, until all samples in the block of length *N* have been exhausted through the successive filterings. Downsampling by two occurs after each filtering operation. As we will see later, the highpass filter outputs are also applied to subsequent low and highpass filterings in the case of the WPT. Because of the recursive nature of this approach, a tree-like structure emerges when we construct either a time or frequency domain representation of it.

Fig.*3-3* shows the tree of filters in a *depth-3* analysis filter bank, where H_0 represents the lowpass filter, H_1 the highpass filter, and the downward arrow followed by the number *2* indicates downsampling by a factor of two.

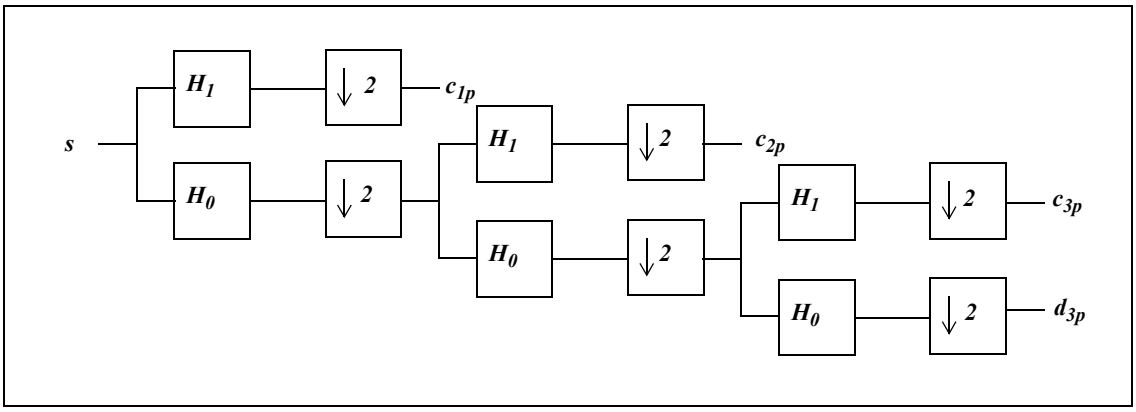

Figure 3-3: WT Filter Bank Structure - Depth-3 Analysis Bank

The downsampling of the filter outputs is an important part of the filter bank operation, resulting in what is called a *critically sampled* transform. Without downsampling, each filtering stage would double the number of samples at its input. Also, by downsampling, we can apply the identical filters at each stage and have them operate at successively larger scales. We downsample by the factor of two specifically because there are two filters at each stage. Also note that it is because we use exactly two filters in our hierarchical splitting of subbands that the so-called *dyadic* spacing we have referred to earlier is produced.

Upon resynthesis, upsampling by two occurs with each stage of lowpass and highpass synthesis bank filtering. With appropriate filter design the entire operation achieves *perfect reconstruction*, with no distortion added to the signal. The only change from the original signal is a delay.

The hierarchical filtering scheme we have just described is in fact a representation of the *pyramid*

algorithm (Mallat 1989). When it is implemented in matrix form it is called the *fast wavelet transform* (FWT). It is so efficient because at each stage other than the first, we are filtering the output of a previous filtering operation. Such a recursive approach results in a maximum of *2LN* multiplications for a length *N* transform when filters of length *L* are used. This calculation is described later in *Section 4.5*.

Tree Diagrams

We now introduce the *tree diagram*. Fig.*3-4* shows the tree diagram associated with a *depth-4* WT. It reflects the structure of its corresponding hierarchical filter bank, such as the structure shown in Fig.*3-3*. Since we will show several tree diagrams throughout this work, we should be familiar with its naming and numbering conventions, and describe them now.

Moving from top to bottom in the diagram of Fig.*3-4*, frequency is divided into ever smaller segments. Each line that emanates down and to the left of a node represents a lowpass filtering operation (h_0), and each line emanating down and to the right a highpass filtering operation (h_1). The nodes that have no further nodes emanating down from them are referred to as *terminal nodes*, *leaves*, or *subbands*. We refer to the other nodes as *non-terminal*, or *internal* nodes. The tree diagrams in this document use a relatively larger font to designate *terminal* nodes that have not had their associated coefficients zeroed, and a relatively smaller font to designate all other nodes (zeroing the coefficients associated with a given terminal node is sometimes done as part of a best basis selection approach described later in this document). As such, the tree node labeling scheme provides a simple mechanism for indicating the nodes in the tree that we can work with when imparting modifications on the signal[†].

For the node (*j,k*), *j* denotes the depth within the transform (tree) and *k* the position. For example, at node (0,0) no filtering has taken place, and we simply have the original sequence of time samples. Lowpass filtering this with h_0 produces node (1,0) and highpass filtering with h_1 produces (1,1). These filtering operations are equivalent to finding the correlation of the signal with the scaling function ϕ_{1p} for node (1,0) and the correlation of the signal with the wavelet function w_{1p} for node (1,1).

[†] When viewing full colour versions of this document (such as the *pdf* electronic file format), the larger font nodes will appear green and the smaller font nodes will appear red.

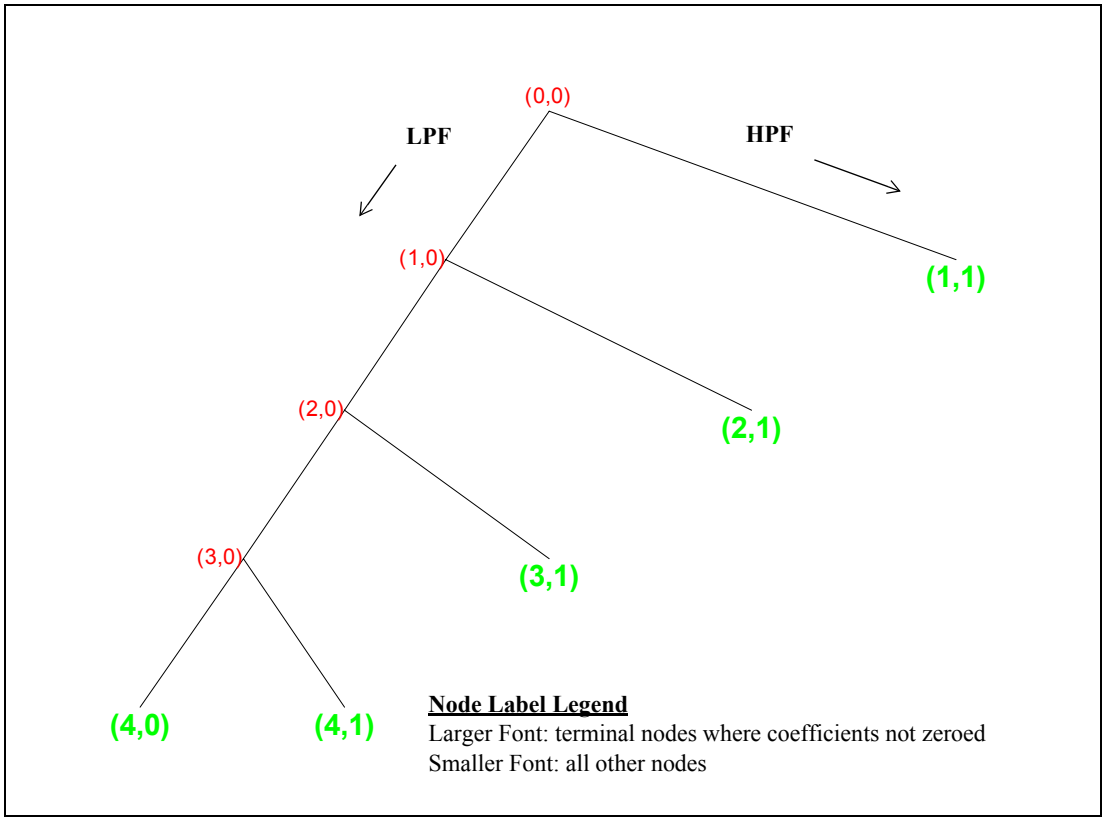

Figure 3-4: Tree Diagram of Wavelet Transform

Going down the tree to the next depth, we see that (2,0) and (2,1) emanate from (1,0). From the filter perspective, the samples at (1,0) are applied to the filters h_0 and h_1. Multiresolution is achieved because the coefficients at (1,0) have been downsampled by two to achieve *critical sampling*. From the wavelet and scaling function perspective, the correlations between both ϕ_{2p} and w_{2p} and the samples of (1,0) are determined through this operation.

Filter Bank Details

Now we describe the details of the filter bank. To help the discussion, we again use the example of the *Haar* basis functions. The starting point is a pair of two-coefficient FIR filters. One is lowpass and the other highpass, and their respective impulse responses h_0 and h_1 are

$$h_0 = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}) \quad , \quad h_1 = (\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}) \tag{20}$$

While these are not particularly good filters in terms of providing isolation between bands, the lowpass filter h_0 does possess a *zero* at the *Nyquist frequency* (π rad/second) and the highpass filter h_1 does possess a *zero* at DC (0 rad/second)†. We can see this by forming the *Z Transform* and evaluating at $z = e^{j\omega}\Big|_{\omega = 0, \pi}$. For the lowpass case this results in

$$H_0(z)\Big|_{z = e^{j\pi}} = \left(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}z^{-1}\right)\Big|_{z = e^{j\pi}} = \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}e^{-j\pi} = \frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}} = 0$$

and for the highpass case it results in

$$H_1(z)\Big|_{z = e^{j0}} = \left(\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}z^{-1}\right)\Big|_{z = e^{j0}} = \frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}e^{-j0} = \frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}} = 0$$

Note that the magnitude of the coefficients are $\frac{1}{\sqrt{2}}$ rather than $\frac{1}{2}$. While the latter would preserve energy across the transform without downsampling, the former is required since downsampling is an important part of the transform.

The lowpass filter produces an output $t[m]$ that is a moving average of the input $s[m]$. We can express it as the convolution

$$t[m] = \frac{1}{\sqrt{2}}s[m] + \frac{1}{\sqrt{2}}s[m-1] \qquad \textbf{(21)}$$

We can also express this in the form of an infinite matrix as

$$\begin{bmatrix} \vdots \\ t[-1] \\ t[0] \\ t[1] \\ \vdots \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} \ddots & \ddots & & & \\ 1 & 1 & & & \\ & 1 & 1 & & \\ & & 1 & 1 & \\ & & & \ddots & \ddots \end{bmatrix}\begin{bmatrix} \vdots \\ s[-1] \\ s[0] \\ s[1] \\ \vdots \end{bmatrix} \qquad \textbf{(22)}$$

or,

$$\boldsymbol{t_0} = \boldsymbol{H_0 s} \qquad \textbf{(23)}$$

The rows of $\boldsymbol{H_0}$ are the basis vectors, which are the time-reversed elements of h_0. This time-reversal is necessary for expressing convolution as an inner product, and is only evident in the case of the highpass filter

---

†. The importance of zeroes at π is related to the *accuracy of approximation* achievable in representing a continuous signal by the *scaling function* and *wavelets* associated with the discrete time filters (*Section 3.3.3.*).

for the *Haar* case, where the filter's elements are actually different (see later matrices).

The highpass filter can be expressed similarly as

$$\boldsymbol{t_1} = \boldsymbol{H_1 s} \tag{24}$$

A perfect reconstruction filter bank can be created with these two filters, such as in Fig.*3-5*. Filter inputs and outputs are vectors. The blocks are all linear operators based on filter definitions such as that given for the lowpass analysis case in (22). Since the analysis filters are orthonormal in this case, the resynthesis filters are simply the transpose of the analysis filters.

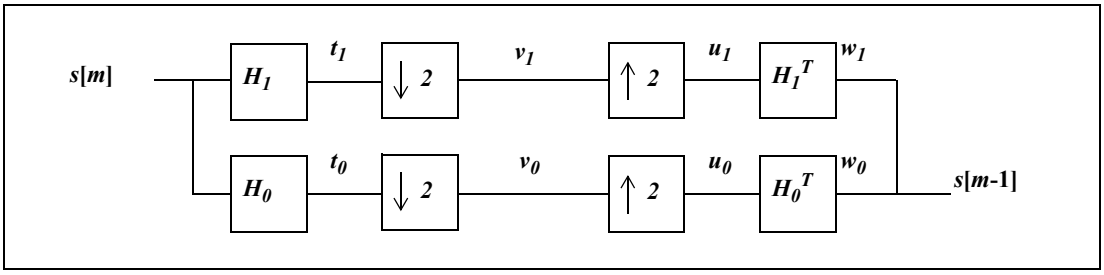

Figure 3-5: Simple Two-Channel Orthogonal Filter Bank

There are at least two benefits that can come with critical sampling: minimization of storage requirements, and minimization of processing requirements. In the scheme described and illustrated in Fig.*3-5* we realize only the first benefit. Downsampling does indeed occur and therefore we minimize storage needs. But the downsampling occurs after the filtering on the analysis side, and before the filtering on the resynthesis side. This sequence of operations means that all filtering is done at full rate, which is not efficient. After filtering on the analysis side for example, half the output samples are discarded due to downsampling. It would be more efficient to not produce those discarded samples in the first place. Such an approach can be implemented using the *polyphase* form.

Downsampling occurs before *analysis* filtering and after *resynthesis* filtering in the *polyphase* form. This requires grouping the input signal samples into *phases* prior to filtering. In the *Haar* example, this results in filtering $s[0]$ and $s[1]$, then $s[2]$ and $s[3]$, etc. We do not filter $s[1]$ with $s[2]$, nor $s[3]$ with $s[4]$. In terms of the block form matrix representation of (22), this is equivalent to removing every second row.

Let us call the downsampled version of the lowpass filter matrix $\boldsymbol{H}_{0\,\mathrm{D}}$. We can stack it on top of the downsampled highpass filter to create a matrix representing the entire analysis filter bank.

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{H}_{0\,\mathrm{D}} \\ \boldsymbol{H}_{1\,\mathrm{D}} \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & & & \\ & & 1 & 1 & \\ -1 & 1 & & & \ddots \\ & & -1 & 1 & \\ & & & & \ddots \end{bmatrix} \tag{25}$$

Notice that the rows are mutually orthogonal. The orthogonality amongst the rows of either the lowpass or highpass filter in this matrix is called *double shift orthogonality.* The *double shift* comes from downsampling by two. This condition must exist for all dyadic wavelet filters. The orthogonality between the rows of the lowpass filter and the rows of the highpass filter results in a fully orthogonal transform. When only *double shift orthogonality* exists, the transform and wavelets are said to be *biorthogonal*. Also note the time-reversal of the highpass elements as mentioned earlier.

Since in the fully orthogonal case the inverse is the transpose, we can express the entire synthesis bank as

$$\begin{bmatrix} \boldsymbol{H}_{0\,\mathrm{D}} \\ \boldsymbol{H}_{1\,\mathrm{D}} \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{H}_{0\,\mathrm{D}} \\ \boldsymbol{H}_{1\,\mathrm{D}} \end{bmatrix}^{T} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 & & & \\ 1 & 1 & & & \\ & & 1 & -1 & \\ & & 1 & 1 & \\ & & & & \ddots \end{bmatrix} \tag{26}$$

If we follow one matrix with the other we get perfect reconstruction.

$$\boldsymbol{A}^{T}\boldsymbol{A} = \begin{bmatrix} \boldsymbol{H}_{0\,\mathrm{D}}{}^{T} & \boldsymbol{H}_{1\,\mathrm{D}}{}^{T} \end{bmatrix} \begin{bmatrix} \boldsymbol{H}_{0\,\mathrm{D}} \\ \boldsymbol{H}_{1\,\mathrm{D}} \end{bmatrix} = \boldsymbol{H}_{0\,\mathrm{D}}{}^{T}\boldsymbol{H}_{0\,\mathrm{D}} + \boldsymbol{H}_{1\,\mathrm{D}}{}^{T}\boldsymbol{H}_{1\,\mathrm{D}} = \boldsymbol{I} \tag{27}$$

We described the *pyramid* algorithm earlier. To see it fully from the filter bank perspective, which is the FWT, we take the case where $N = 4$ as an example. The depth of the WT in this case will be two. With reference to Fig.*3-3*, this corresponds to the first two filtering stages. To reflect this operation with the matrices just described here for the *Haar* case, the overall analysis filtering operation can be represented as the concatenation of two matrices. If we call the analysis matrix that results from these two wavelet stages as $\boldsymbol{A}_{w2}$, then

$$\boldsymbol{A}_{w2} = \begin{bmatrix} \boldsymbol{A}_{\mathrm{D}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} [\boldsymbol{A}] \tag{28}$$

where $\boldsymbol{A}$ is the basic analysis matrix of (25), and $\boldsymbol{A}_{\mathrm{D}}$ is $\boldsymbol{A}$ with every second row removed to reflect the downsampling by two. With more levels, there is more downsampling, and the additional matrices become more and more sparse ($\begin{bmatrix} \boldsymbol{A}_{\mathrm{D}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}$ is already quite sparse). In (28), the identity matrix $\boldsymbol{I}$ reflects that the highpass output is not further filtered in the WT.

We now link our example lowpass and highpass *Haar* filters to their associated scaling function and wavelet. Rather than deriving the result, we simply state it and then show it is true. Recall that the scaling function must satisfy *dilation equation* (18). The solution, if there is one, depends only on the lowpass filter h_0. It turns out that the solution is the *box function*.

$$\phi(t) = \begin{cases} 1 \text{ for } 0 \le t < 1.0 \\ 0 \text{ otherwise} \end{cases} \quad \textbf{(29)}$$

Fitting this into the dilation equation, we can see the $\phi(2t)$ is a half box extending from $t = 0$ to 0.5. $\phi(2t-1)$ is another half box extending from $t = 0.5$ to 1.0. As such, the multi-scale equation $\phi(t) = \phi(2t) + \phi(2t-1)$ is satisfied.

The corresponding wavelet is easily obtained from the *wavelet equation* (19), by inserting our solution for the scaling function in (29). Specifically, for the *Haar* case the wavelet equation is

$$w(t) = \phi(2t) - \phi(2t-1) \quad \textbf{(30)}$$

and the result is a difference of half boxes given by

$$w(t) = \begin{cases} 1 \text{ for } 0 \le t < 0.5 \\ -1 \text{ for } 0.5 \le t < 1.0 \\ 0 \text{ otherwise} \end{cases} \quad \textbf{(31)}$$

Cascade Algorithm

In the preceding *Haar* example, rather than finding $\phi(t)$, we merely stated it. Given $\phi(t)$, solving (19) to find $w(t)$ is trivial. But how is $\phi(t)$ found? There are many approaches, including the *cascade algorithm* and the *infinite product formula* (Strang and Nguyen 1997). We describe the former.

The *cascade algorithm* iterates the *dilation equation* with an initial guess of the scaling function. If convergence is achieved, the result solves the dilation equation. Typically the cascade algorithm uses the *box function* for its initial guess $\phi^0(t)$ (as defined in (29)), where the superscript denotes the iteration count. Then the dilation equation is iterated as follows

$$\phi^{(i+1)}(t) = \sqrt{2}\sum_m h_0[m]\phi^{(i)}(2t-m) \quad \textbf{(32)}$$

The *Haar* example is trivial, since convergence occurs immediately due to the initial guess being equal to the solution. Of course this in not otherwise the case.

3.2.4 Wavelet Packet Transform

When the WT is generalized to the WPT, not only can the lowpass filter output be iterated through further filtering, but the highpass filter can be iterated as well. This ability to iterate the highpass filter outputs means that the WPT allows for more than one basis function (or *wavelet packet*) at a given scale, versus the WT which has one basis function at each scale other than the deepest level, where it has two.

The set of *wavelet packets* collectively make up the complete family of possible bases, and many potential bases can be constructed from them. If only the lowpass filter is iterated, the result is the wavelet basis. If all lowpass and highpass filters are iterated, the *complete tree* basis results. This basis has a time-frequency partitioning like that of the STFT. Between these two extremes lie a large number of possible bases and their associated *subtrees*. Nodes can be merged or split based on whatever requirements are at hand. In all cases, the *leaves* of each connected subtree of the complete wavelet packet tree form a basis of the initial space; they span the space in a linearly independent fashion.

In our work we have generally created the *complete tree* basis first, and then either re-formed the basis to a given fixed basis, or applied a *best basis* algorithm to find a basis that minimizes a given *cost function*. THe process of basis selection is described in *Sections 3.6* and *3.7*.

Fig.*3-6* shows the tree diagram of a *depth-3 complete tree* basis. As with the WT tree diagram of Fig.*3-4*, *j* denotes the depth within the transform and *k* the position of each node (*j,k*). But now the position index *k* conveys more information, telling us specifically which wavelet packet it corresponds to for a given scale. We refer to the associated wavelet packet as w_{jkp}; analogous to the wavelet w_{jp}. The tree diagrams do not convey time domain information, and so the index *p* is not used in node naming. With regards to wavelet packet naming, if all our packets are at the same scale, we may occasionally simply refer to them as w_k, as in $w_0 - w_7$ for the wavelet packets at *depth-3*. Furthermore, while w_{j0} is actually either the scaling function, or derived from the scaling function, rather than use the separate symbol ϕ to denote it as we did in the WT case, we find it convenient here to label it consistently with the other wavelets.

.

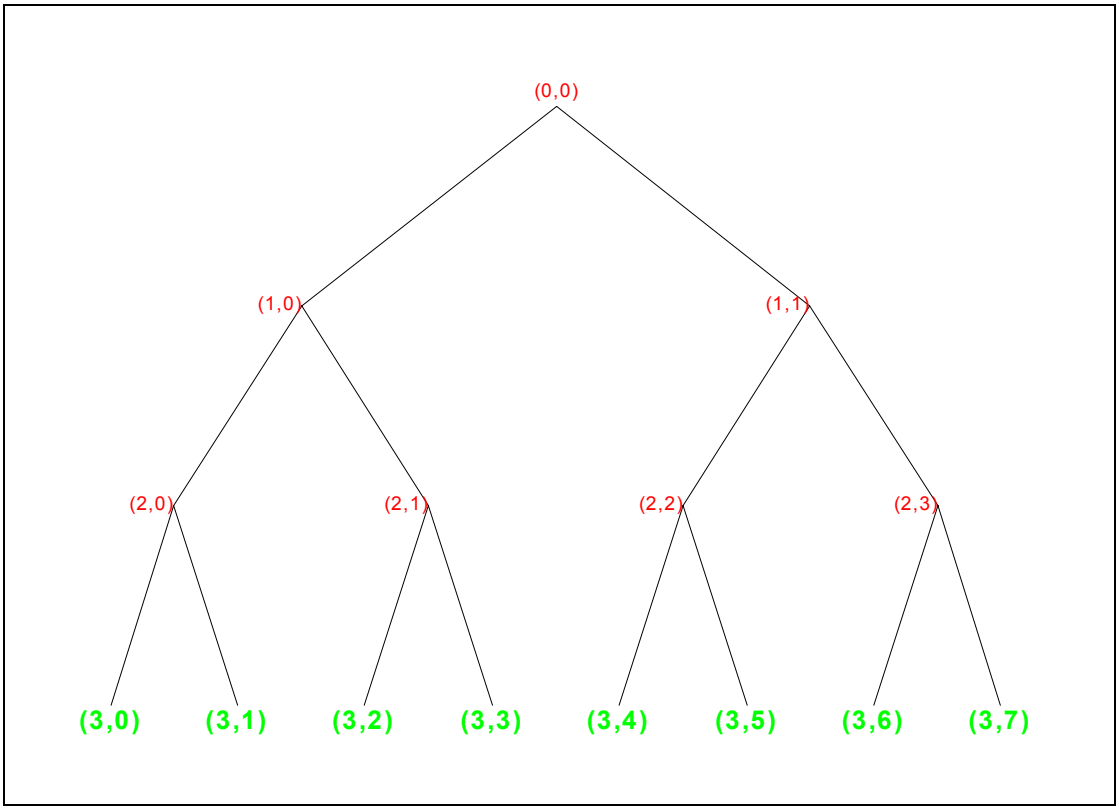

Figure 3-6: Tree Diagram of Wavelet Packet Transform - Depth-3

As with the WT, the WPT does not require the explicit definition of the wavelet. The filter definitions are enough, and we sometimes refer to this filter bank based implementation as the *fast* WPT (FWPT). To see what the equivalent *wavelet packets* would be at a given level of decomposition, we can do a recursion of them at each node moving down the tree, to get the wavelets at the next level. Specifically, if we split a wavelet packet node at level j and position k into two nodes at level $j+1$ and locations $2k$ and $2k+1$, we get the following two packets

$$w_{j+1,2k}(t) = \sum_{m=0}^{M-1} h_0[m] w_{j,k}(2t-m) \qquad \textbf{(33)}$$

and

$$w_{j+1,\,2k+1}(t) = \sum_{m=0}^{M-1} h_1[m]\, w_{j,\,k}(2t-m)\,. \tag{34}$$

Then analogous to the wavelet transform case of (13), the WPT transform coefficients c_{jkp} are given by

$$c_{jkp} = \sum_{m} s[m]\, w_{jkp}[m] \tag{35}$$

and the original signal can be expressed in terms of these coefficients and the corresponding wavelet packets as

$$s[m] = \sum_{j,\,k,\,p} c_{jkp}\, w_{jkp}[m] \quad , \quad (j,k) \in \text{ all leaf nodes of basis} \tag{36}$$

where p ranges over all time offsets at scale j for which the signal s is defined†.

Coming back to the *Haar* wavelet example, Fig.*3-7* shows the wavelet packets associated with the *depth-3* complete tree basis.

These packets are numerically arranged in what is referred to as the *natural ordering*. Wavelet packets *w0* through *w7* correspond in sequence to terminal nodes (*3,0*) through (*3,7*) found in moving from left to right along the bottom row of nodes in Fig.*3-6*. This sequence does not result in an ordering of wavelets at a given level from left to right of lowest frequency to highest frequency. We discuss this further in *Section 3.4*.

The efficiencies of the FWT are realized to a large extent when applied to the WPT as well. In the *depth-2* WT example of the *Haar* wavelet given earlier, we arrived at (28). While the additional processing associated with iterating the highpass filter outputs are incurred in the WPT case, its recursive nature still provides great benefit. For that same example, if $\boldsymbol{A}_{wp2}$ is the analysis matrix for the *depth-2* WPT, then

$$\boldsymbol{A}_{wp2} = \begin{bmatrix} \boldsymbol{A}_{\mathbf{D}} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{A}_{\mathbf{D}} \end{bmatrix} \begin{bmatrix} \boldsymbol{A} \end{bmatrix} \tag{37}$$

As with the WT case, the additional matrices become more and more sparse as depth increases due to downsampling. Later, in *Section 4.5.1*, we show that the computations required for a length N WPT are of order $N \log_2 N$; the same as for the *Fast Fourier Transform* (FFT).

---

†.Use of the *Haar* filter based wavelet packets all at the same scale (depth) in the WPT is equivalent to performing the *Walsh Transform*.

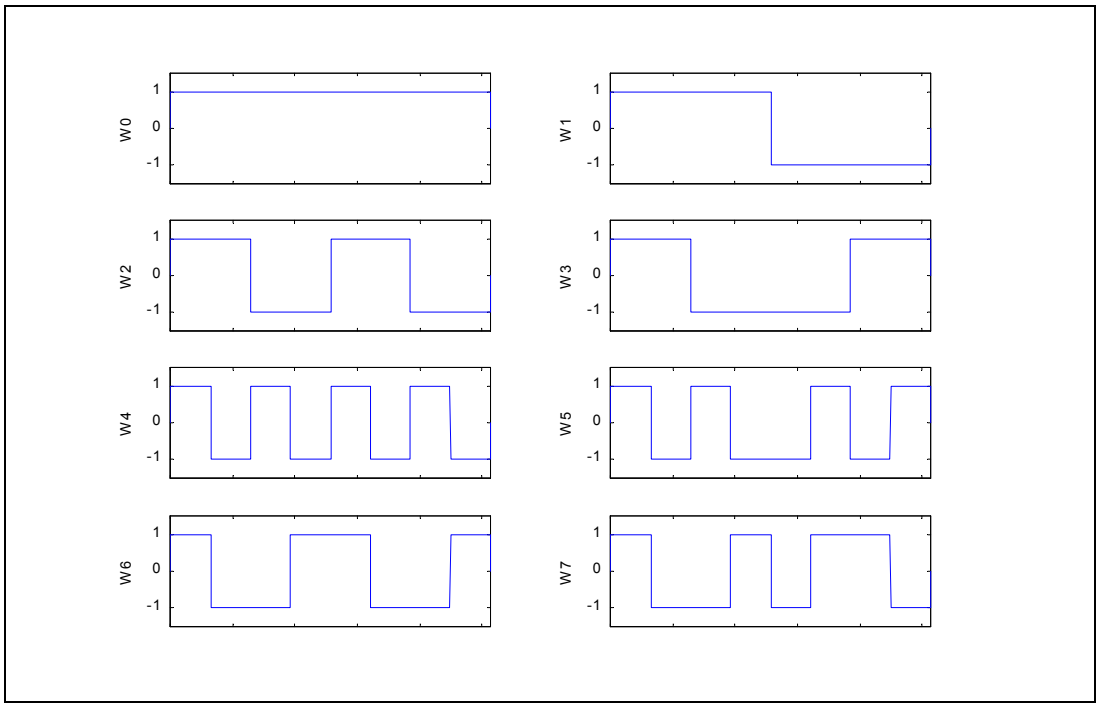

Figure 3-7: Haar Wavelet Packets for Depth-3 Transform (natural ordering)

3.3 Filter Design

We now show how the lowpass and highpass filters that comprise the filter bank are designed. Since the filters are directly related to corresponding wavelets, one can just as well view this as wavelet design. The approach we take is to form the complete filter bank transfer function, impose the requirements of *perfect reconstruction* and complete *alias cancellation* on this transfer function, and then look at what filters might satisfy these requirements. We look at the specific case of the Daubechies construction, which includes as a subset a class of orthonormal wavelets, one of which is the familiar *Haar* wavelet used as our example in the preceding discussions.

3.3.1 Perfect Reconstruction and Alias Cancellation

To begin, we repeat the basic filter bank block diagram given earlier in Fig.*3-5*, but remove the orthogonality requirement for the moment. Removal of the orthogonality requirement means that we do not simply transpose the analysis filters to obtain the synthesis filters. Instead, we refer to these lowpass and highpass synthesis filters more generally as F_0 and F_1 respectively.

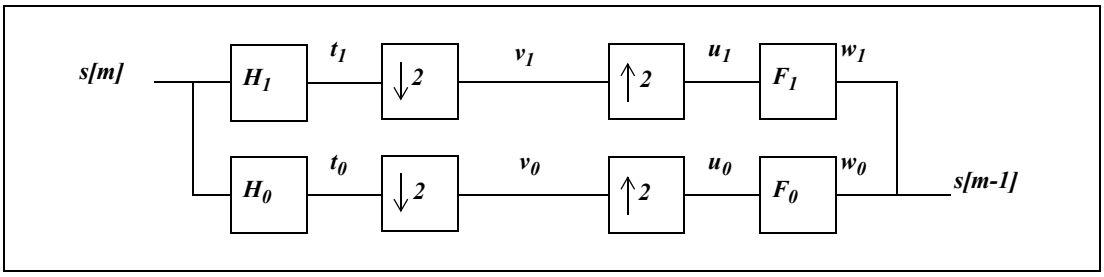

Figure 3-8: General Two-Channel Filter Bank

To achieve perfect reconstruction, whereby no distortion is introduced into the signal by the filter bank, the synthesis filters should undo what the analysis filters have done. This "undoing" is equivalent to requiring that the synthesis filters be the inverse of the analysis filters. For our filters to be causal, the filter bank as a whole will introduce some delay *l*. As such, we can express the perfect reconstruction requirement in the *Z* domain as

$$F_0(z)H_0(z) + F_1(z)H_1(z) \;=\; 2z^{-l} \qquad \textbf{(38)}$$

where the factor of *2* that multiplies z^{-l} is due to the requirement that the filters preserve energy after downsampling.

The filters we use can never be perfect *brick wall* filters. The lowpass filter has some non-zero response in the highpass region ($\frac{\pi}{2} \le \omega < \pi$) and vice-versa. Since we downsample and upsample after this imperfect analysis filtering, *aliasing* will inevitably result.

The combination of downsampling followed by upsampling results in the zeroing out of all odd-numbered components of the input. For a two-channel filter bank, this results in the appearance of "*-z"* terms (or $\omega + \pi$ in the frequency domain) in the overall response. Specifically, if the sequence $\underline{\mathbf{u}}$ is the sequence **t** after the combination of *downsampling-by-M* followed by *upsampling-by-M*, then the *z*-transform *U(z)* of **u** is (Strang and Nguyen 1997)

$$U(z) \;=\; \frac{1}{M}[T(z) + T(ze^{(i2\pi)/M}) + \ldots + T(ze^{(i2\pi(M-1))/M})] \qquad \textbf{(39)}$$

and for *M=2*, our two-channel case,

$$U(z) = \frac{1}{2}[T(z) + T(-z)] \quad \textbf{(40)}$$

Based on this relationship, the transform of $\mathbf{u_0}$ for the lowpass case in Fig.*3-8* is given by

$$U_0(z) = \frac{1}{2}[H_0(z)S(z) + H_0(-z)S(-z)] \quad \textbf{(41)}$$

The aliasing term $H_0(-z)S(-z)$ cancels all odd components, resulting in an *even* function.

This aliasing term is multiplied by the filter $F_0(z)$ during synthesis, resulting in $F_0(z)H_0(-z)S(-z)$. For alias cancellation to occur, this lowpass aliasing term must be cancelled by the highpass aliasing term $F_1(z)H_1(-z)S(-z)$. Therefore, we can formally state the alias cancellation condition on the filter bank as

$$F_0(z)H_0(-z) + F_1(z)H_1(-z) = 0 \quad \textbf{(42)}$$

Equations (38) and (42) make up the two conditions required for the filter bank to exhibit perfect reconstruction and alias cancellation, respectively.

3.3.2 Design Method

To design the four filters of the filter bank, we start with the following choice to guarantee the alias cancellation requirement of (42).

$$F_0(z) = H_1(-z) \quad \text{and} \quad F_1(z) = -H_0(-z) \quad \textbf{(43)}$$

We further define the following product filters as the concatenation of the analysis and synthesis filters for both the lowpass and highpass channels as

$$P_0(z) = F_0(z)H_0(z) \quad \text{and} \quad P_1(z) = F_1(z)H_1(z) \quad \textbf{(44)}$$

where these terms come directly from the perfect reconstruction requirement of (38). Given the alias cancellation choices of (43), it follows directly that

$$P_1(z) = -H_0(-z)H_1(z) = -H_0(-z)F_0(-z) = -P_0(-z) \quad \textbf{(45)}$$

which simplifies (38) to

$$P_0(z) - P_0(-z) = 2z^{-l} \quad \textbf{(46)}$$

We now have the necessary relationships to design a two-channel perfect reconstruction filter bank with

alias cancellation. The steps are

1. Design the lowpass product filter P_0 such that (46) is satisfied.
2. Factor $P_0(z)$ into $F_0(z)H_0(z)$, and then find the highpass channel filters H_1 and F_1 from (43).

3.3.3 Daubechies Construction Example

There are many approaches to designing P_0. One popular choice uses the Daubechies construction

$$P_0(z) = (1 + z^{-1})^{2p} Q(z) \quad \textbf{(47)}$$

where $Q(z)$ is a polynomial of degree *2p-2* chosen such that (46) is satisfied. The binomial factor $(1 + z^{-1})^{2p}$ ensures a maximum number of zeros (*p*) at z = -1 (or $\omega = \pi$), and filters possessing this maximum number of zeroes are often referred to as *maxflat* or *binomial*. The number of zeroes possessed by the filter at $\omega = \pi$ is important when wavelets and scaling functions in continuous time are used rather than discrete time filter banks. In this continuous time case, the issue of the *accuracy of approximation* of the signal by the wavelet expansion arises. This accuracy depends both on the signal and the wavelets and scaling function chosen. For a given signal and wavelet type, the approximation gets better as more wavelets of increasingly fine time resolution are added to the expansion. For a given wavelet type, the more zeroes at $\omega = \pi$ possessed by the corresponding product filter, the more accurate an approximation of the signal it will provide. Specifically, if the filter has *p* zeroes at $\omega = \pi$, then *p* will be the degree of the first polynomial that gives an error in the approximation of the signal[†](Strang and Nguyen 1997).

Example: p = 2

If we choose $p = 2$ in (47), the resulting maxflat product filter turns out to be

$$P_0(z) = (1 + z^{-1})^4(-1 + 4z^{-1} - z^{-2}) = \frac{1}{16}(-1 + 9z^{-2} + 16z^{-3} + 9z^{-4} - z^{-6}) \quad \textbf{(48)}$$

Notice how this satisfies (46). There are many possible factorizations of P_0 in (48). There are a total of six roots. Four are at z = -1, and the two from $Q(z)$ are at $c = 2 - \sqrt{3}$ and $1/c = 2 + \sqrt{3}$. A few possible choices for F_0 (or H_0) are

---

[†] Locally, smooth functions all look like polynomials, which is the basis of the Taylor series and calculus.

- $(1 + z^{-1})$
- $(1 + z^{-1})^2$
- $(1 + z^{-1})(c - z^{-1})$
- $(1 + z^{-1})^3$
- $(1 + z^{-1})^2(c - z^{-1})$

Choosing the second option, for example, results in a *biorthogonal 5/3* filter, denoting a 5 coefficient analysis filter and a 3 coefficient synthesis filter. In that case, $F_0(z) = \frac{1}{2}(1 + 2z^{-1} + z^{-2})$ and $H_0(z) = \frac{1}{8}(-1 + 2z^{-1} + 6z^{-2} + 2z^{-3} - z^{-4})$. It turns out that the scaling function associated with F_0 is the *hat* function†.

Choosing the last option on the list results in *orthogonal 4/4* filters, and this balanced spectral factorization of the halfband product filter results in an orthogonal filter bank and wavelets (Strang and Nguyen 1997). From the matrix perspective, this is equivalent to the synthesis filters simply being the transpose of the analysis filters, as stated earlier in *Section 3.2*.

Fig.*3-9* shows some of the characteristics of the lowpass and highpass analysis filters associated with this orthogonal Daubechies design (the *db2* wavelet filters).

---

†. The *hat function* is a piecewise linear function shaped like a triangle, where $\phi(0) = 0$, $\phi(1) = 1$, and $\phi(2) = 0$.

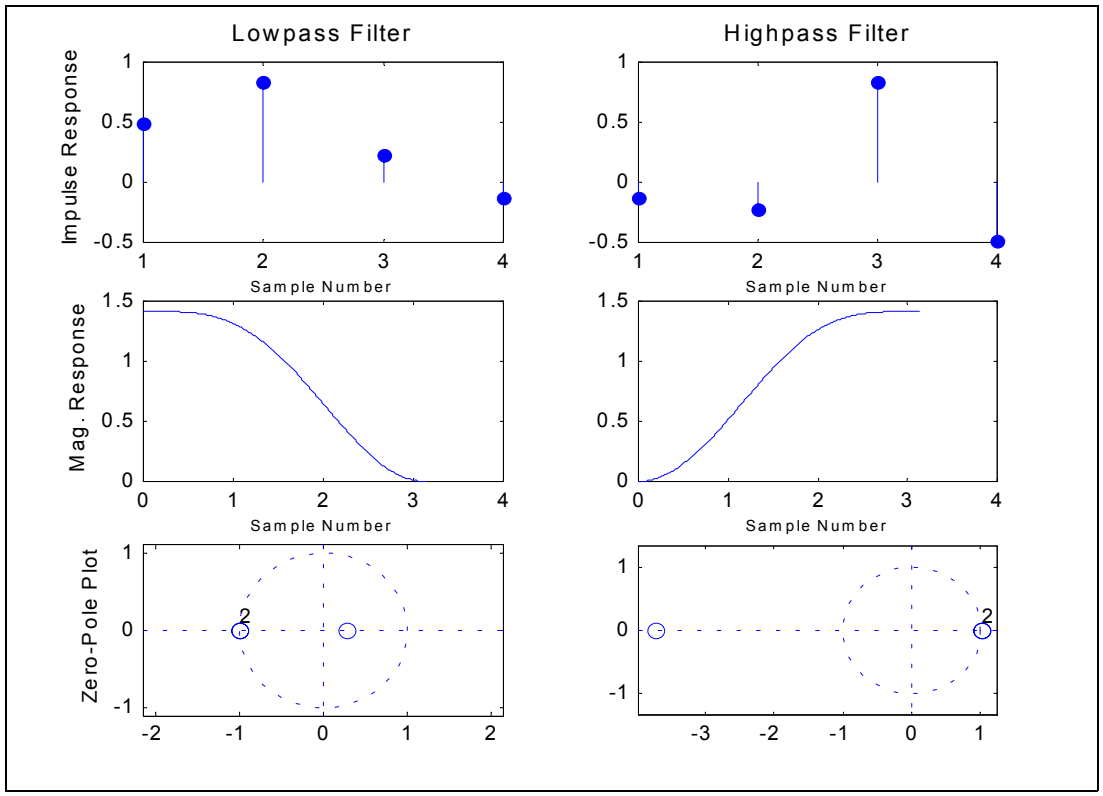

Figure 3-9: db2 Daubechies Orthogonal Filter Characteristics (p = 2)

Example: p = 1

If $p = 1$ is chosen for the product filter,

$$P_0(z) = \frac{1}{2}(1 + z^{-1})^2 \tag{49}$$

With a balanced spectral factorization, this produces the *Haar* filter bank. Specifically,

$$H_0(z) = F_0(z) = (1 + z^{-1})/(\sqrt{2}) \tag{50}$$

and, from the alias cancellation choice (43), the highpass filters are

$$H_1(z) = -F_1(z) = (1 - z^{-1})/(\sqrt{2}) \tag{51}$$

3.4 Aliasing Considerations

We now discuss aliasing issues that arise due to the downsampling of subbands after filtering. One issue concerns the aliasing of energy from one subband into another due to the non-brick wall nature of the lowpass and highpass filters used. While our filters are designed for alias cancellation, if any transform domain data is eliminated or altered prior to resynthesis, then complete alias cancellation will not occur. We describe this next in Section *3.4.1 Aliasing From Altered Data*. The other issue concerns the correspondence between the leaf or subband positions in the WPT tree versus the frequencies associated with each of those positions, and we describe that in Section *3.4.2 Aliasing of Subbands*.

3.4.1 Aliasing From Altered Data

In Section *3.3 Filter Design*, we showed how filter banks can be explicitly designed to cancel all alias terms created due to the combination of imperfect filtering and downsampling. However, this naturally assumes that the data in the transform domain is not altered in any way. If it is, such as when we modify the amplitudes of some transform coefficients, or when we only resynthesize a subset of the subbands, then some aliasing will inevitably occur.

How then do we best deal with this, given that in our work we do generally want to alter the data? The solution is to choose filters that have good isolation between bands. While we cannot achieve *brick wall* characteristics, we can achieve a high enough degree of isolation that the audible effects of aliasing are relatively minor for most sounds. For example, the worst filter we could choose from an isolation perspective is the *Haar* based filter bank that has served as an example for many of the concepts in this chapter, and the *db2* filter shown in Fig.*3-9* is not much better. Compare the magnitude response of the *db2* filter with that of a much longer orthogonal Daubechies filter, the *db15* (p=15, or 15 zeros at π), as shown below in Fig.*3-10*.

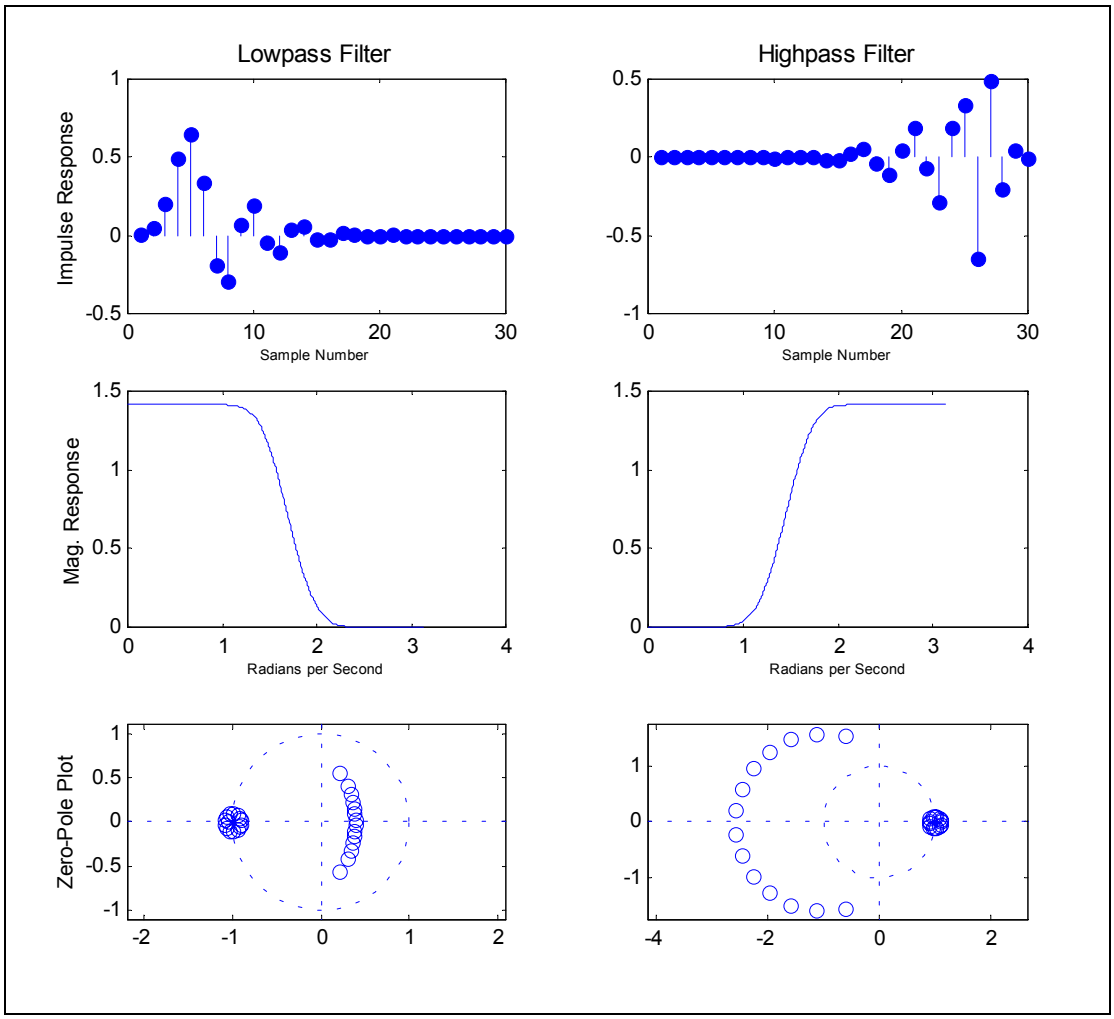

Figure 3-10: db15 Daubechies Orthogonal Filter Characteristics (p = 15)

Fig.*3-11* shows the result of aliasing on the resynthesized subbands when a short filter with poor isolation is used. In this case the *Sweep* signal was transformed with the *db2* filter. The *Sweep* signal is simply a narrow band tone that linearly increases in frequency over a four second period from 440 Hz to 4 kHz (see *Table 5.2*). Without aliasing, we would expect to see only the prominent band of energy stretching between the bottom left and top right corners of the plot in Fig.*3-11*. However, we also see energy emanating perpendicular to the main sweep signal at frequencies of 1 kHz, 1.5 kHz, 2 kHz, 2.5 kHz, 3 kHz, and 3.5 kHz. In particular, since *Sweep* has a sample rate of 8 kHz, the half way point on the frequency scale corresponds to 2 kHz (*fs/4*), which occurs at the boundary of nodes (4,4) and (4,12). It is at this frequency

that the most prominent alias terms emanate from, and it corresponds to the first level of filtering and downsampling in the transform.

We can explain how the aliased energy comes about as follows. The transform that produced the plot of Fig.*3-11* is the result of a series of successive two-band filterings with downsampling. We can visualize this splitting by thinking of the tree diagram associated with the transform (such as Fig.*3-6* with an added layer of filtering at the bottom to make it *depth-4*). The first split, which splits the root node of the tree, divides the signal into two bands, a lowpass and a highpass band. But the filters that split the signal do not possess ideal brickwall responses. The lowpass filter has some non-zero response in the highpass band and vice versa. When the outputs of these imperfect filters are downsampled, their respective out-of-band energies are aliased back into their allocated bandwidths. Since the downsampling that occurs at this first level results in a new *Nyquist frequency* (*fs/2)* of 2 kHz, it is around this frequency line that the aliasing, or *foldover*, occurs. Use of the *Sweep* signal makes the alias quite prominent, such that as the original signal approaches the new *Nyquist frequency*, the aliased energy does too. Since three more levels of filtering and downsampling occur for this *depth-4* transform, the same process repeats itself. For example, the lowpass output is again filtered and downsampled, producing a *Nyquist frequency* of 1 kHz, such that aliasing occurs around this frequency as well. While these and the other aliased components cancel each other out if no subbands are dropped or modified, they will not cancel completely when we make changes.

The disadvantage of choosing filters with better isolation is that they will tend to cause more blurring in the time domain. Recall the time-frequency resolution trade-off discussed in Section *2.3.2 Time-Frequency Uncertainty*. This blurring may or may not be a problem, depending on what we do with the signal in the transform domain. Ultimately, we must balance these the conflicting needs of time resolution and subband isolation in our filter choice, based on the relative importance of each. It is also possible that in some cases the aliasing effect may be desirable, adding another dimension to the modifications we make.

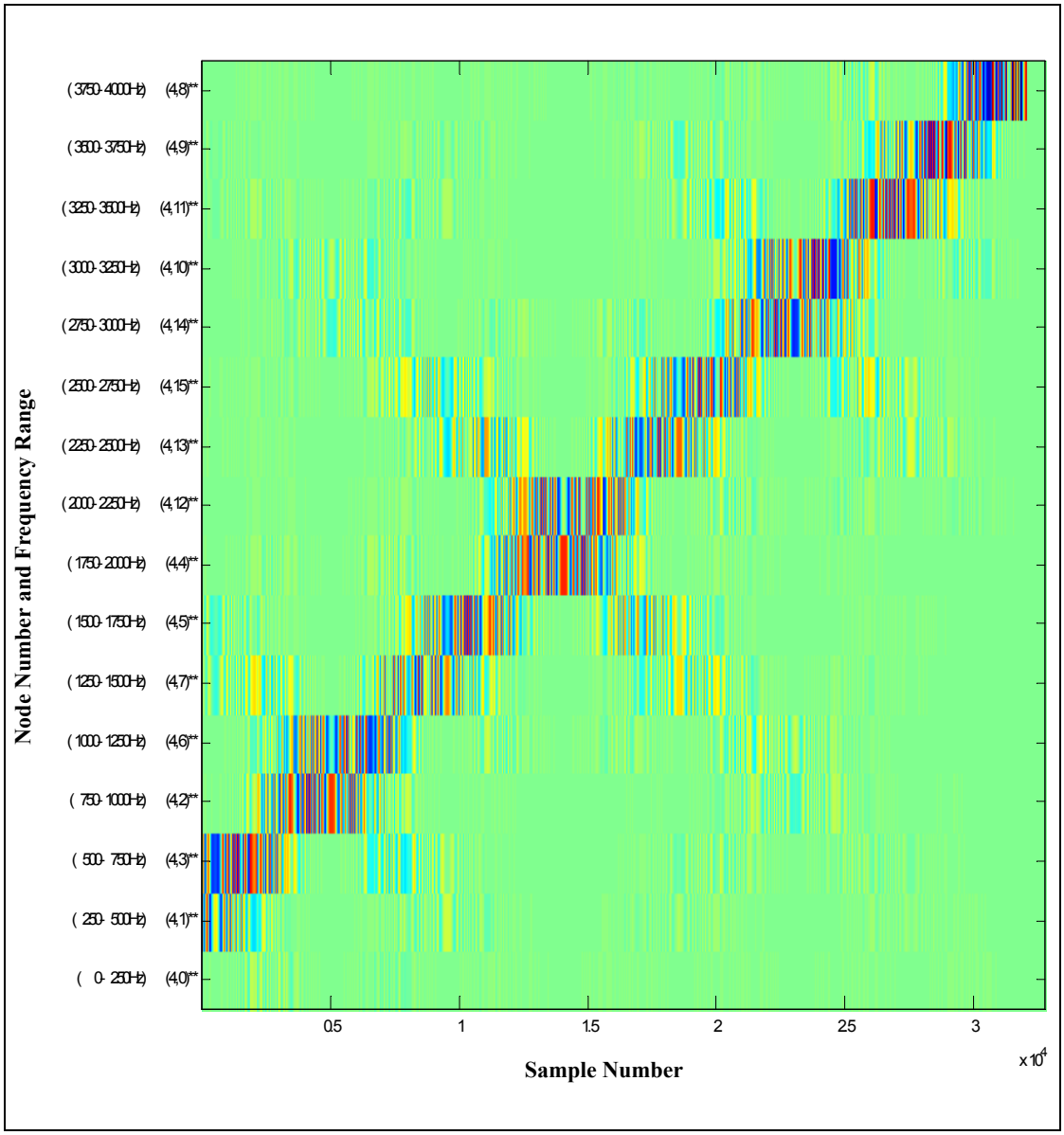

Figure 3-11: Sweep Signal WPT with db2 Filter Showing Aliasing

We should also point out that *imaging* minimization also benefits from a longer filter with better isolation. Whereas *aliasing* occurs due to downsampling, *imaging* occurs as a result of upsampling. The resynthesis or interpolation filter removes most of the imaging components after upsampling, but just like the analysis side filter, it is not perfect. And while the complete effect of aliasing and imaging is cancelled when coefficients are left unchanged, this is not the case when alterations are made, or completely new

coefficients are created. In fact, with the *granular synthesis* approach described in *Section 4.2.1.4*, we show how the use of a longer filter greatly reduces the imaging components, since in that case there is no pre-existing signal that has been transformed and from which aliased products can occur. Rather, coefficients are created from scratch, and upon upsampling imaged products are created that are to a greater or lesser extent removed by the resynthesis filter.

3.4.2 Aliasing of Subbands

We stated in *Section 3.2.4* that wavelet packets are not ordered according to frequency when we view (and number) them from left to right moving across the terminal nodes of the wavelet packet tree. We can see this by referring back to Fig.*3-7* and counting zero crossings for each wavelet packet from *w0* through *w7*. If we were to order them from least to most zero crossings, the sequence would be:

w0, *w1, w3*, *w2, w6*, *w7, w5*, *w4.*

While the lack of frequency ordering is not a problem, it needs to be understood so that subbands can be arranged in an order that allows for an interpretation of frequency versus position. Such a *frequency ordering* is useful since it helps the user relate a given subband to its approximate frequency range based on the subband's relative location.

First we explain what is happening, and why we consider the process a form of aliasing. For the sake of explanation, imagine that our signal possesses frequencies right up to the *Nyquist frequency* (i.e., π rads/second), and that the filters possess ideal brick wall responses. Recall that along with each filtering operation there occurs a downsampling by a factor of two. The first level of filtering results in a lowpass filtering of the signal, and a highpass filtering. The passband of the lowpass filter extends from $0 - \pi/2$. Downsampling by two reduces the sample frequency from π down to $\pi/2$, making it coincide with the upper band edge of the lowpass filter response. As such, no aliasing has occurred. This node is *(1,0)* in the tree, corresponding to wavelet packet w_0.

The passband of the highpass filter extends from $\pi/2 - \pi$. However, as just stated, the downsampling by two moves the sample frequency to $\pi/2$, making it in this case coincide with the lower band edge of the highpass filter response. This downsampling results in aliasing of the energy to baseband. The corresponding node is *(1,1)* in the tree, corresponding to wavelet packet w_1.

At the second level of filtering, we apply the lowpass and highpass filters to the coefficients of *(1,0)* and

(1,1) from the first level of filtering just described. Lowpass filtering *(1,0)* results in node *(2,0).* The sample frequency is now $\pi/4$, which corresponds to the new upper band edge, so again no aliasing occurs. Highpass filtering *(1,0)* results in node *(2,1).* The sample frequency of $\pi/4$ corresponds to the new lower band edge, so the energy in this band is aliased.

Now we filter node (1,1) which was aliased after the first level filtering and downsampling. As such, the lowpass filter output, corresponding to node (2,2) and wavelet packet w_2, contains higher frequency content than the output of the highpass filter corresponding to node at (2,3) and w_3.

So far we have accounted for the frequency ordering of the first four wavelets. That ordering from lowest to highest frequency is: *w0*, *w1, w3*, *w2.* This process continues down to deeper levels in the tree. Whenever highpass filtering with downsampling occurs energy is aliased and consequently the outputs of the lowpass and highpass filters that filter this aliased signal will have their frequencies reversed. The lowpass filter will correspond to a higher frequency portion of the spectrum than the highpass filter.This aliasing of basis function locations does not occur with the WT since only lowpass filter outputs are further iterated.

Wickerhauser (1991, 1994) provides a formal description of this phenomenon. The *natural* ordering is also referred to as the *Paley* ordering, while the frequency ordering described here is also referred to as a *sequency* based ordering. He proves that this frequency based ordering can be obtained via an *inverse Gray code* mapping.

In our time-frequency plots, we generally order the subbands from lowest frequency to highest, following this frequency ordering. The tree diagrams, on the other hand, follow the natural ordering from left to right, preserving the convention that a left downward line emanating from a node represents a lowpass filter operation, and a right downward line emanating from a node represents a highpass filter operation.

Table 3.1 provides a cross reference between node numbers and the main frequency range that each node corresponds to for WPTs up to *depth-4*. This cross referencing is done for several sample rates. The *Depth 3* column, for example, shows the frequency ordered sequence we described earlier in this section.

.

Table 3.1 Node Name vs. Frequency Range

Sample Rate (kHz)						Node Name				
44.1	**32**	**22.050**	**16**	**11.025**	**8**	**Depth 4**	**Depth 3**	**Depth 2**	**Depth 1**	**Depth 0**
0.000 - 1.3781	0.000 - 1.000	0.000 - 0.6891	0.000 - 0.500	0.000 - 0.3445	0.000 - 0.250	4,0	3,0	2,0	1,0	0,0
1.3781- 2.7563	1.000 - 2.000	0.6891 - 1.3781	0.500 - 1.000	0.3445 - 0.6891	0.250 - 0.500	4,1				
2.7563 - 4.1344	2.000 - 3.000	1.3781- 2.0672	1.000 - 1.500	0.6891 - 1.0336	0.500 - 0.750	4,3	3,1			
4.1344 - 5.5125	3.000 - 4.000	2.0672 - 2.7563	1.500 - 2.000	1.0336 - 1.3781	0.750 - 1.000	4,2				
5.5125 - 6.8906	4.000 - 5.000	2.7563 - 3.4453	2.000 - 2.500	1.3781 - 1.7227	1.000 - 1.250	4,6	3,3	2,1		
6.8906 - 8.2688	5.000 - 6.000	3.4453 - 4.1344	2.500 - 3.000	1.7227 - 2.0672	1.250 - 1.500	4,7				
8.2688 - 9.6469	6.000 - 7.000	4.1344 - 4.8234	3.000 - 3.500	2.0672 - 2.4117	1.500 - 1.750	4,5	3,2			
9.6469 - 11.025	7.000 - 8.000	4.8234 - 5.5125	3.500 - 4.000	2.4117 - 2.7563	1.750 - 2.000	4,4				
11.025 - 12.403	8.000 - 9.000	5.5125 - 6.2016	4.000 - 4.500	2.7563 - 3.1008	2.000 - 2.250	4,12	3,6	2,3	1,1	
12.403 - 13.781	9.000 - 10.000	6.2016 - 6.8906	4.500 - 5.000	3.1008 - 3.4453	2.250 - 2.500	4,13				
13.781 - 15.159	10.000 - 11.000	6,890.6 -7,579.7	5.000 - 5.500	3.4453 - 3.7898	2.500 - 2.750	4,15	3,7			
15.159 - 16.538	11.000 - 12.000	7.5797 - 8.2688	5.500 - 6.000	3.7898 - 4.1344	2.750 - 3.000	4,14				
16.538 - 17.916	12.000 - 13.000	8.2688 - 8,957.8	6.000 - 6.500	4.1344 - 4.4789	3.000 - 3.250	4,10	3,5	2,2		
17.916 - 19.294	13.000 - 14.000	8.9578 - 9.6469	6.500 - 7.000	4.4789 - 4.8234	3.250 - 3.500	4,11				
19.294 - 20.672	14.000 - 15.000	9.6469 - 10.336	7.000 - 7.500	4.8234 - 5.1680	3.500 - 3.750	4,9	3,4			
20.672 - 22.050	15.000 - 16.000	10.336 - 11.025	7.500 - 8.000	5.1680 - 5.5125	3.750 - 4.000	4,8				

3.5 Boundary Extension

Unlike images, audio signals are generally very long. Therefore the issue of distortion arising from having no data before the start of the signal and after its termination is relatively minor. A three minute audio clip at 44.1 kHz is comprised of close to eight million samples. Even a long impulse response filter would not cause any significant portion of the signal to become distorted due to boundary effects. However, the proportion of the signal influenced by the boundary is larger if we use fairly short audio clips, which we do in much of our work. While *perfect reconstruction* exists regardless of the extension mode used, again this is only the case when we do not modify coefficients. Perhaps more importantly, our choice of boundary extension method can affect what the transform coefficients look like at the boundaries, and therefore it can affect how we might interpret and/or modify these coefficients. Furthermore, our choice may affect what happens if we splice pieces together in composition.

At least three possibilities exist: *zero-padding*, *periodic extension*, and *symmetric extension*. With *zero-padding,* zeros are assumed to exist before the start and after the end of the signal. In *periodic extension*, the signal is assumed periodic, and repeated in that fashion. In *symmetric extension*, the signal is mirrored about the end points.

Strang and Nguyen (1997) treat this topic in depth, and find that for images, *symmetric extension* tends to provide the best *mean squared error* (MSE*)* and *peak signal-to-noise ratio* (PSNR); it is only very marginally worse than *periodic extension* in *maximum* error. *Zero-padding* performs the poorest by far. As described later in *Section 5.2*, we will choose to use *symmetric extension* for most of our transforms.

3.6 Fixed Bases

If sufficient justification can be found for a particular fixed basis for our signal processing needs, then our task of pruning the original wavelet packet tree becomes simple. Such justification might be related to the nature of the signal itself, to the nature of the listener, or both[†]. In the following, we discuss two possibly useful bases derived from the *complete tree* of the WPT.

[†] In *lossy audio compression* for example, the nature of both the signal and the listener are exploited. Redundancy in the representation of the signal is reduced towards the true entropy of the signal, and, to achieve greater compression the nature of the human auditory system is taken into account and sounds that are deemed *irrelevant* to the listener are eliminated (such as sounds that the listener would not hear due to the presence of other sounds or noise).

Wavelet Transform Basis

We have already discussed the WT to some extent in previous sections. Here we focus on the structure of the WT basis and how time-frequency resolution is partitioned.

The wavelet basis is the most popular example of a fixed basis derived from the WPT. In fact the wavelet transform (WT) preceded the wavelet packet transform, and the latter may be viewed as a generalization of the former. The tree diagram of a *depth-4* wavelet basis was given earlier in Fig.*3-4*.

A key feature of the WT is that the bands scale in time and frequency resolution logarithmically, just as does the human auditory system's perception of pitch and critical band characteristic (*Section 2.3.1*). So in a general sense the WT is well suited to the human auditory system. However, the filter bank based DWT is *dyadically* spaced[†]. Moving up from the base of the WT tree, each higher level possesses one half the frequency resolution of the previous level, and the entire upper half of the spectrum is represented by just a single band. Fig.*3-12* shows the relative bandwidths associated with a *depth-4* wavelet transform. For a sample rate of 44,100 Hz, the upper subband would be 11,025 Hz wide. Imagine wanting to separately manipulate several narrowband sound components all in the 11,025 to 22,050 Hz frequency range. The dyadic DWT would lump all these components together, and working with them individually would not be possible.

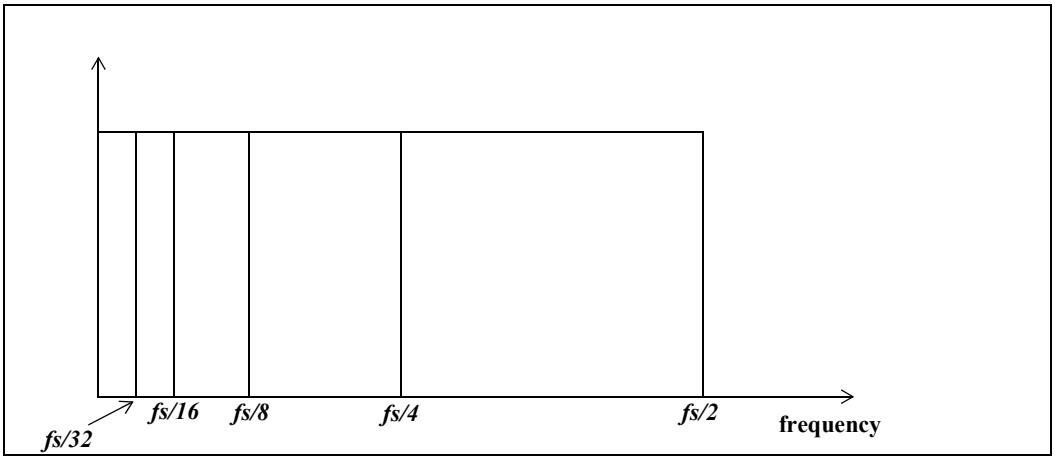

Figure 3-12: Relative Bandwidths for a Depth-4 DWT

[†] Other spacings are possible with the *continuous wavelet transform* (CWT), and the DWT in general, however the dyadic DWT used in the filter bank implementation relies on this *power-of-two* relationship for much of its efficiency. An analogous efficiency advantage is realized in the dyadic DWPT.

Also, while the frequency resolution is generally far too poor, the time resolution for most applications is much greater than required. Since time and frequency resolution can be traded-off against each other, further subdividing the upper bands ought to make sense. Recall that the trade-off between time and frequency resolution is formalized in the uncertainty principle given in (2) of *Section 2.3.2*.

At the relatively low sample rate of 16 kHz, where the upper subband bandwidth is 4 kHz wide, the time resolution would be lower bounded by

$$\Delta t \geq 1 / (4,000 \times 4\pi) \geq 20 \ \mu\text{seconds}$$

Whether the application is sound modification, synthesis, or analysis in a general sense, such high resolution is not required. To get a sense of how short this is, consider a relatively fast piece of music at 200 *bpm* and *4/4* time. Each quarter note lasts $60 / 200 = 0.3$ seconds. A 32^{nd} note would be $1/8^{th}$ as long, or 0.0375 seconds, which is still over 1,800 times longer than the lower bounded time resolution. Such a 32^{nd} note can occur in this frequency range, such as the 4^{th} octave above *A 440* at *7,040 Hz*, which is near the upper edge of the upper band.

Other Fixed Bases

Many other fixed basis solutions are possible. One other would appear especially relevant. To align the basis more closely with psychoacoustic realities, the subband bandwidths could be based on the *critical band* bandwidths. One such approach would involve use of *third-octave* spacing, as in a *third-octave graphic equalizer*. By approximating these bandwidths, such a fixed basis could comprise the front-end filter bank of an equalizer function.

Why is division of the spectrum into *third-octave* and/or *critical bands* useful? Imagine instead dividing the spectrum into narrow bands of equal bandwidth from *0* through *fs/2* rad/second, and providing an equalization control for each band. At the high end of the spectrum, there are several bands per critical band. Yet the human auditory system cannot resolve the various frequency components of these multiple bands within the critical band, since the critical bandwidth represents the frequency selectivity of the human ear. These components are fused together. Therefore there is little or no value in being able to manipulate each of these narrow bands individually. At the low end of the spectrum, unless the fixed bands are extremely narrow, there will be multiple critical bands per subband. In that case, the equalization control will be too coarse. So a filter bank that divides the spectrum into critical bands, or something close to that such as third-octaves, would seem to make the most sense, at least for applications like equalization. Later, we define the

implementation of something we call a *pseudo-third-octave basis* (PTOB) to address this need.

The fixed basis approaches just discussed are motivated by how we hear. With the best basis approaches to be described next, other factors drive the design. These include meeting a given bit rate, attaining a low number of subbands, or learning something about the sound itself through the basis that is chosen. This change in focus from the receptor (the ear) to the source parallels the two strategies in audio compression of irrelevancy removal versus redundancy removal, respectively.

3.7 Best Bases

While fixed bases are useful in some situations, the fact that they are fixed makes them inflexible. A fixed basis does not adapt in any way to the signal, nor does it respond to the wishes of the user. On the other hand, manual selection of a basis that is optimal in some sense is generally not possible given the prohibitively large space of potential solutions. As we discuss later in Section *4.5 Complexity*, there are roughly 2^N possible bases associated with a transform of block length N. For example a 16 sample block, which incidentally can support a maximum *depth-4* transform, would have over 65,000 possible bases derivable from the initial WPT basis. Furthermore, if the *best* basis from such a manual approach were to be determined by subjective user assessment, such as via a listening test or visual assessment of the signal projected onto the basis, remembering the subjectively assessed quality of each basis to the extent that effective comparisons can be made across all bases would be even more difficult. All these factors motivate seeking an automated approach to basis selection.

Strang and Nguyen (1997) state that, "We need a reasonable algorithm to choose the nearly best M atoms for a given $f(t)$, out of a dictionary of P atoms. A single basis may be too inflexible, and *the algorithm must adapt to the signal*". The so-called atoms are in our case the basis functions associated with all the nodes of the WPT's complete tree[†]. Our goal then is to derive an algorithm that finds a basis that is *best* in some sense for the signal we are working with. Doing so involves defining what is meant by *best*, finding a *cost function* that is directly related to this definition of "bestness", and then deriving an algorithm that will determine the *best* solution by minimizing this cost function.

There has been considerable activity in the area of best basis selection, though some applies more to the selection of a basis from a set of *frames* that are *over-complete* (frames are not a linearly independent set of

[†] The basis functions are in turn the wavelet filter impulse responses.

basis vectors). Since we start with an orthonormal basis to begin with, the goal is to find which basis is the best from a set of possible bases[†]. The fact that they are all bases means that, by definition, they span the space of the signal, and are, unlike frames, linearly independent. Of particular interest to us are the work of Coifman and Wickerhauser (1992), and Ramchandran and Vetterli (1993). The latter may be viewed as an adaptation of the former, optimizing the basis for rate-distortion rather than for entropy (Strang and Nguyen 1997).

An important requirement of any best basis approach we consider is that it should lend itself to an efficient algorithm. Already, by virtue of using the WPT, we have a fairly efficient transform that produces the full wavelet packet tree in $O(N \log N)$ operations (see *Section 4.5*). We would like any best basis approach we adopt to not add a great deal of processing to this. Use of the orthonormal bases associated with the WPT means that energy is conserved in the transform. This orthogonality allows us to define an additive cost functional J, such that $J = \sum_i J_i$, where the J_i are the costs of the individual nodes of the candidate pruned tree. We then can find the pruned tree with the lowest cost in an efficient way, considering the cost of any given branch of the tree in isolation of other branches. We should also note that, while the *biorthogonal* bases associated with the WPT are not strictly orthogonal, many are close enough for practical purposes to also be used with the basis selection algorithms we develop.

We will see later that the cost functional can be defined to allow for either *lossless* or *lossy* basis selection. In the former case, the cost might relate to the required amount of data required to store or transmit the signal. In data compression terminology, this entity is referred to as the rate R. Information theory tells us the rate can be no lower than the *entropy* of the signal. In the latter case, the cost might include both the amount of data required to store or transmit the signal and the distortion added to the signal in reducing the data. Because of the orthonormal basis functions at each step in the tree, both the rate and distortion of any two branches emanating from the same (parent) node are additive. They are also additive across multiple blocks of transforms for the same node.

To summarize the general approach, first the WPT is performed specifically with orthonormal basis functions, thereby allowing the definition of additive cost functionals. The complete tree can then be

---

[†] In the case of the *BDBBA* that we describe later (*Section 4.4.2*), transform coefficients associated with some basis functions might be quantized or deleted. In that case, we can still say that the basis itself is complete and spans the space of the original signal, but that some of those basis functions are not used due to the deletion of coefficients.

efficiently *pruned* such that the resulting tree corresponds to a minimum of the cost functional across all possible trees. This search for the minimum cost pruned tree would take very long if no structure existed. However, since each node with its children can be treated individually, efficient basis selection is possible. Both the work of Coifman and Wickerhauser (1992), and that of Ramchandran and Vetterli (1993) use this general approach, and are discussed next. While these approaches are primarily motivated by the needs of data compression, we will find that they act as good starting points for our own purposes too.

3.7.1 Entropy Best Basis Algorithm (EBBA)

Here we introduce an approach to basis selection developed by Coifman and Wickerhauser (1992). Since it involves minimization of entropy in the transform coefficients, we refer to it for convenience as the *Entropy Best Basis Algorithm*, or EBBA.

Any scheme we consider should lend itself to an efficient algorithm, and the EBBA does. The approach is a *lossless* form of basis selection in that no coefficients are discarded. Coifman and Wickerhauser's requirement for a functional was that it "describe concentration or the number of coefficients required to accurately describe the sequence." Further, they say that the cost should "be large when the coefficients are roughly the same size and small when all but a few coefficients are negligible".

The additive cost functional they use is the Shannon entropy function, but applied to the square of the transform coefficient values, rather than probabilities of values as in the case of a true entropy measure. The cost can be expressed as:

$$J = -\sum_i \|v_i\|^2 \ln \|v_i\|^2 \qquad \textbf{(52)}$$

where $v = \sum_i v_i$ is the input signal and the v_i are the orthonormal components of v resulting from the WPT transform. $\|v\| = 1$.

The algorithm uses this cost functional to find a *best basis*, which is the basis that corresponds to the lowest value of this cost functional. Since we assume an orthogonal WPT, the cost function is additive and the nodes can be treated individually. The algorithm proceeds by starting at the nodes that reside one level up from the terminal nodes of the complete tree, and comparing the cost of each of those nodes with the sums of the costs of the two child nodes emanating from each of them. If the cost of the parent is less, then it is

retained and the child nodes are pruned. Otherwise, the child nodes are kept, and the cost of the parent is replaced with the sum of the child costs. Then the nodes one level up are subjected to the same procedure, and so on. Finally the root node is checked and the best basis is known.

As we will see later, while this approach is not entirely without merit for our needs, it does suffer from some shortcomings. Primarily, when we are finding an averaged best basis over a signal comprised of multiple blocks, the entropy cost functional will not discriminate between the case when the *all but few negligible coefficients* are scattered sparsely over time across many subbands, versus when they are all more densely packed over time in very few subbands[†]. Both cases could correspond to the same cost, yet we will seek a functional that favours the latter.

Also, this approach does not afford any user control over the process. There are no parameters to set and in some way guide the process. Modifications are however possible, such as the replacement of *Shannon Entropy* with something called *Threshold Entropy* (Misiti, Misiti, Oppenheim and Poggi 1997).

3.7.2 Rate-Distortion Best Basis Algorithm (RDBBA)

We now describe an approach by Ramchandran and Vetterli (1993) that finds an optimal basis in a *Rate-Distortion* sense. We refer to this for convenience as the *Rate-Distortion Best Basis Algorithm* (RDBBA). Shannon (1959) laid out the fundamentals of *source coding* theory, showing the relationship between the minimum possible *distortion* for a given channel *rate* constraint and conversely, the minimum *rate* for a given *distortion* constraint. The RDBBA uses this framework in the context of a finite set of possible quantizers and the distortions that result from their use.

Overview

The RDBBA can be viewed as a modification to the EBBA just described; albeit a significant one. Since the RDBBA is the starting point for our own *Bi-rate Distortion Best Basis Algorithm* (BDBBA), we describe it here in some detail to provide a solid foundation for our later description of the BDBBA (*Section 4.4.2*).

The RDBBA lends itself to an efficient implementation, as did the EBBA. The orthonormality of the WPT allows use of a fast *pruning* algorithm by enabling both the distortion and rate of the terminal nodes of

---

[†] The EBBA does find an average best basis over multiple blocks.They do not speak about multiple M blocks of length N. They do claim to "build out of the library functions an orthonormal basis relative to which the given signal or collection of signals has the lowest information cost". As such, they seem to seek an average single best basis over the duration of the signal (though nothing in the approach precludes finding a new basis for each N samples).

any pruned subtree to be additive. Implicit in any *rate-distortion* based approach is the function of quantization, which in turn implies the scheme is *lossy.* Such allowance for signal loss is a significant departure from the EBBA, but one that gives it a good deal more flexibility and compression capability. Note that the quantization can also be used as a mechanism to finding a basis, after which time the coefficients can be restored to their full precision values.

The RDBBA is motivated by the application of *data compression*. While Ramchandran and Vetterli (1993) state that their algorithm can be applied to '*all classes of structures which permit the construction of a hierarchy of basis covers for the input signal space*', their paper deals largely with orthonormal *wavelet packet* basis functions. The authors solve the problem of jointly finding a best basis and the optimal quantizers for that basis' terminal nodes in a way that minimizes an additive distortion function with respect to an l_2 norm (such as weighted or unweighted MSE). Furthermore, they do it in a way that is computationally efficient.

The application of data compression requires removal of redundancies (or correlations) within the signal. There are several prevalent approaches to this including *vector quantization* (VQ) and linear transform coding methods based on the *Discrete Cosine Transform* (DCT) or *Karhunen-Loeve Transform* (KLT). VQ is computationally expensive but allows for adaptation to non-stationarities in the signal, whereas a linear transform coding approach is fast but does not adapt. Ramchandran and Vetterli's desire was to combine the potential adaptability of VQ with the speed of linear transform coding. Wavelet packets allow for this to occur[†].

The approach may be summarized as follows. First WPTs of *depth-d* ($d = \log_2 N$) are performed on the M blocks of length N of the signal[††]. For each node in each of the M trees a rate versus distortion profile is calculated based on a pre-determined set of available quantizers. The range of quantizers in this set would be chosen such that the minimum and maximum distortion and rate requirements of the compression scheme can be achieved, while the number of quantizers in the set would be determined by balancing the higher

---

[†] We note that our application is not *compression*, but rather *signal characterization* with efficient and effective opportunities for manipulation. In many cases we would like a stable basis over the duration of the signal, averaging over any non-stationarities. But we do value adaptability in that the basis is chosen based on the particular signal. The basis will, in some averaged sense, be optimal for that signal, giving us additional information over that obtained from a fixed basis, and permitting modification of the signal using a basis that is in some sense aligned to the nature of that signal.

[††] For example, a signal comprised of 65,536 samples (2^{16}) that is transformed to *depth-5* will have blocks of length 32 ($5 = \log_2 32$), and there will be 2,048 ($2^{16}/32$) of these blocks.

processing requirements of a large set with the finer resolution such a large set would provide. Completely arbitrary quantizers and coding schemes are assumed, such as for example the use of *vector quantization* (VQ) and *entropy coding*. Next, an efficient search is performed that simultaneously finds a basis and the requisite quantizer for each terminal node in the basis that is optimal in a distortion sense that also meets a specified *rate budget*. This so-called efficient search employs a *Lagrangian* cost function along with the method of *bisection* to arrive at the solution. We now provide a detailed description.

Detailed Description

The constrained problem of finding the basis that minimizes average distortion for a given rate budget R_{budget} can be converted to an unconstrained problem through the introduction of a *Lagrange multiplier* λ. The cost function to be minimized for a given λ is

$$J(\lambda) = D + \lambda R \tag{53}$$

where D and R are the total distortion and rate respectively across all terminal nodes of the basis tree.

It can be shown (Everett 1963, Shoham and Gersho 1988) that for any $\lambda \geq 0$ the solution to the unconstrained problem (53) is also the solution to the original constrained problem with the constraint that R_{budget} is equal to the R in (53), corresponding to the minimum $J(\lambda)$. In other words, we need to solve (53) with a value of $\lambda^* \geq 0$ such that R meets the rate budget. Later we will describe the bisection algorithm that finds the solution λ^*.

Before any solution search can begin, several one-time operations must be performed. First the WPT to sufficient depth must be performed. Then for every node of the complete tree, whether internal or terminal, a rate versus distortion (RD) profile must be created using the transform coefficients and set of available quantizers.

Creation of the RD profile requires that a set of one or more quantizers for each node in the full tree be defined. In the so-called *toy example* given in Ramchandran and Vetterli (1993), *coarse*, *medium* and *fine* resolution uniform quantizers are available at each node, using 4, 6, and 8 bits respectively. Use of three quantizers naturally results in a three-point RD profile per node. A point in the profile for a given node is calculated by quantizing the transform coefficients using the discrete levels of the given quantizer and calculating the distortion associated with this quantization by summing the square of the quantization error

across all coefficients. This distortion value then corresponds to the quantizer rate used, which must be scaled by the relative number of coefficients associated with that node's level in the tree. Given the 6 bit quantizer (64 level), for example, and a signal of length $N = 4$ samples, the rate R associated with quantization at the root node is $6 \times 4 = 24$ bits. A *depth-1* node with the same quantizer corresponds to a rate of $6 \times 2 = 12$ bits, since due to downsampling by two the *depth-1* node has half the coefficients of its parent node.

Fig.*3-13* shows the RD profile for the *depth-1* highpass filter output node of the *toy example* mentioned earlier. Notice the rate values are double the number of quantizer bits because there are two coefficients to quantize.

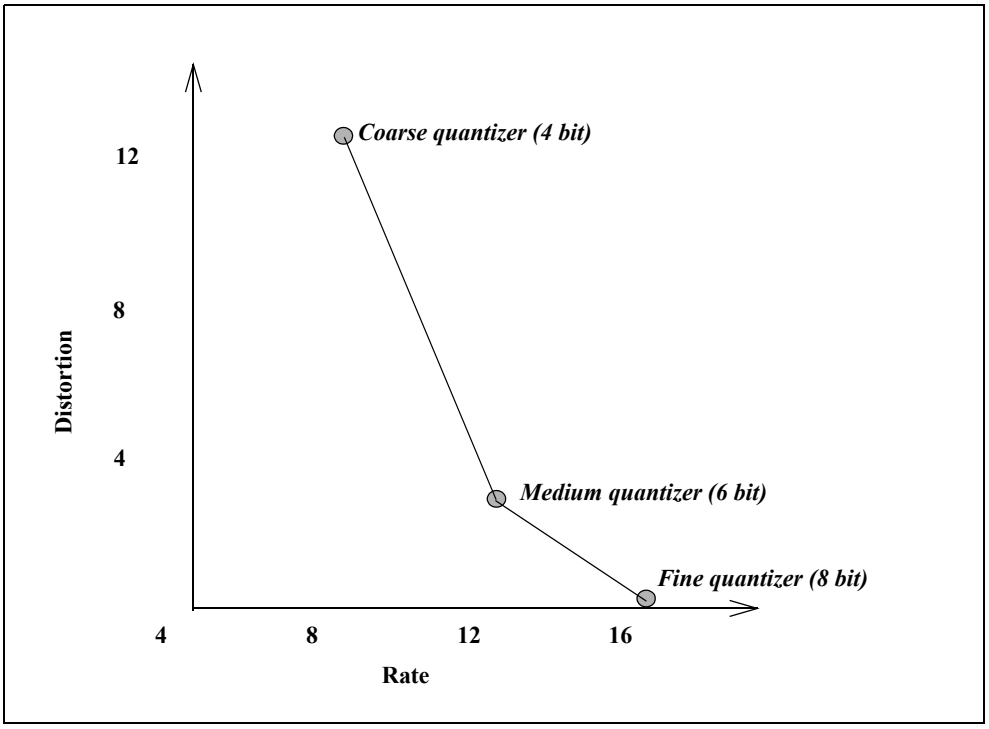

Figure 3-13: Sample Depth-1 Node RD Profile

Turning back to the search algorithm for a solution, the *biased Lagrangian cost function W* is now formed, defined as:

$$W(\lambda) = J^*(\lambda) - \lambda R_{budget} \quad \textbf{(54)}$$

where $J^*(\lambda)$ is the minimum cost for the given λ.

Ramchandran and Vetterli (1993) show in *Theorem 2* of that reference that the λ^* and the associated operating point $x^*(\lambda^*)$ (i.e., the basis along with the optimal quantizer for each leaf) that maximizes W in (54) are also optimal solutions to the unconstrained problem of minimizing (53) such that the rate budget is also satisfied. Therefore we are left with solving the following, which also takes into account multiple blocks M:

$$W(\lambda^*) = \max_{\lambda \geq 0} \left(\sum_{i=1}^{M} \left[\min_{S_i} \left[\sum_{n \in \tilde{S}_i} \min_{q} [D_q(n) + \lambda R_q(n)] \right] \right] - \lambda R_{budget} \right) \quad \textbf{(55)}$$

S_i denotes the subtree and $\tilde{S}_i$ the subtree leaves of the ith block, and $D_q(n)$ and $R_q(n)$ are the distortion and rate associated with quantizer q at the nth node. So, by finding the maximum of W as per (55), the solution is found. Since (55) splits naturally into three separate optimization operations, solving it lends itself well to algorithmic implementation. We express (55) as:

$$\max_{\lambda}^{(3)} \left(\sum_{M} \left[\min_{S_i}^{(2)} \left[\sum_{\tilde{S}_i} \min_{q}^{(1)} [D(n) + \lambda R(n)] \right] \right] - \lambda R_{budget} \right) \quad \textbf{(56)}$$

where each operation is numerically referenced and described below.

(1) Inner Minimization - This operation chooses for the given λ the lowest cost quantizer for each node of the complete tree of each block.

(2) Outer Minimization - This operation finds the best or lowest cost pruned subtree S for the given λ for each block. A fast algorithm that exploits the additive nature of the cost function is used to prune the tree.

(3) Outer Maximization - This operation determines the optimum Lagrange multiplier λ^* for R_{budget} by maximizing $W(\lambda)$. Ramchandran and Vetterli show that a fast bisection search algorithm may be used because $W(\lambda)$ is a concave function of λ.

We now elaborate on each of these steps in order to complete the description of the overall algorithm. The inner minimization of step 1, which is the minimization of equation (53), is based on the premise that the lowest cost solution requires the same value for λ across all leaves or subbands. This premise is so because

any change in allocated bits (one or more) must result in the same change in total distortion, regardless of which subband the change is applied to; if this were not the case, the total distortion could be lowered by reallocating bits across the subbands (without adding any new bits), and consequently the operating point would not be optimal. Therefore all nodes or subbands must operate at the same λ and there is justification in seeking to minimize cost for a fixed λ. Ramchandran and Vetterli express it as "The best quantizer for a particular tree node is that one which 'lives' at absolute slope λ on the convex hull of the operational R-D curve for that node.". λ is referred to as the (operating) slope because $-\lambda$ is the negative of the slope of the RD curve. This relationship is readily seen by taking the partial derivative of D with respect to R in (53).

So the inner minimization step of the algorithm, which is run for each λ, is equivalent to choosing the operating point (rate and distortion) for each node that corresponds to the point of the RD curve where the slope is $-\lambda$. This is accomplished by calculating the cost $J = D + \lambda R$ for each quantizer rate available, and choosing the minimum.

The outer minimization in step 2 is crucial to the speed of the algorithm, and is the same process used in the EBBA. Starting at the parent nodes of the terminal nodes of the complete tree and working up to and including the root node, the cost associated with each node (which is the minimum cost for that node if it were to become a terminal node of the final tree) is compared to the sum of the minimum costs of the subtrees rooted at each of the child nodes of that parent. If the cost of the parent is less, then the subtrees associated with each of the child nodes are pruned back to the parent. Otherwise, the node is designated as a split node and its child nodes and any subtrees that survived pruning that emanate from them are kept. In either case, however, the parent node may well be later pruned as the optimization proceeds up towards the tree root. This pruning operation as a whole is complete when it has moved up the tree to include the root node, and the corresponding pruned subtree is optimal in a minimum cost sense for the given λ. As described in *Section 4.4.2.2*, this pruning operation is relatively fast†.

The algorithm described thus far in steps 1 and 2 will find the lowest cost basis for a given operating slope λ, and is referred to in the reference as the *Phase I* subroutine. Now an approach is required that

---

†. The subtree could wind up being the complete tree associated with the original WPT, the root node only, or any of the other subtrees that can possibly be derived from the complete tree. Since there are roughly 2^N possible trees for a length N block, with several potential quantizers to choose from at each node, finding the lowest cost subtree without such a fast algorithm would be very slow by comparison.

submits λ 's to this subroutine in a way that rapidly converges to a solution. This solution will be such that the rate associated with it will be as close to the rate budget as possible without exceeding it. The maximization of step 3 is derived by taking the partial derivative of W with respect to λ in (55) and setting it to zero. In other words, the zero crossing of $\frac{\partial W}{\partial \lambda}$ corresponds to the solution. This results in

$$\frac{\partial W}{\partial \lambda} = \sum_i R_i^*(\lambda) - R_{budget} \quad \textbf{(57)}$$

where $R_i^*(\lambda)$ is the rate associated with the optimal subtree for block i as determined by steps 1 and 2. For $\frac{\partial W}{\partial \lambda} = 0$, $\sum_i R_i^*(\lambda)$ should approach R_{budget}. When $\lambda < \lambda^*$, $\sum_i R_i^*(\lambda) > R_{budget}$, since the smaller that λ is, the less costly a higher rate will be. Conversely, when $\lambda > \lambda^*$, $\sum_i R_i^*(\lambda) \le R_{budget}$.

So the fast bisection proceeds as follows. First pick upper and lower bounds on the value of λ^*, called λ_u and λ_l respectively. These should satisfy

$$\sum_i R_i^*(\lambda_u) \le R_{budget} \le \sum_i R_i^*(\lambda_l) \quad \textbf{(58)}$$

and if equality holds on either side, the solution is found.

Defining λ_{next} as the next value to submit to the Phase I subroutine (steps 1 and 2), let λ_{next} equal the magnitude of the slope of the line connecting the operating points associated with λ_u and λ_l on the composite RD curve. Stated mathematically:

$$\lambda_{next} = \left(\left|D^*(\lambda_l) - D^*(\lambda_u)\right| / \left|R^*(\lambda_l) - R^*(\lambda_u)\right|\right) + \varepsilon \quad \textbf{(59)}$$

where ε is a vanishingly small number to ensure that the lower rate point is picked if λ_{next} is singular, since the rate budget must not be exceeded (see also *Appendix C*).

Then iteration towards convergence proceeds as follows:

If $\sum_i R_i^*(\lambda) > R_{budget}$, then $\lambda < \lambda^*$ and therefore λ_l is replaced with the λ_{next} just used.

If $\sum_i R_i^*(\lambda) < R_{budget}$, then $\lambda > \lambda^*$ and therefore λ_u is replaced with the λ_{next} just used.

Eventually this leads to convergence.

3.8 Summary

In this chapter we have reviewed the fundamentals of the *wavelet packet transform* (WPT). Our approach was to describe the *wavelet transform* (WT) first, and then extend to the WPT as required. We started by providing a historical overview of wavelets and filter banks, the connection between the two, and some of the key areas of their application. We then introduced the concept of basis selection, the various types of basis selection approaches that we might want to consider, and some of the motivations for deriving a new basis from the complete tree produced by the WPT.

Our detailed description of the WPT involved first describing the mathematical underpinnings of the WT. We focused exclusively on the discretized case, since this is necessary for a fast implementation.We showed the connection between wavelets and discrete time filters, which are linked via the so-called *dilation* and *wavelet equations*. Solving the dilation and wavelet equations to determine the wavelets associated with a given set of filters is not always straightforward. We described one algorithm that can sometimes be used to determine this solution, the *Cascade Algorithm*. From filters the connection was made to filter banks, and in particular a fast implementation called the *Fast Wavelet Transform* (FWT). The *Haar* filters and associated wavelets were used as an example to demonstrate many of the concepts. We then described the WPT by extending the concepts already introduced. The WPT differs from the WT in that not only is the lowpass filter output filtered further, but the highpass filter is as well.

Since we work exclusively with the filter bank form of the WPT, and therefore do not require the continuous time wavelet function definition, the task of designing a transform with certain characteristics is one of designing a pair of filters. We introduced the requirements of *perfect reconstruction* and *alias cancellation*, and showed how these two requirements result in a well defined design method. The Daubechies construction was provided as an example design choice that meets the requirements.

Aliasing considerations were described next. These included both the aliasing of signal content as a result

of altering the data and thereby partially defeating the alias cancellation built in to the transform, and the aliasing of subbands such that they are not ordered from lowest to highest frequency content. In this latter case there is no concern with adding aliasing distortion to the signal, but rather the concern is with the user having an understanding of the correspondence between a given subband and the approximate spectral content it contains.

After briefly describing the issues related to boundary extension, and the various approaches available to us, we then described several known basis selection approaches. First we discussed *fixed bases*, which correspond to a fixed filter bank configuration, independent of the nature of the transformed signal. Then we discussed *best basis* selection algorithms, which find a basis that is in some way 'best' for the signal at hand as well as possibly for the needs of the user. We first described the *Entropy Best Basis Algorithm* (EBBA), which chooses a basis that minimizes a cost function based on the application of the Shannon entropy function to the transform coefficient amplitudes. Finally, we described the *Rate-Distortion Best Basis Algorithm* (RDBBA), which finds a basis and corresponding set of quantizers that minimizes the distortion imparted on a signal for a certain rate budget. We described this algorithm in detail since we have used it as the foundation for our own best basis selection approach, the *Bi-Rate Distortion Best Basis Algorithm* (BDBBA).

In the next chapter we describe our application of the WPT to sound signal processing, focussing on a description of the techniques and underlying fundamentals.

Chapter 4
Wavelet Packets Applied to Sound

In this and the next chapter we describe the specifics of our own work. In the current chapter we describe the application of the WPT[†] to sound signal processing, focussing on a description of the techniques and any mathematical underpinnings. In the next chapter we provide a description of the software developed to experiment with the concepts described here, and then describe the nature and results of our experimentation.

Recall from *Chapter 1* that we had two primary questions in our research. First, we asked how useful is the discrete WPT when applied to the analysis, modification and resynthesis of environmental sounds for aesthetic purposes. To be useful in general, we need to demonstrate a framework and potential interface that is workable for the user. More importantly, we should demonstrate that sonically interesting sound modifications are possible within this WPT framework. The sections on *Sound Modifications* in this and the next chapter, and the *Graphical User Interface* (GUI) description at the start of the next chapter address this question, and in our view, demonstrate that the WPT does indeed have great value as a transform approach for sound signal processing.

Our second question concerned the extraction of useful bases from the WPT complete tree. The complete tree associated with a WPT to any reasonable depth will be comprised of many subbands, and has equal time-frequency resolution across all frequencies. In the sections on *Basis Selection* in this and the next chapter we consider both specific fixed bases and basis selection algorithms to find so-called best bases. Different motives drive the selection of these bases, but we will find they all have something to offer over the complete tree basis in most cases.

4.1 Target Sounds

Before describing our sound modification and basis selection work, we describe the types of sounds we are primarily interested in, and why the WPT would appear to be a good fit for these sounds.

We refer to the types of sounds our research is focused on as *environmental sounds*. By this we mean sounds that occur in the environments we find ourselves in, whether naturally occurring or so-called man-

---

[†] While our implementation uses the *discrete* WPT (DWPT) exclusively, we will generally refer to it simply as the WPT.

made. These *environmental sounds* are in contrast to the sounds created specifically for the purpose of making music, which we refer to as *music sounds*.

Environmental sounds are generally not strongly pitched nor are they comprised of harmonically related components the way that traditional music sounds usually are. There tend to be more *noise-like* components in the overall *mix*. Compared to the sustain we associate with a good deal of the music we hear, environmental sounds are often characterized by relatively sharper onsets and decays. These onsets and decays tend naturally to be broadband, and their time precision must be maintained in any transform we apply if we are to preserve the sound's basic character. Therefore use of a transform that supports fine time resolution and correspondingly broad bands when it is required will help accurately capture these types of sounds. The multiresolution capability of the WPT supports this.

We also point out that many of the environmental sounds we work with are quite short in length. We focus on such short sounds primarily because the environmental soundscape is generally complex and ever changing. To be successful at isolating a somewhat stationary portion of the soundscape, we usually must sample it over a relatively short time. Doing so allows us to work more intentionally with the sound, since it will be comprised of fewer sources in that case. Furthermore, when finding a best basis, we will be more justified in choosing an averaged best basis over the duration of the signal when that signal is short.

Some examples of the environmental sounds we have worked with are foot steps, chirping birds, scraping sounds, running engines, knocking sounds, and buzzing flies. Note that the sounds need not come from the natural environment. They can just as well come from an urban landscape. They are simply sounds that exist in our environment. A list of the sounds used in our testing is provided in Section *5.3 Test Sounds Used*.

Beyond the correspondence between the flexible time-frequency resolution capabilities of the WPT and the types of sounds we are interested in, no further assumptions regarding the types of sounds we are working with are made. To the extent that the WPT is a model, it is *non-parametric*. By projecting a sound onto the basis functions of the WPT, we are breaking the sound up into time-frequency elements of equal resolution. By selecting one of many possible bases from the complete tree basis, we are re-partitioning the sound into time-frequency elements of varying resolution. The WPT itself pre-supposes nothing of the nature of the sound. It simply partitions it. The bases we consider also presuppose nothing of the nature of the sound, but rather they are formed based on some combination of the human hearing process, the make-up of the particular sound being analyzed, and possibly certain user requirements.

We also point out that, precisely because we are working with environmental sounds, it is unlikely that a more *parametric* approach would yield results of sufficient generality. These sounds are not sufficiently structured. Compare this to the case of a musical instrument sound, such as that of the trumpet. While the WPT could be used to transform it and work with it, one might have greater success with a model that specifically models the tonal nature of the sound, such as a model comprised of complex sinusoids whose amplitudes and phases are variable, or a *physical model* that models the waves that exist inside the trumpet. Such models are parametric, and while they will be useful with a small set of sounds, they lack general applicability. With environmental sounds, we do not forego much in using a non-parametric approach since the occurrence of highly tonal sound sources like trumpets is low relative to the environmental soundscape as a whole. What we gain is a good deal in generality.

Note that in this and the next chapter, we will occasionally use Fourier Transform based *spectrograms* to show results. We use these spectrograms because they provide a familiar way of viewing the energy of a signal, coinciding with our well-worn view of sounds in terms of different frequency components. The fact that such a spectrogram uses an over-complete set of continuous, complex tones to project onto also helps make the plots often more readable than our critically sampled, discrete, real-valued wavelet packet transform plots. We find the spectrogram especially useful when the sound does in fact possess a significant tonal aspect to it, such as when we stretch sounds to a great extent. Our spectrograms are created using the *CoolEdit* audio editing software package, using their default colour map[†]. We use DFTs ranging in size between 256 to 1024 samples, depending on whether the signal is best portrayed with more time or frequency resolution. The actual DFT size used for a given spectrogram is listed in the corresponding figure. For windowing of the signal, we have used the *Blackmann* window function.

4.2 Sound Modifications

One of the fundamental questions in this research was to gauge the usefulness of the discrete WPT in the analysis, modification and resynthesis of sound signals. The primary aim in transforming the signal is to be in a position to apply modifications to it that are not possible when working solely in the time domain. If these modifications are aesthetically interesting, then the effort is justified. In the following we describe the collection of modifications developed with this goal in mind, and in the next chapter provide our

[†] *CoolEdit* is a product of *Syntrillium* Software Corporation.

experimental results in working with them. In our estimation, many of these modifications will be of interest to the computer musician.

Unlike either the act of transforming the signal, or of determining a basis for it, both of which involve a high degree of mathematical rigor, sound modification approaches will not in general require such rigorous underpinnings. If the result is interesting, reproducible, and in some way explainable, that will suffice.

The modifications are divided into three main categories. First, in Section *4.2.1 Coefficient Modification and Creation*, we discuss modifications that are performed on the transform coefficients themselves. We also include in this section the case where we create new transform coefficients, also referred to as *granular synthesis*. Next in Section *4.2.2 Resynthesis Modifications* we describe approaches that modify the signal through a variation in the resynthesis process. Finally in Section *4.2.3 Post-resynthesis modifications* we describe modifications that occur after each subband has been resynthesized, but before they have been combined into a single time-domain sample stream. In other words, we take advantage of the signal being split into various subbands, as it is in the case of a graphic equalizer.

While many of these modifications have been applied before with other time-frequency methods, their application with the WPT presents new and interesting twists and challenges. In applying the modifications, we may work directly with the *complete tree* of WPT transform coefficients, or with a subtree of the complete tree, using a *fixed basis* or *best basis* of some sort.

4.2.1 Coefficient Modification and Creation

Here we describe modification approaches that involve changing in some way the transform coefficients themselves. We exploit the time-frequency nature of the signal representation in the transform domain, and in both this section and the next, also take advantage of the separation between *transform coefficient* and *basis function*. In this section we work with the *transform coefficient*, and in *Section 4.2.2* we work with the *basis function*.

The modifications implemented here are *time stretching and contraction*, *coefficient thresholding*, and *coefficient filtering*. We also look at the creation of new coefficient streams via a type of *granular synthesis* operation.

4.2.1.1 Time Stretching and Contraction

With these modifications, we either stretch the signal in time by some factor or contract it. These modifications are useful both for purely aesthetic reasons and also to resize a sound segment for reasons of synchronizing it in some way with other events. From the aesthetic perspective, we will generally be more interested in stretching than contraction, even though the latter can be interesting. For example, in Truax (1994) it is stated that '...(granular) time stretching provides a unique way to experience the inner structure of timbre, hence to reveal its deeper imagery'.

The approach is straightforward, in that we simply resample the transform coefficient stream to achieve the desired length change. To achieve arbitrary length change ratios, we need to be able to both upsample and downsample, with the requisite filtering to avoid images and aliases, respectively.

Specifically, the following steps are taken in resizing the signal[†]:

1. Determine the length change ratio p/q. If $p/q > 1$ then the signal is to be stretched. If $p/q < 1$ then the signal is to be contracted.
2. Design a lowpass filter based on p/q to eliminate aliases and images. An FIR filter is designed using the method of *Least Squares* (Oppenheim,. Schafer and Buck 1999), combined with the application of a *Kaiser* window ($\beta = 5$) to the resulting filter to further smooth the edges in the time domain. The filter cutoff is $\omega_c = \left(\frac{\omega_s}{2}\right)/(max(p,q))$ where ω_s is the sample frequency, and its length is $l = 20max(p,q) + 1$. For example, to either stretch or contract by a ratio of 2.5, a filter length of 51 is used. The frequency response for this particular case is shown in Fig.*4-1*. Use of such a long filter provides for smooth interpolation and good anti-aliasing properties. In comparison, testing with *nearest neighbour interpolation* ($l = max(p,q) + 1$) results in a harsh sounding stretch.
3. Perform upsampling by inserting zeros, if required. For example, if we were stretching by a factor of 2.5, this is equivalent to upsampling by 5 followed by downsampling by 2 after filtering. If we were contracting by 2.5, we would instead upsample by 2, and later downsample by 5 after filtering. Upsampling is not required if we are contracting by an integer ratio amount.
4. Perform filtering of the signal with the filter designed in step 2. This provides interpolation, or image elimination, in case of upsampling, and the necessary anti-aliasing in the case of any subsequent downsampling. Since the filter cutoff frequency $\omega_c = \left(\frac{\omega_s}{2}\right)/(max(p,q))$ is based on the maximum of *p* and *q*, it will be low enough regardless of whether we do more upsampling or downsampling.
5. Perform downsampling by removing samples, if required. Downsampling is not required if we are stretching by an integer ratio amount.

[†] The *Matlab* function *resample* (*Signal Processing Toolbox*) conveniently implements steps 2-5. We used this function with the default filter parameters (filter length, windowing).

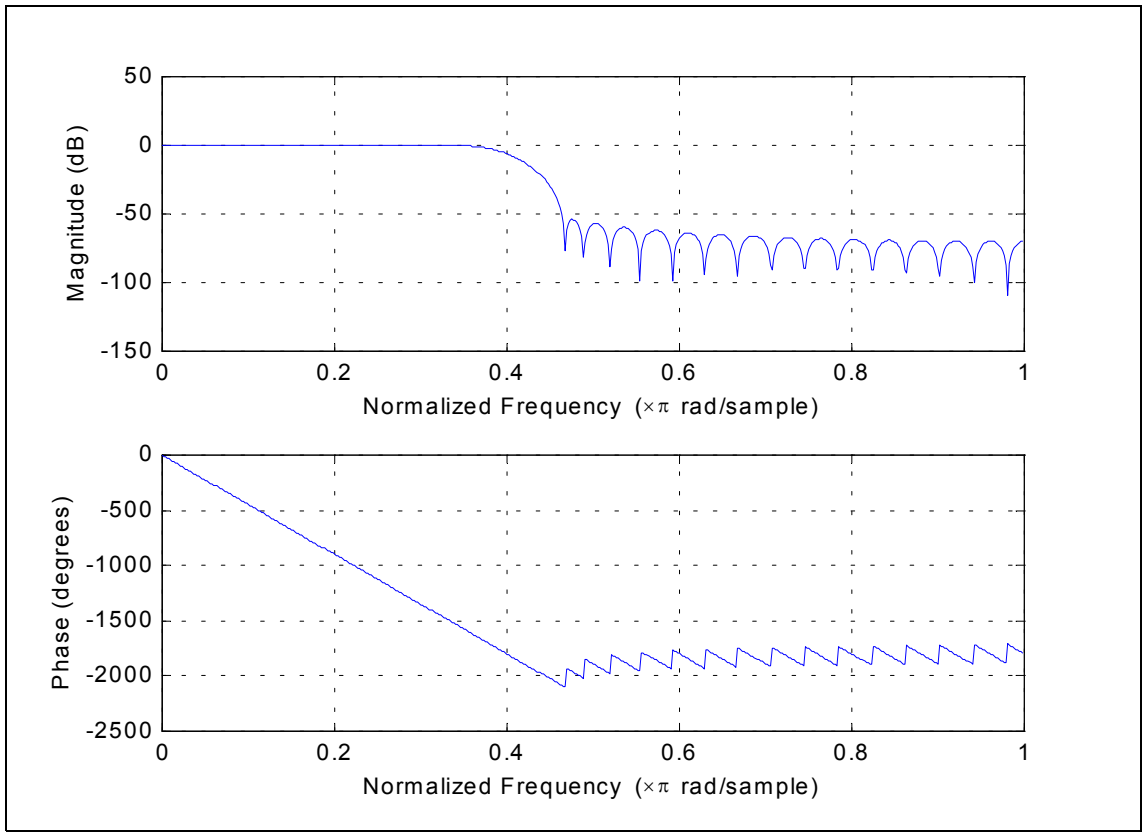

Figure 4-1: Frequency Response of Resampling Filter (2.5x time change)

It is worthwhile to consider the spectrum of the resulting signal. Resynthesis involves filtering (with upsampling) of each subband coefficient stream with the corresponding synthesis filter, followed by addition of all subband results. In the frequency domain, this filtering operation corresponds to multiplication of the coefficient spectrum $C(\omega)$ with the wavelet packet basis function spectrum $W(\omega)$. The output spectrum S_{out} is given by

$$S_{\text{out}}(\omega) = \sum_{\text{all subbands}} C(\omega)W(\omega) \qquad \textbf{(60)}$$

Equation (60) makes it clear that if we change the spectrum of the coefficient stream, the output spectrum will also change. If for example we stretch the coefficient stream, the overall frequency content of $C(\omega)$ will shift down, and we can expect a lower frequency content for the output spectrum as a whole. We do indeed hear this lowering of pitch in our experimentation, and as such, cannot expect to preserve pitch during

stretching. In *Chapter 5* we will have more to say about specific tests and their results.

We can contrast time stretching and contraction with the WPT to the results achievable with an STFT-based transform. As we saw in *Section 2.5.1*, the STFT with *oscillator bank* (OB) resynthesis can be effective in this area, especially if the sound is highly tonal in nature. On the other hand, the target sounds of our work are not highly tonal in general, so an approach that is less tailored to tones is justified.

When comparing with the STFT using *overlap-add (*OA) resynthesis, while the OA approach exploits the use of a highly overlapped transform and a great deal of redundant data, the critically sampled WPT relies only on the interpolation previously described. It is not clear that in theory there should be any benefit to this high degree of overlap, other than ease of implementation due to no interpolation filtering being required. With the WPT, there is no redundant overlapping data to draw from when the coefficients are stretched in time. But upsampling with interpolation should achieve the same result.

4.2.1.2 Thresholding

With the technique of *thresholding*, the basic idea is to set an amplitude threshold such that any coefficients that are less than this threshold are set to zero. The threshold can be set on a subband basis, or globally, so that a single value is used across all subbands. We can choose to threshold any or all of the subbands.

An option also exists whether to perform *hard* or *soft* thresholding. If we call the threshold t, then for signal s the *hard* thresholding operation can be defined as

$$s_{\text{threshhard}} = \begin{cases} s \text{ if } |s| > t \\ 0 \text{ if } |s| \le t \end{cases} \quad \textbf{(61)}$$

Whereas *soft* thresholding can be defined as

$$s_{\text{threshsoft}} = \begin{cases} \operatorname{sgn}(s)(|s| - t) & \text{if } |s| > t \\ 0 & \text{if } |s| \le t \end{cases} \quad \textbf{(62)}$$

Thresholding based modifications have their origins in the applications of noise reduction and data compression. In the case of noise reduction, when the noise superimposed on a signal is high in frequency relative to the frequency content of the signal itself, subband-specific thresholds can be set to favour the elimination of these noise components. Since the noise is additive, it makes sense to use soft thresholding for those subbands, effectively subtracting out the noise component. There are various methods of calculating the required threshold amplitudes depending on the assumptions of the frequency content of the noise and how aggressively it is to be eliminated. In the case of data compression, hard thresholding is applied to the

subbands of the signal, either globally or locally. Coefficients above the threshold are deemed important enough to keep, and coefficients below the threshold are deemed expendable. By employing hard thresholding, we do not alter the coefficients that are above the threshold.

While we might find the noise-reduction capability useful in cleaning-up a sound signal prior to further processing, neither the noise-reduction or compression applications are really central to our purposes. On the other hand, thresholding can be used as a way to modify a sound in interesting ways. In general, if we threshold quite aggressively, we can roughen up the sound of a given signal by creating discontinuities. By being selective in the threshold values used across subbands (which can be time consuming and can therefore be one motivation for finding a best basis with a small number of subbands), we can target specific aspects of the sound. For example, we might want to use high threshold values in subbands that possess high amplitudes and vice versa, thereby skewing the signal energy to the lower energy bands while at the same time chopping it up across all subbands.

4.2.1.3 Filtering

The filtering of one or more of the subband coefficient streams opens up a broad area of possibilities given the multitude of different filters one could use. We have implemented a simple lowpass filtering function to test the basic approach. The order and cutoff frequency of an FIR filter are selectable parameters in this design, and the filter is constructed using the *window method* (Oppenheim, Schafer and Buck 1999).

4.2.1.4 New Coefficients (Granular Synthesis)

The ability to perform *granular synthesis* fulfills the promise of using wavelet packets as a framework within which to do the *analysis-modification-resynthesis* of pre-existing sounds along with the synthesis of completely new sounds.

A granular synthesis capability is achieved by allowing the generation of new coefficients in the transform domain. The wavelet packet impulse responses are the grains, and they are triggered by the transform coefficients. To see this in a way that most resembles traditional granular synthesis, we imagine that the coefficient stream is a series of impulses at a certain frequency. This series of impulses is implemented as a non-zero coefficient, followed by some number of zeros, then another non-zero coefficient, and so on. These impulses effectively trigger or excite the wavelet packet impulse responses when they occur, producing a train of wavelet packet impulse responses[†]. An example is shown in Fig.*4-2*.

We can compare this to a typical granular scheme such as the GSX (*Section 2.6.1.1*), where a waveform

is windowed to produce the grain. That grain is equivalent to our wavelet packet impulse response. The grains in GSX are sent out at some frequency. In our case, this is determined by the rate of the impulses in the coefficient stream.

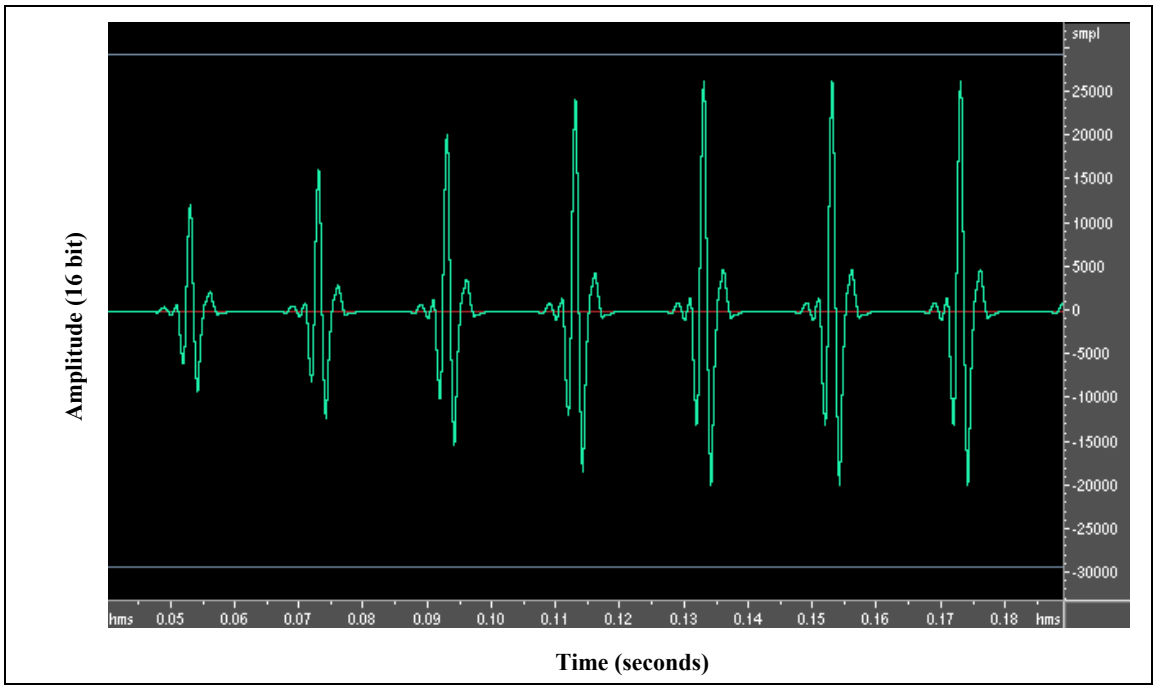

Figure 4-2: Example of Granular Synthesis Waveform
(*sym5* filter / test case *Gran5*)

As stated in our description of granular methods in *Section 2.6*, granular synthesis can be viewed as a form of AM. Looking at Fig.*4-2* one can see the validity of this perspective. We can view the waveform roughly as a high frequency wave that has been modulated by a lower frequency wave. As in the case of AM, a phase change in the carrier does not occur with modulation, since the modulating wave is offset to eliminate zero crossings. The spectrum that results in the case of AM is the carrier frequency, along with a collection of side bands at distance increments from the carrier of the modulating frequency. Fig.*4-3* shows the spectrogram associated with the waveform, and demonstrates the AM nature of the spectrum just described. The wavelet packet (*carrier*) frequency is around 500 Hz, and the lower cluster of spectral lines is centered around this location. The side bands are in 50 Hz increments, which equals the rate of impulses

---

† Convolution is most easily visualized if the filter input is an impulse, in which case the output is simply the impulse response of the filter, which in this case is the wavelet packet impulse response.

(*modulating frequency*), as can be seen in Fig.*4-2*[†].

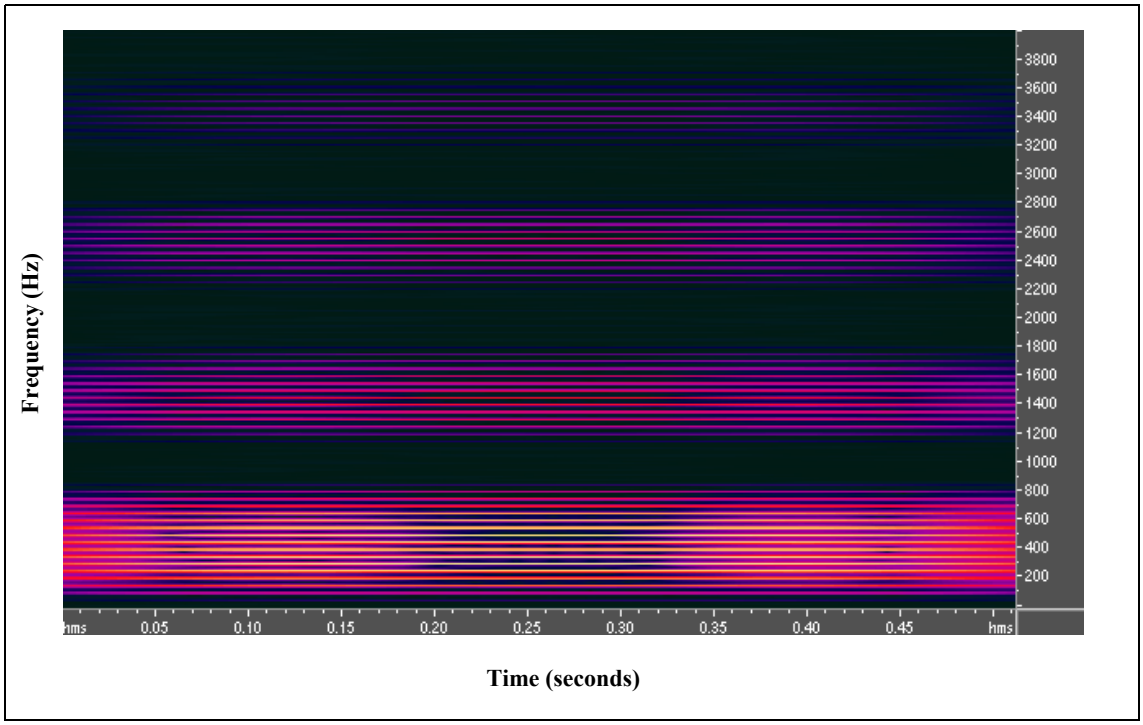

Figure 4-3: Spectrogram of Waveform shown in Fig.4-2
(*sym5* filter / test case *Gran5* / 1024 point DFT)

The upper three frequency clusters in Fig.*4-3* are due to *imaging*. Imaging occurs after upsampling and is generally eliminated by filtering; often called *interpolation* filtering. Imaging is the flip-side of *aliasing*, in that aliasing results in many frequencies (two in the case of downsampling by two) appearing as one, and *imaging* results in one frequency appearing as many (two in case of upsampling by two). As such, our description in *Section 3.4.1* of how aliasing manifests itself in the hierarchical WPT applies in an analogous way to the imaging we now describe here.

Granular waveforms using packets centered on frequencies in the lower half of the spectrum, such as the subband (4,1) packet of our example, will naturally correspond to spectral lines in the lower half of the spectrum (0-2 kHz in the case of an 8 kHz sample rate). Similarly, granular waveforms using packets

---

[†] This waveform corresponds to the test case *Gran5*, which uses the packets from subband (4,1) for its grains. The packet filter used is called *sym5*, and is used in much of our testing. More details on the conditions that produced *Gran5* and the other examples described and shown in this section can be found in *Table 5.6* of *Section 5.4.1.4*.

centered on frequencies in the upper half of the spectrum will have their spectral lines in that upper spectral half (2-4 kHz in the case of an 8 kHz sample rate). These granular synthesized waveforms are implemented as upsampling followed by filtering, where the filter used corresponds to the packet. Since the filters can never be perfect, images will never be completely eliminated. Therefore the lower frequency packets will have images in the upper half of the spectrum, mirrored about $f_s/4$ (2 kHz in this example), and vice versa. But, the granular waveform is really the result of several levels of upsampling and filtering; four in our current example. Therefore images also occur, or are mirrored, around the frequencies *fs/8*, *fs/16*, and *fs/32* in our *depth-4* scenario. So the spectral lines in Fig.*4-3* centred on 1.5 kHz are due to imaging and imperfect filtering at *depth-2*, and those clustered around 2.5 kHz and 3.5 kHz are due to imaging and imperfect filtering at *depth-1*. Furthermore, at depths 3 and 4 there will be some imaging around 500 Hz and 250 Hz respectively, though this is harder to single out given the sidebands that already exist due to the AM modulation process.

Notice that the images just described in Fig.*4-3* are much lower in intensity than the original frequency lines clustered around 500 Hz. In the same way that sharp filters minimize the intensity of aliases, they also minimize images. This is evident in Fig.*4-4*, where the longer 20 coefficient *db10* filter was used, and suggests that if the time resolution of a short filter is not required, a longer filter may be better if images are to be avoided (at least for granular synthesis operations). Alternatively, additional filtering could be employed, but this is computationally expensive, since such filtering would occur at the full sample rate of the signal.

More complex coefficient streams than the impulse train are of course possible, and the resulting spectra are also more complex. We have experimented with various coefficient stream shapes. These include sine, square, and sawtooth waves, as well as the possibility of simply using a constant level (DC). In all these cases and the impulse train case, the overall stream can be subjected to a three-piece envelope with specifiable rise time, plateau time and fall time as relative percentages. This envelope should not be confused with the enveloping of the grain, which again in the case of the wavelet packet is implicit.

When we use a DC coefficient stream the spectrum is simple. This is because the packets get strung together to form a continuous wave. The frequencies of this are those of the packet, which will be some fundamental, and possibly one or more harmonics. As we will see later in *Section 5.2*, the *sym5* packet we use in most of our testing is quite smooth and sine wave-like. As such it produces a strong fundamental. For example, using just subband *(4,1)* and a sample rate of f_s = 8 kHz, *sym5* produces a strong fundamental at

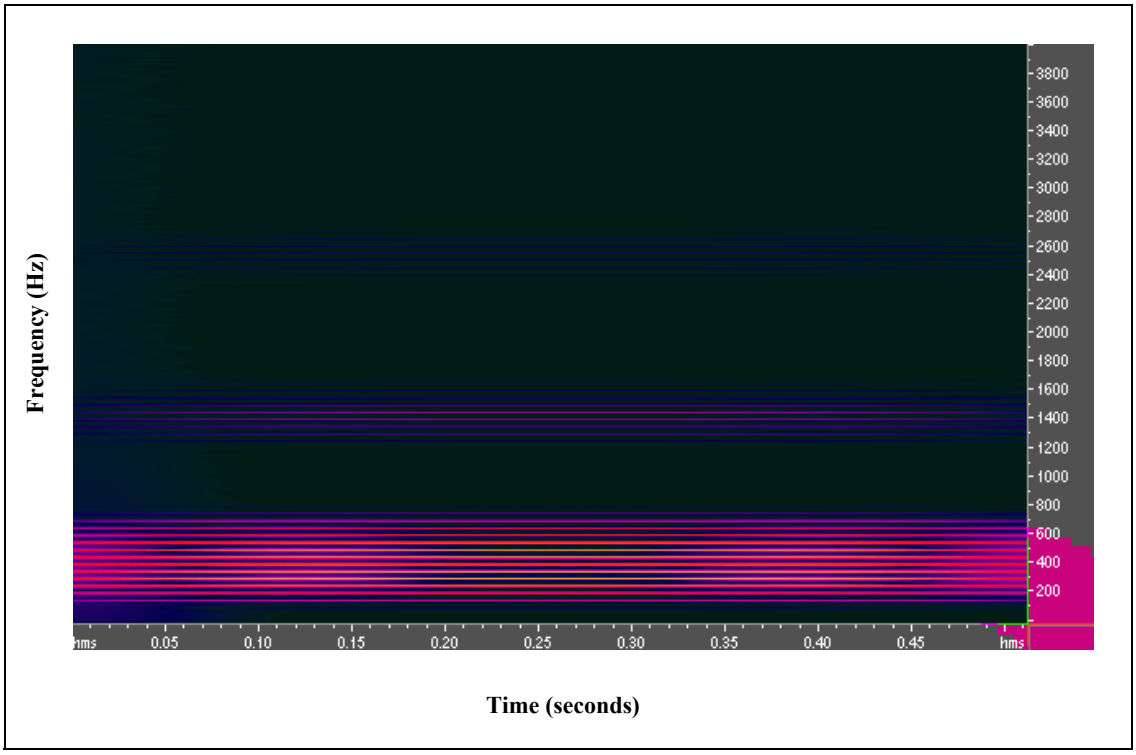

Figure 4-4: Spectrogram of Waveform shown in Fig.4-2 using db10 instead of sym5
(*db10* filter / test case *Gran23* / 1024 point DFT)

500 Hz plus a third harmonic at 1500 Hz[†].

In the case of the other waveforms, phase changes in the resulting signal occur. This is because the coefficient stream is not offset to avoid zero crossings, which it is in the case of our impulse train modulation (and AM). The shape looks like that of the coefficient stream waveform modulating the fundamental of the packet frequency if it were not localized, or effectively windowed, in time[††]. This process is like *Double Sideband Modulation* (DSB), which is similar to AM, but with the exception that the modulating waveform is not offset to avoid zero crossings. Time and time-frequency domain examples of this are shown in Fig.*4-5* and Fig.*4-6* respectively. The resulting spectrum will not contain a component at the carrier frequency, and

---

[†] While a longer wavelet would have provided better alias and image suppression, we chose the 10 coefficient *sym5* wavelet as a compromise between achieving good subband isolation and time resolution. Use of a standard set of test parameters (including the *sym5* wavelet) across all tests provided a standard reference point from which to explore variations In practice, the user might want to vary the wavelet and associated parameters based on the application at hand.

[††] The process is different however. The coefficients that comprise the modulating waveform are in fact filtered by the packet impulse responses (which are local in time). Rodet (1980) used this duality in reverse, starting with the need to generate formant regions based on the pulse train/filter model of vocal sound production, and converting that to equivalent windowed waveforms (Roads 1996).

our example shows this. There is no spectral component at the fundamental of the packet, only on either side of the carrier frequency, at a spacing of roughly the modulating frequency. We can also see the modulating frequency at 50 Hz.

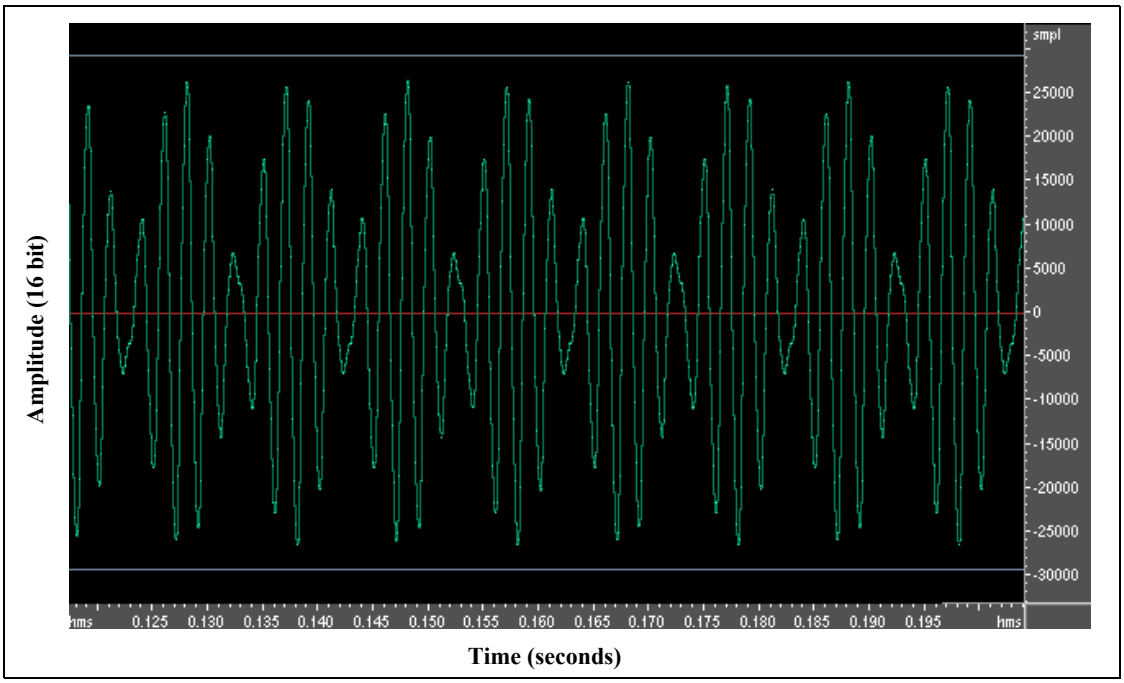

Figure 4-5: Sine Wave Shaped Coefficient Stream Segment
(*sym5* filter / test case *Gran13*)

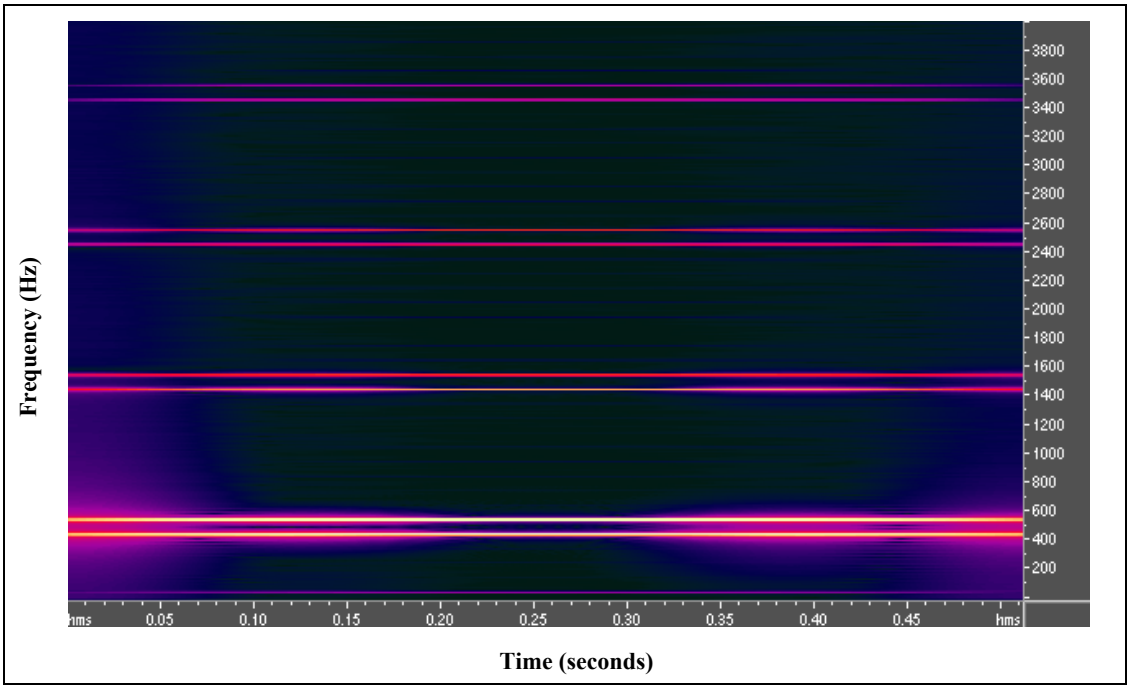

Figure 4-6: Spectrogram of Sine Wave Shaped Coefficient Stream Segment
(*sym5* filter / test case *Gran13* / 1024 point DFT)

4.2.2 Resynthesis Modifications

These modifications are done in the resynthesis process. The modifications implemented are resynthesis with a different wavelet filter, and resynthesis using an arbitrary sound as the filter. As such, the approaches differ only in the resynthesis "*filter*" used.

4.2.2.1 New Filter Resynthesis

In this approach, we simply use a different wavelet packet filter for synthesis than the one used in analysis. We still prefer the analysis filter bank to be orthogonal so that any subsequent best basis searches may be performed effectively. Also, by consistently using the same analysis transform for all sounds and modifications, the user will more easily develop familiarity with transform plots and their visual interpretation. However, for resynthesis we allow the replacement of the synthesis filter bank that corresponds to the one used in analysis with some other wavelet packet synthesis bank.

Of course we cannot expect, and do not seek perfect reconstruction in this case. Our desire is that we may create interesting sounds as a result of the different resynthesis filter. We may also use the technique for specific purposes, such as smoothing a sound that possesses coefficient streams with high frequency content. We would do such smoothing by specifying a very long filter for resynthesis, possibly even a biorthogonal filter. In this biorthogonal case, the filter length between analysis and synthesis need not be the same.

4.2.2.2 Arbitrary Sound Cross Synthesis

The idea in *cross synthesis* is to cross the characteristics of two different sounds, creating a sound with characteristics from both. As with the resynthesis approach in the previous section, we again modify in the overall transform only the filters used in resynthesis. However in this case, any sampled sound is a potential candidate for the resynthesis filtering operation.

4.2.3 Post-resynthesis modifications

These are modifications implemented to operate on the subbands after they are resynthesized. This can be done when the resynthesis has occurred on a separate subband by subband basis prior to any summation of the subband resynthesis results[†]. Since the WPT can create a large variety of different subband widths within

---

[†] We have chosen to implement our resynthesis in this fashion so that we could plot and view the individually resynthesized subbands separately. While this will be less efficient than the FWT, it gives us additional control in our test environment. Also, since our coefficient and resynthesis modifications are often only applied to a subset of the subbands, only the processing for those subbands is performed, and the performance loss relative to the FWT is not as large. Note also that the FWT is still used for the analysis portion of the transform.

a single transform, the result of the modifications we impart here are in part due to the nature of the transform and basis selection that preceded it, even though resynthesis filtering has already taken place.

Modifications done after resynthesis tend to have a short delay since little processing is required beyond that required for the modification itself. We simply operate on the resynthesized subband samples and then add up the results across the subbands. In the case where a faster resynthesis algorithm is used such that the subbands are not resynthesized separately (i.e., the FWT), the modifications we describe here could instead be applied to the transform coefficients themselves with similar results.

The modifications we have implemented are the scaling of subband amplitudes (*equalization*), and the modulation of subband amplitudes.

4.2.3.1 Scaling Subband Amplitudes

In the final adding up of the various resynthesized subbands, we can choose to scale the amplitudes of some of them. In the extreme, we can scale their amplitudes to zero, effectively leaving out those subbands.

A primary application of this function is in sound *equalization*. Since we can select one of many subband decompositions from the complete WPT tree, we can create equalizers that control the spectrum of the sound in varying ways. For example, we can select a basis that conforms in some way to psychoacoustic requirements, such as the *PTOB* described later in *Section 4.3*, and thereby be able to manipulate subbands whose widths are reasonably close to the critical bands of human hearing.

While the complete zeroing of amplitudes is a special case of scaling, it is an important one. By thinking in these terms, we can either strip down a signal one subband at a time to hear the effect, do the opposite to build up the sound from scratch, or do some combination of the two. As with the case of general amplitude scaling, it is usually helpful if we have a basis other than the complete tree, since this provides some structure to the changes. Such a basis might have the subbands coinciding better with psychoacoustic realities, or it might be partitioned in a way that coincides better with the energy distribution of the signal itself.

4.2.3.2 Modulating Subband Amplitudes

Rather than applying a fixed scale amount to a given subband as in the previous section, we can instead modulate a subband's amplitude over time in some prescribed way.

We have looked at applying various wave shapes to the resynthesized subbands, including sine, sawtooth,

and square waves. We have applied them both as AM (*Amplitude Modulation*), whereby the modulating wave is offset above zero, and as DSB (*Double Sideband Modulation*), where the wave is centered around zero. Recall from our granular synthesis discussion in *Section 4.2.1.4* how the spectra resulting from these two modulations are somewhat different.

4.3 Fixed Basis Selection: The PTOB

In this and the next section we describe the approaches taken to *prune* the complete tree wavelet packet basis to something more useful. By *useful* we mean one of many possible things, such as more manageable, meaningful, storable, or computable. Recall we discussed some of the merits of basis selection in *Sections 3.6* and *3.7*.

From *Section 3.7* we know that a manual approach to basis selection, where the user selects each possible basis and subjectively compares the results, will be prohibitively time consuming. It would also seriously challenge the user's ability to remember the quality of each basis over time to compare with the others. Therefore we seek useful *fixed bases* and/or algorithms to help us find useful *best bases*.

In this section we focus on the *fixed basis*. The form of a fixed basis is generally chosen independently of any signal being transformed. Rather its shape is motivated by psychoacoustic considerations. The wavelet basis, the complete tree wavelet packet basis, and any fixed basis between these two extremes are possible. In particular, a basis we refer to as the *pseudo-third-octave basis* (PTOB) is described here. It starts with a WT basis and refines the frequency resolution of each WT subband to align the subband structure more closely with the third-octave spacing and critical bands described in *Section 3.6*.

Before proceeding, we briefly restate some terminology and semantics regarding the ensuing tree diagrams. We use the terms *subband*, *leaf,* and *terminal node* synonymously. In our tree diagrams, a smaller font is used for internal nodes and terminal nodes whose coefficients have been zeroed (zeroing can occur in some of the best basis approaches). A larger font is used for terminal nodes with non-zeroed coefficients.

4.3.1 Overview and Motivation

As we saw earlier, the discrete dyadically spaced WT basis is quite limiting in the frequency resolution it provides. Correspondence with the critical bands is poor, making it of limited value as a basis for equalization applications[†]. Furthermore, the coarse and fixed nature of the subbands does not offer the flexibility generally required for sound manipulation purposes. It is not ideal on either front. In Kahrs and Brandenburg (1998) the authors state with regards to the shortcomings of the resolution offered by the dyadic DWT for data compression that "... at high frequencies for some signals the larger coding gain of a high frequency resolution is needed". They are referring to the poor frequency resolution of the dyadic DWT

[†] The benefit of critical band correspondence was discussed in Section *3.6 Fixed Bases*.

at high frequencies, which allocates only a single subband for the entire upper half of the spectrum. While our application is different, the resolution shortcomings are equally problematic.

Our approach to deriving a better fixed resolution basis than the WT is to model the subband decomposition based on the *third-octave* spacing of many graphic equalizers. In that case, each octave is split into three bands, logarithmically. Referring back to Fig.*2-2*, we can see that the *fourth-octave* curve (i.e., labelled *minor third*†) lies above and approximately parallel to the critical band curve. A true *third-octave* split will lie above this minor third curve since its bands are a little wider, but it should coincide with the general trend of the critical band curve as well.

4.3.2 Description

To determine how to approximate third-octave spacing, we first derive the relationship between bands. We would like a fixed ratio between the upper band edge frequency of a given band and the upper band edge of the band immediately above it, such that exactly three bands fit within one octave (or one doubling of upper band edge frequency)††. Therefore $(x^3 = 2)$, or $x = 2^{(\frac{1}{3})} \cong 1.26$.

In this case, the relative bandwidths occupied by the three bands within an octave are:

- First band: $26\% = ((1.26 - 1) \times 100)\,\%$
- Second Band: $33\% = ((1.26^2 - 1.26) \times 100)\,\%$
- Third Band: $41\% = ((1.26^3 - 1.26^2) \times 100)\,\%$

These relative bandwidths are not too far from a split of 25%, 25%,50%, which is readily attained by filtering a given band into lowpass and highpass components, and then filtering the lowpass output again. If we perform this two-pass filtering on bands that are octave spaced, we should get something close to third-octave spacing. Since the WT basis corresponds to octave spacing already, the PTOB can be viewed as a refinement of the frequency resolution of the WT basis. The original bandwidth of any of the WT basis terminal nodes is split into three subbands, the lower two each occupying 25% of the original bandwidth, and the wider band occupying the upper 50% of the band.

Fig.*4-7* shows a typical tree for a PTOB. Due to the aliasing of subbands (Section *3.4 Aliasing*

---

†. The *minor third* interval spans three semitones, and the octave spans 12 semitones. 12/3 = 4, so this represents *fourth octave* spacing.

††. One could just as well work with the lower band edge, or centre frequency.

Considerations), it is the highpass filter outputs that are filtered again to produce the terminal nodes, with the exception of the lowest subband, since it originates from a lowpass filter output.

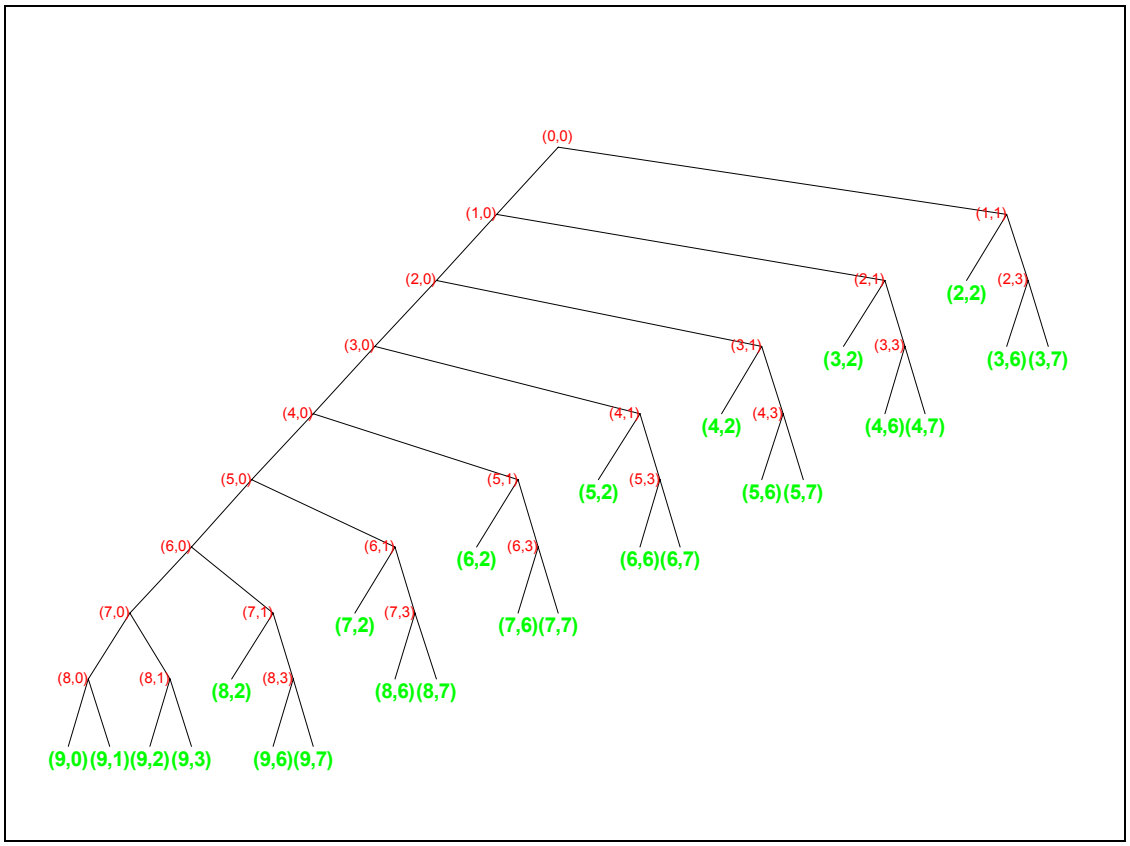

Figure 4-7: Typical PTOB Tree - Depth-9

Note that once we flip all the highpass filter outputs (i.e., nodes (8,1), (7,1), (6,1),...) of the original WT basis left over right to account for aliasing of subbands, at the low end of the spectrum, we get the node sequence (9,0), (9,1), (9,3),(9,2), (9,6), (9,7), (8,2),(8,6),(8,7),(7,2),(7,6),(7,7),... If we had only filtered the low frequency result of the initial split of the WT terminal node, (8,1) would be a terminal node instead of its child nodes (9,3) and (9,2). But doing so would result in a lower resolution band surrounded by higher resolution bands. This low end of the spectrum is a special case due to there being a highpass and lowpass result at the bottom of the original wavelet tree. By also filtering the (8,1) node, the potential anomaly of having a lower resolution band surrounded by higher resolution bands is avoided. The end result is six equal width bands at the low end of the spectrum and three everywhere else except at the high frequency end, where one band of unique width exists, representing 25% of the overall bandwidth.

A simple relationship exists between the number of PTOB terminal nodes and the depth of the original WT basis. The original WT basis possesses $d_{WT} + 1$ terminal nodes, where d_{WT} is the depth of that basis. For each WT terminal node, the PTOB creates 3 terminal nodes, plus one extra at the low end. Therefore, if L denote the number of *leaves* or *terminal nodes* in a tree,

$$L_{\text{PTOB}} = 3(d_{\text{WT}} + 1) + 1 \quad \textbf{(63)}$$

For example, given a *depth-7* WT basis, the PTOB will have 25 terminal nodes. Since the depth of the PTOB will always be two greater than that of the associated WT basis, in this example the PTOB is of *depth-9*.

Fig.*4-8* shows a plot of the PTOB bandwidths versus centre frequency for the *depth-9* case and a sample rate of 32 kHz. In the same figure, we also show the true *third-octave* spacing at 32 kHz for comparison, as well as how both the critical bands defined in (1) and the dyadic DWT basis bandwidths scale with frequency. While the fit of the PTOB to the true *third-octave* spacing appears somewhat choppy, it does in fact trend in the desired way by straddling the third octave curve. As such, it does achieve a somewhat reasonable approximation. Comparing to the *critical bands*, while the bandwidths of the PTOB are often larger, again, the general trend or slope of the curve appears approximately correct.

The dyadic WT basis, on the other hand, can clearly be seen to be a poor match for either the true third octave or critical bandwidth curves. Even at low frequencies, the dyadic WT subband widths rapidly become too large, and the divergence from the other curves with increasing frequency is marked.

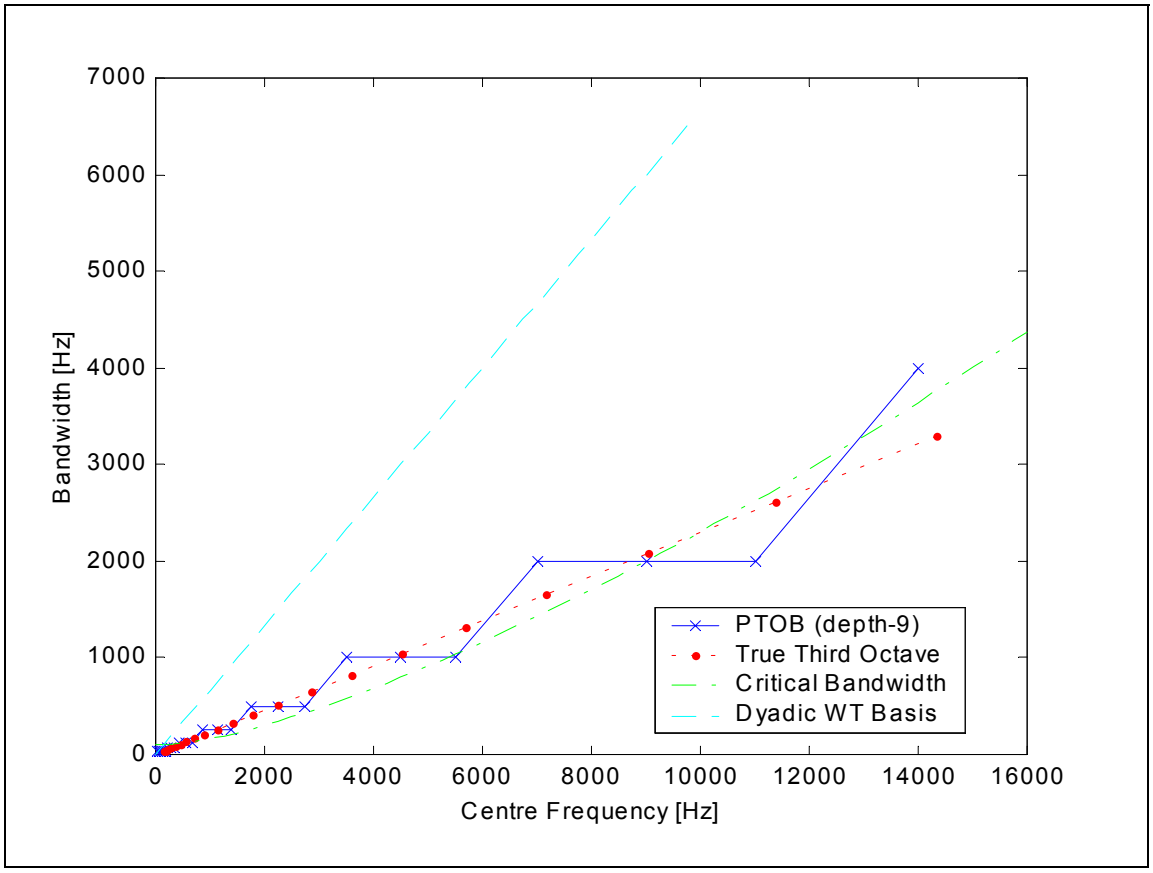

Figure 4-8: Comparison of PTOB Bandwidths with Other Scenarios
(PTOB Depth = 9, f_s = 32 kHz)

For a different sample rate than that used in the example of the figure, the PTOB subband locations and widths will correspond to different frequencies, so that a different tree depth would be required if the approximate correspondence with the critical bands is to be maintained. For example, a 29 band basis using wavelet packets that approximates the critical bands for a 44.1 kHz sample rate is given in Sinha and Tewfik (1993).

4.4 Best Basis Selection

In this section we describe a framework for best basis selection, followed by descriptions of several best basis selection approaches that fit within this framework. Primarily, we focus on an approach we refer to as the *Bi-rate Distortion Best Basis Algorithm* (BDBBA). However, we also briefly consider several other approaches, including the EBBA described earlier. The fact that all our approaches are part of a single

framework within which other schemes may be incorporated is a powerful feature. Future requirements for a basis with certain characteristics could be shaped into a new cost function that our existing framework would use to derive that basis.

Before proceeding, we emphasize several points. First, our application is basis selection for the purposes of sound representation and modification. Signal compression is not the primary objective, even though it might be a benefit in certain contexts. Since many of the transforms and basis selection approaches in existence were originally motivated for compression purposes, they may not be applicable to our work. For example, in compression the goal is reduction of the overall bit rate or number of bits required to represent the signal. While selection of one basis over another in that case may aid in data reduction, the basis itself is a means to an end. A basis with many subbands may well provide equal or better data compression than a basis with few subbands. In our case, the means and the end are switched. Our goal is a desirable basis, and any data reduction will generally be a means to that end. We would like a basis whose subbands exist because they are significant to the representation of that sound.

Second, we will generally seek a basis that is constant over the duration of the signal. This requirement implies either a *fixed basis*, or a *best basis* approach that includes some form of averaging over any non-stationarities in the signal. Along these lines, we also do not attempt to exploit masking effects as is done in audio compression (see Section *2.3.1 Psychoacoustics*), since doing so relies on the particular local details of the signal, which change rapidly over time[†].

Third, we point out that all of the following basis selection approaches can be supplemented with manual choices inside of the software environment that we have designed (see Chapter 5). Manually splitting or joining subbands anywhere within a tree is straightforward. Having this capability gives the user an added degree of control without the burden of full basis selection.

4.4.1 Best Basis Algorithm Framework

Here we provide the general framework for best basis selection that our work has utilized. Central to this framework is the definition of a cost function, and the pruning of the WPT complete tree based on this cost

---

[†] Note that nothing in our approaches preclude updating the basis over the duration of the signal. Doing so may be desirable in certain cases, such as when the analyzed sound is of long duration (recall from *Section 4.1*, that our target sounds are generally of short duration). In the case of longer, changing sounds, a basis that adapts to the given sound over time may help the user better understand and work with those sounds. Then, if a more complex time varying basis within a modification-resynthesis scheme were acceptable to the user, the benefit of a time varying basis could be realized.

function to derive a particular basis. In *Section 3.7* we saw two examples of this general approach. In the case of the EBBA, the cost function is a type of entropy. In the case of the RDBBA, the cost function is *Rate-Distortion* based.

We generalize these schemes with the following cost definition:

$$J(\lambda) = J_1 + \lambda J_2 \quad \textbf{(64)}$$

In the case of the RDBBA, J_1 is distortion and J_2 is rate. In the case of the EBBA, the pruning is lossless so there is no trading off of quality against a resource such as rate. As such, J_2 is not required, and J_1 is an entropy related measure.

For our purposes, we are not interested in rate or entropy reduction as an end in itself. We are often interested in reducing the number of terminal nodes, either through merging nodes or deleting their coefficients. Such a reduction in terminal nodes can make the real-time manipulation of the transformed sound more practical, since the user's two hands limits the number of terminal nodes that he or she can modify at one time. We are also often interested in a basis that maps well to the key features of the signal in terms of time-frequency resolution. We may even hope to learn something about the signal through the basis selection process.

Within the general framework of (64), we can identify at least three classes of cost functions that might be of interest to us:

1. *Rate-Distortion* based cost functions
2. *Entropy* based cost functions
3. *Model* based cost functions

The *Bi-rate Distortion Best Basis Algorithm* (BDBBA), which we describe in the next section, belongs to the first of these three classes. In this scheme, we trade-off quantization at each node with the distortion that will result, but limit the number of potential quantizers at each node to two (thus the *Bi-rate* prefix). This two quantizer approach allows us to make a binary distinction between subbands, those quantized with a high rate and those quantized with a low rate. We consider the extreme case for much of our description and testing, where subbands are either quantized with sufficient bits so that no distortion results, or with zero bits so that the resulting distortion equals all of the energy in the subband. Doing so forces the algorithm to pick the most 'important' parts of the signal, and the basis tree that would be associated with it. We will refer to this quantization scenario for the BDBBA as the *all-or-nothing* quantization case.

In Section *4.4.2.3 Quantizer Alternatives*, we look at other quantizer choices for the BDBBA, such as where we quantize with few bits for the low rate quantizer and something less than the number of bits required to achieve no perceivable distortion for the high rate quantizer. By adjusting the bits used, we can change the thresholds at which the algorithm does or does not split a given node into its child nodes. We also point out that even though the BDBBA involves quantization, we can optionally use the algorithm strictly to find a basis, and afterwards restore all coefficients to their pre-quantized values.

The second class of cost functions involve the application of an entropy-like definition in one way or another. The advantage of these schemes is that they are lossless. The disadvantage is that because they are lossless, the only way to achieve a basis with few nodes, which we will often want, is to have very large bandwidth subbands. In any case, these bases offer the prospect of somehow allocating the flexible time-frequency resolution of the WPT to the signal in a way that is suitable for the signal. In addition to considering the EBBA approach of Coifman and Wickerhauser (1992), we look at other entropy-like definitions, such a *threshold entropy*, and also consider basing the cost on an estimate of the real entropy of the signal. We discuss this class of algorithms in Section *4.4.3.1 Information Cost Based Schemes*.

The third class of cost functionals we refer to as *model based* because the idea is to model the coefficient streams in each subband. Doing so may help us better understand and/or create a simplified model for the sound. We look at the particular case of applying a *predictor* to each subband. In that case, our cost function trades-off prediction error power with either predictor order, or the rate associated with encoding the predictor. The lower our prediction error, the better the model is. The higher the order and/or rate, the higher the cost associated with the model. We discuss this class of algorithms in Section *4.4.3.2 Model Based Schemes*.

We now describe the BDBBA. Its starting point is the RDBBA described in *Section 3.7.2*, with changes to suit the needs of sound analysis and modification. Since the BDBBA is relatively complex compared to the other approaches, and also because it contains the most in terms of original contribution, its exposition occupies more space than other schemes.

4.4.2 Bi-rate Distortion Best Basis Algorithm (BDBBA)

This section describes the *Bi-rate Distortion Best Basis Algorithm* (BDBBA). After a brief overview of its functionality, we review the motivation for such an algorithm, and follow that with a detailed description.

4.4.2.1 Overview and Motivation

The BDBBA provides a means for selecting a *best basis* from the original complete tree of the WPT. It can be viewed as a special case of the RDBBA where exactly two quantizer possibilities exist for each node in the tree. By using this binary scheme for quantizing the node coefficients, we effectively divide the terminal nodes of the resulting basis into two categories: those that are significant enough to warrant quantization with the higher rate quantizer, and those that may be quantized with the lower rate quantizer. A basis and associated tree are simultaneously determined in the process.

The dividing of subbands into the two quantization categories can delineate energy bands within the signal spectrum so that subbands of appropriate depth may be formed. The cutoff or boundary between the two categories of subbands is determined by a user specified *quality budget*. As such, we have a user directed algorithm that finds a lowest cost (distortion) basis for a specified budget. The distortion added by the quantization of a subband can optionally be weighted by its psychoacoustic relevance to favour those parts of the signal that are most important to the listener.

Once the BDBBA has selected a basis, one can still choose whether to use each subband's coefficients in their quantized form, or to return them to their original undistorted values. If one is only interested in the selection of a basis, then restoring to the unquantized form might make sense. In this case, the high and low rate subbands can still be delineated for the user, so that subsequent sound modification efforts can be focused on the more significant high rate subbands. If one is interested in a reduction of the data in some way, such as a reduction in the number of terminal nodes of the basis, or one wants to explicitly modify the sound via a quantization process, then retaining the quantization will be more appropriate.

A selection of the two quantizers must be made for a given run of the BDBBA. We highlight the *all-or-nothing* quantization choice mentioned earlier, since it is so useful and has been used as the default mode for much of our work. This choice effectively involves the use of zero bits for the low rate quantizer, and a sufficient number bits to impart no distortion on the signal for the high rate quantizer, such as a *double* data type (32 bits) as output by the WPT. The *all-or-nothing* case utilizes the two extremes in number of bits used for a uniform scalar quantizer, from the least to the most. Consequently, it is consistent with our general conception of quantization and the framework of the BDBBA. Any basis produced using this case will have a collection of terminal nodes where some nodes will have had their coefficients zeroed (the zero bit quantizer), and other nodes will have had their coefficients unaltered from the values produced by the WPT. Therefore the *all-or-nothing* case can be used to reduce the bandwidth of the signal that the user must work

with during subsequent sound modifications and resynthesis. In our descriptions of the BDBBA in the next sections, any aspects that are dependent on the quantizer choice will assume use of this *all-or-nothing* case. We discuss other quantizer choices separately in Section *4.4.2.3 Quantizer Alternatives*.

The BDBBA offers two modes of operation. In one mode, the user only specifies a *quality budget* that determines what proportion of the resulting nodes use the high rate versus the low rate quantizer. This is specified as a ratio relative to the high rate quantizer bits[†]. The ratio is related to the resulting audio 'quality' of the basis, since a higher budget will result in less distortion. In the case of our default *all-or-nothing* quantization case, we refer to this mode as *bandwidth mode*, since the budget is equivalent to the bandwidth that is retained in the basis. The *quality budget* in this case is then also referred to as the *bandwidth budget*. The BDBBA finds the lowest distortion basis for the budget supplied.

The other mode of operation seeks to find a basis with a specified number of *high rate* terminal nodes. Both a terminal *node budget* and *quality budget* must be supplied. The algorithm finds the lowest distortion basis for the *node budget* supplied that is as close as possible to the *quality budget*. If no basis is possible with the specified number of nodes, the basis with the number of nodes closest to the node budget is used. In all cases, the basis chosen will have the minimum distortion possible for the *quality budget* specified. In the *all-or-nothing* quantization case, the number of *high rate* terminal nodes retained is equivalent to the number of nodes retained, since any *low rate* nodes are effectively not there due to their coefficients having been zeroed.

Several motivations exist for the particular approach to basis selection embodied by the BDBBA. We list several, focusing in particular on the *all-or-nothing* quantization case. Many of the same benefits will apply to other quantization scenarios as well.

First, allowing for the elimination of some data can make the signal more workable for the user in many ways. Imagine the real-time manipulation of subbands, such as in performing cross synthesis, gain adjust, or modulation operations. With two hands, one cannot be confronted with many more than two subbands to control if one's manipulations are to have a quick and definitive impact. So the user may want to work with only a small number of nodes; perhaps between two and four.

[†] As we describe later, we can talk about bits directly with our quantizers since we are not interested in bit rate reduction per se, and consequently need not consider *entropy coding* of the coefficients. For the same reason that we are not motivated by overall bit rate reduction, we also need only consider *scalar quantization* of the coefficients.

Second, the elimination of data can make resynthesis in real time faster and therefore more feasible to implement. Fewer narrow subbands mean less processing, both for imparting modifications on the sound and resynthesis[†]. Further, the real-time synthesis scenario may well impose more constraints on the amount of memory available, such as with an embedded *granular synthesis* application, thereby making data reduction attractive. In non-real-time, the abstraction offered by the new basis will speed the compositional process, since modifications can be focused exclusively on the remaining important subbands.

Third, the algorithm affords the user a measure of control. In an algorithm like the EBBA, there are no user input parameters, and therefore no control is possible. All bandwidth is kept, and the resulting number of nodes can not be influenced or chosen. If we value a small number of nodes, the EBBA may or may not produce them, and if so, they will have to be very wide since the approach is lossless. In the BDBBA, bandwidth and optionally the number of terminal nodes retained can be specified.

Fourth, by allowing deletion of less significant subbands, a basis that best fits the 'most important features' of the sound signal can be found. Conversely, constraining an algorithm to retain subbands with little energy during the basis selection process can inhibit the selection of a basis that 'best' represents the sound[††]. We may be seeking a type of abstraction of the sound to provide insight into its nature. Imagine the extreme case where what was cut out was the background noise in an environmental sound recording. It can also make it easier to parametrically model the sound, since the first step in modeling a sound usually involves understanding its time-frequency nature.

Finally, the use of data elimination, especially when aggressive small budgets are specified, can itself be used as a strategy for sound modification. Keeping all coefficients and modifying them is one type of distortion. Eliminating or quantizing subbands is another kind. If those distortions are sonically interesting, and especially if the user can build some intuition in using them, then they can be of value.

4.4.2.2 Description

The BDBBA uses the RDBBA of Ramchandran and Vetterli (1993) as its starting point. To avoid repetition, as well as to help delineate what is the same versus what is different between these two algorithms, we rely heavily on the description and theoretical background of the RDBBA given in *Section 3.7.2*, focusing here

---

[†] A narrow subband has less samples than a wide one.

[††] Note that those subbands deemed expendable as part of the basis selection can in fact still be retained after the basis has been selected, depending on the goals of the user.

on describing what is different. For completeness, we also provide in *Appendix B* a flowchart and cross referenced pseudo-code of the overall BDBBA algorithm.

Cost Function: The Bi-rate-Distortion Framework

We start with the cost function used by the RDBBA, repeating it here for convenience.

$$J(\lambda) = D + \lambda R \qquad \textbf{(65)}$$

Comparing with the generalized framework of (64), we see that $J_1 = D$ and $J_2 = R$.

The rate R used by a given node is a function of two things, the number of coefficients that need to be quantized at that node, and the number of bits per coefficient utilized by the chosen quantizer. We will find it convenient to separate these two parts of the rate. The number of coefficients that need to be quantized at a given node is proportional to the depth of the node. Because of the downsampling by two that accompanies each filtering operation, the number of coefficients is halved with each increase in depth. Therefore we can express the cost associated with the *j*th node at depth *i* in the tree as

$$J_j^i = D_j^i + \lambda(\tfrac{1}{2})^i R_j^i \qquad \textbf{(66)}$$

where R_j^i is the number of bits used to quantize a single coefficient of the *j*th node at depth *i*, and D_j^i is the distortion that results from using that quantizer. Note that in the formulation defined by (66), we have precluded the use of both *vector quantization* (VQ) and *entropy coding*. In the case of VQ, several coefficients are quantized at once, yet here we are working with single individual coefficients. In the case of *entropy coding*, rates are assigned based on their probability of occurrence, rather then there being a fixed rate per coefficient. The processes of quantization and bit assignment are separate in entropy coding. To include entropy coding in the formulation of (66), we could replace R_j^i with the average number of bits per coefficient after entropy coding was performed. However, since our motivation is not rate reduction, neither VQ or entropy coding are useful to us. They would merely add unnecessary complexity.

Let us move immediately to the *Bi-rate* case, where exactly two quantizers are available to choose from at each node. Let R_{Hi} be the rate associated with the highest rate quantizer and R_{Lo} be the rate associated with the lowest rate quantizer. Without loss of generality, we can replace the rate term R_j^i in (66) with the ratio of the quantizer rate used at that node to R_{Hi}. This modification will only change the value of the

Lagrange multiplier λ associated with the solution, but not the solution itself. As such, we have

$$J_j^i = D_j^i + \lambda(\frac{1}{2})^i (R_j^i / R_{\text{Hi}}) \quad \textbf{(67)}$$

Finally, if we include a weighting factor to account for the psychoacoustic significance of the energy in the frequency range associated with each node $(K_{psych})_j^i$ (see the sub-heading *Psychoacoustic Weighting of Distortion* later in this section), we obtain the equation

$$J_j^i = (K_{psych})_j^i D_j^i + \lambda(\frac{1}{2})^i (R_j^i / R_{\text{Hi}}) \quad \textbf{(68)}$$

With the formulation in (68) we can accommodate all possibilities of R_{Hi} and R_{Lo}.

In the *all-or-nothing* quantization case described earlier, R_{Hi} is a sufficient number of bits such that quantization with that number of bits results in no distortion. We call this rate $R_{\text{No Dist.}}$, and its use at node j results in a cost contribution from that node of

$$J_j^i = 0 + \lambda(\frac{1}{2})^i (R_{\text{No Dist.}} / R_{\text{No Dist.}}) = \lambda(\frac{1}{2})^i \quad \textbf{(69)}$$

This is simply λ times the relative bandwidth of that node, where we define the node bandwidth as its fractional width. For example, a *depth-4* subband has a fractional width of 1/16 or 0.0625†.

Continuing with the *all-or-nothing* quantization case, if we assign zero bits to our second quantizer R_{Lo}, then when that quantizer is used the resulting distortion is simply the accumulated energy of those coefficients††, weighted by $(K_{psych})_j^i$. Rate in that case adds nothing to the node cost since R_{Lo} is zero. As such, we can replace the ratio R_j^i / R_{Hi} in (68) with the binary variable $keep_j^i$, which is zero when we do not keep the coefficients of the node (i.e., quantize with the zero bits of R_{Lo}), or one when we do keep the coefficients of the node (i.e., quantize with the $R_{\text{No Dist.}}$ bits of R_{Hi}). Therefore we can re-write the cost as

---

†. Note that *bandwidth* as defined above is not identical with the fraction of the frequency spectrum that the subband may occupy. While they would be identical with perfect brickwall analysis and resynthesis filters, this can never be the case in practice.

††. Ideally this requires the use of an orthogonal basis, which we normally use.

$$J_j^i = (K_{psych})_j^i \times E_j^i \times (1 - keep_j^i) + \lambda(\frac{1}{2})^i \times keep_j^i \quad \textbf{(70)}$$

In this paradigm, we can think of trading off bandwidth with distortion. The budget we specify is the fraction of the signal bandwidth we wish to retain. The total bandwidth of the original signal is equal to one, and each terminal node that is kept (*keep* =1) contributes its fractional bandwidth. The Lagrange multiplier λ is modified via a bisection algorithm until the budget is satisfied. Large λ values result in a high cost for keeping subbands, so that relatively fewer subbands will be kept and the distortion associated with that basis will be relatively large. Small λ values will favour keeping subbands, and the corresponding basis will result in relatively less distortion of the signal.

In the remainder of this thesis, unless specified otherwise, use of the term BDBBA will be synonymous with this *all-or-nothing* quantization approach. When we do look at other quantization alternatives for the general *bi-rate* scheme, such as in *Section 4.4.2.3*, we will make the new quantization choice clear.

Pruning the Tree

The *all-or-nothing* form of quantization motivates a slight but important change to a detail of the original RDBBA approach. In *Section 3.7.2* we state regarding the pruning operation that '*If the cost of the parent is less, then the subtrees associated with each of the child nodes are pruned back to the parent*. *Otherwise, the node is designated as a split node and*...'[†]. Since the RDBBA would typically employ a least a few possible quantizers at each node, none of which would quantize with zero bits or a number of bits that results in no distortion, it would be rare that the parent cost is identical to the sum of the child costs. When such equality does occur, it is clear that the parent node will split. But it is not terribly important which occurs, since it is a rare event. It will also not effect higher layer (lower depth) decisions about splitting, since the cost associated with the parent is the same in either case.

Things are quite different in the case of the BDBBA when *all-or-nothing* quantization is employed. The sum of the child node costs can only ever be less than that of the parent node cost if one of the child nodes is not kept, independent of λ. And in that case, they will always be less and the split must always occur. We first show why either keeping or deleting both child nodes results in equal costs between the sum of the two child nodes and the parent node.

[†] In Ramchandran and Vetterli (1993), this is stated by the line '*if* $J_j^i(\lambda) < \tilde{J}_{2j-1}^{i+1}(\lambda) + \tilde{J}_{2j}^{i+1}(\lambda)$,' on p. 168 of that reference.

Start by assuming that both child nodes contribute the lowest cost for the given λ if they are kept. From (69) we see that each of the child nodes will have bandwidth $1/2^i$, which when summed together gives a bandwidth of $1/2^{i-1}$. But this is the bandwidth of the parent node, which is at level *i*-1, and is not surprising since we already know that the parent covers the same bandwidth as that covered by its two child nodes. Since all nodes operate at the same operating slope λ, the costs due to bandwidth will be identical for the parent versus the sum of the children if they are all kept.

Since the transform is orthonormal, energy is conserved. Therefore the two child nodes collectively possess the same energy as the parent node. We have assumed both child nodes are kept at this stage. Therefore distortion must be zero and makes no contribution to the cost defined in (70). What then is implied regarding the parent node? If it is kept, the distortion component is also zero, and all costs are due to bandwidth. In that case, as stated in the previous paragraph, costs will be identical. If instead we consider that the parent node may not have been kept, then the cost of the parent is equal to the energy of its coefficients. But if it is cheaper to not keep the parent, then it must also be cheaper to not keep the children, since due to orthogonality, the sum of the child energies equals that of the parent, and the costs of keeping are identical when both children are kept. In other words, if it is lowest cost to treat both sibling child nodes identically and either keep them both or discard them both, the cost of the parent will always be the same as that of the sum of those child nodes.

As such, the case of equality occurs frequently, and must be handled to best meet our goals. Since we generally want to reduce the number of nodes to some reasonable number, it makes more sense to not split when costs are equal. We want the algorithm to highlight differences in energy between subbands, and split when that occurs. The BDBBA is motivated to split in such cases, given an appropriate bandwidth budget, since only then will there be a cost saving.

Now we must show why the keeping of a single child node will always result in splitting. Imagine the scenario where a parent splits into two subbands, high and low frequency, and the energy is all in the lowpass subband. Therefore the highpass node can be deleted, and since there is no energy in it, it adds no cost. If the cost of keeping is greater than of not keeping, then the lowpass subband is also deleted, and we are back to the scenario given above (no split occurs). However, if the cost of keeping the lowpass subband is less, then the cost of the sum of the child nodes would be only due to the bandwidth of the one child. If it were cheapest at the parent level to keep (which will depend on the particular value of λ), then the sum of the child node

cost would be 1/2 that of the parent, and the split would occur. If it were cheaper to delete the parent node, then the child cost would again be cheaper, since we know by virtue of the child node being kept that the cost ($\lambda\ 1/2^i$) is less than the total distortion (which is the same in the remaining child node as in the parent) and therefore less than the parent. Again, the split must occur[†]. When we discuss the inclusion of psychoacoustic weighting to the distortion calculation, we'll see that it makes the probability of equality much lower. However, this *split* decision rule in the algorithm can remain the same.

We should emphasize that this decision to split or not split only effects the shape of the final tree we carve out. All cost calculations are independent of this decision rule regarding splitting. The cost is the same for each node and tree as a whole in either case, since we are deviating only in the case of equality, where the parent node cost is equal to the sum of the child costs. The only difference is what the tree looks like, and consequently what subbands we have access to when working with the signal.

One additional distinction between our approach and that described in Ramchandran and Vetterli (1993) concerns the so-called *carving out* of the optimal subtree. This *carving out* process involves forming the optimal tree and corresponding data structure based on each node's *split* data element that was determined in the pruning process. While Ramchandran and Vetterli describe doing this for each λ tried, we only do so after the solution is found. We know whether the current λ corresponds to the solution (i.e., meets the bandwidth budget) based on the bandwidth associated with the root node, since this is the sum of all terminal node bandwidths. This root node bandwidth information is available without explicitly carving out the tree, so we forego this step until a solution is found.

Psychoacoustic Weighting of Distortion

In (70) we have included a weighting factor $(K_{psych})_j^i$ to account for the human ear's perception of loudness as a function of frequency. As discussed in Section *2.3.1 Psychoacoustics*, the human auditory system loudness response is frequency dependent. We can account for this in the algorithm by weighting the distortion added as a result of deleting a node by the inverse of the *equal loudness curve* at a frequency that corresponds to the frequency coverage of the node.

In particular, our implementation uses as its reference the *60 phon* curve from Fig.*2-4*. This corresponds

---

[†] The only exception would be the very rare case when the energy of the remaining subband is precisely equal to the cost associated with keeping it. In that case, since costs between parent and child nodes is equal, no split would occur.

to the curve with a sound intensity level of 60 dB at 1 kHz, where 60 phons is the loudness associated with 'normal conversation' (Truax 1999)†.

In order to use this curve to weight the distortion function meaningfully, it must first be manipulated. Looking back at the 60 phon curve of Fig.*2-4*, we see that a sound must be 10 dB stronger at 8 kHz than at 1 kHz (70 dB versus 60 dB respectively) to be perceived as equal in loudness. Since 10 dB equals a power ratio of 10, when we calculate the distortion contributed by deleting a subband at 8 kHz, we should weight it by a factor of one-tenth relative to the distortion contributed by the deletion of a subband at 1 kHz. As another example, at roughly 4 kHz the signal will sound as loud as at 1 kHz when it is 3 dB lower in intensity (57 dB). Since 3 dB is equivalent to a factor of two in power ($10\log(2/1) = 3$ dB), the distortion contributed by deleting a subband at 4 kHz should be weighted twice as much as that at 1 kHz. So our desired weighting curve should be formed by converting the dB values of the *equal loudness curve* to linear values, taking their inverse, and optionally applying a normalization across the entire resulting curve. Fig.*4-9* illustrates the result of this operation. In this case, the weights were normalized by the average weight. Inspection of the curve shows that the weight at 8 kHz is indeed roughly one-tenth that at 1 kHz (0.2 versus 2), and at 4 kHz it is roughly double that at 1Khz (4 versus 2).

While Fig.*4-9* shows the general shape of the weighting curve used, there is still the practical matter of how to construct it for different sample rates and transform depths. We proceeded as follows. The initial curve as in Fig.*4-9* simply has enough points to describe the relationship, at least up to one half the maximum sample frequency encountered. In general, these points are not equally spaced. If anything the distance might be constant with the log of frequency to represent the sound well, though this is not a requirement. The number and location of points that are needed for a particular basis selection depends on the depth of the transform and the sample rate. We want one representative weighting point for each node of the complete tree (terminal nodes and internal nodes, since the basis has not been chosen) up to the maximum frequency subband whose upper frequency subband edge is at one half the sample frequency. So we resample the initial curve to obtain a number of points equal to the number of full depth nodes (it will have the same shape after this, unless our sample rate (number of nodes) is too low, in which case averaging

---

†. *Sound Intensity* in decibels is based on the ratio of the intensity of the signal (Watts/m^2) to the intensity of a signal at the threshold of hearing (10^{-12} Watts/m^2). Therefore $Intensity = 10\log\left(\frac{I_s}{10^{-12}}\right)$ dB, where I_s is the signal intensity, and the logarithm is *base 10*.

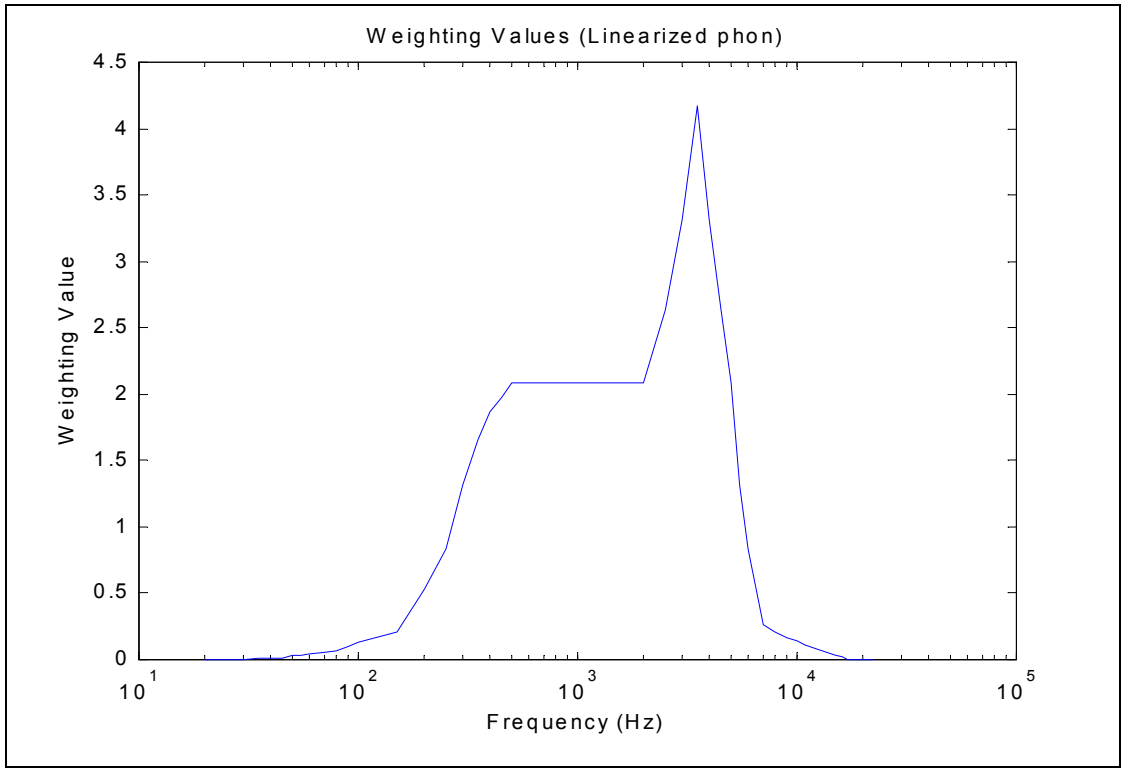

Figure 4-9: Psychoacoustic Based Weighting Values

occurs via the interpolation. Then for each level up above the full depth nodes for which we already have weights, paired sibling nodes are averaged to arrive at a parent node weight. In this way, the sum is the same†.

We said earlier that the inclusion of weighting will reduce the probability of equality between the sum of child costs and the parent cost. It is unlikely that two sibling nodes will have the identical weights. The weighted sum of the child energies may not be that of the weighted parent. Depending if the energy splits such that more energy goes in the subband with the higher weight or lower weight, the sum of the weighted child distortions could be more or less than that of the parent, respectively. Imagine the extreme case where the weight of one child is zero, and of the other twice that of the parent, which it would have to be since the weight curves are constructed through averaging. Further imagine that all the energy goes into the band with zero weight, which is not likely but illustrates a point. Then both child bands can be deleted with no cost,

---

† One might also consider interpolating for all levels (not just the full depth level as described) separately, and then setting the mean of the curve for each level to be the same. However, in this case it is possible for a parent node weight to be greater than sum of its child weights, in which case the sum of child costs can be less even with no split occurs. This often leads to erroneous results.

which will have to be less than the best cost of the parent. In that case, a split would occur, and a 'don't keep' condition for the child node coefficients. While this seems somewhat paradoxical, it is easily dealt with. We simply prune the nodes since they are of no consequence, but make sure the cost associated with the parent is that of the sum of the children, since they are lower cost. Note also that this rather extreme case is unlikely.

An Averaged Best Basis

We would like a stable single best basis over the duration of a given sound signal. In the case of the RDBBA, *M* signal blocks of size *N* are coded independently. Each block has its own basis and tree associated with it, such that the sum of the rates of the *M* blocks meets the overall rate budget. The total distortion, which is the sum of the distortions across all terminal nodes of each of the *M* trees, will be the minimum possible for that budget.

To achieve the desired single best basis in the BDBBA, we determine an average best basis over all *M* blocks within the signal. The cost associated with a given node is based on the associated distortion and bandwidth across all of those *M* blocks. For example, if a given node is kept (*keep* =1), then it is kept across all blocks, and the bandwidth of that node is added to the cost functional. If the node is not kept (*keep* =0), then the distortion added to the signal by that node, which is the summation of the distortion values from each block for that node, is added to the cost functional.

This averaging approach versus the changing best bases of the RDBBA illustrates the different goals of data compression versus signal representation. Averaging would not lead to optimal compression since it does not adapt to the changes of the signal. However, since we want some type of abstraction or simplification of the signal, the averaging is generally what we want. An averaging approach is also less computationally intensive, since we must only find a best basis for each λ once, not for *M* separate blocks. We note that the EBBA also appears to employ such an averaging approach (*Section 3.7.1*).

Iterating Towards a Solution with Bisection

Thus far we have described finding a best tree for a given operating slope λ. We must then find the optimal λ through multiple invocations of finding the best tree for a given λ. We achieve this using the *bisection* approach described for the RDBBA in *Section 3.7.2*, which involves the *outer maximization* (*step 3*) of expression (57) in that section. We can justify using the same approach here since the BDBBA may be viewed as a special case of the RDBBA, where specifically two quantizers are used. As such its *Bandwidth-*

Distortion characteristic will also be convex and bisection can be used to obtain a fast solution. Fig.*4-10* shows a typical *Bandwidth-Distortion* curve for the BDBBA†.

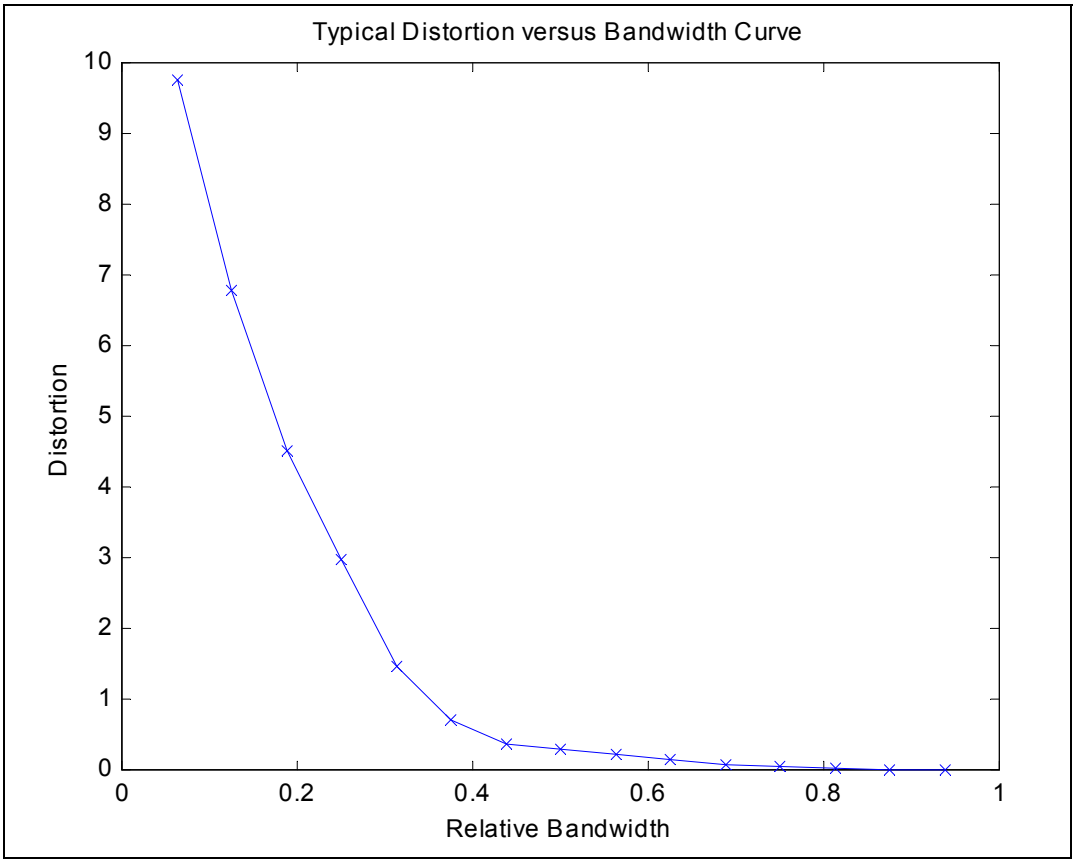

Figure 4-10: Typical Bandwidth-Distortion Curve for the BDBBA

Given that the approaches are the same with *rate* replacing *bandwidth*, we need not discuss the algorithm further here. However, we do elaborate on one detail of the process, namely that of selecting the initial lower and upper Lagrange multiplier estimates, λ_l and λ_u. As Ramchandran and Vetterli (1993) point out, "... the choice of a good initial operating point is the key to a fast convergence".

The approach we take depends on whether the *bandwidth* or *node* mode of the algorithm is invoked. In bandwidth mode we start with values that are quite likely to surround the solution operating point, but are not guaranteed to. That way, we get some benefit when they do surround the solution in that since the initial

† This curve is based on the *Stairs1* sound (*Section 5.3*) transformed to *depth-4* using the *sym5* filter (*Section 5.2*).

values need not be adjusted, and since they surround the solution more tightly, few iterations are required. When one of the end points is not acceptable, the algorithm adjusts the value during initialization until it is. This approach realizes some benefit in a tight initial estimate, but is still without risk since the values can be modified until they are acceptable. In node mode, a history of solutions for the bandwidth budgets tried in the process of finding a solution for the node budget is accumulated, and the values for the initial upper and lower points are selected from this history. We now elaborate on the bandwidth mode approach, and describe the node mode approach in our description of node mode later.

We have used the values 0.5 and 10 as our initial bandwidth mode estimates for λ_l and λ_u. If the resulting rates, or bandwidths in this BDBBA case, are such that inequality (58) is satisfied, then they stay as they are. If one or both sides of the inequality do not hold, then the values are changed by a factor of 10 until the inequality is satisfied. For example, if $R_{budget} > \sum_i R_i^*(\lambda_l)$, then λ_l is decreased by a factor of 10, the associated basis is determined, and the inequality is re-checked.

While the values for λ_l and λ_u are easily changed, we chose them because they seemed like good compromise values. In the test cases *Stairs1BDBBA1- Stairs1BDBBA15* described in *Table 5.13* of the next chapter, one can see that this range covers bandwidth budgets between 37.5 - 68.75%. Furthermore, our testing showed that having a much larger range means more iterations, and we should not have to suffer a large number of iterations for all budgets tried.

This completes our description of the core operation of the BDBBA when using the default *all-or-nothing* quantization case. This description is also applicable for the most part to other quantizer scenarios. In *Section 4.4.2.3* we describe those scenarios and how they do differ from this default case[†]. First, however, we describe a mode of operation we refer to as *Node Mode*, as well an option for lossless operation with the algorithm.

[†] We note that the *all-or-nothing* version of the BDBBA could be implemented in a simpler fashion, though this is not the case for other quantizer choices. Such a simpler implementation would involve creating a ranked list of the subbands of the complete tree from most to least psychoacoustically weighted energy, and then selecting a number of subbands equal to the bandwidth budget from the top of this ranked list going down. The tree would then be built such that nodes would only split where needed to reach the selected subbands. Any sibling subbands that are both part of the basis would be merged.This should give the same result as the BDBBA, since the BDBBA produces the minimum distortion solution, which the approach just described also does.

Node Mode

The core BDBBA algorithm described thus far is controlled by a *bandwidth* or *quality budget*. The number of terminal nodes in the resulting tree that this budget produces is not predictable. Essentially, the number of terminal nodes and the corresponding bandwidth that they occupy are not proportional, and no monotonic relationship between the two exists. In fact, there is not even a one-to-one relationship between leaves and bandwidth (or distortion). For example, both a large and a small bandwidth budget could produce a three leaf tree, since the leaves in these two cases can have different bandwidths.

In *node mode*, an outer *node loop* is wrapped around the *bandwidth loop* associated with the core bandwidth-distortion driven algorithm described so far. This *node loop* is driven by a *node budget*, and operates by invoking the *bandwidth loop* with an ordered series of *bandwidth budgets* until either a basis is found that meets the node budget or the list of bandwidth budgets has been exhausted. In the latter case, the basis that had the closest number of nodes to the node budget is chosen. In all cases the basis will have the minimum distortion possible for the bandwidth retained[†]. The *ordered series of bandwidth budgets* is arranged such that the budgets move progressively further away in magnitude from the bandwidth budget originally specified by the user. This ensures that the final solution has a bandwidth budget as close as possible to the original budget specified, even though the node budget takes higher precedence.

This search is implemented efficiently by retaining the Bandwidth-Distortion results associated with each bandwidth loop result, so that the bisection process uses end points that tightly surround the new budget values. The algorithm finds the basis for the specified bandwidth budget, and then compares the resulting number of nodes with the node budget. If they are the same, the process is complete. Otherwise repeated iterations of the bandwidth loop using the ordered budget list as described above occurs[††].

Upon entry to node mode, we select initial λ_l and λ_u values of zero and infinity respectively. This selection ensures that any budgets within the range of possible budgets will fall within the maximum and minimum λ values that start the bisection process for the first bandwidth budget we try. We gain bisection efficiency in node mode by recording in a table all solutions for each bandwidth budget tried during the node mode search. The values of zero and infinity occupy the end points of the table. Each new bandwidth budget

---

[†] There may not even be a basis with the number of nodes specified. By choosing the basis with the number of nodes closest to the node budget when no solution exists, the algorithm presents a best effort solution.

[††] Note that since no one-to-one leaf-to-bandwidth relationship exists, the algorithm does not *converge* to a solution. Rather it seeks a solution that comes closest to the node budget requirement.

we try adds an entry to the table, and creates another potential λ_l or λ_u for the next invocation of the bandwidth loop to use. As the node mode search for a solution proceeds, the table grows, and consequently the initial values of λ_l and λ_u for each successive bandwidth loop invocation surround the solution λ for that invocation ever more tightly.

Lossless Operation

Sometimes we may not want to incur any distortion in the signal. At the same time, we want our basis structured in a way that has delineated and highlighted the key subbands. In this case, we can use the BDBBA to find a best basis and highlight to the user those subbands that were selected as part of that basis, such as via an asterisk in the user interface, and then not delete the coefficients associated with the subbands that were not selected as part of the basis. The user now has a basis that is in some way tailored to the signal. By knowing which subbands are part of the basis, the user can work with those subbands on the premise that they are the most important to the sound. The others simply fill in the sound, but can be ignored by the user when imparting modifications. We refer to this as *lossless* operation of the BDBBA.

What then is the justification for specifying a particular bandwidth, and optional node budget, if all subbands will be kept? It is simply that the budget indicates what portion of the signal, in terms of bandwidth or number of terminal nodes, the user wants to work with. The user can focus on these nodes exclusively, with the knowledge that the others are there to round out the sound.

Use of this mode of operation can influence the choice of bandwidth budget, since regardless of the budget chosen, all of the sound will still be retained[†]. One might, for example, want to focus on just a small part of the signal bandwidth. As in all cases, the BDBBA will retain the most significant parts of the spectrum for the assigned budget, though the basis itself may say little about the signal.

Use of *lossless* operation in node mode is perhaps more intuitive. We want a basis with a certain number of terminal nodes, where those nodes represent the most important part of the signal spectrum. We can then work with these nodes in our sound manipulations, yet keep the other nodes to fill in the sound.

[†] In our GUI, we still have the option on playback to include or not include the subbands that are not strictly part of the basis (i.e., that would have been deleted without lossless operation).

4.4.2.3 Quantizer Alternatives

Now we consider briefly other quantizer choices for the BDBBA. Thus far we have focused primarily on one of many possible configurations of the algorithm, which used the *all-or-nothing* quantization choice. In that case, we employed the two extremes of zero bits for the low rate quantizer, and enough bits for no distortion for the high rate quantizer. One advantage of this approach is it makes the interpretation of the resulting basis straightforward. It finds the lowest distortion basis for the bandwidth budget provided. In Node mode, it finds the lowest distortion basis that meets the node budget and is as close as possible to the specified bandwidth budget. However, the BDBBA readily supports any other integer quantizer choices between the two extremes of the *all-or-nothing* case, and the foregoing description of the BDBBA in *Section 4.4.2.2* is for the most part applicable to these other cases.

Other quantizers may well lead to useful bases, and in that case the range of basis selection options can be expanded. Once the basis is selected, we can again either choose to use the coefficients in their quantized form, or restore them to their pre-quantized values. The latter case corresponds to the so-called *lossless* operation described earlier. We might restore the coefficients if we simply seek a suitable basis with no distortion. We may preserve their quantization if we are interested in the resulting sound modification such quantization imparts, if we feel the quantization has provided a useful simplification or abstraction of the sound, or if our implementation can take advantage of the data reduction in some way, such as by achieving faster resynthesis or by requiring less memory.

Recall the general form of the BDBBA cost function as defined in (68). It is repeated here for convenience:

$$J_j^i = (K_{psych})_j^i D_j^i + \lambda(\frac{1}{2})^i (R_j^i / R_{\mathrm{Hi}})$$

As an example, we could set R_{Hi} to 12 bits and R_{Lo} to 4 bits. Then R_j^i would take on one of these values, and D_j^i would be the corresponding quantization distortion†.

There are several possible ways to do the quantization. We chose to use *mid-tread* quantization, since it supports a zero value as one of the quantizer outputs. Quantizer levels are uniformly spaced and can be based

† Recall from *Section 4.4.2.2* that since our motivation in basis formation and quantizer selection is not specifically rate reduction, there is no need to consider entropy coding of the coefficients. Furthermore, even if we might value the reduced data set quantization can provide, the added complexities of entropy coding would work against the processing improvements that the smaller quantized data set would bring.

on either a global peak across all subbands or on a per subband basis.

We note several differences relative to the *all-or-nothing* case. First, the cost calculations are more complex. It is entirely possible, for example, that the low rate quantizers offer the lowest cost when applied to two sibling nodes, whereas the high rate quantizer provides the lowest cost for the corresponding parent node. What actually happens depends on the 'fit' of each quantizer to the coefficients of the given node and the resulting distortion, versus the number of bits used by each quantizer. The *all-or-nothing* case costs were simply based on the energy of the node coefficients and the fractional bandwidth of the node.

Second, when we choose to retain the quantizations once the basis is selected, the resulting sound will often possess quite different, and possibly interesting artifacts. Due to the rounding operation of quantization, energy will not only be removed from the signal (as in the *all-or-nothing* case), it can also be added. Given that both quantizer rates can be selected, many possibilities exist.

Third, the minimum quality budget is not zero. For example, in the case where we have set $R_{\mathrm{Hi}} = 12$ bits and $R_{\mathrm{Lo}} = 4$ bits, the minimum budget would be $4/12 = 1/3$, since this would be the fraction of the peak rate used if all nodes were quantized with the low rate quantizer. In the *all-or-nothing* case, where we have referred to the *quality budget* as the *bandwidth budget*, since R_{Lo} is effectively zero bits, the lowest budget is always zero. The maximum budget will always be one, as before, since our budget is defined as a ratio relative to the high rate quantizer.

Finally, the case of equal costs between a parent mode and its sibling child nodes is far less likely. However, the rule we use to split a node only if the parent cost is greater is still acceptable in this case.

4.4.3 Best Basis Cost Function Alternatives

4.4.3.1 Information Cost Based Schemes

Recall in Section *4.4.1 Best Basis Algorithm Framework* that we provided a general framework within which to view *Best Basis Algorithms*. Regarding *information cost* based approaches where we add no distortion to the signal (*lossless*), we briefly discuss three possibilities.

The first such cost function we have already considered in some detail. This is the EBBA described in *Section 3.7.1*, where the *Shannon Entropy* functional is applied to coefficient amplitudes. Other information cost measures can be applied to the coefficient amplitudes, however, and Wickerhauser (1991) lists three in addition to the Shannon Entropy cost. One of these is a threshold related cost that we refer to here as the

Threshold Best Basis Algorithm (TBBA). In this scheme, a threshold value *t* is set, and the number of elements for a given node that are above *t* are counted and added to the cost. Everything below *t* is considered unimportant. This approach is motivated by the compression application, where coefficient values below *t* could be deleted and not sent, thereby justifying not adding them in to the cost measure.

How could such a scheme be useful to us? We can imagine setting a low threshold *t*. If the splitting of a given node into its child nodes is such that most of the energy turns up in just one of the nodes, then the other node should have many zero values, or at least many low values. This case will result in the sum of the child costs being less than the parent cost, and will encourage a split in the tree. Such a measure values concentration of energy, as does the Shannon Entropy cost. However, in this case the user is given some control. By setting the threshold larger, possibly less of an energy differential between subbands is required to cause a split, and the threshold could perhaps be used to determine the nature of the resulting tree.

Finally, we might consider using real entropy as a cost. A number of quantization levels would need to be chosen, and then the coefficients would be assigned to their corresponding quantization level based bins. Probabilities would then calculated and the overall entropy for each subband would be determined. Again, this would seem to make sense when energy exists primarily in one of two child subbands, since the other will have many zeros, and a correspondingly lower total entropy for the child subbands versus the parent would occur.

4.4.3.2 Model Based Schemes

Here we speculate briefly on a model based cost functional, where we attempt to model the coefficient streams in each subband. This could help us better understand a given sound, and may help in the creation of a simplified model for the sound.

The model considered here is that of the *forward predictor*. Implicit in this is that the coefficient streams can be modeled as an auto-regressive (AR) process. The degree to which they match such a model will determine the size of the prediction error. The proposed cost function trades-off prediction error power with predictor order. This is a real cost, since order is related to the amount of processing required and prediction error power is related to the *closeness-of-fit* of the model. Prediction order also reflects the complexity of the model, and we seek the simplest model possible. We could alternatively trade-off prediction error power with the rate associated with encoding the predictor. In that case, we would need to encode both the prediction error filter coefficient values, and the prediction error itself.

For completeness we show what the cost functional might look like. Starting with the most general form of the cost function given in (64), we substitute weighted prediction error power (P_{PredErr})and filter order (O_{Pred})to obtain

$$J_j^i = (K_{psych})_j^i \times P_{\text{PredErr}} + \lambda(\frac{1}{2})^i O_{\text{Pred}} \quad \textbf{(71)}$$

As with all the other best basis schemes, we would then prune the tree by minimizing this functional. As with the BDBBA, multiple values of λ will likely need to be tried before a solution meeting the budget is found. Also, since the suggested scheme has not been pursued further, but is rather an example of what might be possible, we have not pursued the mathematical characteristics of such a potential scheme. For example, it may be that the relationship between order and prediction error is not monotonic for many subband coefficient streams. If it is not monotonic, this might preclude a fast algorithm such as the bisection used in the BDBBA, and a slower method may be required.

4.5 Complexity

In this section we touch on some aspects of computational complexity for the transforms and basis selection algorithms we have considered.

4.5.1 Complexity of the Transforms

The fast WPT (FWPT) is an $O(N\log N)$ operation, where N is the number of samples applied to the transform or block length. It is implicit in the definition of block length that it will be transformed to a maximum depth of $\log_2 N$. Longer signals are blocked into M blocks of length N. We also note that in the case of orthogonal filter banks, which we are primarily interested in here, the inverse transforms will require exactly the same number of computations. For the biorthogonal case, the order of operations will be the same for analysis and synthesis, however, the number computations will differ in proportion to the difference in analysis and synthesis filter lengths.

It is instructive to see where the $N\log N$ number comes from. Consider the application of the simple two coefficient *Haar* filter based WPT[†] on a signal of length $N = 8$. The tree for this was shown earlier in Fig.*3-6*. We start with the initial first level filtering of the samples. The highpass and lowpass filter operations are

[†] The WPT with the *Haar* filter is also known as the *Walsh-Hadamard* transform.

initiated every second sample due to downsampling by 2, so that $N/2$ lowpass filter operations and $N/2$ highpass operations each are performed. This results in a total of *N* filter operations at the first level. Since the *Haar* filter lowpass coefficients are $(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$, and the highpass coefficients are $(\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}})$, each filter operation involves two computations† or multiply-adds, even though in this special case there is only one add, and the multiplies can be eliminated on either the WPT or inverse WPT if a constant overall scaling of energy in the transform domain is allowed. So we incur $2N = 16$ computations at the first level.

At the next level, each of the two filter outputs from the previous level has *N/2* points. These are each split by highpass and lowpass filters again. For each of these there are *N/4* filter operations ((*N/2)/2*), giving a total of another *(N/4)*4 = N* operations. Again each of these involves two computations, for another $2N = 16$ computations. This pattern repeats to full depth. At each new level we have twice the number of filter outputs with half the number of samples, resulting in a constant *N* filter operations (or *2N* computations for the *Haar* case) at each level. Since the number of levels for the full depth transform is log_2N, the total filter operations are $N\ log_2N$, and the total computations for this *Haar* case is $2N\ log_2N = 2*8*3 = 48$.

Note that this splitting of a signal into highpass and lowpass outputs and correspondingly downsampling by two results in a *critically sampled* transform. This downsampling is especially important in data compression applications, where the transform itself should not cause any overall data expansion. Furthermore, the fact that the highpass and lowpass filter operations are only initiated every second sample keeps the computation count down. Such downsampling before filtering can be implemented using the so-called *polyphase* form. A less efficient implementation would apply the filter at each sample point, and then downsample by two, producing the same result in twice the number of operations.

The log function is *base 2* because the discrete WPTs levels are related by factors of two due to the two channel filtering and corresponding downsampling. If we replace the two coefficient *Haar* filter with a filter of length *L*, then the number of computations becomes $LN\ log_2N$. The order is still $Nlog_2N$, since *L* is a multiplicative factor, albeit a possibly significant one when *N* is not large relative to *L*.

---

†. A *computation* in this context implies a maximum of one addition plus one multiplication, and is sometimes called a 'multiply-add'. Sometimes the multiplication is not required, such as when one of the multiplicands is unity.Also, sometimes the addition is not required, such as in the first multiplication of a longer multiply-add. Most DSP chips have a multiply-accumulate function that combines these two operations efficiently, further justifying treating them as a single 'computation'.

For comparison purposes, the FWT achieves, as Strang and Nguyen (1997) put it, the *Holy Grail* of complexity theory, requiring *O(N)* computations (see *Appendix A* for a derivation). Specifically, *2LN* computations are required. The FFT is an $O(N \log_2 N)$ computation, just like the FWPT. It requires $2N \log_2 N$ computations, which is identical to the *Haar*-based FWPT. What we gain for the added computations in going to longer filters than the *Haar* is not only better isolation between bands, but also smoother transitions between blocks, thereby avoiding some of the blocking effects that can occur with transforms such as the FFT (see the section titled *Overlap to Avoid Blocking versus Added Complexity* in *Appendix A* for more details on this).

Table 4.1 summarizes the complexities of the three transforms.

Table 4.1 Comparison of Transform Complexities[1]

Transform	Order	Computations
WT	N	$2LN$
WPT	$N \log_2 N$	$LN \log_2 N$
FFT	$N \log_2 N$	$2N \log_2 N$

1. N = transform block size. L = filter length.

4.5.2 Complexity of Basis Selection

Having discussed the complexity of the WPT and related transforms, we now consider the best basis selection approaches, focusing on the BDBBA in its *all-or-nothing* quantization form. The fixed basis cases such as the PTOB are not discussed, since their formation is a straightforward process of joining and splitting certain nodes[†].

Coifman and Wickerhauser (1992) state that the complexity of the *EBBA* search is proportional to the number of nodes in the tree, or *O(N)*. For each non-terminal node of the original tree, we decide whether the node itself, or the sum of its children nodes has the lowest cost. Since there are N -1 non-terminal nodes in the tree, the operation is *O(N)*. Each operation is comprised simply of an addition (of the sibling child nodes) and a compare.

In the case of the RDBBA and BDBBA, for each non-terminal node of the original tree, we decide

[†] In fact they could be viewed (and implemented) as part of the transform itself.

whether the node itself, or the sum of its children nodes has the lowest cost. Given the *N-1* non-terminal nodes of the tree, there are again *O(N)* operations. However, as we will see, there is a larger multiplicative factor on *N*, resulting in more computations. As with the EBBA there is also a one-time computation that occurs before the search, in this case, of the distortion associated with each possible quantizer.

For each node we have multiple potential quantizers. In the BDBBA there are two. For each one, we must calculate the cost given the current Lagrange multiplier λ. For Q quantizers, this adds $2Q$ multiply-add computations, where the two multiply-adds per quantizer are due to calculation of the cost, such as the cost function in (68).

Further, achieving convergence of the bisection algorithm will require some *I* iterations (one or more) to satisfy the budget. Therefore the computations will be *I*2Q*(N-1)*. This computation count does not include the determination of the minimum cost per node across the candidate quantizers. Recall also from *Section 4.4.2.2* that, unlike Ramchandran and Vetterli (1993) in their RDBBA description, we do not carve out the optimal tree for each Lagrange multiplier λ attempted. Instead we wait until the solution that meets the budget has been found, and then carve out the tree.

When operating in Node Mode, we can incur another *K* iterations of the overall BDBBA, bringing the total to *K*I*2Q*(N-1)* computations.

As we stated earlier, to keep the overall computations low we should try to minimize the number of iterations *I* of the bisection process, and in the case of Node Mode, the number of iterations of the core BDBBA bandwidth loop. For the former, this involves choosing good upper and lower bounds for λ in the bisection algorithm. For minimizing the number of core bandwidth loops invoked while in Node Mode, we can store key information during each loop to shorten the overall process. We treat both these issues in more detail below in the subsection titled *BW and Node Loop Iterations (BDBBA)*.

To gain some perspective on the relative efficiency of the BDBBA/RDBBA algorithms, we compare with a simple *brute force* approach to finding the lowest distortion basis for a given bandwidth or rate budget. Coifman and Wickerhauser (1992) estimate that the number of bases for an N sample transform is at least 2^N. The average tree across this set of possible bases (or trees) will have $N/2$ terminal nodes, or half way between no nodes and the N terminal nodes of the complete tree. If each node can use one of Q possible quantizers, we have $Q^{N/2}$ possibilities per tree, and 2^Nx $Q^{N/2}$ total tree-quantizer possibilities to try. Using the rather

modest numbers of N=16 and Q =3, this amounts to roughly 430 million cases to check to find the one with the lowest distortion for the target rate budget. Then for each tree we must add up the distortion and rate separately, over an average again of $N/2$ terminal nodes as well. The resulting order and number of computations are equal to $N * 2^N * Q^{N/2}$.

Table 4.2 summarizes the complexity relationships we have described for basis selection.

Table 4.2 Comparison of Basis Selection Method Complexities[1]

Basis Selection Method	Order	Computations
EBBA	N	*N-1*
RDBBA	N	*I*2Q*(N-1)*
BDBBA - Bandwidth Mode	N	*I*2Q*(N-1)*
BDBBA - Node Mode	N	*K*I*2Q*(N-1)*
Brute Force Rate (for Optimal Bandwidth -Distortion Basis)	$N * 2^N * Q^{N/2}$	$N * 2^N * Q^{N/2}$

1.N = transform block size. Q = number of quantizers per node. I = number of bisection algorithm iterations. K = number of *Bandwidth Mode* iterations.

BW and Node Loop Iterations (BDBBA)

Our approach to selecting the initial values for λ_l and λ_u was described in *Section 4.4.2*. The *Bandwidth Mode* case was described in the subsection *Iterating Towards a Solution with Bisection*, and the *Node Mode* case was covered in the sub-section dealing with that mode. Here we discuss some issues related to choosing and testing λ_l and λ_u, since they can impact the amount of processing required by the algorithm. We emphasize that the algorithm will never fail as a result of the choice here, since these values are automatically moved further apart when a signal requires it. The choice is simply a matter of efficiency. We begin with *Bandwidth Mode*.

We have used values of 0.5 and 10 for the initial values of λ_l and λ_u respectively. They work well because we often achieve desireable bases using a *Bandwidth Budget* in the mid-ranges (i.e., 35 - 70%), which tends to correspond to operating slope magnitudes in the 0.5 to 10 range. At least this has been the case for the sounds we tested against. An example of this can be seen in Fig.*4-10*, where the slope magnitudes corresponding to the 35-70% bandwidth range are indeed in the 0.5 to 10 range (the negative slope corresponds to λ). For the same signal, Fig.*4-11* shows the total number of iterations of the bisection algorithm. The average for budgets whose operating slopes fall within the 0.5 to 10 range is four, whereas the

global average is roughly five.

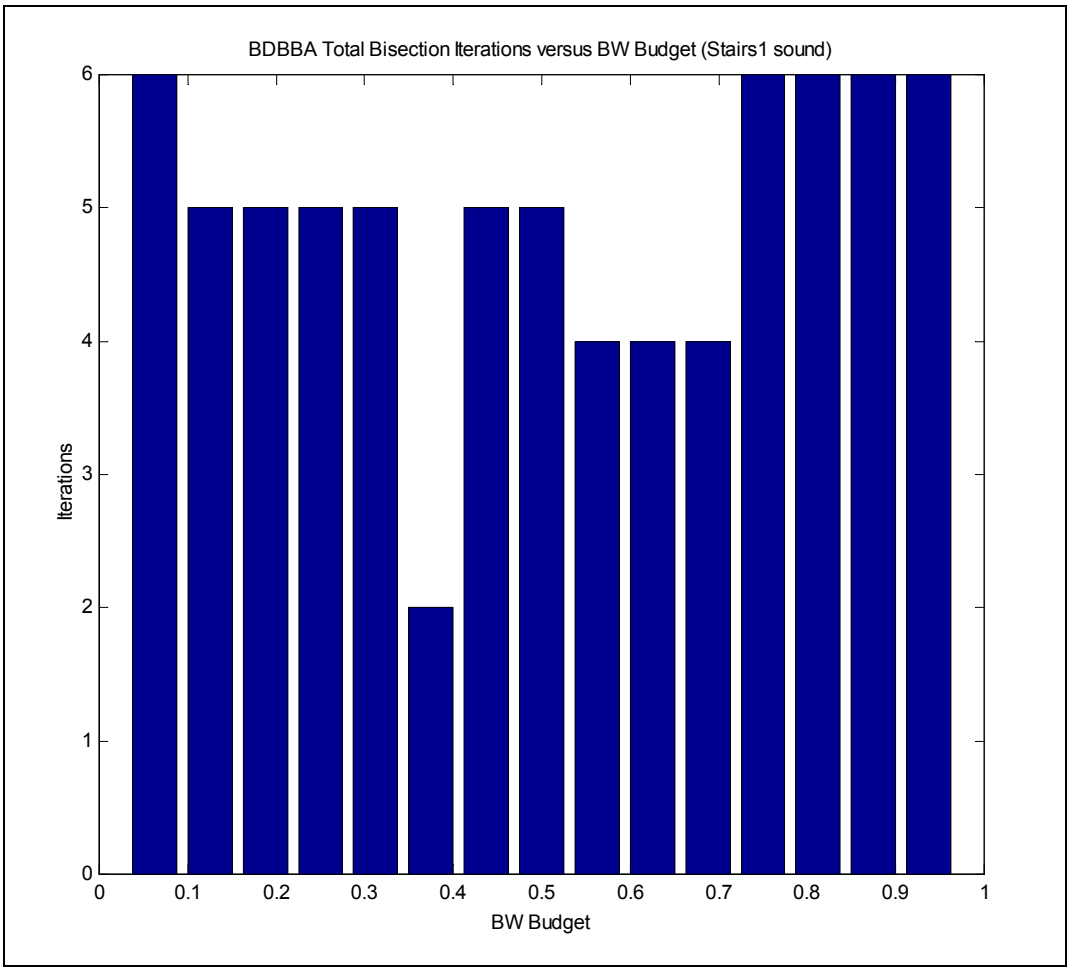

Figure 4-11: Total Bisection Iterations for BDBBA and Stairs1 Sound

As we have described earlier, in *Node Mode* we build up a table of λ values versus the resulting bandwidth and distortion for each new *Bandwidth Budget* we try. The table provides some savings since for each new *Bandwidth Budget*, we can choose initial values for λ_u and λ_l that more closely surround the solution λ for that budget than simply choosing the conservative values of infinity and zero[†].

A simple illustration of the savings in iterations of the bisection algorithm achieved by retaining past *bandwidth-distortion* results is given in *Table 4.3*. We look at a sequence of four *Bandwidth Budget*s,

[†] We could have gone one step further and saved the data point for each λ tried, not just the solution values for each budget. However, it is not clear whether the added complexity would be warranted.

comparing the number of iterations of the bisection process necessary when each *Bandwidth Budget* is submitted independently as in *Bandwidth Mode*, versus when that sequence of budgets is submitted in *Node Mode* in an attempt to find a solution for a *Node Budget* of four and an initial bandwidth budget of 81.25% (left-most column value). The test sound *Stairs1* was used with the *WPT1* transform configuration in this test†. The *Bandwidth Mode* cases correspond to test cases *StairsBMode13, StairsBMode14, StairsBMode12, StairsBMode15* (*Table 5.13*) for the budgets 81.25%, 87.5%, 75%, and 93.75% respectively. *Node Mode* in this example requires a total of 16 invocations versus 24 for *Bandwidth Mode*, for a saving of over 33%. This was typical of the bisection iterations saved in *Node Mode* when utilizing past *bandwidth-distortion* results.

Table 4.3 Bisection Iterations: A Comparison of *Bandwidth Mode* versus *Node Mode*

Case	Bandwidth Budget (%)				Total and Average Iterations
	81.25	87.50	75.00	93.75	
***Bandwidth Mode* Iterations**	6	6	6	6	24 Total 6 Average
***Node Mode* Iterations[1]** **(*Node Budget* = 4, initial *Bandwidth Budget* = 81.25%)**	6	3	5	2	16 Total 4 Average

1.The budgets listed in the columns from left to right represent the order in which *Node Mode* would set the *Bandwidth Budget* to in its search for a solution.

4.6 Summary

In this chapter we have described our own application of the WPT to sound signal processing. We started by describing the types of sounds that our research is focused on. We refer to these as *environmental sounds* and distinguish them from *music sounds*. We claimed that the multiresolution nature of the WPT makes it a good fit for environmental sounds.

Next we described the various approaches we took to sound modifications, dividing them into three main categories: *coefficient modification and creation*, *resynthesis modifications*, and *post-resynthesis modifications*. The first category includes techniques such as time stretching and thresholding of coefficients, as well as the creation of new coefficients as is done in granular synthesis. In the second category we described two cross synthesis approaches, one where we use a different wavelet filter from that used during analysis, and the other where we use as our resynthesis filter a waveform that is not related to a

†.See *Table 5.1* for a description of the transform parameters associated with configuration *WPT1*, and *Table 5.2* for details on the test sound *Stairs1*.

wavelet at all, but rather is a sound signal different from the one transformed. In this case, we are trying to fuse the characteristics of the two sounds in some way. In the third category, we describe approaches that can be applied after resynthesis.

We then described a fixed basis that we have called the *pseudo-third-octave basis* (PTOB). In creating this particular arrangement of subbands we have tried to approximate the third-octave bandwidths found on many graphic equalizers. This spacing is also more aligned with the critical bands of human hearing than say the fixed spacing of the WPT complete tree basis or the *power-of-two* spacing of the dyadic DWT. As such, the PTOB is well suited for applying sound modifications that will be distinct to the listener.

Our investigations in the area of best basis selection were described next. We provided a general framework within which all our best basis selection approaches fit. This framework consists of the pruning of the complete tree basis based on the minimization of a cost function. By changing the cost function, we change the technique, and the basis that results. Our main focus was in the development of an algorithm we call the BDBBA, which can be used in one of two modes of operation: *Node Mode* or *Bandwidth Mode*. *Node Mode* is used to reduce the number of nodes associated with a basis to something more manageable. *Bandwidth Mode* is used to attain a better correspondence between the energy distribution of the signal and the time-frequency resolution of the transform. The BDBBA relies on the quantization of subbands using one of two possible quantizers. This allows for the delineation of subbands between those that are more versus less important to the makeup of the sound. By changing a user supplied budget, we can change where the boundary between these two classes of subbands exists. In most of our work we have focused on a particular choice for the two quantizers, where the low rate uses no bits, and the high rate uses enough bits such that the original transform coefficient is unchanged. We have called this the *all-or-nothing* quantization case. We also look at other quantization scenarios, and the pros and cons associated with them versus the *all-or-nothing* case. Finally, still within the best basis selection framework, we considered other cost function definitions, such as those that are entropy based (the EBBA and TBBA), as well as possible model-based cost functions, such as a predictor-based scheme to model the coefficients of each subband.

In the last section of this chapter we considered issues of algorithm complexity, since they relate directly to the potential speed of our transformations. We considered both the transform itself (WPT), and the basis selection algorithms.

In the next chapter we provide a description of the software developed to experiment with the concepts

described thus far, and then describe the nature and results of our experimentation. We include cross referenced audio clips that help demonstrate the results of these experiments.

Chapter 5
GUI Description and Testing Results

This chapter focuses on the implementation and testing of the ideas and algorithms described in *Chapter 4*. Specifically, we have developed a *graphical user interface* (GUI) based application using the *Matlab* programming environment to serve as a test-bed for our experimentation[†]. The GUI has allowed for experimentation with the transforms, modification schemes, and basis selection approaches described previously, as well as the prototyping of how a user might work with these tools. Following a description of the GUI, we move on to describing the results of our testing and experimentation. Our results include a collection of audio samples as well, all cross referenced in the descriptions.

5.1 Overview of Graphical User Interface Tool - "*Sharp Attack*"

We call our GUI tool *Sharp Attack*, since the name evokes several aspects of the types of sounds it was designed to work with. These sounds are environmental in nature, often comprised of the *sharp* rise-time components or *attacks* that are typical in natural soundscapes. We associate the *shark* with the natural environment, and its primitive untamed nature evokes the nature of environmental sounds, which, when compared to so-called man-made sounds, are generally less organized, structured, and conducive to characterization and modeling. We will use the term GUI and the name *Sharp Attack* interchangeably.

The intent of the GUI is to act as a platform for convenient experimentation. This experimentation is with the WPT itself, with basis selection approaches, and with modifications that take advantage of the WPT structure in some way. It also involves experimentation with how a user might interact with the WPT during the act of either composition or performance. The GUI embodies at least one approach to working with the WPT. To emphasize speed and create an interface that more closely resembles what might be possible in hardware, we have favoured the use of push buttons, selection lists, and check boxes over drop down menu items.

Sharp Attack is useful in a non-real-time mode such as would be the case in composition. However, by emphasizing flexibility and generality in making the tool a broadly useful platform, not all functions currently operate in real-time[††]. This compromise between generality and performance was inevitable and is

[†] The *Matlab* environment is available from *The MathWorks, Inc*.

not uncommon.

The GUI is comprised of a series of windows, each associated with a particular functionality. The two main windows associated directly with the WPT itself are the WPT *Analysis* window and the WPT *Resynthesis* window. We devote a section to each in the following. The *Basis Selection* window provides an interface for all aspects of basis selection, and we discuss this in its own section as well. Finally, we have a series of windows that provide the interface for imparting various modifications on the transform coefficients. Currently, we have the *Filter and Create* window, the *Time Scaling* window, and the *Thresholding* window. These windows are collectively discussed in a single section.

Note that while the example windows shown in subsequent figures provide a reasonable idea of the GUI makeup in general, a good deal is lost in terms of image quality. Subtle color changes and resolution in general both suffer in the transfer, and the window size is significantly smaller than that normally used in the GUI application.

5.1.1 Analysis Window

The *Analysis* window provides an interface for the user to experiment with the application of different WPT-based analysis transforms of the signal. Fig.*5-1* shows an example of this window. Primarily, it is where one selects the wavelet packet filter to be used and the depth of the transform.

In the example of the figure, the *sym5* filter was used to *depth-4* to analyze the *Bird* signal, resulting in the 16 subband energy plot at the right. For boundary extension, the *symmetric* form was chosen. Above this colour energy plot is a time domain representation of the signal, and below it is a plot of a selected subband's coefficient energies. The time axis along the horizontal dimension is shown under this lower plot and applies to all the plots above it. It is in units of samples at the original sample rate of the signal. Subband coefficients are of course downsampled relative to the original, so to keep a consistent representation, these samples are duplicated in order to plot them with the original, and with subbands at other resolutions.

As with the other windows in the GUI, there are several options in how the transform data is plotted in three dimensions. We will refer to this three dimensional plot as the *image,* or the *image plot*. In terms of the plot data itself, the subbands can be either *frequency* ordered or *naturally* ordered within the image (*Section*

---

††. Though some of the less processor intensive operations do operate in real-time, such as the scaling or disabling of subbands while in a looping mode.

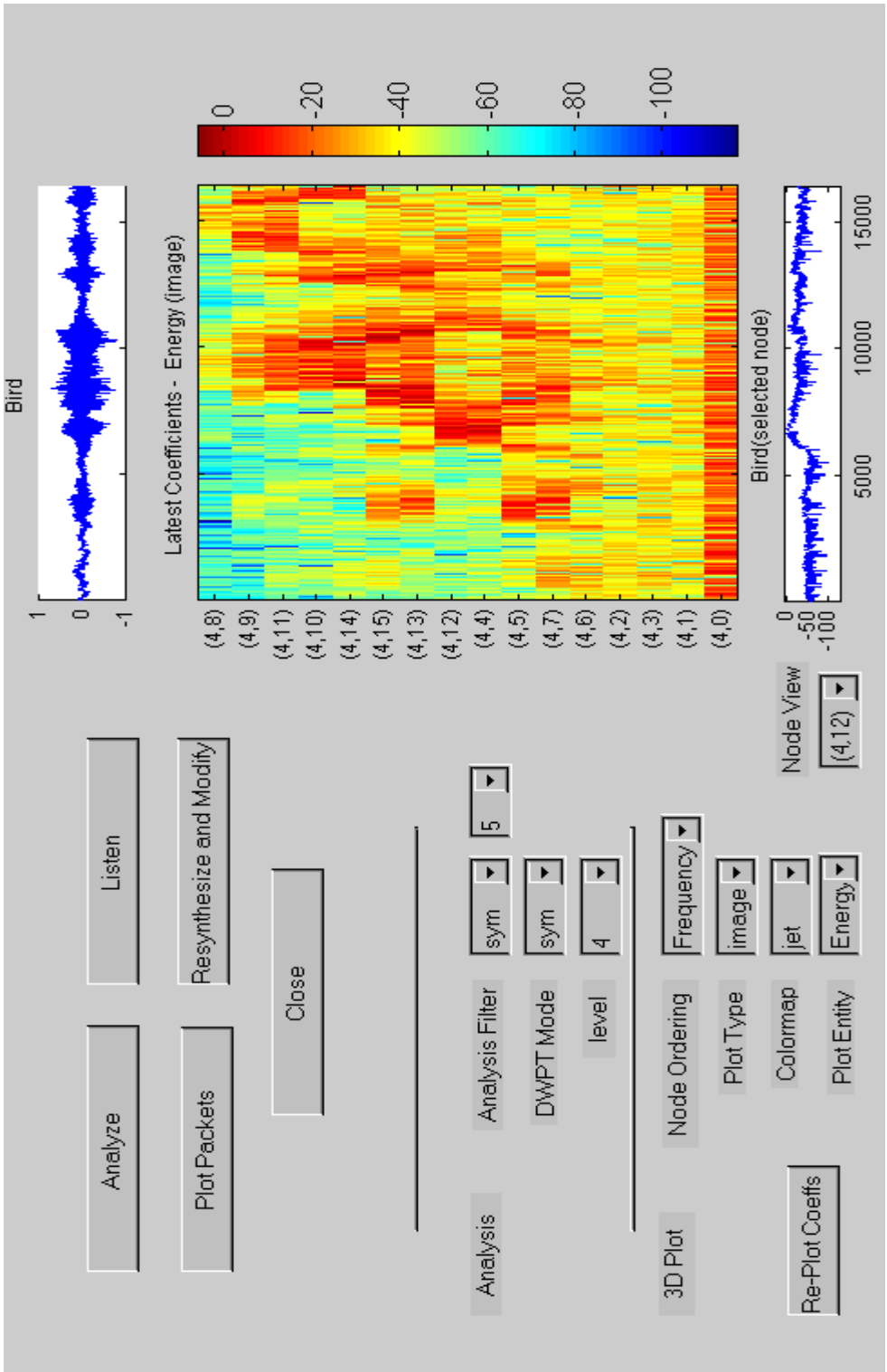

Figure 5-1: GUI Analysis Window

3.4.2), and either coefficient amplitudes or energy (dB) can be shown. Use of a decibel (dB) scale can be useful since the dynamic range across subbands is often so large that subbands other than the ones containing most of the energy look blank (all zeroes). In terms of the image in general, various colour schemes can be selected and the type of image can be chosen as well. Either a flat colour-based two dimensional image (as shown in the figure), or a three dimensional projection can be selected.

The window allows the user to listen to the signal, and also to plot out the characteristics of the chosen filter, such as its impulse and frequency responses, such as was shown in Fig.*3-7* for the *Haar* filter. The *Resynthesis* window described next is opened from the *Analysis window* by pressing the *Resynthesize and Modify* button.

5.1.2 Resynthesis Window

The *Resynthesis* window is where the user would spend the most time. After performing the initial WPT analysis, the user will want to impart various modifications on the signal and resynthesize. Resynthesis modifications and post-resynthesis modifications occur from within this window, and the other modifications occur from windows opened from it. Fig.*5-2* shows an example of the window, carrying forward the *Bird* signal example used in the *Analysis* window of Fig.*5-1*.

The resynthesized original is shown in the time domain based two dimensional plot at the top right of the figure ("*Bird(saved)"*). The plot below this ("*Bird(latest)"*) shows a resynthesis with various modifications that we will describe later. Since such modifications were made, this plot is not identical to the one above it. The *saved* and *latest* nomenclature denote two separate data structures. *Latest* holds the set of transform coefficients that have been modified but not yet saved. The user can cumulatively impart any number of modifications from any of the potential coefficient modification mechanisms on the coefficients in *latest* (i.e., *latest* gets passed to the coefficient modification and basis selection windows). When the user is satisfied with the sound achieved via a set of modifications, *latest* can be transferred to the *saved* structure. Now the user can again experiment with modifications without worry that the previous set of successful changes will be destroyed by a poorly chosen subsequent modification. The *saved* structure initially holds the original resynthesized coefficients from the analysis transform. This initial resynthesis of the signal is performed by default when the resynthesis window is opened.

The *image* plot is similar to the one in the analysis window, but more information is included. In addition to being capable of showing coefficient amplitudes and energies, this window can also display the

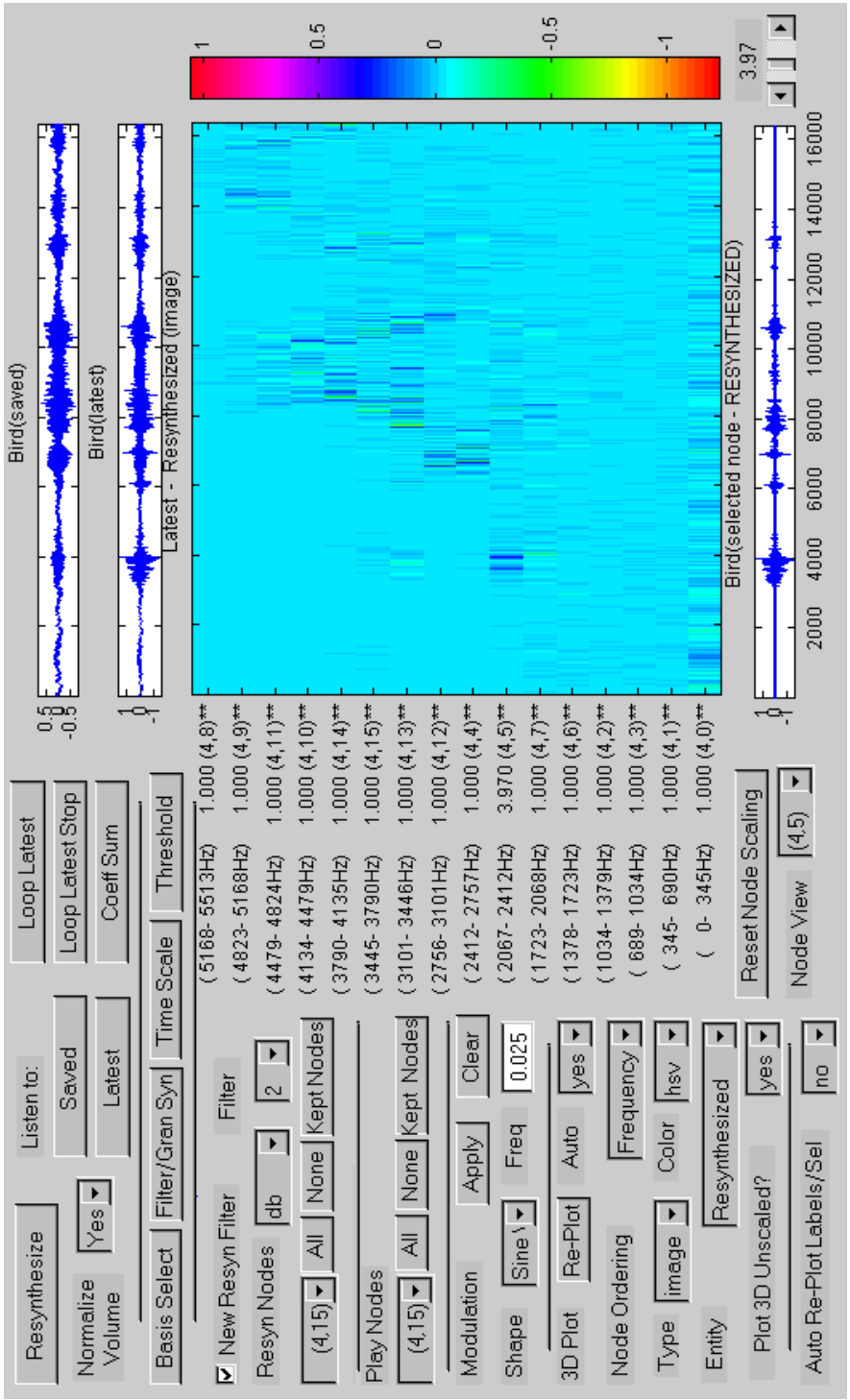

Figure 5-2: GUI Resynthesis and Modification Window

resynthesized subbands, as is the case in the figure. The plot below it is again the selected subband plot. Also, this selected subband can be scaled according to the amplitude scaling slider to the right of that plot.

To the left of the image plot is some detailed information associated with each subband. Starting from the left, there is the frequency range associated with each subband[†]. Next there is the scale value currently applied to each subband, which is selected using the aforementioned slider. In the example of the figure, subband (4,5) is the only scaled subband. To the right of the scale value is the subband number in *depth/position* format. Finally beyond that is the optional presence of a double asterisk (**). This comes into play when basis selection is done and the coefficients of only some subbands are marked to be retained or *kept*. In that case, we still may decide to keep the unmarked subbands, and simply pay little attention to them when performing other modifications. This corresponds to the *lossless operation* case described earlier in *Chapter 4*. The double asterisk is a way of identifying the important, or *kept*, subbands.

At the top of the window on the left half we have a button to trigger resynthesis, and a series of controls associated with listening to the signal. In addition to being able to listen to the *latest* and *saved* signals, we can also loop the *latest* signal indefinitely. This looping is a useful feature, which allows us to keep trying different changes to hear the result. It also allows for experimentation with continuous variations on a given sound. Some later examples in *Section 5.4.3* demonstrate this. When in *loop* mode, resynthesis occurs automatically after each change, and options exist to select what if anything gets replotted between resynthesis, since plotting can be quite time consuming.

The *Normalize Volume* option allows for selection of roughly constant volume independent of signal energy. This option can be useful as we strip away subbands of the signal to listen to what is left. Without normalization, the volume can get quite low. In other cases, this volume change may be exactly what we want.

The four buttons beneath this region open separate windows, either to do *basis selection*, or to modify the coefficient streams in some way. We will discuss these further in the sections dealing with the respective windows.

[†] As pointed out in *Section 4.4.2*, the actual bandwidth is not identical to the fractional subband width due to imperfect filtering. The frequency ranges next to each subband correspond to the frequencies associated with that subband given perfect filtering. In practice some aliased energy will exist in each subband corresponding to frequencies outside the band, and upon resynthesis images will be created at frequencies outside the ranges listed that are not completely eliminated. Only if no modifications are made will the aliases and images cancel for *perfect reconstruction*.

The region of the window below this is concerned with *new filter resynthesis* and *cross synthesis*. Other legitimate wavelet packet filters or completely arbitrary sounds employed as filters can be selected for the resynthesis operation. Which nodes the resynthesis applies to is also selectable. While on the topic of selective resynthesis, we point out that resynthesis due to coefficient modification only occurs for those subbands that have changed or have new coefficient values. This selective resynthesis can save a significant number of computations.

The next lower controls in the window allow the user to selectively disable some subbands on playback. This functionality combined with the amplitude scaling capability already mentioned comprises the *Scaling Subband Amplitudes* functionality.

Next there are controls for applying some form of *modulation* to the resynthesized subbands. The modulation waveform and frequency are selectable, as well as which subbands the modulation is applied to, and whether the modulation is raised to avoid zero crossings (AM), or maintained as a zero-offset signal (DSB). The distinction between AM and DSB is treated more fully in the *granular synthesis* portion of *Section 5.4.1*.

Finally at the bottom left of the figure are a series of plotting related controls, most of which have been mentioned earlier.

5.1.3 Basis Selection Window

The *Basis Selection* window allows the user to select a new basis from the complete tree basis associated with the initial WPT. Fig.*5-3* shows an example of the window, again with the *Bird* signal input.

The plotting options in the lower left portion of the window are similar to the other windows. The top left portion has three buttons. The *Resynthesize* button invokes the *Resynthesis and Modification* window described in the last section. As with the other modification windows and associated functions to be described in the next section, there is an option to not replot the result of a change. This saves time and is justified when we are primarily focussed on the sound created, or to get to another window with the result. The *Revert to Saved* button restores the original basis that existed upon opening the window, and *Done* simply closes the window.

Basis selection is controlled in the two centre left sections of the window. The upper of these sections is for the BDBBA basis selection approach, and is devoted a good deal of space due to the many options and

parameters associated with it. The section below the BDBBA section covers the other basis selection modes. These are the EBBA and TBBA best basis selection approaches, and the PTOB, WT and Complete Tree fixed bases. Also the user can split any terminal node in the tree using the *Split to Child Nodes* function, or join two sibling nodes to their parent node using the *Join at Parent Node* function.

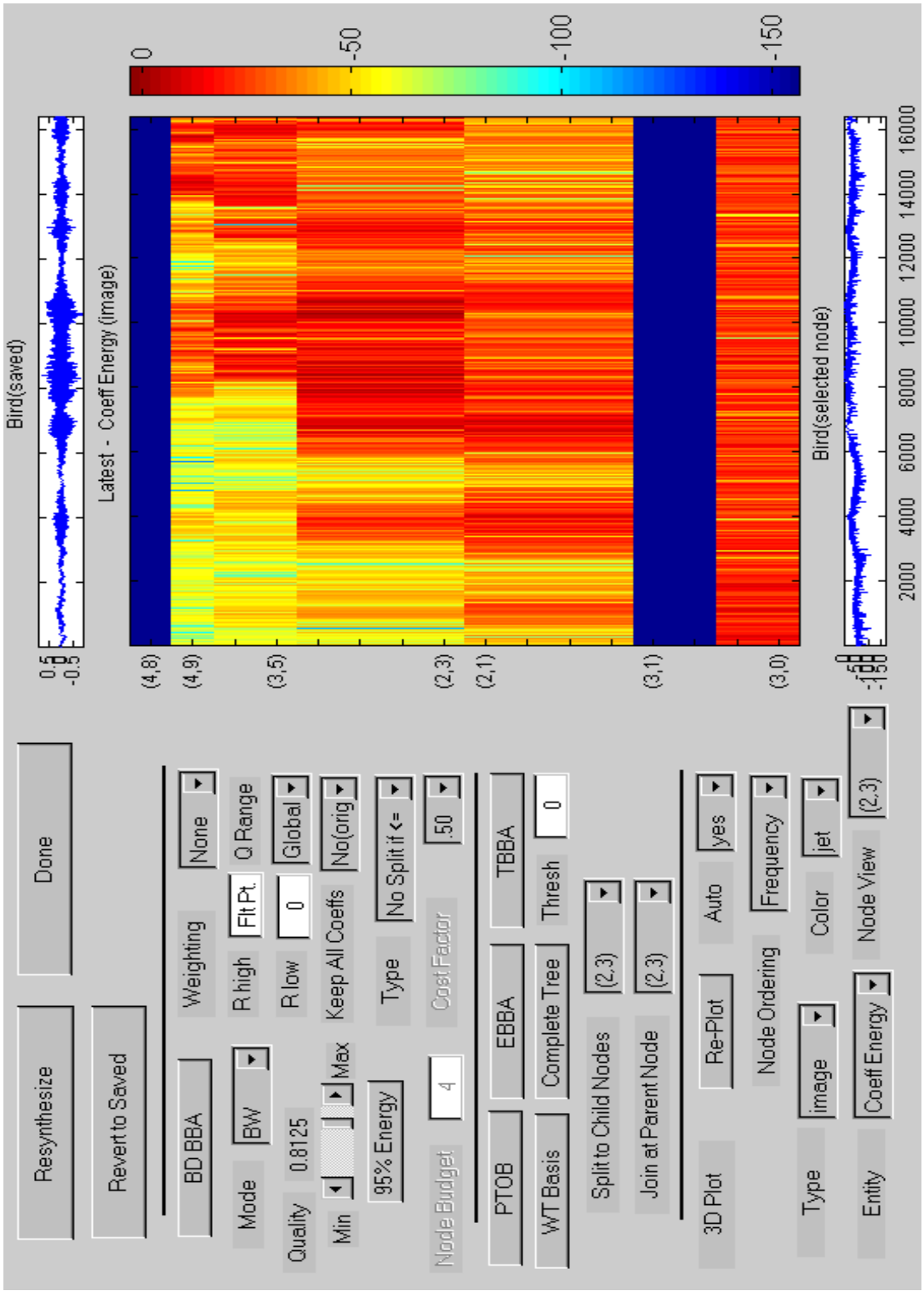

Figure 5-3: GUI Basis Selection Window

5.1.4 Coefficient Modification Windows

The GUI currently has three coefficient modification windows: *Filter and Create*, *Time Scaling*, and *Thresholding*. As with the *Basis Selection* window described in the previous section, these windows are opened from the *Resynthesis and Modifications* window, and return back to it once the modifications are made.

The *Filter and Create* window provides an interface for the filtering of one or more coefficient streams, and/or the creation of new coefficient steams. This latter function can be viewed as *granular synthesis*. The *Time Scaling* window allows for both stretching and contracting of coefficient streams in time, and the *Thresholding* window allows for the zeroing of coefficients whose amplitudes are below some configurable threshold value.

As an example of these modification windows we show the *Filter and Create* window in Fig.*5-4*.

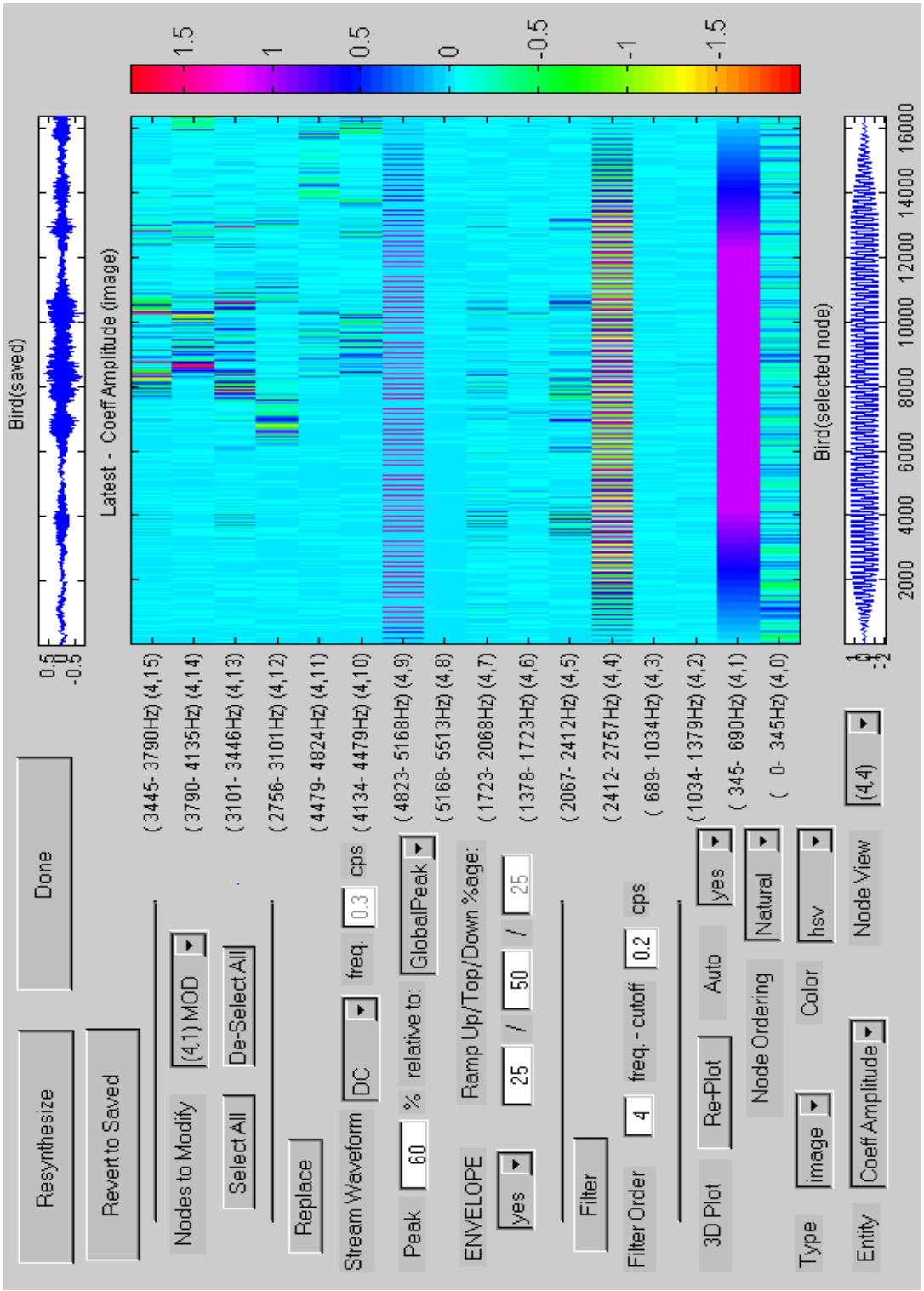

Figure 5-4: GUI Filter and Create Window

5.2 Test Transform Parameters Used

Here we state the default parameter choices made for the analysis and synthesis transforms in our research. These choices seemed to provide good performance in general, even though specific applications may well be better served with other parameter choices. Use of this consistent set of transform parameters facilitates comparison across test cases, and provides a standard reference point from which to explore variations. The purpose of listing these choices here is to avoid needless repetition of these parameters throughout the remainder of this chapter. The parameters are summarized at the end of this section in *Table 5.1*. *WPT1* in that table refers to the standard set of parameters that we describe in the following. The other cases are variations on this basic parameter set.

For our prototype filter we use the *sym5* FIR filter. It is from the *Symlet* family of filters proposed by Daubechies (1992), which are desirable because they are orthogonal and nearly symmetric. Recall from *Section 3.2* that achieving orthogonality and symmetry in the same wavelet filter is not possible for anything longer than a two coefficient filter (*Haar*). By achieving near symmetry, the filters exhibit near linear phase, and as such offer a nice compromise. The *sym5* filter is 10 coefficients in length, long enough to provide reasonable inter-band isolation, yet not so long as too overly blur events in the time domain. As can be seen from Fig.*5-5*, it also have a relatively smooth quality that resembles a windowed and scaled sinusoid. The packets in that figure are arranged in *frequency order* going from left to right and top to bottom (see *Section 3.4* for a discussion of *frequency* versus *natural* ordering).

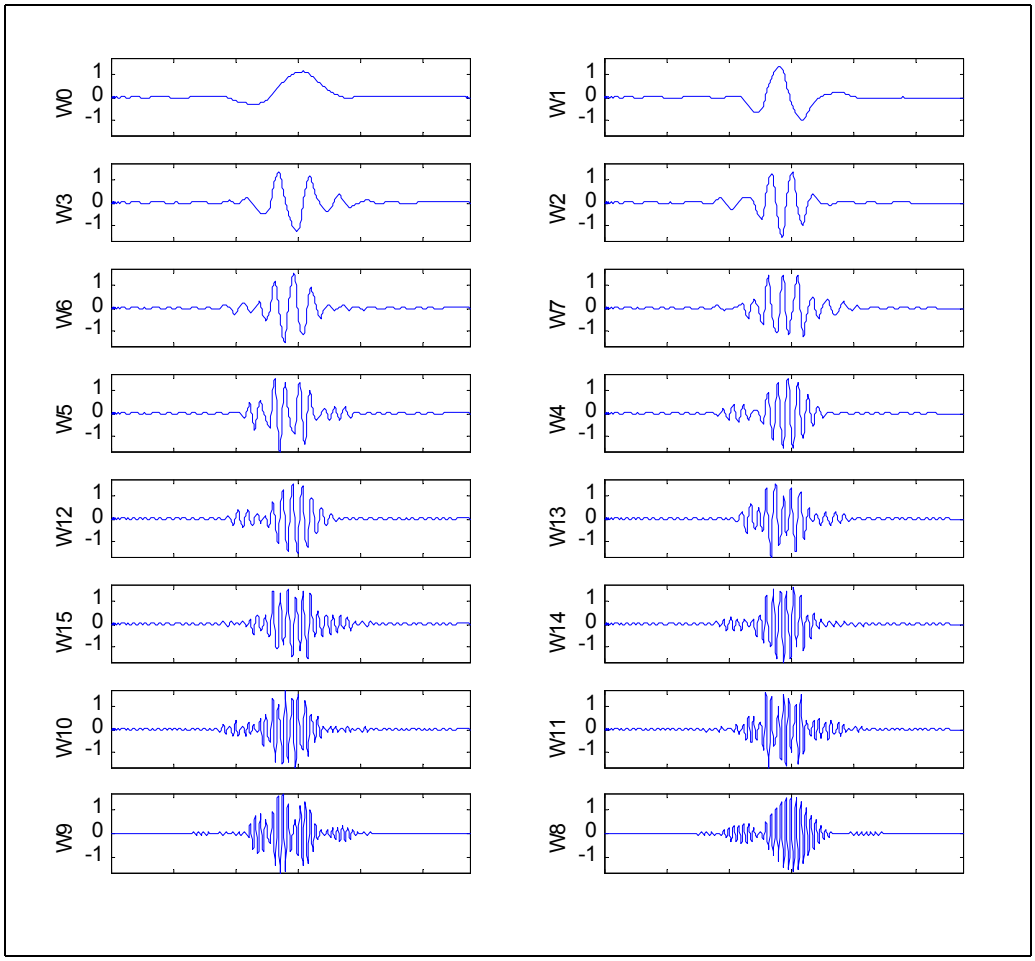

Figure 5-5: Sym5 Wavelet Packet Impulse Responses (Depth-4)

Use of a *depth-4* transform results in a complete tree of 16 subbands. Going much deeper results in too much information for the user and is difficult to plot and interpret. Also subbands get quite narrow and are, especially for typical environmental sounds, which are not primarily tonal, too narrow to characterize the energy events that comprise the sound. Going too shallow does not provide enough frequency detail and opportunities for signal modifications. Also note that when we perform *best basis* selection, the 16 subbands are generally reduced, and in the case of *fixed bases*, we may go deeper, allocating bandwidth based more on psychoacoustic concerns.

The three main options for boundary extension were described in *Section 3.5*. We have chosen to use *symmetric extension*. To some extent the best approach depends on the signal and the length of filter used[†]. If

the signal tapers gradually to zero at both ends, then perhaps *zero-padding* is acceptable. However, since we are often extracting segments of larger audio events, and not necessarily including their natural onsets and decays, *zero-padding* would seem inappropriate. Furthermore, there is no reason to assume the segment we have is periodic, so *periodic extension* seems inappropriate. The onset could be quite different from the end.

Symmetric extension is premised on the most reasonable assumption for our case, namely that the signal before its start would be similar to the way it is after its start, and analogously after its end it would be similar to before its end. In cases where the clip does taper gradually to zero, this approach should still provide good results.

Table 5.1 Wavelet Packet Transforms Tested: Name and Parameters

WPT Name	Depth	Filter[1]	Boundary Extension
WPT1	4	*sym5*	*symmetric*
WPT2	3	*sym5*	*symmetric*
WPT3	5	*sym5*	*symmetric*
WPT4	4	*haar*	*symmetric*
WPT5	4	*db10*	*symmetric*
WPT6	5	*sym8*	*symmetric*

1.If *New Filter Resynthesis* or *Cross Synthesis* is applied, then the listed filter here is what was used for the analysis filterbank, and the resynthesis filter is explicitly listed.

5.3 Test Sounds Used

The source sounds used in our testing are listed in *Table 5.2*. These sounds are later modified in various ways, and the resulting files are named such that the source sound names form a prefix to these test case names. Each entry in *Table 5.2* includes the sound name, a brief description, its sample rate, and its length. All sounds are single channel (*mono*). Most of the sounds are derived from the environment, whether *natural* or *man-made* (see also Section *4.1 Target Sounds*). We are, however, also interested in how well our techniques apply to other classes of sounds, and so have included several sounds of a more tonal nature, as well as a sound that is somewhat formant based.

---

†. Use of a 10 coefficient filter (such as the *sym5)* in a *depth-5* WPT decomposition results in an effective filter length of $2^5 \times 10 = 320$ points. Applying this to a one second file at 16 kHz would result in a total of $((2 \times 320) / 16000) \times 100\% = 4\%$ of the signal being affected. Unlike a full song, this is a little more significant.

The test sounds were either recorded by us, or taken from the CD-ROM edition of the *Handbook for Acoustic Ecology* (Truax 1999). In the latter case, some of the sounds were resampled to other rates and shortened. Since all files contain linear uncompressed sounds, they are stored in the *wav* format, and have a "*.wav"* extension†. Also, the sounds have all been normalized to a consistent peak amplitude value.

Table 5.2 Test Sound Names and Descriptions

Sound Name	Description	Sample Rate (Hz)	Length
Can	Metal can dropped on concrete	8,000	4,096 samples; 0.512 seconds
Knock	Hand knocking plastic	8,000	4,096 samples; 0.512 seconds
Stairs1	Person walking down outside wood stairs	8,000	16,384 samples; 2.048 seconds
Lake	White noise of wilderness lake and creaking wood sound	16,000	65,536 samples; 4.096 seconds
Bird	Bird chirp near seashore	11,025	16,384 samples; 1.486 seconds
Fly	Housefly	11,025	8,192 samples; 0.743 seconds
Train	Shunting train at train station	22,050	65,536 samples; 2.972 seconds
Doors	Closing doors at train station	32,000	65,536 samples; 2.048 seconds
Dragging	Dragging plastic pale	8,000	8,192 samples; 1.024 seconds
Stairs2	Dog walking down outside wood stairs	8,000	8,192 samples; 1.024 seconds
Diesel	Diesel engine starting	44,100	65,536 samples; 1.486 seconds
Shotgun	Shotgun	44,100	32,768 samples; 0.743 seconds
Rain	Rain	32,000	32,768 samples; 1.024 seconds
Noise	White noise	44,100	65,536 samples; 1.486 seconds
Voice	Resonant voice	32,000	131,072 samples; 4.096 seconds
440Hz	440 Hz tone	8,000	16,384 samples; 2.048 seconds
Gamelan	Gamelan	16,000	32,768 samples; 2.048 seconds
Scale	Equal tempered scale	11,025	65,536 samples; 5.944 seconds
Foghorn	Foghorn	8,000	32,768 samples; 4.096 seconds
NoiseChirp	Chirp with noise	8,000	1,024 samples; 0.128 seconds
Sweep	Sweeping Tone - 440 Hz to 4 kHz - Linear	8,000	32,000 samples; 4.00 seconds

† By providing the sound files in a "data" file format (such as the *wav* file format), rather then an audio file format (CD), we can preserve the sample rates used in their processing. Doing so makes it possible to reproduce the examples (processed sounds, plots, etc.) presented in this thesis.

5.4 Sound Modification Test Results and Observations

We now describe the results of various tests that we have performed using the *Sharp Attack* GUI. The tests are based on the test conditions and sounds described in the preceding sections. In this section we describe the results of our *sound modification* experimentation, and in the next section we describe our *basis selection* results.

In all cases, we provide audio clips cross referenced to the test cases performed. The name assigned to a given test case is a concatenation of the original sound name from *Table 5.2*, a short abbreviation of the modification or basis selection technique applied, and a sequence number. For example, Can*TSC1* denotes the first test case using the *Can* sound submitted to *Time Stretching and Contraction (TSC)* modifications. A table summarizing the test cases is provided at the end of each section.

5.4.1 Coefficient Modification and Creation

5.4.1.1 Time Stretching and Contraction

With these modifications, we either stretch or contract the signal in time by some factor. Recall that we achieve this by resampling the transform coefficient stream by a factor equal to the desired length change. To avoid alias and image creation within the coefficient stream itself, we apply the necessary anti-aliasing and interpolation filtering. Our experimentation has shown that time stretching and contraction are quite effective in producing interesting modifications to the source sound.

In the case of stretching, one quite pronounced effect is that the sound becomes increasingly tonal in character the more it is stretched out. Since the wavelet packets are always comprised of an oscillatory component, this is not too surprising. As the sound is stretched out, the tonality of the packets themselves emerge, eventually dominating the overall character of the sound. Recall the plot of the wavelet packets used in our testing (*Sym5)* given in Fig.*5-5*. It shows how the packets resemble windowed scaled sinusoids. While not all packets are this smooth, such as the *Haar* packets shown in Fig.*3-7*, the emergence of a tonality still occurs.

The sound sample *Can* was subjected to stretches of varying amounts. For example it was stretched by a factor of 10 to produce *CanTSC1*. The ringing tonal quality of this clip is quite prominent, especially when compared to the original, and the spectrograms bears this out. The spectrogram of *Can* unmodified is shown below in Fig.*5-6*. Frequency up to one half the sample frequency is shown on the vertical scale, and time is indicated along the horizontal scale[†]. Compare this to the spectrogram of the stretched signal, shown in

Fig. 5-7. The frequency banding due to the emerging of the tonality of the packets is prominent. Essentially, the stretching of the coefficient streams in each subband (by upsampling and interpolation filtering) has made the bandwidth of the coefficients in those subbands much narrower. In the extreme, each coefficient stream would be flat in amplitude, and the bandwidth of each resynthesized subband would be that of the packet.

---

† These spectrograms use a 256 point FFT with a Blackmann window. Use of the relatively short 256 points helps preserve timing information better.

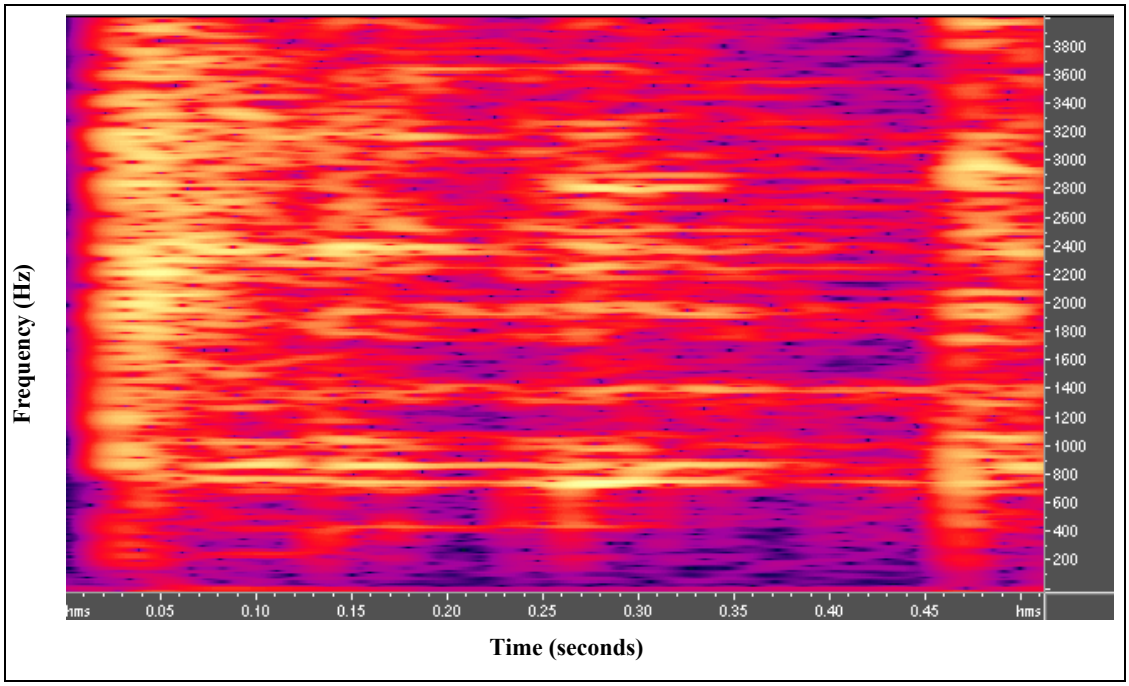

Figure 5-6: Spectrogram of *Can*
(256 point DFT)

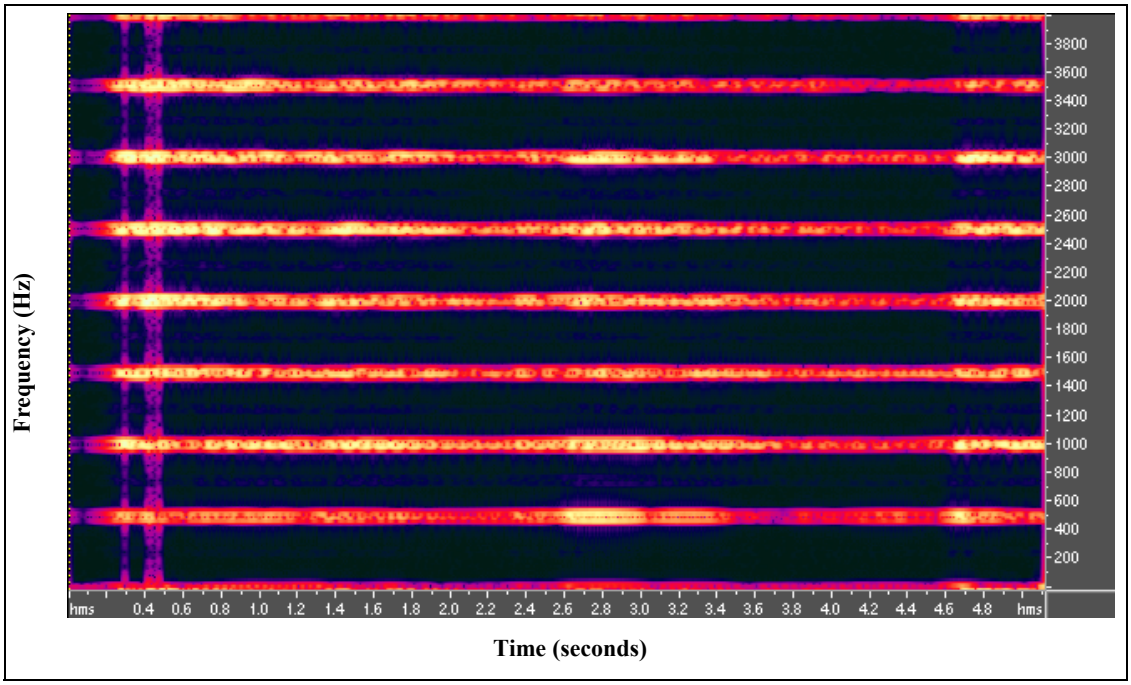

Figure 5-7: Spectrogram of *CanTSC1*(*Can* stretched by factor of 10)
(256 point DFT)

Note the location and number of frequency lines in Fig.*5-7*. While there are 16 wavelet packets, there are only nine frequency lines, occurring in equal spacing from DC to $f_s/2$. One might expect a full 16 lines, however, if we look closely at the packets, we can see that pairs of packets have very similar frequency content. This similarity is easiest to see with the eight *Haar* packets shown earlier in Fig.*3-7*, but one can also see the similarity in pairs of other packets, such as our *sym5* packet of Fig.*5-5*. Referring back to the *Haar* packets, if we put them in the *frequency ordering* and then take pairs in sequence, excluding the lowest and highest frequency packets, we find that the spectral contributions are the same for each packet within the same pair. In our example, the groupings are:

- *w0*
- *w1* and *w3*
- *w2* and *w6*
- *w7* and *w5*
- *w4*

In fact in the *Haar* case, if packets are strung together in succession, which is roughly what occurs during time stretching, the resulting waveforms of these packet pairs will look identical with the exception of a phase shift. This pairing is analogous to the real and imaginary parts of the complex Fourier basis functions. While the paired packets are less identical for other packets, the grouping still occurs, accounting for the noted number of frequencies.

Other examples of stretching are given in *CanTSC2-CanTSC5.* As we stretch from a factor of two (*CanTSC2)*, to four (*CanTSC3)*, to eight (*CanTSC4)*, we hear a lowering of the pitch of the original sound accompanied by the emerging of the packet tonality, which is already prominent at a stretch factor of four in this case. In *CanTSC6*, we stretched by a factor of 1.25. Such small length changes can be useful for matching the length of a sound clip to some other event. While a slight pitch change can still be heard, the overall character of the sound is preserved.

While we also shortened *Can* by two in *CanTSC7*, since *Can* is quite short to begin with, the result is not as interesting as the contraction of a longer sound. *LakeTSC1* demonstrates a contraction by two and *LakeTSC2* by four. The clips sound progressively more hurried compared to the original, with an expected overall pitch increase. We also stretched *Lake* by a factor of three in *LakeTSC3*. The result is an airy sustained quality reminiscent of many granular synthesized sounds.

In our description of the process used for time stretching and contraction in *Section 5.4.1*, we stated that we use a relatively long lowpass filter for interpolation and/or anti-aliasing. This long filter is our default

approach. However, we also considered using no filter, or *nearest neighbour* interpolation. In other words, in the case of stretching, the new samples are obtained simply by duplicating the existing sample that is closest. In the case of contraction, it simply means that no anti-aliasing filter is used prior to downsampling. One can compare the result of using the filter versus nearest neighbour interpolation by comparing sounds *CanTSC4* and *CanTSC5* respectively. Both stretch the source sound by a factor of eight. The latter is much rougher or grittier sounding.

We also looked at the effect basis choice can have on the time stretching result. *CanTSC8*, for example, uses the standard wavelet (WT) basis with octave spaced subbands. Like *CanTSC1* it is stretched by a factor of 10. The spectrogram of the WT basis case *CanTSC8* is shown in Fig.*5-8*.

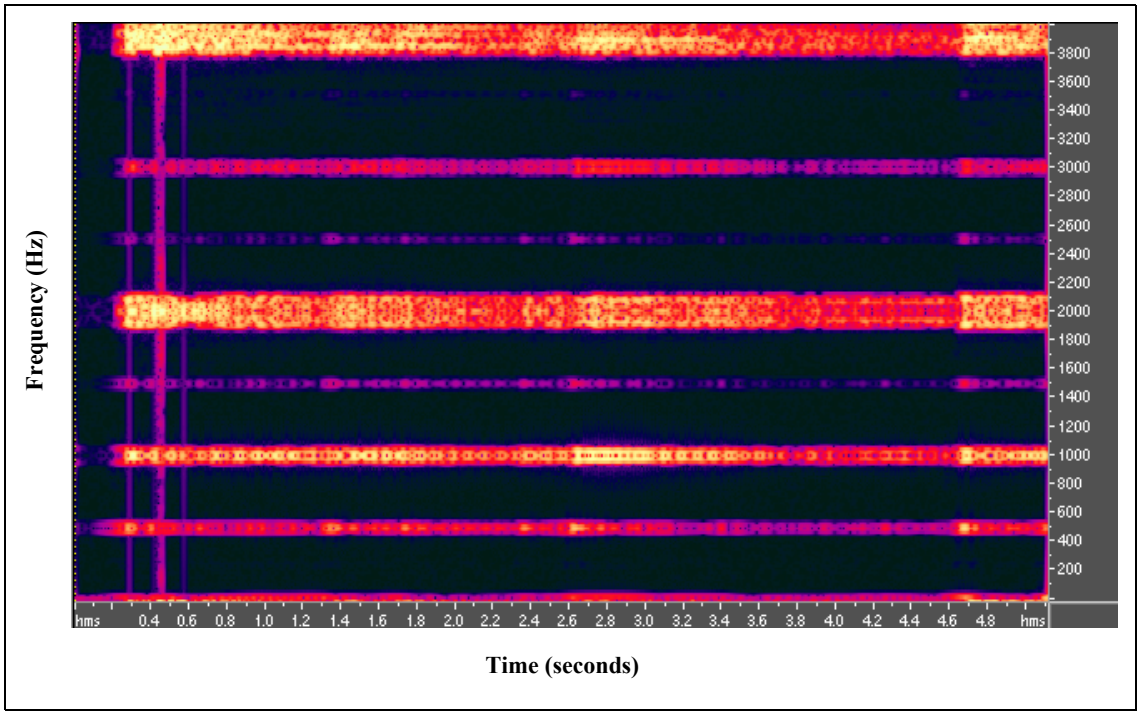

Figure 5-8: Spectrogram of *CanTSC8* (WT basis)
(256 point DFT)

One of the first things to notice is how the bands get wider with increasing frequency, reminding us that the WT basis is multiresolution. The higher frequency bands are associated with basis functions that are shorter in time. As such, they will have a wider bandwidth. The primary subbands are at *DC*, 0.5 kHz, 1 kHz, 2 kHz, and 4 kHz, as expected. One can also see some energy at the intermediate frequencies of 1.5 kHz and 2.5 kHz due to imaging (see *Section 4.2.1.4* for a description of imaging effects). The sound in the WT basis case is more shrill and less dense compared to the WPT complete tree basis.

If we compare the two cases of the WT basis and the WPT complete tree basis relative to musical notes and chords, we can observe that the WT case (*CanTSC8)* is comprised strictly of octave spaced notes. The first non-DC tonal component at 500 Hz is a little higher than a *B*. Let's call it a *B* for convenience. Then we have additional *B* notes one, two, and three octaves above this *B*. The WPT complete tree basis of *CanTSC1* is more complex. We again have the *B* note as the first non-DC component. Then we have a *B* note an octave above that. Above this *B* we now take two steps to reach the next *B*. These steps are evenly spaced, which corresponds to successive intervals of a *flat fifth* (*b*5). As such the note at the intermediate step (1500 Hz) would correspond to an *F*. Now proceeding upward from the *B* at 2 kHz there are four steps to reach the octave at 4kHz. These *1/4th* octave steps correspond to successive intervals of a *minor third*. Again, assuming our note at 2 kHz is a *B*, the notes going up would be *B*, *D*, *F*, *G#*, and finally the *B* again at 4 kHz. It turns out that this corresponds to a diminished chord (*B dim*). So we could roughly view this collection of tones as a *B* diminished chord with extra bass *B* notes and a *flat 5* note all below it. As such, we would expect a richer, more complex sound than the simpler octave spaced 'chord' of the WT basis case.

Other examples of aesthetically interesting stretches include *FlyTSC1*, a fly sound stretched by a factor of four, and *FoghornTSC1*, a fog horn stretched by a factor of 2.5. The example sounds are summarized in *Table 5.3*.

Table 5.3 Summary of Time Stretching and Contraction Sound Examples

Sound Name	Original Sound	Transform / Basis	Modifications
CanTSC1	*Can*	*WPT1 / Complete Tree Basis*	*-Stretch:10x*
CanTSC2			*-Stretch:2x*
CanTSC3			*-Stretch:4x*
CanTSC4			*-Stretch:8x*
CanTSC5			*-Stretch:8x (nearest neighbour interpolation)*
CanTSC6			*-Stretch:1.25x*
CanTSC7			*-Stretch:0.5x (=contract by 2)*
CanTSC8		*WPT1 / WT Basis*	*-Stretch:10x*
LakeTSC1	*Lake*	*WPT1 / Complete Tree Basis*	*-Stretch:0.5x (=contract by 2)*
LakeTSC2			*-Stretch:0.25x (=contract by 4)*
LakeTSC3			*-Stretch:3x*
FlyTSC1	*Fly*		*-Stretch:4x*
FoghornTSC1	*Foghorn*	*WPT1 / Nodes (2,0), (4,5), (3,3) selected with BDBBA*	*-Stretch:2.5x*

5.4.1.2 Thresholding

With *thresholding* modifications, we zero all coefficients in each subband that are below some configurable amplitude threshold. The threshold can be set on a subband basis, or globally, so that a single value is used across all subbands. Our testing of thresholding modifications has shown that the technique can produce some aesthetically interesting effects, especially when applied quite aggressively to the sound. While the precise result is not always predictable, some type of thinning out and/or roughening of the sound often occurs. Attacks are often sharpened due to the elimination of some or all of the ramping up of amplitude that usually occurs with attack onsets. An analogous effect occurs at the end of a sound event. Furthermore, elimination of coefficients can bring out strong alias components whose elimination relies on the presence of those coefficients, since alias cancellation only occurs fully when coefficients are not removed or altered.

We provide examples from four source sounds: *Can, Dragging, Diesel,* and *Scale.* In test cases *CanThesh1* - *CanThesh3*, we apply increasingly large (aggressive) soft global thresholds. Only the lowest

frequency subband is left unaltered[†]. Starting with the original sound, one can hear it progressively diverge from the original sound, taking on a sparser, crisper, more percussive character with each higher threshold. The results retain much of the large scale shape of the sound, but at the same time change within that to something new and often interesting. We found that the use of successively higher thresholds on a given sound provides an interesting way of slowly eroding the sound, perhaps until nothing is left of it.

In the case of *Dragging*, the result of applying a moderate amount of soft global thresholding can be heard in *DraggingThresh1*. While the original sound is quite rough to begin with, this is partially replaced with larger scale distortions.

DieselThresh1 was created using *hard thresholding* in a subband-selective way, with the lowest two subbands being thresholded the most. One might not relate it to the original *Diesel* sound, unless it is heard immediately after.

Finally, *ScaleThresh1* and *ScaleThresh2* show how a simple sequence of notes rising up through one octave of the 12 note chromatic scale are altered via thresholding. In the latter case, the threshold is relatively high, and the lowest subband is dropped entirely. This test shows clearly the effect of alias cancellation breaking down. Fig.*5-9* shows the spectrogram of *ScaleThresh2*, which shows the spectrum aliased around the half band ($f_s/4$) axis. The aliased portion is lower energy presumably because some alias cancellation is still occurring.

Table 5.4 provides a summary of the test cases discussed, and provides additional detail on the threshold values used

[†] Complete retention of the lowest subband is not essential, but stems from the initial implementation based on noise thresholding and compression, where generally the lowest subband is not modified. In the case of noise reduction, the noise components will rarely fall in this subband, and in the case of compression, the lowest frequencies often contain a large and significant part of the original signal energy. When we apply a non-global threshold, we allow for thresholding the low frequency subband as well.

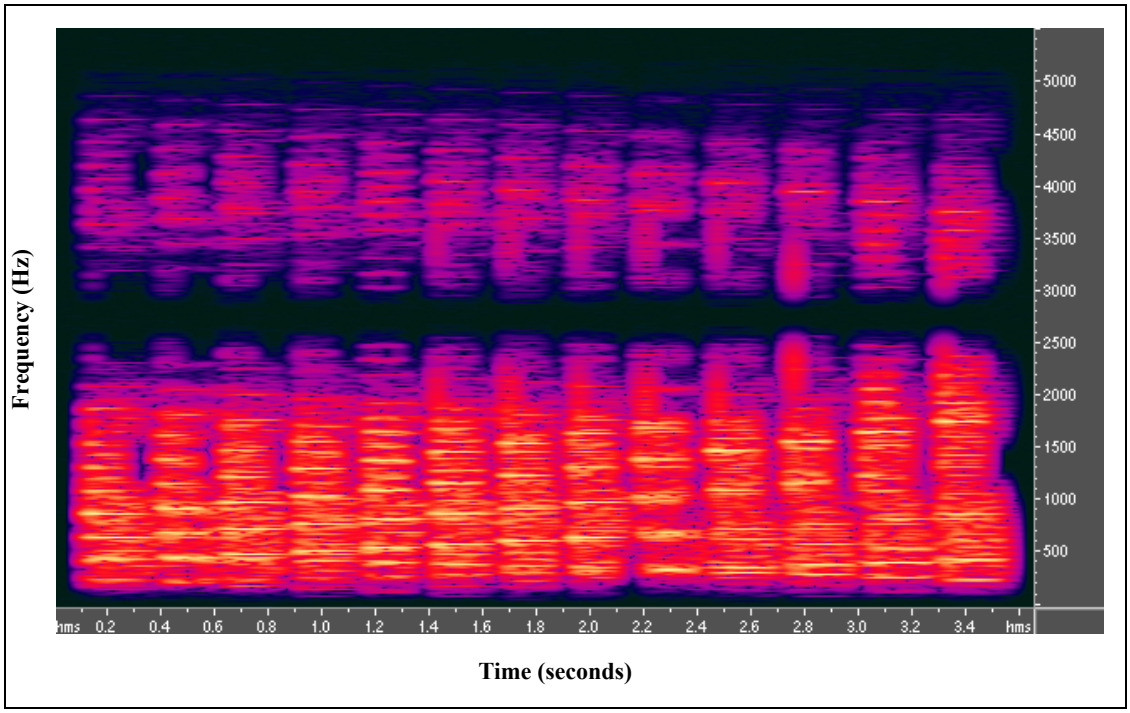

Figure 5-9: Spectrogram of *ScaleThresh2*
(1024 point DFT)

Table 5.4 Summary of Thresholding Sound Examples

Sound Name	Original Sound	Transform / Basis	Modifications
CanThesh1	*Can*	*WPT1 / Complete Tree Basis*	*-Threshold: Soft Global to 0.1474 / Keep DC*
CanThesh2			*-Threshold: Soft Global to 0.2460 / Keep DC*
CanThesh3			*-Threshold: Soft Global to 0.4747 / Keep DC*
DraggingThresh1	*Dragging*		*-Threshold: Soft Global to 0.1833 / Keep DC*
DieselThresh1	*Diesel*		*-Threshold: Hard.* *(4,0): 1.42* *(4,1): 1.02* *(4,3)-(4,15): 0.32*
ScaleThresh1	*Scale*		*-Threshold: Hard Global to 0.2835 / Keep DC*
ScaleThresh2			*-Threshold: Hard Global to 0.6835 / Discard DC*

5.4.1.3 Filtering

With our filtering modifications, we subject one or more of the subband coefficient streams to a filtering operation, separate from the filtering that occurs as part of resynthesis. We have implemented a lowpass

filtering function to test the basic approach, where the order and cutoff frequency are selectable parameters. Our experimentation showed that many effects are possible, even with just the lowpass filter option, since the filtering can be applied selectively to individual subbands with potentially varying parameters.

We found that with a low cutoff frequency and sufficient order to achieve good roll-off, similar to the time-stretching case, we could hear the tonality of the wavelet packets themselves emerge. This emergence of packet tonality makes sense since the more details of the signal coefficient stream we filter out, the less we are left with of the original character of the signal. In the extreme, we are left with a stream of equal-valued coefficients modulating the wavelet packets. In that case, the character of the sound will be entirely that of the wavelet packets.

RainFilt1 is a simple example of this effect, since it was subjected to a 15^{th} order filter with a cutoff frequency of 0.05% of the sample frequency. Given that most of the energy of the original signal is in the 0 - 3 kHz range, the result is a somewhat shrill tonal sound, accompanied by some of the white noise of the original *Rain* sound sample.

DraggingFilt1 shows how the filtering can be applied selectively to only some subbands. In this case, the subbands in the bottom half of the spectrum were left unfiltered and the upper subbands were filtered aggressively with an order 40 filter with a cutoff frequency of 1% of the sample frequency. One can hear the roughness of the original in the lower bands, and the pitched quality of the upper subbands that came about due to filtering. *DraggingFilt2* filters all subbands, but with a slightly less extreme lowpass response. The result is a sound with a similar envelope to the original, but with the grainy roughness replaced with a tonal quality. This example demonstrates an interesting application of lowpass filtering of coefficients: non pitched sound can be converted to a pitched sound while maintaining the large scale characteristics of the time-domain envelope.

DoorsFilt1 is the result of only filtering the lower two subbands. It is a mild but interesting change to the original. *DoorsFilt2* is more extreme, filtering all subbands with the same filter as used for *DraggingFilt2*. Again, a strong pitched component was introduced as a result.

Finally, the sound *Scale* also produced interesting effects. In *ScaleFilt1* the scale is still recognizable, though the timbres have been significantly altered. There are also some descending tones towards the end of the rising scale. These are likely due, at least in part, to aliasing, although it is less clear than in the thresholding case. The other effect here is that by virtue of lowpass filtering the subband coefficient stream,

we are changing its frequency, and therefore the overall frequency of that resynthesized subband will be changed (this is described in *Section 4.2.1.1*). These two phenomena of aliasing and frequency alteration combine to produce potentially significant changes to the signal. *ScaleFilt2* is more extreme in the filtering applied, since a relatively lower cutoff frequency and higher filter order are used. The extreme smoothing that results causes the elimination of parts of some tones, and thereby the creation of a new and somewhat interesting rhythm in addition to the changed tones.

Table 5.5 provides a summary of the test cases discussed.

Table 5.5 Summary of Filtering Sound Examples

Sound Name	Original Sound	Transform / Basis	Modifications
RainFilt1	*Rain*	*WPT1 / Complete Tree Basis*	*-Filter:all nodes:Cutoff = 0.0005f_s Order = 15*
DraggingFilt1	*Dragging*		*-Filter:(4,8)-(4,15):Cutoff = 0.01f_s.Order = 40*
DraggingFilt2			*-Filter: all nodes:Cutoff = 0.05f_s.Order = 20*
DoorsFilt1	*Doors*		*-Filter: (4,0),(4,1): Cutoff = 0.15f_s. Order = 10*
DoorsFilt2			*-Filter:all nodes: Cutoff = 0.05f_s. Order = 20*
ScaleFilt1	*Scale*		*-Filter: all nodes:Cutoff = 0.125f_s Order = 15*
ScaleFilt2			*-Filter: all nodes:Cutoff = 0.025f_s Order = 30*

5.4.1.4 New Coefficients (Granular Synthesis)

Many possibilities are added to the sound palette when we allow for the creation of new transform coefficients. First, there are all the options for generating coefficients within a single subband. Then there are all the possibilities in how such a new subband might be integrated into the overall sound. We may mix it with an existing sound's subband or replace it altogether. In this section, we list a few of the tests performed in this area. We focused in our testing on demonstrating basic characteristics.

The GUI allows for the specification of a coefficient stream for each subband in the wavelet packet tree. As with most of our testing, we work with the 16 subband complete tree associated with a *depth-4* transform. We can specify which of the subbands will receive a granular coefficient stream, as well as the following parameters for each individual stream:

- *stream type*: impulse train, sine wave, square wave, sawtooth wave, or DC
- *wave frequency* (not applicable for DC)
- *peak amplitude*

• *stream envelope shape*: relative rise, plateau, and fall times

In describing our results, we start with sound generated by an impulse train of coefficients. *Gran1* uses the packet associated with subband *(4,1)* as its grain (later we use others), which has a centre frequency of 500 Hz. The shape of the grain corresponding to a given subband can be seen in Fig.*5-5*, where subband *(4,n)* corresponds to packet Wn in the figure. In the case of subband *(4,1)* used in *Gran1*, the corresponding packet (and grain) is shown in the upper right corner and labeled W1. The grain stream that modulates these grains is an impulse train with a frequency of roughly 250 Hz. As stated in *Section 4.2.1.4*, we can view the granular synthesis process using an impulse train as AM in order to understand the resulting spectrum. In the case of *Gran1* we get tonal components at 500 Hz and also side bands a distance of 250 Hz on either side of this. *Imaging* also results in images centred at 1500 Hz, 2500 Hz and 3500 Hz. *Table 5.6* lists the details for this and the other granular synthesis sound examples.

Gran2 - Gran5 go progressively down in pitch from *Gran1*, where all parameters are the same except the impulse train frequency drops down in 50 Hz increments. One can hear the pitch drop steadily.

Gran6 - Gran12 capture the case where a DC stream modulates various subbands. We find in these cases that the pairing of wavelet packet frequencies, as explained earlier when describing the results of *Time Stretching and Contraction*, emerges strongly. For example, the outputs of subbands *(4,1)* and *(4,3)* sound and look the same, as in sounds *Gran6* and *Gran8* respectively. Also note that subbands with a lower fundamental can in some cases sound higher pitched than subbands with higher fundamentals due to imaging. For example, while the fundamental of *(4,1)* is at 500 Hz, it also has images at 1500, 2500 and 3500 Hz (*Gran6*). Subband *(4,5)* on the other hand, has its fundamental at 1500 Hz, with images at 500, 2500 and 3500 Hz (*Gran10*). However, since the image at 3500 Hz of the latter is weaker than the former, *(4,1)* actually sounds higher pitched than *(4,7)*.

With *Gran13-Gran14* we use coefficient streams that are derived from a sine wave. *Gran14* has a warm, bell-like sound. In *Gran15-Gran17* we experimented with a different filter; the shorter, more angular *db2* filter.

We have also tested a few slightly more complex cases. These are summarized near the end of *Table 5.6.* Of particular note are *Gran21*, which is a complex collection of different modifications across the subbands, and *Gran22* and *Gran23,* which show how a longer filter significantly reduces the effects of imaging.

Table 5.6 Summary of Granular Synthesis Sound Examples[1]

Sound Name	Description	Notes
Gran1-Gran5	*Subbands: (4,1). Type: Impulse Train (pos). Envelope:25/50/25. Filter: sym5. f_s = 8kHz/16 = 500 Hz. Frequency: Gran1=0.5f_s. Gran2=0.4f_s. Gran3= 0.3f_s. Gran4=0.2f_s. Gran5=0.1f_s.*	*- (4,1) wavelet packet fundamental frequency ~=500 Hz* *- Coefficient (Grain) Frequency=kf_s (e.g. k = 0.5 for Gran1, so the coefficient stream frequency is 250 Hz)* *- AM results in main spectral lines around 500 Hz with kf_s Hz spaced sidebands* *- Imaging produces the other lines.*
Gran6-Gran12	*Type:DC (pos). Envelope:25/50/25. Filter: sym5. f_s = 8kHz/16 = 500 Hz. Frequency: N/A. Subbands: Gran6=(4,1). Gran7=(4,2). Gran8=(4,3). Gran9=(4,4). Gran10=(4,5). Gran11=(4,8). Gran12=(4,11).*	*- Spectrum fundamental at centre of wavelet packet frequency.* *- Images add complexity.*
Gran13-Gran14	*Subbands: (4,1). Type:Sine Wave. Envelope:25/50/25. Filter: sym5. f_s = 8kHz/16 = 500 Hz. Frequency: Gran13=0.1f_s. Gran14=0.3f_s*	*- No fundamental due to DSB-like modulation (carrier suppressed).* *- Only sidebands and their images.*
Gran15-Gran17	*Envelope:25/50/25. Filter: db2. f_s = 8kHz/16 = 500 Hz. Frequency: 0.3f_s. Other: (1)Gran15:Type=DC (pos), Subbands=(4,2). (2) Gran16:Type=Impulse Train (pos), Subbands =(4,1). (3)Gran17:Type=Sine Wave, Subbands=(4,1)*	*- dB2 filter case* *- Compare Gran 15 to Gran7* *- Compare Gran 16 to Gran3* *- Compare Gran 17 to Gran14* *In all cases, the sym5 filter case sounds warmer, since the filter is smoother.*
Gran18	*Subbands: (4,0)-(4,3). Type:Triangle Wave. Envelope:25/50/25. Filter: sym5. f_s = 8kHz/16 = 500 Hz. Frequency: 0.05f_s.*	*- Low frequency triangle wave for coefficients* *- Cat purr sound*
Gran19	*Subbands: all nodes. Type:Impulse Train (pos). Envelope:25/50/25. Filter: sym5. f_s = 8kHz/16 = 500 Hz. Frequency: 0.05f_s.*	*- Low frequency impulse train for coefficients* *- Sparser than Gran18.*
Gran20	*Subbands: (4,0)-(4,3). Type:Sine Wave. Envelope:25/50/25. Filter: sym5. f_s = 8kHz/16 = 500 Hz. Frequency: 0.05f_s.*	*- Low frequency sine wave for coefficients* *- More tonal than Gran 18*
Gran21	*Filter: sym5. f_s = 8kHz/16 = 500 Hz.* *(1) (4,1) Type:DC (pos.,boosted 8x). Frequency: N/A. Envelope:15/5/80* *(2) (4,2),(4,3). Type:Sine Wave. Frequency: 0.14f_s. Envelope:15/5/80* *(3) (4,0),(4,6),(4,5),(4,12), (4,15),(4,10),(4,9). Type:Sawtooth Wave. Frequency: 0.075f_s. Envelope:85/5/10* *(4) remaining nodes. Type:Impulse Train (pos). Frequency: 0.075f_s. Envelope:85/5/10*	*-Complex.*

Table 5.6 Summary of Granular Synthesis Sound Examples[1]

Sound Name	Description	Notes
Gran22	*Subbands: (4,1). Type:Impulse Train (pos). Envelope:25/50/25. Filter: db10. f_s = 8kHz/16 = 500 Hz. Frequency: 0.5f_s.*	*-Compare to Gran1. The longer filter used here significantly reduces the images.*
Gran23	*Subbands: (4,1). Type:Impulse Train (pos). Envelope:25/50/25. Filter: db10. f_s = 8kHz/16 = 500 Hz. Frequency: 0.1f_s.*	*-Compare to Gran5. The longer filter used here significantly reduces the images.*

1.All sounds are at a sample rate of 8 kHz.

5.4.2 Resynthesis Modifications

5.4.2.1 New Filter Resynthesis

With our *new filter resynthesis* modifications, we select a different wavelet packet filter for synthesis than the one implied by the analysis filter that is necessary for perfect reconstruction and alias cancellation. We did not find the results of our testing with these modifications particularly interesting. While a change to the sound of the resynthesized signal can be achieved, it tends to either be quite subtle, or quite noisy.

For example, *FlyNFR1* is the fly sound analyzed with our standard *sym5* filter, and resynthesized with the short *Haar* (or *db1*) filter. It is quite harsh sounding relative to the original *Fly.* On the other hand, when we resynthesize with the longer *db10* filter (20 coefficient FIR filter) as in *FlyNFR2*, the result is much smoother, though the difference to the original is not very significant. *FlyNFR3* demonstrates that even resynthesizing with a longer filter can create a noisier result. *FlyNFR4* reverses the analysis and resynthesis filters of *FlyNFR3*, which results in a small change in the sound.

KnockNFR1 provides a slightly more interesting example. Resynthesis with a longer *db10* filter in this case results in a greater variation from the original, due in part to more smearing of the sharp attack onsets associated with the source sound. *KnockNFR2* uses the short *Haar* (*db1*) filter, resulting in a slighter change to the sound.

Table 5.7 summarizes the results. It is likely that greater differences would occur if very large length differences between analysis and resynthesis filters were used. The largest length difference we tested with was 18, which occurs between the *db1* and *db10* filters.

Table 5.7 Summary of New Filter Resynthesis Sound Examples

Sound Name	Original Sound	Transform / Basis	Analysis Filter	Synthesis Filter
FlyNFR1	*Fly*	*WPT1 / Complete Tree Basis*	*sym5*	*Haar (db1)*
FlyNFR2				*db10*
FlyNFR3		*WPT4 / Complete Tree Basis*	*Haar (db1)*	*db10*
FlyNFR4		*WPT5 / Complete Tree Basis*	*db10*	*Haar (db1)*
KnockNFR1	*Knock*	*WPT1 / Complete Tree Basis*	*sym5*	*db10*
KnockNFR2				*Haar (db1)*

5.4.2.2 Arbitrary Sound Cross Synthesis

With these modifications, we select as our resynthesis packet filter the sampled waveform of a given sound, so that upon resynthesis we *cross* the analysis transformed sound with that used for the resynthesis filter. Given the much greater freedom in choice of synthesis packet versus the *new filter resynthesis* approach described in the previous section, it is not surprising that our results here have tended to be significantly more interesting[†].

We describe our testing with three different cross synthesis packets: *bell64, flyl128*, and *diesel1024*. *bell64* is a 64 sample segment of a bell-like sound with a fundamental frequency of 250 Hz., *flyl128* is a 128 sample segment of the test sound *fly*, and *diesel1024* is a 1,024 sample segment of the test sound *diesel*. The shape of each of these packets is shown in Fig.*5-10*.

DraggingCross1 provides an example of how the new packet changes the sound of the original. The rough noisy sound of *Dragging*, when cross synthesized with the bell sound *bell64*, becomes more pitched sounding, while still preserving the larger scale shape or envelope of the sound. *DraggingCross2* provides a slight but interesting variation, where now the fly sound *flyl128* is used for resynthesis. If instead the much longer and noisier *diesel1024* is used with the *Dragging* sound, the original envelope and sound are largely

---

[†] We use the term *packet* loosely in this section, since the waveforms used for resynthesis are not wavelet packets in any strict sense. Rather, they are serving the function performed by the wavelet packet during a normal resynthesis operation.

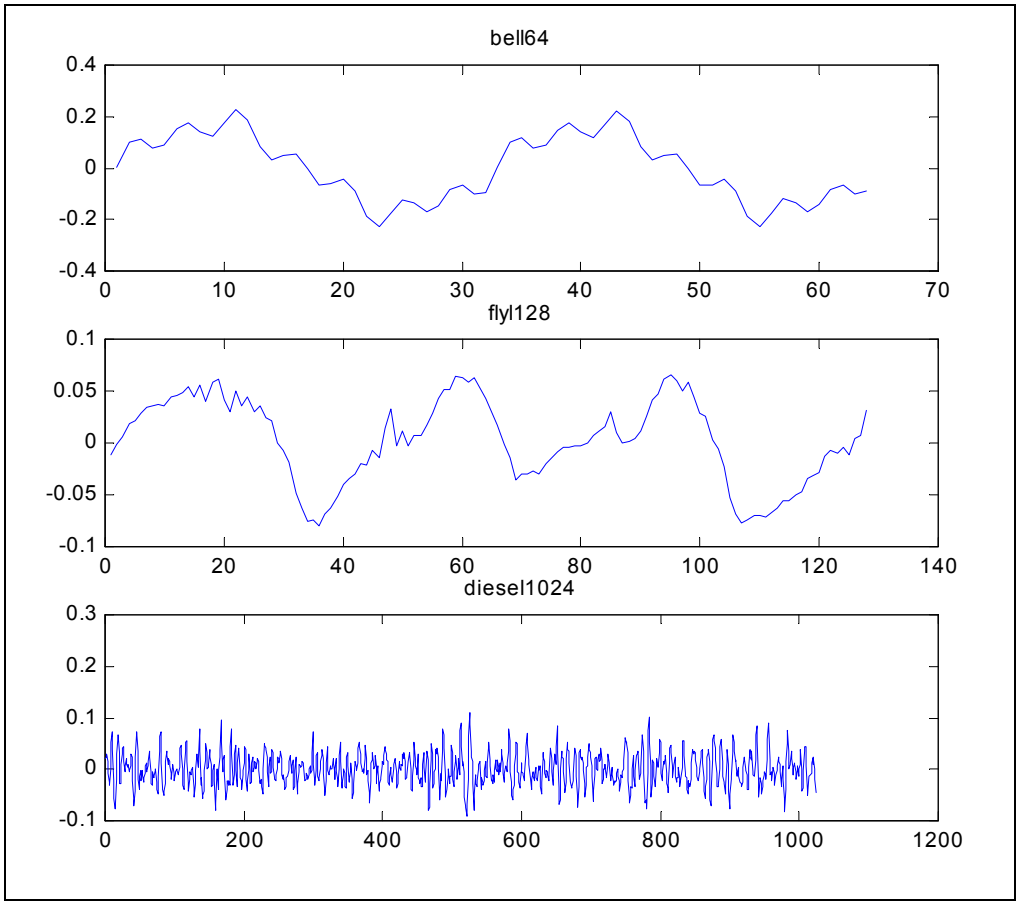

Figure 5-10: Impulse Responses of Cross Synthesis Packets

destroyed. This case is captured in *DraggingCross3*. The change in the envelope should be expected given the length of the resynthesis packet relative to the time structure of the original sound, where significant envelope changes occur over 100-500 sample durations. A packet of length 1024 samples is bound to significantly blur these events.

We can compare the ineffectiveness of using *diesel1024* with the *Dragging* sound in the last example (*DraggingCross3)* with its use on a longer more spread out sound. *DoorsCross1* shows the result of *Doors* cross synthesized with *diesel1024*. Here the envelope is largely preserved because the time scales (in samples here) of *Doors* are much longer. The longer time scale is due to both a less dense sound, and a much higher sample rate (32 kHz versus 8 kHz). *DoorsCross1* uses *bell64* instead for resynthesis, creating a much sparser, wispier sound.

FlyCross1 - FlyCross3 show the cross synthesis of *Fly* with itself, *diesel1024*, and *bell64*, respectively.

BirdCross1 shows another example of cross synthesis with *bell64.*

All the test sounds are summarized in *Table 5.8*.

Table 5.8 Summary of Cross Synthesis Examples

Sound Name	Original Sound	Transform / Basis	Cross Filter Sound
FlyCross1	*Fly*	*WPT1 / Complete Tree Basis*	*flyl128*
FlyCross2			*diesl1024*
FlyCross3			*bell64*
DoorsCross1	*Doors*		*diesl1024*
DoorsCross2			*bell64*
BirdCross1	*Bird*		*bell64*
DraggingCross1	*Dragging*		*bell64*
DraggingCross2			*flyl128*
DraggingCross3			*diesl1024*

5.4.3 Post-resynthesis Modifications

5.4.3.1 Scaling Subband Amplitudes

With these modifications, we scale the amplitudes of some or all resynthesized subbands prior to combining them for playback. In the extreme, we can scale their amplitudes to zero, effectively leaving out those subbands. We have found that subband amplitude scaling or *equalization* provides many opportunities for modifying the sound in interesting ways. We show a few examples here.

First we consider the extreme case where a subband is either scaled to zero, or not scaled at all. In other words, we either exclude or include each of the subbands in the final mix. One way of demonstrating the changes that occur as subbands are brought into and out of the mix is to loop the sound sample repeatedly while successively including or excluding various subbands. This looping approach is taken in the *Stairs2EQ1*, *BirdEQ1,* and *FlyEQ1* test cases.

In *Stairs2EQ1*, we start with no subbands in the mix, and progressively add them in until the sound is completed, with volume kept roughly constant throughout. In *BirdEQ1* we start with the complete signal, eliminate subbands until all are out, and then bring them back in, again with normalized volume. In both cases, there is no particular order to our adding and removing subbands. In the latter case, while adding a

series of subbands into the mix, we even reverse the process and remove one along the way, adding it again later.

In *FlyEQ1*, we again start with silence and progressively add subbands into the mix. However, in this case volume is not normalized, and subbands are added from highest frequency content to lowest frequency content. Also, we slightly re-formed the basis by merging several subbands so that each of our changes would have a more noticeable impact. Without this basis change, since the volume was not normalized in this test case, the higher frequency subbands of the original complete tree basis were either not audible, or at least not significant in their contribution to the overall sound.

The last four examples involve scaling of various amplitudes, without looping. *LakeEQ1* boosts a single subband in the 2.5 - 3 kHz range to achieve an interesting brightening of the overall sound, while *LakeEQ2* boosts 500 - 1,000 Hz and cuts 0 -500 Hz and 1 - 1.5 kHz. *KnockEQ1* uses the PTOB basis, and is equalized such that the lower and higher frequencies are boosted relative to the middle. The boost to the low end is especially noticeable. In *KnockEQ1* we only boost the upper end of the spectrum (2 - 4 kHz), thinning out the sound considerably.

The results are summarized below in *Table 5.9*.

Table 5.9 Summary of Subband Scaling Sound Examples

Sound Name	Original Sound	Transform / Basis	Modifications
Stairs2EQ1	*Stairs2*	*WPT1 / Complete Tree Basis*	*-EQ: Successive enabling of subbands, in no particular order (volume normalized)*
BirdEQ1	*Bird*		*-EQ: Successive disabling of subbands in no particular order, followed by successive enabling - with 1 enable (volume normalized)*
FlyEQ1	*Fly*	*WPT1 / Nodes (4,0), (4,1), (4,2), (4,3), (4,4), (4,5), (3,3), (1,1) selected (EBBA used)*	*-EQ: Successive enabling of subbands, starting from highest frequency and proceeding to lowest (volume not normalized)*
LakeEQ1	*Lake*	*WPT1 / Complete Tree Basis*	*-EQ: (4,7) boosted 10x*
LakeEQ2			*-EQ: (4,0) cut to 0.1x, (4,1) boosted 4.1x, (4,3) cut to 0.1x*
KnockEQ1	*Knock*		*-EQ: (6,0)=8.1x, (6,1)=7.0x, (6,3)=6.2x, (6,2)=6.2x, (6,6)=4.8x, (6,7)=3.8x, (5,2)=2.6x, (5,6)=1.8x, (5,7)=1.8x, (4,2)=1.8x, (4,6)=2.9x, (4,7)=4.0x, (3,2)=5.6x, (3,6)=6.4x, (3,7)=6.4x, (2,2)=10.0x*
KnockEQ2			*-EQ: (3,6)=7.5x, (3,7)=8.9x, (2,2)=9.2x*

5.4.3.2 Modulating Subband Amplitudes

Rather than applying a fixed scale amount to a given subband, with these modifications we *modulate* a subband's amplitude over time in some prescribed way.

A fundamental choice concerns the frequency of the modulation relative to the frequencies in the band being modulated. If the modulating wave frequency is much lower than that of the subband, one can easily see and hear the large scale oscillations that are imparted on the sound. This scenario is demonstrated in *FoghornMod2* and *FoghornMod3*. The 0.00025 Cycles per sample modulation of the 8 kHz sample rate sound result in a 2 Hz modulation. These examples are the same except the former modulates with a sine wave, and the latter uses a sawtooth wave. Both employ AM, thereby raising the modulating waveform above the zero axis.

The other two examples utilize higher relative modulation frequencies, thereby creating more complex sounds. *FoghornMod1* applies an 50 Hz (0.1 cycles per sample at 8 kHz/16) DSB modulation to subband (4,1), which is centered on frequencies in the 250 - 500 Hz range. One can hear the higher frequency

components that result from the modulation. *FoghornMod4* is more complex, whereby a series of subbands have all been DSB modulated. The modulation frequency scales with the frequency of the subband. The frequencies used are provided in *Table 5.10*, which also summarizes the other results.

Table 5.10 Summary of Subband Modulation Sound Examples

Sound Name	Original Sound	Transform / Basis	Modifications
FoghornMod1	*Foghorn*	*WPT1 / Complete Tree Basis*	*-Modulation: DSB Modulation with sine wave: (4,1) = 0.1 Cycle/Sample.*
FoghornMod2			*-Modulation: AM Modulation with sine wave: (4,1), and (4,3) = 0.00025 Cycle/Sample.*
FoghornMod3			*-Modulation: AM Modulation with sawtooth wave: (4,1), and (4,3) = 0.00025 Cycle/Sample.*
FoghornMod4			*-Modulation: DSB Modulation with sine wave: (4,0) = 0.0625 Cycle/Sample, (4,1) = 0.125 Cycle/Sample, (4,3) = 0.1875 Cycle/Sample, (4,2) = 0.25 Cycle/Sample, (4,6) = 0.3125 Cycle/Sample, (4,7) = 0.375 Cycle/Sample*

5.5 Basis Selection Test Results and Observations

We now describe our *basis selection* results. Our focus is on the BDBBA with *all-or-nothing* quantization (*Section 4.4.2*), however, we also describe our testing of the PTOB, the EBBA and TBBA, and the use of other quantizer choices with the BDBBA.

5.5.1 PTOB

The *pseudo-third-octave basis* (PTOB) is a fixed basis whose subband structure approximates the *third-octave* spacing typical of many graphic equalizers. Such spacing is also much better aligned to the *critical bands* of human hearing than either the WT basis or the complete tree basis of the WPT (see also *Section 4.3*).

Fig.*5-11* shows a typical PTOB coefficient energy plot. Since we will see several of these coefficient energy plots, we first describe its format in some detail. Time in units of samples progresses along the horizontal axis, and the subbands that make up the basis are stacked on top of each other proceeding from the lowest frequency subband at the bottom to the highest frequency subband at the top (i.e., the *frequency ordering* of subbands). Each subband is labelled with its number in *depth/position* format and the frequency range it occupies. The energy of each transform coefficient is indicated in the plot by its colour, which can be related to a decibel value with the colour bar to the right of the plot. The thickness of a given subband is

directly proportional to its bandwidth. For example, node (2,2) occupies one quarter of the height of the plot to correspond to the one quarter of the total bandwidth that it occupies (i.e., 1 kHz of the 4 kHz total). Note also that the time and frequency resolutions of the subbands follow the expected inverse proportionality. Comparing node (2,2) with the narrow bandwidth nodes at the bottom of the plot, for example, we can see how the time resolution of the former is much finer. With each deeper level in the basis, the bandwidth precision doubles and the time precision halves.

We can now describe the PTOB-specific aspects of Fig.*5-11*. Since the sample rate of 8 kHz is one quarter of the 32 kHz sample rate used to derive the bandwidths in the PTOB example given earlier in *Section 4.3.2*, and further since the PTOB of that example was of *depth-9*, we use a *depth-7* PTOB here to obtain the same bandwidths as in that example. Even a plot of this depth becomes cluttered and difficult to read, especially the textual subband information along the vertical axis. The text associated with the *depth-9* plot is unreadable without some simplification of the textual information.

Usage of the PTOB is simple. Given its fixed nature, there are no parameters required. Our subjective testing has shown that the PTOB can indeed be quite useful in providing a psychoacoustically appropriate time-frequency partitioning of the signal. For example, applying a *depth-7* PTOB to the *Knock* signal, we found that alternately boosting the gain by 10 dB on each of the subbands going up in frequency while keeping the others at the original levels seemed to produce a reasonable sweep up through the frequencies. Changing each subband results in a noticeable change to the sound, except the lowest subband (7,0), which we could not hear (0-31 Hz). The same impact per subband that we achieved with the PTOB is not heard for all subbands when adjusting the levels of the subbands in the WPT complete tree basis. Given that the best basis approaches partition the sound based on the characteristics of the sound and not the listener, they too may at times produce subbands that are too wide or narrow from a psychoacoustically optimal perspective.

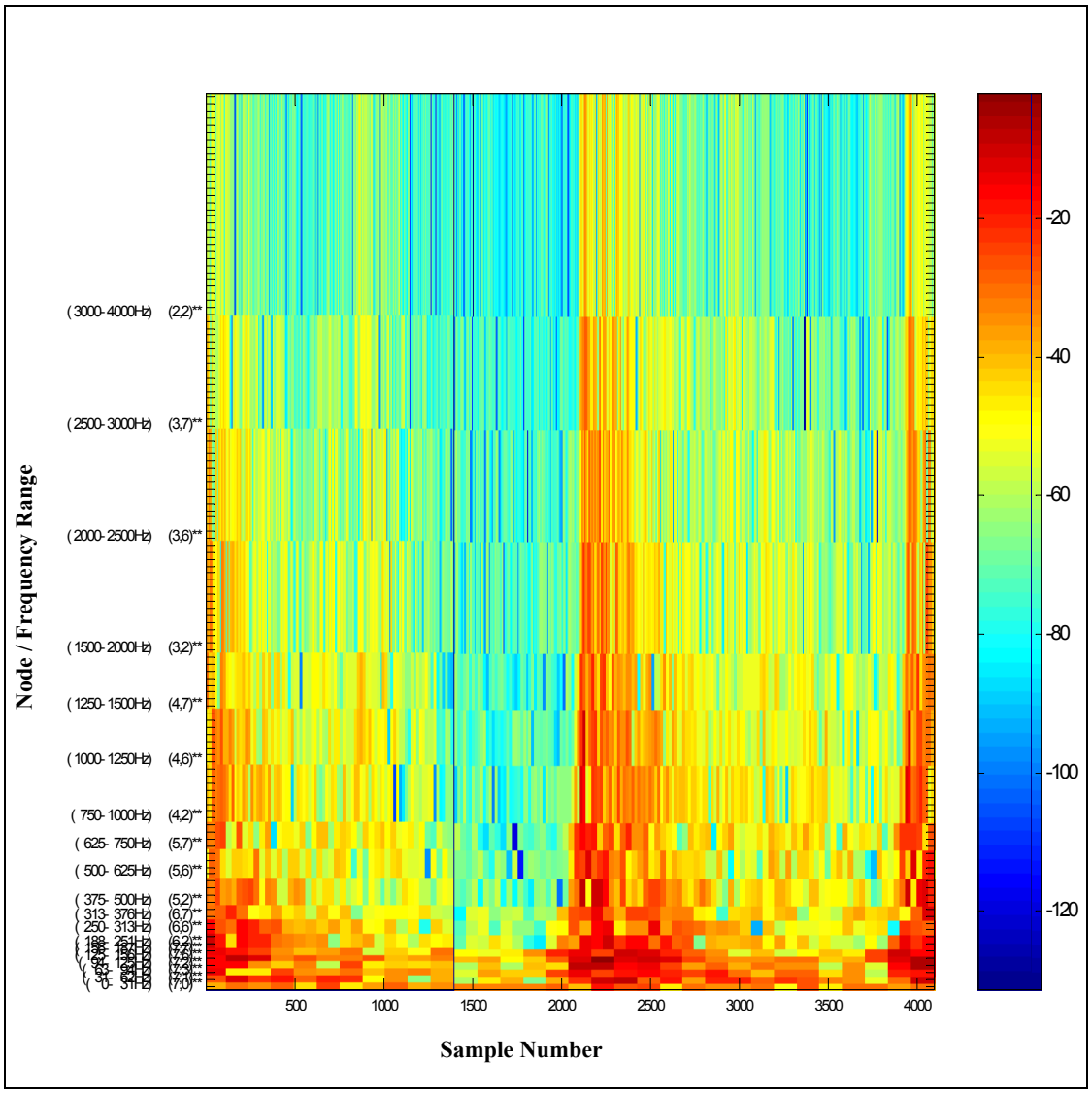

Figure 5-11: WPT Energy Plot (db): Depth-7 PTOB of Knock signal
(node number and frequency range on vertical axis)

5.5.2 BDBBA

Here we describe our test results in working with the *Bi-rate Distortion Best Basis Algorithm* (BDBBA). Recall that this algorithm attempts to derive a basis from the complete tree basis of the WPT that is best in the sense of minimizing the added distortion for a given amount of quantization. The objectives of performing such quantization include the finding of a subband structure that maps well with the energy

distribution of the signal, reducing the number of subbands the user must work with and the computer must resynthesize, abstracting the key features of the signal, and simply imparting changes to the signal that might be aesthetically interesting. Most of our testing used the *all-or-nothing* quantization case, where subband coefficients are either kept without any effective quantization, or are zeroed. Most of our description in *Section 4.4* focused on this case. However, we also did perform some testing with other quantizer values, and include some observations regarding these cases later in *Section 5.5.2.4*.

5.5.2.1 Usage

There are two modes of operation of the BDBBA along with several additional parameters. The modes of operation are *Bandwidth Mode* and *Node Mode*. In *Bandwidth Mode*, the algorithm finds the lowest distortion basis for a given *Bandwidth Budget*. This budget specifies what percentage of the signal, spectrally, should be retained. In *Node Mode*, the user also specifies a subband or terminal *Node Budget*. This budget specifies the number of subbands the user would like in the chosen basis, and the algorithm attempts to find such a basis. The example *Basis Selection* window in Fig.*5-3* shows the different mode and parameter selection options. Choice of *Bandwidth Budget* (*Quality* in the figure) and *Node Budget* depend on the mode of operation and the particular goals of the user. Their selection has a significant impact on the basis that results.

The other main parameters are for choice of whether or not psychoacoustically based weighting of distortion is used, whether to keep all nodes or not (*lossless* versus *lossy* operation), and whether or not splitting of nodes should occur in the case of cost equality with child nodes. When we are not using *all-or-nothing* quantization we must also select the quantizers used, including the bits (or number of levels) assigned to each quantizer, and whether quantization levels are uniform across all subbands or are subband specific. Our test results in the following are grouped according to which mode was used, *Bandwidth* or *Node*.

Each of the main aforementioned parameter choices are briefly discussed before moving on to the *Bandwidth Mode* and *Node Mode* results.

Mode

The *Node Mode* of operation will often be the most useful to the user, especially when achieving a basis with a given small number of terminal nodes is of primary importance. In comparison to *Bandwidth Mode*, there is also a clearer connection between the setting of a parameter and the nature of the result. The user requests

a basis with a certain number of nodes, and the algorithm does its best to find one. However, if the user has other motives, such as reduction of bandwidth or a reasonable mapping of basis subbands to the time-frequency structure of the sound, then *Bandwidth Mode* may be more suitable.

Note also that since *Node Mode* is layered on top of the bandwidth loop of *Bandwidth Mode*, use of either mode uses the *Bandwidth Mode* functionality. As such, in both modes the user must still choose a bandwidth budget, even though the strategy for choosing the budget will generally differ between the two modes. We discuss both cases in the corresponding subsections below, after a brief discussion of the other parameter settings.

Bandwidth (Quality) Budget

In the GUI, this budget parameter is labelled *Quality* to cover the more general case where arbitrary quantizer choices are possible. For the particular case of *all-or-nothing* quantization, we refer to the parameter as the *BW Budget*. The user interface permits selection from any of the possible budgets. Given a *depth-4* complete tree for example, since there are 16 subbands the bandwidth resolution is 1/16, or 6.25%. Therefore 15 budgets are possible, ranging from 6.25% to 93.75% in steps of 6.25%[†]. Budgets of 0% and 100% are of no practical interest.

Several approaches are possible in the bandwidth budget selection. Here we discuss the issues, first focusing on *Bandwidth Mode* operation, and then seeing how that might also apply to the selection of the *BW Budget* in *Node Mode*.

One approach would be to choose the extreme minimum or maximum budget values, or perhaps to retain one half of the bandwidth. Or it might be that the user has a specific goal in mind that is related to bandwidth reduction, such as reducing storage needs, speeding resynthesis time, or simple curiosity regarding what the signal will sound and look like with a given amount of bandwidth retained. These and many more choices can all be legitimate.

However, in the absence of specific bandwidth related goals, the previously listed choices would appear somewhat arbitrary, even if they result in interesting and useful bases. Short of operating in *Node Mode*,

---

[†] By restricting choices to possible budget values, the user is made clear on possible values and not mislead. Also, sometimes we would get a solution point one to the left (in bandwidth) of the budget, which is a resolution issue. Solution values cannot have a bandwidth greater than the budget, so if even infinitesimally larger, we get the next lower operating point. As a practical solution to avoid this we add 0.01% to the budget, so the solution point will be to left of budget, and therefore accepted. This eventuality would almost never happen if we didn't force selected budgets to be operating point values in the first place.

what approach to choosing the bandwidth budget might we justify?

Imagine that most of the energy in the discrete time signal resides in roughly 40% of the total bandwidth, as dictated by the signal's sample rate. Directing the BDBBA to find a basis that retains only 6.25% of the bandwidth, which is the equivalent of one full depth subband for a *depth-4* transform, will clearly result in significant distortion of the signal. On the other hand, directing the BDBBA to retain 75% of the signal bandwidth will not motivate the algorithm to discern components of the signal that are localized in frequency. This lack of attained frequency resolution is because the algorithm splits a subband on the basis of there being a sufficiently large (weighted) energy differential between its child subbands and the cost of keeping a subband being high (which it will not be when a high budget is used). Keeping too much bandwidth might be likened to insisting a sculptor retain too much of the original stone, and thereby being unable to etch out the details of a face.

Consideration of the shortcomings with choosing too little or too much bandwidth leads us to the notion of choosing an amount of bandwidth that would capture most of the signal energy. We have defined one option called the *95% Energy Bandwidth*, which captures 95% (at least) of the signal energy. We calculate this basis by ranking the energy of the full depth subbands from maximum to minimum and then, starting at the maximum energy subband end of the list, adding up the subbands and corresponding bandwidths until we have captured 95% of the energy. The total bandwidth associated with these subbands is the *95% Energy Bandwidth*.

We chose 95% as a good compromise between insisting on capturing all of the energy, which would often require all the bandwidth to account for even minute amounts of energy in the lesser subbands, and capturing too little of the energy to preserve the sound. By using this 95% bandwidth amount, we should retain most of the audible characteristics of the sound. If, for example, a signal is comprised of a single narrow band of energy, then the *95% Energy Bandwidth* should be roughly that of one subband, such as 6.25% in the case of a *depth-4* transform. Setting the bandwidth budget to this amount will allow us to isolate that band. If there are two narrow bands with a low energy band in between them, using a budget corresponding to two subbands (12.5% in this case) would allow the BDBBA to isolate them as well. If on the other hand the signal is broadband, such as white noise, attempting to find a basis for it with 12.5% of the bandwidth makes little sense, unless it is done with the specific intention of sound modification.

A specific example of the usefulness of the *95% Energy Bandwidth* is given by the chirp sound

NoiseChirpBMode1. Since its frequency sweeps through all subbands, there will be little justification in deleting one subband over another. Most, if not all, of the subbands are important. We are guarded here by selecting the *95% Energy Bandwidth*, since all bands are of roughly equal importance and so the *95% Energy Bandwidth* turns out to be almost all of the bandwidth (93.75% in this example). Another example showing how effectively the *95% Energy Bandwidth* works is provided by *FoghornBMode1*. Both these cases are included in the summary of Bandwidth Mode test cases provided later in *Table 5.13*.

In *Node Mode*, we must still supply a *Bandwidth Budget*, even though the *Node Budget* takes precedence. The specified *Bandwidth Budget* is a starting point for the algorithm, and indicates the user's preference. The bandwidth associated with the solution will generally differ from the *Bandwidth Budget* specified. However, this budget is still important since it will often bias what type of basis gets chosen, since there can be more than one solution with the same number of nodes.

Choosing a small bandwidth budget will tend to favour a basis with narrow subbands for a given node budget, while choosing a large bandwidth budget will favour wider bands. As such, one strategy for *Node Mode* is to specify the desired *Node Budget* and either the maximum or minimum *Bandwidth Budget*. In the former case, one gets the basis as close as possible to the node budget supplied that has the maximum retained bandwidth. In the latter case one gets the basis with the minimum retained bandwidth.

Alternatively, one may want to start with the *95% Energy Bandwidth* and seek a basis as close to that as possible. In this case, we want a basis with a specified number of nodes that captures as close as possible the bandwidth that comprises most of the signal. This approach also seems reasonable and works fairly well, especially if the specified number of nodes corresponds to a basis in the vicinity of the *95% Energy Bandwidth* basis.

Node Budget

Specification of the *Node Budget* is simple. One simply chooses the number of nodes one is willing to work with. If the specific application involves real-time manipulation of the nodes, a small *Node Budget* would make sense, both in terms of user ease of interaction, and processing time. If real-time is not required, a larger number of nodes may be permissible, even if the user does not want to work with all the subbands of the complete tree basis.

Psychoacoustic Weighting of Distortion

We would expect that for general use, psychoacoustically based weighting of the cost calculation should be used. We have found it has a noticeable effect on the basis chosen in many cases. The particular effect it has depends on both the sample rate of the signal and the signal's spectral characteristics relative to that of the weighting curve.

Psychoacoustic weighting is selected in the GUI via the *Weighting* parameter. We have not included it in many of our tests simply to limit the number of variables in those tests and better isolate the particular function or parameter being studied. *Appendix D* provides some test results showing the difference in basis that occurs with and without the weighting.

Lossy versus Lossless Operation

Regarding *Lossless* versus *Lossy* operation, for testing purposes we have normally operated in *Lossy* mode. This mode is selected in the GUI by setting *Keep All Coeffs* to *No*. In actual operation, we often use *Lossless* mode, since we can choose to include or exclude these subbands later in the Resynthesis window (Fig.*5-2*) via the setting of *Play Nodes* to either *All* or to *Kept Nodes,* as desired.

No Split on Equality

Regarding the choice to split or not split in the case of equality between parent and child costs, we choose to not split. The reasoning is described in *Pruning the Tree* of *Section 4.4.2.2*. In the GUI, this is selected by setting *Type* to *No Split if* <=.

5.5.2.2 Node Mode

The *Node Mode* of operation is essentially an outer loop wrapped around the *Bandwidth Mode* loop. It finds the best basis for the number of nodes specified by invoking the inner bandwidth loop with different bandwidth budgets until a solution for the node budget is found.

We mentioned in *Section 4.4.2.1* several motivations for use of a lossy basis selection algorithm like the BDBBA. The first two listed had to do with ease of use, especially in real time, and feasibility of operation in real time. Specifically, by reducing the number of nodes the user is confronted with, the BDBBA affords the user the opportunity to work in real time, where the precious resource of the user's hands are put to good use due to the extraction of nodes that are important to the overall sound. And by allowing the nodes to be not only few, but also narrow, processing times are reduced, and the possibility of working in real time is more likely for more functions and on more platforms. A third reason listed was that it gives the user a measure of

control, especially relative to an approach like the *EBBA*, where no user input parameters are used.

Node Mode answers the collection of needs listed above fairly well. Primarily, we seek a basis with a certain number of nodes[†]. This mode answers our user interface need. However, we also may have bandwidth based motives. Most likely, we will want to minimize the change to the sound in going to the new basis. In that case, choosing the *95% Energy Bandwidth* makes sense as a starting point, even though the resulting bandwidth can end up being different. In many cases, this will result in a significant bandwidth reduction, and, of course a large node reduction. Each modification imparted on a node with such a reduced basis will have a noticeable impact, and can occur more quickly. There are two reasons why such modifications are generally more noticeable. First, high impact nodes are found and isolated. These nodes might be at full depth in the tree, or they may be wider bands higher up in the tree, created through aggregation of several lower energy nodes. Second, the least important nodes are dropped altogether, resulting in less energy for the kept nodes to compete against.

Imagine working with this reduced basis versus working with and resynthesizing the complete tree basis for 16 subbands or more. It is not surprising that the reduced basis is more manageable, and we compare with two test cases in the following.

Stairs1NMode1 is the result of using *Node Mode* to find a four node basis for the signal *Stairs1*. The specific parameters and results for this and other cases described in this section are summarized in *Table 5.11* at the end of this section. This basis retains exactly one half (50%) of the subband bandwidth, which is close to the *bandwidth budget* requested of 43.75% (*95% Energy*). The result sounds close to the original, with just a bit of the crispness of the higher frequencies lost. To demonstrate the utility of forming such a basis, we will compare the effectiveness of working with this basis versus working with the original, and will also look at what result the *EBBA* would provide.

Stairs1NMode2 is an audio segment capturing the looping of *Stairs1NMode1*, whereby modifications are imparted on that signal over time[††]. The corresponding entry in *Table 5.11* provides the details. In particular,

---

[†] It would also be useful for the user to be able to specify a range of acceptable nodes, thereby increasing the likelihood that the basis will also meet the bandwidth budget, or at least come fairly close. Unfortunately, however, this feature is not currently implemented.

[††] Some minor editing of the looped track produced by the GUI was done in order to even out the silences between loops. This is because some of the cross synthesis operations were too slow (barely) on the machine used (200 MHz Pentium/ 128 M RAM) to occur in real -time (i.e., they took longer than the time to play one time through the loop). This is not generally a problem on faster machines (such an 870 MHz Pentium with 256 M RAM), where we have noted much better user response.

the chronology of modifications are provided in the *Modifications* column. For example, the entry *"Cross Synthesis: (2,0)/bell64/Loop 5"* indicates that subband *(2,0)* was cross synthesized using the *bell64* 'filter' starting at the 5th looping of the basic *Stairs1* sound. One can hear the change when listening to the corresponding audio. In fact, all the changes indicated can be easily heard in the track. Not only do we have a manageable number of nodes to work with, but the modification of each of the subbands, as demonstrated in that example, has a noticeable effect on the sound.

We now compare the case of *Stairs1NMode2* with that of using the complete tree, as given in *Stairs1CT1*. In this latter case, we have 16 subbands that can be modified. We start adding changes to the subbands, targeting those that look like they have the most energy first. It takes five modifications before anything other than very subtle changes can be heard. In other words, the basis is not particularly responsive to individual user inputs. Several changes must occur before the cumulative effect can be heard. The exception will be sounds that have most of their energy concentrated in few subbands, in which case a greater impact may occur in modifying one subband. However, even in that case, a more concentrated basis will not hurt, and will still be more manageable.

If instead we run the *EBBA* (Shannon Entropy), the basis is reduced to 11 nodes. This is still a large number of nodes for the user to cope with, and again, the changes per node are much subtler.

A second example is provided by *DoorsNMode1* and *DoorsNMode2*. *DoorsNMode1* is a three subband best basis, and *DoorsNMode2* is that signal with (2,0) cross synthesized with *diesl1024*. Again, it is convenient to work with fewer nodes. By comparison, the *EBBA* only reduces the basis to 13 nodes.

A third example is provided by *FlyNMode1*. The basis chosen has the nodes (3,0), (4,3), (5,5) and (5,13) for the requested four nodes associated with the node budget. The original basis is *depth-5* in this case, and the slightly longer *sym8* filter was used. For a *depth-5* basis, the complete tree is comprised of 32 subbands; even more motivation to find a new and smaller basis. *FlyNMode2* demonstrates the effect of modifying two of the four chosen subbands.

As we stated earlier, in node mode we must still supply a bandwidth budget, and that choosing a small bandwidth budget will tend to favour a basis with narrow subbands for a given node budget, while choosing a large bandwidth budget will favour wider bands. Thus far we have worked with the *95% Energy Bandwidth* budget. In the following, we show typical results when working with either maximum and minimum bandwidth starting points.

We compare the test cases *BirdNMode1* and *BirdNMode2*. Both bases are derived with a Node Budget of four. The former started with a specified bandwidth budget of 6.25%, and the basis that resulted has a bandwidth of 31.25%. The latter started with a specified bandwidth budget of 93.75%, and the basis that resulted has a bandwidth of the same (93.75%). Both have four terminal nodes or leaves, so the smaller budget case *BirdNMode1* must have some narrower subbands. Its nodes are (4,0), (4,10), (4,13), (4,15), whereas the nodes of the larger bandwidth budget case *BirdNMode2* are (1,0), (4,9), (3,5), (2,3). While the latter does retain more of the bandwidth and sound closer to the original, the former isolates a band that is unique within the overall sound, namely (4,0). This narrow subband isolates a drone sound of a nearby airplane. In the wider bandwidth subband (1,0) associated with *BirdNMode2*, the drone is lumped in with part of the bird chirp spectrum. Fig.*5-12* shows the time-frequency plot of *Bird* prior to any basis selection. The low frequency energy band isolated in (4,0) is quite prominent.

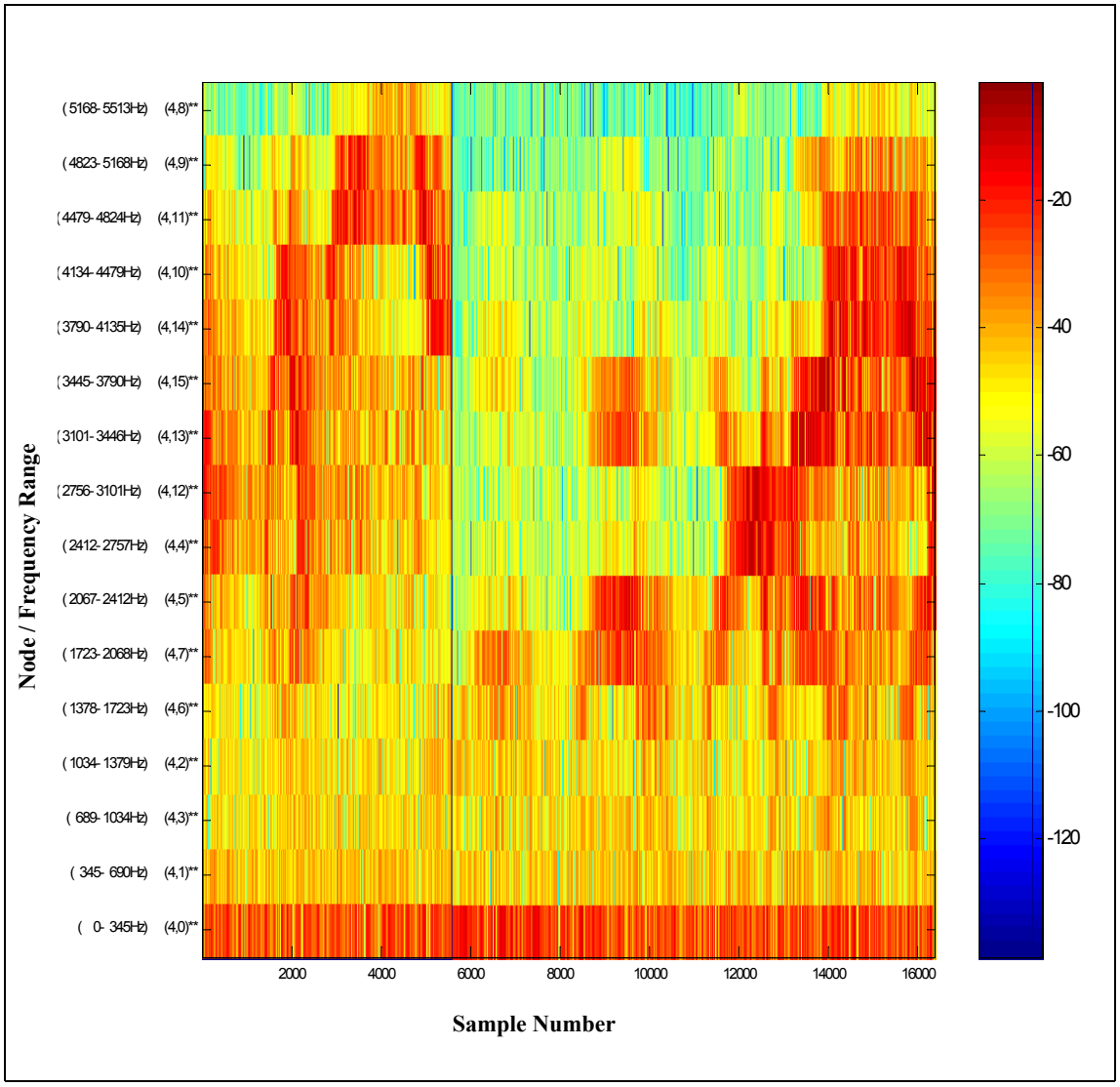

Figure 5-12: WPT Energy Plot (dB) of 'Bird' Sound* to *Depth-4
(node number and frequency range on vertical axis)

Table 5.11 Summary of Node Mode Test Cases

Sound Name	Original Sound / Transform / Original Basis	Basis Selection Parameters / New Basis	Modifications
Stairs1NMode1	*Stairs1 / WPT1 / Complete Tree Basis*	*-Mode = Node -BW Budget = 43.75% (95% Energy) -Node Budget = 4 -Weighting / (2,0), (3,3), (4,5), (4,15) (Actual BW = 50%)*	*- None*
Stairs1NMode2			*- Loop Mode is used and modifications are imparted in sequence: -Cross Synthesis: (2,0)/bell64/Loop 5, (3,3)/ bell64/Loop 13, (4,5)/flyl128/Loop 21, (4,15)/ diesl1024/Loop 29 - EQ: (4,15)/10x/Loop 37 -Modulation: (4,15)/DSB with sine wave at 0.005 Cycle/Sample/Loop 45 - EQ: (2,0)/disabled/Loop 53,(4,5)/disabled/ Loop 57, (3,3)/disabled/Loop 57, (2,0)(4,5)(3,3) /enabled/Loop 61 -Cross Synthesis: (2,0)/diesl1024/Loop 65 - EQ: (all)/1x/Loop 73 -Cross Synthesis: (all)/disable/Loop 77*
Stairs1CT1 (CT = Complete Tree)		*None / WPT1*	*- Loop Mode is used and modifications are imparted in sequence: -Cross Synthesis: (4,0)/bell64/Loop 5, (4,3)/ bell64/Loop 9, (4,6)/flyl128/Loop 13, (4,5)/ flyl128/Loop 17,(4,4)/flyl128/Loop 21, (4,2)/ flyl128/Loop 25.*
Stairs1EBBA1		*(4,0),(4,1),(4,2),(4,3),(4,4),(4, 5),(3,3),(3,4),(4,10),(4,11),(2, 3)*	*- None*
DoorsNMode1	*Doors / WPT1 / Complete Tree Basis*	*-Mode = Node -BW Budget = 18.75% (95% Energy) -Node Budget = 3 /* (2,0), (4,6), (4,12) / *(Actual BW = 37.5%)*	*-None*
DoorsNMode2			*-Cross Synthesis: (2,0)/diesel1024*
FlyNMode1	*Fly / WPT6 / Complete Tree Basis*	*-Mode = Node -BW Budget = 93.75% -Node Budget = 4 / (3,0), (4,3), (5,5), (5,13) (Actual BW = 25%)*	*-None*
FlyNMode2			*-Cross Synthesis: (3,0)/bell64, (4,3)/diesl1024. -EQ: (4,3) = 4.5x*

Table 5.11 Summary of Node Mode Test Cases

BirdNMode1	*Bird / WPT1 / Complete Tree Basis*	*-Mode = Node* *-BW Budget = 6.25%* *-Node Budget = 4 /* (4,0), (4,10), (4,13), (3,7) / *(Actual BW = 31.25%)*	*-None*
BirdNMode2		*-Mode = Node* *-BW Budget = 93.75%* *-Node Budget = 4 /* (1,0), (4,9), *(3,5), (2,3) /* *(Actual BW = 93.75%)*	
BirdNMode3		*-Mode = Node* *-BW Budget = 93.75%* *-Node Budget = 3 /* (4,0), (4,10), (2,3) *(Actual BW = 37.5%)*	

5.5.2.3 Bandwidth Mode

Some of the remaining motives listed in *Section 4.4.2.1* for performing the kind of lossy basis selection the *BDBBA* offers will be best satisfied with *Bandwidth Mode*. If a basis that best fits the 'most important features' of the sound signal is desired for example, then *Bandwidth Mode* makes more sense, since the amount of bandwidth allocated will not be modified due to Node Budget constraints. Having such a best fit may allow us to more effectively add modifications to the sound, since time-frequency resolution will be allocated more appropriately than the uniform resolution produced by the initial WPT. We can also often learn something of the nature of the signal from the basis that is selected. If a signal has a narrow band of energy and the basis reflects this in its time-frequency allocations, then we learn more about the signal. For example, in the previous discussion of Node Mode, recall that while *BirdNMode1* isolated a distinct feature in the sound, *BirdNMode2* did not. This difference was due to a different starting bandwidth budget. Then comparing *BirdNMode2* to *BirdNMode3*, where the starting bandwidth budget is the same but the node budget is different, again the resulting basis is quite different. Use of *Bandwidth Mode* avoids the compromises associated with also specifying a Node Budget.

Alternatively, we may simply want a basis that retains some fraction of the subbands for data reduction and/or simplicity reasons, regardless of how the energy in the signal is actually distributed. *Bandwidth Mode* lets us specify exactly how much is to be kept. We may in fact want to use a low bandwidth budget to intentionally change the sound through this basis selection process.

To gain some sense of the range of basis trees that occur and how they change from one budget to the next, we provide an example using the *Stairs1* signal, sweeping the budget from the minimum to maximum values. The total subbands retained are plotted versus bandwidth budget in Fig.*5-13*, and more details, including the particular subbands retained and the corresponding reference sound names are provided in *Table 5.12*.

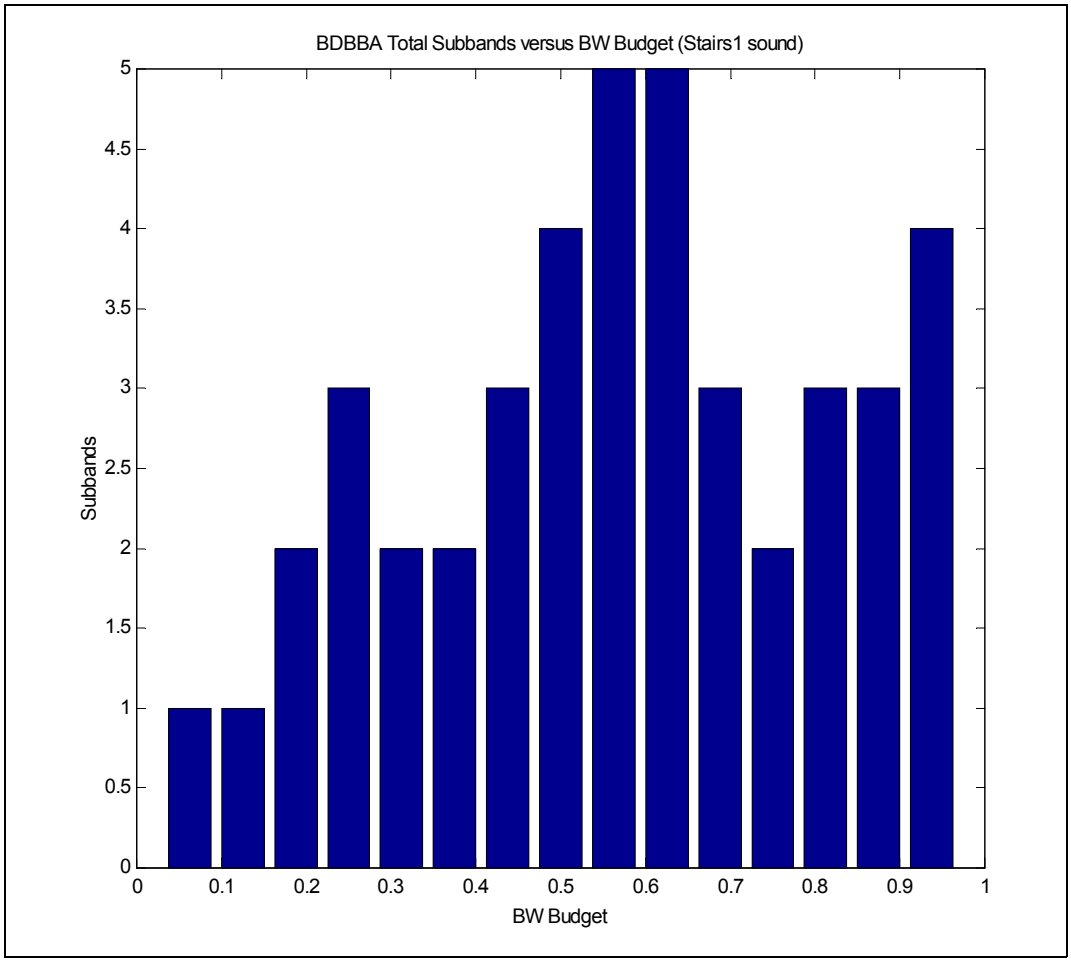

Figure 5-13: Total Subbands versus BW Budget for BDBBA of Stairs1

Table 5.12 Summary of Possible Bases for *Stairs1* over All *BW Budgets*[1]

Sound Name	BW Budget	Subbands	
		Retained	Total
Stairs1BMode1	*6.25%*	(4,2)	1
Stairs1BMode2	*12.5%*	(3,1)	1
Stairs1BMode3	*18.75%*	(4,1), (3,1)	2
Stairs1BMode4	*25%*	(4,1), (3,1), (4,6)	3
Stairs1BMode5	*31.25%*	(2,0), (4,6)	2
Stairs1BMode6	*37.5%*	(2,0), (3,3)	2
Stairs1BMode7	*43.75% (95% Energy case)*	(2,0), (4,5), (3,3)	3
Stairs1BMode8	*50%*	(2,0), (4,5), (3,3), (4,13)	4
Stairs1BMode9	*56.25%*	(2,0), (4,5), (3,3), (4,13), (4,15)	5
Stairs1BMode10	*62.5%*	(2,0), (4,5), (3,3), (3,6), (4,15)	5
Stairs1BMode11	*68.75%*	(1,0), (3,6), (4,15)	3
Stairs1BMode12	*75%*	(1,0), (2,3)	2
Stairs1BMode13	*81.25%*	(1,0), (4,10), (2,3)	3
Stairs1BMode14	*87.5%*	(1,0), (3,5), (2,3)	3
Stairs1BMode15	*93.75%*	(1,0), (4,9), (3,5), (2,3)	4

1.Transform WPT1 used in all cases.

To relate the basis subbands listed in *Table 5.12* to the filterbank tree as a whole, refer to the *depth-4* complete tree basis shown in Fig.*5-14*. Use of this complete tree allows one to visualize the basis tree for a given case. For example, *Stairs1BMode7* is comprised of terminal nodes (2,0), (4,5), and (3,3). The basis tree branches to the extent required to terminate in these nodes, and no further. All other terminal nodes required to produce these branches have their coefficients zeroed. The tree diagram for this case is shown in Fig.*5-15*. As always, valid or kept terminal nodes are indicated in a larger font. To see the trees for all the bandwidth budget cases, refer to *Appendix C*. *Sequence of Trees Example*.

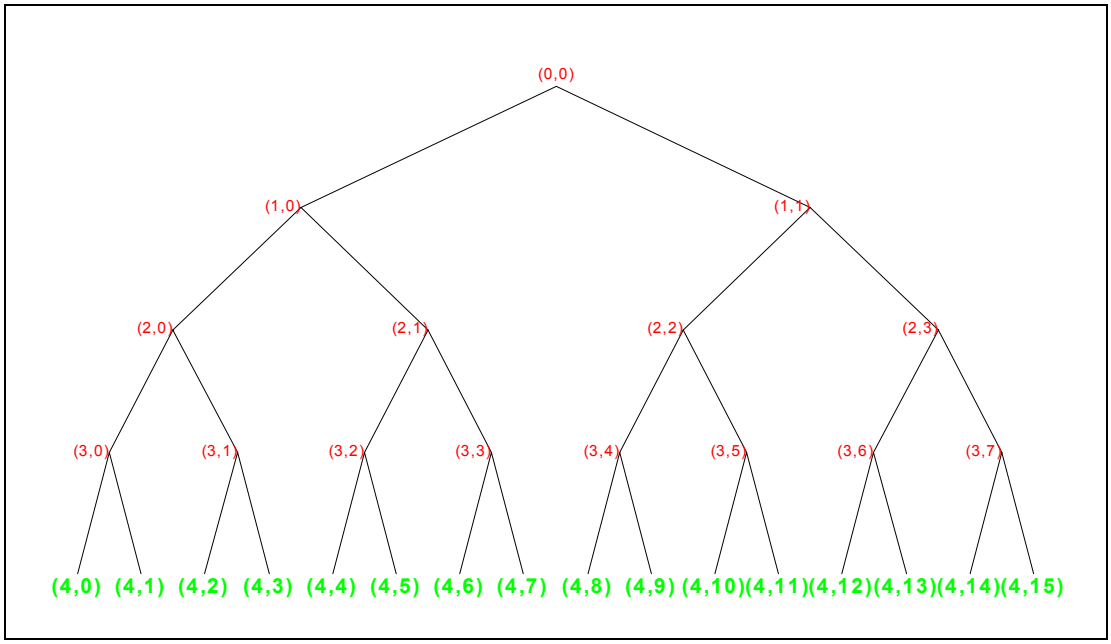

Figure 5-14: WPT Complete Tree Basis to Depth-4

.

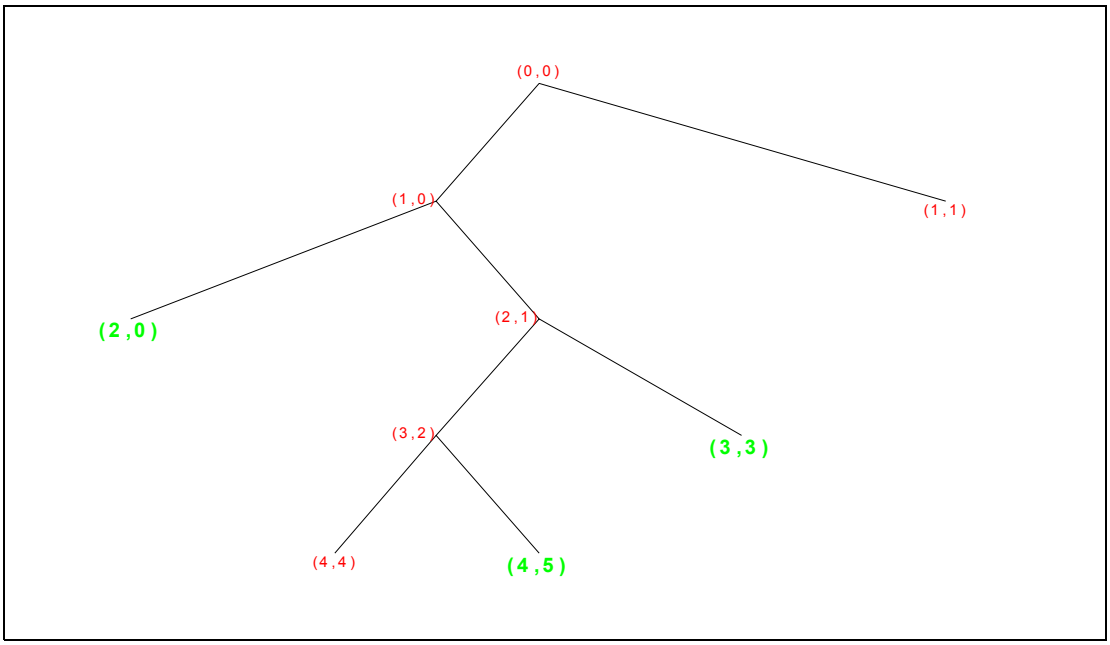

Figure 5-15: Stairs1 BDBBA Basis for B_{budget}= 43.75% (Stairs1BMode7)

Notice in this example how the spectrum occupied by the bases changes as the *Bandwidth Budget* is swept through its range from smallest to largest values. For each incremental increase in the *Bandwidth Budget,* the corresponding basis gains the next higher frequency portion of the spectrum. This relationship exists because the energy in the *Stairs1* signal is maximum at low frequencies, and decays gradually with increasing frequency. To see this relationship one must consider the frequency ordering of the subbands, as identified in *Table 3.1*. Note also that this would not be the case if psychoacoustic weighting of distortion were used. For example, in *Stairs1NMode1* as given in *Table 5.11* the basis is different from that of *Stairs1BMode8*, even though both ultimately use a bandwidth budget of 50%. The former does not occupy contiguous frequencies starting from the low end because the weighting operation puts a greater value on certain higher frequencies.

DraggingBMode1 captures the original sound *Dragging* quite well. The *95% Energy Bandwidth* in this case requires retaining 75% of the signal, and seven leaves result. But the result is quite faithful to the original, and each of the leaves is distinguishable in the overall sound, as demonstrated in the test case *DraggingBMode2*. We can also compare the spectrogram of *Dragging*, as shown in Fig.*5-16*, with the frequency-ordered nodes and corresponding frequencies associated with the basis *DraggingBMode1*:

- (4,1): 0.25 - 0.50 kHz
- (4,3): 0.50 - 0.75 kHz
- (4,7): 1.25 - 1.50 kHz
- (3,2): 1.50 - 2.00 kHz
- (2,3): 2.00 - 3.00 kHz
- (3,5): 3.00 - 3.50 kHz
- (4,9): 3.50 - 3.75 kHz

One can see that the retained nodes correspond to the highest energy parts of the signal. For example, node (4,1) corresponds well with the prominent lowest frequency band of energy in the spectrum. Also, since there is relatively little energy between 0.75 and 1.25 kHz, no nodes in the basis exist to cover that energy.

The correspondence between the BDBBA subbands and the signal spectrum should always exist, since the BDBBA finds the lowest distortion basis for the budget specified, and finding the lowest distortion subbands is equivalent to selecting the highest energy subbands[†]. Given this correspondence, we will also

[†] The only exception is when we use psychoacoustic weighting for the distortion metric (as discussed earlier). In that case we will not see as strong a correspondence, however, we still expect the BDBBA to discern between the relatively more important and less important subbands.

often find that, at least over much of the signal spectrum, the time-frequency resolution is appropriate for the signal energy distribution. Broad bands of energy will tend to have fewer wider nodes associated with them, since splitting them will not reduce their cost, and narrow bands of energy will correspond to narrower bandwidth nodes. While this mapping will be far from perfect, it will tend to make subsequent sound modification attempts more effective.

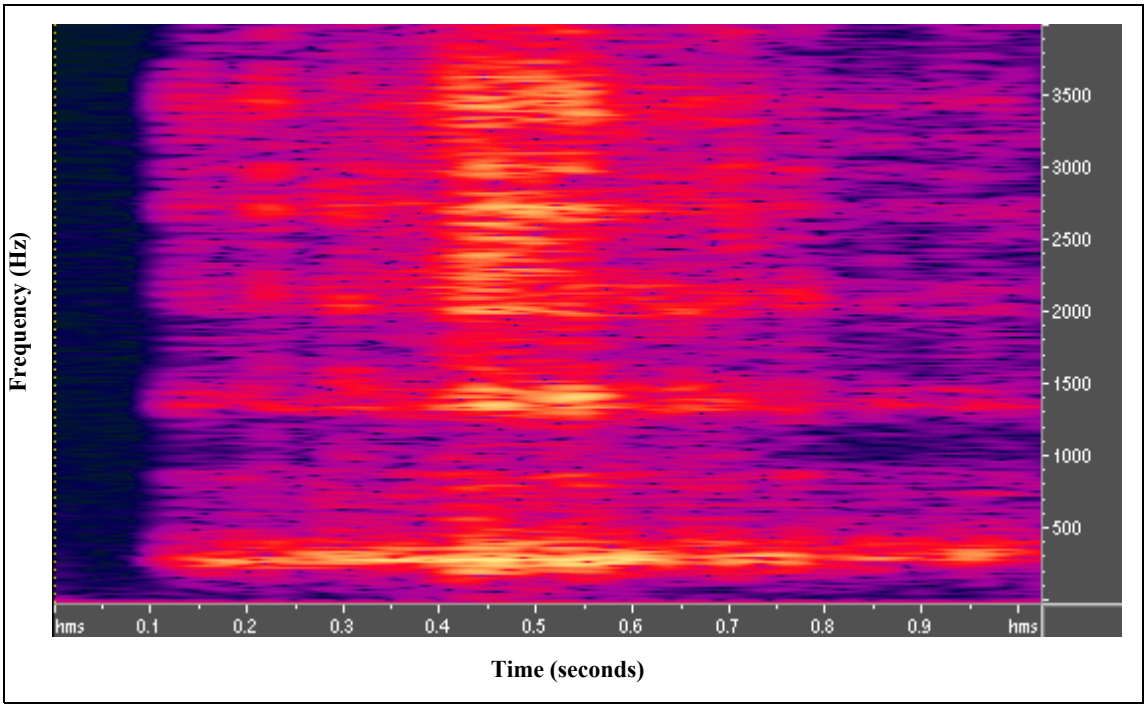

Figure 5-16: **Spectrogram of *Dragging***
(512 point DFT)

LakeBMode1 uses less nodes (two) then the previous case, and approximates the original fairly well with 31.25% bandwidth (*95% Energy Bandwidth* is 18.75%). The main difference compared to the original is loss of the high frequency wind sounds.

The Bandwidth Mode test cases are all summarized in *Table 5.13*.

Table 5.13 Summary of Bandwidth Mode Test Cases

Sound Name	Original Sound / Transform / Original Basis	Basis Selection Parameters / New Basis	Modifications
NoiseChirpBMode1	*NoiseChirp / WPT1 / Complete Tree Basis*	*-Mode = BW -BW Budget = 93.75% (95% Energy) / (2,0), (4,4), (3,3), (1,1)*	*- None*
FoghornBMode1	*Foghorn / WPT1 / Complete Tree Basis*	*-Mode = BW -BW Budget = 18.75% (95% energy) / (3,0), (4,3)*	*- None*
DraggingBMode1	*Dragging / WPT1 / Complete Tree Basis*	*-Mode = BW -BW Budget = 75% (95% energy) / (4,1), (4,3), (4,7), (3,2), (2,3), (3,5), (4,9)*	*- None*
DraggingBMode2			*-EQ: Successive enabling of subbands, in no particular order (volume normalized)*
LakeBMode1	*Lake / WPT1 / Complete Tree Basis*	*-Mode = BW -BW Budget = 31.25% / (2,0), 4,7)*	*- None*

5.5.2.4 Quantizer Alternatives

In *Section 4.4.2.3* we described the possibility of using quantizer choices different from the *all-or-nothing* case used in most of our work.

The option to use other quantizer choices is supported in the GUI. With reference to Figure 5-3: *GUI Basis Selection Window*, one can see in the basis selection portion of the window fields to set *R high*, *R low*, and *Q Range*. *R high* and *R low* set the number of bits for the high and low rate quantizers respectively. *R high* is set to *Flt*. *Pt*. and *R low* to 0 for the *all-or-nothing* quantization case. *Q Range* selects whether the quantization levels are determined by the global peak over all coefficients, or the peak for each subband.

In our work, the main purpose for implementing the BDBBA such that quantizer choices other than the *all-or-nothing* case could be used was to verify that the general framework of the BDBBA is useful[†]. No extensive testing has been performed with quantizer variations at this time, however, we did look at a few scenarios as summarized in *Table 5.14*.

[†] The BDBBA could have been implemented in a simpler fashion without this general quantization concept.

It seems that by not using *all-or-nothing* quantizer values, the BDBBA becomes a more subtle instrument that can better reveal the main features of a signal. Rather than being forced into either retaining or discarding a subband, the algorithm can keep just some of it. For example, given a subband with low average energy but a few peaks, the peaks can be approximated at little cost with the an appropriate low rate quantizer. Fig.*5-17* shows the result of the BDBBA being applied to the *Stairs2* signal, with high and low rate quantizers of 12 and 4 bits respectively (see the entry for *Stairs2QA1* of *Table 5.14* for more details on test conditions). Like many environmental sounds, the *Stairs2* sound is highly percussive with sharp attacks. The quantization helps highlight these attacks, and as such, a key feature of using such quantization is that it can help model sounds by revealing their most prominent characteristics. We might, for example, want to create a WPT based model where sounds are modeled as simplified coefficient streams applied to a set of variable bandwidth subbands and corresponding resynthesis filters. This quantization function as part of the basis selection process would be a way to help develop such models. Many different combinations of quantizers could be used in that case, such as 0 bits for the low rate, and also a relatively low rate for the high rate quantizer. Different budgets could be experimented with until a basis/model was found that provided the right balance between model simplicity and sound accuracy.

Unlike the *all-or-nothing* case, nodes split even if both child nodes are quantized with the same quantizer. This tends to result in more terminal nodes for the resulting basis. Of course a split will occur for the reason it does in the *all-or-nothing* case as well, namely when the sibling node contents differ from one another and each is best quantized with a different quantizer. If the user desires few subbands, the low rate subbands can always be ignored through clearly identification of the high rate nodes in the user interface.

Regarding the *Q Range* variable, we found that the *Global* choice quantizes the low rate more coarsely. Much detail is lost in this case. If we want the sound that some quantization brings, then the *Subband* choice will generally be better. The basis also tends to have more subbands than when the *Global* option is selected. Since in the *Global* case many sibling nodes will have all zeros, it is just as cheap cost-wise to merge them, and consequently the basis will have relatively less terminal nodes than the *Subband* based quantization choice. Compare *FlyQA1* to *FlyQA2*, for example. The latter uses the *Global* option, has far less terminal nodes, and is not as bright sounding since less of the signal is preserved.

Finally, as with the *all-or-nothing* case, aliasing can result due to defeating some portion of the alias cancellation mechanism by altering coefficient values. Again, a longer filter with better isolation will help mitigate this potential problem.

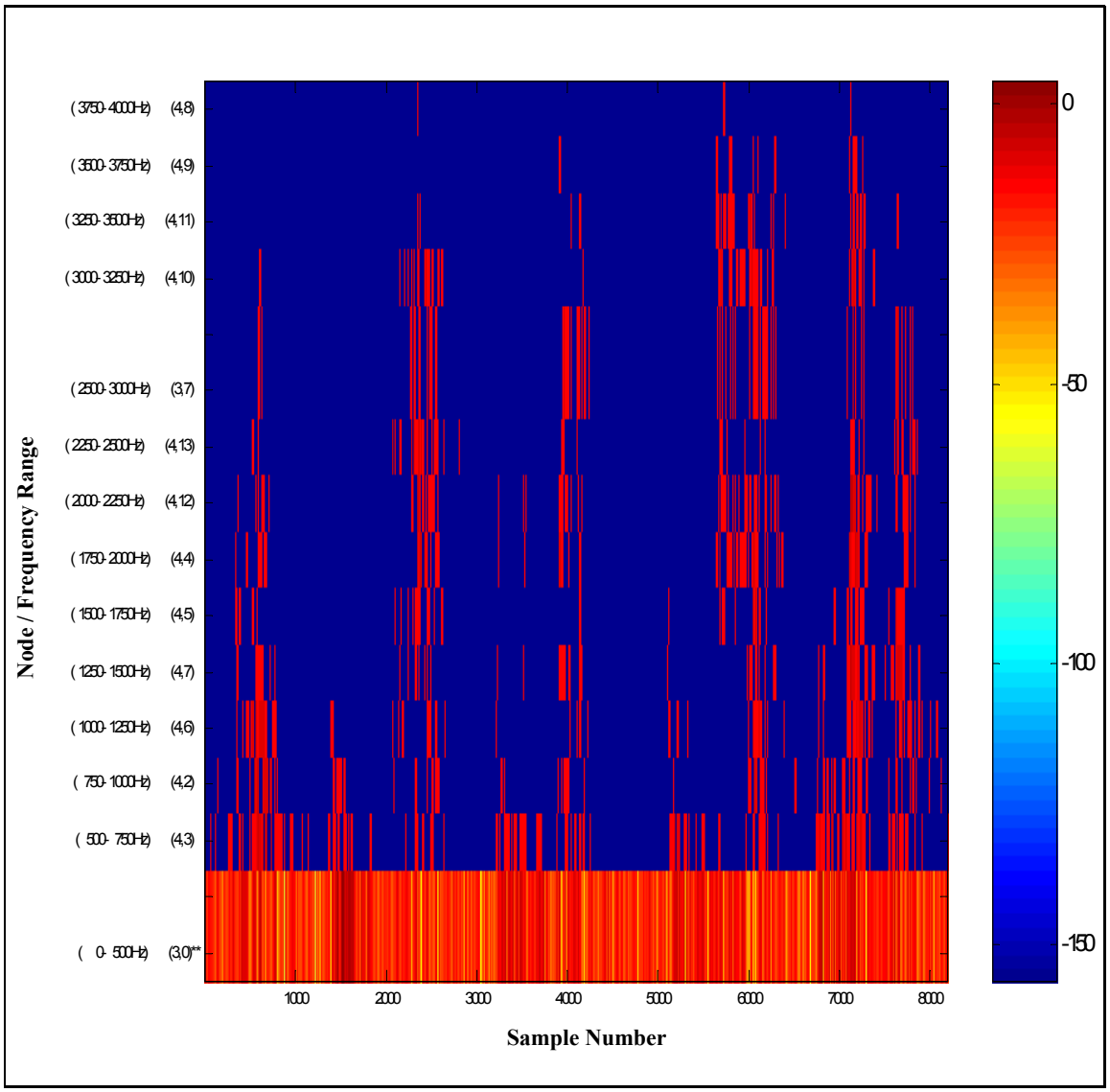

Figure 5-17: WPT Energy Plot (dB) of Stairs2QA1

Table 5.14 Summary of Quantizer Alternatives (QA) Test Cases

Sound Name	Original Sound / Original Basis	Basis Selection Parameters / New Basis	Modifications
Stairs2QA1	*Stairs2 / WPT1 / Complete Tree Basis*	*-Mode = BW* *-Quantization: R high = 12, R low = 4. Q Range = Global.* *-Quality Budget = 41.67% /* *R high Nodes: (3,0)* *R low Nodes: (4,2) - (4,13), (3,7)*	*- None*
Stairs2QA2		*-Mode = BW* *-Quantization: R high = 12, R low = 4. Q Range = Global.* *-Quality Budget = 41.67% /* *R high Nodes: (3,0)* *R low Nodes: (4,2) - (4,13), (3,7)*	*- Only Nodes (4,2), (4,6), (4,12), (4,13) enabled on resynthesis*
FlyQA1	*Fly / WPT1 / Complete Tree Basis*	*-Mode = BW* *-Quantization: R high = 11, R low = 3. Q Range = Subband.* *-Quality Budget = 54.54% /* *R high Nodes: (4,0) - (4,3), (3,3)* *R low Nodes: (4,4), (4,5), (4,8), (4,9),(3,5), (4,12) - (4,15)*	*- None*
FlyQA2		*-Mode = BW* *-Quantization: R high = 11, R low = 3. Q Range = Global* *-Quality Budget = 54.54% /* *R high Nodes: (4,0), (4,1), (3,1), (3,3)* *R low Nodes: (3,2), (1,1)*	*- None*
KnockQA1	*Knock / WPT1 / Complete Tree Basis*	*-Mode = BW* *-Quantization: R high = 16, R low = 3. Q Range = Subband* *-Quality Budget = 28.91%* *- Weighting /* *R high Nodes: (4,1), (4,6)* *R low Nodes: (4,0), (3,1), (4,4), (4,5), (4,7) - (4,11), (3,6), (3,7)*	*- None*

5.5.3 EBBA and TBBA

We conclude our discussion of basis selection test results with a brief look at the entropy-based EBBA and TBBA algorithms.

While the EBBA does have the virtue of adding no distortion to the signal, we did not find it very useful. In particular, the fact that it often only reduces the number of leaves in the basis tree by one or two means

that it is not useful for the purpose of terminal node reduction. There were cases where it did a reasonable job of reducing leaves, but these were the exception. Also, there is no way to steer the algorithm, so the user must simply accept what it produces.

Table 5.15 shows the bases that result from applying the EBBA to the complete tree of various *depth-4* transformed (16 leaves originally) signals. As can be seen, most reductions in the number of leaves are minimal. Compare, for example, test case *Stairs1EBBA* with *Stairs1BMode7* of *Table 5.12*, which was obtained using the BDBBA with *95% Energy Bandwidth*. In the latter case, the nodes retained were (2,0), (3,3), and (4,5), which is eight fewer than the number of nodes associated with *Stairs1EBBA*. Furthermore, it is not the case that the *Stairs1* signal has so much fine detail in the spectrum that would justify the many bands (of width 250 Hz) the EBBA produces. In this and the other cases, the minimum entropy basis produced by the EBBA does not seem to correspond to anything useful for our purposes.

Table 5.15 Summary of EBBA Test Cases[1]

Test Name	New Basis	Total Leaves
Stairs1EBBA	*(4,0), (4,1), (4,2), (4,3), (4,4), (4,5), (3,3), (3,4), (4,10), (4,11), (2,3)*	*11*
DraggingEBBA	*(4,0), (4,1), (4,2), (4,3), (4,4), (4,5), (4,6), (4,7), (2,2), (4,12), (4,13), (3,7)*	*12*
BirdEBBA	*(4,0), (4,1), (3,1), (4,4), (4,5), (4,6), (4,7), (4,8), (4,9), (4,10), (4,11), (4,12), (4,13), (4,14), (4,15)*	*15*
FlyEBBA	*(4,0), (4,1), (4,2), (4,3), (4,4), (4,5), (3,3), (1,1)*	*8*
TrainEBBA	*(4,0), (4,1), (4,2), (4,3), (4,4), (4,5), (4,6), (4,7),(3,4), (4,10), (4,11), (4,12), (4,13), (4,14), (4,15)*	*15*
GamelanEBBA	*(4,0), (4,1), (3,1), (4,4), (4,5), (3,3), (4,8),(4,9), (4,10), (4,11), (3,6), (4,14), (4,15)*	*13*
VoiceEBBA	*(4,0), (4,1), (4,2), (4,3) (4,4), (4,5), (4,6), (4,7), (4,8),(4,9), (4,10), (4,11), (3,6), (4,14), (4,15)*	*15*

1.All test cases use the transform WPT1. Also, we do not save the sound associated with these bases since they are identical to the original due to the lossless nature of the basis selection approach.

The TBBA offers some improvement over the EBBA in terms of both user control and node reduction. The user specifies a threshold value to guide the algorithm as described earlier in *Section 4.4.3.1*. By selecting different values for this threshold, bases of various shapes and number of terminal nodes are produced. As the threshold is swept from small to large values, in the cases we tried, the basis ranges from few nodes out to a peak number of nodes, and then back to a small number of nodes. One might decide to use thresholds in the lower range, so that only small coefficient values are discarded in the basis selection

process. A threshold of 0.0001 was used for both *Stairs1TBBA2* and *DraggingTBBA2*, and both have about 50% less nodes than their EBBA counterparts. In comparing the frequency divisions associated with these TBBA bases and the spectra of the signals, one can see at least some correspondence. The spectrum of *Dragging,* for example, was shown in Fig.*5-16*, and the frequency-ordered nodes and corresponding frequencies associated with *DraggingTBBA2* are:

- (4,0): 0 - 0.25 kHz
- (4,1): 0.25 - 0.50 kHz
- (3,1): 0.50 - 1.00 kHz
- (4,6): 1.00 - 1.25 kHz
- (4,7): 1.25 - 1.50 kHz
- (3,2): 1.50 - 2.00 kHz
- (2,3): 2.00 - 3.00 kHz
- (2,2): 3.00 - 4.00 kHz

A reasonable correspondence between the two appears to exist. To determine if the algorithm would be useful over a more general collection of sounds, however, more extensive testing of the TBBA would be required

Table 5.16 Summary of TBBA Test Cases[1]

Test Name	Threshold	New Basis	Total Leaves
Stairs1TBBA1	0.00001	*(0,0)*	*1*
Stairs1TBBA2	0.00010	*(4,0), (4,1), (3,1), (2,1), (3,4), (3,5), (2,3)*	*7*
Stairs1TBBA3	0.00050	*(4,0) - (4,5), (3,3), (4,8) - (4,11), (2,3)*	*12*
Stairs1TBBA4	0.00100	*(4,0) - (4,11), (2,3)*	*13*
Stairs1TBBA5	0.00500	*(4,0) - (4,5), (3,3), (4,8) - (4,13), (3,7)*	*14*
Stairs1TBBA6	0.01000	*(4,0), (4,1), (3,1), (4,4), (4,5), (3,3), (3,4), (4,10) - (4,13), (3,7)*	*12*
Stairs1TBBA7	0.05000	*(3,0), (3,1), (4,4), (4,5), (3,3), (4,8) - (4,11), (2,3)*	*10*
Stairs1TBBA8	0.10000	*(2,0), (4,4), (4,5), (3,3), (1,1)*	*5*
Stairs1TBBA9	0.250000	*(3,0), (4,2), (4,3), (3,2), (4,6), (4,7), (1,1)*	*7*
Stairs1TBBA10	0.500000	*(0,0)*	*1*
DraggingTBBA1	0.00001	*(2,0), (2,1), (1,1)*	*3*
DraggingTBBA2	0.00010	*(4,0), (4,1), (3,1), (3,2), (4,6), (4,7), (2,2),(2,3)*	*8*
DraggingTBBA3	0.00050	*(4,0), (4,1), (3,1), (4,4), (4,5), (3,3),(1,1)*	*7*
DraggingTBBA4	0.00100	*(4,0), (4,1), (3,1), (3,2), (4,6) - (4,11), (3,6), (4,14), (4,15)*	*13*
DraggingTBBA5	0.00500	*(4,0) - (4,9), (3,5) - (3,7)*	*13*
DraggingTBBA6	0.01000	*(3,0), (4,2) - (4,9), (3,5), (4,12), (4,13), (3,7)*	*13*
DraggingTBBA7	0.05000	*(4,0) - (4,15)*	*16*
DraggingTBBA8	0.10000	*(4,0) - (4,3), (3,2), (4,6) - (4,11), (3,6), (4,14), (4,15)*	*14*
DraggingTBBA9	0.250000	*(4,0), (4,1), (3,1), (4,4) - (4,7), (2,2), (3,6), (3,7)*	*10*
DraggingTBBA10	0.500000	*(1,0), (3,4), (3,5), (3,5), (4,12) - (4,15)*	*7*

1.All test cases use the transform WPT1. Also, we do not save the sound associated with these bases since they are identical to the original due to the lossless nature of the basis selection approach.

5.6 Summary

In this chapter we have described our implementation and testing of the ideas developed in *Chapter 4* for WPT-based sound signal processing. We first described the *graphical user interface* (GUI) that we developed for testing purposes. This GUI has enabled us to experiment with various sound modification and basis selection schemes, as well as with the WPT itself. The GUI has also served as a vehicle with which to prototype how a user might practically interact with the WPT and related algorithms. In our description of

the GUI we included examples of several of the main windows, including the ones used for signal analysis, synthesis, and basis selection.

We next described the default parameters chosen for the transforms used in our testing. These choices concerned filter type and length, transform depth, and the method of boundary extension. A listing of the sound clips used for our testing was then provided. This listing of over 20 sounds included details on the sample rate and length of each clip, as well as a brief description of it.

The detailed results of our testing were described in the last two major sections of this chapter. First we described our results regarding sound modifications, and then our results on basis selection. For all cases, we provided audio examples cross referenced to the test cases performed. Regarding the sound modifications tested, we found that many produced aesthetically interesting results. In this category we also tested the granular synthesis capability, again, with interesting results. Our tests also clarified on a practical level the relationship between coefficient alteration or removal and the aliasing and imaging that results. In some of these cases, a longer filter may have been better if these effects were to be minimized,.

Regarding our basis selection testing, we started with the PTOB. We showed a typical plot of transform coefficient energies for the time-frequency partitioning provided by the PTOB. Subjective testing of the basis indicated that it is useful relative to our hearing capabilities.

We then went on to describe our testing with the BDBBA, focusing primarily on the case of *all-or-nothing* quantization. We first discussed usage issues regarding the algorithm, such as selection of *Node Mode* versus *Bandwidth Mode*, and how to specify the budgets. We then described details of *Node Mode* operation, followed by *Bandwidth Mode*. We found the *Node Mode* of operation particularly useful, but also found that *Bandwidth Mode* could be helpful in repartitioning the time-frequency resolution of the transform, and possibly providing an abstraction of the signal that would allow the user to focus on its most important aspects. Finally, for the BDBBA, we described the rudimentary testing we performed with other quantizer settings. Use of other quantizer settings creates new possibilities, including the provision of a less binary abstraction of the sound. Low energy subbands are not simply deleted as they are in the *all-or-nothing* case, but rather are quantized more coarsely. This often results in an abstraction or simplification of the transform coefficients which allows one to more clearly see the main features of the signal. The explicit intentional addition of quantization noise can also produce interesting audible effects.

Finally, we provided some test results for the EBBA and TBBA entropy based schemes. We found the

EBBA to be of little value to us since it is neither steer-able by the user, nor is it effective at significantly reducing the basis terminal node count. The TBBA is at least steer-able to some extent, and we found in our limited testing that with appropriate specification of the threshold parameter, bases that map reasonably well with the spectral content of the signal are possible (though more testing would be required to verify this).

This completes the main description of our work. We provide our conclusions and recommendations for possible future work in the next chapter.

Chapter 6
Conclusions

We have demonstrated that the discrete wavelet packet transform provides a useful framework for the analysis, modification and resynthesis of sounds, especially environmental sounds. The WPT can also be used as the synthesis engine for granular synthesis methods. The approach is of reasonable complexity by virtue of using a fast multiresolution filter bank implementation. Compared to the WT, the WPT provides more flexibility in how a given sound can be partitioned in time-frequency. The WT is simply a special case of the WPT. Many other fixed bases can be extracted from the complete tree basis of the WPT, and best basis algorithms can be used to find bases that are optimal with respect to some objective measure.

To facilitate the testing of various transform configurations, modification approaches, and best basis selection methods, we have developed a MATLAB-based GUI. This GUI has also allowed us to prototype how a user might practically work with these tools in the process of sound modification and creation. The GUI and the approaches we investigated work particularly well on simple, short environmental sound clips, which are then transformed, modified, and resynthesized in interesting ways.

We have looked at the basics of the WPT and how it might be configured for our purposes of sound processing, settling on the use of a certain filter choice (*sym5*), transform depth (4), and boundary extension method (symmetric). This standardized transform configuration helped keep the number of variables between our various tests to a minimum, thereby enhancing our ability to make comparative observations. We do not claim that this parameter set is in any way globally optimal for all application possibilities.

We next considered a series of modifications that all take advantage of some aspect of the time-frequency partitioning of the sound produced by the transform. These modifications included time stretching and contraction, coefficient filtering, cross synthesis, subband modulations, and equalization. Additionally, transform coefficients were created afresh, to test the possibilities of the transform for granular synthesis purposes.

Finally, we looked in detail at several approaches of re-partitioning the time-frequency resolution. This included the development of a fixed basis we call the *pseudo third-octave basis* (PTOB), as well as looking at cost based approaches to best basis selection. The PTOB is useful for equalization purposes, emulating the frequency spacings associated with *third-octave* graphic equalizers. For best basis selection, we developed

the *Bi-rate Distortion Best Basis Algorithm* (BDBBA), which uses techniques from the field of data compression to find bases that have a limited number of subbands and/or that partition the time-frequency resolution in a way that lines up well with the spectral composition of the signal. The algorithm allows the user to pare down a basis with a large number of subbands to something manageable, while still providing a degree of frequency resolution where it is justified. While we focused on the *all-or-nothing* quantization case of the BDBBA, we also considered other quantization scenarios. We next looked at inherently lossless best basis approaches such as an entropy based approach (EBBA), and a variation thereof based on thresholding (TBBA). Finally, still in the area of best basis selection, we briefly considered potential model-based approaches, such as a predictor-based scheme.

We feel that our results point to a great deal of promise for the WPT in the field of computer music and sound signal processing. The transform itself provides the opportunity for trading off time and frequency resolution in a flexible way. For example, the ability to achieve fine time resolution without uniformly compromising frequency resolution is an important feature that distinguishes the WPT from fixed resolution transforms like the STFT. The basis selection approaches we considered helped us to successfully exploit this flexible time-frequency resolution capability. The BDBBA in particular is a useful tool in this area. Finally, most of the sound modifications that we experimented with in the WPT framework produced aesthetically interesting results, as did our *granular synthesis* experiments involving the creation of entirely new transform coefficient streams.

On the other hand, as with all methods, there are limitations. For highly tonal material, Fourier based methods are often preferable since their basis functions are (windowed) complex sinusoids[†]. Since we are accustomed to the basis functions of the Fourier Transform, a continuous or highly redundant windowed Fourier Transform such as the STFT is still often the most popular choice for the analysis and interpretation of a sound in the time-frequency domain. We are accustomed to seeing sounds in this representation. Conversely, we are less used to seeing sounds projected onto a discrete critically sampled wavelet packet basis, especially since there are many possible bases given the filter options and the flexible time-frequency resolution trade-offs available with the WPT. With time, however, we may well grow accustomed to viewing WPT based spectrograms, and thereby further benefit from the multiresolution the WPT offers.

[†] Although when the Fourier approach is discrete with a finite set of frequencies, matching subband frequencies to the tonal content of the sound is still required for truly meaningful transform coefficients.

The fact that the DWPT is critically sampled also means that aliasing and imaging effects will crop up. Since filter responses are never *brick-wall*, some energy from outside a given band will be aliased into the band. On the synthesis side, the images caused by upsampling are never completely removed by filtering. In both cases, the transform relies on cancellation of these products by addition of subbands during resynthesis. Depending on the modification we make, and the sound we are working with, these effects will be more or less noticeable. In some cases, they add an interesting dimension to the sound. In cases where they are problematic, longer filters can be used to improve the frequency response and isolation between bands, so that these products are minimized. Alternatively, the aliasing and imaging could be avoided altogether by using expansions that are not subsampled, such as *Laplacian pyramids* (Burt and Adelson 1983). The degree to which aliasing is problematic for a given application would dictate whether such an approach is justified, since a performance penalty would necessarily result.

6.1 Summary of Contributions

To our knowledge, this work represents the first systematic research done in the computer music field that applies the discrete wavelet packet transform to sound analysis, modification, and synthesis/resynthesis. The sound modifications designed and implemented are novel given their use with this transform in particular. Many of the modifications produced subjectively interesting results. Along with this modification capability, we have designed and implemented the ability to synthesize new sounds using a wavelet packet basis as sound grains in a granular synthesis approach, thereby tying together in one framework the ability to work with pre-existing sounds and to generate completely new sounds that are not based on pre-existing material. Combining these two capabilities in our sound creation efforts is powerful since separate environments for sound analysis-resynthesis and granular synthesis are not required. The user can work with and modify pre-existing sound material and generate completely new sounds afresh within the same environment.

Regarding basis selection, the fixed PTOB basis was designed and tested to approximate *third-octave* spacing using wavelet packets. More significantly, we have developed a best basis algorithm (the BDBBA) based on rate-distortion techniques that simultaneously finds a best basis while optionally eliminating subband coefficients whose energies contribute least to the sound signal's content. This algorithm is based on the work of Ramchandran and Vetterli (1993), but differs in several ways. For the *all-or nothing* quantization case, which most of our testing was focused on, we work with bandwidth instead of rate. Each node is either kept or eliminated. We do not carve out the tree for each Lagrange multiplier λ either, doing so only after convergence. We also only split a parent node if the sum of its child costs is *less than* its own, versus splitting

if the sum of the child costs is *less than or equal*. This decision rule is necessary because the equality condition occurs frequently, and represents no cost improvement over the parent. We have created an outer loop in the algorithm to find a basis with a given number of terminal nodes, for when this is the driving concern. Also, we have added to the algorithm a basic psychoacoustic model, which weights distortion costs according to its relative psychoacoustic importance within the spectrum.

Finally, we have created a GUI based tool to prototype not only the underlying algorithms, but also how a user might work with them.

6.2 Future Research

We now list several avenues of additional research that might be fruitful.

Regarding the GUI itself, there are many possible extensions and enhancements. From the perspective of creating a fast tool with real-time capabilities, a stripped down version of the GUI with only the necessary functionality, implemented efficiently in a compiled language such as *C* might be useful. A specific requirements list could be drawn up based on having a group of computer music users experiment with the current GUI to determine what changes would be beneficial. An architecture that allowed for the convenient patching in of stored transform coefficients from previously analyzed, re-partitioned (via basis selection) and modified sounds, along with the ability to analyze and modify new sounds *on the fly*, and to create new coefficients via granular synthesis would be useful. The ability to change the resynthesis filter over time and across subbands in a flexible way would allow for fast selective cross synthesis opportunities. Attachment of more convenient user interface controls, such as rotary knobs or sliders, would also be helpful, and further reinforce the benefit of basis subband reduction via best basis selection.

An additional feature that might be useful is a front end on the analysis process that resamples the input signal to a lower rate such that the maximum frequency content of the signal always coincides with the Nyquist frequency. In many cases, all of a signal's energy is well below the Nyquist frequency, and consequently the upper subbands of the WPT contain no energy. Basis selection in those case is less efficient, since subbands that are irrelevant to the sound are still being processed and displayed in some way.

Regarding other sound modifications to experiment with, there are many possibilities within each category. We list just a few. One possibility might be the phase shifting of some subbands, achieved by advancing or retarding in time the coefficients in a given subband. Such coefficients could be rotated as if the subband buffer were circular, or simply flow out one end with zeros flowing in at the other end. Many

strategies for phase shifting can be imagined, such as random shifts, linearly increasing shifts, or patterned shifts according to specified shapes or waveforms.

In the filtering of coefficients we tried, only a lowpass filter was experimented with. Many possibilities in terms of other filter types are available here. While we looked at time stretching and contraction, we did not experiment with *pitch shifting* via changing the wavelet filter to one associated with a different subband and frequency range. Such pitch shifting would be equivalent to moving coefficients from one subband to another.

Regarding basis selection, the framework we identified allows for extension in several directions. One is the consideration of other quantization strategies in the BDBBA case to find different ways to drive the basis selection process. Model based approaches might also be pursued. On the lossless side, other entropy-like cost functions might be investigated. Also, the possibility of an application framework that supports a more dynamic basis over time in a way that is still useful to the user might be looked at. This approach would allow for a better mapping of basis to sound, but would also add complexity for the user interface and basis selection process.

Finally, an investigation into non-subsampled transforms such as the *Laplacian pyramid* (Burt and Adelson 1983) would be useful in determining the trade-offs between the efficiencies enjoyed by the FWT-based approach, and the lack of aliasing and imaging issues associated with a non-subsampled approach.

Bibliography

Aggarwal, A., V. Cuperman, K. Rose, and A. Gersho. Perceptual Zerotrees for Scalable Wavelet Coding of Wideband Audio. (*WEB pre-print*: Available at http://scl.ece.ucsb.edu/pubs/pubs_E/e99_7.pdf).

Allen, J. B., and L.R. Rabiner (1977). A Unified Approach to Short-Time Fourier Analysis and Synthesis. *Proceedings of the IEEE,* 65. pp. 1558-1564.

Arfib, D. (1991). Analysis, Transformation, and Resynthesis of Musical Sounds with the Help of a Time-Frequency Representation. In G. De Poli, A. Piccialli, and C. Roads, editors, *Representations of Musical Signals*, chapter 3. The MIT Press, Cambridge, Massachusetts.

Boyer, F., and R. Kronland-Martinet. (1989). Granular Resynthesis and Transformation of Sounds Through Wavelet Transform Analysis. *Proceedings of the 1989 International Computer Music Conference*, San Francisco: CMA: 51-54.

Brandenburg, K. and G. Stoll. (1996). ISO-MPEG-1 Audio: A Generic Standard for Coding of High-Quality Digital Audio. *Collected Papers on Digital Audio Bit-Rate Reduction. Audio Engineering Society (AES)*.

Brandenburg, K. and H. Popp. (2000). An Introduction to MPEG Layer - 3. *EBU Technical Review.* (Available at http://www.mp3-tech.org/ and follow *Programmer's Corner* link, then *Technical Audio Papers* link.)

Burt, P.J., and E.H. Adelson. (1983, April). The Laplacian Pyramid as a Compact Image Code. *IEEE Transactions on Communications*. Vol. 31, No.4. pp. 532-540.

Clarke, M. (1996). Composing at the intersection of time and frequency. *Organised Sound*, 1(2): 107-117.

Coifman, R. and M. Wickerhauser. (1992, March). Entropy-Based Algorithms for Best Basis Selection. *IEEE Transactions on Information Theory*. Vol. 38, No.2.

Crochiere, R.E., S.A. Weber, and J. L. Flanagan. (1976, October). Digital Coding of Speech in Subbands. *Bell Systems Technical Journal*. Vol. 55, pp. 1069-1085.

Crosier, A., D. Esteban, and C. Galand. (1976, August). Perfect Channel Splitting by Use of Interpolation, Decimation, Tree Decomposition Techniques. *International Conference on Information Sciences/Systems*. Patras. pp. 443 - 446.

Daubechies, I. (1988). Orthonormal Bases of Compactly Supported Wavelets. *Comm. in Pure and Applied Mathematics*. Vol. 41, No. 7. pp. 909-996.

Daubechies, I. (1992). *Ten Lectures on Wavelets*. SIAM.

De Poli, G., A. Piccialli, and C. Roads, editors. (1991). *Representations of Musical Signals*. The MIT Press, Cambridge, Massachusetts.

De Poli, G. and A. Piccialli. (1991).Pitch-Synchronous Granular Synthesis. In G. De Poli, A. Piccialli, and C. Roads, editors, *Representations of Musical Signals*, chapter 6. The MIT Press, Cambridge, Massachusetts.

Dolson, M. (1986). The Phase Vocoder: A Tutorial. *Computer Music Journal*, 10(4): 14-27.

Esteban, D., and C. Galand. (1977, May). Application of Quadrature Mirror Filters to Split-Band Voice Coding Schemes. *International Conference on Acoustics, Speech and Signal Processing*. Hartford, Connecticut. pp. 191-195.

Evangelista, G. (1991). Wavelet Transforms that We Can Play. In G. De Poli, A. Piccialli, and C. Roads, editors, *Representations of Musical Signals*, chapter 4. The MIT Press, Cambridge, Massachusetts.

Evangelista, G. (1997). Wavelet representations of musical signals. In C. Roads, S.T. Pope, A. Piccialli, and G. De Poli, editors, *Musical Signal Processing*, chapter 4. Swets and Zeitlinger B.V., Lisse, the Netherlands.

Everett, H. (1963). Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. *Operation Res*. vol. 11. pp. 399-417.

Flanagan, J.L., and R. Golden. (1966). Phase Vocoder. *Bell System Technical Journal.*, 45. pp. 1493 - 1509.

Gabor, D. (1946). Theory of Communication. *Journal of the IEE*, Vol. 93. 429-457.

Gilbert, A.C., W. Willinger, and A. Feldman. (1999, April). Scaling Analysis of Conservative Cascades, with Applications to Network Traffic. *IEEE Transactions on Information Theory*. Vol. 45. No. 3.

Grossman, A., and J. Morlet. (1984, July). Decomposition of Hardy Functions into Square Integrable Wavelets of Constant Shape. *SIAM Journal of Mathematical Analysis*. Vol. 15, No.4. pp. 723 - 736.

Helmholtz, H. (1863). *On the Sensations of Tone as a Physiological Basis for the Theory of Music*. Reprinted 1954, A. Ellis, trans. New York: Dover.

Johnston, J.D. (1988). Transform coding of audio signals using perceptual noise criteria. *IEEE Journal on Selected Areas of Communications*. Vol. 6. pp. 314-323.

Joint Technical Committee (JTC) 1. (1993). *ISO/IEC 11172-3 Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s - Part 3: Audio*. ISO/IEC.

Joint Technical Committee (JTC) 1. (2000). *ISO/IEC 15444 JPEG 2000 International Standard.* ISO/IEC.

Kaegi, W., and S. Tempelaars. (1978). VOSIM - a new sound synthesis system. *Journal of the Audio Engineering Society*. 26(6). 418-426.

Kahrs, M., and K. Brandenburg. (1998). *Applications of Digital Signal Processing To Audio And Acoustics*. Kluwer Academic Publishers.

Knuth, D. (1997). *The Art of Computer Programming - Volume 1 - Fundamental Algorithms*. Third Edition. Addison Wesley.

Kovacevic, J. and I. Daubechies, eds. (1996, April). Special Issue on Wavelets. *Proceedings of the IEEE*, Vol. 84, No.4.

Kronland-Martinet, R. (1988, Winter). The Wavelet Transform for Analysis, Synthesis, and Processing of Speech and Music Sounds. *Computer Music Journal*, 12(4): 11-20.

Kronland-Martinet, R., and A. Grossman. (1991). Application of Time-Frequency and Time-Scale Methods (Wavelet Transforms) to the Analysis, Synthesis, and Transformation of Natural Sounds. In G. De Poli, A. Piccialli, and C. Roads, editors, *Representations of Musical Signals*, chapter 2. The MIT Press, Cambridge, Massachusetts.

Kronland-Martinet, R., Ph. Guillemain and S. Ystad. (1997). Modeling of natural sounds by time-frequency and wavelet representations. *Organised Sound*, 2(3): 179-191.

Kussmaul, C. (1991). Applications of the Wavelet Transform at the Level of Pitch Contour. *Proceedings of the 1991 International Computer Music Conference*, San Francisco: CMA: 483-486.

Loughlin, P. editor. (1996, September). Special Issue on Time-Frequency Analysis. *Proceedings of the IEEE*, Vol. 84, No.9.

Mallat, S. (1989, July). A Theory for Multiresolution Signal Decomposition: the Wavelet Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol. 11, No. 7, pp. 674-693.

Malvar, H. (1992). *Signal Processing with Lapped Transforms*. Artech House.

Marven, C. and G. Ewers. (1996). *A Simple Approach to Digital Signal Processing*. John Wiley and Sons, Inc.

Masri, P., A. Bateman and N. Canagarajah. (1997). A review of time-frequency representations, with application to sound/music analysis-resynthesis. *Organised Sound*, 2(3): 193-205.

Masri, P., A. Bateman and N. Canagarajah. (1997). The importance of the time-frequency representation for sound/music analysis-resynthesis. *Organised Sound*, 2(3): 207-214.

Meyer, Y. (1990). Ondelettes et Operateurs, Tome I. *Ondelettes*. Herrmann ed. Paris.

Misiti, M., Y. Misiti, G. Oppenheim, and J-M. Poggi (1997). *Wavelet Toolbox User's Guide.* The MathWorks, Inc.

Moorer, J. A., and J.M. Grey. (1977). Lexicon of analyzed tones. Part I: a violin tone. *Computer Music Journal*, 1(2): 39-45. Part II: clarinet and oboe tones. *Computer Music Journal*, 1(3): 12-29.

Oppenheim, A.V., R. W. Schafer and J.R. Buck. (1999, February). *Discrete-Time Signal Processing*. Prentice-Hall.

Pielemeier, W., G. Wakefield and M. Simoni. (1996) Time-Frequency Analysis of Musical Signals. *Proceedings of the IEEE*, 84(9):1216-1230.

Petitcolas, Fabien.Freeware MPEG1 Layer 1-2 Psychoacoustic Model Demo. University of Cambridge (Available at http://www.cl.cam.ac.uk/~fapp2/software/mpeg/).

Polotti, P. and G. Evangelista. (2001, Fall). Fractal Additive Synthesis via Harmonic-Band Wavelets. *Computer Music Journal*, 25(3): pp.22-37.

Princen, J. and A. Bradley. (1986, October). Analysis/Synthesis Filter Bank Design Based on Time Domain Aliasing Cancellation. *IEEE Transactions on Acoustics, Speech, and Signal Processing*.

Ramchandran, K. and M. Vetterli. (1993, April). Best Wavelet Packet Bases in a Rate-Distortion Sense. *IEEE Transactions on Image Processing*. Vol.2, No. 2. 160-174.

Rioul, O., and M. Vetterli. (1991). Wavelets and Signal Processing. *IEEE Signal Processing Magazine*. October 1991.14-38.

Risset, J-C. (1969). An introductory catalog of computer synthesized sounds (with sound examples). *Bell Laboratories Report*. Murray Hill: Bell Laboratories.

Risset, J-C. (1991). Timbre Analysis by Synthesis: Representations, Imitations, and Variants for Musical Composition. In G. De Poli, A. Piccialli, and C. Roads, editors. *Representations of Musical Signals,* chapter 1. The MIT Press, Cambridge, Massachusetts.

Roads, C. (1985). Automated granular synthesis of sound. *Computer Music Journal*, 2(2). Reprinted in C. Roads and J. Strawn, eds. *Foundations of Computer Music*. Cambridge, MA: MIT Press.

Roads, C. (1991). Asynchronous Granular Synthesis. In G. De Poli, A. Piccialli, and C. Roads, editors. *Representations of Musical Signals,* chapter 5. The MIT Press, Cambridge, Massachusetts.

Roads, C. (1996). *The Computer Music Tutorial*. The MIT Press, Cambridge, Massachusetts.

Roads, C. (2002). *Microsound*. The MIT Press, Cambridge, Massachusetts.

Rodet, X. (1980). Time-domain formant-wave-function synthesis. In *Spoken Language Generation and Understanding*, J.G. Simon editor. Dordrecht: D. Reidel. Reprinted in *Computer Music Journal*, 8(3) (1984). 9-14.

Roederer, J. (1975). *Introduction to the Physics and Psychophysics of Music*. *Springer*.

Rothweiler, J. (1983). Polyphase Quadrature Filters - A New Subband Coding Technique. *ICASSP 83, Boston. IEEE.*

Sayood, K. (2000). *Introduction to Data Compression. Second Edition*. Morgan Kaufman.

Serra, M-H. (1997). Introducing the phase vocoder. In C. Roads, S.T. Pope, A. Piccialli, and G. De Poli, editors, *Musical Signal Processing*, chapter 2. Swets and Zeitlinger B.V., Lisse, the Netherlands.

Serra, X. (1997). Musical sound modeling with sinusoids plus noise. In C. Roads, S.T. Pope, A. Piccialli, and G. De Poli, editors, *Musical Signal Processing*, chapter 3. Swets and Zeitlinger B.V., Lisse, the Netherlands.

Shannon, C.E. (1959). Coding Theorems for a Discrete Source with a Fidelity Criterion. *IRE International Convention Records*. vol. 7, part 4. pp. 142 - 163.

Shapiro, J. M. (1993). Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*. Vol. 41. pp. 3445-3462.

Shoham, Y., and A. Gersho. (1988, September). Efficient Bit Allocation for an Arbitrary Set of Quantizers. *IEEE Transactions on Acoustics, Speech, and Signal Processing.* Vol. 36, No. 9.

Sinha, D. and A. Tewfik (1993, December). Low Bit Rate Transparent Audio Compression Using Adapted Wavelets. *IEEE Transactions on Signal Processing*.

Srinivasan, P. and L. Jamieson. (1998, April). High Quality Audio Compression Using an Adaptive Wavelet Packet Decomposition and Psychoacoustic Modeling. *IEEE Transactions on Signal Processing*.

Strang, G., and T. Nguyen. (1997). *Wavelets and Filter Banks*. WellFesley-Cambridge Press.

Truax, B. (1987). Real-time Granulation of Sampled Sound with the DMX-1000. In J. Beauchamp, ed. *Proceedings of the 1987 International Computer Music Conference*, San Francisco. International Computer Music Association. pp. 138-145.

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

Truax, B. (1994, Summer). Discovering Inner Complexity: Time Shifting and Transposition with a Real-time Granulation Technique. *Computer Music Journal*, 18(2): pp.38-48.

Truax, B. (1999). *Handbook For Acoustic Ecology*. CD-ROM Edition. Version 1.1. Cambridge Street Publishing.

Vercoe, B., J. Fitch, R. Eckman, R. Whittle, and G. Maldonado. (1998). Csound Windows Help file version 4.00.950 for Csound version 3.48. April 1998. (Latest versions available at: http://www.cs.bath.ac.uk/pub/dream/).

Wickerhauser, M.V. (1989). Acoustic Signal Compression with Wave Packets. In *Wavelets: A Tutorial in Theory and Applications* (ed. C.K. Chui). Academic Press, New York. pp. 679-691.

Wickerhauser, M.V. (1991, November). *Lectures on Wavelet Packet Algorithms*. (Available at http://www.mathsoft.com/wavelets.html).

Wickerhauser, M.V. (1994). *Adapted Wavelet Analysis from Theory to Software*. A.K. Peters. Ltd.

Zwicker, E., and H. Fastl. (1990). *Psychoacoustics Facts and Models*. Springer-Verlag.

Appendix A
Further Complexity Considerations

Here we provide some more detailed analysis of certain complexity related concerns.

Derivation of the FWT Complexity Order

As stated earlier in Section *4.5 Complexity*, the WT computational complexity is *O(N)*. We provide a derivation of this in the following.

The beauty of the WT is that with each deeper lever of decomposition, there are less operations to be performed. Consider a similar example to the WPT case given in *Section 4.5*: a *depth-3* WT performed on an 8 sample input with the 2-point *Haar* filter. The first level of filtering is the same as the WPT case. The high and lowpass filter operations are initiated every second sample so that $N/2$ lowpass filter operations and $N/2$ highpass operations each are performed. This results in a total of *N* filter operations at the first level. Since the Haar filters involve a simple addition of adjacent samples in the lowpass case, and a simple subtraction in the highpass case, each filter operation involves 2 computations. Therefore we incur *2N* or 16 computations at the first level.

Since this is the *WT* and not the *WPT*, only lowpass filter outputs are further processed. The lowpass filter output from the previous level has *N/2* points. These are each split by high and lowpass filters again. For each of these there are *N/4* filter operations ((*N/2)/2*), giving a total of another *(N/4)*2 = N/2* operations. Again each of these involves 2 computations, for another $N = 8$ computations. At the next level, *N/4* operations are required, and so on to full depth. At each new level we have half the number of filter operations. We can write this as the following sequence:

$$Operations = (N + \frac{N}{2} + \frac{N}{4} + \ldots + 2) = N(1 + \frac{1}{2} + \frac{1}{4} + \ldots + \frac{2}{N})$$

but $N = 2^{depth}$, therefore,

$$Operations = N\left(1 + \frac{1}{2} + \frac{1}{4} + \ldots + \frac{1}{2^{depth-1}}\right)$$

and we can invoke the following relationship for geometrical progressions:

$$G_n = a + aq + aq^2 + \ldots + aq^n = a \times (1 - q^{n+1})/(1 - q) \quad \textbf{(72)}$$

Letting $a = 1, q = 1/2$, and $n = depth - 1$, we get

$$Operations = N \times 2 \times \left(1 - \frac{1}{2}^{depth}\right) \quad \textbf{(73)}$$

and as the depth grows, $1 - \frac{1}{2}^{depth} \rightarrow 1$ (i.e., even at *depth-4*, we have $(1 - \frac{1}{16} = \frac{15}{16}) \cong 1$), so that we can use the approximation that $Operations \cong 2N$.

For the *Haar* example, there are 2 computations per filter operation so we have roughly $4N = 32$ computations.

Generalizing to filters of length *L,* we get:

$$Computations = 2LN \quad \textbf{(74)}$$

which is order *N* with a multiplicative factor of *2L (*see also Theorem 1.3 of Strang and Nguyen (1997)).

Overlap to Avoid Blocking versus Added Complexity

We have stated that both the FFT and the simplest WPT, the *Haar*-based WPT, require $2N \log_2 N$ computations. One distinction between the FFT and the WPT is that in the case of the former, the block size and the spectral resolution are tied together. Our choice of block length determines the frequency resolution we achieve, or, the frequency resolution we require dictates the input signal block length.

For example, if we want 512 spectral lines (256 unique lines if it is a real signal, since negative frequencies are just the complex conjugates of the positive frequencies), we must use a block size of 512 samples. If we want to partition the spectrum into 16 parts, our input signal will only be 16 samples in length. But what happens between blocks? In image coding, they speak of *blocking artifacts*. These are the result of the discontinuities between blocks, since no filtering is done with samples that span two contiguous blocks. For short blocks, this can result in a good deal of blocking.

One way to deal with this and achieve smooth transitions from one transform block to the next is to overlap successive transforms, with for example a 50% overlap factor. This doubles both the number of transforms done (thereby doubling the computations to $4N \log_2 N$) and the number of transform coefficients produced, and means that the signal is no longer *critically sampled*.

In the case of the WPT, as long as the filter length is longer than the decimation factor, the transform operates as a lapped transform[†]. A lapped transform is one that has overlap across signal blocks in order to achieve smooth transitions. In the case of the FWPT, the decimation factor is 2, since we downsample by 2 with each filtering. If the filter impulse response spans even 3 samples for example, then any two adjacent filter results are based on at least one common sample point, independent of the depth of filtering. This means that as we transition from one block to the next, there is continuity.

Whereas in the case of the FFT we doubled our MIPS to get overlap, here we will also increase the complexity. Specifically, the *Haar* case corresponded to $2N \log_2 N$ computations, and its filter length is precisely equal to the decimation factor. As such it operates as a block transform, not a lapped transform. We showed earlier that for a filter of length L, the number of computations for the WPT are $LN \log_2 N$. So, for our L = 3 example, we have $LN \log_2 N$ computations. Longer filters will mean more complexity, but also better overlap and more subband isolation. In all cases, the transform will remain *critically sampled*.

[†] Malvar (1992, p. 224) states: "If the filter length for a particular subband is equal to its decimation factor R_k, that subband operates as a block transform, with block length R_k. If the filter length for the kth subband is greater than R_k, that subband operates as a lapped transform."

Appendix B
BDBBA Flowcharts and Pseudo-Code

This appendix provides a detailed description of the BDBBA through a series of flowcharts and accompanying algorithm descriptions. The flowcharts provide an overview of the routine as a whole, and the algorithm description segments, which are cross referenced to the flowcharts by step number within a given routine, add to the flowcharts by providing more detail†. *All-or-nothing* quantization is assumed in the description, however, it is easily generalized to other quantization choices. Also note that to reconcile the description of the BDBBA in *Chapter 4* with the information in this appendix, one must replace all *bandwidth B* related parameters here with the relative rate entity R/R_{Hi} of *Chapter 4*.

We have divided the BDBBA into three routines for the purpose of description††. These routines are the *Initialization*, *Main*, and *BestTree* routines. The *Initialization* (*I*) routine executes a series of one-time operations which must occur before *Main (M)*. *Main* then controls the overall operation of the algorithm, including the bisection operation to determine the optimal solution for the bandwidth budget specified (the *inner* or *bandwidth* loop), and the optional outer node loop when operating in *node mode*. To determine the optimal best basis tree for a given Lagrange multiplier, *Main* calls the *BestTree (B)* routine. As such, *BestTree (B)* is a subroutine of *Main*†††.

These routines are described in the following. The terminology used in those descriptions is defined in the following tables. As described earlier in Section *3.7.2 Rate-Distortion Best Basis Algorithm (RDBBA)*, we use the following two modifiers for the variables. The asterisk (*) is appended to a variable that is associated with a solution. For example $S^*(\lambda)$ is the solution or best tree for the given λ. λ^* is the Lagrange multiplier solution which produces a basis that best satisfies the budget and results in convergence of the bandwidth (inner) loop of the algorithm. $S^*(\lambda^*)$ is the tree associated with that multiplier. A tilde (~) is

---

† The style of algorithm description and flowcharts is taken from Knuth (1997), as described on pp.2-3 of that reference.

†† These descriptions are intended to describe the basic functionality and general architecture of the code. They should not be interpreted to imply any details of code implementation (for example, the "*If - Go To"* approach used to describe the loops would typically be implemented as *while* loops).

††† In our implementation, *BestTree* is also used by the *Initialization* routine to verify its initial values for the Lagrange multipliers surround the eventual solution.

placed above a variable that is associated with the lowest cost solution for a given node and Lagrange multiplier λ For example, $\tilde{D}_j^i$ is the distortion of node (*i, j*) associated with the minimum cost solution for that node.

Table B.1 summarizes the inputs to the algorithm, and *Table B.2* summarizes the algorithm variables and data structures.

Table B.1 BDBBA Inputs

Name	Description
B_{Budget}	Bandwidth budget parameter, which specifies the relative amount of bandwidth the BDBBA should retain in its selection of a basis ($0 \le B_{Budget} \le 1$).
$BWResolution$	Resolution of bandwidth variable *(*and smallest bandwidth budget possible ($1/2^d$).
$\{c_j^i\}_k$	WPT transform coefficients. *i, j,* and *k* denote scale, node number, and block respectively.
d	Depth of the tree T associated with the initial WPT ($d = log_2 N$).
L_{budget}	Specifies number of *leaves* or *terminal nodes* BDBBA should keep when *Mode = Node* ($1 \le L_{Budget} \le N$).
M	Number of blocks (each of size N) in the input signal.
Mode	Variable specifying either Bandwidth or Node mode of operation.
N	Number of samples in signal block ($N = 2^d$).
T	Tree associated with the original WPT.

Table B.2 BDBBA Variables and Data Structures

Name	Description
λ	Lagrange multiplier.
$B^*(\lambda)$	Total bandwidth of best subtree $S^*(\lambda)$ for given λ.[1]
$BWBudgetsList$	Ordered list of bandwidth (BW) budgets. B_{Budget} is first entry. Next entries are progressively further from B_{Budget}, alternating above and below it, in steps of the BWResolution (*Node* mode only).
$[BD]$	Matrix of 2-point Bandwidth versus Distortion curves for each node in the complete tree T.
$\{\tilde{B}_j^i \tilde{D}_j^i \tilde{J}_j^i split(n_j^i)\}$	Structure for each node n_j^i containing $\tilde{B}_j^i$, $\tilde{D}_j^i$, $\tilde{J}_j^i$, and split(n_j^i). $\tilde{B}_j^i$ is the bandwidth[2], $\tilde{D}_j^i$ is the distortion, $\tilde{J}_j^i$ the Lagrangian cost, and split(n_j^i) the *Split/Don't Split* binary variable of the subtree rooted at n_j^i for the given λ. ($[\tilde{B},\tilde{D},\tilde{J},split]$ denotes the matrix of these structures for all nodes in the tree).
Converged	Binary variable indicating whether BDBBA has converged for the given budget B_{Budget}.
$D^*(\lambda)$	Total distortion of best subtree $S^*(\lambda)$ for given λ.
E_j^i	Energy of WPT coeffs of node n_j^i across all blocks in the input signal.
L	Number of *leaves* or *terminal nodes* associated with a given subtree
L_Δ	Delta between L of a subtree tree and the leaf budget L_{budget}.
$L_{\Delta\min}$	Smallest *NodeDelta* produced by current run of BDBBA so far.
LamdaBWTable.	Ordered entries of [λ^*, B_{Budget}, $D^*(\lambda^*)$] for the BW budgets (B_{Budget}) tried in the process of finding a basis to meet the $Nodes_{Budget}$ when *Mode = Node*.
L_Δ	Delta between $NodeCount$ of a subtree tree and the $Nodes_{Budget}$.
$L_{\Delta\min}$	Smallest *NodeDelta* produced by current run of BDBBA so far.
n_j^i	The *j*th node at the *i*th scale of T.
$S(\lambda)$	A pruned subtree of T.($S^*(\lambda^*)$ is the pruned best tree that satisfies the bandwidth budget).
TreeFound	Binary state variable indicating whether BDBBA has run to completion. In *BW* Mode,*TreeFound* = 1 when inner loop converges (*Converged* = 1). For *Node* Mode, *TreeFound* = 1 only when $Nodes_{Budget}$ is satisfied, or all possible B_{Budget} 's have been tried.

1.Constrained to be from a set of discrete values of resolution *BWResolution.*
2.This is analogous to the variable *rate* used in the rate-distortion approach (Ramchandran and Vetterli 1993).

B.1 *Initialization* Routine

As stated earlier, the *Initialization* routine performs a series of one-time initialization operations prior to the main basis search in routine *Main* (*M*). A flowchart of the *Initialization* routine is given in Fig.*B-1*.

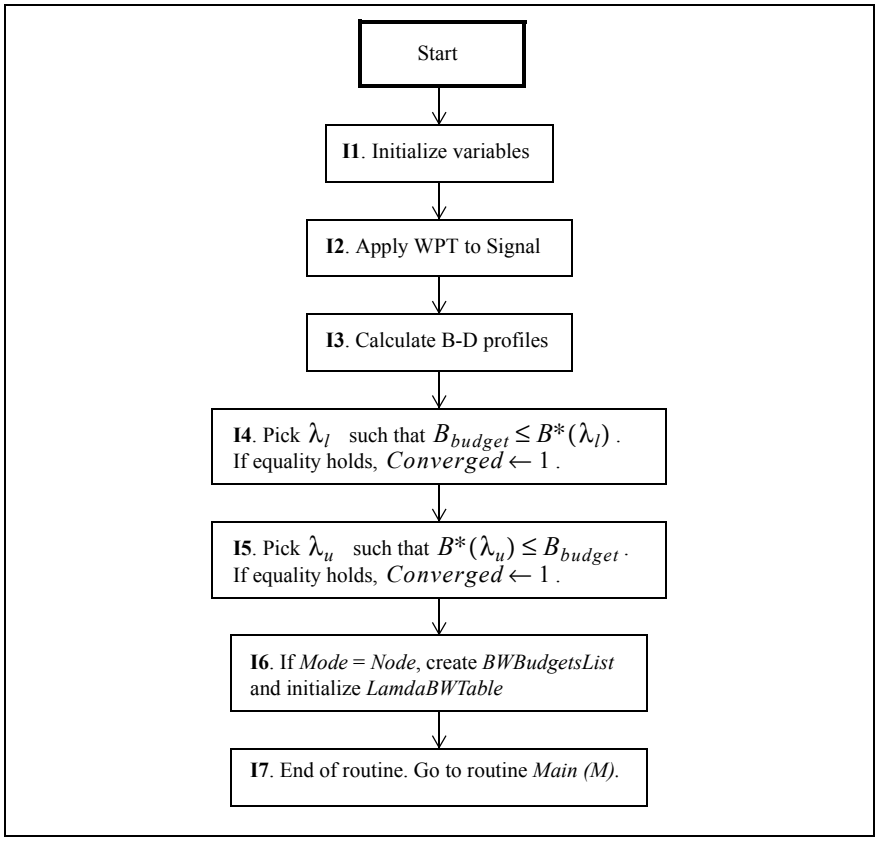

Figure B-1: **Flowchart of *Initialization (I)* Routine**

Now we provide a detailed description of the routine.

***Initialization* Routine Algorithm Description**

I1. [Initialize variables].
Initialize inner loop control variable *Converged* to 0. Initialize outer loop control variable *TreeFound* to 0. If *Mode = Node*, initialize $NodeDelta_{min}$ to ∞.

I2. [Apply WPT to signal.]
Perform WPT on *M* blocks of *N* input samples each to generate complete set of transform coefficients $\{c_j^i\}_k$ to depth *d* (*d* = $log_2 N$).

I3. [Calculate *B-D* profiles].
For each node in the complete tree T calculate the 2-point Bandwidth vs. Distortion profile. Store result in matrix [*BD*]. Bandwidth $B_j^i = keep_j^i \times (\frac{1}{2})^i$. Distortion $D_j^i = (1 - keep_j^i) \times (K_{psych})_j^i \times E_j^i$ (see *Equation* (70)). One point on the

curve corresponds to keeping the node ($keep_j^i$ = 1). Therefore no distortion is added. The other point corresponds to discarding the node coefficients ($keep_j^i$ = 0). Then the distortion added is the energy of that node's coefficients across all blocks *M* of the signal E_j^i, weighted by a psychoacoustic weighting factor.

I4. [Pick λ_l such that $B_{budget} \le B^*(\lambda_l)$. If equality holds, $Converged \leftarrow 1$.]

Pick lower bound for the Lagrange multiplier λ such that bandwidth budget is less than or equal to the bandwidth associated with this lower bound ($B_{budget} \le B^*(\lambda_l)$). ($B^*(\lambda_l)$ is determined by successive calls to routine *BestTree* with decreasing values of λ_l, as in: $[\tilde{B}\tilde{D}\tilde{J}split]$ = *BestTree*(λ_l,$[BD]$,T), until the inequality is satisfied†.). If $B_{budget} = B^*(\lambda_l)$, $Converged \leftarrow 1$. (Note: in Node Mode, λ_l is always set to 0).

I5. [Pick λ_u such that $B^*(\lambda_u) \le B_{budget}$. If equality holds, $Converged \leftarrow 1$.]

Pick upper bound for the Lagrange multiplier λ such that the bandwidth budget is greater than or equal to the bandwidth associated with this upper bound ($B^*(\lambda_u) \le B_{budget}$). ($B^*(\lambda_u)$ is determined by successive calls to routine *BestTree* with increasing values of λ_u, as in:

$[\tilde{B}\tilde{D}\tilde{J}split]$ = *BestTree*(λ_u,$[BD]$,T), until the inequality is satisfied). If $B_{budget} = B^*(\lambda_u)$, $Converged \leftarrow 1$. (Note: in Node Mode, λ_u is always set to infinity).

I6. [If *Mode* = *Node*, create *BWBudgetsList* and initialize *LamdaBWTable*.]

If *Mode* = *Node*, create the *BWBudgetsList* with all possible budgets, ordered from the selected budget first, followed by budgets whose difference in magnitude from the selected budget progressively increase. Initialize *LamdaBWTable*. This initial version will have two entries, one corresponding to λ_l and one for λ_u.

I7. [End of routine. Go to routine *Main*(*M*).]

End of *Initialization* (*I*) routine. Go to Routine *Main (M)*. |

B.2 *Main* Routine

The *Main* routine controls overall operation of the BDBBA, including the inner and optional outer loops of the BDBBA. When in bandwidth mode, there is only an inner loop (the outer loop is a formality). In that case, the algorithm must only find the best basis for the supplied bandwidth budget B_{Budget}. When in node mode, the outer loop attempts to find a basis that meets or comes closest to the supplied node budget L_{Budget}. This will involve invoking the inner loop a number of times with successive bandwidth budgets until the node budget requirement is met or there are no more bandwidth budgets to try. A flowchart of *Main* is given in Fig.*B-2*.

---

†. This final value is retained during the basis selection session so that subsequent invocations of the algorithm can start with a more accurate estimate for this bound. The same occurs for the upper bound in the next step. This approach is more efficient than simply choosing safe but highly conservative values for these bounds, such as zero and infinity.

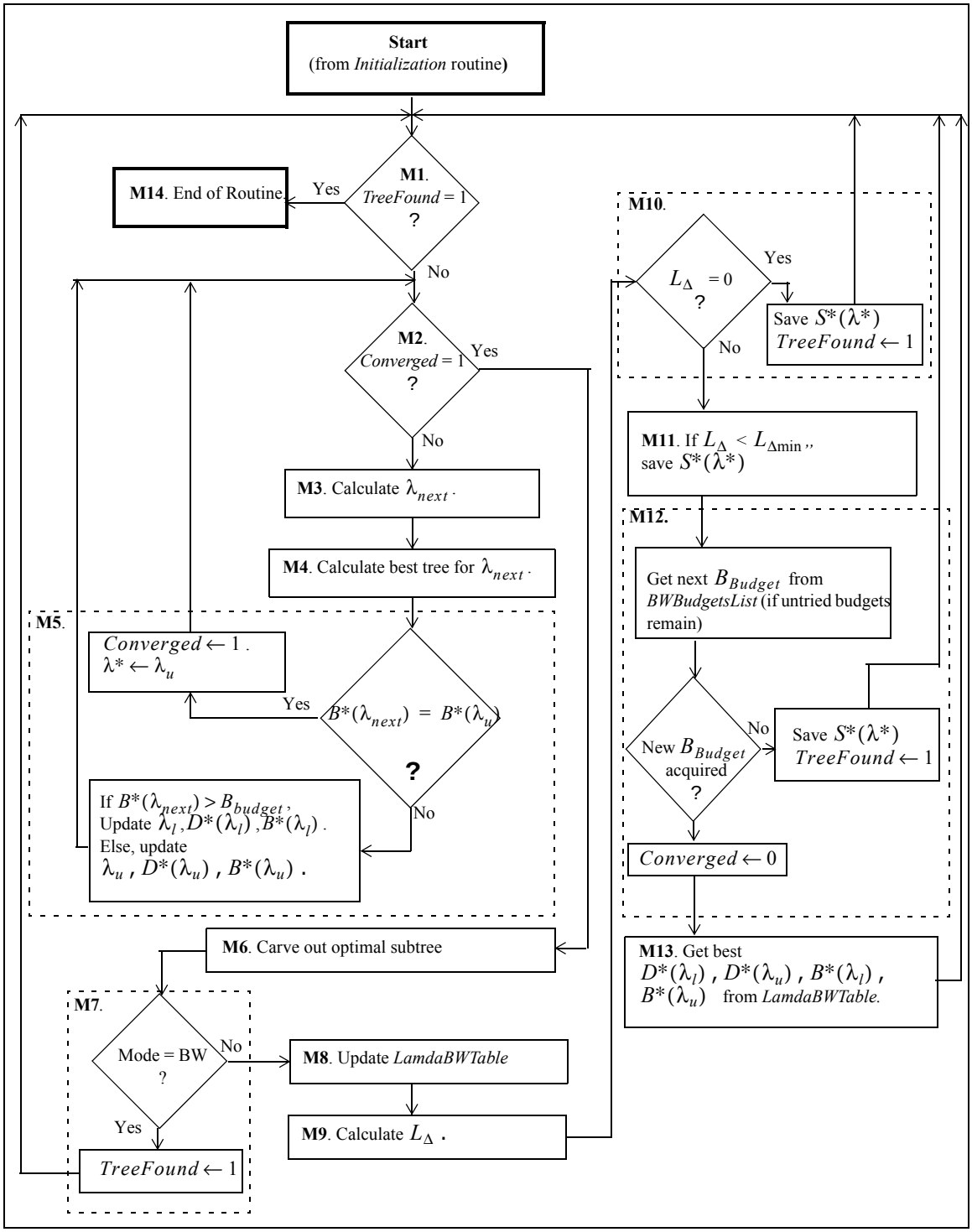

Figure B-2: **Flowchart of *Main (M)* Routine**

Now we provide a detailed description of the routine.

Main Routine Algorithm Description

M1. [If *TreeFound* = 1, go to M14.]

If the best tree has been found (*TreeFound* = 1), the algorithm has finished so go to the end (M14). This step controls execution of the BDBBA *Node (*outer*) Loop*. (*TreeFound* is never equal to 1 upon entry to this routine (*Main*) form the *Initialization* routine).

M2. [If *Converged* = 1, go to M6.]

If the inner loop has converged (*Converged* = 1), then a solution for the given bandwidth budget has been found. Proceed to step M6. This step controls execution of the BDBBA *Bandwidth (*inner*) Loop*.

M3. [Calculate λ_{next}.]

Calculate next Lagrange multiplier λ_{next} via bisection as follows:

$\lambda_{next} \leftarrow (\left|D^*(\lambda_l) - D^*(\lambda_u)\right| / \left|B^*(\lambda_l) - B^*(\lambda_u)\right|) + \varepsilon$ where ε is a vanishingly small number (to ensure that.... (p.169 Step 2. --> show what I use --> look for notes on reasoning)

M4. [Calculate best tree for λ_{next}.]

Calculate the lowest cost basis by calling *BestTree* routine with λ_{next} as per: $[\tilde{B}\tilde{D}\tilde{J}split]$ = *BestTree*(λ_{next}, $[BD]$, T).

M5. [If $B^*(\lambda_{next}) = B^*(\lambda_u)$, $Converged \leftarrow 1$, $\lambda^* \leftarrow \lambda_u$. Else, update $\lambda_l, D^*(\lambda_l), B^*(\lambda_l)$, or $\lambda_u, D^*(\lambda_u), B^*(\lambda_u)$. Go to M2.]

If $B^*(\lambda_{next}) = B^*(\lambda_u)$, $Converged \leftarrow 1$, $\lambda^* \leftarrow \lambda_u$.

Elseif $B^*(\lambda_{next}) > B_{budget}$

$\lambda_l \leftarrow \lambda_{next}$, $D^*(\lambda_l) \leftarrow D^*(\lambda_{next})$, $B^*(\lambda_l) \leftarrow B^*(\lambda_{next})$.

Else

$\lambda_u \leftarrow \lambda_{next}$, $D^*(\lambda_u) \leftarrow D^*(\lambda_{next})$, $B^*(\lambda_u) \leftarrow B^*(\lambda_{next})$.

Go to M2.

M6. [Carve out optimal subtree.]

Carve out optimal subtree $S^*(\lambda^*)$ based on *split* variable and zero coefficients of nodes with *keep* = 0 (if *KeepAllCoeffs* = No).

M7. [If Mode = BW, $TreeFound \leftarrow 1$. Go to M1.]

If in bandwidth mode (Mode = BW), set $TreeFound \leftarrow 1$. Go to M1.

M8. [Update *LamdaBWTable*.]

The updated *LamdaBWTable* gains a new entry corresponding to the pruned subtree $S^*(\lambda^*)$ from the last bandwidth budget B_{Budget} run.

M9. [Calculate L_Δ.]

Calculate $L_\Delta = \left|L - L_{budget}\right|$.

M10. [If L_Δ = *0,* save $S^*(\lambda^*)$ and $TreeFound \leftarrow 1$. Go to M1.]

If the number of nodes L is equal to the node budget L_{Budget} then the best basis tree has been determined. This occurs when L_Δ = *0.* Set $TreeFound \leftarrow 1$, save $S^*(\lambda^*)$, and go to M1.

M11. [If L_Δ < $L_{\Delta\min}$ save $S^*(\lambda^*)$.]

If the number of terminal nodes associated with this basis is the closest to the node budget so far ($L_\Delta < L_{\Delta\min}$), then save $S^*(\lambda^*)$, since it is possible that no solution for the given L_{Budget} exists. In that case, the pruned tree $S^*(\lambda^*)$ with the smallest L_Δ is declared the solution. To achieve this efficiently (more memory, but less MIPS), the best tree so far is saved.

M12. [Get next B_{Budget} from *BWBudgetsList* if untried budgets remain, and $Converged \leftarrow 0$. Else, $TreeFound \leftarrow 1$ and go to M1.]

If not all bandwidth budgets B_{Budget} in the ordered list *BWBudgetsList* have been tried, set B_{Budget} to the next value in the list. $Converged \leftarrow 0$. Else, set $TreeFound \leftarrow 1$ and go to M1.

M13. [Get best $D^*(\lambda_l), D^*(\lambda_u), B^*(\lambda_l), B^*(\lambda_u)$ from *LamdaBWTable*. Go to M1.]

Update $D^*(\lambda_l), D^*(\lambda_u), B^*(\lambda_l), B^*(\lambda_u)$ based on location of current B_{Budget} relative to entries in *LamdaBWTable*.

First, get entry *i* in *LamdaBWTable* for which B^* value is the smallest possible value that is still larger than B_{Budget}. Then $\lambda_l \leftarrow \lambda_i$, $D^*(\lambda_l) \leftarrow D^*_i$, and $B^*(\lambda_l) \leftarrow B^*_i$

Next, get entry *i* in *LamdaBWTable* for which B^* value is the largest possible value that is still smaller than B_{Budget}. Then $\lambda_u \leftarrow \lambda_i$, $D^*(\lambda_u) \leftarrow D^*_i$, and $B^*(\lambda_u) \leftarrow B^*_i$.

Go to M1.

M14. [End of routine.]

End of *Main* routine.|

B.3 *BestTree* Routine

This routine is called from both the *Main* and *Initialization* routines. It determines the minimum cost pruned tree $S^*(\lambda)$ of T for a given λ [†]. A flowchart of *BestTree* is given in Fig.*B-3*.

[†] The routine would lend itself to an "in-place" implementation, even though separate matrices are currently used for the output matrix that holds the final result $[\tilde{B},\tilde{D},\tilde{J},split]$ and the matrix that has the per node BW and Distortion information independent of other nodes and splits (NDJSoptMatrix).

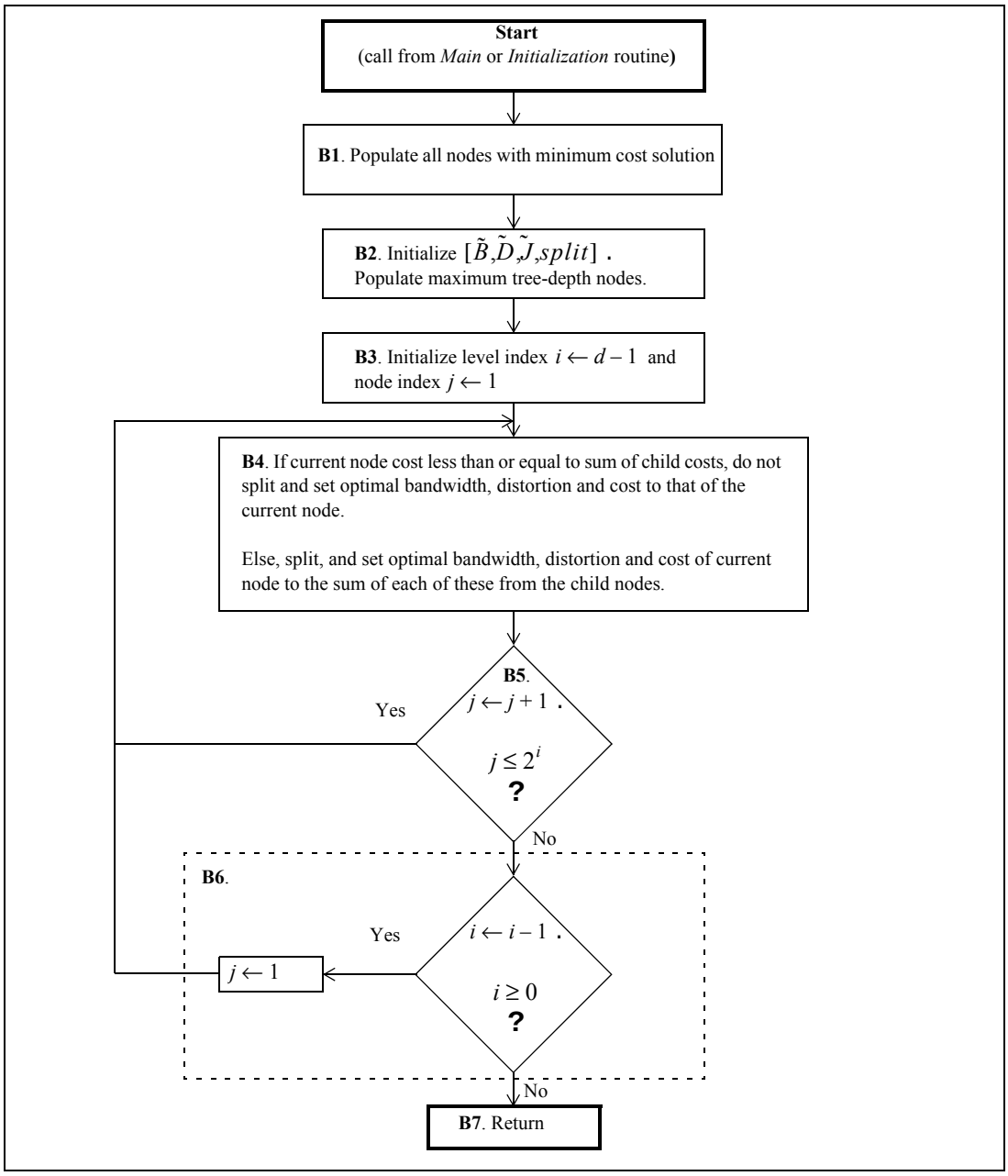

Figure B-3: Flowchart of BestTree Routine

Now we provide a detailed description of the routine.

`BestTree` Routine Algorithm Description

(Call format:$[\tilde{B},\tilde{D},\tilde{J},split]$ = `BestTree`$(\lambda,[BD],T)$,

B1. [Populate all nodes with minimum cost solution.]

Calculate $(J_j^i)_{min} = min(D_j^i + \lambda B_j^i)$ (from (66)) for all nodes in T given the specified λ and $[BD]$ array of 2-point Distortion-Bandwidth curves. Store the corresponding solution $(D_j^i)_{min}$ and $(B_j^i)_{min}$ that results in the minimum cost.

B2. [Initialize $[\tilde{B},\tilde{D},\tilde{J},split]$. Populate maximum tree-depth nodes.]

The return matrix $[\tilde{B},\tilde{D},\tilde{J},split]$ is created and its maximum tree-depth nodes are populated with the minimum cost ($(J_j^i)_{min}$), distortion ($(D_j^i)_{min}$), and bandwidth ($(B_j^i)_{min}$) values calculated for those nodes in B1.

B3. [Initialize level index $i \leftarrow d-1$ and node index $j \leftarrow 1$.]

Set tree level index $i \leftarrow d-1$. Set Node index $j \leftarrow 1$.

B4. [If current node cost less than or equal to sum of child costs, do not split and set optimal bandwidth, distortion and cost to that of the current node. Else, split, and set optimal bandwidth, distortion and cost of current node to the sum of each of these from the child nodes.]

If $J_j^i(\lambda) \le \tilde{J}_{2j-1}^{i+1}(\lambda) + \tilde{J}_{2j}^{i+1}(\lambda)$

$split(n_j^i) \leftarrow NO;\ \tilde{B}_j^i \leftarrow B_j^i;\ \tilde{D}_j^i \leftarrow D_j^i;\ \tilde{J}_j^i \leftarrow J_j^i$

Else

$split(n_j^i) \leftarrow YES;\ \tilde{B}_j^i \leftarrow (\tilde{B}_{2j-1}^{i+1} + \tilde{B}_{2j}^{i+1});\ \tilde{D}_j^i \leftarrow (\tilde{D}_{2j-1}^{i+1} + \tilde{D}_{2j}^{i+1});\ \tilde{J}_j^i \leftarrow (\tilde{J}_{2j-1}^{i+1} + \tilde{J}_{2j}^{i+1})$

B5. [$j \leftarrow j+1$. If $j \le 2^i$ go to B4.]

Increment j by 1($j \leftarrow j+1$). If $j \le 2^i$ go to B4.

B6. [$i \leftarrow i-1$. If $i \ge 0$, $j \leftarrow 1$ and go to B4.]

Decrement by 1($i \leftarrow i-1$). If $i \ge 0$, set $j \leftarrow 1$ and go to B4.

B7. [Return.]

Return to calling routine.|

Appendix C
Sequence of Trees Example

In this appendix we show the sequence of trees (or bases) that result from sweeping the bandwidth budget from its minimum non-zero value to its maximum non-unity value. A budget of zero is not shown, since it is uninteresting to keep nothing of the signal. A budget of one (100% of bandwidth) is also uninteresting. The trees are all based on the *Stairs1* sound clip.

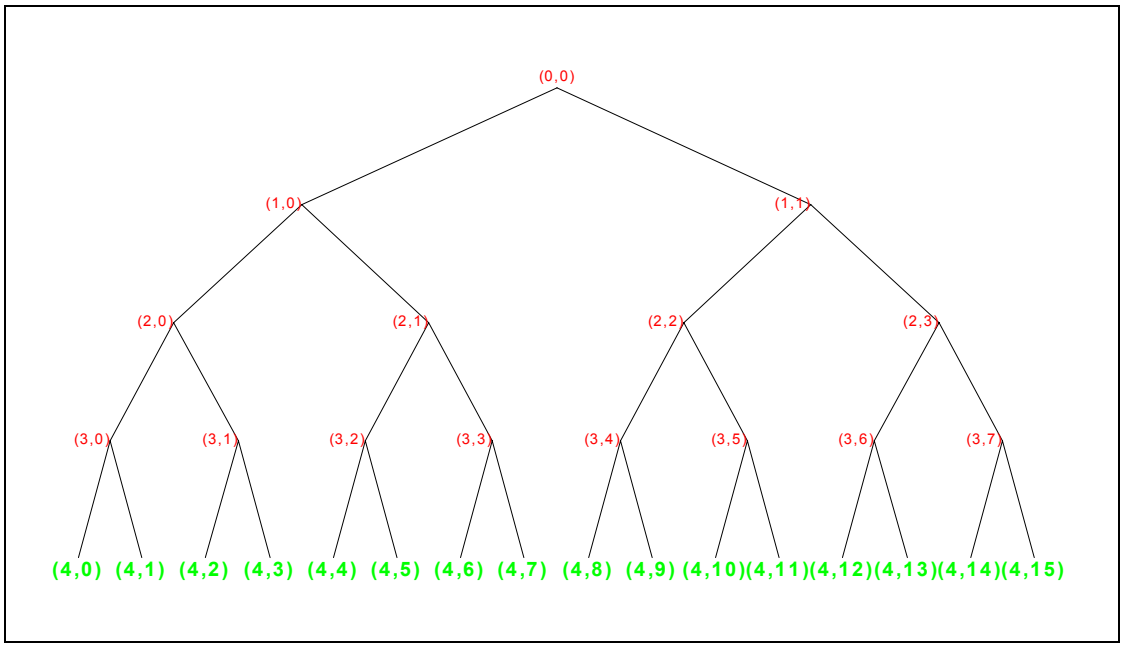

Figure C-1: Full Depth-4 Basis for Stairs1

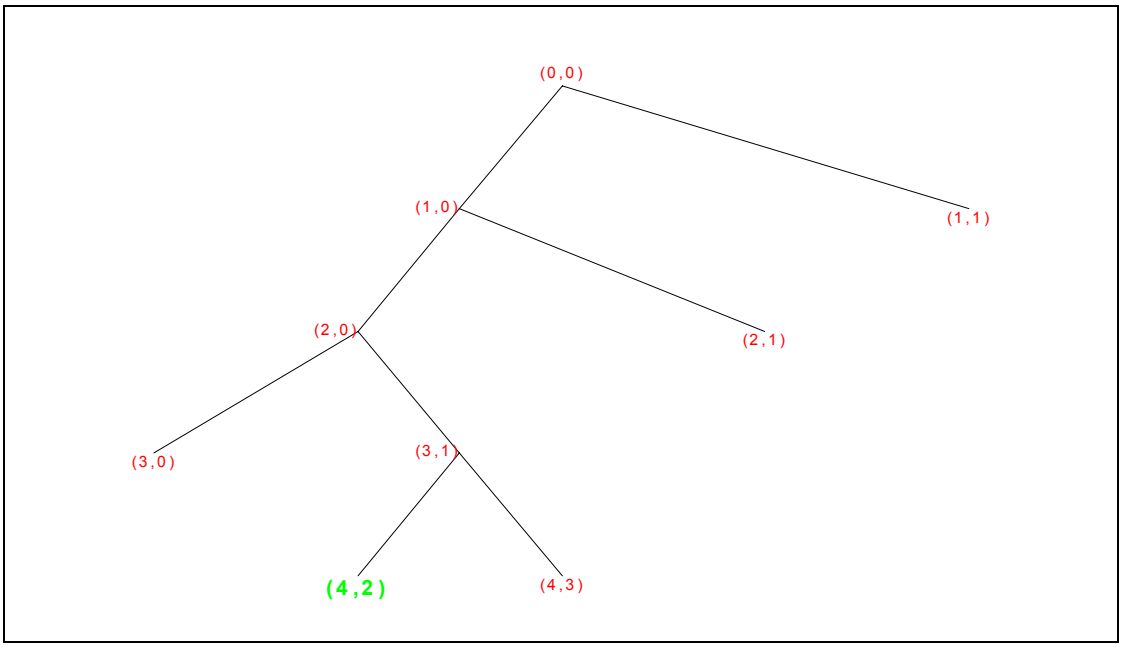

Figure C-2: BDBBA Basis for B_{budget}= 6.25% (Stairs1BMode1)

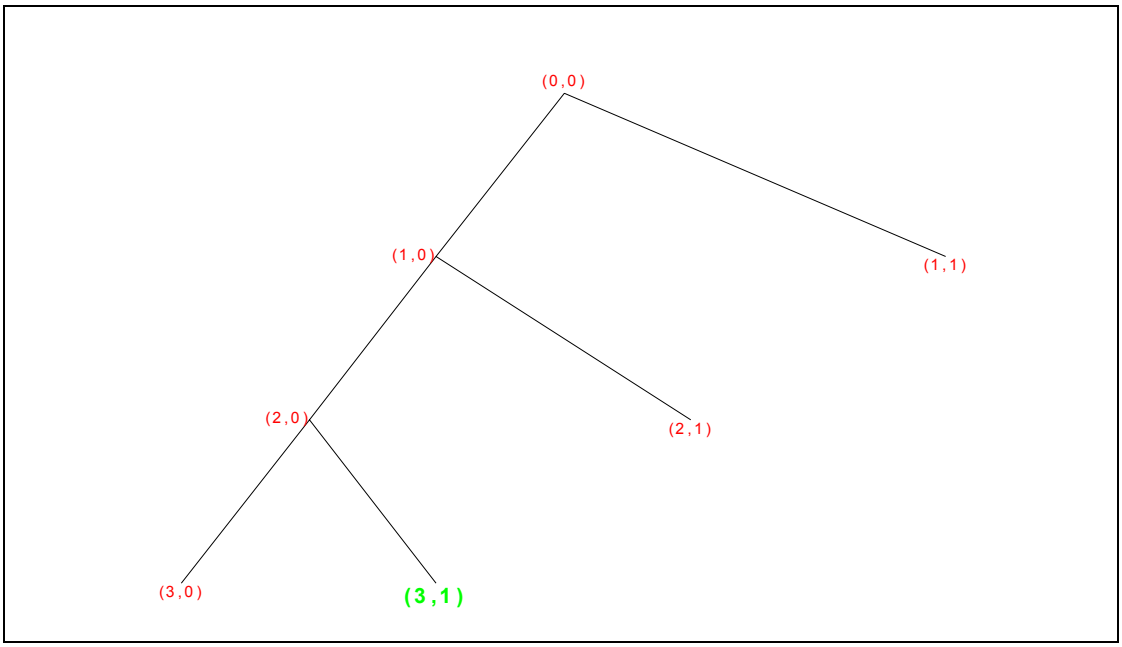

Figure C-3: BDBBA Basis for B_{budget}= 12.5% (Stairs1BMode2)

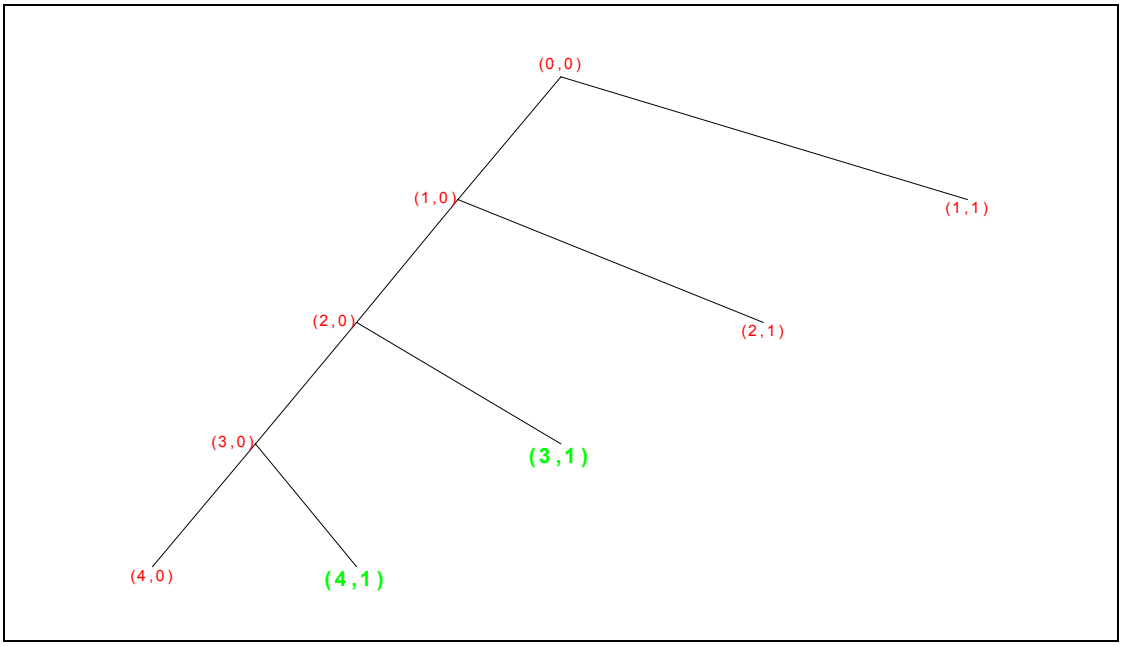

Figure C-4: BDBBA Basis for B_{budget}= 18.75% (Stairs1BMode3)

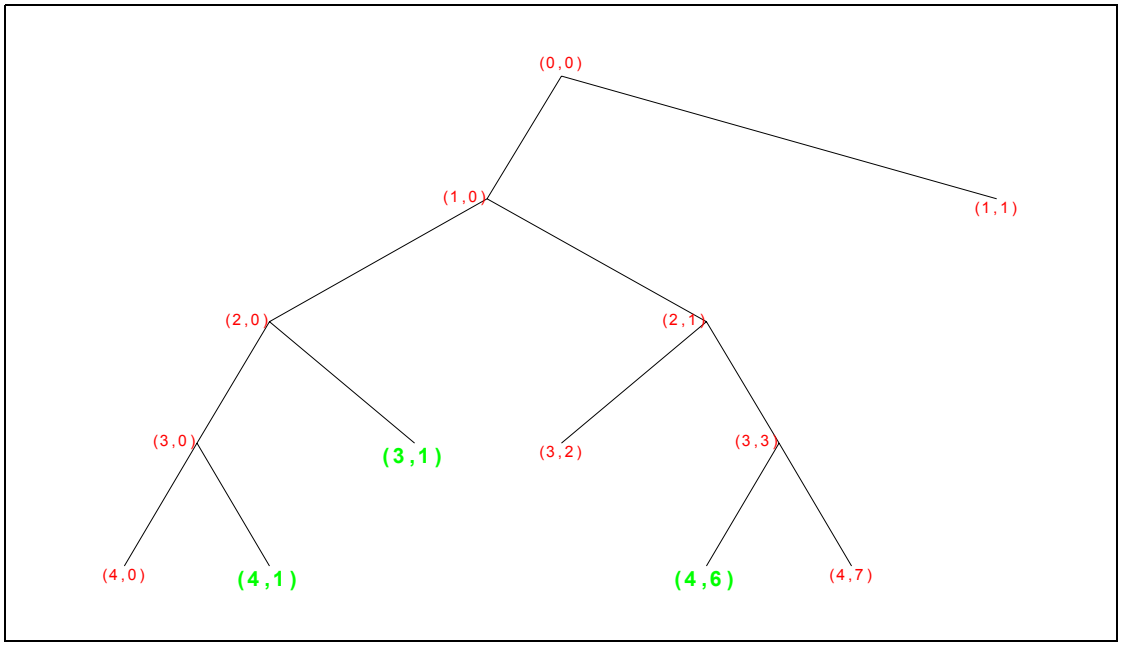

Figure C-5: BDBBA Basis for B_{budget}= 25% (Stairs1BMode4)

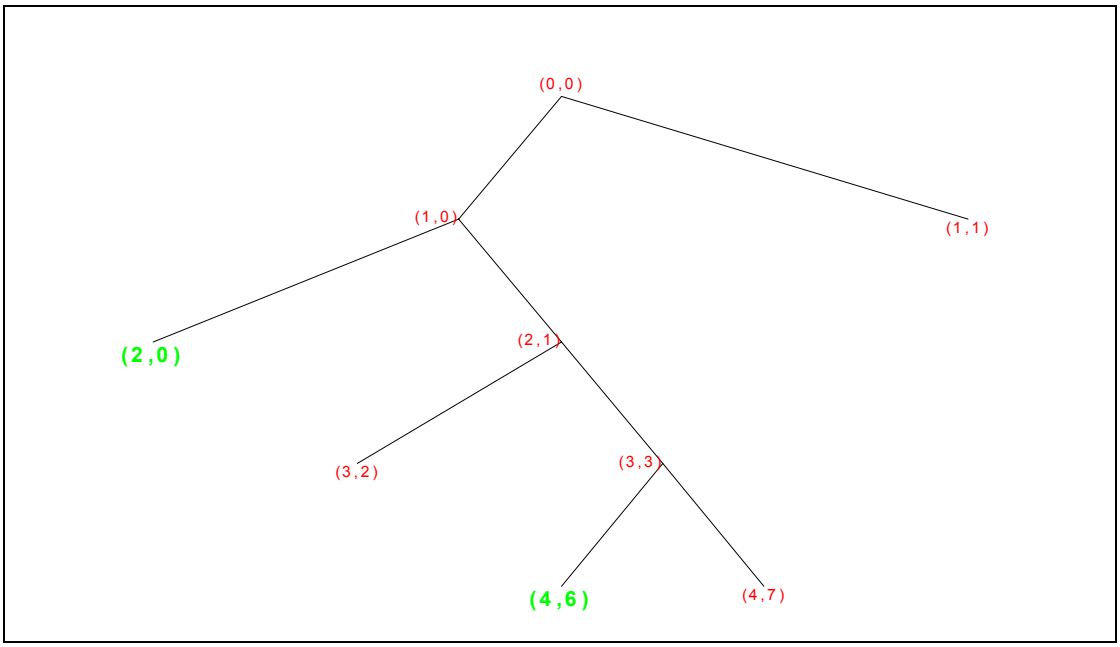

Figure C-6: BDBBA Basis for B_{budget}= 31.25% (Stairs1BMode5)

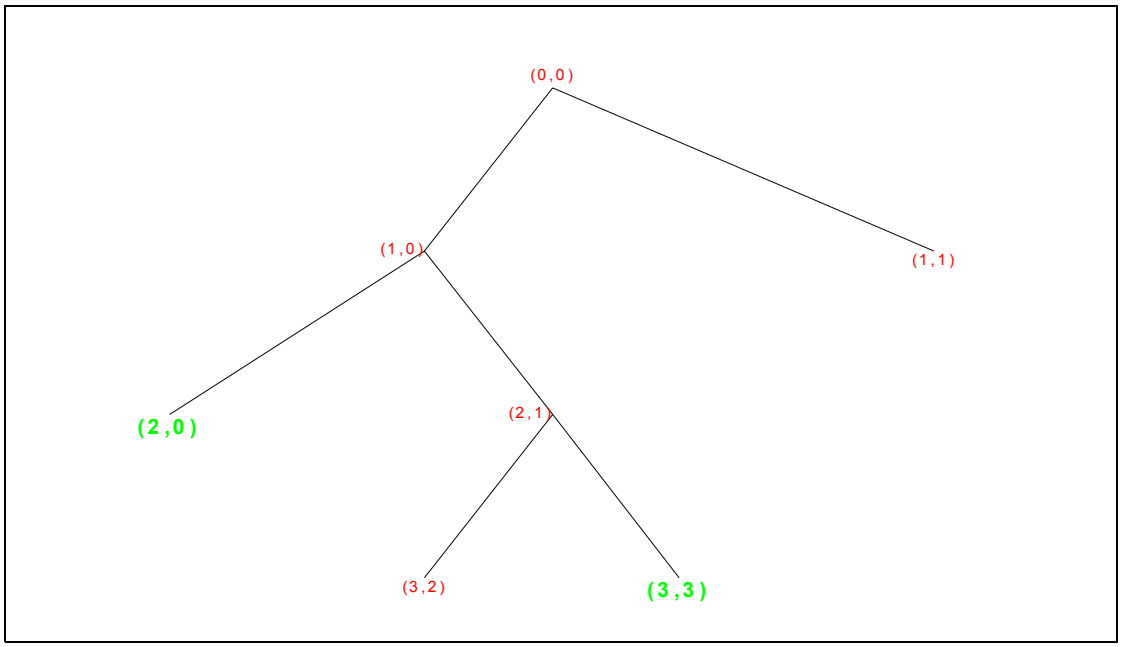

Figure C-7: BDBBA Basis for B_{budget}= 37.5% (Stairs1BMode6)

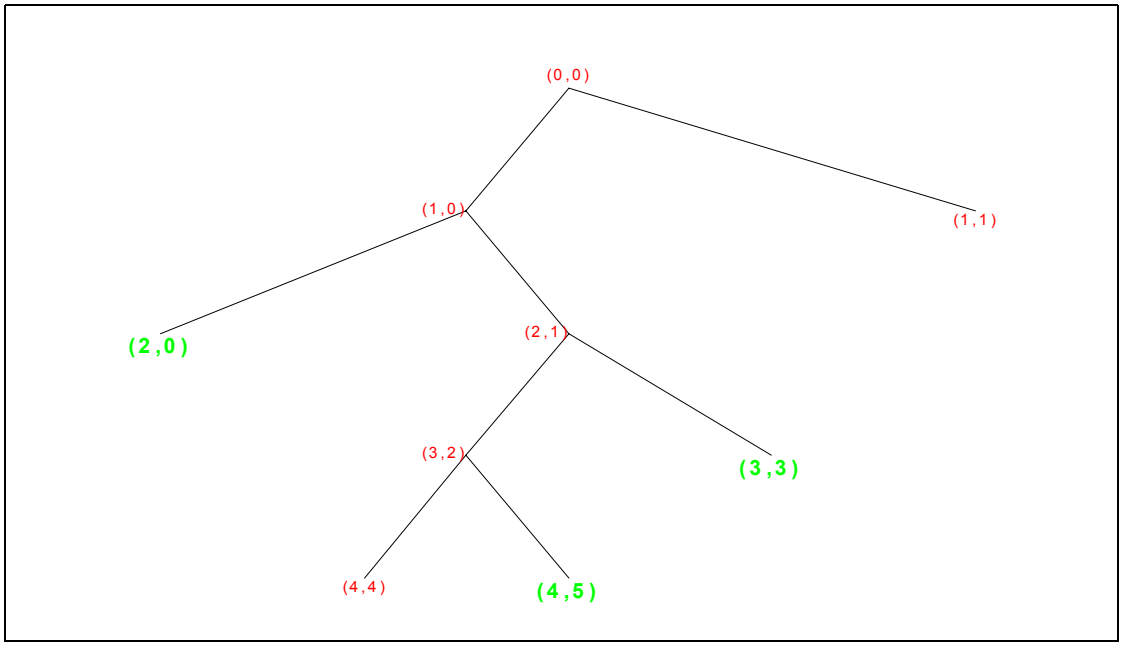

Figure C-8: BDBBA Basis for B_{budget}= 43.75% (Stairs1BMode7)

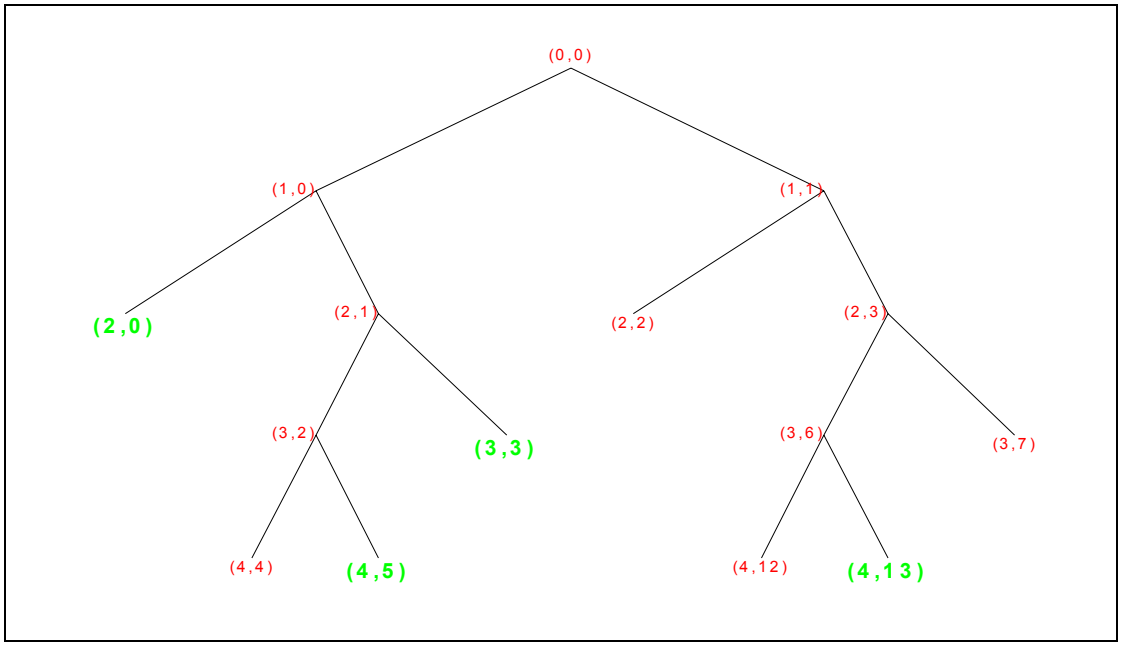

Figure C-9: BDBBA Basis for B_{budget}= 50% (Stairs1BMode8)

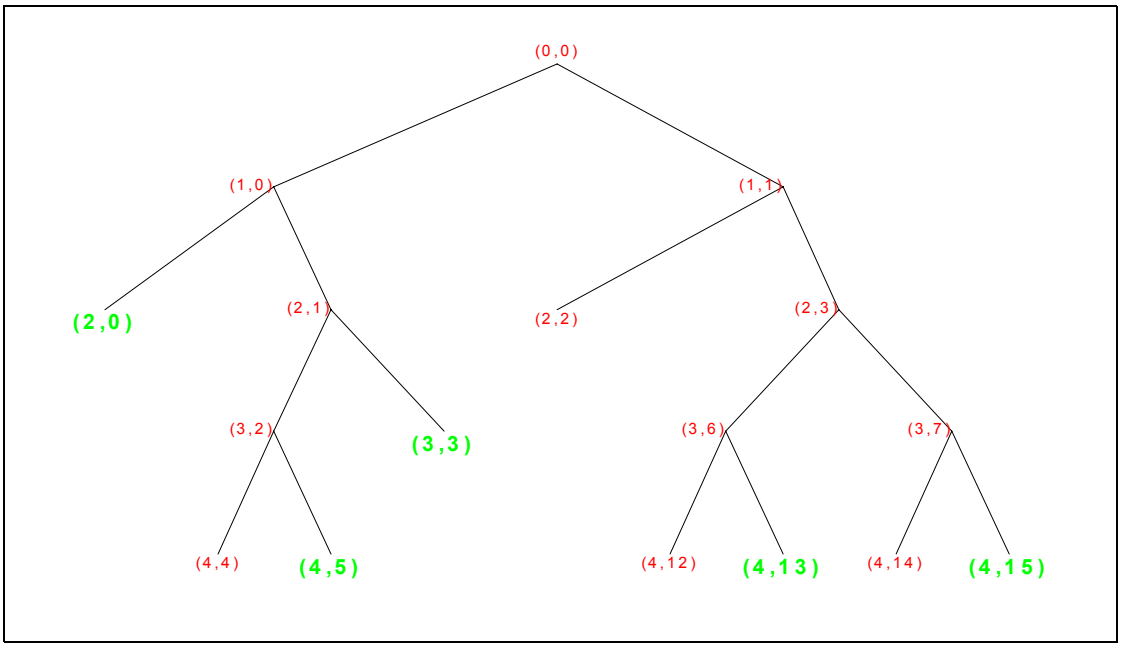

Figure C-10: BDBBA Basis for B_{budget}= 56.25% (Stairs1BMode9)

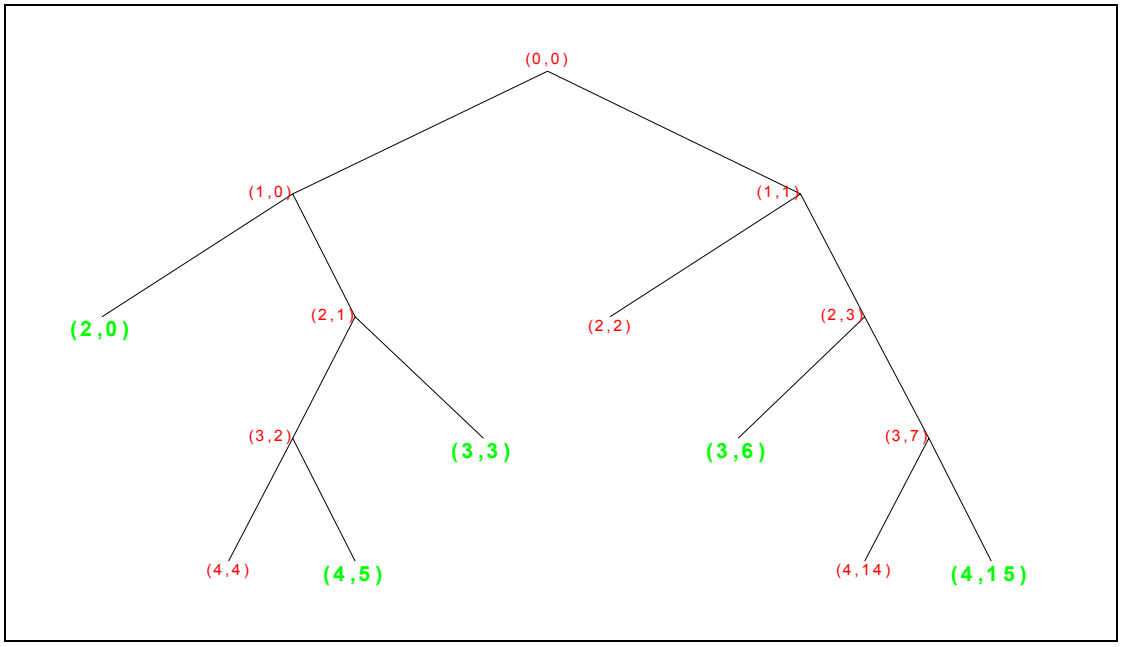

Figure C-11: BDBBA Basis for B_{budget}= 62.5% (Stairs1BMode10)

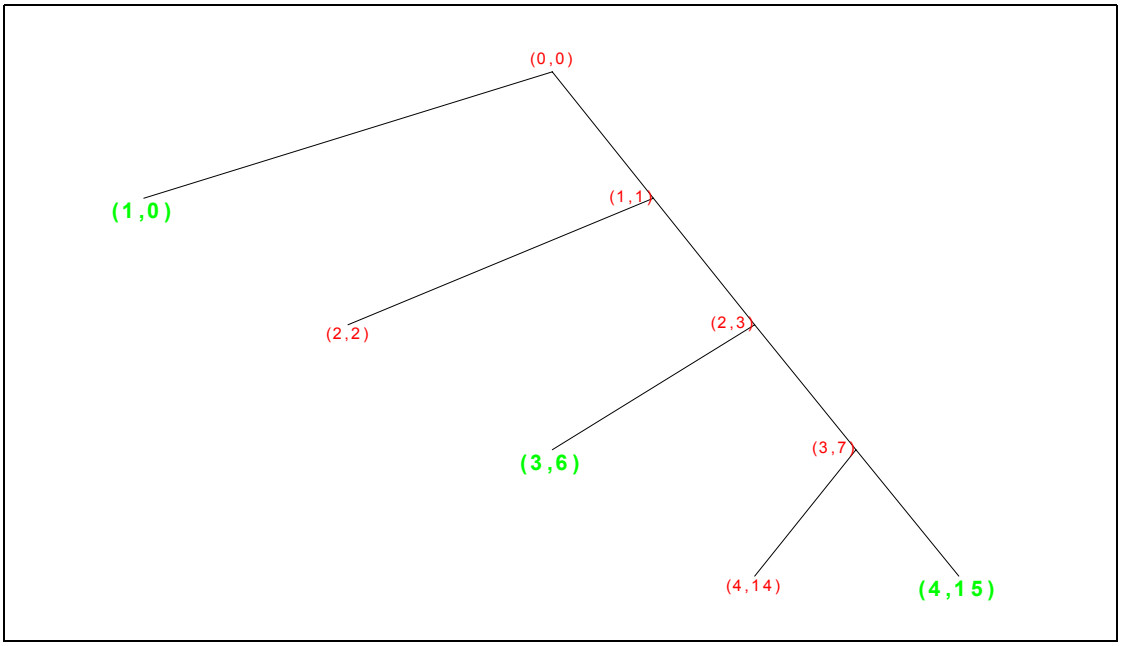

Figure C-12: BDBBA Basis for B_{budget}= 68.75% (Stairs1BMode11)

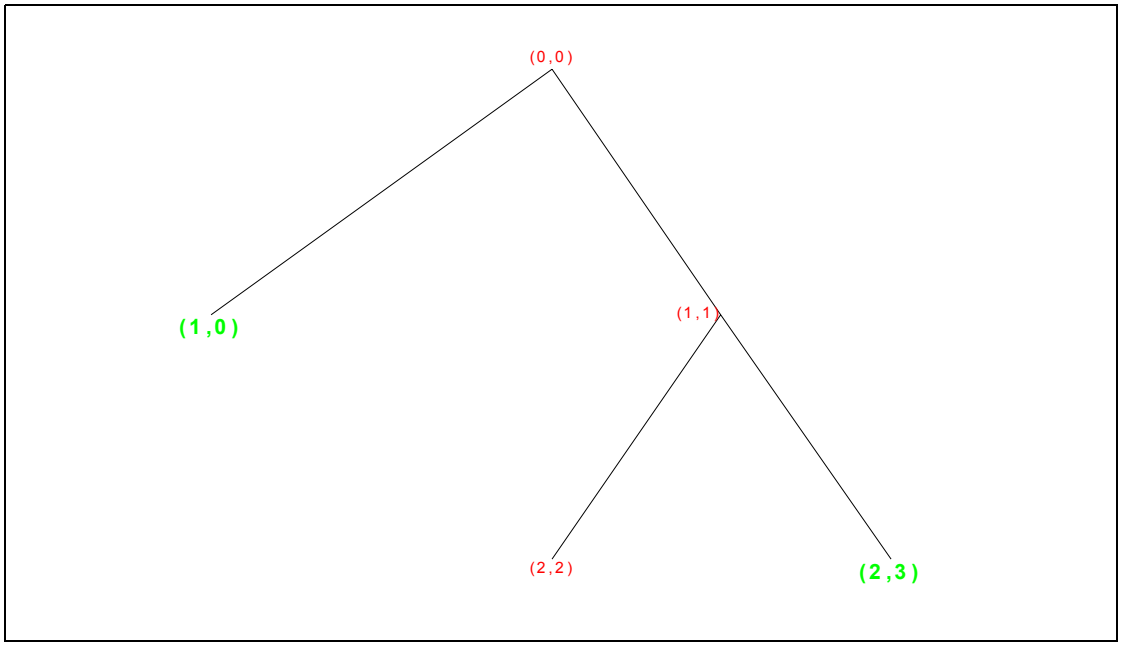

Figure C-13: BDBBA Basis for B_{budget}= 75% (Stairs1BMode12)

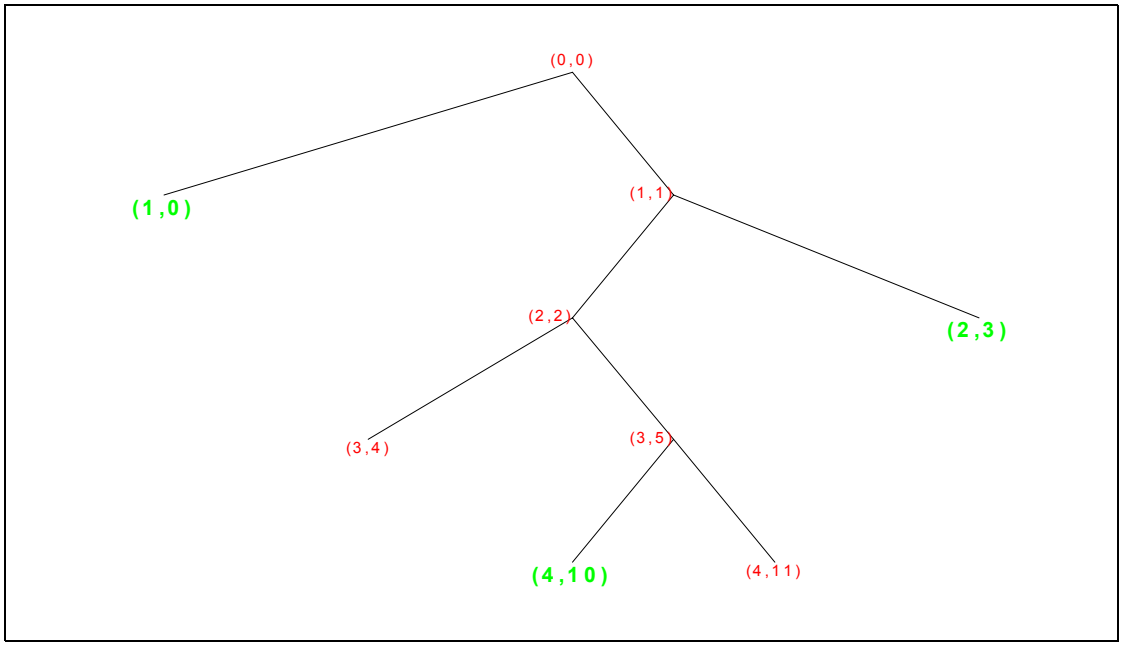

Figure C-14: BDBBA Basis for B_{budget}= 81.25% (Stairs1BMode13)

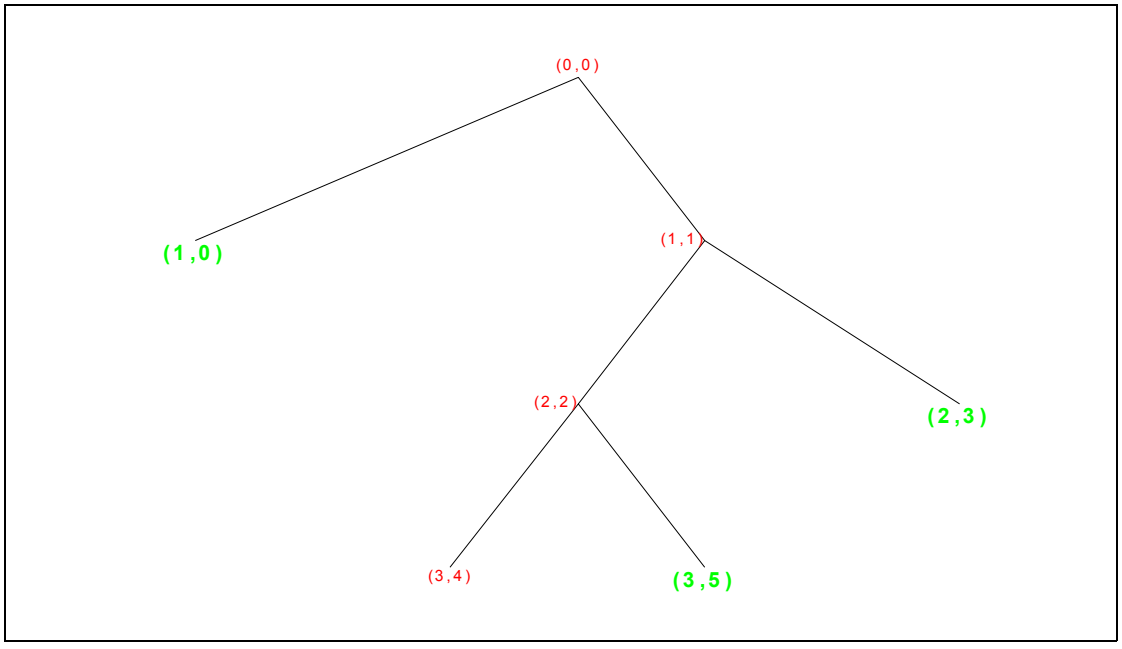

Figure C-15: BDBBA Basis for B_{budget}= 87.5% (Stairs1BMode14)

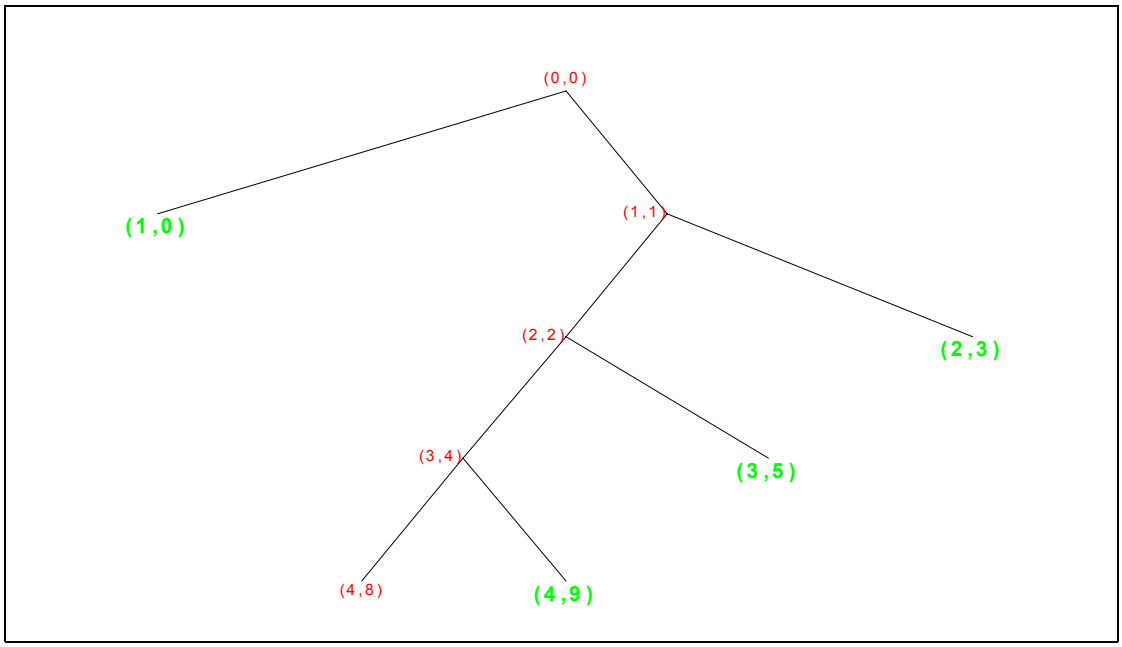

Figure C-16: BDBBA Basis for B_{budget}= 93.75% (Stairs1BMode15)

Appendix D
Psychoacoustic Weighting: Test Results

The addition of psychoacoustically based weighting to the cost calculation has a noticeable effect on the basis chosen in many cases. The particular effect it has depends on both the sample rate of the signal and its own spectral characteristics relative to that of the weighting curve.

In the case of *Doors* (32,000 Hz sample rate), the addition of weighting results in favouring a contiguous block of frequencies at the low end of the spectrum. Test case *DoorsPsych1* corresponds to no weighting with a budget of 43.75% and is shown in Fig.*D-1*. The resulting nodes are (2,0), (4,4), (4,6), and (4,12), which corresponds to spectrum from 0 kHz to 5 kHz, followed by a gap from 5 kHz to 7 kHz, and then a final subband from 7 kHz to 9 kHz. Test case *DoorsPsych2* uses the same budget and transform parameters, but with weighting applied. The resulting tree is shown in Fig.*D-2*. The retained nodes (2,0), (3,3), and (4,5) correspond to a contiguous chunk of spectrum from 0 kHz to 7 kHz (refer to *Table 3.1* for a mapping of node number to frequency range). Referring to the weighting characteristic of Fig.*4-9*, one can see that frequencies beyond 7 kHz are considered significantly less important than frequencies lower than this but above roughly 500 Hz.

Another example is given by the test cases *Stairs1BMode11* and *Stairs1Psych1*. Here we find that weighting causes signal in the 1.75 - 2 kHz range to be replaced with signal in the 2.75 - 3 kHz range. Referring again to Fig.*4-9*, we see that spectrum from roughly 2 - 5 kHz is favoured significantly over spectrum above and below that, explaining why the bases turned out different.

A third example is provided by *NoiseChirpBMode1* versus *NoiseChirpPsych1*, which involve the use of *Node Mode* with a *95% Energy Bandwidth* budget and a node budget of 4. Weighting in this case eliminates the lowest subband since the sample rate is low enough (8 kHz) that this subband occupies 0-250 Hz, which

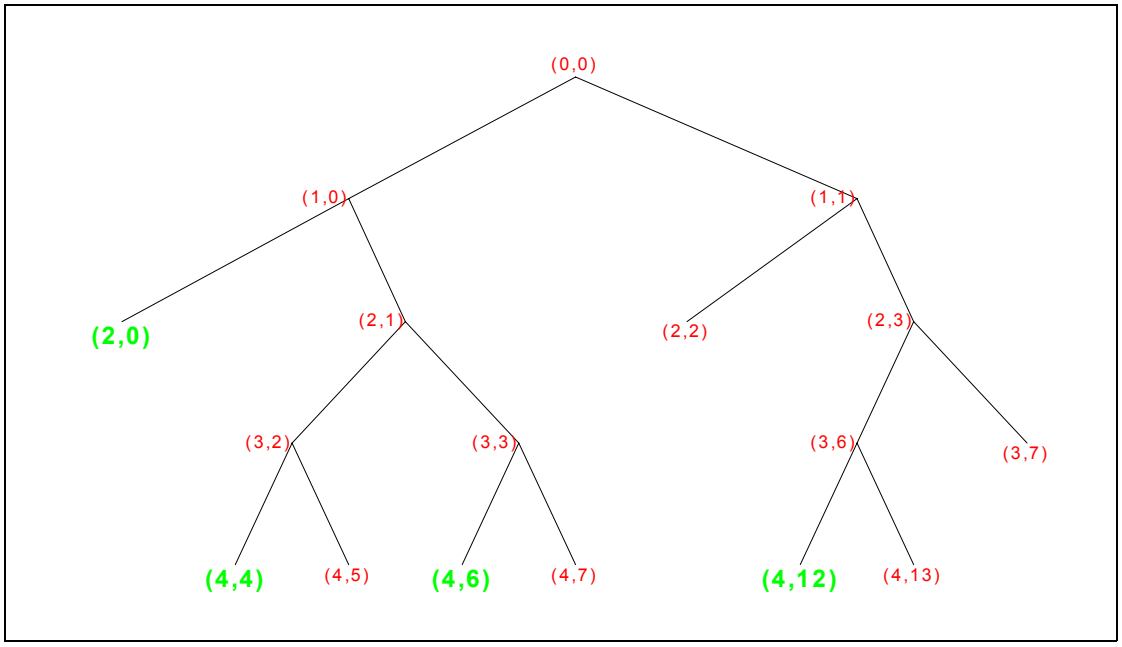

Figure D-1: BDBBA Basis for B_{budget}= 43.75%, No Weighting (DoorsPsych1)

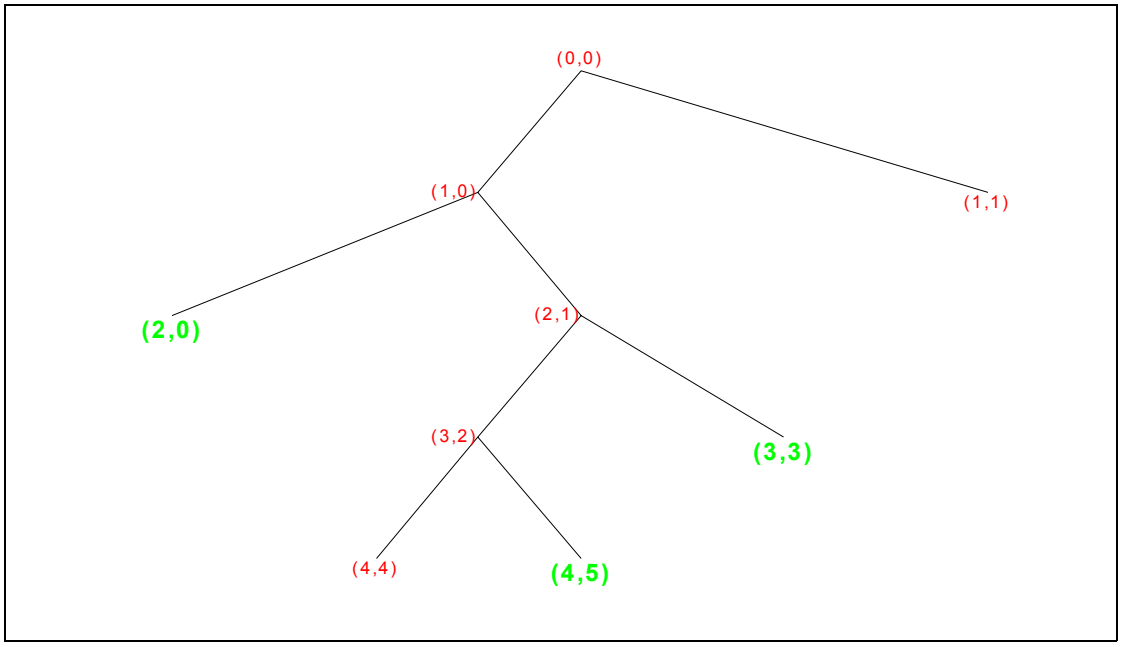

Figure D-2: BDBBA Basis for B_{budget}= 43.75%, Weighting (DoorsPsych2)

the weighting curves weights quite low. Without weighting, node (4,5) is eliminated instead.

Table D.1 Summary of Psychoacoustic Weighting Test Cases

Sound Name	Original Sound / Original Basis	Basis Selection Parameters / New Basis
DoorsPsych1	*Doors / WPT1*	*-Mode = BW -BW Budget = 43.75% / (2,0),(4,4),(4,6),(4,12)*
DoorsPsych2	*Doors / WPT1*	*-Mode = BW -Weighting -BW Budget = 43.75% / (2,0),(4,5), (3,3)*
Stairs1BMode11	*- see Table 5.12.*	
Stairs1Psych1		*-Mode = BW -BW Budget = 68.75% -Weighting / (2,0),(4,5), (3,3), (2,3)*
NoiseChirpBMode1	*- see Table 5.13.*	
NoiseChirpPsych1		*-Mode = BW -BW Budget = 93.75% -Weighting / (4,1),(3,1),(2,1),(1,1)*

Appendix E
Test Sound Cross Reference

Here we provide a master cross reference table for the test sounds discussed in the main body of this thesis. *Table E.1* links a given test sound name to the section it is discussed in, as well as the summary table and corresponding page that it appears in within that section.

Table E.1 Test Sound Cross Reference

Sound Name	Section	Table and Page Number
Bird	*Section 5.3*	Table 5.2 (p. 148)
BirdCross1	*Section 5.4.2.2*	Table 5.8 (p. 165)
BirdEQ1	*Section 5.4.3.1*	Table 5.9 (p. 167)
BirdNMode1 - 3	*Section 5.5.2.2*	Table 5.11 (p. 180)
Can	*Section 5.3*	Table 5.2 (p. 148)
CanThesh1 - 3	*Section 5.4.1.2*	Table 5.4 (p. 157)
CanTSC1 - 8	*Section 5.4.1.1*	Table 5.3 (p. 155)
Diesel	*Section 5.3*	Table 5.2 (p. 148)
DieselThresh1	*Section 5.4.1.2*	Table 5.4 (p. 157)
Doors	*Section 5.3*	Table 5.2 (p. 148)
DoorsCross1 - 2	*Section 5.4.2.2*	Table 5.8 (p. 165)
DoorsFilt1 - 2	*Section 5.4.1.3*	Table 5.5 (p. 159)
DoorsNMode1 - 2	*Section 5.5.2.2*	Table 5.11 (p. 180)
DoorsPsych1 - 2	*Appendix D*	Table D.1 (p. 229)
Dragging	*Section 5.3*	Table 5.2 (p. 148)
DraggingBMode1 - 2	*Section 5.5.2.3*	Table 5.13 (p. 187)
DraggingCross1 - 2	*Section 5.4.2.2*	Table 5.8 (p. 165)
DraggingFilt1 - 2	*Section 5.4.1.3*	Table 5.5 (p. 159)
DraggingThresh1	*Section 5.4.1.2*	Table 5.4 (p. 157)
Fly	*Section 5.3*	Table 5.2 (p. 148)
FlyCross1 - 3	*Section 5.4.2.2*	Table 5.8 (p. 165)
FlyEQ1	*Section 5.4.3.1*	Table 5.9 (p. 167)
FlyNFR1 - 4	*Section 5.4.2.1*	Table 5.7 (p. 163)

Table E.1 Test Sound Cross Reference

Sound Name	Section	Table and Page Number
FlyNMode1 - 2	*Section 5.5.2.2*	Table 5.11 (p. 180)
FlyQA1 - 2	*Section 5.5.2.4*	Table 5.14 (p. 190)
FlyTSC1	*Section 5.4.1.1*	Table 5.3 (p. 155)
Foghorn	*Section 5.3*	Table 5.2 (p. 148)
FoghornBMode1	*Section 5.5.2.3*	Table 5.13 (p. 187)
FoghornMod1 - 4	*Section 5.4.3.2*	Table 5.10 (p. 168)
FoghornTSC1	*Section 5.4.1.1*	Table 5.3 (p. 155)
Gamelan	*Section 5.3*	Table 5.2 (p. 148)
Gran1 - 23	*Section 5.4.1.4*	Table 5.6 (p. 161)
Knock	*Section 5.3*	Table 5.2 (p. 148)
KnockEQ1 - 2	*Section 5.4.3.1*	Table 5.9 (p. 167)
KnockNFR1 - 2	*Section 5.4.2.1*	Table 5.7 (p. 163)
KnockQA1	*Section 5.5.2.4*	Table 5.14 (p. 190)
Lake	*Section 5.3*	Table 5.2 (p. 148)
LakeBMode1	*Section 5.5.2.3*	Table 5.13 (p. 187)
LakeEQ1 - 2	*Section 5.4.3.1*	Table 5.9 (p. 167)
LakeTSC1 - 3	*Section 5.4.1.1*	Table 5.3 (p. 155)
Noise	*Section 5.3*	Table 5.2 (p. 148)
NoiseChirp	*Section 5.3*	Table 5.2 (p. 148)
NoiseChirpBMode1	*Section 5.5.2.3*	Table 5.13 (p. 187)
NoiseChirpPsych1	*Appendix D*	Table D.1 (p. 229)
Rain	*Section 5.3*	Table 5.2 (p. 148)
RainFilt1	*Section 5.4.1.3*	Table 5.5 (p. 159)
Scale	*Section 5.3*	Table 5.2 (p. 148)
ScaleFilt1 - 2	*Section 5.4.1.3*	Table 5.5 (p. 159)
ScaleThresh1 - 2	*Section 5.4.1.2*	Table 5.4 (p. 157)
Shotgun	*Section 5.3*	Table 5.2 (p. 148)
Stairs1	*Section 5.3*	Table 5.2 (p. 148)
Stairs1BMode1 - 15	*Section 5.5.2.3*	Table 5.12 (p. 183)
Stairs1CT1	*Section 5.5.2.2*	Table 5.11 (p. 180)

Table E.1 Test Sound Cross Reference

Sound Name	Section	Table and Page Number
Stairs1NMode1 - 2	*Section 5.5.2.2*	Table 5.11 (p. 180)
Stairs1Psych1	*Appendix D*	Table D.1 (p. 229)
Stairs2	*Section 5.3*	Table 5.2 (p. 148)
Stairs2EQ1	*Section 5.4.3.1*	Table 5.9 (p. 167)
Stairs2QA1 - 2	*Section 5.5.2.4*	Table 5.14 (p. 190)
Sweep	*Section 5.3*	Table 5.2 (p. 148)
Train	*Section 5.3*	Table 5.2 (p. 148)
Voice	*Section 5.3*	Table 5.2 (p. 148)
440Hz	*Section 5.3*	Table 5.2 (p. 148)