AUDIO REPRESENTATIONS FOR DATA COMPRESSION AND COMPRESSED DOMAIN PROCESSING

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Scott Nathan Levine

December 1998

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Julius Orion Smith, III
(Principal Advisor)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Martin Vetterli

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Nick McKeown

Approved for the University Committee on Graduate Studies:

AUDIO REPRESENTATIONS FOR DATA COMPRESSION AND COMPRESSED DOMAIN PROCESSING

Scott Nathan Levine
Stanford University, 1999

In the world of digital audio processing, one usually has the choice of performing modifications on the raw audio signal or performing data compression on the audio signal. But, performing modifications on a data compressed audio signal has proved difficult in the past. This thesis provides new representations of audio signals that allow for both very low bit rate audio data compression and high quality compressed domain processing and modifications. In this system, two compressed domain processing algorithms are available: time-scale and pitch-scale modifications. Time-scale modifications alter the playback speed of audio without changing the pitch. Similarly, pitch-scale modifications alter the pitch of the audio without changing the playback speed.

The algorithms presented in this thesis segment the input audio signal into separate sinusoidal, transients, and noise signals. During attack-transient regions of the audio signal, the audio is modeled by transform coding techniques. During the remaining non-transient regions, the audio is modeled by a mixture of multiresolution sinusoidal modeling and noise modeling. Careful phase matching techniques at the time boundaries between the sines and transients allow for seamless transitions between the two representations. By separating the audio into three individual representations, each can be efficiently and perceptually quantized. In addition, by segmenting the audio into transient and non-transient regions, high quality time-scale modifications that stretch only the non-transient portions are possible.

Acknowledgements

First I would like to thank my principal advisor, Prof. Julius O. Smith III. In addition to being a seemingly all-knowing audio guy, our weekly meetings during my last year in school helped me out immensely by keeping me and my research focused and on track. If it were not for the academic freedom he gives me and the other CCRMA grad students, I would not have stumbled across this thesis topic. My next thanks goes out to Tony Verma, with whom I worked closely with for almost a year on sinusoidal modeling research. Through our long meetings, academic and otherwise, he was a great sounding board for all things audio-related. A CCRMA thank you list would not be complete without mentioning the other folks in our DSP group, including Bill Putnam, Tim Stilson, Stefan Bilbao, David Berners, Yoon Kim, and Mike Goodwin (via Berkeley). There is hardly an audio or computer issue that one of these folks could not answer.

From my time as an undergraduate at Columbia University, I would like to thank two professors. First, I would like to thank Prof. Brad Garton who let me, as a Freshman, into his graduate computer music courses. His classes introduced me to the world of digital audio, and laid the foundation for a long road of future signal processing education ahead. Secondly, I would like to thank Prof. Martin Vetterli, who took me into his image/video processing lab during my Junior and Senior years. It was during this time I first got a taste of the research lifestyle, and set me on my path towards getting my Ph.D. I would also like to thank his graduate students at the time who taught me my first lessons of wavelets, subband filtering, and data compression: Jelena Kovačević, Cormac Herley, Kannan Ramchandran, Antonio Ortega, and Alexandros Eleftheriadis.

In chronological order, I would next like to thank all the folks I worked with during my short stints in industry: Raja Rajasekaran, Vishu Viswanathan, Joe Picone (Texas Instruments); Mark Dolson, Brian Link, Dana Massie, Alan Peevers (E-mu); Louis Fielder, Grant Davidson, Mark Davis, Matt Fellers, Marina Bosi (Dolby Laboratories); Martin Dietz, Uwe Gbur, Jürgen Herre, Karlheinz Brandenburg (FhG); Earl Levine, Phil Wiser (Liquid Audio). By working with so many great companies and great people, I was able to learn some of the real tricks of the audio trade.

On a more personal note, I would like to thank Maya for keeping me sane, balanced, and motivated throughout the last two years, with all of its ups and downs. And last, but certainly not least, I would like to thank my family for always being there for me.

Contents

List of Figures

Chapter 1

Audio Representations

Reproducing audio has come a long way since the encoding of analog waveforms on wax cylinders. The advent of digital audio has enabled a large jump in quality for the end user. No longer does playback quality deteriorate over time as in the case of vinyl records or analog magnetic cassettes. While the quality of the compact disc (CD) is sufficiently high for most consumers, the audio data rate of 1.4 Mbps for stereo music is too high for network delivery over most consumers' home computer modems. In 1998, most people can stream audio at either 20 or 32 kbps, depending on the quality of their modem. These compression rates of 70:1 and 44:1, respectively, point towards the need for sophisticated data compression algorithms.

Most current audio systems employ some form of transform coding which will be introduced in Section 1.1.3. While transform coding allows for high quality data compression, it is not a malleable representation for audio. If the user desires to perform any modifications on the audio, such as change the playback speed without changing the pitch, or vice-versa, a significant amount of post-processing is required. These post-processing algorithms, some of which will be discussed in Section 1.2, require a significant amount of complexity, latency, and memory.

The goal of this thesis is to create an audio representation that allows for high-quality data compression while allowing for modifications to be easily performed on the compressed data itself. The new decoder can not only inverse quantize the data compressed audio, but can cheaply perform modifications at the same time. While the encoder will have a slightly higher complexity requirements, the decoder will be of the same order of complexity as current transform coders.

One could surmise that a greater percentage of future audio distribution will be over data networks of some kind, instead of simply distributing audio on high-density storage media, like the current CD or DVD. While data rates over these networks will undoubtedly increase over the years, bandwidth will never be *free*. With better data compression, more audio channels can be squeezed over the channel, or more video (or any other multimedia data) can be synchronously transmitted. Current music servers already have a considerable investment in mastering their audio libraries in

a compressed audio format; a small incremental layer of complexity would allow end users to not simply play back the audio, but perform modifications as well.

This chapter is divided into three sections. In Section 1.1, a short history is presented of various audio representations for data compression. Following in Section 1.2 is an abridged list of audio representations used primarily for musical modifications. In the final part of the chapter, Section 1.3, previous methods that allow both compression and some amount of compressed-domain modifications is presented. The scope of this last section will include not only audio, but speech and video as well.

To preface the following sections, we assume that the original analog audio input source is bandlimited to 22 kHz, the maximum absolute amplitude is limited, and then sampled at 44.1 kHz with a precision of 16 bits/sample (the CD audio format). This discrete-time, discrete-amplitude audio signal will be considered the reference signal. While new audio standards now being discussed, such as DVD-Audio and Super Audio CD, have sampling rates as high as 96 kHz or even 192 kHz, and bit resolutions of 24 bits/sample, the CD reference specifications are sufficient for the scope of this thesis.

1.1 Audio Representations for Data Compression

This section will deliver a brief history of lossy digital audio data compression. Digital audio (not computer music) can be argued to have begun during 1972-3 when the BBC began using 13 bit/sample PCM at 32 kHz sampling rate for its sound distribution for radio and television, and Nippon Columbia began to digitally master its recordings (Nebeker, 1998; Immink, 1998). In its relative short history of approximately 25 years of digital audio, researchers have moved from relatively simple scalar quantization to very sophisticated transform coding techniques (Bosi *et al.*, 1997). All of the methods to be mentioned in this section *cannot* be easily time-scale or pitch-scale modified without using some of the post-processing techniques later discussed in Section 1.2. However, they performed their designated functions of data compression and simple file playback well for their respective times.

In the following sections on quantization, it is helpful to think of lossy quantization as an additive noise process*. Let x_n be the input signal, and $Q(x_n)$ be the quantized version of the input, then the quantization error is $\epsilon_n = Q(x_n) - x_n$. With some rearrangement, the equation becomes:

$$Q(x_n) = x_n + \epsilon_n \tag{1.1}$$

which can be seen in Figure 1.1.

The objective in any perceptual coding algorithm is to *shape* this quantization noise process in both time and frequency. If this shaping is performed correctly, it is possible for the quantized

*Natural audio signals are not memoryless, therefore the quantization noise will not be precisely white. However, this assumption is close enough for the purposes of this discussion.

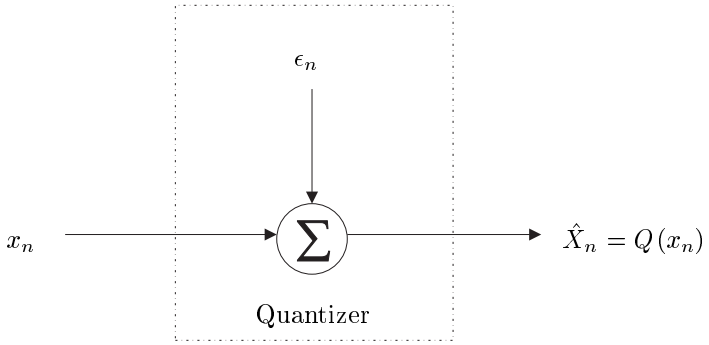

Fig. 1.1. Additive noise model of a quantizer

signal $Q(x_n)$ to *mask* the quantization noise ϵ_n, thus making the noise inaudible. These concepts of masking and noise shaping will first be discussed in Section 1.1.3.

1.1.1 Scalar Quantization

Perhaps the simplest method to represent an audio signal is that of scalar quantization. This method lives purely in the time domain: each individual time sample's amplitude is quantized to the nearest interval of amplitudes. Rather than transmitting the original amplitude every time sample, the codebook index of the amplitude range is sent.

Uniform Scalar Quantization

Uniform scalar quantization divides the total signal range into N uniform segments. As a graphical example, see Figure 1.2. With every added bit, r, of resolution for uniform scalar quantization ($r = log_2 N$), the signal-to-noise (SNR) ratio increases approximately 6 dB. For a more detailed and mathematical treatment of scalar quantization, see (Gersho and Gray, 1992). Although scalar quantization produces low MSE, perceptual audio quality is not necessarily correlated with low mean-square error (MSE). Perhaps more important than MSE is the spectral shape of the noise. In all forms of scalar quantization, the quantization noise is mostly white, and thus spectrally flat. Figure 1.3 gives a graphical example of uniform quantization, using six bits of resolution. The magnitude spectrum on the left is from a short segment of pop music. The approximately flat spectrum on the right is the quantization error resulting from 6 bit uniform quantization, with the maximum value of the quantizer set to the maximum value of the audio input. At lower frequencies, below 3000 Hz, the noise is much quieter than the original and will thus be inaudible. But notice how at high frequencies, the quantization noise is louder than the quantized signal. This high frequency

quantization error will be very audible as *hiss*. Later in Section 1.1.3, it will be shown that the noise can be shaped underneath the spectrum of the original signal. The minimum dB distance between the original spectrum and the quantization noise spectrum, such that the noise will be inaudible, is termed the frequency-dependent signal-to-masking ratio (SMR).

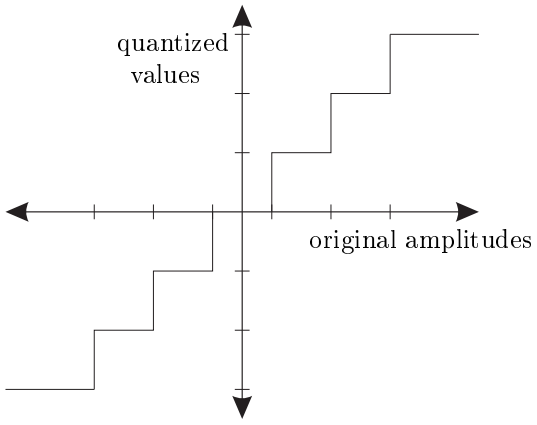

Fig. 1.2. An example of a uniform scalar quantizer

One benefit of scalar quantization is that it requires little complexity. It is the method used for all music compact discs (CD) today, with $N = 65,536$ (2^{16}) levels. With $r = 16$, the SNR is approximately 96 dB. Even though the noise spectrum is flat, it is below the dynamic range of almost all kinds of music and audio recording equipment, and is therefore almost entirely inaudible. In 1983, when Sony and Philips introduced the CD, decoding complexity was a major design issue. It was not possible to have low-cost hardware at the time that could perform more complex audio decoding. In addition, the CD medium could hold 72 minutes of stereo audio using just uniform scalar quantization. Since 72 minutes was longer than the playing time of the analog cassettes or vinyl records they were attempting to replace, further data compression was not a priority.

Nonuniform Scalar Quantization

Better results can be obtained with scalar quantization if one uses a *nonuniform* quantizer. That is, not all quantization levels are of equal width. The probability density function (pdf) of much audio can be approximated roughly by a Laplacian distribution (not a uniform distribution). Therefore, one would want to match the quantization levels to roughly the pdf of the input signals. This is performed by first warping, or *compressing* the amplitudes such that large amplitude values are compressed in range, but the smaller values are expanded in range. Then, the warped amplitudes are quantized on a linear scale. In the decoder, the quantized values are inverse warped, or *expanded*. The current North American standard for digital telephony, G.711, uses an 8 bit/sample, piecewise

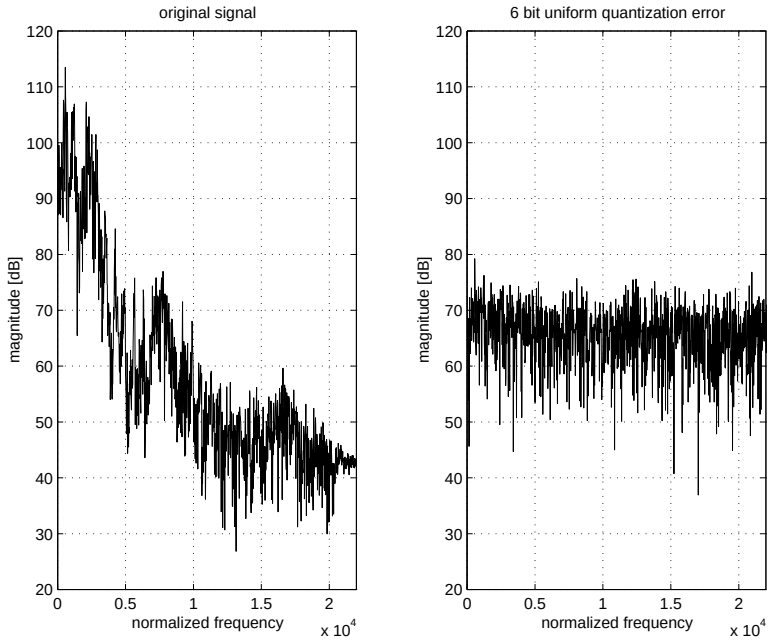

Fig. 1.3. Example of the quantization error resulting from six bit uniform scalar quantization

linear approximation to the nonuniform $\mu-$law compressor characteristic (Gersho and Gray, 1992):

$$G_\mu(x) = V\frac{\ln(1+\mu|x|/V)}{\ln(1+\mu)}\text{sgn}(x), \quad |x| \leq V$$

Even though the SNR is better using nonuniform scalar quantization than uniform scalar quantization, the quantization noise is still relatively flat as a function of frequency. While this might be tolerable in speech telephony, it is not *perceptually lossless* for wideband audio at 8 bits/sample and is thus unacceptable for that application. The notion of perceptual losslessness is a subjective measurement whereby a group of listeners deem the quantized version of the audio to be indistinguishable from the original reference recording (Soulodre *et al.*, 1998).

Predictive Scalar Quantization

Another form of scalar quantization that is used as part of a joint speech/audio compression algorithm for videoconferencing is Adaptive Delta Pulse Code Modulation (ADPCM) (Cumminskey, 1973). In this algorithm, the original input signal itself is *not* scalar quantized. Instead, a difference signal between the original and a predicted version of the input signal is quantized. This predictor can have adaptive poles and/or zeroes, and performs the coefficient adaptation on the previously quantized samples. In this manner, the filter coefficients do not need to be transmitted from the encoder to the decoder; the same predictor sees the identical quantized signal on both sides.

The CCITT standard (G.722) for 7 kHz speech+audio at 64 kbps uses a two-band subband ADPCM coder (Mermelstein, 1988; Maitre, 1988). The signal is first split into two critically sampled

subbands using quadrature mirror filter banks (QMF), and ADPCM is performed independently on each channel. The low frequency channel is statically allocated 48 kbps, while the high frequency channel is statically allocated 16 kbps.

Differences between Wideband Speech and Audio Quantizers

Because G.722 is a *waveform* coder, and not a *speech source model* coder, it performs satisfactorily for both speech and audio. But if the system had speech inputs only, it could perform much better using a speech source model coder like one of the many flavors of code excited linear prediction (CELP) (Spanias, 1994). If the input were just music, one of the more recent transform coders (Bosi *et al.*, 1997) would perform much better. Being able to handle music and speech using the same compression algorithm at competitive bitrates and qualities is still a difficult and an open research problem. There are some audio transform coders that have enhancements to improve speech quality (Herre and Johnston, 1996) in MPEG-AAC. Then, there are some audio coders that use linear prediction coding (LPC) techniques from the speech world in order to bridge the gap (Moriya *et al.*, 1997; Singhal, 1990; Lin and Steele, 1993; Boland and Deriche, 1995). There are also speech codecs that use perceptual/transform coding techniques from the wideband audio world (Tang *et al.*, 1997; Crossman, 1993; Zelinski and Noll, 1977; Carnero and Drygajlo, 1997; Chen, 1997). Most commercial systems today, such as RealAudio^{TM} and MPEG-4 (Edler, 1996), ask the user if the source is music or speech, and accordingly use a compression algorithm tailored for that particular input.

1.1.2 Transform Coding

Transform coding was the first successful method to encode perceptually lossless wideband audio at low bit rates, which today is at 64 kbps/ch (Soulodre *et al.*, 1998). What sets transform coding apart from previous methods of compression is its ability to *shape* its quantization noise in time and frequency according to psychoacoustic principles. In the previous Section 1.1.1, the scalar quantization methods minimized MSE, but left a flat quantization noise floor which could be audible at certain times and frequency regions. By using a *psychoacoustic model*, the compression algorithms can estimate at what time and over what frequency range human ears cannot hear quantization noise due to masking effects. In this manner, the transform coding can move the quantization noise to these inaudible regions, and thus distortion-free audio is perceived.

For a simplified diagram of most transform coders, see Figure 1.4. Every transform coding system has at least these three building blocks. At the highest level, the *filter bank* segments the input audio signal into separate time-frequency regions. The *psychoacoustic modeling* block determines where quantization noise can be injected without being heard because it is being masked by the original signal. Finally, the *quantization* block quantizes the time-frequency information according to information provided by the psychoacoustic model and outputs a compressed bitstream. At the decoder, the bitstream is inverse quantized, processed through an inverse filter bank, and the audio

reconstruction is complete. Each of these three encoder building blocks will be described in more detail in the following three subsections. These subsections will describe these building blocks on a basic, high level. For more detailed information on the methods used in commercial transform coding, see Section 4.1.

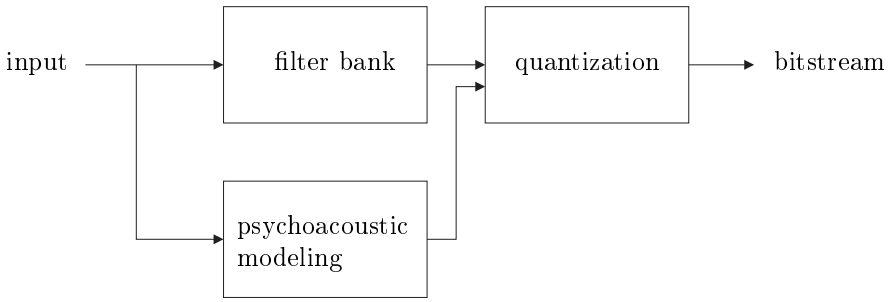

Fig. 1.4. A basic transform encoding system

Filter Banks

Most current audio compression algorithms use some variant of Modified Discrete Cosine Transforms (MDCT). Credit for this filter bank is often given to Princen and Bradley (1986), where it is referred to as Time Domain Aliasing Cancellation (TDAC) filter banks. The oddly-stacked[†] TDAC was later recognized as specific case of the more general class of Modulated Lapped Transforms (MLT) (Malvar, 1990).

The beauty of the MDCT is that it is a critically sampled *and* overlapping transform. A transform that is critically sampled has an equivalent number of transform-domain coefficients and time-domain samples. An example of an older compression algorithm that was *not* critically sampled used FFTs with overlapping Hanning windows by 6.25% ($\frac{1}{16}^{th}$) (Johnston, 1988*b*). That is, there were 6.25% more transform-domain coefficients than time-domain samples, thus making the overlapping FFT an *oversampled* transform. In order for an FFT to be critically sampled and have perfect reconstruction, the window would have to be rectangular and not overlap at all. Because of quantization blocking artifacts at frame boundaries, it is desirable to have overlapping, smooth windows. When using the MDCT, there are the same number of transform-domain coefficients as time-domain samples. Therefore, there are fewer transform-domain coefficients to quantize (than with the overlapped FFT),

[†]The subbands of an oddly-stacked filter bank all have equal bandwidth, while the first and last subbands of an evenly-stacked filter bank have half of the bandwidth of the others.

and the bit rate is lower. Other examples of earlier audio compression algorithms that used filter banks other than MDCTs were the Multiple Adaptive Spectral Audio Coding system (Schroder and Vossing, 1986) that used DFTs and the Optimum Coding in the Frequency Domain (OCF) that used DCTs (Brandenburg, 1987).

The MDCT used today has a window length equal to twice the number of subbands. For example, if the window length L=2048, then only 1024 MDCT coefficients are generated every frame and the window is hopped 1024 samples for the next frame (50% overlap). There is significant aliasing in the subband/MDCT data because the transform is critically sampled. However, the aliased energy is completely canceled in the inverse MDCT filter bank. The MDCT can be thought of as a bank of M bandpass filters having impulse responses: (Malvar, 1992):

$$f_k(n) = h(n)\sqrt{\frac{2}{M}}\cos\left[\left(n + \frac{M+1}{2}\right)\left(k + \frac{1}{2}\right)\frac{\pi}{M}\right] \tag{1.2}$$

for $k = 0, 1, \ldots, M-1$, and $n = 0, 1, \ldots, L-1$, where $L = 2M$. All M filters are cosine modulations of varying frequencies and phases of the prototype window/lowpass filter, $h(n)$. Notice that since the modulation is real-valued, the MDCT coefficients will also be real-valued. A simple prototype window that is often used for the MDCT in MPEG AAC is the raised sine window:

$$h(n) = \sin\left[\frac{\pi}{N}\left(n + \frac{1}{2}\right)\right], \qquad 0 \le n < N \tag{1.3}$$

Because of this cosine modulation in the filter bank, fast algorithms are available that can compute the transform using FFTs or Discrete Cosine Transforms (DCT), with order $O(N\log_2 N)$ complexity (Malvar, 1991).

It was shown that it is necessary for both of the following conditions of the prototype lowpass filter to be true to allow perfect reconstruction (Princen and Bradley, 1986):

$$h(L-1-n) = h(n) \tag{1.4}$$

which forces even symmetry, and

$$h^2(n) + h^2(n+M) = 1 \tag{1.5}$$

which regulates the shape of the window. The raised sine window of Equation (1.3) also satisfies these properties. The length, L, of the window can be adaptively switched according to the signal. Because of quantization properties discussed in the next few subsections, it is desirable to have short windows around short-time transient-like signals. When switching between window lengths, perfect reconstruction can still be achieved by assuring that the *tails* of the overlapping windows still have energy summing to one (T. Sporer, 1992):

$$h_{t-1}^2(n+M) + h_t^2(n) = 1 \tag{1.6}$$

where t is the frame number. As a graphical example in Figure 1.5, length 2048 windows are switched to length 256 windows, and back. The pictured window switching scheme is used currently in MPEG-AAC (Bosi *et al.*, 1997) and in PAC (Johnston *et al.*, 1996*b*).

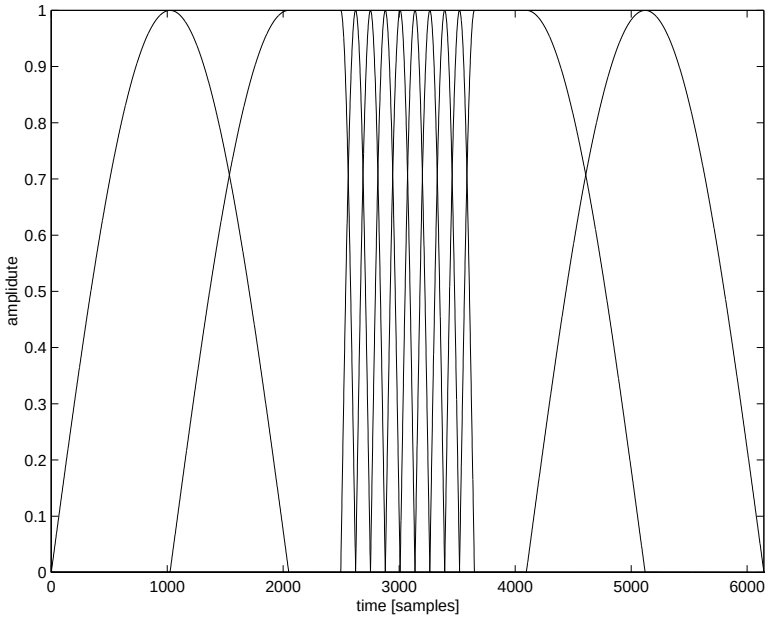

Fig. 1.5. Window switching using the MDCT

Psychoacoustic Modeling

As is shown in Figure 1.1, quantization can be thought of as adding white noise to the original signal. Because the transform-coding quantization is being performed independently over small non-uniform frequency bands, the quantization noise levels are bounded in time (by the filter bank frame length) and in frequency. The purpose of the psychoacoustic model is to determine how much noise can be added to each time/frequency region before the quantization noise becomes audible. The reason the noise would *not* be audible is because it is being *masked* by the original signal. Simply put, if two signals reside in the same frequency region, and there is an energy difference between the two that is larger than a certain threshold, the softer signal will be inaudible. That is, a human listener would never have known the softer signal was present.

To explain how a psychoacoustic model derives these just-noticeable noise levels, the specific MPEG Psychoacoustic Model 2 (Brandenburg and Stoll, 1994) will be detailed. It is the suggested encoder psychoacoustic model to be used in MPEG-II Layer 3, and MPEG-AAC. The model flow diagram is shown in Figure 1.6. Many other systems have been developed over the years, (Johnston, 1988*a*; Wiese and Stoll, 1990; Fielder *et al.*, 1996) but it would be out of the scope of this thesis to detail them all. Another good reference for other psychoacoustic models, in addition to a detailed history of perceptual audio coding, is Painter (1997).

Psychoacoustic modeling can be computed separately in both the time and frequency domain. Time domain psychoacoustic modeling simply states that the masking effect of a loud sound extends

before (backward masking) and after (forward masking) the sound is actually playing. That is, about 5 milliseconds before, and anywhere between 50 to 200 milliseconds after a sound is played, some masking will occur. These durations depend on the energy and frequency of the sound being turned on, along with the energy and frequency of the softer sound it may be masking (Zwicker and Fastl, 1990). The effect of time domain masking allows compression algorithms to use high-frequency resolution filter banks, with somewhat limited temporal resolution.

Frequency domain psychoacoustic modeling is best described in the Bark domain (Zwicker and Fastl, 1990). One Bark is the difference in frequency between adjacent critical bands. Critical bands are defined as the maximum bandwidth within which the intensities of individual tones are summed by the ear to form auditory thresholds. Frequency domain masking occurs within a critical band and between critical bands. The masking that occurs *within* a critical band depends on the *tonality* of the signal in that band (Hellman, 1972). In the MPEG system (Brandenburg and Stoll, 1994), if a signal is determined to be tone-like, as a pure sinusoid would be, then quantization noise will be masked as long as it more than 18 dB below the signal itself. But, if a signal is determined to be pure noise in the critical band, then the quantization noise will be masked as long as it only 6 dB (or more) below the signal itself. If the signal is somewhere between pure noise or tone, then the masking threshold is accordingly somewhere between -6 and -18 dB. This is a simplification of measured experimental masking thresholds, which have masking thresholds that depend of amplitude and frequency of masker (Zwicker and Fastl, 1990). But, certain constraints on complexity and memory must be made in order to make a realizable compression system. In MPEG-AAC, a tonality measure is derived (on a one-third Bark band scale) from a second order linear predictor of the FFT magnitude and phases, among other factors.

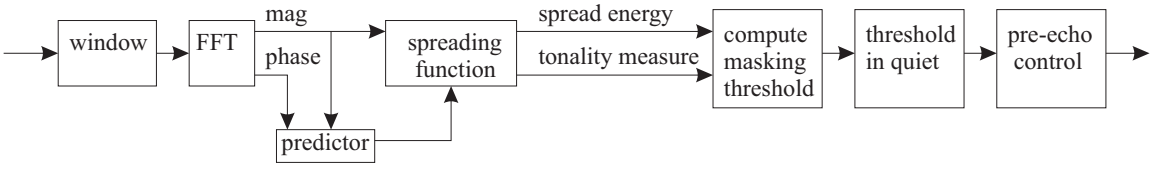

Fig. 1.6. Computing psychoacoustic masking thresholds, as performed in MPEG-AAC

In addition to masking *within* a critical band, there is masking *across* critical bands, which is modeled by the spreading function in Figure 1.6. This spreading function spreads the energy from a single Bark band to its neighboring Bark bands. It is a simplification of the cochlea in the human ear, which performs a similar spreading. The masking is not symmetric in frequency; a loud Bark band will mask higher frequencies more that lower frequencies. See Figure 1.7 for an example of the spread masking thresholds of three pure sinusoids at 500, 1500, and 3200 Hz. If the masking threshold is lower than the absolute threshold level of hearing, the masking threshold is clipped at this level. This effect can be seen above the 53^{rd} one-third Bark band in Figure 1.7.

In conclusion, the psychoacoustic model dictates how much quantization noise can be injected

in each region of time-frequency without being perceptible. If enough bits are available, the quantization noise will not be heard due to the properties of auditory masking. While this process is not exact, and is only an approximation to idealized sines and noise listening experiments, the masking thresholds seem to work quite well in practice.

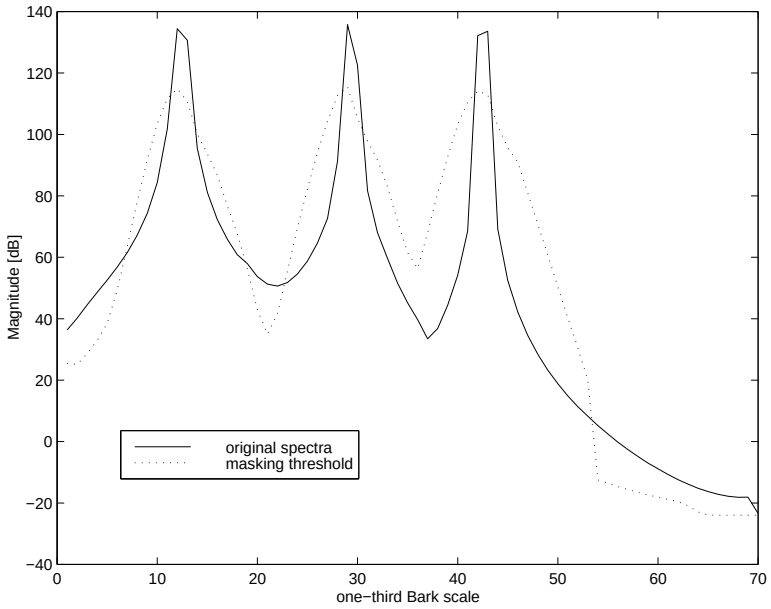

Fig. 1.7. This figure shows the masking thresholds of three synthesized sinusoids. The solid line is the original magnitude spectrum, and the dotted line is its masking threshold. The original spectrum was generated by first using a 2048 point, raised sine, windowed FFT. The FFT magnitude spectrum was then mapped to the one-third Bark scale by summing together bin energies in each one-third Bark band. The dB difference between the original spectra and the masking threshold dictates how much quantization noise can be injected at that particular frequency range. This dB difference is also termed the signal-to-mask ratio (SMR). For example, at one-third Bark number 30, the frequency range of the second sinusoidal peak, the original signal is 18 dB above its masking threshold. Therefore, any quantization noise must be at least 18 dB quieter than the original signal at that frequency to be *masked*, or inaudible. If the quantization noise is any louder, it will be detectable.

Quantization

Now that the signal has been segmented in time and frequency, and the masking thresholds have been computed, it is time to quantize the MDCT coefficients. There are many ways to quantize these coefficients, and several of the commercial methods will later be described in Section 4.1. A general trend among compression algorithms is to group MDCT coefficients into non-uniformly spaced bands. Each band is assigned its own quantization scalefactor, or *exponent*. These scalefactor bands (their name in the MPEG literature) lie approximately on the half-band Bark band scale. The quantization resolution in each scalefactor band is dependent on the masking thresholds in that

frequency region. Ideally, there would be enough bits to quantize each scalefactor band of MDCT coefficients such that its mean-square quantization error be just less than the amount of quantization noise allowed by the psychoacoustic model. If this condition is met, then no quantization error should be audible. If there are not enough bits available, approximations must be made. This scenario is usually the case for low bit rate audio coding ($\leq 128 kbps/ch$). Both MPEG Layer III (Brandenburg and Stoll, 1994) and AAC (Bosi *et al.*, 1997) have the same joint rate-distortion loops, that attempt to iterate towards a solution that satisfies both conditions to some degree. Since both rate and distortion specifications cannot be met simultaneously, it is the decision of the encoder to decide which tradeoffs to make.

Both systems also have a *bit reservoir* that allocates some frames fewer bits, while more difficult frames get allocated more bits. This system of sharing a pool of bits among frames increases the quality markedly, due to the variance of difficulty in actual frames of audio. One disadvantage of using a bit reservoir is that the system becomes a *variable* bit rate (VBR) system. For fixed bandwidth, low-latency, real-time applications, VBR may not be acceptable. In contrast, Dolby's AC-3 is not VBR, but rather is *constant* bit rate (CBR), having a fixed number of bits per frame. It was designed primarily for film and digital television broadcast, both of which are fixed bandwidth media. More about the differences among commercial audio compression systems will be discussed later in Section 4.1. The number of bits allowed to be taken out or placed into the bit reservoir pool is a function of some *perceptual entropy* metric (Johnston, 1988*b*). The perceptual entropy (PE) is a function of the dynamic range between the input signal's energy envelope and the envelope of the quantization noise allowable. In MPEG-AAC and Layer III, the PE also determines when to switch between short and long windows, as illustrated in the previous section on filter banks.

1.1.3 Wavelet Coding

In the last five years, there has been a large number of papers written about wavelet audio coding. The quantization and psychoacoustic modeling techniques are similar to those used in transform coding, but the filter bank is completely different. The filter bank for wavelet coding is generally the tree structured wavelet transform, or wavelet packets (Coifman *et al.*, 1992). The filter bank is generally designed to have nonuniform segmentation of the frequency axis to approximately match the Bark bands of the human ear (Zwicker and Fastl, 1990). These structures are iterated filter banks, where each iterated section is a two-band filter bank, as shown in Figure 1.8. The first such filter bank in the audio wavelet coding literature was presented by Sinha and Tewfik (1993), which uses 29 bands and 8 levels of depth. This structure can be seen in Figure 1.9.

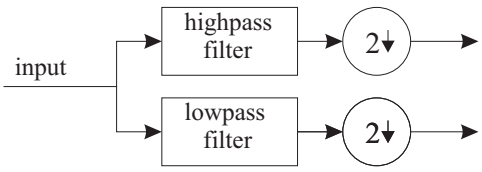

Fig. 1.8. A two channel section of the wavelet packet tree in Figure 1.9

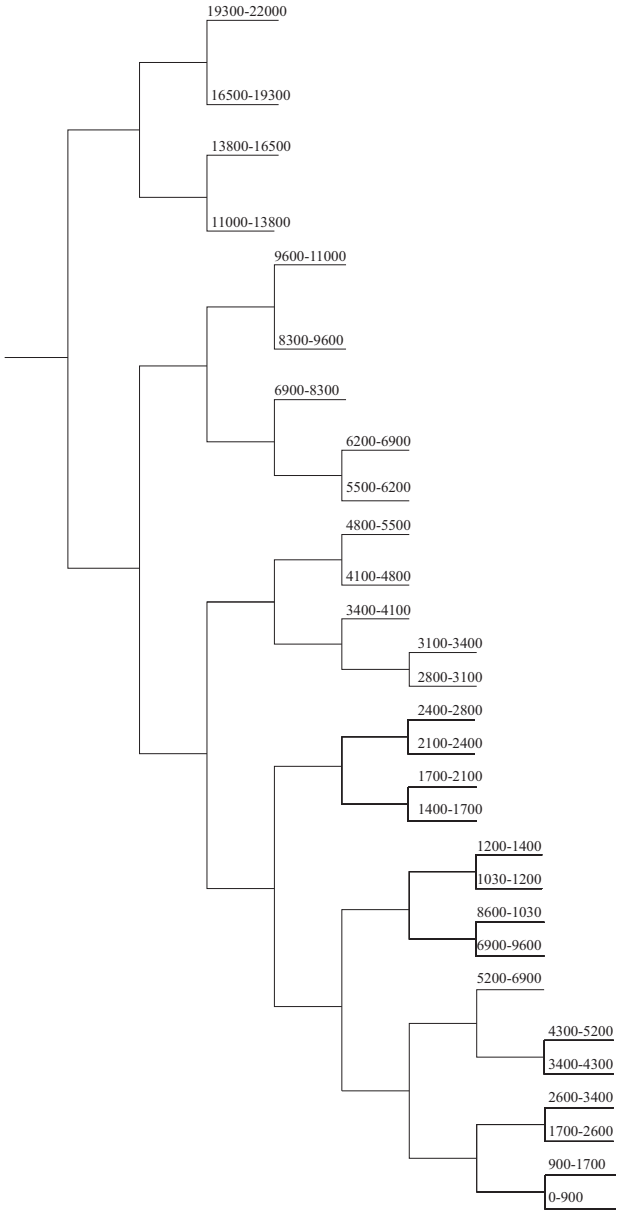

Fig. 1.9. The wavelet packet tree used by Sinha and Tewfik (1993)

In addition to the frequency axis having a nonuniform segmentation, the time axis is also similarly nonuniform. At low frequencies, the frequency resolution is good, but the time resolution is very poor since the windows are so long. At high frequencies, the frequency resolution is poor, but the time resolution is very good due to short windows. Several papers have been written discussing optimal choice of wavelet filters to use (Sinha and Tewfik, 1993; Phillipe, 1995; Kudumakis and Sandler, 1995). Other wavelet-only audio coders made improvements in pre-echo and quantization (Tewfik and Ali, 1993), bitrate scalability (Dobson *et al.*, 1997), and using both scalar and vector quantization techniques (Boland and Deriche, 1996).

Hybrid Wavelet Coders

While wavelet coders seem to represent transients and attacks well, they do not efficiently represent steady-state signals, like sinusoidal tones. To remedy this problem, Hamdy *et al.* (1996) first extracts most almost of the sinusoids in the signal using sinusoidal modeling, which will be later discussed in Section 1.2.3. Then, a wavelet coder (Sinha and Tewfik, 1993) quantizes most of the residual signal deemed *not sinusoidal*. Above 11 kHz, the wavelet information is efficiently encoded simply as edges and noise processes. The following year, a similar approach using sines+wavelets+noise was presented by S. Boland (1997). Another interesting hybrid coder uses MDCT transform coders with good frequency resolution and poor temporal resolution for all steady-state signals (Sinha and Johnston, 1996). But, at attack transients, the coder is switched to a wavelet packet scheme similar to that shown in Figure 1.9, but only using four bands. Special considerations are taken to ensure orthogonality in the transition regions between the two types of filter banks.

Comparisons to Transform Coding

The Bark band filter bank used by the wavelet coders is approximated in the transform coding world by first making a high (frequency) resolution transform, and then grouping together MDCT coefficients into Bark bands (called scalefactor bands in the MPEG world). These scalefactor bands (sfbs) are made up of 4 to 32 MDCT coefficients, depending on frequency range, and have their own quantization resolution determined by its range's SMR. Although the temporal resolution of transform coding is uniform across all frequencies, transform coders switch to fine temporal (and poor frequency) resolutions at attack transients, as discussed in the previous Transform Coding section. Otherwise, transform coders have poor temporal resolution and good frequency resolution at all other times. In comparison, wavelet coders have poor time resolution at low frequencies but good time resolution at high frequencies for all frames.

Despite all the work on wavelet audio coding in the past few years, it seems that these compression algorithms still cannot beat uniform, transform coders in terms of quality at low bit rates. One theory is that the high temporal resolution across high frequencies over all time is just not necessary for high quality audio coding (Johnston, 1998). With an eight band system, the highest octave of the

wavelet filter bank has 256 ($= 2^8$) times the temporal resolution of the lowest octave, and thus 256 times the data rate. Perhaps the reason commercial systems, as discussed in Section 4.1, do not use wavelets is due to wavelet's large latency and memory requirements, both of which are problems with wavelet coders. The memory increases as a function of the square of the number of octaves, as does the latency.

1.2 Audio Representations for Modifications

This section will discuss the representations of audio that allow for modifications to be performed. Examples of desired modifications are time-scale and pitch-scale modifications. In these cases, the time-scale and the pitch-scale can be independently altered. For example, time-scale modification would allow a segment of music to be played at half of the original speed, but would still retain the same pitch structure. This property is in contrast to the process of playing an analog, vinyl record at a slower-than-intended speed. When the record is played slower, the pitch drops accordingly.

Three classes of audio representations will be discussed in the following subsections: time-domain overlap-add, the phase vocoder, and sinusoidal modeling. Most methods used today can be categorized as one of these systems. With the exception of certain cases of sinusoidal modeling, these audio representations are not suitable for wideband audio data compression. If just data compression were the goal, some of the methods mentioned in the previous Section 1.1 would fare better. Rather, the methods described in this subsection have been optimized for their quality of modifications.

As an aside, only algorithms that perform modifications on pre-recorded audio will be considered in this section. This eliminates any parametrically *synthesized* audio, such as frequency modulation (Chowning, 1973), wavetable synthesis (Bristow-Johnston, 1996), or physical modeling synthesis (Smith, 1996). It is relatively simple to modify the time and pitch scale independently because of their respective representations, which can be expressed at very low bitrate (Scheirer and Ray, 1998). But, it is a very difficult problem for a computer to map an audio signal into parameters of these previously mentioned synthesized audio representations.

1.2.1 Time-Domain Overlap-Add (OLA)

Perhaps the simplest method for performing time-scale modification is a straightforward time-domain approach. Several papers show good results segmenting the input signal into overlapping windowed sections and then placing these sections in new time locations in order to synthesize a time-scaled version of the audio. The ranges of time-scale modification are somewhat limited compared to the phase vocoder, which will be discussed in the next subsection. This class of algorithms is referred to as Overlap-Add (OLA). To avoid phase discontinuities between segments, the synchronized OLA algorithm (SOLA) uses a cross-correlation approach to determine where to place the segment boundaries (Verhelst and Roelands, 1993). In the time-domain, pitch-synchronized OLA algorithm

(TD-PSOLA), the overlapping procedure is performed pitch-synchronously in order to retain high quality time-scale modification (Moulines and Charpentier, 1990). A more recent synchronization technique, called waveform similarity OLA (WSOLA), ensures sufficient signal continuity at segment joints by requiring maximal similarity to natural continuity in the original signal via cross-correlation (Verhelst and Roelands, 1993).

All of the preceding algorithms time-scale all regions of the input signal equally. That is, transients and steady-state segments are stretched equally. For high quality modifications, in both speech and audio, better results are obtained when only non-transient segments are time-scaled. The transient segments are not time-scaled, but rather translated. The earliest found mention of this technique is the Lexicon 2400 time compressor/expander from 1986. This model detected transients, left them intact, and time-scaled the remaining audio using a pitch-synchronous overlap add (TD-PSOLA) style algorithm (Dattorro, 1998). Improvements in speech intelligibility have been shown when using time-scale modification on only non-transient segments of the dialog (Lee *et al.*, 1997; Covell *et al.*, 1998). The idea of handling transients and non-transients separately will arise again later in this thesis, in Section 2.2. In the newly presented system, transients and non-transients are handled differently for both reasons of quantization coding gain and quality of modifications.

1.2.2 Phase Vocoder

Unlike the previous time-domain OLA section, the phase vocoder is a frequency domain algorithm. While computationally more expensive, it can obtain high quality time-scale modification results even with time stretching factors as high as 2 or 3. The technique is a relatively old one, that dates back to the 1970's (Portnoff, 1976; Moorer, 1978). For an excellent recent tutorial on the basics of the phase vocoder, see Dolson (1986). The input signal is split into many (256 to 1024) uniform-frequency channels each frame, usually using the FFT. The frame is hopped usually 25% to 50% of the frame length, thus using an *oversampled* representation. Each complex-valued FFT bin is decomposed to a magnitude and phase parameter. If no time-scale modification is performed, then the original signal can be exactly recovered using an inverse FFT. In order to perform time-scale modification, the synthesis frames are placed further apart (or closer together for time-compressing) than the original analysis frame spacing. Then, the magnitudes and phases of the FFT frame are interpolated according to the amount of time compression desired.

While the magnitude is linearly interpolated during time-scale modification, the phase must be carefully altered to maintain phase consistency across the newly-placed frame boundaries. Incorrect phase adjustment gives the phase vocoder a *reverberation* or a *loss of presence* quality when performing time-scaling expansion. Recent work has shown that better phase initialization, phase-locking among bins centered around spectral peaks, and peak tracking can greatly improve the quality of time-scale modification for the phase vocoder (Laroche and Dolson, 1997). With these added peak tracking enhancements on top of the filter bank, the phase vocoder seems to slide

closer to sinusoidal modeling (to be introduced in the next section) in the spectrum of audio analysis/transformation/synthesis algorithms.

A traditional drawback of the phase vocoder has been the frequency resolution of its filter bank. If more than one sinusoidal peak resides within a single spectral bin, then the phase estimates will be incorrect. A simple solution would be to increase the frame length, which would increase the frequency bin resolution. But if a signal changes frequency rapidly, as in the case of vibrato, the frequency changes could get poorly *smoothed* across long frames. The frame length is ultimately chosen as a compromise between these two problems. Because of the bin resolution dilemma, phase vocoders also have some difficulty with *polyphonic* music sources, since the probability is then higher that different notes from separate instruments will reside in the same FFT bin. In addition, if there is a considerable amount of *noise* in the signal, this can also corrupt the phase estimates in each bin, and therefore reduce the quality of the time-scale modification.

1.2.3 Sinusoidal Modeling

Sinusoidal modeling is a powerful modification technique that represents speech or audio as a sum of sinusoids tracked over time (Smith and Serra, 1987; McAulay and Quatieri, 1986*b*). A brief introduction will be presented here, but many more details will be shown later in Chapter 3. At a given frame of audio, spectral analysis is performed in order to estimate a set of sinusoidal parameters; each triad consists of a peak amplitude, frequency, and phase. At the synthesis end, the frame-rate parameters are interpolated to sample-rate parameters. The amplitudes are simply linearly interpolated, however the phase is usually interpolated using a cubic polynomial interpolation filter (McAulay and Quatieri, 1986*b*), which implies parabolic frequency interpolation. These interpolated parameters can then be synthesized using a bank of sinusoidal oscillators. A more computationally efficient method is to synthesize these parameters using an inverse FFT (McAualay and Quatieri, 1988; Rodet and Depalle, 1992).

With a parametric representation of the input signal, it is now relatively simple to perform both pitch and time-scale modifications on these parameters. To perform pitch-scale modifications, simply scale the frequency values of the all of the sinusoidal parameters accordingly. To perform time-scale modifications, simply change the time between the analysis parameters at the synthesis end. The amplitude is easily interpolated across this different time span, but frequency and phase must be interpolated carefully in order to maintain smooth transitions between frames. Details about phase interpolation will be later discussed in Section 3.4.

Sinusoidal modeling has historically been used in the fields of both speech data compression (McAulay and Quatieri, 1995) and computer music analysis/transformation/synthesis (Smith and Serra, 1987). In the computer music world, sinusoids alone were not found sufficient to model high quality, wideband audio. From computer music, Serra (1989) made the contribution of a residual noise model, that models the non-sinusoidal segment of the input as a time-varying noise source.

These systems are referred to as sines+noise algorithms. More recently, people have started using sinusoidal modeling (Edler *et al.*, 1996) and sines+noise modeling (Grill *et al.*, 1998*a*) in wideband audio data compression systems and modifications. Some found that sines and noise alone were not sufficient, and added a transient model. These systems are referred to as sines+transients+noise systems, and were used for audio data compression (Ali, 1996; Hamdy *et al.*, 1996) and audio modifications (K.N.Hamdy *et al.*, 1997; Verma and Meng, 1998). The goal of this thesis is to use a sines+transients+noise representation that is useful for both audio data compression and modifications (Levine and Smith, 1998). Much more about this system will explained later in the thesis.

1.3 Compressed Domain Processing

This section investigates compression methods that allow modifications to be performed in their respective compressed domains. This is contrast to the methods in Section 1.1 that were optimized for data compression, but *not* modifications; or, the methods of Section 1.2 which were optimized for modifications and *not* data compression. As more and more media (speech, audio, image, video) are being transmitted and stored digitally, they are all data compressed at some point in the chain. If one receives data compressed media, modifies it somehow, and then retransmits the data compressed, then one would want to operate on the compressed data itself, as seen in Figure 1.10. Otherwise, one would have to uncompress the incoming data, modify the data, and then compress the data again, as can be seen in Figure 1.11. These extra steps of decompression and compression can be of very high complexity, not to mention requiring more memory and latency. In addition, there is a loss of quality every time most transform coders repeat encoding and decoding steps. Examples of such media modifications would be adding subtitles to video, cropping or rotating an image, or time-scaling an audio segment.

Fig. 1.10. Performing modifications in the compressed-domain

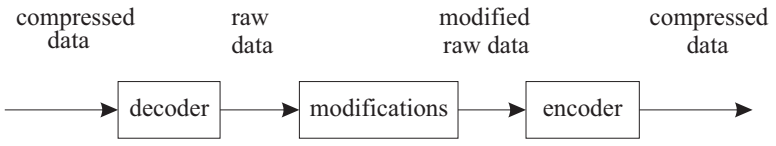

Fig. 1.11. Performing modifications by switching to the uncompressed domain

1.3.1 Image & Video Compressed-Domain Processing

With the advent of DVD, digital satellite broadcast, and digital High Definition Television (HDTV), there is now an enormous amount of digital, data-compressed video in the world. The data rate for raw, NTSC-quality television is approximately 100 Gbytes/hour, or about 27 Mbytes/sec. In addition, the complexity for video encoding using MPEG (Le Gall, 1991) or M-JPEG (Pennebaker, 1992) is enormous. Therefore, a huge savings is possible if modifications can be performed in the compressed domain. In the past few years, several researchers have been able to perform quite a few modifications on MPEG or M-JPEG standard bitstreams.

Video compression algorithms like MPEG and M-JPEG begin by segmenting images into blocks of pixels (usually 8x8). In M-JPEG, a block of pixels is first transformed by a two-dimensional DCT, then quantized, scanned in a zig-zag manner, and finally entropy encoded. MPEG encodes each frame in a similar manner to M-JPEG, but uses motion compensation to reduce the temporal redundancy between video frames. The goal of compressed-domain video and image processing is to perform the operations on the DCT values instead of having to use an inverse-DCT to get back to the spatial domain, perform the modifications, and then encode/compress the images again. Because of the entropy encoding and motion compensation, a partial decode of the bitstream is necessary. But, this partial decoding complexity is still far less than a full decode and encode.

The first results for compressed domain image & video processing were performed on M-JPEG data, which enabled compositing, subtitles, scaling, and translation using only compressed bitstreams (Chang *et al.*, 1992; Smith, 1994). It was also shown that the same modifications can be performed with an MPEG stream while accounting for the motion compensation vectors (Shen and Sethi, 1996). In addition, it is also possible to perform indexing and editing directly on MPEG streams (Meng and Chang, 1996). Indexing is based on searching the compressed bitstream for scene cuts, camera operations, dissolve, and moving object detection. The compressed domain editing features allows users to cut, copy, and paste video segments.

1.3.2 Audio Compressed-Domain Processing

While much work has been done in the image & video compressed domain world, comparatively little has been done in the audio compressed domain world. In the AC-3 compression standard, it is also possible to perform dynamic range compression on the compressed bitstream (Todd *et al.*, 1994). Dynamic range compression makes the loudest segments softer and the softest segments louder (Stikvoort, 1986). In this manner, the listener hears a smaller dynamic range, which is beneficial for non-ideal listening environments like car radios and movie theaters. For MPEG Layers I and II (Brandenburg and Stoll, 1994), it was shown that it is possible to downmix together several compressed data streams into one compressed stream, and still maintain the original bitrate (Broadhead and Owen, 1995). This MPEG standard will be discussed in more detail in Section

4.1. The same paper also discusses issues in performing equalization (EQ) on the compressed data, by using short FIR filters on the 32 uniform subband signals. These filters try to minimize the discontinuity in gain between adjoining critically sampled subbands, which could cause problems with the aliasing cancellation properties. In addition to EQ and mixing, it is also possible to perform effects processing on MPEG Layer I,II compressed audio streams. It was shown that effects such as reverberation, flanging, chorus, and reverb are possible on the MPEG Layer I,II subband data (Levine, 1996). These MPEG subbands are 687 Hz wide, and this frequency resolution never changes. Therefore, each subband signal can be treated as a time domain signal, that happens to be critically downsampled.

Because there are only 32 bandpass filters, each of the length 512 polyphase filters can have very sharp stopband attenuation, and little overlap between channels. While this method of effects processing works well for the relatively older MPEG Layers I & II, the same quality of results are not expected for the newer MPEG-AAC (Bosi *et al.*, 1997) (see Section 4.1 for more AAC details). In MPEG-AAC, the 1024 bandpass filters of length 2048 have a much slower rolloff, and thus more spectral overlap between channels. This high amount of overlap between the AAC subbands would make it difficult to perform independent effects processing on each channel. In addition, the window switching changes the frequency resolution from 1024 to 128 subband channels. Therefore, each subband output cannot be considered a static critically sampled signal, since the number of subbands changes during transients.

Time-scale modification is extremely difficult to perform in the compressed domain, assuming that the compression algorithm is a transform coder using MDCTs. First, consider each k^{th} bin coefficient over time as a signal, $x_k(t)$. This signal is critically sampled and will therefore be approximately white (spectrally flat), assuming a large number of bins. As was shown in Herre and Eberlin (1993), the spectral flatness of MDCT coefficients increases as the number of subbands increases. Also, the lowest frequency bins may be somewhat predictable (*i.e.*, colored), but the majority of the remaining bins are spectrally flat. In some cases of very tonal, steady-state and monophonic notes, the coefficients may be predictable over time. But, this case is rare in wideband, polyphonic audio. Attempting to interpolate an approximately white signal is nearly impossible. Therefore, interpolating critically-sampled MDCT coefficients across time, as is done with oversampled FFT magnitudes and phases for the phase vocoder, is extremely difficult. Also, it has recently been shown that time-scale modification via frequency domain interpolation is possible (Ferreira, 1998), but this is using the 2x oversampled DFT, not the critically-sampled MDCT, and has only shown good results for tonal signals.

Operations like dynamic range compression are still relatively simple for the most complex transform coders, but mixing compressed streams together becomes much more difficult. In order to mix N AAC streams, for example, one would have to perform a partial decoder of the N streams, sum their MDCT coefficients together, compute another psychoacoustic masking function of the summed

sources, and run the joint rate-distortion loops again. The psychoacoustic model and rate-distortion loops comprise the majority of the encoder complexity, so not much complexity is saved by working in the compressed domain. Of course, once these N streams are mixed, they cannot again be separated.

Therefore, modern transform coders, while getting very high compression rates, are not well suited for compressed domain processing. The main goal of this thesis is to provide a system that allows for high quality compression rates, while enabling the user to perform compressed-domain processing. While there are some slight tradeoffs in compression quality in this system relative to transform coders, there are great benefits in having a flexible audio representation that can be easily time/pitch modified. In the next chapter, the system level overview will be presented, showing how audio can be segmented and represented by sines+transients+noise. Afterwards, there is a detailed chapter for each of these three signal representations. Chapter 6 will then discuss how these representations can be time and pitch-scale modified.

Chapter 2

System Overview

As stated in the previous chapter, the goal of this thesis is to present a system that can both perform data compression and compressed-domain modifications. In this chapter, the overall system design will be described at a high level. In Section 2.1, the segmentation of the time-frequency plane into sines, transients, and noise will be discussed. The transient detector, which assists in segmenting the time/frequency plane, will be discussed in Section 2.2. The compressed domain processing algorithms will then be briefly discussed in Section 2.3. After the high level description in this chapter, the next three chapters will describe the sines, transients, and noise representations in much more detail.

Because this system is designed to be able to perform both compression and compressed-domain modifications, some tradeoffs were made. If one wanted just high quality data compression, one would use some variant of the modern transform coders (Bosi *et al.*, 1997). If one wanted the best wideband audio modifications possible today, one would most likely use some sort of phase vocoder (Laroche and Dolson, 1997) or time-domain techniques (Moulines and Laroche, 1995), depending on the time-scaling factor (even though both have difficulties with transients). While both of these systems are good in their own realms, neither can perform both data compression *and* modifications. This system is designed to be able to compress audio at 32 kbits/ch with quality similar to that of MPEG-AAC at the same bitrate (Levine and Smith, 1998). This bitrate is usually one of the benchmarks for Internet audio compression algorithms, since most 56 kbps home modems can easily stream this data rate. In addition to data compression, high quality time-scale and pitch-scale modifications can also be performed in the compressed domain.

2.1 Time-Frequency Segmentation

This hybrid system represents all incoming audio as a combination of sinusoids, transients, and noise. Initially, the transient detector segments the original signal (in time) between transients and

non-transient regions. During non-transient regions, the signal is represented by sinusoids below 5 kHz, and the signal is represented by filtered noise from 0 to 16 kHz. During short-time transient regions (approximately 66 msec in length), a transform coder models the signal. At the transitions before and after the transients, care is taken to smoothly cross-fade between the representations in a phase-locked manner. A high level system flow diagram can be seen in Figure 2.1.

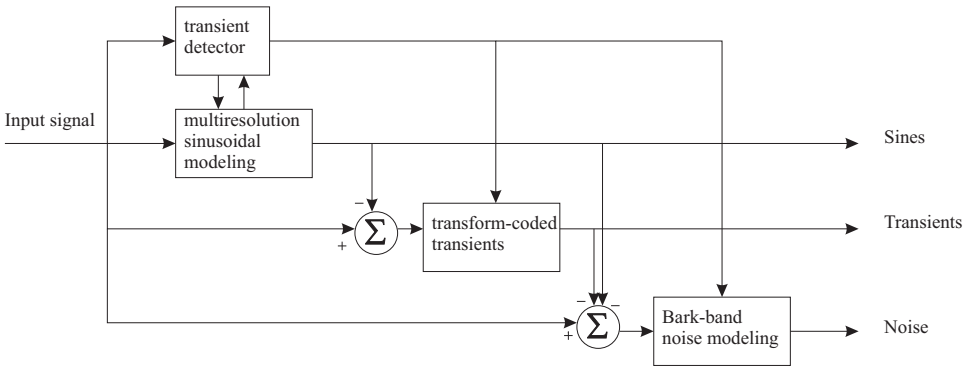

Fig. 2.1. A simplified diagram of the entire compression system

For a graphical description of the time-frequency segmentation, see Figure 2.2. The region from 45 to 110 msec is clearly marked as a transient region. The transient signal is a bass drum hit. During that time period, the time-frequency plane is uniformly segmented into equal area rectangles corresponding to the uniform bin widths of an MDCT transform coder. This transform coder will briefly be discussed in Section 2.1.2. During the non-transient regions and below 5 kHz, the time-frequency plane is non-uniformly divided using multiresolution sinusoidal modeling techniques, to be introduced in Section 2.1.1. Above 5 kHz during the non-transient regions, the time-frequency plane is uniformly divided in time, but nonuniformly in frequency. These time-frequency rectangles are each represented by noise modeling, to be discussed in Section 2.1.3. Below 5 kHz during the nontransient region, there is also noise modeling that performs a non-uniform division in frequency. But for simplicity of the following time/frequency diagrams, only the multiresolution sinusoidal modeling is shown below 5 kHz.

2.1.1 Sines

As will be discussed in much more detail in Chapter 3, the non-transient region from 0 to 5 kHz is modeled as a sum of time-varying multiresolution sinusoids (Levine *et al.*, 1998). The representation is termed *multiresolution* because the analysis front-end used to obtain the parameters is an octave-spaced filter bank. The parameters in the lowest octave are obtained with good frequency and poor time resolution. The highest octave has good time and poor frequency resolution. By varying the time-frequency resolutions, artifacts such as pre-echo are reduced, and quality is generally improved for polyphonic audio. This multiresolution analysis is the reason that the time-frequency tiling in

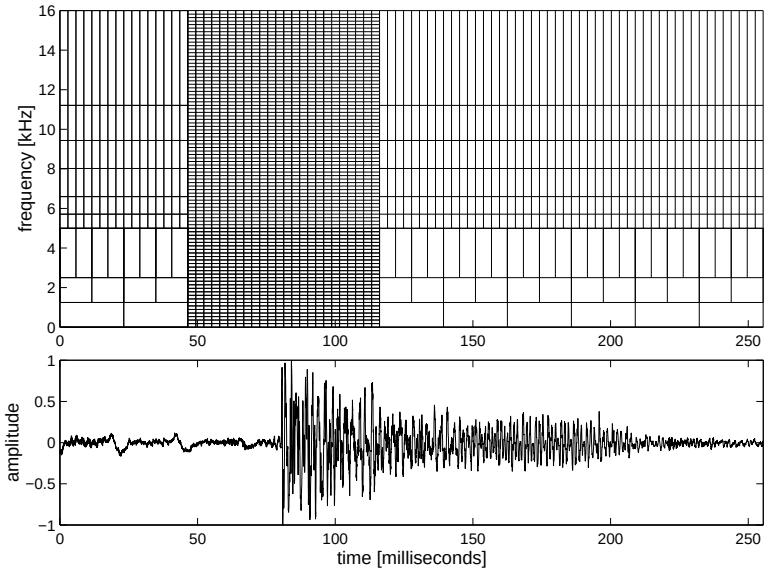

Fig. 2.2. The initial time-frequency segmentation into sines, transients and noise

Figure 2.2 is non-uniform below 5 kHz. Each non-uniform rectangle represents a set of sinusoidal parameters at a given octave at a given time-frequency resolution.

Sinusoids model only the bandwidth from 0 to 5 kHz for several reasons. First of all, no signal is represented by sinusoids above 5 kHz for bitrate constraint reasons. For most music at the low target bitrates ($\leq$ 32 kbps/ch), modeling the bandwidth between 5 to 16 kHz as filtered noise is not an unreasonable approximation. One can generate worst-case signals that would break this approximation, such as solo, bright-timbre harmonic instruments. But, this system was created for any generic polyphonic audio input. Most music has few isolated sinusoidal peaks above 5 kHz; thus, this seems like a reasonable approach. Ideally, and with more system complexity, the cutoff frequency for sinusoids could adapt with the input signal. If one desires a system for higher quality modification and synthesis with no bitrate constraints, sinusoidal modeling can be used over the entire bandwidth of the audio signal. In this case, a residual noise model would also span the entire bandwidth, as was performed in Serra (1989).

2.1.2 Transients

Sinusoidal modeling proved to be insufficient to model signals such as sudden attack transients well. Most attack transients are broadband signals, meaning that there are no *tonal* peaks. One could model an attack with hundreds of sinusoids (additionally requiring costly phase information), but this would require too many bits. In addition, the beginning time-domain envelope is crucial in representing these attacks well. The window lengths necessary for sinusoidal modeling to resolve tones in the lower octaves are too long to model the sudden changes in amplitude. If one tried

to approximate attacks using sinusoidal modeling, the many sinusoids would slowly be ramped on before on the transient, resulting in *pre-echo*. For these reasons, transform coding is used over the short time region (approximately 66 msec) of the attack transient onset. Modeling transients using transform coding will be described in more detail later in Chapter 4.

2.1.3 Noise

During all non-transient regions of audio, all frequencies above 5 kHz are efficiently modeled as Bark-band noise. As mentioned previously, certainly not all audio is simply filtered noise above 5 kHz. But at very low bitrates, the inaccuracies of modeling high frequency signals with noise sounds much better than the artifacts of a bit-starved transform coder or a sparsely populated sinusoidal coder. The bandwidth between 5 and 16 kHz is divided into six Bark bands, and every approximately 3 msec, an energy gain in each Bark band is estimated. To synthesize the noise, the decoder generates white noise, and is then filtered by the Bark-band gains. During non-transient regions of audio below 5 kHz, the residual between the multiresolution sinusoids and the original signal is also modeled as Bark-band noise. The low frequency, residual noise model has a slower frame rate than that of the high frequency noise model, but the noise is modeled in mostly the same manner. More details on the psychoacoustics and the quantization methods of noise modeling will be discussed in Chapter 5.

2.2 Transient Detection

The transient detector determines when a signal switches between the parametric representation of sines+noise and the non-parametric, transform coded transients. By switching back and forth, the transients are faithfully represented by transform coding for only a short amount of time (66 msec). These transient signals change so quickly in time and frequency, the sinusoidal modeling and Bark-band noise modeling cannot represent the waveform well. Because the system needs to perform time and pitch-scale modifications on the compressed data, transform coding is only used for a very short amount of time. Transform coding is *not* a parametric representation; that is, its critically sampled coefficients cannot be stretched, scaled, or manipulated to perform modifications. As will soon be discussed briefly in Section 2.3, these transients are translated in time to perform time-scale modifications, rather than *stretching* the transients themselves.

2.2.1 Other Systems

The only other reference found that uses transient detection with sinusoidal modeling is Masri and Bateman (1996). In that system, which is based on a traditional sines+noise representation (Serra, 1989), an FFT magnitude transient detector with a linear frequency weighting factor locates transient locations. Care is taken such that no sinusoidal analysis window straddles a transient region. It

is assumed that the sinusoidal parameters are reasonably stationary before and after the transient point in time. Since analysis windows are placed in time such that none crosses a transient boundary, it is assumed that the sinusoidal parameters are reasonable estimates of the signal. The signal at the transient region is extrapolated from the sinusoidal analysis windows just before and after the transient region, which are crossfaded across a 5.8 msec region. While improvements are shown for some transients, they mention some degradation for bass drum and snare hits. This is most likely due to the crossfading of synthesized frames strictly before and after the attack transient. No explicit analysis is performed during the attack transient, which holds much of perceptually important information of the entire sound (Grey, 1975).

2.2.2 Transitions between Sines and Transients

In the system presented in this thesis, no sinusoidal analysis window overlaps the transient location either. The time region not covered by analysis windows is covered by a series of short (2.9 msec) windows for an MDCT transform coder. This property can be seen in Figure 2.3. Notice that none of the three octaves' sinusoidal analysis windows cover the time range of the initial drum attack at t=80 msec. These separate windows in each octave will be described in more detail later in Section 3.1. But, in the time region surrounding the drum attack, the short (2.9 msec) windows from the MDCT transform coder are used. It is important to note that the three octaves of windows correspond to sinusoidal modeling while the top plot of short windows correspond to transform coding the transient.

2.2.3 Transient Detection Algorithm

Much care is taken in choosing the correct transient onset times. On the one hand, one doesn't want too many transient onsets per second. Taken to the extreme and the entire signal were classified as transient onsets, then the compressed transform-coded transient data could *not* be time or pitch scaled (only sines+noise can be scaled; transients are translated). On the other hand, if the transient detector were not sensitive enough, some attacks would be modeled by sines+noise and would have dull onsets.

After much experimentation, the transient detector uses two combined methods to find the onset times. The first method looks at the rising edges of the short-time energy of the input signal. This can be seen as the bottom chain in Figure 2.4. The energy estimate is taken over 512 point (@ 44.1 kHz) Hamming windows, with an overlap of 50%. The *rising edge detector* is a simple predictor that looks at the present frame's energy estimate versus a weighted sum of previous frames' energies. If the current energy is much larger than the average of the previous short-time energies, then this frame is a candidate for being a transient region.

The second method first looks at the short-time energy of the *residual* between the original signal

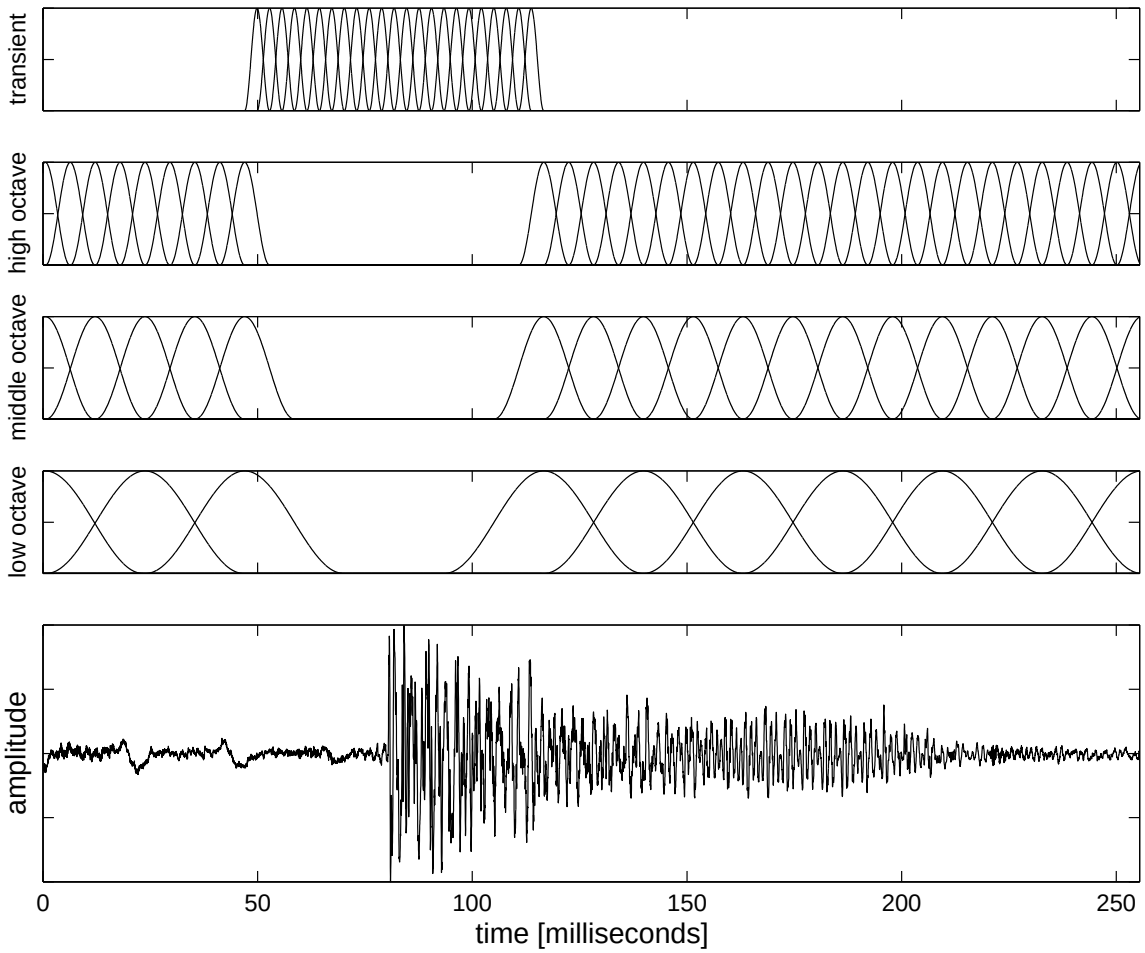

Fig. 2.3. This figure shows the analysis windows for the transform-coded transients (top) and the three octaves of multiresolution sinusoidal modeling (middle three). The bottom plot shows the original time domain signal of a bass drum hit.

and its synthesized version using multiresolution sinusoidal modeling. At each frame l, with hop size M, a ratio is taken between the short-time energies:

$$ratio(l) = \frac{\text{residual energy}(l)}{\text{original energy}(l)} = \frac{\sum_{n=lM}^{l(M+1)-1} w(n)[y(n) - x(n)]^2}{\sum_{n=lM}^{l(M+1)-1} w(n)x^2(n)} \tag{2.1}$$

In the above equation, $x(n)$ is the original signal, $y(n)$ is the signal synthesized using multiresolution sinusoidal modeling, and $w(n)$ is the Hamming window. All phase information is used in this instance of multiresolution sinusoidal synthesis in order to create a meaningful residual. At the decoder, only phase information surrounding the transient region will be received. More on these phase specifics will be discussed in Section 3.4.3.

When the ratio is near zero, sinusoidal modeling was a reasonable representation of the original

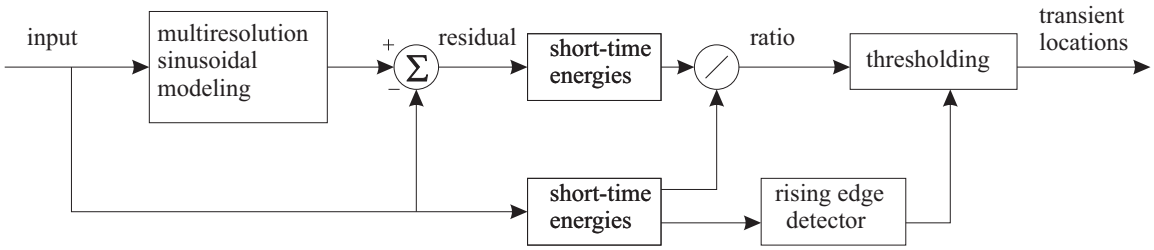

Fig. 2.4. The transient detector

signal in that frame (in a mean square sense). Because sinusoidal modeling represented the signal well, it is not likely an attack transient occurred in this frame. When the ratio is near one, the sinusoidal modeling did not perform a close match to the original, and makes this frame a likely candidate for a transient region. The ratio can actually be much larger than one in the case of a rapid attack transient due to pre-echo problems. Pre-echo, which will be discussed later in Section 3.1, is less of a problem using multiresolution sinusoidal modeling than traditional sinusoidal modeling. Nonetheless, pre-echo is still present and audible even with multiresolution techniques. As a graphical example, see Figure 2.5. The top plot shows the original bass drum attack, with its sudden attack at t=80 milliseconds. When the signal is synthesized using only multiresolution sinusoidal modeling, the synthesized signal's amplitude is ramped up *before* t=80 msec due to the analysis window lengths. Therefore, the energy of the residual gets much larger than the original signal's energy right around the attack transient. This makes the energy ratio large, and thus detects a transient location.

Finally, the *thresholding* decision in Figure 2.4 looks at the onset time region candidates from both methods. If both methods agree in a given frame that the ratio of energies is high and there is a rising edge in the original short-time energy, then the current frame is labeled as a transient region. Once a transient frame is found, the surrounding frames in a neighborhood of the previous 50 msec and the future 150 msec are forever labeled as non-transient regions. In this manner, spurious transients are eliminated and the total number of transients is limited to five per second. To view all of the steps in determining the final transient locations, see Figure 2.6.

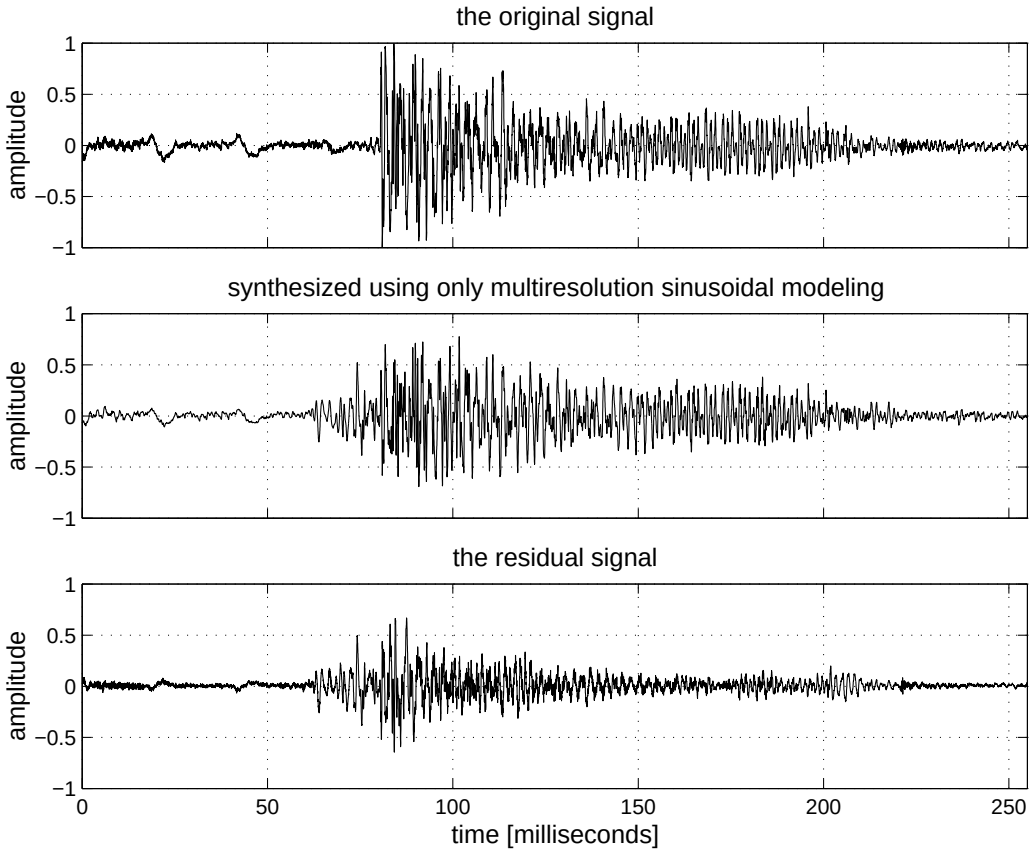

Fig. 2.5. This figure shows the pre-echo problems due to sinusoidal modeling. The top plot shows the original signal, and the middle plot shows the signal modeled using multiresolution sinusoidal techniques. The bottom signal shows the residual between the top two plots. Pre-echo artifacts are evident between t=60 to 75 msec, whereby the sinusoids are ramped from zero amplitude towards the intended amplitudes during the attack transient. The transient detector will locate evidence of the pre-echo artifacts and determine this time as a transient region. Therefore, transform coding will be used during the time region instead of sinusoidal (and noise) modeling. An example of the different models adaptively switching can be seen later in Figure 4.3.

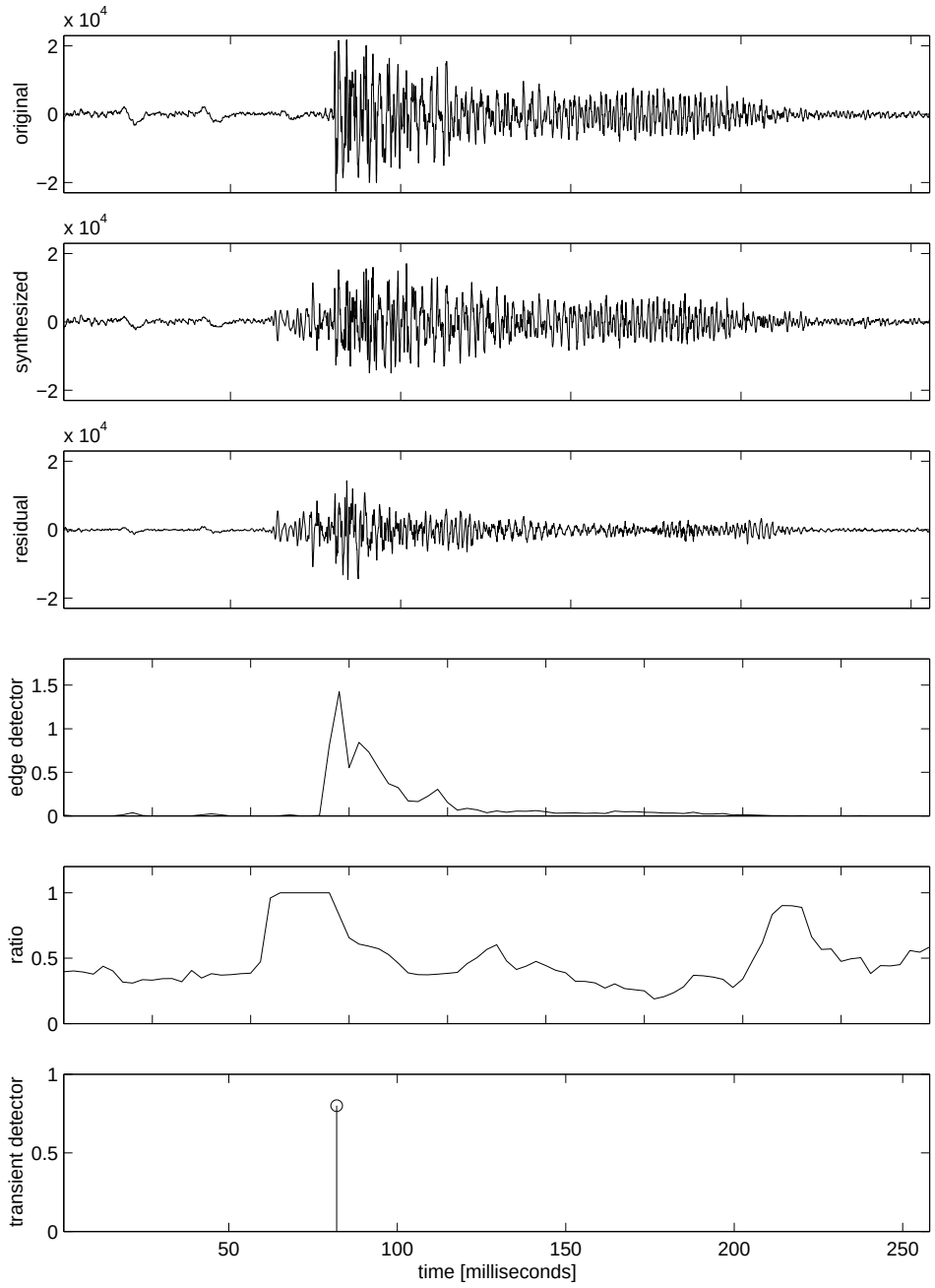

Fig. 2.6. This figure shows the steps in finding a transient location. The top plot shows the original bass drum hit, as was shown in Figure 2.5. The second plot shows the sinusoidal synthesized version of the original signal. The next plot is the residual signal between the top two plots. The fourth plot shows the output of the rising edge detector of the original input signal. The next figure shows the ratio between the short-time energies of the residual and the original energies, as shown in Equation (2.1). The final plot at the bottom shows the result of the complete transient detector. At approximately t=80 msec, a transient is successfully located at the bass drum attack time. Both the rising edge detector of the original input energy, and the short-time energy ratio are high at that point in time.

2.3 Compressed Domain Modifications

The benefit of segmenting audio into sines+transient+noise is that each of the three representations can be handled separately when performing modifications. Because multiresolution sinusoidal modeling and Bark-band noise modeling are parametric representations, they can be stretched, scaled, and manipulated easily. But, the MDCT coefficients in transform coding cannot be so easily modified. Transform coding is chosen primarily for its data compression capabilities, especially during times of transients, when sines and noise modeling fail. This inability to modify MDCT coefficients is not a liability because they only model very short regions of the audio signal. More details of these compressed-domain modifications will be discussed in Chapter 6.

2.3.1 Time-Scale Modifications

Since the transients are separated from the non-transient signals (sines+noise) in this representation, they are handled separately for time-scale modification. When slowing a signal, one desires to still have the attacks sudden and crisp. If a drummer plays a song at half the original speed, the drumsticks still hit the drum head at the same speed. The attacks are just as abrupt; they just happen less often. The same paradigm is present in this system; the non-transients are stretched in time, but the transients are simply translated to their new position in time. Figure 2.7 gives a graphical result of this system. Notice that the abrupt attack transient is present in both the original and the time-scaled synthesized version. More details of the time-scale modification, and precisely how sines, transients, and noise are altered will be presented in Section 6.1.

2.3.2 Pitch-Scale Modifications

Pitch-scale modification is processing that changes the pitch structure of a signal without changing its playback speed. In this system, the pitches of the sinusoidal modeling region can easily be changed by altering their respective frequency parameters. The transient information is currently not pitch shifted, since the attacks are usually broadband and do not contain pitched information. The noise signal is also not pitch-altered. More details on pitch-scale modification will be shown later in Section 6.2.

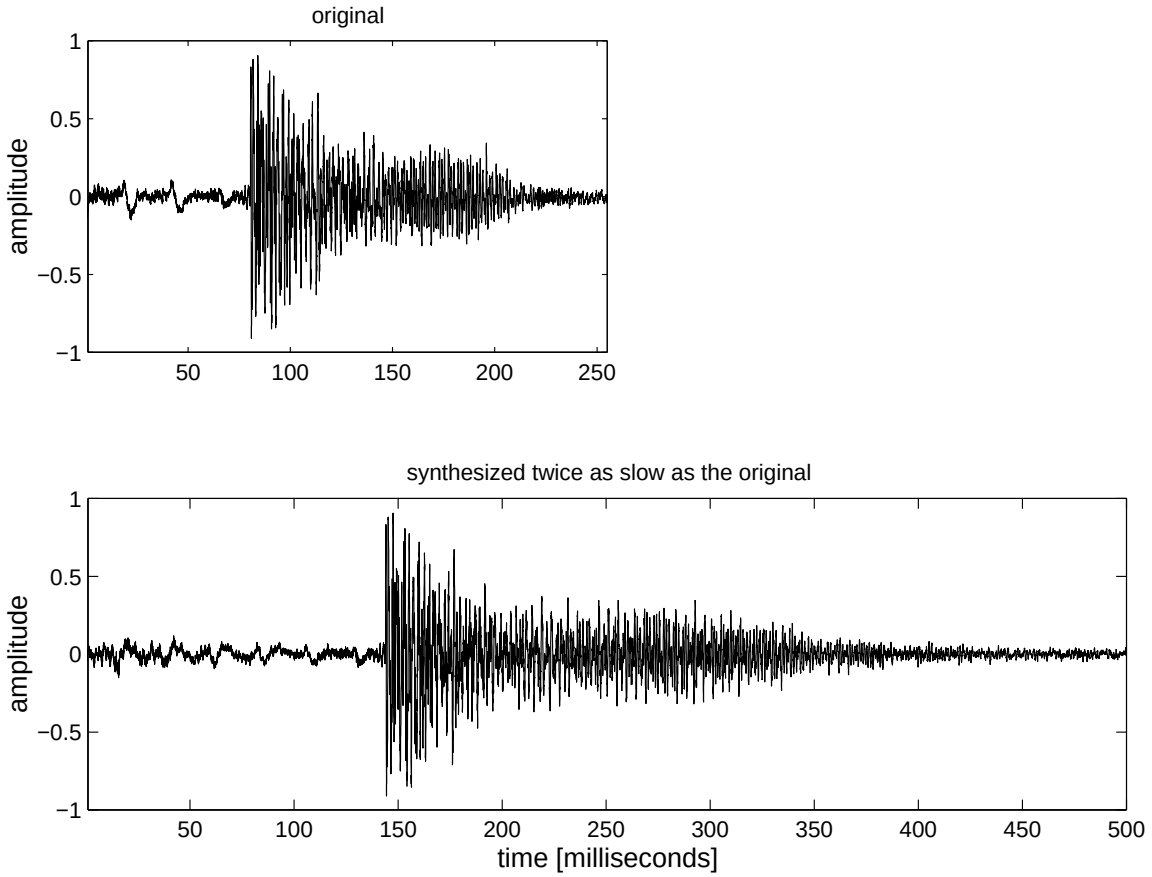

Fig. 2.7. This figure shows an example of time-scale modification. The top figure shows the original signal of a bass drum hit, previously shown in Figures 2.5 and 2.6. The bottom plot shows the signal time-scale modified slower by a factor of two. Notice that the attack is just as sudden as in the original signal, but the decay region has been stretched.

Chapter 3

Multiresolution Sinusoidal Modeling

Sinusoidal modeling has been shown over the years to be a powerful signal representation for speech and monophonic audio (Serra, 1989; McAulay and Quatieri, 1986*a*). By modeling the input signal as a set of slowly varying sinusoids, one gains both a bitrate efficient and malleable representation. In the simplest form, sinusoidal modeling can be pictured as in Figure 3.1. The signal is windowed into frames, and then processed through a filter bank, with K uniformly spaced frequency bins. Parameter estimation is performed to obtain amplitude and frequency (and sometimes phase) parameters of R meaningful spectral peaks. After the parameter estimation, some sort of tracking is usually performed, that groups sinusoidal parameters from frame to frame into trajectories. To synthesize the signal, each set of sinusoidal parameters in a trajectory is sent to an oscillator or an IFFT (Rodet and Depalle, 1992), and the result is summed.

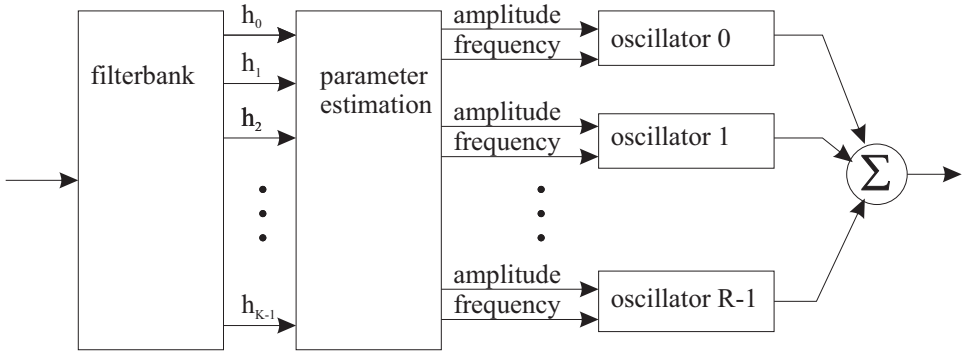

Fig. 3.1. A simplified model of sinusoidal modeling

Classical sinusoidal modeling does have some drawbacks, however. For one, transients are not well modeled with a bank of sinusoids because transients are usually broadband in frequency and

not tonal. This problem has been mostly solved by segmenting the signal and allowing a transform coder to model all the transient-like segments of the input, as will be later discussed in Chapter 4. Therefore, the sinusoidal modeling section only observes *non-transient signals*. A second drawback of classical sinusoidal modeling is its inability to handle polyphonic inputs well (as opposed to speech and solo instruments). By using multiresolution analysis techniques to be discussed in this chapter, polyphonic audio inputs are now well modeled. Another development presented in this chapter is the psychoacoustically motivated quantization methods of the sinusoidal parameters in order to minimize the bitrate requirements of such a representation.

An overview of the newly developed sinusoidal modeling subsystem can be seen in Figure 3.2. Initially, the input audio is first split into three octave-band signals by the *multiresolution filter bank*, which will be discussed in Section 3.1. Each octave will then be further split into uniformly spaced frequency divisions using FFTs. Sinusoidal parameters, such as amplitude, frequency, and phase, are derived in the *parameter estimation* section, which will be discussed in Section 3.2. These parameters found in each frame of audio are formed into trajectories that can span several frames in the *tracking* subsystem, detailed in Section 3.3. Parallel to the tracking subsystem, the *masking thresholds* subsystem estimates the perceptual importance of each sinusoidal parameter and trajectory. These steps will be discussed in Section 3.5. The *transient detector* is the same algorithm as discussed is Section 2.2, and was shown previously in Figure 2.1. If the input signal is currently classified as a transient, then no trajectories are selected at that time so as to allow only the transient-modeling transform-coder to model the current input signal. Therefore, depending on the tracks' perceptual importance and the state of the transient detector, the *trajectory selection* subsystem chooses which sinusoidal tracks to keep and which to eliminate. This selection process will be discussed in Section 3.6. After all the tracks have been chosen, then the *trajectory quantization* subsystem efficiently represents the sinusoidal trajectory, which will be discussed in Section 3.7.

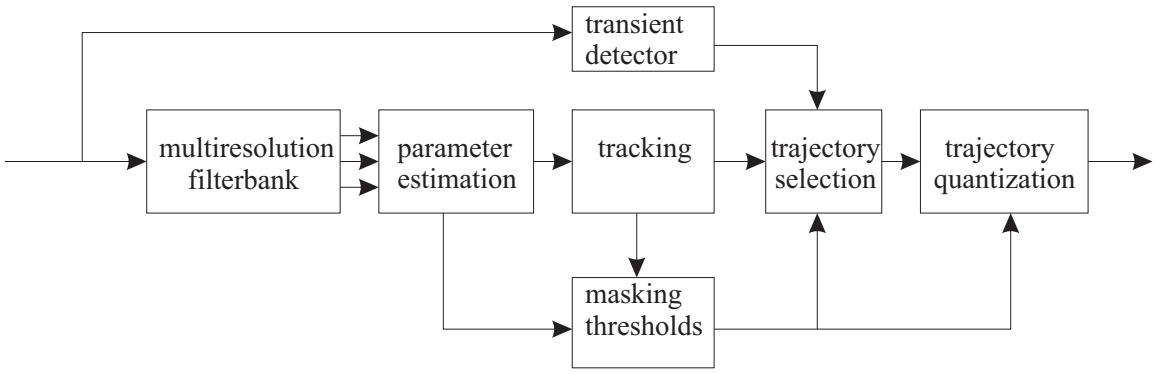

Fig. 3.2. An overview of the multiresolution sinusoidal system

3.1 Analysis Filter Bank

In sinusoidal modeling, the parameter estimation is always preceded by an analysis filter bank. The analysis filter bank, and thus the parameter estimation, is a frame-based algorithm. That is, parameters are updated once every frame. When the signal is eventually synthesized, sample-rate sinusoidal parameters are interpolated from the frame-rate parameter estimations that are deemed to be in the same trajectory. This decision of which frame-rate parameters belong to which trajectories will later be discussed in Section 3.3.

More specifically, for each frame l of hop-size S, the synthesized sound, $s(m+lS)$, can be described as

$$s(m+lS) = \sum_{r=1}^{R_l} A_{r,l} \cos[m\omega_{r,l} + \phi_{r,l}] \qquad m = 0, \ldots, S-1 \tag{3.1}$$

The sinusoidal frame-rate parameters $\{A_{r,l}, \omega_{r,l}, \phi_{r,l}\}$ show the estimated amplitude, frequency, and phase of the r^{th} partial in the l^{th} frame. To avoid frame-rate discontinuities, Equation (3.1) is altered to use interpolated parameters:

$$s(m+lS) = \sum_{r=1}^{R_l} \hat{A}_{r,l}(m) \cos[\hat{\theta}_{r,l}(m)] \qquad m = 0, \ldots, S-1 \tag{3.2}$$

The sample-rate amplitude parameters are simply linearly interpolated from the frame-rate parameters:

$$\hat{A}_{r,l}(m) = A_{r,l} + (A_{r,l+1} - A_{r,l})\frac{m}{S} \qquad m = 0, \ldots, S-1 \tag{3.3}$$

The instantaneous phase, $\cos[\hat{\theta}_{r,l}(m)]$, is a more complex interpolation utilizing the adjacent-time estimated frequencies $\{\omega_{r,l}, \omega_{r,l+1}\}$ and phases $\{\phi_{r,l}, \phi_{r,l+1}\}$. This phase interpolation algorithm will be described in more detail in Section 3.4.

3.1.1 Window Length Choices

No matter how elaborate the parameter interpolation algorithms are, there will still be synthesis artifacts due to the limited time-resolution of the frame-rate parameters. As a simplified example of this problem, view Figure 3.3. This artificial input signal contains a sinusoid that instantaneously begins at t=16 msec. The analysis window used in this example is 1024 points long (23 msec @ 44.1 kHz). In the synthesized signal, shown in the middle, the amplitude is linearly ramped, using Equation (3.3) over 23 msec., from t=6 to t=29 msec. From the bottom error signal, it is apparent that the error energy begins *before* the original signal begins. This artifact is called *pre-echo*, a term also used widely in the transform-coding audio compression world to describe the quantization noise spread evenly throughout a frame that could occur prior to an attack transient in that frame.

To reduce this pre-echo effect, one could simply reduce the analysis window length to below the perceptible temporal resolution of the ear. But, since the pre-masking effects of hearing is only

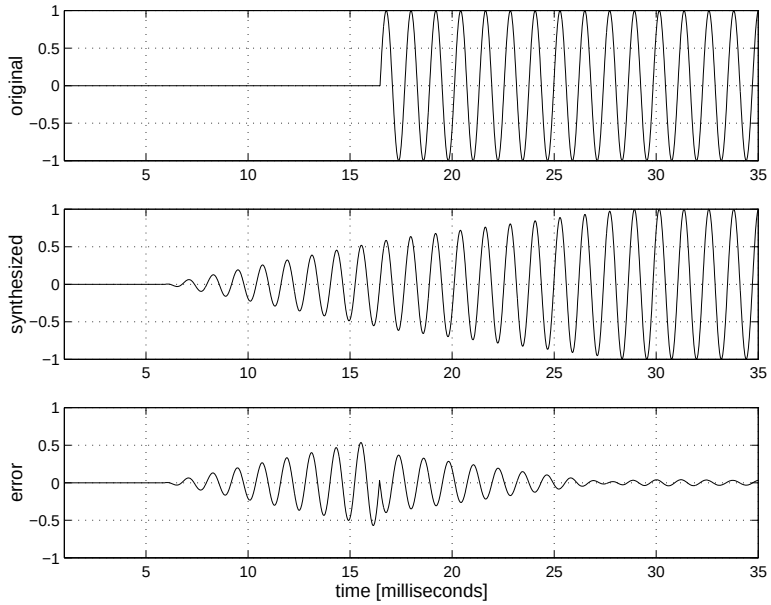

Fig. 3.3. Pre-echo artifacts in sinusoidal modeling

about 2-3 msec (Zwicker and Fastl, 1990), this analysis window would have extremely poor frequency resolution at low frequencies. A 2 millisecond window would have a bin spacing of 500 Hz. If two sinusoids were within the same 500 Hz wide bin, they could not be reliably resolved. Because the pitch scale is logarithmic, lower pitches (or keys on a piano) are much closer in frequency than high pitches. Thus, all lower pitches require higher frequency resolution than higher pitches. Put another way, it is desirable to have a window length at least two to three times the length of the sinusoid currently being estimated. Herein lies the problem: a sinusoidal analysis system desires both:

- *Good time resolution* in order to reduce pre-echo artifacts and track fast moving sinusoids like signals with vibrato
- *Good frequency resolution* in order to resolve two closely spaced sinusoids, especially at low frequencies

But, a single analysis window determines *both* time and frequency resolution. The better the time resolution, the poorer the frequency resolution, and vice-versa.

Previous works have chosen a pitch-synchronous analysis in order to resolve the time-frequency trade-offs (Serra, 1997; McAulay and Quatieri, 1995). As often as once a frame, a pitch estimate is calculated. Then, the analysis window is adjusted accordingly; the lower the pitch, the longer the window. This pitch-estimation does not solve the pre-echo problem, but does assure that all partials equal and above the fundamental will be resolved. In fact, the lower the fundamental pitch, the longer the window, and the worse the pre-echo problem. This approach works reasonably well

if the input signal can be assumed to be monophonic and single-pitched. For polyphonic audio, it is impractical to attempt to discern multiple pitches. Even if all pitches could reliably be discerned, then one still must choose one of the fundamental pitches to use for the *single* analysis window.

3.1.2 Multiresolution Analysis

To solve this window-length problem, the input signal is split into several bandlimited frequency ranges, and a window length is designated for each channel individually. This allows the system to have good frequency resolution in a low frequency range (with poor time resolution), but also good time resolution at higher frequencies (with poor frequency resolution). Thus, the sinusoidal analysis is termed *multiresolution*.

There have been several different previous approaches to solving the sinusoidal parameter estimation problem in a multiresolution manner. One method is to input a signal through an octave-spaced, critically sampled, wavelet filter bank (Goodwin and Vetterli, 1996; Anderson, 1996; Rodriguez-Hernandez and Casajus-Quiros, 1994), and perform sinusoidal modeling on the channels. This results in relatively low complexity, but there is no known way to eliminate all aliasing between channels in the filter bank (without performing an inverse filter bank). Therefore, each channel estimates sinusoidal parameters of the actual bandpassed-octave signal, in addition to parameters of the aliased octaves adjacent in frequency. It is possible to reduce these cross-talk aliasing terms (Anderson, 1996; Edler, 1992; Tang *et al.*, 1995), but complexity is now raised, and the results have not been sufficient for high quality, wideband sinusoidal modeling.

Another multiresolution approach to sinusoidal modeling is to use a parallel bank of constant-Q bandpass filters as a front end (Goodwin and Vetterli, 1996; Ellis and Vercoe, 1991). Then, one can perform sinusoidal modeling on each bandpass filter output. Although this method works well, there is no downsampling, and the structure is highly oversampled. The amount of data storage and complexity increases linearly with the number of bandpass filters.

A third recent approach is to take multiple sliding FFTs of the same audio input (Anderson, 1996). Each FFT has a different window length, and thus a different time-frequency plane tiling. The long windowed, low frequency FFT output is fed into sinusoidal modeling for low frequency sinusoids; similarly, short windowed, high-frequency FFT data is fed into sinusoidal modeling for high frequency sinusoids. Complexity in this algorithm also linearly increases with the number of octaves, of FFTs sets, taken.

In this system, an octave-spaced, complementary filter bank (Fliege and Zolzer, 1993) is the front end to a bank of sinusoidal modeling algorithms. Each channel output goes into a separate sinusoidal modeling block, with its own window and analysis parameters, as seen in Figure 3.4. Notice that there is no synthesis filter bank. The $\{A, \omega, \phi\}$ parameters are extracted from the several independent parameter estimation blocks, and then are fed into a sinusoidal synthesizer.

Thus, the two main problems of previous schemes are avoided: with the filter bank discussed in

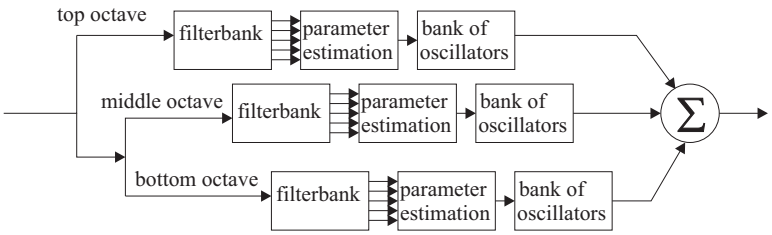

Fig. 3.4. The overall multiresolution sinusoidal modeling system. The front-end is a series of iterated complementary filter banks, of which each section is shown in Figure 3.5. For simplicity, the tracking, quantization, and psychoacoustic modeling are not shown.

the next section, the aliasing cross-talk problem as seen in wavelet filter bank front-ends is avoided. By introducing downsampling into the filter bank, the high costs of storage, memory, and complexity as seen in the constant-Q non-decimated filter banks, or the multiple FFT schemes, are eliminated.

3.1.3 Alias-free Subbands

The octave-spaced, oversampled filter bank is designed to assure that the subband signals are alias-free. There is overlap in frequency ranges between the channels, but no frequencies get folded over due to the subsampling. This filter bank structure is based upon the Laplacian pyramid structure (Burt and Adelson, 1983) from the multiresolution image compression world. The enhancement made to the Laplacian structure is the intermediate filter H_b, as seen in Figure 3.5. The filter H_b eliminates the spectral copy shifted to $\omega = \pi$ after the lowpass filter H_d and downsampling. If H_d were an ideal lowpass (brickwall) filter, then there would be no overlapping spectral copy; but this is not the case in practice. Each of the lettered steps in Figure 3.5 correspond to the simplified spectral diagrams in Figure 3.6.

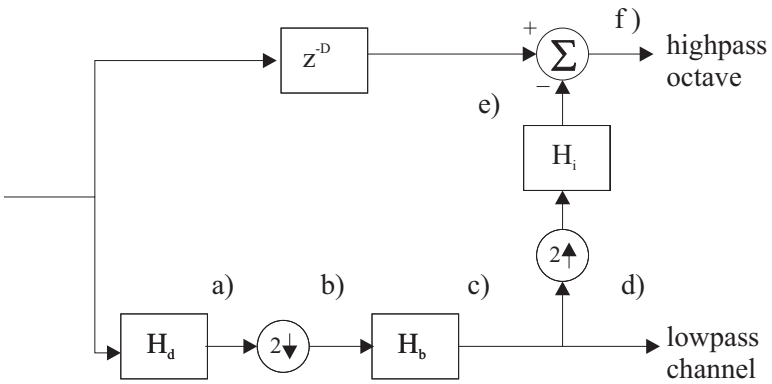

Fig. 3.5. A section of a complementary filter bank. The input signal, at sampling rate f_s is split into a highpass signal of rate f_s and a lowpass signal of rate $f_s/2$. Each lettered segment of the diagram is pictured in the frequency domain in Figure 3.6.

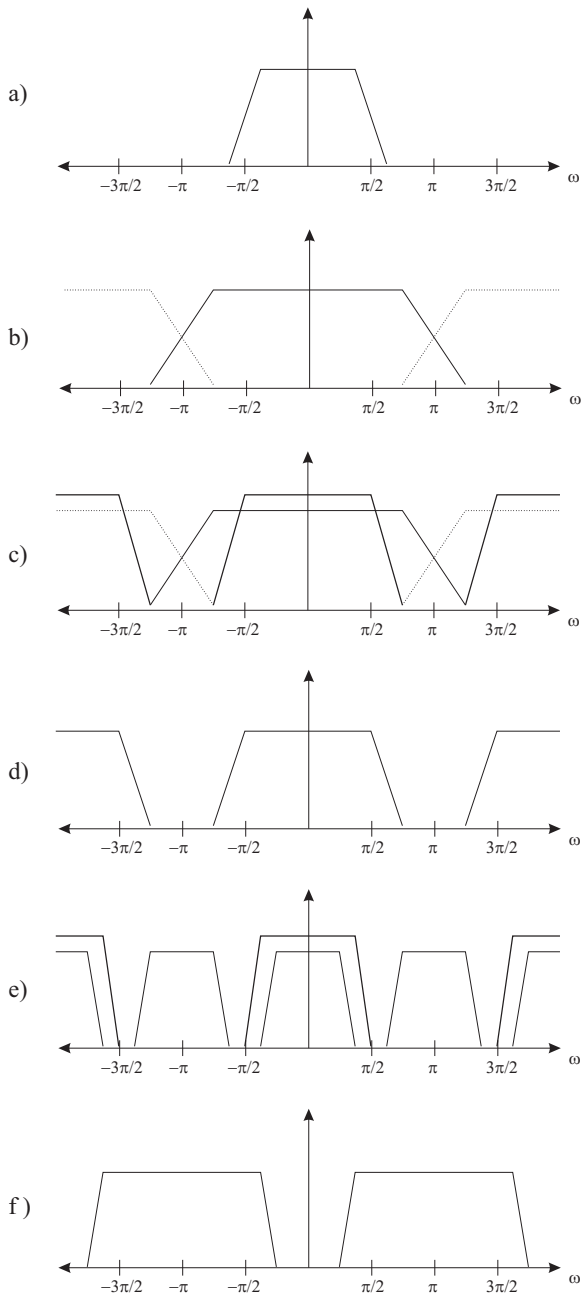

Fig. 3.6. The frequency domain steps of a complementary filter bank that attenuates any aliasing energy due to downsampling with non-ideal interpolation filters. Each lettered step corresponds to a signal diagram of Figure 3.5. Step a) shows the initial signal lowpass filtered. Step b) shows the lowpass signal after downsampling by two, along with the aliased energy in the neighborhood of $\pm\pi$. Step c) filters the previous signal with the bandlimiting filter H_b. This filter assures that the aliased energy is attenuated beyond perceptual limits. Part d) shows the resulting filtered signal, which is the subsampled, lowpass octave signal. Part e) shows the preceding signal when upsampled by 2, and then lowpass filtered by the interpolation filter H_i. The final step shows the preceding signal upsampled by two, which is the final highpass octave signal.

It is important to guarantee no aliasing in the subbands; if there were aliasing, the sinusoidal modeling blocks may generate parameters due to the aliased sinusoids in addition to the actual sinusoids. Since there is no synthesis filter bank, we cannot rely upon aliasing cancellation in synthesis. The sinusoidal synthesizer generates sinusoids from either a bank of oscillators, or from a block IFFT.

For this benefit of alias-free subbands, the filter bank can no longer be critically sampled; but rather, it is oversampled by a factor of two. This factor of two is independent of the number of octaves in the system. This is in contrast to the methods of Anderson (1996), and Goodwin and Rodet (1994), whose complexity and data rate grow linearly as a function of the number of octaves.

As a graphical example of these 2x oversampled octaves, see Figures 3.7 and 3.8. Figure 3.7 shows the original time-domain signal of a tenor saxophone note. Figure 3.8 shows the original magnitude spectrum at the top, from 0 to 5 kHz. The next three plots show the spectra of the 2x oversampled octaves. The top octave is still at the original sampling frequency, f_s, but the middle octave is downsampled 2x at $f_s/2$, and the lowest octave is downsampled 4x at $f_s/4$. Because of the successive downsampling, the harmonic peaks seem to be *stretched* apart in the lower octaves.

Notice also that there is no audible aliasing between channels. Since these are not ideal filters, there is some overlap in energy between the octaves. But, when synthesized with the correct phase, any partials that are contained in both channels constructively sum to the single, original partial. In cases where the sinusoids are synthesized without explicit phase information, as in Section 3.4.3, the system can look for these boundary case sinusoids, and alter the parameters such that they only appear in one of the octaves. Also notice the frequency regions of nearly zero energy in the plots of Figure 3.8. This is also due to the fact that the filter bank is two times oversampled. Thus, there is almost half of the bandwidth with zero energy. If the filter bank were critically sampled, then the plots would have no *dead-zones*, or frequency regions lacking energy. But, the energy would be partially aliased.

3.1.4 Filter Bank Design

In this system, sinusoidal modeling is only utilized between 0 and 5 kHz, and three octaves in this frequency range are desired. Since the input is at a sampling rate of 44.1 kHz, some adjustments must be made. The input signal could have been sample-rate converted from 44.1 kHz to 11.025 kHz, and then used two sets of iterated complementary filter bank sections to generate the three octave-band signals. Instead, the original signal is iterated through *four* complementary filter bank sections, thus producing five octave-band signals. Currently the top two octaves, from {11 to 22 kHz} and {5.5 to 11 kHz} are discarded in the analysis. Only the lower three octaves, ranging from approximately {2.5 to 5 kHz}, {1.25 to 2.5 kHz}, and {0-1.25 kHz} are currently used. For future work, and perhaps with higher allowable bitrates, these top octaves of sinusoidal modeling could also be used.

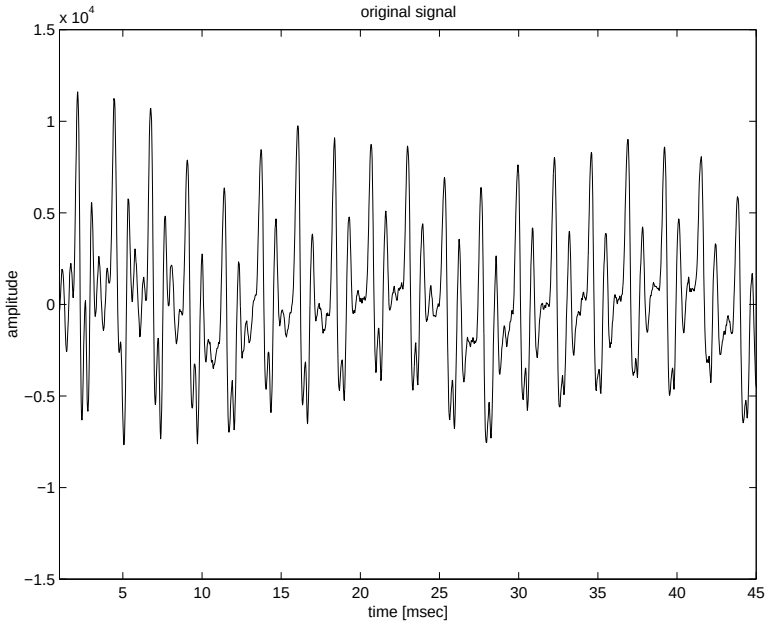

Fig. 3.7. The waveform of a saxophone note, whose spectrum is seen in Figure 3.8

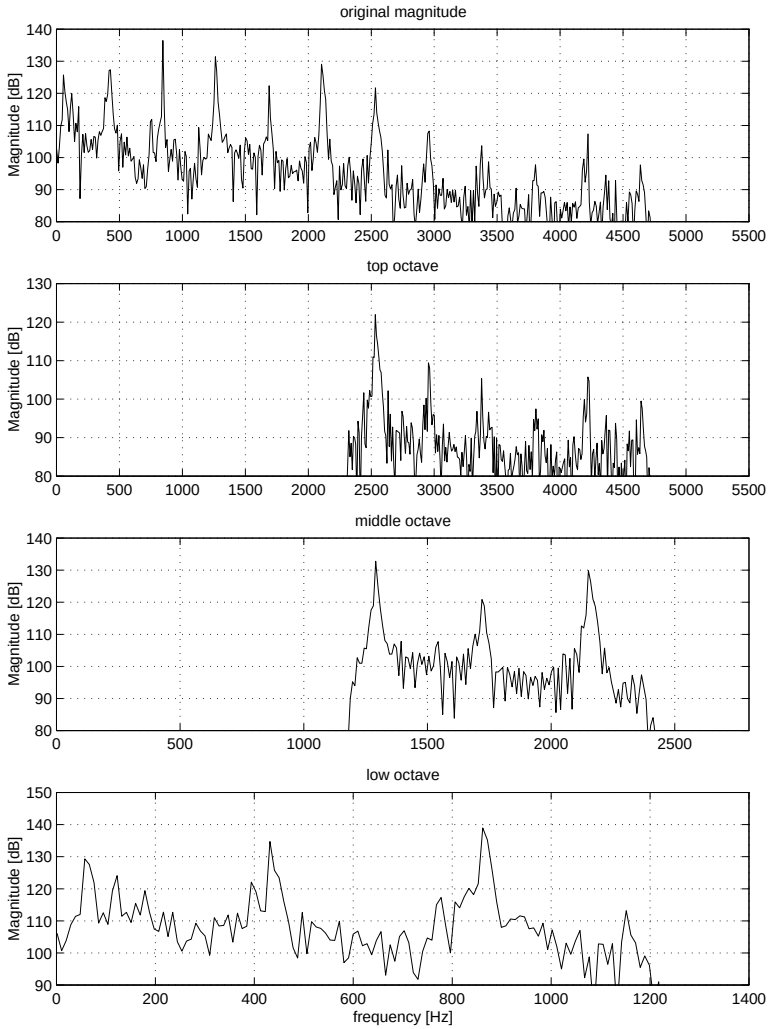

Fig. 3.8. The top signal is the spectrum of a saxophone note. The next three spectra show the oversampled octaves of the multi-complementary filter bank. Each frequency axis stretches from 0 to π radians, but has been relabeled using the original frequency [Hz] axis to correlate with the frequency axis of the original signal.

Each of the three filters $\{H_d, H_b, H_i\}$ are all 128 point FIR filters designed using MATLAB and the Remez-exchange algorithm. The interpolation and decimation filters are equivalent lowpass filters, having a cutoff frequency of $.45\pi$ and stopband starting at $.55\pi$. The bandlimiting filter, H_b, is a lowpass filter of almost twice the bandwidth due to its position after a downsampling operator. Its cutoff frequency is $.8\pi$ and bandstop initial frequency is $.9\pi$.

3.2 Parameter Estimation

Instead of using traditional FFT magnitude peak picking algorithms in order to locate spectral peaks (Serra and Smith, 1990), an approximate maximum likelihood estimation technique is used, based on the work of Thomson (1982). It was more recently used by Ali (1996) for audio sinusoidal modeling applications. The spectral estimation techniques used in this thesis are identical to that of Ali (1996), except that in this case, a multiresolution approach is used. Separate parameter estimation is performed on each of the three octaves, each with a different effective window length. In this manner, higher frequency resolution is shown at the lower octaves, where it is perceptually needed.

Where most other methods would find peaks in the spectrum by first windowing a signal, taking the FFT, and then locating peaks, this technique employs a set of orthogonal windows, called discrete prolate spheroidal windows, and then uses a statistical test called the *F-test* to decide if a sinusoid exists at a particular frequency (Thomson, 1982). The F-test, simply put, is a measure of the ratio between estimate of the spectral peaks at their respective frequencies to that of the continuous, non-harmonic part of the spectrum. If there are peaks in output of the F-test, called the *F-value*, at a given frequency, then it is assumed that there is a sinusoidal peak at that given frequency. An F-value peak is usually determined by first, being larger than a preset threshold, and second, being larger than its adjacent F-values in frequency. In addition to finding the magnitude at a given frequency, an estimation of phase is also given. This approach of sinusoidal estimation has not yet been compared directly to the simpler method of single-windowed FFT magnitude peak finding (Smith and Serra, 1987; Serra and Smith, 1990).

The results of the sinusoids estimated in this section are then checked against the results of the psychoacoustic masking thresholds of that particular frequency region, which will be discussed in more detail in Sections 3.5 and 3.6.

As mentioned in Section 3.1.2, an octave-band, 2x oversampled filter bank is used as a front-end to the parameter estimation and the rest of the sinusoidal modeling. By splitting the original signal into separate, non-aliased subbands, parameter estimation algorithms can be adjusted individually for each octave. In this system, the subband ranges and corresponding parameter estimation window lengths can be seen in the following table:

frequency range	window length	hop size
0-1250 Hz	46 ms	23 ms
1250-2500 Hz	23 ms	11.5 ms
2500-5000 Hz	11.5 ms	5.75 ms

To realize this system, the same length analysis windows were used on each subband. The effective window lengths are different in each octave because of the sample rate changes in each filter bank section, as pictured in Figure 3.5. The top octave has data rate f_s, and the middle octave has data rate $f_s/2$. Since the data rate of the middle octave is half of the top octave, then the effective window length is double that of the top octave. When the middle octave is iterated through another complementary filter bank section, then the lowest octave is now at a sampling rate of $f_s/4$, which makes its effective window length 4 times longer than that of the top octave. The resulting multiresolution time-frequency segmentation can be pictured in Section 3.9. In each time-frequency rectangle, a set of R_l sinusoidal parameters exist. Notice that the update rates for the sinusoidal parameters in each octave are different.

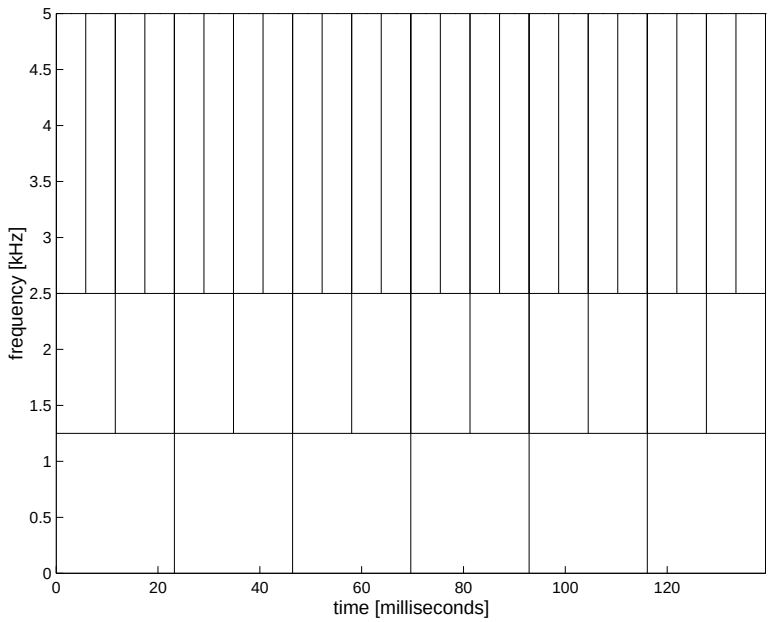

Fig. 3.9. The time-frequency segmentation of multiresolution sinusoidal modeling. Each rectangle shows the update rate of sinusoidal parameters at different frequencies. In the top octave, parameters are updated every 5.75 ms, while at the lowest octave the update rate is only 23 ms. There are R_l sets of sinusoidal parameters present in any one rectangle.

3.3 Tracking

Once sinusoidal parameters have been estimated at each frame l in each octave, then these parameters are tracked together in interframe *trajectories*. The notion of having trajectories is mostly for quantization purposes. For stationary segments of input signals, the variance of sinusoidal parameters in a single trajectory over time can be very low, and is thus easy to represent in few bits. Quantization details will be discussed later in Section 3.7. In addition, trajectory length can be an important metric in deciding if the sinusoidal parameters in the trajectory should be kept at all. The basic premise is that trajectories that are too short may not be suitable for sinusoidal modeling, but rather some other sort of noise modeling. This topic will further be discussed in Section 3.6.

The tracking system occurs in two stages for this system: the first stage pieces together parameters that fall within certain minimum frequency deviations. These trajectories are further split into more (shorter) trajectories if they do not satisfy amplitude deviation requirements. First, the frequency deviation tracking will be discussed. At frame l, there exists R_l sinusoidal parameters, and the frequencies are $\{\omega_{0,l}, \omega_{1,l}, \cdots, \omega_{R_l-1,l}\}$. At the next frame, $l+1$, there could be a different number of sinusoidal parameters than in the previous frame: $R_l \neq R_{l+1}$. The goal is then to match up as many parameters from frame l, as defined earlier, to those of frame $l+1$: $\{\omega_{0,l+1}, \omega_{1,l+1}, \cdots, \omega_{R_{l+1}-1,l+1}\}$. Several passes are made through the two frames of sinusoidal frequencies to make trajectories that assure that $|\omega_{m,l} - \omega_{n,l+1}| < \beta\omega_{m,l}$ (McAulay and Quatieri, 1986*a*; Smith and Serra, 1987). The frequency range multiplier, β, is usually set to 0.1, or a 10 cent deviation. If there are sinusoidal parameters in frame $l+1$ that could be connected to more than one parameter in frame l, then the algorithm chooses trajectories that minimizes frequency distance between the two. If a sinusoidal parameter $\omega_{n,l+1}$ could not be successfully matched with any parameter in frame l or $l+2$, it is then declared a trajectory of length one.

For every trajectory that has its first parameter begin at frame l: $\{A_{n,l}, \omega_{n,l}\}$, an extra parameter is appended to the beginning of the trajectory, with $A_{n,l-1} = 0, \omega_{n,l-1} = \omega_{n,l}$. This extra *ramp-on* parameter allows for a smooth onset in magnitude, but also contributes to the pre-echo artifacts as previously discussed in Section 3.1.1. The initial phase is set accordingly:

$$\phi_{n,l-1} = \phi_{n,l} - \omega_{n,l} \cdot S \tag{3.4}$$

where S is the frame's hop size (McAulay and Quatieri, 1986*a*). In the same manner, the end of each sinusoidal trajectory is appended with an additional parameter set to smoothly ramp-off the signal. If the trajectory were estimated to end at frame p, then $A_{n,p+1} = 0$ and $w_{n,p+1} = w_{n,p}$. The ending phase is computed in the same manner as the previous Equation (3.4).

After the trajectories have been formed to minimize frequency deviations, the trajectories are searched for any amplitude discontinuities. The rationalization behind this secondary process is that if a trajectory were found to have a large jump, say larger than $15dB$ jump in one frame length, then this is most likely two separate trajectories from two separate sound sources. For a graphical

example, see Figure 3.10.

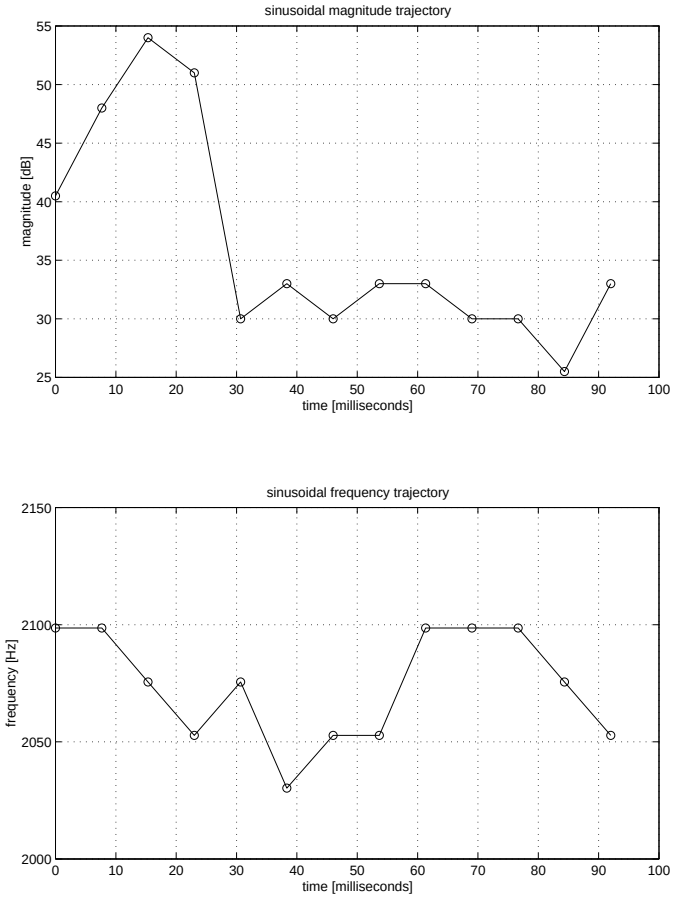

Fig. 3.10. These two plots show the sinusoidal magnitude and frequency trajectories. The circled parameters were estimated using the F-test at the frame rate. These parameters were quantized to an approximate just noticeable difference (JND) scale, to be discussed later in Section 3.7. At time 25 msec, there is a large magnitude jump of about 20 dB. In this case, the length 13 trajectory would be split into two shorter trajectories. The first one would last for four frames and then be ramped-off to zero magnitude. The second would ramp-on during the fifth frame and continue through the thirteenth frame.

3.4 Sinusoidal Phases

In sinusoidal modeling, transmitting phase information is usually only necessary for one of two reasons. The first reason for keeping phases is to create a residual error signal between the original and the synthesized signal. This is needed at the encoder's transient detector (Section 2.2), but not at the decoder. Thus, we need not transmit these phases for this purpose.

The second reason for transmitting phase information is for modeling attack transients well. During sharp attacks, the phases of sinusoids can be perceptually important. But in this system, no sharp attacks will be modeled by sinusoids; they will be modeled by a transform coder. Thus, we will not need phase information for this purpose.

A simple example of switching between sines and transients during attack transients is depicted in Figures 3.11 and 3.12. At time=40 ms, the sinusoids are cross-faded out and the transients are cross-faded in. Near the end of the transients region at time=90 ms, the sinusoids are cross-faded back in. The trick is to phase-match the sinusoids during the cross-fade in/out times while only transmitting the phase information for the frames at the boundaries of the transient region. This transient boundary phase-locking assures that no discontinuities will be heard when switching between signal modeling, even when performing time-scale modification.

To accomplish this goal, cubic polynomial phase interpolation (McAulay and Quatieri, 1986*a*) is used at the boundaries between the sinusoidal and transient regions. Phaseless reconstruction sinusoidal synthesis is used at all other times. Because only explicit phase information is sent at transient boundaries, which happen at most several times a second, the contribution of phase information to the total bitrate is extremely small.

First, the cubic-polynomial phase interpolation will be described. Then, the differences between it and *phaseless* reconstruction will be described. Afterwards, it will be shown how to switch seamlessly between the two phase interpolation schemes.

3.4.1 Cubic-polynomial Phase Reconstruction

Recall from Section 3.1 that during the l^{th} frame, we estimate the R sets of parameters $\{A_{r,l}, \omega_{r,l}, \phi_{r,l}\}$. These parameters must be interpolated from frame to frame to eliminate any discontinuities at the frame boundaries. The amplitude is simply linearly interpolated from frame to frame, as was shown in Equation (3.3). The phase interpolation is more complicated. We first create an instantaneous phase parameter, $\theta_{r,l}(m)$, which is a function of surrounding frequencies, $\{\omega_{r,l}, \omega_{r,l+1}\}$ and surrounding phases, $\{\phi_{r,l}, \phi_{r,l+1}\}$. In the following equations, the subscripts for the r^{th} parameter in that frame are dropped for simplicity. In addition, m will span from the frame boundary of frame l to the boundary of frame $l+1$, which of length S samples.

$$\hat{\theta}_l(m) = \zeta + \gamma m + \alpha m^2 + \beta m^3 \tag{3.5}$$

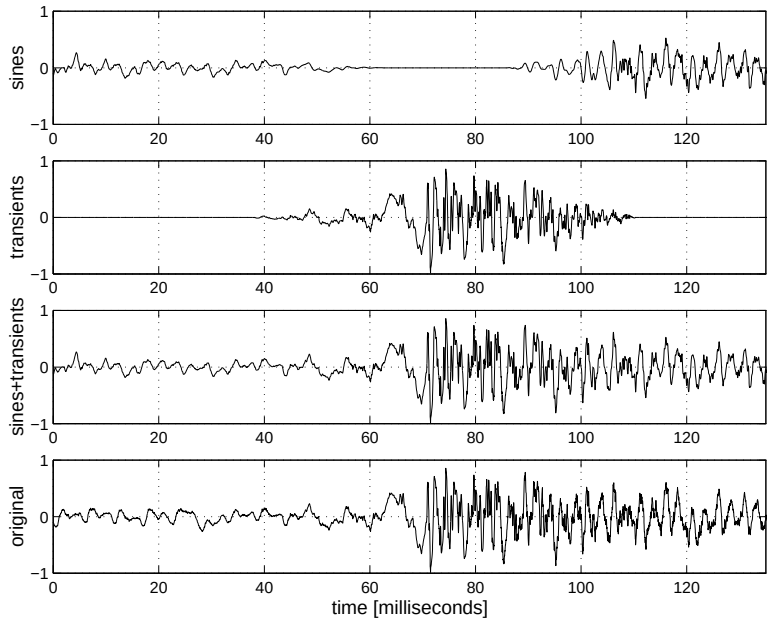

Fig. 3.11. This figure shows how sines and transients are combined. The top plot shows the multiresolution sinusoidal modeling component of the original signal. The sinusoids are faded-out during the transient region. The second plot shows a transform-coded transient. The third plot shows the sum of the sines plus the transient. For comparison, the bottom plot is the original signal. The original signal has a sung vowel through the entire section, with a snare drum hit occurring at t=60 ms. Notice that between 0 and 30 ms, the sines are *not* phase-matched with the original signal, but they do become phase-matched between 30-60 ms, when the transient signal is cross-faded in. For a zoomed-in view of this phase matching, view Figure 3.12.

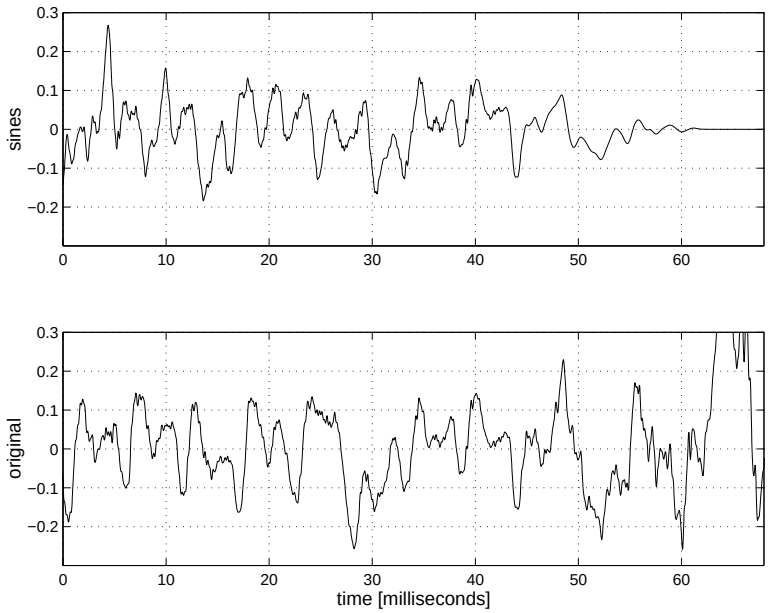

Fig. 3.12. This figure shows how the sinusoids become phase-locked to the original signal during the switching time between the sinusoidal and transient models. These views are zoom-in plots of those seen in Figure 3.11. The top plot, showing the sinusoidal modeling, uses phaseless reconstruction before t=30 msec, and cubic phase interpolation from t=30 to 65 milliseconds. Notice that before t=30, the sinusoids are not phase-locked, but after t=30, they are phase locked to the original signal. During this latter stage, the sinusoids are being cross-faded out to allow the transform-coded transients be heard.

To solve this equation, it is necessary that the above phase equation and its derivative equal the phases and frequencies estimated at the frame boundaries (frequency is the derivative of phase). Through some mathematics (McAulay and Quatieri, 1986*a*), the instantaneous phase can be found to be:

$$\hat{\theta}_l(m) = \phi_l + \omega_k m + \alpha(M^*)m^2 + \beta(M^*)m^3 \tag{3.6}$$

The parameters α and β are solved through a set of linear equations from the boundary conditions:

$$\begin{bmatrix} \alpha(M) \\ \beta(M) \end{bmatrix} = \begin{bmatrix} \frac{3}{S^2} & -\frac{1}{S} \\ -\frac{2}{S^3} & \frac{1}{S^2} \end{bmatrix} \begin{bmatrix} \phi_{l+1} - \phi_l - \omega_l S + 2\pi M \\ \omega_{l+1} - \omega_l \end{bmatrix} \tag{3.7}$$

There are many possible values of M, each delivering a different interpolated phase function that matched the boundary frequency and phase conditions. But, $M = M^*$ is chosen that finds the *smoothest* function, or the one that minimizes the integral of the square of the second derivative of $\hat{\theta}_l(m)$. After some algebra, M^* can be found to be:

$$M^* = \text{round}\left(\frac{1}{2\pi}\left[(\phi_l + \omega_l S - \phi_{l+1}) + (\omega_{l+1} - \omega_l)\frac{S}{2}\right]\right) \tag{3.8}$$

While using this phase interpolation algorithm allows for a high-quality, approximate phase-locked sinusoidal synthesis, it does come at a price. First of all, all estimated phases, $\phi_{r,l}$ must be quantized and transmitted for each frame. The trajectories of steady-state sinusoidal magnitudes and frequencies are reasonably smooth, but the phase trajectories are *not*. For polyphonic data, the phase trajectory signals are almost white, and therefore require more bits/parameter to encode than the magnitude or frequency trajectories. In addition to the added bitrate requirements, the complexity at the decoder necessary to perform this cubic phase interpolation is quite high. Therefore, cubic phase interpolation is only used when necessary: at the relatively seldom switching points between sinusoidal modeling and transform-coded transient modeling.

3.4.2 Phaseless Reconstruction

Phaseless reconstruction is termed *phaseless* because it does not need explicit phase information transmitted in order to synthesize the signal. The resulting signal will not be phase aligned with the original signal, but it will not have any discontinuities at frame boundaries.

Instead of deriving the instantaneous phase from surrounding phases and frequencies, phaseless reconstruction derives the instantaneous phase as the integral of the instantaneous frequency (Smith and Serra, 1987). The instantaneous frequency, $\omega_{r,l}(m)$, is obtained by linear interpolation:

$$\hat{\omega}_{r,l}(m) = \omega_{r,l-l} + \frac{(\omega_{r,l} - \omega_{r,l-1})}{S}m \qquad m = 0, \ldots, S-1$$

Therefore, the instantaneous phase for the r^{th} trajectory in the l^{th} frame is:

$$\hat{\theta}_{r,l}(m) = \theta_{r,l-1} + \hat{\omega}_{r,l}(m) \tag{3.9}$$

The term $\theta_{r,l-1}$ refers to the instantaneous phase at the last sample of the previous frame. The signal is then synthesized using Equation (3.2), but using $\theta_{r,l}(m)$ from Equation (3.9) instead of the result of a cubic polynomial interpolation function. For the first frame of phaseless reconstruction, the initial instantaneous phase is randomly picked from $[-\pi, \pi)$.

3.4.3 Switched Phase Reconstruction

This section will show how to switch between phase interpolations algorithms seamlessly. As mentioned earlier, phaseless reconstruction will be used for the majority of the time of the input signal. The synthesized sinusoids will *not* be phase-matched to the original. But for steady-state signals, maintaining original phase information among the many sinusoids is not perceptually important. In return, no phase information needs to be transmitted. When the transient detector locates a transient, the phase interpolation algorithm switches from the phaseless to the cubic polynomial phase interpolation algorithm. In this manner, the explicit phase information is transmitted, and then the synthesized signal becomes approximately phase-matched with the original.

As a simple example of how to switch between the interpolation algorithms, let the first transient in a signal begin at frame l. All frames $(0, 1, \ldots, l-2)$ will be synthesized using the phaseless reconstruction algorithm outlined in section 3.4.2. During frame $l-1$, we must seamlessly interpolate between the estimated parameters $\{\omega_{l-1}\}$ and $\{\omega_l, \phi_l\}$, using cubic interpolation of Section 3.4.1. Since there were no quantized and transmitted phases in frame $l-1$, we let $\phi_{l-1} = \theta_{l-1}(S)$, at the last sample of the instantaneous phase of that frame. In frame l, cubic interpolation is performed between $\{\omega_l, \phi_l\}$ and $\{\omega_{l+1}, \phi_{l+1}\}$. But, $\omega_{l+1} = \omega_l$, and ϕ^{l+1} can be derived from $\{\omega^l, \phi^l, S\}$, as was shown in Equation (3.4). Therefore, only the phase parameters, ϕ_r^l, for r=$(1, 2, \ldots, R)$ of each transient onset detected are required.

To graphically describe this scenario, see Figure 3.13. Each frame is 1024 samples long, and the frames $l-1$ and l are shown. That is, the transient begins at t=1024 samples, or the beginning of frame l. Over the length of frame $l-1$, the sinusoid in the middle plot slowly and smoothly becomes phase-locked to the sinusoid in the top figure. The bottom plot shows the error between the two higher plots. A similar algorithm is performed at the end of the transient region to ensure that the ramped-on sinusoids will be phase matched to the transient being ramped-off.

Due to the phase interpolation between the previous phaseless reconstruction phase $\theta_{l-1}(S)$ and the estimated phase of ϕ_l, there might be some slight frequency modulation artifacts. But due to the fact that this phase interpolation occurs precisely before a loud attack transient, any frequency modulation artifacts will mostly be masked. Several tests were performed using worst case synthetic examples of a single low frequency sinusoid whose phase is interpolated over the frame just before an

inserted transient. Transients were spliced in from other encoded audio sources, and scaled to be 3 dB louder than the sinusoid (a reasonable approximation in practice). After comparing this synthetic example with another low frequency sinusoid (which had no artificial phase interpolation) that was also interrupted by transients, very slight differences could be heard in informal listening tests. But, during natural audio input signals, no frequency modulation artifacts have been detected. The alternative to possible, slight frequency modulation artifacts is much worse amplitude discontinuities (or *clicks*) due to mismatched phase alignments between the sines and the transform-coded transient. If in the future, artifacts somehow become noticeable in yet untested audio input material, the phase interpolation procedure could occur over more frames than just one. By lengthening the time frame over which the phase is interpolated, the frequency modulation artifacts are reduced further.

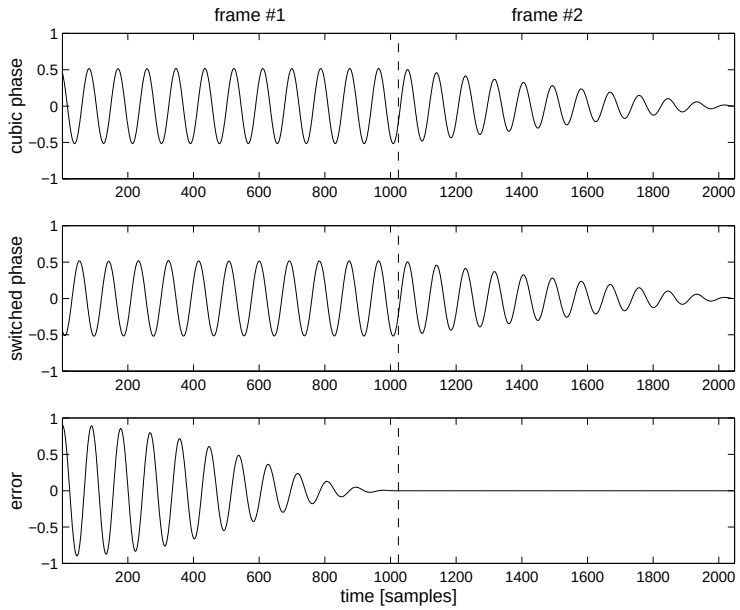

Fig. 3.13. The top signal shows a signal synthesized with phase parameters, and the phase is interpolated between frame boundaries using a cubic polynomial interpolation function of McAulay and Quatieri (1986*a*). The middle signal is synthesized using no explicit phase information except at the transient boundary, which is at time = 1024 samples. The initial phase is random, and is otherwise interpolated using the switched method of Section 3.4.3. The above signals contain two frames, each 1024 samples long. Frame #1 shows the middle signal slowly becoming phase locked to the signal above. By the beginning of frame #2, the top two signals are phase locked. The bottom plot is the difference between the top two signals.

3.5 Multiresolution Sinusoidal Masking Thresholds

While the parameter estimation subsystem uses a relatively sophisticated analysis (Thomson, 1982) to locate sinusoids as discussed in Section 3.2, that analysis has no notion of psychoacoustic properties. While a sinusoid may seem statistically valid, it is not worthwhile keeping if it will be *masked*

by surrounding sinusoids. The goal of this section is to find the masking thresholds of the estimated sinusoids. The log magnitude [dB] distance of each sinusoid above or below the masking threshold will help determine if the sinusoid is kept in the representation, which will be discussed in Section 3.6. This magnitude distance will be referred to as the signal-to-masking ratio (SMR). If the sinusoid is kept, the SMR then determines how much the sinusoid will be quantized, which will be discussed in Section 3.7.

In psychoacoustic experiments, it has been shown that the signal-to-masking ratio (SMR) of pure sinusoids is much larger than that of white noise signals (Zwicker and Fastl, 1990). In the MPEG-AAC standard (Bosi *et al.*, 1997), a frequency region that contains only pure sinusoids has an SMR of 18 dB, while one having pure noise has an SMR of only 6 dB. These frequency regions are called *threshold calculation partitions* in the MPEG specification, but are approximations to the one-third Bark band scale (Zwicker and Fastl, 1990). These masking thresholds determine how much quantization noise can be injected into each frequency region while still being inaudible. In the case of pure sinusoids, the quantization noise must be 18 dB quieter than the original signal in that frequency region in order to be masked. Since the masking threshold for noise is only 6 dB, then it follows that fewer bits can be allocated to noisy regions than for sinusoidal regions, in order to keep the quantization noise masked. If the signal in the frequency region is somewhere in between pure noise and pure sinusoids, then the corresponding SMR lies between 6 and 18 dB. As was originally mentioned earlier in Section 1.1.3, the particular psychoacoustic model used in this work implements the model based on MPEG-AAC (Bosi *et al.*, 1997). But, this particular model is only an approximation to a psychoacoustic model, given that true masking thresholds and spreading functions vary as a function of amplitude and frequency (Zwicker and Fastl, 1990). Also, it is not straightforward to argue that these masking thresholds, measured based on experiments using pure sine tones and noise, can accurately reflect the masking caused by complex audio input signals.

To begin some graphical examples of psychoacoustic modeling, Figure 3.14 shows a time-domain segment of the decay portion of a guitar pluck and Figure 3.15 shows both the signal's original spectral energy and its spread energy. The spread energy models the natural excitation spreading along the basilar membrane in the cochlea. The masking threshold is applied to this *spread* energy instead of the original energy. The frequency axis on this and the next two figures have been normalized to the one-third Bark band scale.

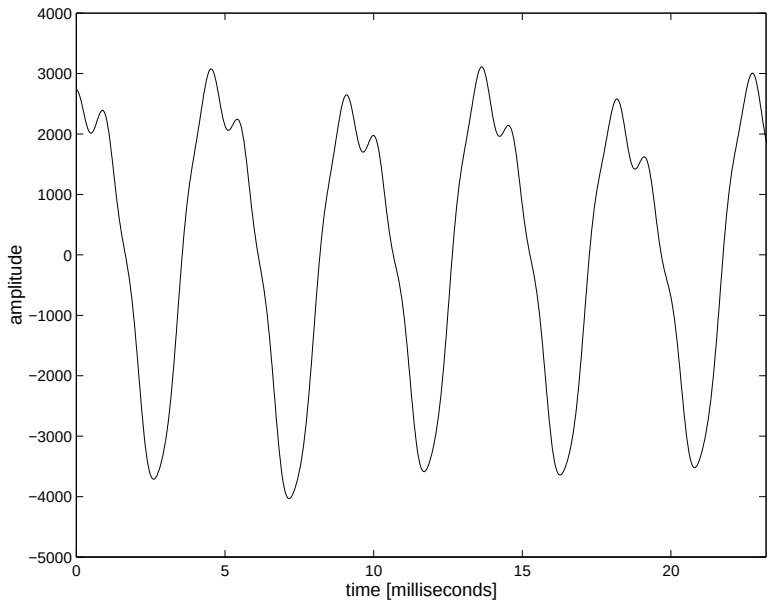

Fig. 3.14. A short segment of the synthesized sinusoids of a guitar pluck. This time-domain waveform is used for the spectral diagrams in Figures 3.15, 3.16, & 3.17

.

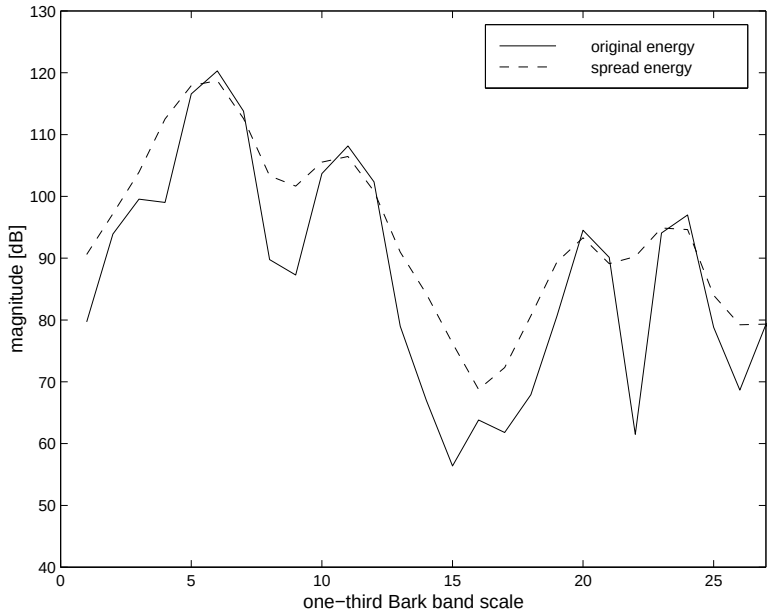

Fig. 3.15. The solid line shows the original magnitude spectrum of the guitar waveform of Figure 3.14. The magnitude has been converted to the one-third Bark band scale. In linear frequency, the displayed frequency range is from 0 to 1250 Hz. The dotted line shows the spread energy spectrum. This figure, along with Figures 3.16 and 3.17, only show this low frequency because it corresponds only to the lowest of the three multiresolution octaves.

The next step is to determine how tonal or noisy each one-third Bark band is. This will ultimately determine how far the masking threshold must be below the spread energy in order to be inaudible. In MPEG-AAC, this tonal/noise measure is based upon a linear predictor of FFT bin magnitude and phase. Each frame, an FFT is taken, and the complex FFT coefficients are converted into a magnitude $r(w)$ and a phase $f(w)$. The predicted magnitude $r_{pred}(w)$ and phase $f_{pred}(w)$ at FFT coefficient $w = 0 \ldots S-1$, and at frame l are as follows:

$$r_{pred}(w) = 2r(l-1) - r(l-2) \tag{3.10}$$

$$f_{pred}(w) = 2f(l-1) - f(l-2) \tag{3.11}$$

The *unpredictability measure*, $c(w)$, is the normalized Euclidean distance between the two complex vectors $\{r(w), f(w)\}$ and $\{r_{pred}(w), f_{pred}(w)\}$:

$$c(w) = \frac{\sqrt{[r(w)\cos(f(w)) - r_{pred}(w)\cos(f_{pred}(w))]^2 + [r(w)\sin(f(w)) - r_{pred}(w)\sin(f_{pred}(w))]^2}}{r(w) + |r_{pred}(w)|} \tag{3.12}$$

If this $c(w)$ measure is near zero, then the predictor is close to the original magnitude and phase, and therefore the FFT bin seems tonal. The process between computing $c(w)$ in each FFT bin to computing a tonality measure in each one-third Bark band is lengthy and complex, and can be best viewed in MPEG specifications (ISE/IEC JTC 1/SC 29/WG 11, 1993). But it suffices to say that this linear predictor is the major factor determining the tonality. As a graphical example, view Figure 3.16. It shows the signal-to-masking (SMR) ratio, based on the tonality measure. For example, at one-third Bark band #6, there is a spectral peak. The tonality measure has determined that this spectral region is tonal, and therefore has a SMR of 18 dB. That is, quantization noise must be quieter than 18 dB below the *spread* energy previously shown in Figure 3.15 to be masked and inaudible. But, at band #16, there are no spectral peaks, and the tonality measure determines that this region is more noisy than tonal. Therefore, it designates a smaller 8 dB SMR.

Once the masking threshold distance is computed, it is applied to the spread energy, as can be seen in Figure 3.17. The solid top line shows the spread energy, while the lower dotted line shows the masking threshold, which is simply the spread energy lowered by the masking threshold distance shown in Figure 3.16. Also shown as vertical stems are the estimated sinusoids. The signal-to-masking ratio (SMR) can now be computed as the log magnitude distance between the sinusoidal peak and the masking threshold in its corresponding third-octave Bark band. The four loudest sinusoidal peaks, in bands $\{6, 11, 20, 24\}$, are clearly above the masking threshold and will be audible. Some of the softest sinusoids, such as those in bands $\{1, 13, 14, 19\}$, are below the masking threshold and will most likely not ever be heard due to masking from its louder neighbors in frequency. Therefore, with negative SMRs, it will be safe to eliminate these sinusoids. The spectral peaks in bands $\{3, 16\}$ are close to the masking threshold, so it is difficult to make a decision based only on this frame. As will be discussed in the next two sections, not only will the current masking

thresholds be investigated, but the masking thresholds across an entire sinusoidal trajectory over time.

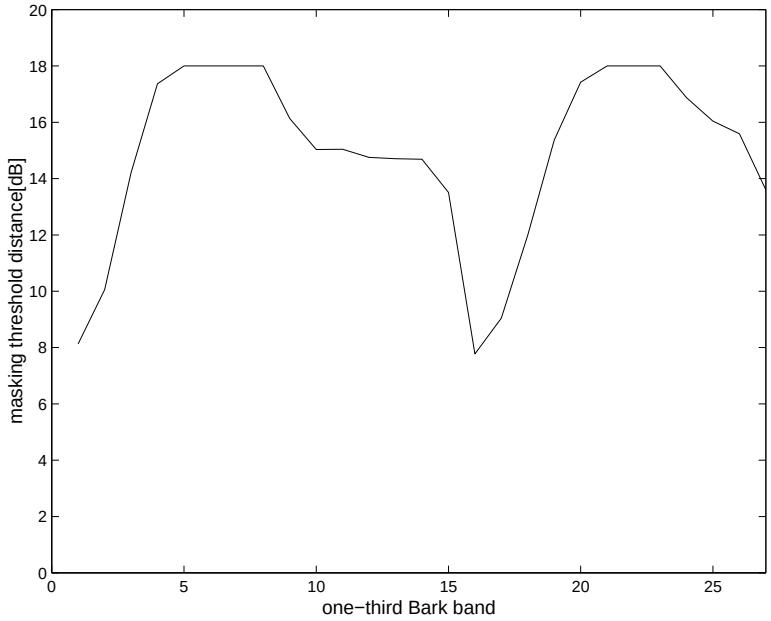

Fig. 3.16. This figure shows the results of the tonality measure as computed in MPEG-AAC. The vertical axis shows the SMR, which is how much the quantization noise must be quieter than the original signal in order to be inaudible. For example, between bands 5 through 8, the tonality measure deemed the frequency region of bands 5 through 8 to be tonal, and thus gave an 18 dB masking threshold distance. For a noisy region, around band 16, the masking threshold distance is only 8 dB.

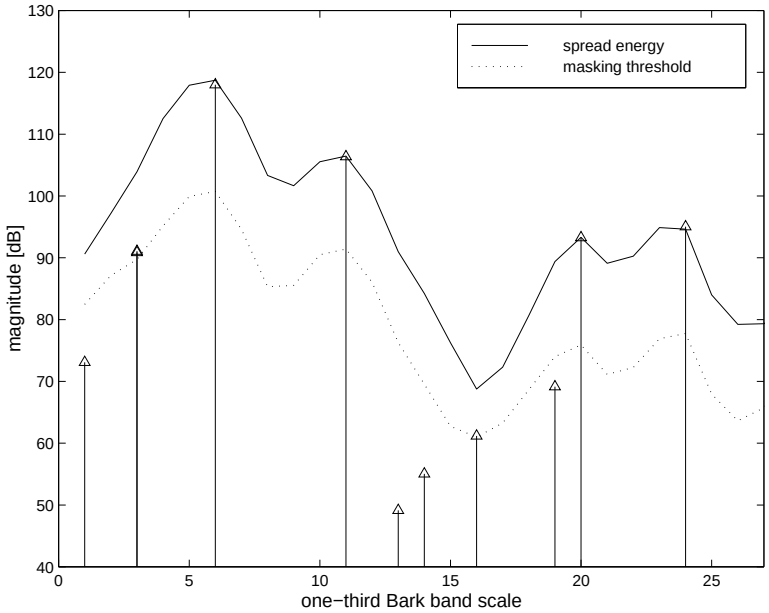

Fig. 3.17. This figure shows the signals spread energy (top) versus its masking threshold (below). The vertical peaks mark where sinusoids were synthesized. If a sinusoid is well above the dotted masking threshold, it will *not* be masked, and thus audible; *i.e.* peaks at bands 6, 11, 20, and 24. But, sinusoidal peaks below the masking threshold will be *masked*, and thus inaudible; *i.e.* peaks at bands 1, 13, 14, and 19.

In previous sinusoidal modeling audio compression systems, the masking thresholds were assumed to all be due to pure sinusoids (Ali, 1996). Therefore, the SMR is always a conservative, large distance. But, this assumes that all sinusoids estimated using the statistical methods are true, tonal sinusoids. After some informal listening tests, this was found *not* to be true. Using the same parameter estimation methods, and then listening to only the sinusoids with small or negative NMR, the synthesized audio certainly did not sound like tones. It sounded like a noise process placed through a flanger. Since these sinusoids sound *noisy*, they should not have masking thresholds as if they were tonal; they should have a smaller SMR. Thus, the psychoacoustic model from MPEG-AAC, with its own tonality measure, is used to double-check the validity of the sinusoids detected from the parameter estimation techniques by computing its own masking threshold. As shown earlier in Figure 3.4, all processing on sinusoids is performed separately on each octave band. Due to the fact that each octave has a different frame rate (Figure 3.9), each octave must perform psychoacoustic modeling separately, using a different frame length of 256, 512, or 1024 points.

3.6 Trajectory Selection

The goal of this section is to choose which sinusoidal parameters and trajectories to keep and which to discard. It is most efficient to represent only stable, tonal sinusoids with sinusoidal parameters. Very few signals can be well represented by just sinusoids. The remainder of the signal can be modeled well as either transients or a colored noise process. Because the region of signal described in this chapter has been selected as a non-transient region, the signal is represented as a sum of sines plus noise. It is possible to represent noise as a sum of many, many sinusoids; but, this would require a large number of bits, and time-scale modifications on these parameters would not sound as good. Therefore, only stable sinusoidal parameters and trajectories are selected to represent the stable, tonal segments of the non-transient parts of the signal. The residual between the original signal and these sinusoids will be represented by a Bark-band noise process, to be described later in Chapter 5.

There are two steps by which sinusoidal parameters and their trajectories must pass before being deemed *stable sinusoids*:

- Individual sinusoidal parameter triads having their SMR greater than SMR_{min}
- Sinusoidal trajectories having sufficiently high time-averaged SMR and length

The first step is necessary to eliminate any extremely quiet sinusoids. These sinusoids may have been estimation errors due to finding sidelobes of spectral peaks, or spectral peaks that happen to be much quieter than their neighbors. Either way, these sinusoidal parameter triads (containing a single frequency, amplitude, and phase) are so far below the masking threshold that they can easily be eliminated, and the difference will not be audible. On average, 10% of the parameters can be eliminated in this step.

The second selection step is more complex, and it is based on trajectory length *and* the trajectory's time-averaged SMR. Sinusoidal parameter triads are no longer considered as individual entities to be kept or eliminated. The entire trajectory will now be kept or eliminated. Certain sinusoidal parameter triads may lie below the masking threshold, but because its time-average SMR is greater than a certain threshold, the trajectory will be kept. If all sinusoidal parameter triads below the masking threshold (SMR ≤ 0) in a given trajectory were eliminated, the resulting trajectory would be repeatedly ramped on and off, and could have a displeasing audible result. Other sinusoidal modeling quantization schemes (Ali, 1996; Edler *et al.*, 1996) decide to keep sinusoidal triads based only on the SMR information in a given frame; not by looking across the entire sinusoidal trajectory. Because of parameter quantization, it is desirable to represent fewer, longer trajectories than a greater number of shorter trajectories. These bitrate considerations will be discussed in the following Section 3.7.

The overall strategy in determining whether or not to keep a trajectory is twofold: Do not keep the trajectory if it will not be heard (due to masking) and do not keep the trajectory if it is believed to be modeling noise. Because of these two constraints, both the SMR and the trajectory length are

used. A trajectory will not be heard if its *time-averaged* SMR is below a certain threshold, and it is believed to be noise-like if the trajectory is sufficiently short. From the tracking algorithms described in Section 3.3, noisy-signals should not represented with long tracks. After much experimentation, it was found that a joint metric works best: If a trajectory is very short, its average SMR needs to be considerably greater than zero in order to be kept. If the trajectory is longer, then the average SMR can be slightly lower and still be kept. As can be seen in Figure 3.18 trajectories that live in the upper right of the {average SMR,trajectory length} space are kept. Trajectories that have a low average NMR *and* a short trajectory will be discarded. Any trajectory longer than 46 msec is kept, regardless of its time-averaged SMR, which is always greater than SMR_{min}. If the tracking found the trajectory to have lived for such a relatively long amount of time, then it must represent a stable sinusoid. Due to the multiresolution nature of the sinusoids, trajectories in the lowest octave can have lengths in multiples of 23 msec., the middle octave can have lengths in multiples of 11.5 msec., and the top octave can lengths in multiples of 5.75 milliseconds.

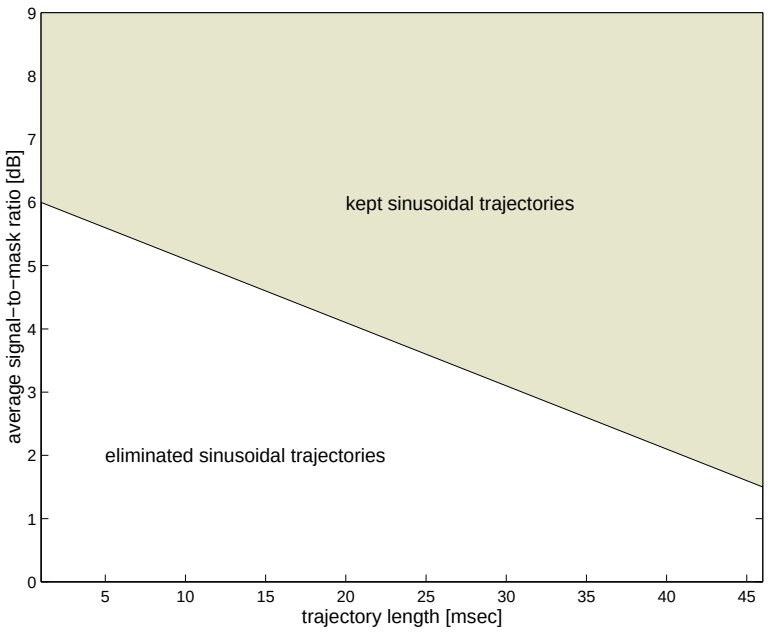

Fig. 3.18. This figure shows which sinusoidal trajectories will be kept and which will be eliminated (whose energy will later be modeled by a noise model in Chapter 5). Keeping a trajectory depends on both its trajectory length and the trajectory's time-averaged SMR. If a combination of these two parameters lie in the lower left portion of the above space, then the trajectory is eliminated from the sinusoidal representation. Otherwise, the trajectory is kept, quantized, and transmitted.

A general trend of the sinusoidal parameters returned from the original parameter estimation reveals that the longer the trajectory, the greater the average SMR. This trend can be seen in the solid lower plot in Figure 3.19. The shown statistics represent sinusoidal analysis in the top octave of eight seconds of music from Mozart's *Le Nozze di Figaro*. The number of trajectories at a given length exponentially decreases as the length increases, which can be seen in the upper solid plot in Figure 3.20. After eliminating the trajectories that are both too short and have too low of SMR, the statistics greatly change. In Figure 3.19, the top dotted line shows the new average trajectory SMR as a function of trajectory length after eliminating the undesirable trajectories. Notice that the only short trajectories that survive the elimination process are those with a very high average SMR. In Figure 3.19, the lower dotted line shows how the histogram of trajectory number versus length greatly changes after eliminating unwanted trajectories. For the shorter lengths, almost 7 out of every 8 original trajectories are eliminated. There are still short trajectories remaining, but they are deemed perceptually important. With far fewer trajectories, the bitrates now drastically diminish. Using the quantization techniques to be discussed in the following Section 3.7, removing these short and low SMR trajectories reduced the bitrate by 43% on average from the original set of sinusoids derived from the parameter estimation.

The alternative to this process in Serra (1989) was to set a user-defined parameter to *clean* out all parameters shorter than a given length. But, this blind process makes no distinction between perceptually important short and long trajectories.

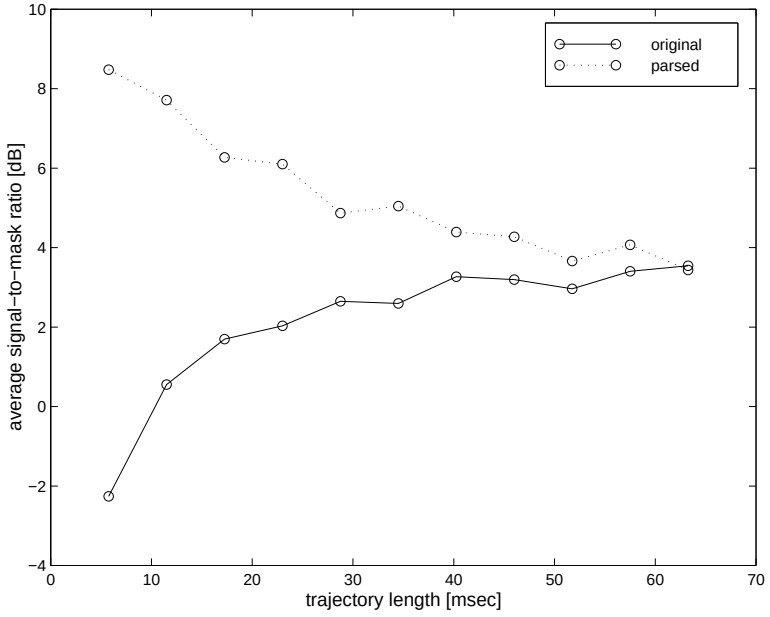

Fig. 3.19. The lower solid plot shows the time-averaged SMR of sinusoid trajectories of eight seconds of the top analysis octave of *Le Nozze di Figaro*. The general trend of the data is that the shorter the trajectory, the lower the time-averaged SMR. This would suggest that parameters modeling noise-like signals could not be tracked from frame to frame, and thus has a short trajectory length. The longer the trajectory, the more stable the sinusoids, and thus the higher the average SMR. As was discussed in Section 3.5, the assumed SMR for sinusoids is 18 dB, while the SMR for noise is only 6 dB. The upper dotted plot shows the statistics of the trajectories after the short and low SMR ones have been eliminated, as shown in Figure 3.18. Notice that the shorter trajectories that remain have a much higher average SMR. Even though they are short-lived, they are deemed perceptually important.

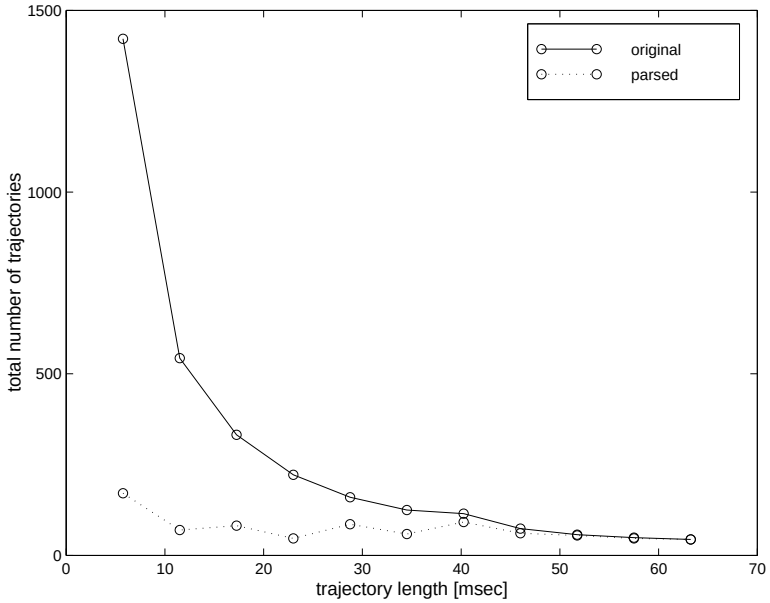

Fig. 3.20. The top solid line shows the histogram of original sinusoidal trajectory lengths for the top analyzed octave of eight seconds of music, corresponding to the data in Figure 3.19. The lower, dotted plot, shows the histogram of trajectory lengths after pruning the data as described in Figure 3.18. The majority of the short-lived trajectories have been eliminated; the only ones remaining are those with high average SMR. As the trajectories get longer, more of the original trajectories are kept.

3.6.1 Sinusoidal Residual

Now that many of the shorter and low SMR trajectories have been eliminated, another model must represent the remaining energy. After the remaining sinusoidal trajectories are quantized, the synthesized signal is subtracted from the original signal and is represented by a Bark-band noise model, which will be discussed later in Chapter 5. As a graphical example, see the residual in the time domain (Figure 3.21) and in the frequency domain (Figure 3.22). This sound example shown is the same segment used in the previous psychoacoustic modeling examples for Figures 3.14 through 3.17. For most signals not during transients, a sines + noise model is sufficient. As will be discussed in the next two chapters, the added bitrate due to the residual noise model is small relative to the bitrate saved by not encoding the shorter and low SMR trajectories.

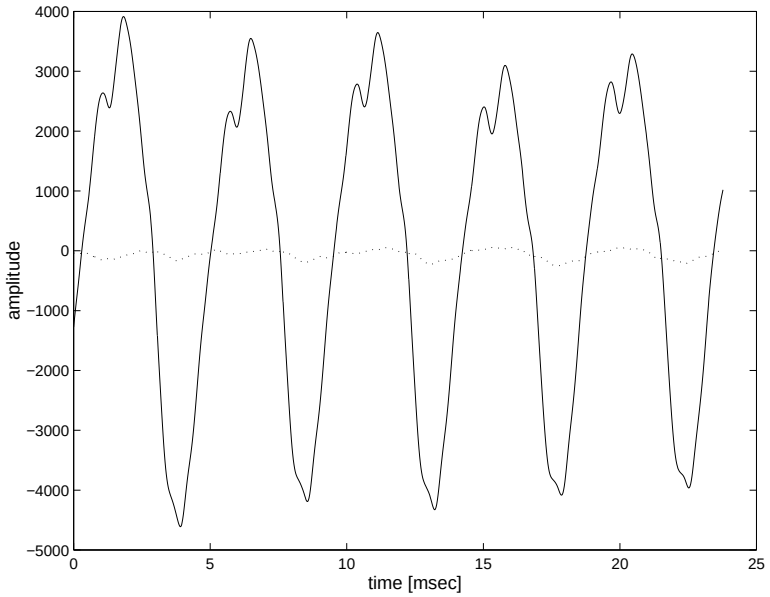

Fig. 3.21. The solid line is the original guitar pluck segment shown in Figure 3.14 and was used in the succeeding psychoacoustic modeling examples. The lower-gain dotted line is the residual between the original signal and the multiresolution synthesized sinusoids.

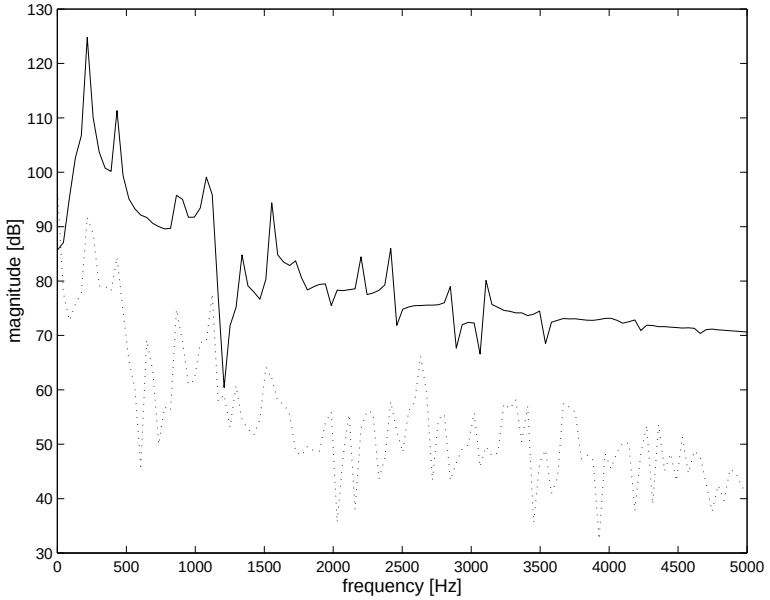

Fig. 3.22. The figure shows the magnitude spectra of the original guitar pluck (solid) and its residual (dotted) shown in Figure 3.21.

3.7 Trajectory Quantization

The previous section described at length which sinusoidal trajectories to keep; this section will describe how to efficiently quantize the remaining trajectories. The quantization occurs in multiple stages, that contain both lossless and lossy compression:

- Quantize each amplitude and frequency to their just-noticeable differences (JND).
- Reduce the temporal resolution of trajectories with a low average signal-to-mask (SMR) ratio.
- Quantize each trajectory by entropy coding the temporal and frequency changes.

3.7.1 Just Noticeable Difference Quantization

The first step in the quantization process is to quantize the amplitude and frequency sinusoidal parameters to a approximated, just noticeable difference (JND) scale. By quantizing these parameters to their approximated respective JND scales, the values are not identical to the original parameters; but for most music (with exception of only synthetic cases, such as single-sinusoid tests), they are perceptually identical. This was verified in informal listening tests between the original and quantized parameters. Therefore, it's mathematically lossy, but almost always perceptually lossless quantization.

If the sinusoidal amplitudes and frequencies were quantized strictly according their respective JND scales, then too many bits would be required. Also, psychoacoustic masking occurs from many simultaneous sinusoids and a residual noise model (to be discussed later in Section 5.2), which lowers the amount of quantization resolution needed. Therefore, the amplitudes are scalar quantized to the closest multiple of 1.5 dB, which is several times less resolution than the JND would dictate for isolated test tones. Ideally, the quantization of the sinusoidal amplitudes would be adaptively matched with its psychoacoustic masking threshold, which remains a possibility for future work. The frequencies from 0 to 500 Hz are quantized to a scalar uniform scale, with quantized increments of 3 Hz, which is below the JND for frequency. Between 500 and 5000 Hz, frequency is quantized on a logarithmic scale, consistent with the findings from psychoacoustics (Zwicker and Fastl, 1990). The frequency parameters are quantized to every 10 cents, which is one-tenth the distance between two keys on a piano. While this is about half of the resolution that the frequency JND dictates, no differences could be heard in informal listening tests between quantized and non-quantized frequencies for real-world audio inputs (not synthetic test cases).

3.7.2 Trajectory Smoothing

For certain sinusoidal trajectories, the time resolution (hop size) may be too high. That is, it would not be perceptible if these trajectories' time resolution were worsened, and thus the trajectories

themselves were smoothed. In this system, trajectories with a time-averaged SMR of less than 6 dB are lowpass filtered, and then downsampled by two before quantization and transmission.

In the previous Section 3.6, it was shown that trajectories with less than a certain average SMR and shorter than a certain length were eliminated from the sinusoidal representation. Therefore, the remaining low SMR trajectories last longer in time, and were kept because it was believed that there were perceptually important. But, because these trajectories' average SMR are low, they are not *as* perceptually important as the trajectories with a high SMR. To further reduce the sinusoidal bitrate, these longer, low SMR trajectories are represented with half of the time resolution of their higher SMR counterparts.

Let each amplitude and frequency trajectory be considered as time series $a(n), f(n)$, for $n = 1 \ldots N$, for length N trajectories. Then let $\hat{a}(n), \hat{f}(n)$ be the reconstructed trajectories. The goal is to generate the reconstructed trajectories from only 50% of the original trajectories. Traditionally, one would accomplish this by lowpass filtering $\{a(n), f(n)\}$, downsampling by two, upsampling by two, and then lowpass filtering again to generate $\{\hat{a}(n), \hat{f}(n)\}$. But, convolving any signal with a filter increases the length of the results to the sum of the lengths of the signal and its filter (minus one). To remedy this, short two tap FIR filters are used, and the tails are cutoff, such that only $N/2$ interpolated coefficients are transmitted.

View Figure 3.23 for a demonstration of actual smoothed trajectories. The top figure shows the original (solid) and reconstructed (dashed) amplitude trajectory of length 80 milliseconds. All these parameters have previously been quantized to the approximated JND amplitude scale. The lower figure shows the original versus reconstructed frequency trajectories. In this example, the original trajectory was 15 elements long, but only 8 coefficients had to be quantized and transmitted in order to reconstruct the smoothed trajectory.

In this system, all remaining trajectories with a time-averaged SMR below 6 dB are smoothed by this process. In the suite of tested audio signals, this represented between 50 to 70% of the trajectories. After informal listening tests, no pre-echo or smearing artifacts were heard due to the smoothing algorithm. If the cutoff of 6 dB SMR were significantly raised, then artifacts began to appear. Due to the trajectory smoothing process, the total sinusoidal bitrate decreased an additional 25 to 30% from the previous set of already parsed sinusoidal parameters due to short and low SMR trajectories.

3.7.3 Time-Differential Quantization

After all the trajectories are quantized to the approximated JND scale, and the ones with low SMR have been downsampled and smoothed, all the trajectories are now further time-differentially scalar quantized and then entropy encoded. Using time-differential scalar quantization of the trajectories is also used by several other sinusoidal coders (Ali, 1996; Hamdy *et al.*, 1996; Edler *et al.*, 1996). Because the tracking, as described in Section 3.3, limits the variance of amplitude and frequency

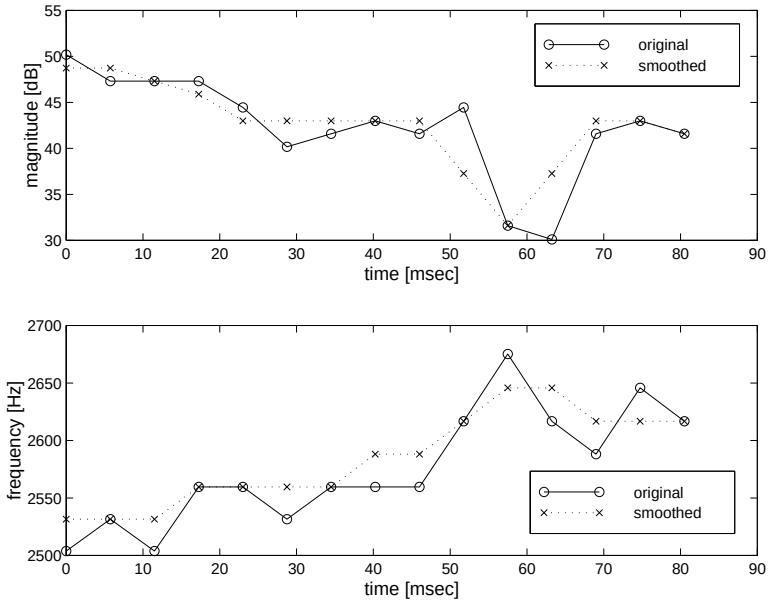

Fig. 3.23. This figure shows how certain trajectories are smoothed. The top graph shows the original (solid) versus the smoothed (dashed) magnitude trajectories, and the lower graph shows the original versus smoothed frequency trajectory.

from frame to frame, an obvious coding gain is achieved by quantizing only these frame to frame differences. These differences are then Huffman coded, using separate codebooks for each octave. There is no codebook choice overhead because the frequency values dictate which octave the trajectory resides in, and thus determines which Huffman codebook to use. On average, the Huffman encoded differential amplitudes take 4 bits/parameter. The Huffman encoded differential frequencies only consume 2.5 bits/parameter (bpp) in the top two octaves, and 3.2 bpp in the lowest octave. This high bitrate in the lowest octave is due to the higher quantized frequency resolution in that octave, which is psychoacoustically required (Zwicker and Fastl, 1990). Each initial parameter in the trajectory is encoded absolutely, with no differential coding, and then Huffman encoded. On average the initial amplitude requires 4.7 bpp, while the initial frequencies require 5.4 bpp in the top two octaves and 6.7 bpp in the lowest octave.

3.7.4 Sinusoidal Bitrate Results

For a summary of the bitrate improvements each quantization technique employs, see Figure 3.24. Each of these methods progressively lowers the total bitrate, assuming time-differential scalar quantization followed by entropy coding. The four techniques used are as follows:

- Quantize every parameter to its JND scale in amplitude and frequency (*method #1).*
- Eliminate individual parameter triads out of their respective trajectories if its SMR is sufficiently low (*method #2*).
- Eliminate trajectories whose time-averaged SMR is sufficiently low and the trajectory length is sufficiently short (*method #3*).
- Downsample and smooth any remaining longer trajectories whose time-averaged SMR is sufficiently low (*method #4*).

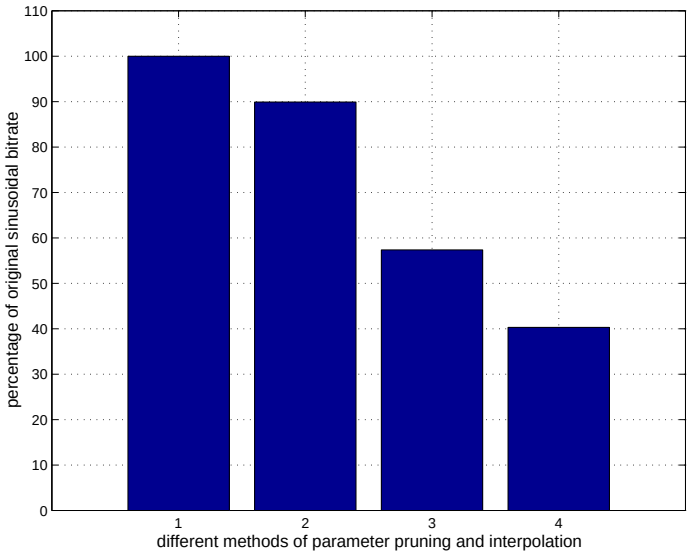

Fig. 3.24. This figure shows how each step of the quantization process reduces the bitrate required to represent the sinusoidal component of the signal. Rates are shown in percentage of bitrates of the original parameters returned from the statistical parameter estimation process, and are averaged over several pieced of audio input. Method #1 is the reference bitrate, where no sinusoidal parameters were eliminated from the set of originally estimated parameters, but are quantized to their JND scales. Method #2, which eliminates all individual parameters with less than -6 dB SMR, only uses 90% of the original bitrate. Method #3, which eliminates all sinusoidal trajectories that are both too short and a sufficiently low average SMR, uses roughly 57% of the original bitrate. The final method #4, which downsample and smoothes longer trajectories with sufficiently low SMR, lowers the bitrate further to only 40% of the initial bitrate.

To summarize how many bits are spent on amplitude, frequency, phase, and side information, see the following table. The amplitude and frequency data rates were described in detail in the previous

Section 3.7. The phase bitrate is due to quantizing only the phase parameters at the transient boundaries to 6 bits (uniformly from π to $-\pi$). As was discussed in Section 3.4.3, only these boundary phases are necessary to phase-lock these cross-faded sinusoids to the overlapping region of the transients. The side information is mostly comprised of encoding the length of trajectory. Each frame of data contains all trajectories that *begin* at that time. Therefore, a packet contains all the sinusoidal parameter triads for the entire length of each of the enclosed trajectories.

Bitrates for Sinusoidal Modeling

audio	all	amplitude	%	frequency	%	phase	%	side info	%
cantaloop	9.1 kbps	4.2 kbps	45.6	3.4 kbps	37.8	0.6 kbps	7.0	0.9 kbps	9.6
rob base	8.0 kbps	3.6 kbps	45.2	3.2 kbps	40.1	0.4 kbps	5.2	0.8 kbps	9.6
dances	8.9 kbps	3.9 kbps	44.3	3.4 kbps	38.9	0.6 kbps	7.1	0.9 kbps	9.7
figaro	9.4 kbps	4.4 kbps	46.6	3.8 kbps	40.3	0.3 kbps	3.3	0.9 kbps	9.7
guitar	5.9 kbps	2.9 kbps	48.8	2.2 kbps	38.2	0.2 kbps	3.2	0.6 kbps	9.7
take five	8.0 kbps	4.1 kbps	50.6	3.1 kbps	38.4	0.1 kbps	1.5	0.8 kbps	9.5

As can be seen from the above table, the amplitude and frequency trajectories account for the majority of the allocated bits. The phase bit allocation varies from signal to signal, mostly due to the number of transients detected per second in each signal. For every detected transient, phase information will be needed for sinusoids that are at its boundaries.

Through various methods outlined in this chapter, the amount of data for multiresolution sinusoidal modeling has been progressively lowered. But, the sinusoids make up only one of the three models necessary to fully represent the audio: sines + transients + noise. In the following two chapters, the representations and quantization methods for transients and noise models will be shown.

Chapter 4

Transform-Coded Transients

This chapter discusses the representation of transients in the presented sines+transients+noise system. Because of the perceptual importance of the attack transients in audio signals (Grey, 1975), transform coding techniques are used over the brief time-span of these attack transients. Transform coding has been well studied over the past ten years, and has been shown to be robust in representing most audio signals, transients included. Sinusoidal modeling (Chapter 3) and noise modeling (Chapter 5) are not able to model these transients well, due to limitations in their respective representations. Transform coding does have several drawbacks, though.

Firstly, transform coding requires a relatively large number of bits per second, when compared to sinusoidal modeling and noise modeling. While transform coding can achieve perceptually lossless quality on most audio sources, an achievement sinusoidal and noise modeling has yet to show, this quality comes at a high bit price. Secondly, it is currently not possible to perform modifications on transform-coded MDCT coefficients, directly in the transform-domain, such as time-scale modification. This is due to the fact that these coefficients are critically sampled, and thus any MDCT bin examined over time, is mostly white noise (except for the lowest frequency bins). Therefore, it is not straightforward to time-interpolate these MDCT coefficients, as would be needed in transform-domain time-scale modification.

As a result of these advantages and disadvantages, transform coding is used for only a very short amount of time just surrounding the attack transients. By using transform coding sparingly, the overall bit rates are kept low. By using transform coding over small regions of time-frequency, these transients can be *translated* to new times for the purposes of time-scale modification, while still blending with the *stretched* representations of time-scaled sinusoids and noise. Details of the compressed-domain modification processes will be discussed in more detail in Section 6.

In this chapter, a brief history of the currently used systems of audio compression will be discussed in Section 4.1. Afterwards, in Section 4.2, a simplified transform coding algorithm used for only transient modeling will be presented, along with various methods for quantization and bitrate

reductions.

4.1 Evolution of Transform Coding

Transform coding of audio has made great strides since the earliest algorithms (Johnston and Crochiere, 1979; Brandenburg *et al.*, 1982; Brandenburg, 1987; Theile *et al.*, 1988; Johnston, 1988*b*). Compression rates of 4:1 achieving perceptually lossless quality was once remarkable. Today, with the most recent MPEG AAC algorithms, compression rates of 12:1 are now possible with the same quality (Soulodre *et al.*, 1998). This section will briefly describe, in approximate chronological order, some of the published and widely used audio data compression algorithms used today. Because the transform coder used in this thesis is mono (single channel), no stereo transform coding techniques will be discussed. For references on stereo techniques, see Johnston *et al.* (1996*a*); Johnston and Ferreira (1992). Afterwards, in Section 4.1.8, a short quality comparison among the codecs will be presented.

If the last ten years of audio coding work has taught researchers anything, it is that coder quality is always increasing at lower and lower bitrates. This thesis will likely be dated by the transform coding compression systems mentioned in this section, since very few of them will last far into the future. But for historical purposes, these systems will show the current state of the art of audio coding in 1998. People have begun to wonder if in the future, audio compression algorithms will be not stagnant standards, but rather be transmitted as always-updated software along with the compressed audio bitstream. If this were to become true, compression algorithms will continue to innovate, since the barrier to entry of making a new compression standard (currently very difficult) would disappear.

The basic introduction of transform coding was previously discussed in Section 1.1.3. Most all transform coders contain a core of the same three building blocks, which can be pictured as in Figure 4.1.

- filter bank
- psychoacoustic model
- quantizer

Each of the following compression systems to be discussed contributed its own new developments in these core modules, but the basic concepts are the same: A time-domain signal is transformed into some short-time frequency domain representation, a separate psychoacoustic model is performed on the short-time segment to decide where quantization noise can be masked and inaudible, and then a quantizer performs the data compression in the frequency domain. In this manner, quantization error is present, but it is *shaped* in time and frequency such that it is imperceptible.

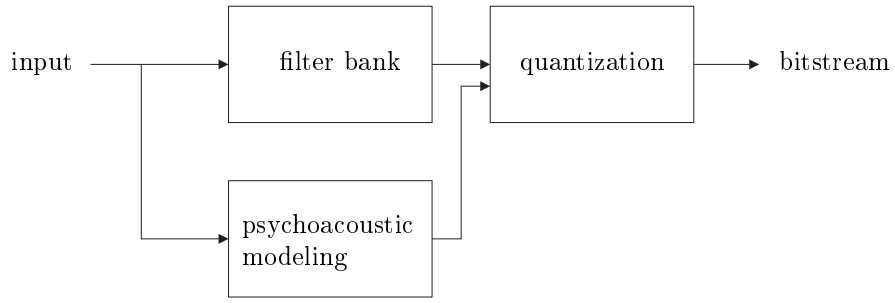

Fig. 4.1. A basic transform encoding system

4.1.1 MPEG 1 & 2; Layer I,II

MPEG-1 and MPEG-2 were the first widely used, standardized, transform-based audio compression algorithms (Brandenburg and Stoll, 1994; Stoll, 1996). Both MPEG-1 and MPEG-2 have three layers each, and each layer defines separate audio compression algorithms differing in complexity and coding efficiency. MPEG-1 was designed for two channel stereo signals at sampling rates of 32, 44.1, and 48 kHz; while MPEG-2 was designed to be used up to seven channels, plus an optional low frequency effects channel (known as the *0.1 channel* in today's surround sound systems). In addition, MPEG-2 was designed to work for a wider range of sampling rates, spanning 16, 22.05, 24, 32, 44.1, and 48 kHz. MPEG-2 is also fully backward compatible towards MPEG-1. Between MPEG-1 and MPEG-2, the basic algorithms of the individual layers do not change much. For example, MPEG-1 Layer III is not significantly different than MPEG-2 Layer III. In the following discussion, each of the layers will be described, without mentioning if they are part of MPEG-1 or MPEG-2.

Layer I,II Filter Bank

Layers I and II are significantly lower complexity and lower coding delay than Layer III, but their coding qualities are not as good as Layer III (Soulodre *et al.*, 1998). Layers I,II are modeled after MUSICAM (Dehéry *et al.*, 1991), where Layer I is a simplified version and Layer II is nearly identical to MUSICAM. The filter bank used in these layers (along with Layer III) is a modulated lapped transform, with 32 subbands (Rothweiler, 1983). Each bandpass filter is of length 512, with a hop size of 32 samples each frame. With such a long filter and relatively few subbands, a high sidelobe rejection of 96 dB is attained.

In hindsight, having only 32 subbands was one of the reasons the coding gain of Layers I,II was not that high. Layer III and MPEG AAC increased the number of subbands, and thus the

quality also increased. More about these systems will soon be discussed. Quantization occurs on the subband resolution, but this resolution of approximately 650 Hz per subband is too low. For example, let an input signal be purely tonal with harmonic peaks. One would only desire to spend bits in the frequency regions just surrounding the peaks, and spend hardly any bits in the filter bank magnitude valleys. With such a coarse frequency representation, as in Layers I & II, each subband would have to be allocated bits to well represent each harmonic.

Layer I,II Psychoacoustic Model

In Layers I and II, MPEG psychoacoustic model I is used. It computes a signal-to-mask ratio (SMR) for each of the 32 subbands that dictates how much quantization noise can injected without being audible. Psychoacoustic model I is a less complex and faster model than model II, which is used in Layer III. For any specific details of these layers, including the psychoacoustic models, see ISE/IEC JTC 1/SC 29/WG 11 (1993).

Layer I,II Quantization

For Layers I and II, the quantization schemes are mostly the same. For each subband, a frame is formed by grouping either 12 (Layer I) or 36 (Layer II) time-consecutive subband samples. A scalefactor for each group describes, with a 2 dB resolution, the maximum absolute value of these subband values. All the subband samples in the frame are normalized by this scalefactor. A bit allocation process iteratively determines how many bits to use in quantizing the samples in each subband frame, which depends on the SMR derived in the psychoacoustic model. The number of bits per subband roughly range from 0 bits (no energy transmitted, coarsest possible quantization) to 15 bits (very fine resolution).

Layer I was the system used in Philip's Digital Compact Cassette (DCC) in the early 1990's, but is currently no longer commercially available (Lokhoff, 1996). Layer II is currently used in point-to-point audio links, digital radio systems for EUREKA 147 (Schröder and Ratliff, 1996) and ASTRA Digital Radio, and direct satellite broadcasting of audio with video (Fielder, 1999).

4.1.2 MPEG 1 & 2; Layer III

The foundation for the algorithms of Layer III came from ASPEC (Brandenburg *et al.*, 1991). All of the three core building blocks of the filter bank, the psychoacoustic modeling, and the quantization are different from those of Layer I and II.

Layer III Filter Bank

The filter bank of Layer III is a hybrid, hierarchical filter bank. The first filter bank contains the same 32 subband structure used in Layers I,II. But as was discussed earlier, higher frequency

resolution was necessary to improve the coding gain. Therefore, each of the 32 subband signals are placed into individual 6 or 18 channel MDCT filter banks, thus increasing the number of frequency regions to 32 x 6 = 192 or 32 x 18 = 576. In the first case of only 192 frequency bins, improved temporal resolution was deemed more important than frequency resolution. This adaptive window switching would occur during attack transient times, and improved temporal resolution is needed to reduce pre-echo distortions.

Because the Layer III filter bank is actually a series connection of two lapped filter banks, there are complex aliasing conditions in the final 192 or 576 frequency values. The first filter bank is a pseudo-quadrature mirror filter bank (PQMF), whose prototype lowpass filter (window) is of length 512, and results in 32 real-valued frequency coefficients every frame. In the audio compression literature, this filter bank is also known as a polyphase quadrature filter bank (PQF) and is originally attributed to Rothweiler (1983). But according to Vaidyanathan (1993), the filter bank was first developed by Nussbaumer (1981). This filter bank is not perfect reconstruction, but the reconstruction error is approximately -96 dB, and thus well below audibility or quantization error. The second filter bank, which is cascaded with each of the 32 PQMF subbands, is a modified discrete cosine transform (MDCT), which was described in more detail in Section 1.1.2.

Because the stopband transition region of the 32 PQMF subbands is several MDCT bins wide, substantial cross-talk aliasing occurs. Therefore, if an input signal were a pure sinusoid near a PQMF bin boundary, the output spectra of 192 or 576 MDCT bands would have one major peak corresponding to the sinusoidal input. But, another peak, several MDCT bins away and only 10-15 dB lower, would be due to the cross-talk among the PQMF subbands being cascaded by higher frequency resolution MDCTs. Either of these two filter banks, on their own, do not have these cross-talk problems; only when cascaded do these problems arise. By using a butterfly cross-multiplication operation, these aliasing artifacts are reduced (Edler, 1992; Brandenburg *et al.*, 1992).

Layer III Psychoacoustic Model

The decision to switch between the improved frequency resolution mode of 576 channels or the improved temporal resolution mode of 192 channels is a function of the perceptual entropy (Johnston, 1988*a*), which is determined by the psychoacoustic model. Layer III uses MPEG psychoacoustic model II, which was described earlier in Section 3.5. This model, which is also used in MPEG AAC, is a much higher complexity model than psychoacoustic model I used in Layer I and II, but seems to obtain higher quality results. The perceptual entropy (PE), which is based on the total signal-to-mask ratio, is an estimated measure of how many bits are required to encode a frame of audio with perceptually lossless quality. Experimental data suggests that a large surge in the estimated bits required to encode a frame almost always coincides with a transient event that requires a window length switch (Brandenburg and Stoll, 1994).

Layer III Quantization

Quantization improvements introduced in Layer III are the bit reservoir, the joint rate-distortion quantization loops, and the use of lossless entropy coding. The bit reservoir allows the bitrate to be buffered between frames, resulting in a variable bitrate (VBR) system. This is different than the constant bitrate (CBR) techniques of Layer I,II and AC-3, which constrain every frame of compressed audio to have the same number of bits. Certain frames, like those with attack transients, require more bits to encode with perceptually lossless quality. But other frames, such as those with steady-state tones, require fewer bits to attain the same perceptual quality. In Layer III, a buffer called the bit reservoir, saves bits during frames that are simple to encode, and spends those extra bits when the material becomes more difficult. The decision on how many bits to spend is dependent upon the perceptual entropy (PE), as discussed earlier. The use of the bit reservoir enables Layer III to become a variable rate encoder, which in turn, allows for higher quality than the constant rate coders like Layer I,II and AC-3. The disadvantage of variable rate encoders is that latency is now increased, and the decoder must have additional memory. Also, for certain transmission media, only constant bitrate streams are desired.

Once it has been determined how many bits are to be allocated in a given frame, the joint rate/distortion quantization loops are begun. The goal of these quantization loops is to iterate towards a quantization solution that ensures:

- All quantization noise is quieter than the masking threshold across all frequencies
- The bitrate necessary to quantize the MDCT coefficients, along with all the scalefactors and side information, is less than the pre-allotted bit budget determined by the PE and the bit reservoir.

Initially, the MDCT coefficients are quantized using the same resolution over all scalefactor bands (sfbs), which are 49 non-uniform regions of frequency. The quantization resolution is made coarse enough uniformly across all sfbs until the bit budget is met. After this rate loop, each sfb is checked to see if its quantization noise is below the masking threshold. If any sfb does not meet this threshold, then that particular sfb's quantization resolution improves by a few dB. If after this distortion loop, the bitrate is still within budget *and* no sfb has quantization noise above its own masking threshold, the quantization loops are complete. Otherwise, the rate loops are again started, and all the sfbs are uniformly quantized coarser until the bit budget is met. Slowly, the algorithm will converge to a solution, although there is no guarantee that this solution is a global optimum. Typically, this takes 10-16 iterations through both loops (Gbur *et al.*, 1996). If the bitrate and quantization noise constraints cannot be met, there are several exit conditions to the iteration loops.

The MDCT quantization process contains both a lossy and a lossless stage. The initial lossy stage quantizes each coefficient on a logarithmic scale, according to the rate-distortion quantization loops as previously described. Afterwards, a lossless quantization stage uses both run-length coding (across

frequency) and Huffman coding to further reduce the bitrate of the quantized MDCT coefficients. The high frequency values are run length encoded to represent the long string of zeroes. The remaining coefficients are subdivided in up to three sections, and each section can use one of several available Huffman tables stored in the encoder and decoder.

Layer III is perhaps most well known today as mp3, the file extension used on MPEG Layer III audio files transmitted across the Internet. Layer III is also to be used in WorldSpace digital radio.

4.1.3 Sony ATRAC

Unlike MPEG-Audio, Sony's ATRAC algorithm is not an open standard (Tsutsui *et al.*, 1992). It stands for Adaptive Transform Acoustic Coding for MiniDisc, and it is the proprietary format for Sony's MiniDisc. The MiniDisc is a recordable, optical data-storage format for audio storage. A MiniDisc, while physically smaller than a CD, can store 74 minutes of audio through data compression of 5:1 using ATRAC.

Like MPEG-Audio, ATRAC uses filter banks, psychoacoustic models, and quantizers. But, the filter bank in ATRAC is significantly different. As can be seen in Figure 4.2, the input signal is first split into three octaves by using two stages of two-band, critically sampled, QMF analysis filter banks. The top octave has a sampling rate of $f_s/2$ with a bandwidth of 11-22 kHz, while the lower two bands have a sampling rate of $f_s/4$ with respective bandwidths of 0-5.5 kHz and 5.5-11 kHz. Each of these octave band signals are then fed into separate MDCT filter banks. The window lengths of the MDCTs get shorter during transient attacks, similar to the adaptive window switching of Layer III, in order to reduce pre-echo artifacts.

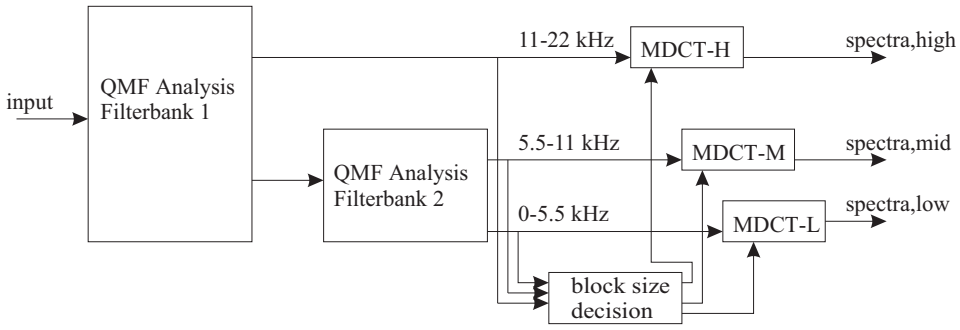

Fig. 4.2. The filter banks used in Sony's ATRAC compression algorithm

The input signal is also fed through a psychoacoustic model, in order to determine what regions of time-frequency the quantization noise can be injected and be perceptually inaudible. This information is sent to the MDCT coefficient quantization algorithm. The raw MDCT coefficients are grouped into block floating units (BFUs). Each BFU has a fixed number of coefficients, but BFU groups have different time-frequency resolutions depending upon octave. For example, in the low octave, each BFU spans 2.9 msec and a small frequency range. But in the top octave, each BFU

spans only 1.45 msec and a larger frequency range. When quantizing each BFU, a scalefactor is transmitted which determines the largest absolute value MDCT coefficient in that BFU. The other coefficients in the BFU are normalized by the scalefactor, similar to Layer I,II, and then quantized with b bits, which are determined by ATRAC's bit allocation algorithm.

4.1.4 Dolby AC-3

Dolby Laboratories' AC-3 (Fielder *et al.*, 1996) audio data compression algorithm is a proprietary system that has been chosen as the audio system for America's high definition television (HDTV) standard (U.S. Advanced Television Systems Committee, 1992) and North America's standard for digital versatile disc (DVD). It is also currently being used in many digital satellite and digital cable systems.

AC-3 uses only a single MDCT filter bank, as opposed to the three MDCT filter banks of ATRAC or the cascaded set of PQMF and MDCT filter banks of Layer III. The length of the filter bank window in AC-3 varies between 512 and 256 points, depending if an attack transient is present. By only having a single adaptive-length MDCT, the cross-talk aliasing problems of the iterated Layer III filter bank are avoided. The transient detector is a function that looks for surges in short-time energy, as opposed to the perceptual entropy measure of Layer III.

The window design choices for AC-3 are also different than those of Layer III. Layer III uses raised sine windows of differing lengths. When transitioning between short and long windows, the tails of the windows always overlap by 50% of the length of the shorter window. This method is similar to the method used for MPEG AAC, which can be seen graphically in the earlier Figure 1.5 in Section 1.1.2. But, AC-3 uses a Kaiser-Bessel window instead of a raised sine window in order to more closely match the psychoacoustic masking curve characteristics. One possible shortcoming of AC-3 is that the same shape window is used for either long or short blocks. When a short block is used, only half (256 points) of a Kaiser-Bessel window is used, and the other half is set to zero (Shlien, 1997). Therefore, two consecutive short windows do not overlap at all. The window switching algorithm still obeys the perfect reconstruction Equation (1.6), but the frequency response of a short window with either its right or left half set to zero is not optimal. Because these short windows are only used during attack transients, high quality is still obtained in spite of their poor frequency response. The relative rankings of AC-3 among the other coders will later be discussed in Section 4.1.8.

The quantization method for AC-3 is also much different than any of the other transform coders described in the section. AC-3 first encodes a coarse spectral envelope. This envelope has a 6 dB resolution in gain, so that the band gains become power-of-2 scalefactors for the bands. Scaling by powers of 2 is equivalent to bit shifting in hardware, and is thus computationally very efficient. The envelope is updated every single, dual, or four consecutive coefficients across frequency. In order to save bits, the envelope would only be updated every four coefficients, but quality would drop.

The spectral envelope can be updated in time as often as every frame or as seldom as every six frames. For steady-state tonal signals, a seldom updated spectral envelope would be sufficient, but an attack transient would necessitate a spectral envelope update every frame. The spectral envelope is quantized differentially across frequency, representing either $\pm 6dB, \pm 12dB$ or no change.

A parametric psychoacoustic model computes noise-to-mask ratios (NMR) for each coefficient depending on the spectral envelope, the desired data rate, and other encoder details. The MDCT coefficients, normalized by the spectral envelope, are now quantized according to the NMR from the parametric psychoacoustic model. An interesting feature of AC-3 is that the bit allocation and NMR information does *not* need to be transmitted to the decoder. Only the coarse spectral envelope and the quantized MDCT coefficients are sent. The identical NMRs are computed again at the decoder using only the spectral envelope information. Afterwards, the quantized MDCT coefficient bitstream can be decoded. When designing AC-3, the designers also recognized that better (and more computationally expensive) psychoacoustic models may someday arrive. With this in mind, the encoder may use more expensive, higher quality psychoacoustic models and bit allocation algorithms than the inexpensive decoder may have. But, the encoder can send a relatively small amount of auxiliary information dictating the difference in bit allocation between the inexpensive decoder's algorithm and the expensive encoder's algorithm. In this manner, the millions of AC-3 decoders in DVD players and cable boxes already in circulation will remain backward compatible to the improved, future AC-3 encoders.

4.1.5 NTT Twin-VQ

Another low bitrate proprietary transform coder is called Twin-VQ, designed by NTT (Iwakami *et al.*, 1995; Iwakami and Moriya, 1996). Like AC-3 and Layer III, it includes an adaptive length MDCT filter bank. Twin VQ has three different window lengths, allowing more degrees of freedom in choosing between better frequency or time resolution in any given frame. Twin-VQ differs the most from the previous schemes in its quantization methods.

After an MDCT is taken of an audio frame, the coefficients are flattened in two separate steps. First, linear prediction is performed across frequency, and the prediction error is then a flattened version of the coefficients. By flattening, it is meant that the variance of the coefficients is reduced across frequency. The parameters of the linear prediction are converted to line spectrum pairs (LSP) (Itakura *et al.*, 1982), and then quantized using multi-stage vector quantization (Kataoka *et al.*, 1993). After LPC flattening, the spectrum is again flattened by its coarse Bark-band envelope, which itself is quantized using weighted vector quantization. After flattening by both LPC and Bark-band smoothing, the MDCT coefficients are interleaved into subvectors and are vector quantized using a perceptually weighted distortion measure. The distortion measure is dependent upon a psychoacoustic model.

4.1.6 MPEG Advanced Audio Coding (AAC)

MPEG AAC (Advanced Audio Coding) was originally designed to obtain a higher quality multichannel standard than achievable while requiring MPEG-1 compatibility (Bosi *et al.*, 1997). Initially, it was called MPEG-NBC for Non-Backward Compatibility. The goal set in 1994 by the International Telecommunication Union, Radiocommuncation Bureau (ITU-R) was for five full-bandwidth channels at a total data rate of 384 kbit/s at an *indistinguishable*, or perceptually lossless, audio quality. This goal was met and exceeded at listening tests in 1996, where MPEG AAC met the quality requirements for five channels encoded at a total of 320 kbit/s.

AAC Filter Bank

Much of MPEG AAC has borrowed technology from MPEG Layer III. Perhaps one of the biggest reasons for the improvement in quality in MPEG AAC was the new MDCT filter bank. The AAC filter bank is adaptively switched in length between 128 and 1024 frequency bins (length 256 and 2048 point windows, with 50% overlap), as pictured earlier in Figure 1.5 in Section 1.1.2. Often during an audio signal, a high-frequency resolution filter bank is needed, as was mentioned previously in Section 4.1.2. While the highest frequency resolution mode in Layer III is 576 bins, the longer MDCT transform allows for 1024 frequency bins. The almost two-fold increase in frequency resolution between Layer III and MPEG AAC allows for a significant coding gain improvement for AAC. But high-frequency resolution is not desired during transients. Both Layer III and AAC enable the window lengths to switch to a shorter length during transients. During these transients, Layer III has a frequency resolution of 192 bins, versus 128 bins for MPEG AAC. Because MPEG AAC does not use a cascaded PMQF/MDCT filter bank like Layer III, no cross-talk aliasing reduction is required.

AAC Psychoacoustic Model

The decision to switch between window lengths in AAC is guided by the perceptual entropy (PE) measure, as was the case in Layer III in Section 4.1.2. In fact, the same MPEG psychoacoustic model II is used in AAC as in Layer III.

AAC Quantization

One of the completely new quantization methods introduced in MPEG AAC is temporal noise shaping (TNS). While linear predictive coding (LPC) computes a prediction in the time domain in order to obtain a smoothed spectral envelope, TNS computes a prediction in the frequency domain (over a single frame) to obtain a smoothed time-domain envelope (Herre and Johnston, 1996). TNS uses a prediction filter between order 12 and 20, depending on the complexity profile. In doing so, TNS first reduces the variance of the MDCT coefficients by quantizing only the residual between the

original coefficients and the linearly predicted estimated of these coefficients. Secondly, by convolving the coefficients with this prediction filter in the frequency domain, it is effectively windowing the original signal in the time domain with its temporal envelope. This effective windowing process reduces pre-echo distortions by shaping the quantization noise in the time domain.

TNS proved to be especially useful for signals such as speech and other signals that have both harmonic and impulsive components. With low frequency harmonics, a long window is required in order to obtain high frequency resolution. But with a long window, pre-echo distortions surrounding the pitch pulses of speech become problematic. Therefore, by using TNS and convolving the MDCT data with a prediction filter, a new effective time-domain window is applied that forces the quantization noise to reside only during the pitch pulses, and not during the relatively quiet regions between the pitch pulses. When quantization noise is heard between the pitch pulses, the resulting audio sounds reverberant. This is perhaps due to the fact that the decay region of most reverberations can be thought of as exponentially decreasing filtered white noise (Moorer, 1979).

Another new technique introduced in MPEG AAC is prediction across frames. In this method, each particular MDCT bin is predicted over each successive frame, using second order backward-predictive filters. For very tonal signals, there is some amount of correlation in a given MDCT bin over time. Therefore, the residual between the time-predicted and the actual MDCT coefficients is quantized, which reduces the bitrate for certain signals. As a disadvantage, there is a high complexity and memory cost to using prediction. Assuming only the lower 672 of 1024 MDCT coefficients are predicted, and each predictor requires 66 instructions, then the instructions per block is 44,352. In addition, the required storage buffer is 4032 words. Prediction is only used in the *main* profile of AAC, which contains all the possible modules. In this profile, prediction requires approximately half of both the complexity and memory resources of the entire encoder (Bosi *et al.*, 1997).

The other profiles are *low complexity* and *scalable sampling rate (SSR)*. The low complexity profile does not use prediction, and the TNS filter order is reduced. The quality is still quite high in low complexity mode. As shown in (Bosi *et al.*, 1997), the only sounds tested that made a considerable difference between the main profile and the low complexity profile were the harpsichord and the pitch pipe sounds. Both of these sounds are purely tonal, and must have benefited greatly from the prediction module. The other eight sounds were of nearly indistinguishable quality relative to the original in the low complexity profile. The third profile is SSR, which can be used to make a scalable bitstream by splitting the signal into four uniformly spaced critically-sampled subbands, in addition to using gain modifiers to control pre-echo.

The rate/distortion loops used in Layer III are mostly the same in AAC. The subsequent Huffman lossless coding section has been improved from its version in Layer III to AAC. In AAC, the MDCT coefficients can be dynamically segmented into one to 49 sections, and each section can use one of eleven Huffman tables.

Applications of MPEG AAC include Japanese Digital Satellite Television, USA Digital Radio,

and Internet audio applications. As it is a relatively new standard, it may be surfacing in more applications in the future.

4.1.7 MPEG-4 Transform Coding

MPEG-4 is seen as the successor to MPEG-2 (there was no MPEG-3), and there are many different kinds of audio compression algorithms present in it. MPEG-4 has specifications for CELP-style speech coders, sines + noise coders (Edler *et al.*, 1996), and transform coders, which are designed to span the desired bitrates from 2 to 64 kbps/channel (Edler, 1996). In addition, synthesized audio using such techniques as wavetable synthesis and physical modeling are also described (Scheirer and Ray, 1998). In this section, only the new transform coder will be discussed. The MPEG-4 testing process is still ongoing at the time of this thesis writing, so the details provided below are a current snapshot of the standardization process as of the summer of 1998.

The MPEG-4 transform coding algorithm uses almost all of the functionality of MPEG AAC as its core. In addition to the modules of AAC, several other algorithms have been added to the system. These include Twin-VQ quantization, bitrate scalability quantization, and perceptual noise substitution. Currently, there are three possible quantization schemes for the MDCT coefficients in MPEG-4: the rate/distortion loops of AAC, Twin-VQ quantization, and bitrate scalability quantization. The AAC rate/distortion analysis-by-synthesis quantization loops, followed by Huffman coding was discussed earlier in Section 4.1.6. Twin-VQ was described earlier in Section 4.1.5.

Bitrate Scalability

The third alternative quantization method offered in MPEG-4 is the multi-layer bit-sliced arithmetic coding (BSAC) algorithm (Park *et al.*, 1997). Instead of iterating through analysis-by-synthesis rate-distortion loops as is in AAC, all MDCT coefficients are initially quantized such that the quantization lies just underneath the masking threshold dictated by the psychoacoustic model. Then, groups of four adjacent MDCT coefficients are formed into vectors. A subvector is then formed containing all the most significant bits (MSB) of the four MDCT coefficients, another subvector then contains the second-most significant bits of all four coefficients, and so on, until the least significant bits (LSB) are reached. Each of these subvectors are losslessly and arithmetically encoded. To create a bitstream of a certain bitrate, the encoder will utilize only a certain percentage of the arithmetic coded subvectors, starting with the MSB subvectors. In addition to scaling along this dimension of most significant MDCT coefficients, BSAC will also scale along the dimension of bandwidth. That is, at 16 kbit/s, no MDCT coefficients will be sent above 3.5 kHz. The coefficients below 3.5 kHz will be scaled by the most significant bit subvectors. For more detailed implementation details, see the ISO working draft, Grill *et al.* (1998*b*).

Perceptual Noise Substitution

It has been noticed that for certain *noise-like* regions of frequency in given frames, it is not necessary to encode the actual MDCT coefficients. Rather, it is sufficient to encode the total energy of MDCT coefficients in that frequency region, and then synthesize *random* MDCT coefficients with the same energy at the decoder. This perceptual noise substitution method for wideband audio coding was first published by Schulz (1996), and is currently in the MPEG-4 working draft (Grill *et al.*, 1998*b*). A certain frequency region is deemed noise-like depending on the tonality measure from the psychoacoustic model and if there were no large jumps in energy from the previous frame. More about this method, and other noise modeling techniques, will be discussed later in Section 5.1.1.

4.1.8 Comparison Among Transform Coders

Comparing a wide range of audio compression algorithms is a long and expensive process, but perhaps the best and most recent published comparison is Soulodre *et al.* (1998). The authors compared many coders, including MPEG Layer II, MPEG Layer III, MPEG AAC, and Dolby AC-3 at several bitrates. As an overall level of quality, the algorithms are sorted in order of best to worst : MPEG AAC, Layer III, AC-3, Layer II. The AAC testing was using its main profile. This determination was made over several bitrates, except Layer III, which was only tested at 128 kbit/stereo. See the following table for a comparison of the different algorithms at several bitrates, showing their diffgrades. A diffgrade of 0 means there is no difference heard. A diffgrade of -1 means there is a difference heard, but it is *not annoying*. A diffgrade of -2 means the material is *slightly annoying*. A diffgrade of -3 means the material is *annoying*, and a diffgrade of -4 means *very annoying*. The total bitrates are shown for a stereo signal. The testing error of the diffgrade values is ±0.111.

Codec	96 kbps	128 kbps	160 kbps	192 kbps
MPEG AAC	-1.15	-0.47	–	–
MPEG Layer III	–	-1.73	–	–
Dolby AC-3	–	-2.11	-1.04	-0.52
MPEG Layer II	–	-2.14	-1.75	-1.18

From the above table, it can be seen that MPEG AAC @ 128 kbps is roughly equivalent to AC-3 @ 192 kbps, or 50% higher bitrate. Also, MPEG AAC @ 96 kbps is roughly equivalent to MPEG Layer II at 192 kbps, or 100% higher bitrate. In addition, MPEG Layer III @ 128 kbps is equivalent to MPEG Layer II @ 160 kbps. No publicly available published comparisons could be found detailing the quality of ATRAC and Twin-VQ at these bitrates.

4.2 A Simplified Transform Coder

In the sines + transients + noise system of this thesis, the transient modeling is represented by transform coding. The goal of this thesis is not to further the state of the art of transform coding. Rather, the goal is to have a fast, efficient, and reasonably scalable transform coding algorithm with results close to that of the best available coders today. With this goal in mind, the transform coder used in this system is a simplified version of MPEG AAC. In addition, the transform coder is only active during the frames that contain attack transients, as determined by the transient detector (Section 2.2). A graphical example, in the time domain and time-frequency plane, can be seen in Figure 4.3.

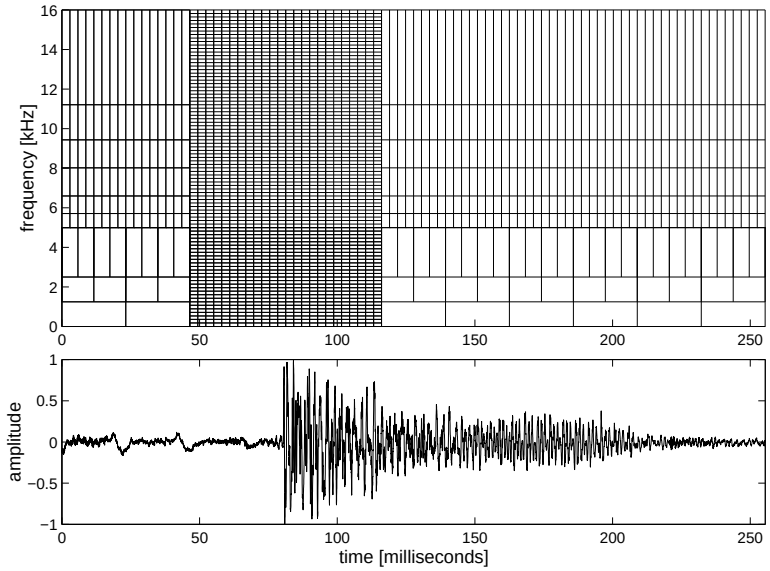

Fig. 4.3. The time region between 50 to 120 msec, which straddles the attack transient, is modeled by transform coding. The other areas of the time-frequency plane are modeled by multiresolution sinusoids and noise modeling.

The input to the transform coder is mostly just the original signal. At the boundaries of the transient region, the input signal is the residual between the original signal and the decaying, windowed sinusoids from the multiresolution sinusoidal modeling octaves. This can be seen graphically in Figure 4.4. Notice that the lowest octave sinusoids ramp-off the slowest due to their longer windows, and thus reach the farthest in the transient region.

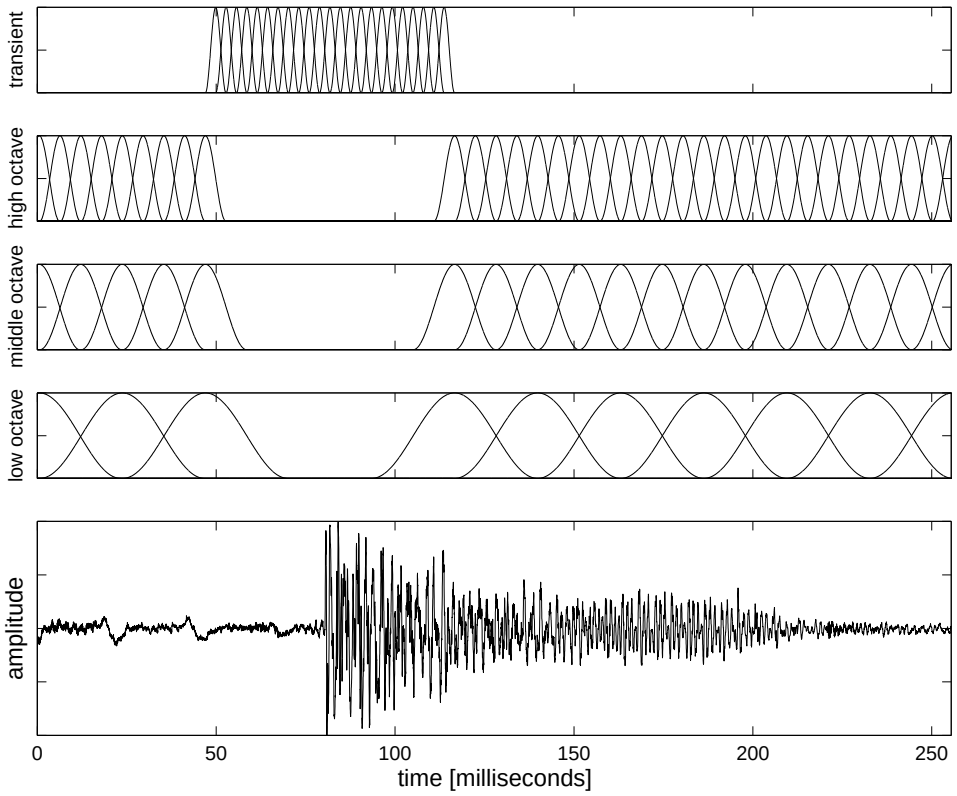

Fig. 4.4. The analysis windows for sinusoidal modeling and transform-coded transients surrounding an attack transient

Once the transient detector determines that a certain frame of audio contains a transient (Section 2.2), the residual signal is formed and is then passed to the MDCT filter bank and the psychoacoustic modeling algorithm. The transform coding flow diagram can be seen in Figure 4.5. The MDCT filter bank uses only short, 256 point raised-sine windows. As opposed to MPEG AAC, there is no need to perform adaptive length window-switching because the transform coding is only active during transients. Because MDCT windows are only used over certain segments of the audio signal, certain care must be taken at the boundaries of the transient region. Due to the implementation of the MDCT filter bank, any region of samples must be covered by two separate, overlapping MDCT windows. This is necessary to perform the alias cancellation. In this system, the transient region is 24*128 = 3072 samples long. In order to implement this with 256 point windows at 50% overlap, 26 MDCT windows are necessary. The MDCT of the first 256 point window outputs 256 samples, but only the last 128 samples are used, because only they are overlapped with its neighboring window. Similarly, only the first 128 synthesized MDCT samples of the last window are used. The additional 2 MDCT short windows amounts to a relatively small overhead. A simplified graphical example of this can be seen in Figure 4.6a. In order to reduce the bit overhead, the first and last window could only be 128 samples long, and still satisfy the overlapping of tails property necessary for alias cancellation as was shown in Equation (1.6). Figure 4.6b shows this case of shorter first and last windows. But by shortening the first and last window in half, as is done for windows during transients in AC-3 (Fielder *et al.*, 1996), the frequency response of each bandpass filter would be considerably worse. In addition, this bitrate overhead increases as the window length increases. Therefore, it is important to keep the windows short (in addition to reducing the pre-echo quantization artifacts).

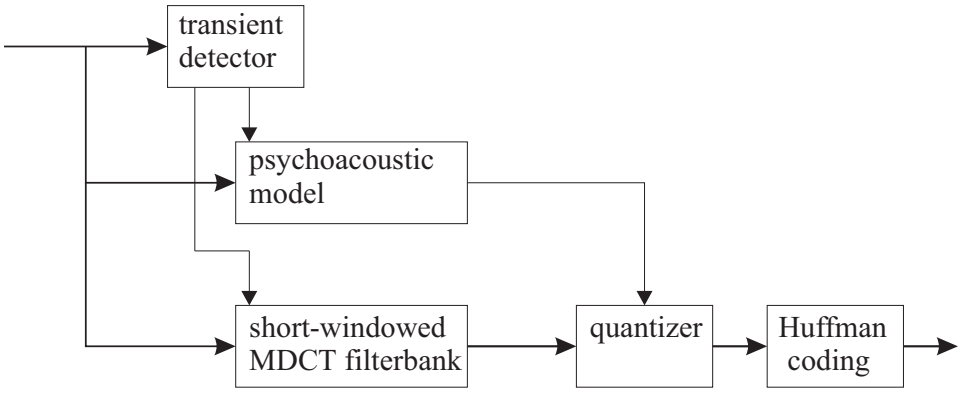

Fig. 4.5. The simplified transform coder used to model transients.

The psychoacoustic model is the same MPEG psychoacoustic model II used in MPEG-Layer III and MPEG AAC. Details of the psychoacoustic modeling, and of this particular model can be seen in Sections 1.1.2 & 3.5. The psychoacoustic model computes the signal-to-mask ratio (SMR) over many separate frequency regions, which determine how much quantization noise can be injected into

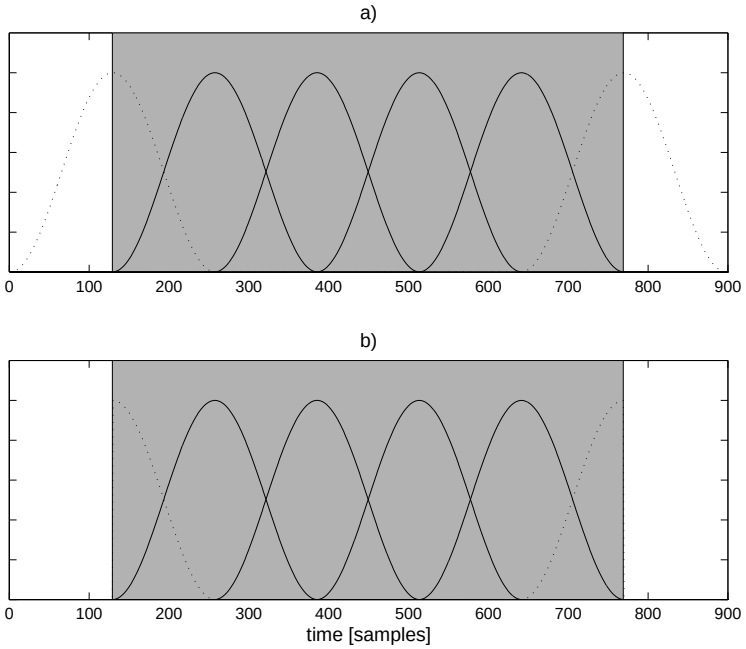

Fig. 4.6. This figure shows the boundary conditions necessary to provide MDCT transform coding over short time regions. The shaded region in both plots distinguish times that are covered by two overlapped windows. Consequently, this is the time region that is covered by transform coding. In the top figure a), the first and last windows (dotted) are of the same length as the other (solid) windows, and thus have the same frequency resolution. But, when synthesized, the time samples outside the shaded area are discarded. In the lower figure b), the first and last (dotted) windows are 50% shorter, and abruptly begin and end. This reduces the bit overhead, since no synthesized MDCT samples are discarded. The perfect reconstruction, overlapping of tails property shown in Equation (1.6) is still satisfied, but the frequency resolution of the first and last windows are inferior to that of the other middle, symmetric windows.

these frequency regions while still being inaudible (or *masked*).

The quantizer inputs the raw MDCT coefficients, along with the SMR data, and then proceeds to quantize each of the MDCT coefficients with just the minimum amount of resolution such that the quantization noise in each frequency band is masked. First, the MDCT coefficients are divided into their scalefactor bands (sfb), as is done in MPEG AAC. There are 14 nonuniformly spaced sfbs from 0-22 kHz. Each sfb has an allowed amount of quantized noise from the psychoacoustic model, called *xmin(b)* in Grill *et al.* (1998*b*). The b parameter is the variable that spans the scale factor bands. The formula for quantization of MDCT coefficient i in sfb b is

$$\mathrm{xquant}(i) = \mathrm{int}\left(\left[\mathrm{mdct}(i)2^{\frac{1}{4}(\mathrm{scalefactor}(b))}\right]^{3/4}\right) \tag{4.1}$$

The inverse quantization method is:

$$\mathrm{y}(i) = \mathrm{xquant}(i)^{\frac{3}{4}}2^{-\frac{1}{4}(\mathrm{scalefactor}(b))} \tag{4.2}$$

Each scalefactor(b) governs the quantization resolution of each scalefactor band. It is usually within

the range of -128 to 0. The closer to 0, the finer the quantization. If it weren't for the non-linearity of raising the values to the $\frac{3}{4}^{th}$ power, then every time scalefactor(b) is raised by one, the quantization resolution gets raised by $1.5dB \approx 20log_{10}(2^{\frac{1}{4}})$. With the extra non-linearity, every increment of scalefactor(b) is $1.1dB \approx 20log_{10}(2^{\frac{1}{4}\cdot\frac{3}{4}})$. As the quantization resolution increases, so does the variance of the quantized values, $xquant(i)$, and in turn, the bitrate.

The goal is to quantize the MDCT coefficients using the coarsest resolution possible (and accordingly, the lowest bitrate), but still have the quantization noise be less than the allowed masking noise in each band, $xmin(b)$. Over each scalefactor band, b, the following optimization is made to find the most negative $scalefactor(b)$ that will still have its quantization noise masked:

$$\min_{scalefactor(b)} \left[\sum_{i \in sfb(b)} [mdct(i) - y(i)]^2 \leq xmin(i) \right] \tag{4.3}$$

In order to quickly find the optimal scalefactor, a binary search is performed in $log_2 128 = 7$ iterations (Gbur *et al.*, 1996). In addition, a neighborhood of ± 3 scalefactors not already checked in the binary search is also tested. This is necessary due to the fact that with smaller scalefactor bands, especially those with only four MDCT coefficients each, quantization noise does not monotonically decrease with increasing scalefactors.

After the MDCT coefficients are quantized with the appropriate scalefactors, the quantized MDCT coefficients and the scalefactors are losslessly encoded using Huffman coding and run-length coding. In this system, in order to reduce complexity, two Huffman tables are used for the MDCT coefficients and are used over fixed frequency ranges (low and high). In order to improve coding gain, the full dynamically segmented Huffman coding using up to 11 tables of MPEG AAC could be used instead (Section 4.1.6). The scalefactors are differentially encoded over frequency each frame, and then Huffman encoded using a separate Huffman table. To reduce these scalefactor bits, promising work has been done using vector quantization compression of the scalefactors (Sreenivas and Dietz, 1998).

4.2.1 Time-Frequency Pruning

Through various rounds of experimentation, it was found that the entire frequency range does not need to be encoded during all of the 24 short windowed (66 msec total) MDCT transform frames of a transient frame, as shown in Figure 4.3. It was found that the high frequency MDCT coefficients only need to encoded during a region of time shorter than the original 66 msec transient frame. Through listening tests, high frequency MDCT coefficients (above 5 kHz) only need to be encoded for 10 of the 24 windows (29 msec total). This process of reducing the total number of MDCT coefficients in time-frequency is referred to as *pruning*. This time-frequency pruning step reduces the number of MDCT coefficients to encode by 29%. These regions of time-frequency not covered by transform coding are now represented by Bark-band noise modeling, which is a much cheaper

bitrate representation per region of time-frequency. Bark-band noise modeling will later be discussed in Section 5.2.

The placement of the 10 high frequency MDCT windows in the larger 24 window transient frame is dependent upon a secondary transient detector. The algorithm first computes the energy in the MDCT coefficients between 5 and 16 kHz over each of the 24 short windows. A large surge of energy, which is always present during a transient, is located among the 24 windows, and the group of 10 windows is centered around this transient point.

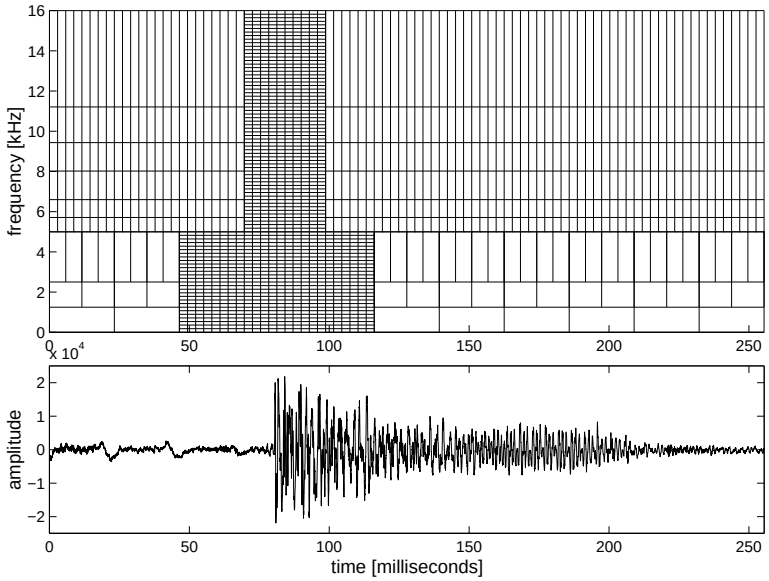

Fig. 4.7. This figure shows how to prune the time-frequency plane for transform coding of a transient. Like Figure 4.3, the lower plot shows 250 milliseconds of a drum attack in a piece of pop music. The upper plot shows the time-frequency segmentation of this signal. During the attack portion of the signal, transform coding is used for about 66 milliseconds between 0 to 5 kHz, but for only 29 milliseconds between 5-16 kHz. The remainder of the 66 msec from 5-16 kHz is encoded using noise modeling. By reducing the time-frequency region of transform coding, the bitrate is reduced as well. During the non-transient regions, multiresolution sinusoidal modeling is used below 5 kHz and Bark-band noise modeling is used from 0-16 kHz.

Given more time, and more complexity in the software, the transient MDCT coefficients in time-frequency could be pruned in such a way to match the smoother characteristics of an impulse response of a continuous wavelet transform (Vetterli and Kovačević, 1995). In this manner, the response in time-frequency would be short in duration at high frequencies, and smoothly taper wider as the frequency becomes lower. If implemented, the time-frequency plane would look similar to Figure 4.8. This hybrid solution of switching between representations at transient signals is also used in Sinha and Johnston (1996). But in that paper, MDCT coding is used during the *non-transient* signals, and a wavelet filter bank during *transient* signals. Special care was taken to ensure orthogonality during the transition between representations.

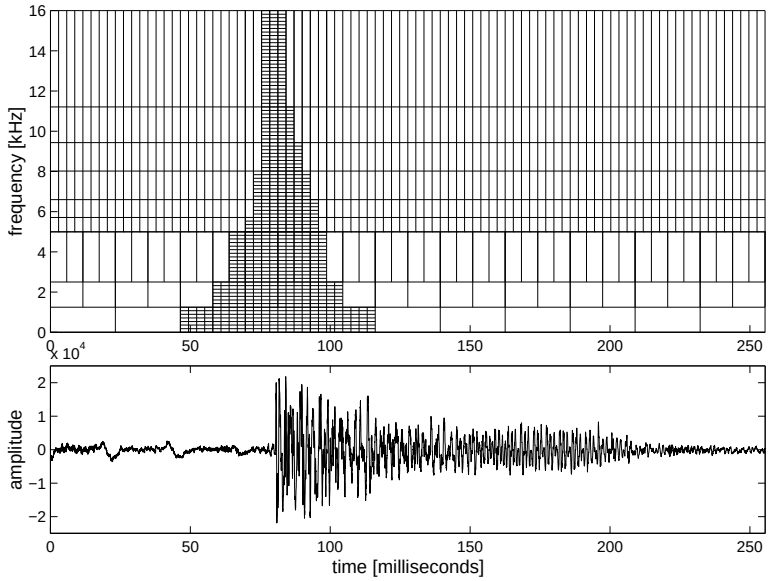

Fig. 4.8. In this figure, the time-frequency is pruned even further than in Figure 4.7 in order to represent only the regions in time-frequency around the transient. In doing so, there is a significant reduction in the number of MDCT coefficients to encode. In their place, much cheaper bitrate representations of sinusoidal and noise modeling are used.

4.2.2 Microtransients

There are certain transients, which will be termed microtransients, that are not broadband or loud enough to be found by the detection algorithm stated in Section 2.2. For example, small drum taps like a closing hi-hat sometimes appear not as a full transient, but rather as a microtransient. If these microtransients are modeled by Bark-band noise modeling, the result will not sound crisp, but rather distorted and spread. The solution is to use transform coding centered around these attacks, but only from 5 to 16 kHz. Because these high frequency transients are very sudden and short, only four transform coding frames of 128 samples each are necessary. Before and after the sudden transient, bark-band noise modeling is used. See Figure 4.9 for an example and further discussion.

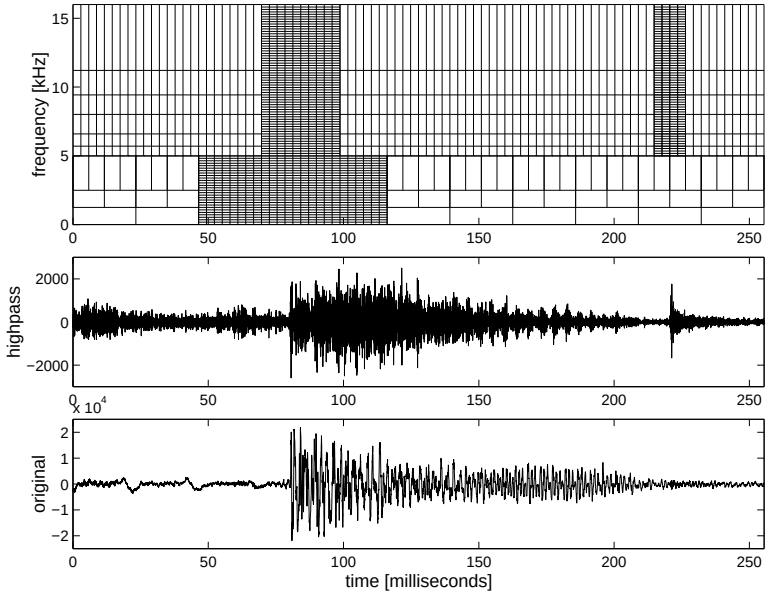

Fig. 4.9. This figure shows how transform coding can preserve sharp, high-frequency attacks. The bottom plot shows the original signal, as shown previously in Figures 4.7 and 4.8. The plot directly above it shows the same signal highpass-filtered, with a cutoff at 5 kHz. Notice that at 220 milliseconds, a transient is observed in the highpassed signal, but not in the lower wideband signal. Accordingly, the time-frequency plane around t=220 milliseconds and between 5 and 16 kHz is encoded using transform coding techniques. This preserves the high-frequency transient onset. Bark-band noise modeling is used for surrounding times above 5 kHz.

4.2.3 Bitrate-Control

As compared to sines and noise modeling, transform coding requires the most bits per area of the time-frequency plane. Because the average number of attack transients per second widely varies between input signals, the bitrates among these input signals will vary widely. In order to reduce the variance of the required bitrates among the signals, some dynamic bitrate control is applied to

the transform-coding of transients. The bitrate control comes in two flavors: a fine tuning, and a coarse tuning. The fine tuning is performed by altering the allowable signal to mask (SMR) ratio. The coarse tuning will alter the sensitivity of transient detector, and thus reduce the number of transients. These two methods will be described briefly in the following subsections. For an overall, updated transient coding flow diagram, see Figure 4.10.

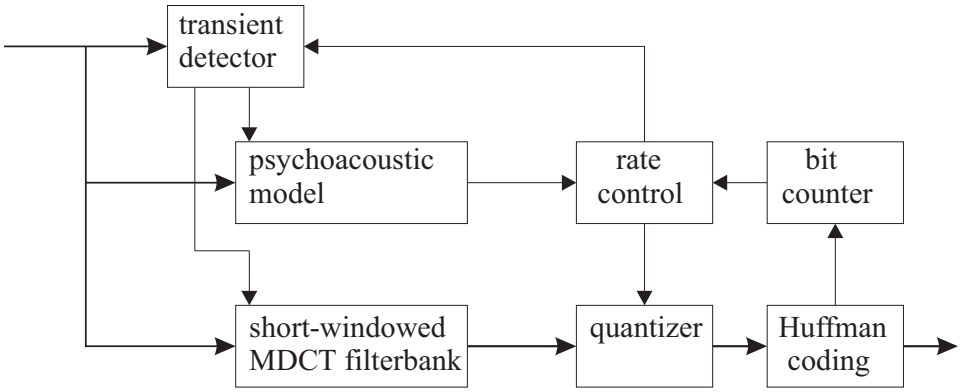

Fig. 4.10. The simplified transform coder, as shown in Figure 4.5, modified to have adaptive rate control.

Masking Threshold Adjustment

Initially, as described earlier using Equation 4.3, the MDCT coefficients in each frame are quantized as coarsely as possible such that the resulting quantization noise is barely softer than what the masking threshold will allow. The masking threshold can be anywhere between 6 to 18 dB below the (spread version of the) energy spectra, which is determined by the psychoacoustic model as described in Section 3.5. In order to make small reductions of the bitrate to encode the MDCT coefficients, the SMR can be slowly reduced from its original value across all frequencies. Lowering the SMR allows more quantization noise to be evenly introduced across all of the scalefactor bands, thus lowering the bitrate. After lossy quantization, the coefficients are Huffman coded and then the bit cost is determined and fed back to the rate control. If the bitrate is still not sufficiently low, then the rate control box, as shown in Figure 4.10 will lower the SMR again.

Transient Detector Adjustment

Data from the bit counter is fed to the rate control, which decides whether to lower the SMR or change the transient detector. If lowering the SMR is not sufficient to obtain a low enough bitrate for a certain signal, then the total number of transients is decreased. The transient detector sensitivity parameters will be altered such that only the most prevalent attack transients will be flagged. The other previously flagged transient regions will be re-encoded and modeled as sines and noise. In the limit, no transients would be encoded using transform coding. After some experimental results,

signals can be encoded with 16-20 kbps using just sines + noise. Signals synthesized using only sines and noise have duller attacks, but this is a necessary trade-off if only a limited transmission bandwidth is available. With all transients encoded using transform coding, the bitrate totals use anywhere between 20-36 kbps, depending on the number of transients detected per second.

Chapter 5

Noise Modeling

For compressing certain *noisy* signals, there are efficient model-based methods, and then there are brute-force waveform-based methods. Defining what *noisy* means is not straight-forward. In speech coding, it could be based on a voiced/unvoiced decision (Dudley, 1939). Alternatively, speech can be voiced or unvoiced in individual frequency bands (Griffin and Lim, 1988). Or, a noisy signal could be the excitation of a series of formant and pitch prediction filters (Atal and Remde, 1982). Audio signals can also be noisy in certain frequency regions, and *tonal* (non-noisy) in others (Schulz, 1996). Or, the residual of an audio signal minus a sinusoidal modeled version can be considered noisy (Serra, 1989; Hamdy *et al.*, 1996; Goodwin, 1996).

Whatever the definition of *noisy* may be, using a model-based approach will most always be more efficient in terms of bit rate than encoding the waveform itself. Encoding the waveform using time domain techniques such as ADPCM (Cumminskey, 1973), or frequency domain techniques such as transform coding (Bosi *et al.*, 1997), preserve the phase of the signal. For stationary noise processes, the phase properties of noisy signals are *not* perceptually important. Psychoacoustics dictates that for a wide class of noisy signals, only short-time energy spectral, envelopes defined over a Bark scale, are perceptually relevant (Zwicker and Fastl, 1990). These slowly, time-varying spectral envelopes can be represented with many fewer bits than needed for ADPCM or transform coding techniques. A challenge for many systems is isolating the *noisy* part of the signal from the rest of the signal. Once the noisy part is segregated from the original, then compression is relatively straight-forward.

In this chapter, previous noise modeling algorithms for speech and audio will initially be discussed in Section 5.1. Afterwards, the noise modeling component used in the sines+transient+noise system will be described in Section 5.2. Quantization techniques used in the noise modeling algorithm will then be presented in Section 5.3.

5.1 Previous Noise Modeling Algorithms

This section will describe how noise modeling has been successfully used in the past in audio and speech compression algorithms. The noise modeling algorithms have been divided into two categories, which will be termed additive and residual noise models. The former additive noise models, which will be discussed in Section 5.1.1, model frequency ranges of audio as *either* noise *or* another audio representation (*e.g.*, sinusoidal modeling or transform coding). Residual noise models model the residual of the original input minus a different initial modeling of the original input (*e.g.*, sinusoidal modeling) as noise. Residual noise models will be discussed in Section 5.1.2.

5.1.1 Additive Noise Models

The systems in this subsection categorize speech & audio as either tonal or noisy, as a function of frequency. The input is split into subbands, and a tonal/noisy determination is made within each band. Based on this determination, separate representations and quantization methods are used. In early speech coders, this tonal/noise determination was made for an entire frame, over all frequency. If a frame were determined to be noisy, then it was modeled as filtered white noise (Atal and Schroeder, 1967). Usually, the filter was a smoothed spectral envelope derived via linear prediction coding (LPC) of the original signal frame.

Multiband Excitation

The Multiband Excitation (MBE) speech coder splits a speech signal into several subbands, whose number can set to be equal to the number of harmonics (Griffin and Lim, 1988). In each subband, the speech is declared to either be tonal or noisy. In the tonal regions, the speech is synthesized as a harmonic set of sinusoids. In the noisy regions, noise is synthesized using IFFT, overlap-add techniques. The sinusoids in the tonal regions are encoded with their own amplitude and phase data, and they are harmonically related to a single fundamental frequency. The amplitudes are given by a smoothed spectral envelope. The subband energy gains in the noisy regions are also determined by the smoothed spectral envelope. The tonal versus noisy determination is made every frame by computing the ratio of the sinusoidal residual energy and the original input energy over each subband. By encoding only the gain of each noise subband and not the phase, the noisy subbands require fewer bits than the tonal subbands. In addition, these noisy subbands give the speech a more natural quality than using just sinusoids.

Transform Coding Noise Substitution

Like the MBE, it is possible for audio transform coding to obtain good results by modeling different subbands as either noisy or non-noisy. But unlike MBE, transform coding noise substitution models separate frequency bands as either noise modeled or MDCT transform coded signals. As first

introduced by Schulz (1996), certain frequency regions above 5 kHz are checked to see if they are noise-like. This topic was briefly introduced earlier in Section 4.1.7. This noisy/non-noisy determination is made by using a combination of two methods, similar to the transient detector presented in Section 2.2.3. The first method looks at the normalized mean squared error between the original MDCT coefficients in a given scalefactor band (sfb) and the predicted MDCT coefficients. Each scalefactor band may have between 4 to 64 MDCT coefficients, depending on the frequency. The second method looks for large positive changes in short-time power levels. When both of these methods coincide, a subband is termed noisy. In noisy sfbs, only an energy gain of the MDCT coefficients present in the sfb is transmitted. At the decoder, MDCT coefficients are synthesized using a random number generator with the given transmitted gain. For non-noisy sfbs, each MDCT coefficient in that sfb is quantized using traditional transform coding techniques (Section 4.1.2), which accounts for a much higher bitrate per sfb than noise modeling would require. According to Schulz (1996), the same quality of audio coding with noise substitution can be realized with traditional MPEG-1 transform coding (ISE/IEC JTC 1/SC 29/WG 11, 1993) using 20% fewer bits. Similar noise substitution techniques are currently being proposed in the MPEG-4 Audio standards (Grill *et al.*, 1998*b*).

5.1.2 Residual Noise Models

The next class of algorithms presented in this subsection represent audio *residuals* as noisy. These residuals are formed as the difference between the original signal and another initial representation of the original signal, such as sinusoidal modeling. Therefore, the final representation is the *sum* of two representations in the same frequency region, one of which is a residual noise model. This is in contrast to the previous Section 5.1.1, where the speech and audio is represented as either a noise model *or* another representation in any given frequency range. Because residual noise models have noise and an additional representation present in a given frequency range, it is sometimes difficult for the sum of the two to *sound* as if it is a single source, not the disjoint sum of two audio representations. More on this ability to perceive a *fused* audio image will be discussed at the end of Section 5.2.

Sines + Residual Noise

A major contribution to the analysis / transformation / synthesis computer music field was the work of Serra (1989). In Serra's thesis, a residual noise model was first introduced to sinusoidal modeling as a method to perform high quality modifications to monophonic musical instruments and voice. Previously, only sinusoidal models were used (McAulay and Quatieri, 1986*b*) in the context of speech. But, using only sinusoids was not sufficient for attack transients of musical instruments, or noisy signals such as breath noise of wind instruments or bow noise of string instruments. To model these more complex signals, a residual is formed with the original signal minus the synthesized

sinusoids. The intra-frame noise envelope is fit using line segment approximation (Phillips, 1968) in the short-time frequency domain. To synthesize the noise, an IFFT with random phase and line segment approximated spectral magnitude is computed. While this system performs very well for high quality modifications, such as time-scaling and pitch-shifting, it was not designed for data compression. But, its contribution of using a residual noise model for sinusoidal modeling has been used by many other researchers in the audio data compression community.

Another system to use sinusoidal modeling with a noise-modeled residual is currently being investigated for the MPEG-4 Audio standard (Edler *et al.*, 1996; Grill *et al.*, 1998*a*). This system, as opposed to the previously mentioned system by Serra (1989), is designed specifically for very low data rates (6 to 16 kbps) and compressed domain modifications. The sinusoids can be synthesized with independent frequency information, or as a harmonically related set (if appropriate). After subtracting the original signal from the synthesized sinusoids, a residual is formed. The residual is then modeled using an IFFT with a spectral magnitude envelope and random phase, similar to (Serra, 1989). To reduce the bitrate of the envelope, the envelope is generated by first computing a DCT of the FFT magnitude of the residual short-time signal. The first seven DCT coefficients are quantized and transmitted. At the decoder, and inverse DCT creates a smoothed version of the residual magnitude spectra. The noise is synthesized using an IFFT with this DCT smoothed envelope and random phase. Therefore, the final sound is a sum of the sinusoids and the residual noise model, both from 0 to $fs/2$ Hz..

A more recent method of sines and residual noise modeling is presented by Goodwin (1996, 1997). In this paper, the residual signal is modeled by first splitting a signal into equivalent rectangular bands (ERBs) (Moore and Glasberg, 1996), which are non-uniform divisions of frequency similar to the critical bands of hearing (Zwicker and Fastl, 1990). Once the signal is split into its ERBs, the total energy is computed in each ERB band, for each frame. Splitting the signal into ERBs can be achieved with either a set of bandpass filters or FFT/IFFT techniques. At the decoder, random spectral magnitudes are generated in each ERB with their corresponding ERB encoded gains, along with random phase values. These synthesized magnitudes and phases are processed via an windowed overlap-add IFFT, and the residual noise is complete. This system in Goodwin (1996) was originally designed for modifications, and was not specifically designed for data compression. But, as will be discussed in more detail later in Sections 5.2 and 5.3, this noise modeling technique can be efficiently quantized.

Sines + Wavelets + Residual Noise

The first in a series of publications of sines+wavelets+noise papers is by Ali (1996), followed by (Hamdy *et al.*, 1996; K.N.Hamdy *et al.*, 1997; S. Boland, 1997). These papers extend the modeling of the residual of sinusoidal modeling to include transient and noise modeling. This section will describe the work done by Ali (1996) and Hamdy *et al.* (1996).

The sinusoidal modeling uses different parameter estimation techniques (Thomson, 1982), but still uses traditional tracking and intra-frame parameter interpolation algorithms (McAulay and Quatieri, 1986*b*; Serra, 1989). The sinusoidal residual below 11 kHz is then encoded using previously established wavelet coding techniques (Sinha and Tewfik, 1993). The residual above 11 kHz is further split between *edges* and *non-edges*. It seems that non-edges of the wavelet transform coefficients can be efficiently modeled by simply synthesizing random coefficients. The variance of the coefficients (assuming zero mean), along with a temporal decay variable is transmitted. At the decoder, non-edge wavelet transform coefficients in each of the bands are synthesized according to the variance and decay constant. It is reported that for most signals, over 80% of wavelet coefficients (above 11 kHz) can be synthesized using noise substitution techniques without any audible artifacts. Because the time resolution is so much finer in high frequency wavelet transform bands, these noise substitution techniques are necessary to reduce the large number of parameters per second. The *edge* wavelet coefficients, which are determined by a transient/edge detector, are encoded separately, and are not synthesized. The method of encoding the *noisy* high frequency wavelet coefficients is very similar to the noise substitution method of MDCT coefficients previously described in Section 5.1.1 (Schulz, 1996). This shows that for high frequency signals in certain cases, audio can be approximated very efficiently by noise processes without much perceptual distortion.

5.2 Bark-Band Noise Modeling

In the system presented for this thesis, the noise model used is a combination of both additive and residual noise models (only during non-transient regions). From 0 to 5 kHz, residual noise modeling is performed on the residual of the original signal minus the synthesized sinusoids. From 5 to 16 kHz, only additive noise modeling is performed on the original signal itself. To maintain a low total bitrate, it is assumed that all non-transient regions from 5 to 16 kHz are noise-like, which is similar to the high frequency assumptions made in (Hamdy *et al.*, 1996). This assumption seems to be reasonable tradeoff in quality; if more bits were taken from the transient or sinusoidal models in order to add higher frequency sinusoids, more perceptible artifacts will be heard in those regions of time-frequency. In other words, noise modeling is a very inexpensive (by bitrate) method to increase the audio bandwidth that does not introduce very objectionable artifacts. For most music tested at bitrates below 32 kbps, artifacts introduced by modeling the non-transient bandwidth from 5 to 16 kHz as only noise (instead of sines + noise) are subtle. Some artifacts heard due to noise modeling will be a slight spreading of any attacks not captured by the transient model. In addition, monophonic, tonal instruments will have their higher harmonics modeled inaccurately by noise. For the future, this particular class of tonal input signals may be detected, and thus increase the frequency range of sinusoidal modeling higher than 5 kHz. But at these very low bit rates, a larger audio bandwidth and slight high-frequency artifacts seems like a better design than a lower

total audio bandwidth with fewer artifacts.*

Through psychoacoustic experiments, it has been found that for stationary, noise-like signals, the ear is not sensitive to variations of energy within a Bark band. Rather, it is sensitive to the total short-time energy within each Bark band. Between 0 and 22 kHz, there are only 25 Bark bands. For a more detailed description of the critical bands of hearing, see the previous discussion on psychoacoustic masking thresholds in Section 1.1.3. Therefore, to efficiently encode the noisy audio signal, the noise modeling encoder divides the audio spectrum into uniform sections on the Bark frequency scale (Zwicker and Fastl, 1990) (but non-uniform on the linear frequency axis) many times a second. This process can be accomplished using a set of bandpass filters, or FFT methods (Goodwin, 1997). In Figures 5.1 and 5.2, the FFT noise encoder and decoder are shown.

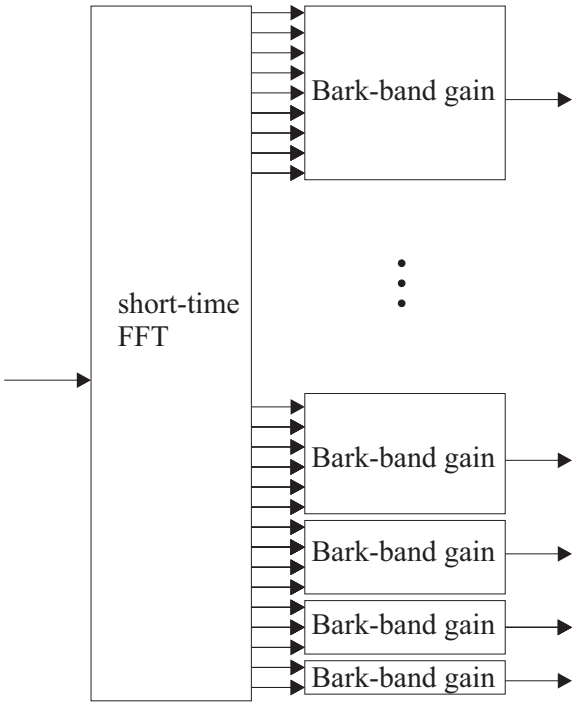

Fig. 5.1. This figure shows the basic noise modeling encoder. The windowed input signal is placed through a short-time FFT. Groups of FFT bins are formed into Bark bands, and a Bark band energy gain is computed for each. The figure is not drawn to scale.

To give a graphical example of a real audio signal, see Figures 5.3 and 5.4. In the left part of Figure 5.3 is the original frame of audio. The right part shows the residual of the original signal (below 5 kHz) minus the multiresolution sinusoidal modeling. In Figure 5.4, there are two spectra. One is the short-time spectral magnitude of the residual, and the other is the Bark-band approximation of the residual. Its spectrum has a flat magnitude over the frequency range of each Bark band.

To assure the signals are perceptually similar, the energy is matched over each Bark band between

*This matter may be a personal preference, but other researchers have also agreed in private communications.

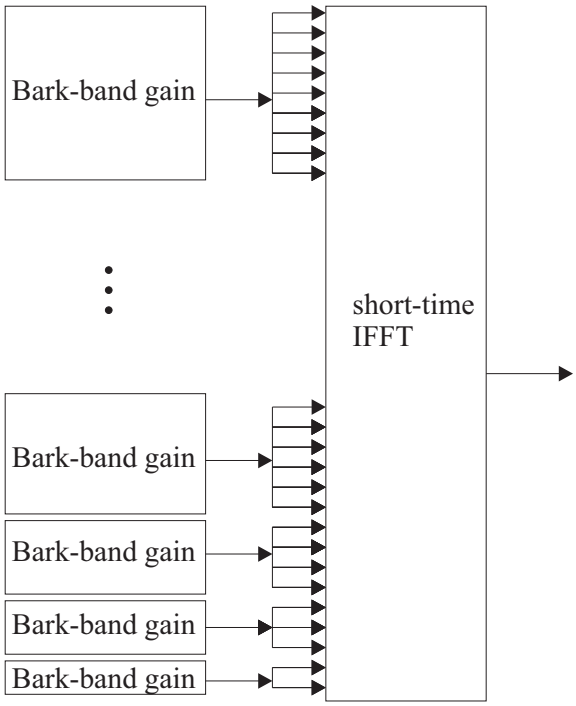

Fig. 5.2. This figure shows the basic noise modeling decoder, to complement the noise encoder in Figure 5.1. Each Bark band gain is used as a uniform magnitude across all FFT coefficients in that particular Bark band. Random phases are generated for each FFT bin. The figure is not drawn to scale.

the residual and the Bark-band synthesized signal. To show this, the same derivation and notation is used from Goodwin (1997). Let $S(k,i)$ denote a particular original analysis FFT bin, with the k^{th} bin and the i^{th} frame. Then $\hat{S}(k,i)$ be the FFT bins of the piecewise constant Bark band synthesized noise. In addition, β_r is the number of FFT bins in the r^{th} Bark band, and M is the size of the analysis/synthesis IFFT. Therefore, the energy of the sum of the FFT bins in the original spectrum and the piecewise uniform Bark-band synthesized noise are equal:

$$\tilde{E}_r(i) = \frac{1}{M} \sum_{k \in \beta_r} |\hat{S}(k,i)|^2 = \frac{1}{M} \sum_{k \in \beta_r} |S(k,i)|^2 \tag{5.1}$$

Because all of the FFT magnitudes are identical in a given Bark band, the left side of the above equation can be rewritten as:

$$\tilde{E}_r(i) = \frac{\beta_r}{M} |\hat{S}(k,i)|^2 \tag{5.2}$$

Then, the piecewise uniform Bark band magnitude can be derived as a function of the energy in the analysis FFT Bark band:

$$|\hat{S}(k,i)| = \sqrt{\frac{M}{\beta_r} \tilde{E}_r(i)} \tag{5.3}$$

Each FFT frame is therefore constructed by a piecewise uniform magnitude across each Bark

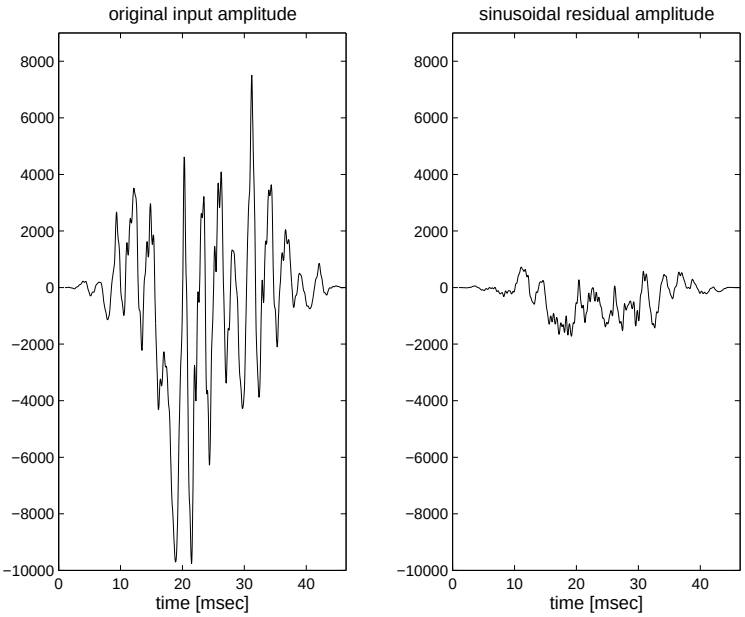

Fig. 5.3. The left signal is a frame of the windowed original signal. The right signal is the windowed residual of the original input audio minus the multiresolution sinusoidal modeling.

band and a random phase component. After an IFFT, the time domain output is windowed and overlapped-added with 50% overlapping Hanning windows.

The method of Goodwin (1997) was chosen as a noise model after comparing it to the implemented results using DCT-smoothed spectral envelopes (Grill *et al.*, 1998*a*) and LPC-smoothed spectral envelopes (Serra, 1989). All three models allowed for low bitrates, but only the Bark band noise model had noise that seemed to *fuse* with the sinusoids. In the other two methods, the sines and noise seemed to be perceptually separate audio streams added together, and did not perceptually form a single audio signal. But, when the sines and the Bark band noise (from 0 to 5 kHz) were added together, the sum seemed like a single source was playing, not just independent sines and noise signals being played simultaneously. This is a difficult phenomenon to describe in words, and perhaps described best by an audio example. While this ability to perceptually fuse sines and Bark band noise seemed to work well for the variety of audio material tested, different noise methods could conceivably work better or worse on different input source material.

5.3 Noise Parameter Quantization

Now that the noise modeling representation is chosen, the next task is to efficiently quantize the parameters in a perceptually meaningful manner. As was mentioned earlier (during non-transient segments), from 0 to 5 kHz there are sines and noise, while above 5 kHz there is only noise. Accordingly, noise is quantized differently above and below 5 kHz. In quantizing these noise parameters,

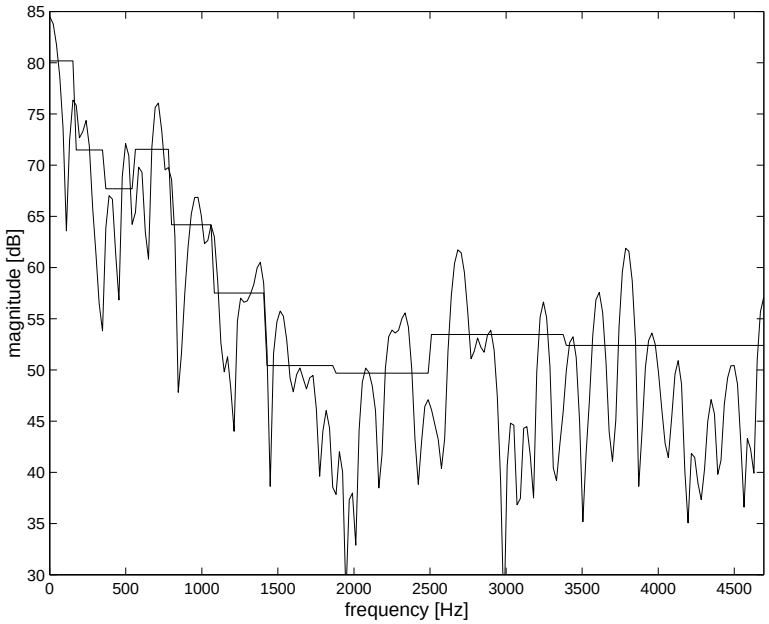

Fig. 5.4. This figure shows the magnitude spectrum of the residual signal, and the piecewise constant magnitude spectrum of the Bark-band noise modeling.

there are several design choices:

- the total number of noise bands over frequency
- the temporal update rate of the Bark band gains
- magnitude resolution of each Bark band gain

From 0 to 5 kHz, after extensive yet informal listening tests, the noise parameters were quantized over 10 equi-Bark spaced bands, the analysis windows were 1024 points long (50% overlap, which is 86 frames/sec at 44.1 kHz), and each band's energy gain was scalar quantized with 1.5 dB magnitude resolution. This amounts to a bit rate of 2 to 3 kbps for most all audio inputs. Because the residual noise signal is considerably quieter (on average) than the synthesized multiresolution sinusoidal signal it is being added with, the frame rate of these noise parameters can be lower. The noise modeling thus has the same temporal resolution as the middle multiresolution sinusoidal modeling octave. To help improve the temporal resolution of the noise modeling, an additional noise envelope can be added (George and Smith, 1992; Edler *et al.*, 1996).

Because additive noise modeling is the only representation of the non-transient regions of audio above 5 kHz, it is important to have good temporal resolution of the Bark band energies. After many listening tests, sufficiently high quality is achieved using 6 Bark bands from 5 to 16 kHz, and an analysis frame of 256 points (50% overlap, which is 344 frames/sec at 44.1 kHz). At this frame

rate, using scalar quantization and Huffman encoding, the noise from 5 to 16 kHz would require about 10 kbps.

In order to lower the bitrate of this high frequency noise information, it was discovered during listening tests that the temporal resolution of the Bark band gains, $\tilde{E}_r(i)$, can be reduced, but *only* in certain regions, while still maintaining high quality. As a graphical example, see the top plot in Figure 5.5. This plot shows the Bark band energies in one band from 8000 to 9200 Hz, over a time period of 330 milliseconds. An example of a smoothed Bark band signal $\tilde{E}_{r,LSA}(i)$ can be seen in the lower plot of Figure 5.5.

The next problem to solve is to distinguish which regions of $\tilde{E}_r(i)$ need to be represented closely, and which regions can be smoothed or approximated. After many listening tests, it seemed that the regions of $\tilde{E}_r(i)$ that needed fine quantization were those surround large sudden increases or decreases in magnitude. In Figure 5.5, sudden changes of $\tilde{E}_r(i)$ occur at times 60 to 80 msec, and from 210 to 220 msec. In other regions, the Bark band energies can be efficiently approximated using relatively long line segments.

To smooth the signal, first a set of *breakpoints* are determined. Breakpoints are so called because they are points at which the straight lines *break* to change slope. These breakpoints are certain elements of $\tilde{E}_r(i)$ that are transmitted. The remaining points at the decoder are linearly interpolated from these breakpoints. In the bottom plot of Figure 5.5, the breakpoints are the circled samples, and the remaining points are clearly linearly interpolated from them. To quantize this representation, the magnitude (in increments of 1.5 dB) and the timing of the breakpoints are differentially scalar quantized, and then Huffman encoded. In order to protect the bitstream from packet or bit errors, the breakpoints are not differentially quantized for all time; every several frames, the breakpoints are coded individually, without knowledge of the previous breakpoint.

In order to determine these breakpoints, a greedy algorithm by Horner *et al.* (1995) is implemented that iteratively decides where a new breakpoint in the envelope would best minimize the error between the original and approximated envelope. The number of breakpoints is set to 20% of the length of the envelope itself. Using fewer breakpoints would lower the bitrate, but would introduce audible artifacts in the synthesized noise. By using line segment approximation, the bitrate is reduced from 10 kbps to 3 kbps, while still maintaining the same perceived quality of synthesis.

While line segment approximation techniques can model most high-frequency transients, the most sudden high frequency transients do not use noise modeling at all. As was discussed earlier in Section 4.2.2, transform coding is utilized over a short period of time at high frequencies for a certain class of transients. These *microtransients*, as can be viewed in Figure 4.9, model short transients that only occur at higher frequencies. Examples of such microtransients are hi-hat closings or cymbal taps.

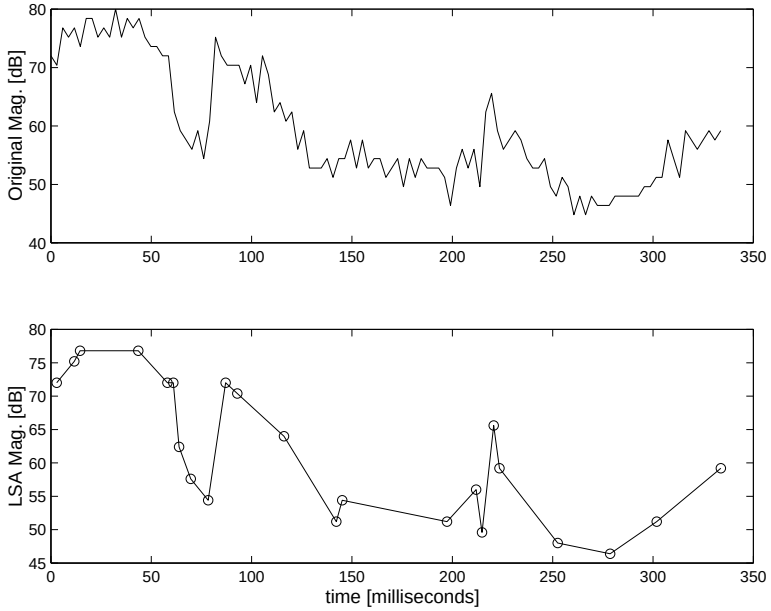

Fig. 5.5. The top plot shows a bark band (8000-9200 Hz) RMS-level energy envelope for about 300 milliseconds. The bottom plot shows the line segment approximated RMS-level energy envelope. The circled points are the transmitted envelope points, and the remaining points are linearly interpolated using the transmitted points.

Chapter 6

Compressed Domain Modifications

As was mentioned earlier in Section 1.3, compressed domain processing is defined as the ability to perform modifications on the audio while in its compressed form. The benefit of this property is a large reduction in complexity and latency. Because modifications are performed separately on the sines, transients, and noise, the quality of the modifications rival the best commercial modification systems available today.

As an example, one user receives a compressed audio bitstream. This user needs to perform time scale modification upon it, and then transmit the compressed data to a second user. If the modifications are performed in the compressed domain, then the bitstream can be simply sent ahead to the next user with only a relatively small amount of partial decoding and processing delay. This can be seen in the flow chart of Figure 6.1.

Fig. 6.1. Performing modifications in the compressed domain

If the modifications were performed on the raw audio pcm file, then the encoded file would first have to be decoded, then modified, then encoded as seen in Figure 6.2.

In the audio compression system presented in this thesis, the encoder has higher complexity and latency than the decoder.* Therefore, adding an extra encoding step every time modifications are desired can significantly increase the complexity and latency requirements. As more and more audio is exchanged in compressed-form across data networks, the need to perform quick modifications with low complexity and latency will only increase.

*This asymmetric complexity property is also shared by all other current commercial audio compression systems.

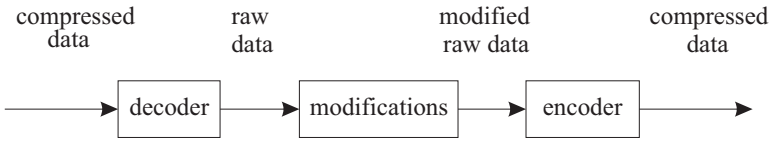

Fig. 6.2. Performing modifications by switching to the time domain

To show more detail of the compressed domain processing system, see Figure 6.3. Initially, the input compressed audio bitstream is demultiplexed into its four separate components: the multiresolution sinusoidal parameters, the low frequency {0-5 kHz} residual Bark-band noise model coefficients, the transform-coded transient data, and the line-segment approximated Bark-band high frequency noise model coefficients. Each of these representations have to be inverse quantized separately. In the case of the sinusoids (Section 3.7), the residual noise parameters (Section 5.3), and the transform coded data (Section 4.2), the inverse quantization is some class of Huffman decoding and table lookups for inverse scalar quantization. For the high frequency noise model (Section 5.3), the Bark-band noise energy gain envelopes are easily recreated from the line segment approximation breakpoints in the Huffman decoded bitstream.

Once the {sines,transients,noise} bitstreams have been inverse quantized into raw parameters, they are then modified in the *compressed domain modifications* box in Figure 6.3. Details of particular time and pitch scale modifications to each of these sets of {sines,transients,noise} parameters will be discussed later in Sections 6.1 and 6.2 respectively.

After modifications, the parameters can either be easily re-quantized and multiplexed back into a single compressed bitstream or decoded into a raw audio signal for playback. For the re-quantization process, shown in Figure 6.3 on the bottom row from right to left, the complexity is very low. In the case of time-scale modification, the parameters before and after the modification process do *not* change. Rather, only the timing information is altered. Therefore, the inverse quantization state information is saved before modifications, and then used again to re-quantize the parameters. In the case of pitch-scale modifications, only the sinusoidal frequency parameters are scaled, and all other parameters remain the same. Since sinusoidal frequency information is quantized in a table-lookup as discussed earlier in Section 3.7, this added encoding takes very little complexity.

6.1 Time-Scale Modifications

As was originally mentioned in Section 2.3.1, time-scale modification is handled separately for sines+noise and for transients. The sines+noise signals are *stretched* in time, while the transients are merely *translated* in time. By using translation and not stretching, the waveform corresponding to the transient is not altered; rather, it is now restored in a new time location. In the specific case of drum hits, when time is slowed, one would desire the drum attacks to be just as sudden and

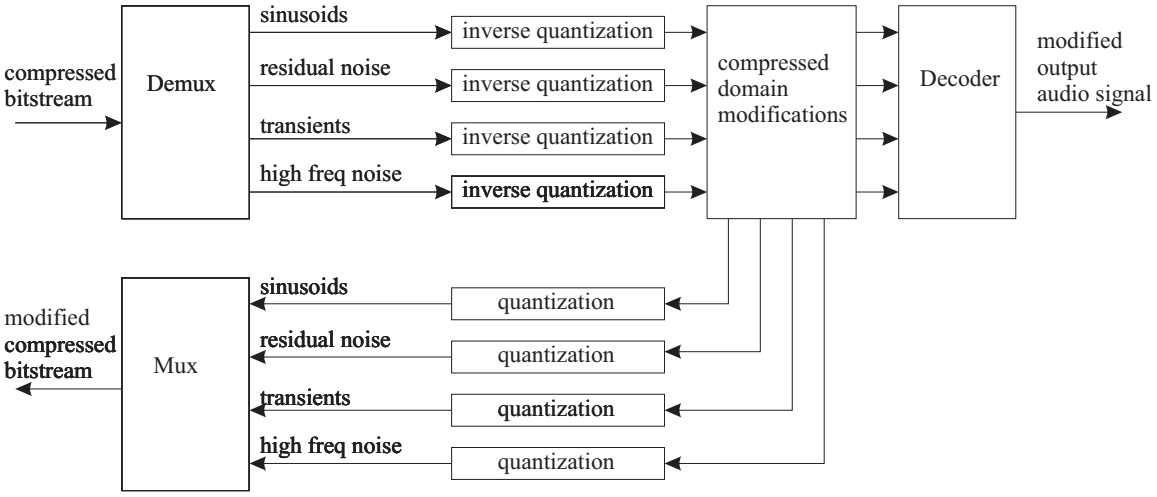

Fig. 6.3. This figure shows in more detail how a compressed bitstream is modified. First, each of the independent signals must be inverse quantized: multiresolution sinusoids, low frequency residual noise, transform-coded transients, and the high frequency noise. This inverse quantization process is usually inverse Huffman coding and table lookups. After the modifications, the data can either be decoded and listened to, or requantized and retransmitted in compressed form.

sharp as the original signal. Using this algorithm, the drum attacks will simply be farther apart in time than the original, but the decay of the drums will now last longer. For a simplified graphical example of this algorithm, see Figure 6.4.

For a more specific, real-world signal example, see Figure 6.5. The left column signals show (from top to bottom) the original signal, the fully synthesized signal, and then the separate sines, transients, residual noise, and high frequency noise signals. These synthesized signals on the left side of the figure all occur at the original time-scale, with $\alpha = 1$. On the right column, the same corresponding synthesized signals are shown synthesized twice as slow as the original, with $\alpha = 2$. Notice that the sines and the two noise signals are stretched by a factor of two, but the transients are merely translated farther apart in time.

In the next several subsections, more details will be given that describe how each of the individual representations are altered. The sines and both noise models will be described in Section 6.1.1, and the transients will be detailed in Section 6.1.2. In Section 6.1.3, the phase-matching between the multiresolution sinusoids and the transform coded transients, even when time-scaled, will be described.

6.1.1 Sines and Noise

In order to *stretch* the sines and noise representations by a factor of α, the simplest method is to alter the synthesis window length by a factor of α. If $\alpha = 1$, then the output is synthesized at the same speed as the original, and if $\alpha = 2$ the output is synthesized twice a slow as the original. Let

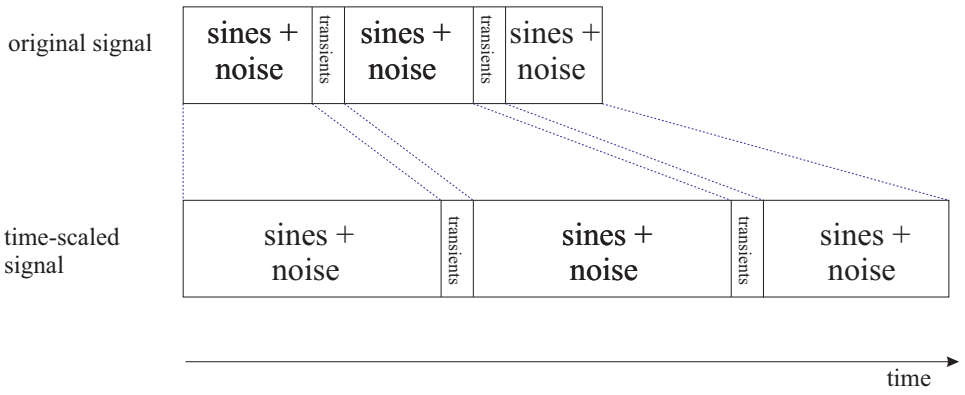

Fig. 6.4. This figure shows how time-scale modification is performed. The top diagram shows the time relationships between the sines+noise and the transients in the original signal. In the lower diagram, the same signal is time-scaled to be slower than the original signal. The sines and noise segments of the audio are *stretched* in time, while the transients are *translated* in time.

the analysis window length be termed L_a, and let the synthesis window be L_s, such that $\alpha = \frac{L_s}{L_a}$. Assume that at any time scale α, the overlap of the windows is always maintained at 50%. The methods of stretching the parameters are slightly different if synthesized in the time or frequency domain. In the next two subsections, each of these cases will be detailed.

Time Domain Methods

Both the sines and the noise can be synthesized in the time domain. While more straightforward to implement in the time domain, the computational complexity is higher than using frequency domain methods discussed in the next subsection. When sinusoids are synthesized using a bank of oscillators in the time domain, sample-rate parameters of {amplitude,frequnency,phase} are interpolated from the frame-rate parameters that were quantized and transmitted from the encoder. Now that the frame length has been stretched, $L_s = \alpha \cdot L_a$, the sample-rate parameters are still interpolated from the transmitted frame-rate parameters using the same methods. But because the distance between frames is now different, the sample-rate parameters have changed. By stretching the sinusoids in this manner, only the time evolution of the signal changes. For example, a sinusoid that swept from 400 Hz to 405 Hz over 20 msec in the original signal now sweeps from 400 Hz to 405 Hz over 20α msec. The pitch is *not* altered due to the time scale changing, as would occur during simple sample-rate conversion of the original signal.

When synthesizing noise in the time domain, white noise is filtered by a bank of bandpass filters whose gains are interpolated at a sample-rate from the frame rate, quantized and transmitted Bark-band energy gains from the encoder. Just as in the case of the sinusoids, to time-stretch the noise, simply spread apart the frame-rate Bark-band gains by a factor of α. Then linearly interpolate the sample-rate energy gains from these stretched frame-rate parameters.

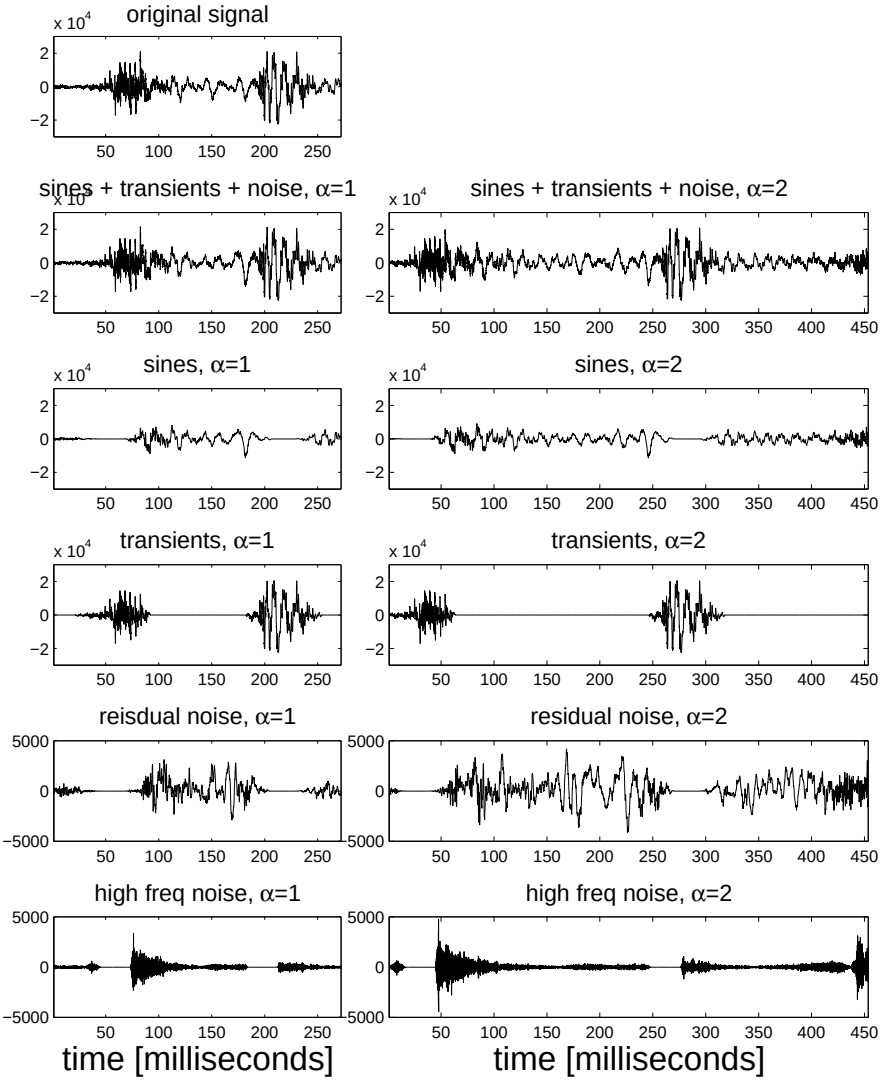

Fig. 6.5. This set of plots shows how time-scale modification is performed. The original signal, shown at top left, shows two transients: first a hi-hat cymbal hit, and then a bass drum hit. There are also vocals present throughout the sample. The left-side plots show the full synthesized signal just below the top, and then the sines, transients, and noise independently. They were all synthesized with no time-scale modification, at α=1. The right-side plots show the same synthesized signals, but time-scale modified with α=2, or twice as slow with the same pitch. Notice how the sines and noise are stretched, but the transients are translated. Also, the vertical amplitude scale on the bottom two noise plots are amplified 15 dB for better viewing. In addition, the silenced time regions in the residual and high frequency noise models occur at different times due to the time-frequency pruning operations above 5 kHz (Section 4.2.1) and the differences in window lengths between the two noise models (6 versus 23 msec).

Frequency Domain Methods

The sines and noise can also be synthesized using more efficient FFT methods, and can still be stretched as was done using time domain methods. Instead of synthesizing sines using a bank of oscillators, a triad of sinusoidal parameters frame can be converted to a main-lobe width of spectral data (Rodet and Depalle, 1992). It is also possible to have linearly changing sinusoidal frequency throughout the frame by using more complex algorithms (Goodwin and Rodet, 1994; Goodwin and Kogon, 1995). In order to synthesize noise in the frequency domain, piece-wise uniform energy gains are placed in the FFT domain, as was discussed earlier in Section 5.2 and shown in Figure 5.4.

Once all the sinusoidal triads and the noise gains in a given frame are converted to spectral data, a single IFFT is performed, time-windowed, and overlapped with the previous frame. In order to stretch the sines and noise using FFT methods, simply change the synthesis IFFT frame length according to the time-scaling factor β. The IFFT-domain sines and noise parameters will be slightly altered when time-scaling due to the different number of spectral bins between the encoder and decoder.

6.1.2 Transients

Handling the transients for time scale modification is much simpler than handling the sines or noise representations. The MDCT coefficients of the transform-coded transients are merely translated to a new position in time. As can be seen in Figure 6.5, the short-time attack regions of the signal modeled by transform coding are moved to their new location, which is directly after the previously stretched region of stretched sinusoids and noise. The MDCT coefficients are not modified, and the frame rate of the short MDCT windows remain the same.

6.1.3 Transition between Sines and Transients

As was seen earlier in Chapter 2, and now again in Figure 6.6, there is a considerable amount of overlap between the multiresolution sinusoidal modeling windows and the transform coding transient MDCT windows. In the lower octaves, the overlap between the sines and transients is larger than that of the higher octaves, due to the longer window lengths. Special care is taken to match the phases of the sinusoids in the frame preceding the beginning and end of the transient region to the overlapping transient signal. The reason for the phase-locking is to ensure a smooth transition between the sines and transients. More details of this procedure were discussed earlier in Section 3.4.3.

When a signal is time-scale modified, all the sinusoidal modeling windows stretch by a factor of α, but the parts of the sinusoidal windows that overlap the transient regions do *not* stretch. The overlapping sinusoidal frames are not stretched in order to ensure that the sum of the sines and transients in the overlapping region is the same as in the original signal. As was mentioned before,

the transient signal is not stretched, but rather translated. Therefore, the sinusoids that are added to the transients during the overlap region must not be stretched either. This concept can be viewed in Figures 6.6 and 6.7. If the sinusoidal windows portions during the transient region were stretched, then the sum of the sines and transients would *not* sum to be roughly equal to the original signal.

Even during time-scale modification, the sinusoids are kept phase-locked to the transient signal during the portion of overlap between the two representations. In Figure 6.7, this overlap region corresponds to time regions of approximately {95 to 110 msec} and {145 to 160 msec}. These overlap time regions are slightly different in each of the multiresolution sinusoidal octaves, but are independent of the time scaling factor, α. In each octave, the frame before the overlap region, F_{t-1}, receives initial sinusoidal phases from the phaseless reconstruction performed in previous frames. Over the period of the frame F_{t-1}, phases must be interpolated from the initial phase at the beginning of the frame to the phase at the end of the frame derived from the encoder's parameter estimation. If this condition is satisfied, then the frame during the overlap region, F_t, will have sinusoids phase-locked to the original signal, and thus the transform-coded transients. By altering and interpolating the phases across F_{t-1}, some amount of frequency shift will be introduced. But, it is not been perceptually noticeable in any of the audio signals already processed. If it were ever to become a perceptually relevant problem, then the phases could be interpolated over more than a single frame before F_t. The more frames the phase is interpolated over, the smaller the frequency shift. For a previous treatment of this topic of switched phase reconstruction, see Figure 3.4.3.

6.2 Pitch-Scale Modifications

Pitch-scale modifications are defined as altering the pitch of a signal without changing its playback speed. The pitch shifting parameter, β, is defined as the ratio of the output pitch to the input original pitch. For example, $\beta = 2$ means that the output signal will be an octave higher than the original. The pitch-scale modifications are somewhat simpler conceptually than time-scale modifications. The only changes to make in this system is to alter the multiresolution sinusoidal frequencies by β.

6.2.1 Noise and Transient Models Kept Constant

The residual noise, the transients, and the high frequency noise are all kept constant. The residual and high frequency noise components are not pitch-scaled since there should not be any tonal components in these representations. It is desirable that only the tonal components be scaled, and the noisy sources should remain the same. Ideally, the residual noise is always the difference between the input signal and the synthesized sinusoids. But the difference signal between the input signal and a pitch-scaled synthesized sinusoids does not deliver a meaningful residual. Therefore, the residual noise obtained from the difference of the original signal and the non-pitch-scaled sinusoids is kept.

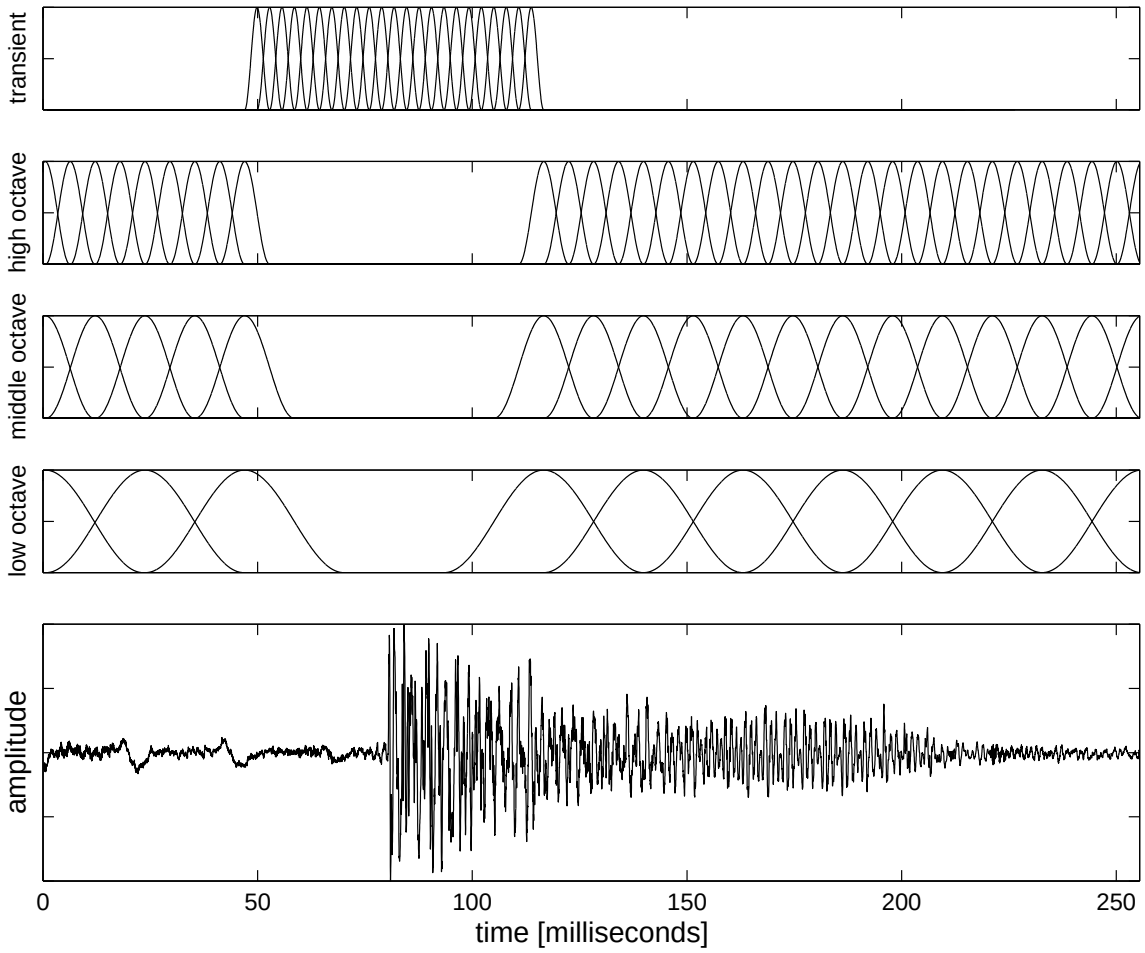

Fig. 6.6. This figure shows the (from top to bottom) the MDCT windows used for transform-coded transients, the analysis windows for multiresolution sinusoidal modeling, and the original synthesized sines+transient signal. In this figure, no time-scale modification, $\alpha = 1$ is performed. This is in contrast to Figure 6.7, when the audio is stretched, with $\alpha = 2$.

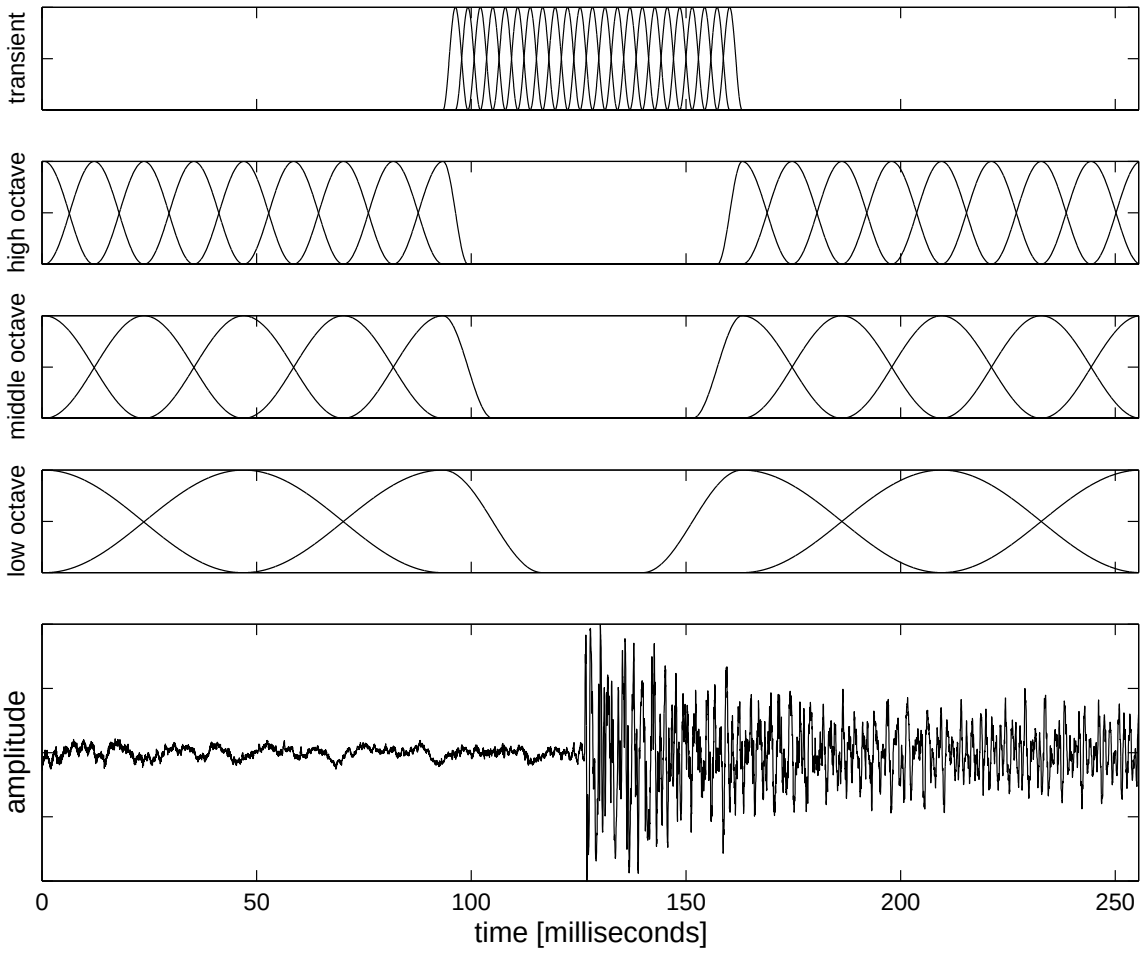

Fig. 6.7. This figure shows the (from top to bottom) the MDCT windows used for transform-coded transients, the analysis windows for multiresolution sinusoidal modeling, and the synthesized sines+transient signal when time-scale modified by $\alpha = 2$. Notice that the sinusoidal modeling windows at the transient region boundaries are non-symmetric. The portion of the window that overlaps with the transient MDCT windows retain the original length as is used with $\alpha = 1$, as shown previously in Figure 6.6.

For future work, the quantized Bark-band residual noise gains could be interpolated up or down in frequency, according to the pitch-scaling factor, β. The attack transients are also kept constant during pitch-scale modifications. The transient detector only locates short signals segments such as voice fricatives or drum attacks. Usually, these types of audio signals are not *pitched*; that is, they do not have a well defined pitch associated with them. Since these types of signals modeled as transients are not pitched, it is not necessary to alter these audio segments when performing pitch-scale modification on the entire signal.

6.2.2 Pitch-Scaling the Sinusoids

Because of the representation of sinusoidal modeling, it is relatively trivial changing the pitch of the modeled audio; simply alter all the sinusoidal frequencies from $f \rightarrow \beta \cdot f$. The only complication is how to handle the sinusoids in the frame that overlap the transient region, F_t.

Below 5 kHz and with no modifications, the original signal during the overlapping frame F_t is modeled as a sum of the synthesized sinusoids and the transients (disregarding the noise models for simplicity):

$$x(n) = s_1(n) + t(n) \qquad n = 0, \ldots, S-1 \tag{6.1}$$

where $x(n)$ is the original input signal, $s_1(n)$ is the synthesized sinusoids with no time-scale modification ($\alpha = 1$), $t(n)$ is the transform-coded transient signal, and S is the frame length. The quantized parameters are transmitted to the decoder to synthesize $s_1(n)$ and $t(n)$ over the frame F_t. The phases of $s_1(n)$ had to be maintained during this frame in order to constructively add to $t(n)$ to recover an approximate version of $x(n)$. The phase coherency is also necessary when performing time-scale modification, as was discussed earlier in Section 6.1.3.

But when performing pitch-scale modifications, it is no longer necessary to maintain phase matching during the frame of overlap, F_t, between sines and transients. Because the sinusoids have all been pitch-scaled, $s_1(n) \rightarrow s_\beta(n)$, Equation (6.1) no longer holds. The phase information for these sinusoids, estimated with no pitch-scale modification at the encoder, are no longer relevant since their frequencies have been scaled at the decoder. Also, since the sinusoids have been pitch-scaled and the transients have *not* been pitch-scaled, there is no phase correction that could phase-match these two signals. A possible fix to this situation would be to either:

- Pitch-scale the sinusoids back to their original frequencies during the frames of overlap with the transients
- Pitch-scale the transients to match the pitch-scaled sinusoids

But, neither of these fixes would work. The first approach would generate audible frequency chirps when sweeping the sinusoidal frequencies from $\beta \cdot f \rightarrow f$ over the period of a frame. The second

approach has not been shown possible in the compressed domain. Perhaps the closest published result has been to perform frequency domain pitch-scaling on DFT magnitudes and phases Ferreira (1998). But in that study, poor results were reported when attempting to scale "*wideband signals, such as impulsive noise and sound attacks*". And, the modifications were performed on 2x oversampled DFT data, not critically sampled MDCT data.

In practice, as long as the transient detector is robust and only detects true attack transients, high quality pitch-scale modification is obtained by scaling only the sinusoids and *not* pitch-scaling the transients. By not having the pitch-scaled sinusoids phase-locked to the transients, there is the possibility of some very slight discontinuities. But psychoacoustically, these discontinuities will be much less perceptible than any artifacts due to quickly scaling the sinusoidal frequencies from $\beta \cdot f \rightarrow f$. Due to pre-masking effects of the transient attack (Zwicker and Fastl, 1990), these possible discontinuities are even less perceptible.

Another possible solution, yet not as elegant, would be to perform pitch-scale modification by first performing compressed-domain time-scale modification, then decoding the signal into raw audio samples, and then performing sample-rate conversion to effectively alter the pitch. Many commercial perform pitch-scale modifications in this manner; first, use a standard time-domain time-scale modification algorithm (Roucos and Wilgus, 1985; Verhelst and Roelands, 1993) to stretch the signal slower by α. Then, resample the slower signal at a sample rate of f_s/α using any number of standard techniques (Smith and Gossett, 1984). While this approach would not allow all of the processing to be performed in the compressed domain, it would alleviate any problems of overlapping pitch-scaled sinusoids with non-pitch-scaled transients. On the other hand, by using standard sample rate conversion post-processing algorithms, transient widths will be altered as function of the pitch scaling factor.

6.3 Conclusions

This chapter showed that pitch and time-scale modifications are possible in the compressed audio domain. Some amount of inverse quantization is required before the modifications can take place, but the inverse quantization requires a very low amount of complexity. Because the transient and non-transient regions are encoded separately, high quality time-scale modifications are possible that only stretch the non-attack regions. Pitch scaling modifications are also possible with almost no added complexity to the decoder.

Chapter 7

Conclusions and Future Research

The goal of this thesis research has been to create a set of audio representations that can enable both high quality data compression at very low bitrates, and to simultaneously allow for compressed domain processing. By using a mix of parametric (sines and noise modeling) and non-parametric (transform coding) representations, this thesis hopefully comes closer to a real-world solution. At all stages of the research, data compression quality had to be exchanged for the ability to modify the compressed data, and vice-versa (Levine and Smith, 1998). In the end, after all the trade-offs, this thesis shows the final results. In this final chapter, the conclusions will be drawn from each of the three audio models of {sines,transients,noise}, along with the compressed domain processing results. Afterwards, possible future directions for research in this field will be discussed.

7.1 Conclusions

While the sines, noise, and transient models presented in this work are not novel audio representations by themselves, their combination together, along with the improvements and quantization methods are novel. In the next few subsections, the improvements for each of these representations will be summarized.

7.1.1 Multiresolution Sinusoidal Modeling

Multiresolution sinusoidal modeling helps increase the audio quality for polyphonic music input signals. While the loudest attack transients are encoded using transform coding, there are not enough bits to encode *all* of the minor transients with transform coding. With shorter analysis windows at higher frequencies, pre-echo artifacts at these minor transients are significantly reduced.

Once the sinusoidal analysis is performed, the next step is to choose which sinusoids to keep, followed by quantizing the remaining sinusoids. It is desirable to only keep the sinusoidal parameters that are modeling only *true* sinusoids. Any parameters that are modeling noise are to be eliminated;

later, the noise energy those sinusoids were attempting to represent will be modeled by a noise model using many fewer bits per region of time/frequency. The decision to keep sinusoids is based on two seemingly independent metrics: 1) sinusoidal trajectory length and 2) time-averaged signal-to-mask ratio (SMR). By including both the tracking length data and the perceptual relevance of each trajectory in the decision process, informal listening tests verify that essentially only true sinusoids remain. Further reductions in bitrate result from downsampling (in time) remaining sinusoidal trajectories that fall below a certain SMR threshold. After these sinusoidal selection and smoothing processes, the amplitude and frequency data are differentially scalar quantized, followed by Huffman encoding. For most audio inputs, multiresolution sinusoidal modeling is encoded using 8 to 12 kbps.

7.1.2 Transform-coded Transients

Transform coding for audio signals is a well established, and some would say mature, field of study. But using transform coding for only short regions of time centered about attack transients is novel. In addition, adaptively switching between transform coding and other parametric coding techniques (sines and noise modeling) is also novel. This approach works well because transform coding models all the signals effectively that sinusoidal and noise modeling *cannot*: attack transients. Because transform coding is only used over very short segments of audio ($< 70msec$), compressed domain time-scale modification is possible by merely translating these short attack transients rather than stretching them.

While using short-time transform coding for transient regions is an effective representation, the bitrate can be prohibitively high. One method to reduce the transform coding bitrate is to limit the time-frequency region in which transform coding is used. The region of transform coding of attack transients is divided into 24 short (256 point at 44.1 kHz) 50% overlapping MDCT windows. Initially, all 24 windows encode MDCT coefficients from 0 to 16 kHz. To conserve bits, some of the 24 windows farther away from the exact transient attack time will only encode MDCT coefficients from 0 to 5 kHz. The remaining bandwidth from 5 to 16 kHz in these windows is represented with noise modeling instead. In order to keep the bitrate even lower, psychoacoustic masking thresholds can be altered, thus allowing more transform coding quantization noise to be added. Another approach to lowering the bitrate is to lower the total number of transients; this can be done by altering the sensitivity in the transient detector. In practice, transform-coded transients require 4 to 16 kbps, but the range could be even wider if extreme cases of audio are considered (from only long steady-state tones to only fast drum hits).

7.1.3 Noise Modeling

Modeling noisy audio signals parametrically has been well known for many years. In this system, two separate noise signals are formed: 1) a residual noise signal that is the difference between the

original signal and synthesized sinusoids, all below 5 kHz and during only non-transient time regions, and 2) a noise signal that models the original signal above 5 kHz and during only non-transient time regions. All energy above 5 kHz during the non-transient regions is modeled as noise in order to lower the overall bitrate. While there may be some cases of isolated tonal signals above 5 kHz, this noise substitution method is a reasonable approximation for most music at 32 kbps/ch and lower. In the future, this 5 kHz bandwidth cutoff could be signal adaptive in order to accommodate very tonal input signals better. If users desire this sines+transients+noise system for only modifications, and are willing to use higher bitrates ($\gg 32kbps$), then sinusoids could be modeled over the entire range of the original audio bandwidth.

For both noise signals, Bark-band noise modeling, as introduced by Goodwin (1997), is used. The input noise signal is split into Bark spaced subbands, and each subband has its average energy sampled at a certain frame rate. Below 5 kHz, this frame rate is approximately 86 times a second, while above 5 kHz, the noise gain frame rate is approximately 345 times a second. A higher frame rate is needed at the higher frequencies in order to preserve sharp attacks. In order to reduce the data rate above 5 kHz, the signal vector comprising a single Bark-band's energy gain over time is smoothed using line segment approximation techniques. After differential scalar quantization and Huffman coding of the noise gains below 5 kHz and the line segment breakpoints above 5 kHz, the bitrate requirements for the total noise modeling is approximately 5 to $7kbps$.

7.1.4 Compressed Domain Processing

High quality compressed-domain time-scale modification is now possible because the transients are modeled separately from the non-transient regions of the input signal. By *stretching* the non-transient, sines+noise signals while *translating* the transform-coded transient regions, attack transients remain as sharp as the original, even while the rest of the signal is time-scaled. This quality of time-scaling, even up to time-scaling factors of 2 to 3, is difficult for most of the commercial time-scale modification systems. Even the phase vocoder, which usually has the highest quality time-scale modification results, does not handle transients well. Compressed-domain pitch-scale modification is also possible, just by merely altering the frequencies of the sinusoidal modeling parameters. The pitch-scaling properties are not novel, given that even the first sinusoidal modeling algorithms were able to pitch-scale in this manner.

7.2 Improvements for the System

The overall goal for this thesis was to match the results of audio data compression, while allowing for compressed domain processing. If this research were to continue on, several straightforward improvements would be made.

The first improvement would be to include a bit reservoir technique, similar to that in MPEG-I

layer III (Brandenburg, 1995) and MPEG-AAC (Bosi *et al.*, 1997). But, this shared bit reservoir would hold available bits for the sinusoidal models, the transient models, and both noise models. If any one of these components were deemed more perceptually relevant at a given frame than the others, then that model would take bits out of the bit reservoir.

Another improvement, as stated in the previous section, would be to adaptively set the high cutoff frequency for sinusoidal modeling. For example, if a long, steady, clarinet tone were the input signal, it would be more perceptually important to include all of the harmonics present. For most polyphonic music, tonal elements cannot be easily discerned above 5 kHz, but an adaptive algorithm should make this decision.

Much improvement could be made in the transform coding section. If the rate/distortion loops were implemented in the current transform coder as they are in MPEG-AAC (Bosi *et al.*, 1997), along with all the complex, multiple Huffman encodings, then the bitrates associated with the transient models could drop significantly. But due to time constraints, only a relatively simple transform coder is implemented for this thesis. In addition, more work could be used for an algorithm that dynamically decided how much bandwidth to allocate for transform coding versus noise modeling in each of the short windows of transient coding. This would allow the transform coding to shape the time/frequency plane precisely around a transient, as was alluded to earlier in Figure 4.8.

7.3 Audio Compression Directions

Transform coding works well for most audio inputs at higher bitrates. In this context, higher bitrates are defined to be the range 64 to $128 kbps/ch$. By first transforming windowed audio samples into the MDCT domain, and then quantizing according to masking threshold criteria, it can handle almost all classes of audio inputs. As long as there are enough bits to meet the psychoacoustic thresholds, the inverse quantized signal will almost always sound identical to the original. The problem with transform coding arises when the desired bitrates drop below $64 kbps/ch$.* Suddenly, the psychoacoustic quantization algorithms are bit-starved, and the results are suboptimal. The current alternative is usually to reduce the audio bandwidth such that high quality audio is obtained, even though it is over a smaller frequency range.

At the bitrates investigated in this thesis, between 20 and $32 kbps/ch$, it is not currently possible to use transform coding (or any other method) to deliver perceptually lossless, CD quality audio (44.1 kHz sampling rate, 16 bits/sample). If the bitrate is constrained to be lower than $32 kbps/ch$, it must be assumed that *some* coding artifacts will be introduced. The next question must be: *What are the least objectionable coding artifacts to introduce?* Limiting the bandwidth of the transform-coded audio can be considered an artifact. The coding artifacts from sinusoidal modeling sound

*In late 1998, this bitrate of $64 kbps/ch$ seems like the point at which perceptually lossless transform coders seem to have reached a limit, from the results of MPEG-AAC listening tests. But no doubt, the future will deliver more complex and better sounding algorithms that will push this limit lower.

different than the coding artifacts from noise modeling. The artifacts heard from a bit-starved transform coder also sound different.

Listening tests at these lower bitrates can become a much more difficult issue than the tests at the higher rates. At the higher bitrates ($\geq 64 kbps/ch$), perceptually lossless quality can be obtained. If a group of trained listeners cannot tell the difference between the original and the encoded version, then the tests are complete. But at very low bitrates, one can certainly determine which is the original and which is the encoded version. The goal is to decide which of the many encoded versions sound *better* than the other encoded versions. This issue can be a tricky one, since it may result in preferences that can vary from person to person. Some may like a wide audio bandwidth coupled with poor pre-echo artifacts, while others may want a smaller audio bandwidth with very few audible artifacts. Some may consider noise modeling at high frequencies reasonable, while others would rather have transform coding used over higher frequencies with more quantization noise injected.

Adding yet another layer of complexity to this quality issue is the question of what signals to use as test inputs? At the higher bitrates, very tough signals like solo castanets, pitchpipe, and glockenspiel were used. Throughout the MPEG standardization process, it is hard to imagine that some of the encoding modules were not included and/or tuned for the explicit purpose of improving the test scores for one particularly tough input signal. For these lower bitrates, would it be better to use average music heard over the internet (or other widely used transmission media) as the input test signals?

The answers to these questions of quality metrics and testing are certainly out of the scope of this thesis. But, they will no doubt be problematic as long as people attempt to compress and transmit music over transmission media with bitrates so low that encoders *cannot* achieve perceptually lossless quality.

Parallel to the question of quality at these lower bitrates, is the question of compressed domain modifications. Will these compressed domain audio modifications become needed or required in the future? This is a difficult one to predict. In the video compression world, compressed domain modifications are already being used widely. If compressed domain modifications for audio do become an important feature in the future, hopefully the work in this thesis will make a contribution towards the field.

Appendix A

The Demonstration Sound CD

The attached CD contains several sounds demonstrating the topics covered in this thesis. First, there will be two sets of mono sounds compressed to 32 kbps, and then compared to MPEG-AAC at the same bitrate. The MPEG-AAC encoder was using source code from FhG, which used a build from September 1998. After the data compression comparison, the audio input signals will be segregated into their independent sines, residual noise, transients, and high frequency noise signals. To end the CD, demonstrations of the compressed-domain time-scaling and pitch-scaling abilities will be played. These sounds files are also accessible from `http://www-ccrma.stanford.edu/~scottl`.

track	description
1	original version of Mozart's *Le Nozze di Figaro*
2	*Figaro*, compressed using MPEG-AAC at 32 kbps
3	*Figaro*, compressed using sines+transients+noise at 32 kbps
4	*Figaro*, just multiresolution sinusoids
5	*Figaro*, just residual Bark-band noise
6	*Figaro*, just transform-coded transients
7	*Figaro*, just high frequency Bark-band noise
8	original version of *It Takes Two* by Rob Base & D.J. E-Z Rock
9	*It Takes Two*, compressed using MPEG-AAC at 32 kbps
10	*It Takes Two*, compressed using sines+transients+noise at 32 kbps
11	*It Takes Two*, just multiresolution sinusoids
12	*It Takes Two*, just residual Bark-band noise
13	*It Takes Two*, just transform-coded transients
14	*It Takes Two*, just high frequency Bark-band noise
15	*It Takes Two*, time-scale modified slower by $\alpha = 2$ with sines+transients+noise
16	*It Takes Two*, time-scale modified slower by $\alpha = 2$ with CoolEdit (commercial software)
17	*It Takes Two*, looped with varying time-scaling factors: $\alpha = \{2.0, 1.6, 1.2, 1.0, 0.8, 0.6, 0.5\}$
18	*It Takes Two*, looped with varying pitch-scaling factors: $\beta = \{0.89, 0.94, 1.00, 1.06, 1.12\}$

References

Ali, M. (1996). *Adaptive Signal Representation with Application in Audio Coding.* Ph.D. thesis, University of Minnesota.

Anderson, D. (1996). Speech analysis and coding using a multi-resolution sinusoidal transform. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Atlanta*, pp. 1037–1040.

Atal, B. and Remde, J. (1982). A new model for LPC excitation for producing natural sounding speech at low bit rates. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing*, pp. 614–617.

Atal, B. and Schroeder, M. (1967). Adaptive predictive coding of speech signals. *Conf. Comm. and Proc.*, pp. 360–361.

Boland, S. and Deriche, M. (1995). High quality audio coding using multipulse LPC and wavelet decomposition. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Detroit*, pp. 3067–3069.

Boland, S. and Deriche, M. (1996). Audio coding using the wavelet packet transform and a combined scalar-vector quantization. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Atlanta*, pp. 1041–1044.

Bosi, M., Brandenburg, K., Quackenbush, S., L.Fielder, Akagiri, K., H.Fuchs, M.Dietz, J.Herre, G.Davidson, and Y.Oikawa (1997). ISO-IEC MPEG-2 Advanced Audio Coding. *J. Audio Eng. Soc.*, 45(10).

Brandenburg, K. (1987). OCF - A new coding algorithm for high quality sound signals. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing.*

Brandenburg, K. (1995). Overview of MPEG-Audio: Current and future standards for low bit-rate audio coding. *Proc. of the 99th Convention of the Audio Engineering Society.* Preprint 4130.

Brandenburg, K., Eberlein, E., Herre, J., and Edler, B. (1992). Comparison of filterbanks for high quality audio coding. *IEEE International Symposium on Circuits and Systems*, 3:1336–1339.

Brandenburg, K., Herre, J., Johnston, J., Mahieux, Y., and Schrohdtveder, E. (1991). ASPEC–adaptive spectral perceptual entropy coding of high quality music signals. 39(380).

Brandenburg, K., Langenbucher, G., Schramm, H., and Seitzer, D. (1982). A digital signal processor for real time adaptive transform coding of audio signals up to 20 khz bandwidth. *Proceedings of the ICCC*, pp. 474–477.

Brandenburg, K. and Stoll, G. (1994). ISO-MPEG-1 audio: A generic standard for coding of high-quality digital audio. *J. Audio Eng. Soc.*, 42(10).

Bristow-Johnston, R. (1996). Wavetable synthesis 101: A fundamental perspective. *Proc. of the 101st Convention of the Audio Engineering Society*. Preprint 4400.

Broadhead, M. and Owen, C. (1995). Direct manipulation of MPEG compressed digital audio. *Proceedings of the Third ACM International Conference on Multimedia*.

Burt, P. and Adelson, E. (1983). The laplacian pyramid as a compact image code. *IEEE Trans. Communications*, 31(4).

Carnero, B. and Drygajlo, A. (1997). Perceptual speech coding using time and frequency masking constraints. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Munich*.

Chang, S., Chen, W., and Messerchmitt, D. (1992). Video compositing in the DCT domain. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, San Francisco*.

Chen, J. (1997). A candidate coder for the ITU-T's new wideband speech coding standard. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Munich*.

Chowning, J. (1973). The synthesis of complex audio spectra by means of frequency modulation. *J. Audio Eng. Soc.*, 21(7):526–534.

Coifman, R., Meyer, Y., and Wickerhauser, M. (1992). *Wavelets and their Applications*, pp. 153–178. Jones and Bartlett.

Covell, M., Withgott, M., and Slaney, M. (1998). Mach1 : Nonuniform time-scale modification of speech. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Seattle*.

Crossman, A. (1993). A variable bit rate audio coder for videoconferencing. *IEEE workshop on speech coding for telecommunications*, pp. 7–8.

Cumminskey, P. (1973). Adaptive quantization in differential pcm coding of speech. In *Bell Syst. Tech. J.*, vol. 52.

Dattorro, J. (1998). Private communication with Jon Dattorro.

Dehéry, Y. F., Stoll, G., and Kerkof, L. (1991). MUSICAM source coding for digital sound. *Symp. Rec. "Broadcast Sessions" of the 17th Int. Television Symp.*

Dobson, W. K., Yang, J. J., Smart, K. J., and Guo, F. K. (1997). High quality low complexity scalable wavelet audio coding. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Munich*.

Dolson, M. (1986). The phase vocoder: A tutorial. *Computer Music J.*, 10(4).

Dudley, H. (1939). The vocoder. *Bell Labs. Rec.*, 17:122.

Edler, B. (1992). Aliasing reduction in sub-bands of cascaded filter banks with decimation. *Electronic Letters*, 28(12):1104–1106.

Edler, B. (1996). Current status of the MPEG-4 audio verification model development. *Proc. of the 101st Convention of the Audio Engineering Society*. Preprint 4376.

Edler, B., Purnhagen, H., and Ferekidis, C. (1996). ASAC - analysis/synthesis codec for very low bit rates. *Proc. of the 100th Convention of the Audio Engineering Society*. Preprint 4179.

Ellis, D. and Vercoe, B. (1991). A wavelet-based sinusoid model of sound for auditory signal separation. *Proc. 1991 Int. Computer Music Conf., Montreal*, pp. 86–89.

Ferreira, A. (1998). A new frequeny domain approach to time-scale expansion of audio signals. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Seattle*.

Fielder, L. (1999). *The Audio Engineers' Reference Book*, chap. Perceptual Audio Coding. Focal Press, 2nd edn.

Fielder, L., Bosi, M., Davidson, G., Davis, M., Todd, C., and Vernon, S. (1996). AC-2 and AC-3: Low complexity transform-based audio coding. In N. Gilchrist and C. Grewin, editors, *Collected Papers on Digital Audio Bit-Rate Reduction*, pp. 54–72. Audio Engineering Society.

Fliege, N. and Zolzer, U. (1993). Multi-complementary filter bank. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Minneapolis*.

Gbur, U., Werner, M., and Dietz, M. (1996). Realtime implementation of an ISO/MPEG layer 3 encoder on pentium pcs. *Audio Eng. Soc. Convention*.

George, E. B. and Smith, M. (1992). Analysis-by-synthesis / overlap-add sinusoidal modeling applied to the analysis and synthesis of musical tones. *J. Audio Eng. Soc.*, 40(6):497–516.

Gersho, A. and Gray, R. (1992). *Vector Quantization and Signal Compression*. Kluwer Academic Publishers.

Goodwin, M. (1996). Residual modeling in music analysis-synthesis. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Atlanta*, pp. 1005–1008.

Goodwin, M. (1997). *Adaptive Signal Models: Theory, Algorithms, and Audio Applications*. Ph.D. thesis, University of California, Berkeley.

Goodwin, M. and Kogon, A. (1995). Overlap-add synthesis of nonstationary sinusoids. *Proc. 1995 Int. Computer Music Conf., Banff.*

Goodwin, M. and Rodet, X. (1994). Efficient Fourier synthesis of nonstationary sinusoids. *Proc. 1994 Int. Computer Music Conf., Århus.*

Goodwin, M. and Vetterli, M. (1996). Time-frequency signal models for music analysis, transformation, and synthesis. *IEEE SP International Symposium on Time-Frequency and Time-Scale Analysis.*

Grey, J. (1975). *An Exploration of Musical Timbre.* Ph.D. thesis, Stanford University.

Griffin, D. and Lim, J. (1988). Multiband excitation vocoder. *IEEE Trans. Acoustics, Speech, Signal Processing*, 36:1223–1235.

Grill, B., Edler, E., Funken, R., Hahn, M., Iijama, K., Iwakami, N., Lee, Y., Moriya, T., Ostermann, J., Nakajima, S., Nishiguchi, M., Nomura, T., Oomen, W., Purnhagen, H., Scheirer, E., Tanaka, N., Tan, A. P., Taori, E., and MPEG-2 AAC authors (1998*a*). ISO/IEC 1/SC N2203PAR: Information technology - very low bitrate audio-visual coding, subpart 2: Parametric coding.

Grill, B., Edler, E., Funken, R., Hahn, M., Iijama, K., Iwakami, N., Lee, Y., Moriya, T., Ostermann, J., Nakajima, S., Nishiguchi, M., Nomura, T., Oomen, W., Purnhagen, H., Scheirer, E., Tanaka, N., Tan, A. P., Taori, E., and Authors, M.-. A. (1998*b*). ISO/IEC 1/SC N2203: Information technology - very low bitrate audio-visual coding.

Hamdy, K., Ali, M., and Tewfik, H. (1996). Low bit rate high quality audio coding with combined harmonic and wavelet representations. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Atlanta.*

Hellman, R. (1972). Asymmetry of masking between noise and tone. *Perception and Psychophysics*, pp. 241–246.

Herre, J. and Eberlin, E. (1993). Evaluation of concealment techniques for compressed digital audio. *Proc. of the 94th Convention of the Audio Engineering Society.* Preprint 3460.

Herre, J. and Johnston, J. (1996). Enhancing the performace of perceptual audio coders by using temporal noise shaping. *J. Audio Eng. Soc.*, 44.

Horner, A., Cheung, N., and Beauchamp, J. (1995). Genetic algorithm optimization of additive synthsis envelope breakpoints and group synthesis parameters. *Proc. 1995 Int. Computer Music Conf., Banff*, pp. 215–222.

Immink, K. (1998). The compact disc story. *J. Audio Eng. Soc.*, 46(5):458–465.

ISE/IEC JTC 1/SC 29/WG 11 (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.

Itakura, F., Koboyashi, T., and Honda, M. (1982). A hardware implementation of a new narrow to medium band speech coding. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing*, pp. 1964–1967.

Iwakami, N. and Moriya, T. (1996). Transform-domain weighted interleave vector quantization (Twin VQ). *Proc. of the 101st Convention of the Audio Engineering Society*. Preprint 4377.

Iwakami, N., Moriya, T., and Miki, S. (1995). High-quality audio coding at less than 64 kbit/s by using twinvq. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Detroit*, pp. 937–940.

Johnston, J. (1988*a*). Estimation of perceptual entropy using noise masking criteria. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing*, pp. 2524–2527.

Johnston, J. (1988*b*). Transform coding of audio signals using perceptual noise criteria. *IEEE Journal on Selected Areas in Communications*.

Johnston, J. (1998). Private communications with J. Johnston.

Johnston, J. and Crochiere, R. (1979). An all-digital, commentary grade subband coder. *J. Audio Eng. Soc.*, 27.

Johnston, J. and Ferreira, A. (1992). Sum-difference stereo transform coding. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, San Francisco*, pp. 569–571.

Johnston, J., Herre, J., Davis, M., and Gbur, U. (1996*a*). MPEG-2 NBC audio – stereo and multichannel coding methods. *Proc. of the 101st Convention of the Audio Engineering Society*, 44. Preprint 4383.

Johnston, J., Sinha, D., Dorward, S., and Quackenbush, S. (1996*b*). AT&T perceptual audio coding (PAC). *Collected Papers on Digital Audio Bit-Rate Reduction*, pp. 73–82.

Kataoka, A., Moriya, T., and Hayashi, S. (1993). An 8 kbit/s speech coder based on conjugate struture CELP. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Minneapolis*, pp. 592–595.

K.N.Hamdy, A.H.Tewfik, T.Chen, and Takagi, S. (1997). Time-scale modification of audio signals with combined harmonic and wavelet representations. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Munich*.

Kudumakis, P. and Sandler, M. (1995). On the performance of wavelets for low bit rate coding of audio signals. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Detroit*, pp. 3087–3090.

Laroche, J. and Dolson, M. (1997). Phase-vocoder: About this phasiness business. *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, New Paltz, NY.*

Le Gall, D. (1991). MPEG: A video compression standard for multimedia applications. *Communications of the ACM*, 34(4):46–58.

Lee, S., Kim, H. D., and Kim, H. S. (1997). Variable time-scale modification of speech using transient information. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Munich.*

Levine, S. (1996). Effects processing on audio subband data. *Proc. 1996 Int. Computer Music Conf., Hong Kong.*

Levine, S. and Smith, J. O. (1998). A sines+transients+noise audio representation for data compression and time/pitch-scale modifications. *Proc. of the 105th Convention of the Audio Engineering Society.* Preprint 4781.

Levine, S., Verma, T., and Smith, J. (1998). Multiresolution sinusoidal modeling for wideband audio with modifications. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Seattle.*

Lin, X. and Steele, R. (1993). Subband coding with modified multipulse LPC for high quality audio. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Minneapolis*, pp. 201–204.

Lokhoff, G. (1996). The digital compact cassette. *Collected Papers on Digital Audio Bit-Rate Reductions.*

Maitre, X. (1988). 7 khz audio coding within 64 kbit/s. *IEEE Journal of Selected Areas in Communications*, 6(2):283–298.

Malvar, H. (1990). Lapped transofrms for efficient transform/subband coding. *IEEE Trans. Acoustics, Speech, Signal Processing*, 38:969–978.

Malvar, H. (1991). Extended lapped transforms: fast algorithms and applications. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Toronto*, pp. 1797–1800.

Malvar, H. (1992). *Signal Processing with Lapped Transforms.* Artech House.

Masri, P. and Bateman, A. (1996). Improved modelling of attack transients in musical analysis-resynthesis. *Proc. 1996 Int. Computer Music Conf., Hong Kong.*

McAualay, R. J. and Quatieri, T. F. (1988). Computationally efficient sine-wave synthesis and its application to sinusoidal transform coding. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing.*

McAulay, R. and Quatieri, T. (1986*a*). Speech analysis/synthesis based on a sinusoidal representation. *IEEE Trans. Acoustics, Speech, Signal Processing.*

McAulay, R. and Quatieri, T. (1986*b*). Speech transformations based on a sinusoidal representation. *IEEE Trans. Acoustics, Speech, Signal Processing*, 34.

McAulay, R. and Quatieri, T. (1995). *Speech Coding*, chap. 4, pp. 121–173. Elsevier Science B.V.

Meng, J. and Chang, S. (1996). Tools for compressed-domain video indexing and editing. *SPIE Conference in Storage and Retrieval for Image and Video Database*, 2670.

Mermelstein, P. (1988). G. 722, a new CCITT coding standard for digital transmission of wideband audio signals. *IEEE Communications Magazine*, 26(1):8–15.

Moore, B. and Glasberg, B. (1996). A revision of Zwicker's loudness model. *Acustica*, 82:335–346.

Moorer, J. (1978). The use of the phase vocoder in computer music applications. *J. Audio Eng. Soc.*, 26(1):42–45.

Moorer, J. (1979). About this reverberation business. *Computer Music J.*, 3(2):13–28.

Moriya, T., Iwakami, N., Jin, A., Ikeda, K., and Miki, S. (1997). A desgin of transform coder for both speech and audio signals at 1 bit/sample. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Munich*.

Moulines, E. and Charpentier, F. (1990). Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones. *Speech Communication*, 9(5/6).

Moulines, E. and Laroche, J. (1995). Non-parametric techniques for pitch-scale and time-scale modification of speech. *Speech Communication*, 16:175–205.

Nebeker, N. (1998). *Fifty Years of Signal Processing: The IEEE Signal Processing Society and its Technologies 1948-1998*. The IEEE Signal Processing Society.

Nussbaumer, H. (1981). Pseudo qmf filter bank. *IBM Technical disclosure Bulletin*, 24:3081–3087.

Painter, T. (1997). A review of algorithms for perceptual coding of digital audio signals. *DSP-97*.

Park, S., Kim, U., Kim, S., and Seo, Y. (1997). Multi-layer bit-sliced bit-rate scalable audio coding. *Proc. of the 103rd Convention of the Audio Engineering Society*. Preprint 4520.

Pennebaker, W. (1992). *JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold.

Phillipe, P. (1995). On the choice of wavelet filters for audio compression. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Detroit*.

Phillips, G. (1968). Algorithms for piecewise straight line approximation. *Computer Journal*, 11:211–212.

Portnoff, M. (1976). Implementation of the digital phase vocoder using the fast Fourier transform. *IEEE Trans. Acoustics, Speech, Signal Processing*, 24(3):243–248.

Princen, J. and Bradley, A. (1986). Analysis/synthesis filter bank design based on time domain aliasing cancellation. *IEEE Trans. Acoustics, Speech, Signal Processing*, 34(5):1153–1161.

Rodet, X. and Depalle, P. (1992). Spectral envelopes and inverse FFT synthesis. *Proc. of the 93rd Convention of the Audio Engineering Society.* Preprint 3393.

Rodriguez-Hernandez, M. and Casajus-Quiros, F. (1994). Improving time-scale modification of audio signals using wavelets. *ICSPAT*, 2:1573–1577.

Rothweiler, J. H. (1983). Polyphase quadrature filters - a new subband coding technique. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Boston*, pp. 1280–1283.

Roucos, S. and Wilgus, A. (1985). High quality time-scale modification of speech. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Detroit.*

S. Boland, M. D. (1997). New results in low bitrate audio coding using a combined harmonic-wavelet representation. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Munich.*

Scheirer, E. and Ray, L. (1998). The MPEG-4 structured audio standard. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Seattle.*

Schröder, E. and Ratliff, P. (1996). Digital audio broadcasting (DAB). *Collected Papers on Digital Audio Bit-Rate Reductions.*

Schroder, E. F. and Vossing, W. (1986). High-quality digital audio encoding with 3.0 bits/sample using adaptive transform coding. *Proc. of the 80th Convention of the Audio Engineering Society.* Preprint 2321.

Schulz, D. (1996). Improving audio codecs by noise substitution. *J. Audio Eng. Soc.*, 44(7):593–598.

Serra, X. (1989). *A System for Sound Analysis/Transformation/Synthsis based on a Determistic plus Stochastic Decomposition.* Ph.D. thesis, Stanford University.

Serra, X. (1997). *Musical sound modeling with sinusoids plus noise.* http://www.iua.upf.es/~sms/.

Serra, X. and Smith, J. (1990). Spectral modeling synthesis: A sound analysis / synthesis system based upon a deterministic plus stochastic decomposition. *Computer Music J.*, 14(4):12–24.

Shen, B. and Sethi, I. (1996). Block-based manipulations on transform-compressed images and videos. *Multimedia Systems Journal.*

Shlien, S. (1997). The modulated lapped transform, its time-varying forms, and its applications to audio coding standards. *IEEE Trans. Speech and Audio Processing*, 5(4):359–366.

Singhal, S. (1990). High quality audio coding using multipulse LPC. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Albuquerque*, pp. 1101–1104.

Sinha, D. and Johnston, J. (1996). Audio compression at low bit rates using a signal adaptive switched filterbank. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Atlanta*, pp. 1053–1056.

Sinha, D. and Tewfik, A. (1993). Low bit rate transparent audio compression using adapted wavelets. *IEEE Trans. Signal Processing*, 41(12).

Smith, B. (1994). Fast software processing of motion JPEG video. *Proceedings of the Second ACM International Conference on Multimedia*, pp. 77–88.

Smith, J. (1996). Physical modeling synthesis update. *Computer Music J.*, 20(2):44–56.

Smith, J. and Serra, X. (1987). PARSHL: An analysis/synthesis program for non-harmonic sounds based on a sinusoidal representation. *Proc. 1987 Int. Computer Music Conf., Champaign-Urbana*.

Smith, J. O. and Gossett, P. (1984). A flexible sampling-rate conversion method. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, San Diego*.

Soulodre, G., Grusec, T., Lavoie, M., and Thibault, L. (1998). Subjective evaluation of state-of-the-art two-channel audio codecs. *J. Audio Eng. Soc.*, 46(3):164–177.

Spanias, A. (1994). Speech coding: A tutorial review. *Proceedings of the IEEE*, 82(10):1541–1582.

Sreenivas, T. and Dietz, M. (1998). Vector quantization of scale factors in advanced audio coder (AAC). *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Seattle*.

Stikvoort, E. (1986). Digital dynamic range compressor for audio. *J. Audio Eng. Soc.*, 34(1).

Stoll, G. (1996). ISO-MPEG-2 audio: A generic standrad for the coding of two-channel and multi-channel sound. *Collected Papers on Digital Audio Bit-Rate Reduction*, pp. 43–53.

T. Sporer, K. Brandenburg, B. E. (1992). The use of multirate filter banks for coding of high quality digital audio. *EUSIPCO*, 1:211–214.

Tang, B., Shen, A., Alwan, A., and Potite, G. (1997). A perceptually based subband speech coder. *IEEE Trans. Speech and Audio Processing*, 5(2):131–140.

Tang, B., Shen, A., Pottie, G., and Alwan, A. (1995). Spectral analysis of subband filtered signals. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Detroit*.

Tewfik, A. and Ali, M. (1993). Enhanced wavelet based audio coder. *Asilomar-93*.

Theile, G., Still, G., and Link, M. (1988). Low bitrate coding of high-quality audio signals: An introduction to the MASCAM system. *EBU Review - Technical*, (230):158–181.

Thomson, D. (1982). Spectral estimation and harmonic analysis. *Proceedings of the IEEE*, 70(9).

Todd, C., Davidson, G., Davis, M., Fielder, L., Link, B., and Vernon, S. (1994). AC-3: Flexible perceptual coding for audio transmission and storage. *Proc. of the 96th Convention of the Audio Engineering Society*. Preprint 3796.

Tsutsui, K., Suzuki, H., Shimoyoshi, O., Sonohara, M., Akagiri, K., and Heddle, R. (1992). Atrac: Adaptive trasform acoustic coding for minidisc. *Audio Eng. Soc. Convention*. Preprint 3456.

U.S. Advanced Television Systems Committee (1992). Digital audio and ancillary data services for an advanced television service. *Dpc. T3/186*.

Vaidyanathan, P. P. (1993). *Multirate Systems and Filter Banks*. Prentice-Hall.

Verhelst, W. and Roelands, M. (1993). An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Minneapolis*.

Verma, T. and Meng, H. (1998). An analysis/synthesis tool for transient signals that allows a flexible sines+transients+noise model for audio. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Seattle*.

Vetterli, M. and Kovačević, J. (1995). *Wavelets and Subband Coding*. Prentice Hall.

Wiese, D. and Stoll, G. (1990). Bitrate reduction of high quality audio signals by modelling the ears' masking threshold. *Proc. of the 89th Convention of the Audio Engineering Society*. Preprint 2970.

Zelinski, R. and Noll, P. (1977). Adaptive transform coding of speech signals. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing*, 25(4).

Zwicker, E. and Fastl, H. (1990). *Psychoacoustics, Facts, and Models*. Berlin: Springer-Verlag.