Pre-Echo Detection & Reduction

by

Kyle K. Iwai

S.B., Massachusetts Institute of Technology (1991)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1994

Author [illegible]
Department of Electrical Engineering and Computer Science
March 23, 1994

Certified by [illegible]
Jae S. Lim
Professor of Electrical Engineering
Thesis Supervisor

Accepted by [illegible]
Frederic R. Morgenthaler
Chairman, Departmental Committee on Graduate Students

Pre-Echo Detection & Reduction

by

Kyle K. Iwai

Submitted to the Department of Electrical Engineering and Computer Science
on March 23, 1994, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

In recent years, there has been an increasing interest in data compression for storage and data transmission. In the field of audio processing, various flavors of transform coders have successfully demonstrated reduced bit rates while maintaining high audio quality. However, there are certain coding artifacts which are associated with transform coding. The pre-echo is one such artifact. Pre-echoes typically occur when a sharp attack is preceded by silence.

This thesis examines the factors which contribute to a pre-echo, and discusses the method of pre-echo detection and reduction implemented on the MIT Audio Coder (MIT-AC) real-time system. The MIT-AC uses an adaptive window selection algorithm to switch between long and short transform windows. Long windows offer higher coding gains and greater frequency selectivity, while short windows reduce the length of a pre-echo. Due to temporal masking effects, pre-echoes which are sufficiently reduced in duration become inaudible to the human ear. Thus, by switching between long and short windows, the MIT-AC is able to maintain high coding gain while reducing the pre-echo effect.

Thesis Supervisor: Jae S. Lim
Title: Professor of Electrical Engineering

Acknowledgments

I would like to thank the many people who have made my six and a half years here a rewarding (and sometimes pleasurable) experience...

My advisor, Prof. Lim, for giving me the opportunity to work on this project.

Everyone in lab and at ATRP for their help and advice.

The guys at Theta Chi for giving me a life even when I should have been tooling.

Creighton for the late nights we spent working (and goofing off) together.

Barbara for her support and understanding, and her ability to make me smile.

Mom, Dad, my sister Seiko and my cat for being supportive (and supporting me for several years).

Contents

List of Figures

Chapter 1

Introduction

Transform coding, in conjunction with psycho-acoustic modeling, is a common technique used to compress digital audio [4, 1, 2, 5]. Yet the use of such a technique can lead to a coding artifact known as a "pre-echo". Pre-echoes typically appear in transient situations, situations where silence is broken by a sharp attack. For example, firecrackers, chimes and castanets all produce sharp attacks which can generate pre-echoes. Quantization noise added by the coding process is normally hidden within the signal. However, the coder assumes stationarity over the window length, an assumption which breaks down in a transient situation. The noise is unmasked in the silence preceding the attack, creating an audible artifact called a pre-echo. Looking at the waveform in the time domain, a pre-echo shows up as noise preceding the attack. To the human ear, this has the effect of blurring the attack, making it less sharp. In most cases, the attack is even preceded by a short hissing noise.

If the length of the noise can be shortened to under 5ms, psycho-acoustic experiments tell us that the noise will not be audible to the human ear. Using a shorter transform window shortens the length of the pre-echo. Therefore, one alternative is to use shorter windows for coding. Unfortunately, short windows also have poorer frequency selectivity and poorer coder efficiency than long windows. A more reasonable solution is to use an adaptive window length algorithm which normally uses longer windows but can switch to shorter windows in a transient situation.

In order for the adaptive window length selection to be effective, the algorithm must be able to look ahead and detect potentially troublesome passages. To this end, a simple detector was designed to find sharp transients in the audio waveform. The detector divides the audio into small sections and calculates the variance within each section. In a transient situation, the variance suddenly increases from one section to the next. So if the detector compares a section of audio with previous sections and finds a large increase in variance, the detector concludes it has found a transient that may lead to a pre-echo. The coder then switches to short windows to reduce the length of the pre-echo, rendering the artifact inaudible.

In order to reduce or eliminate pre-echoes it is of course useful to understand the factors that interact to produce a pre-echo. And in order to understand where these limitations come from, it is important to first understand the basic algorithm of a transform coder. Therefore, chapter 2 looks at transform coding and the different factors that contribute to pre-echoes. From there, chapter 3 discusses some methods for eliminating or reducing the pre-echo effect. In particular, chapter 3 details the adaptive window selection algorithm implemented on the MIT Audio Coder (MIT-AC). The results presented in chapter 4 show that the algorithm succeeds in reducing the duration (and thus the audibility) of pre-echoes. Chapter 5 closes with with some conclusions and offers suggestions for future work.

Chapter 2

Transform Coding & the Origin of Pre-Echo

A pre-echo is an artifact that arises from the transform coding process. Therefore, it is important to look at the general algorithm of a transform coder. Having done that, the rest of the chapter will examine the three factors which contribute to the formation of a pre-echo, namely the quantization noise, the window length, and the stationarity of the audio signal.

2.1 Transform Coding

In the discussion of pre-echo it is useful to understand the basic transform coding algorithm. The encoder accepts digital audio at a given sampling rate. A section of this digital audio is windowed and transformed into the frequency domain using some form of short-time spectral analysis. The encoder then determines the level of precision with which to quantize the different transform coefficients. The quantized coefficients get transmitted to the decoder which inverts the process in order to reconstruct the signal. The coder then windows and processes the next section of audio. For the purposes of discussion, the remainder of the chapter occasionally refers to

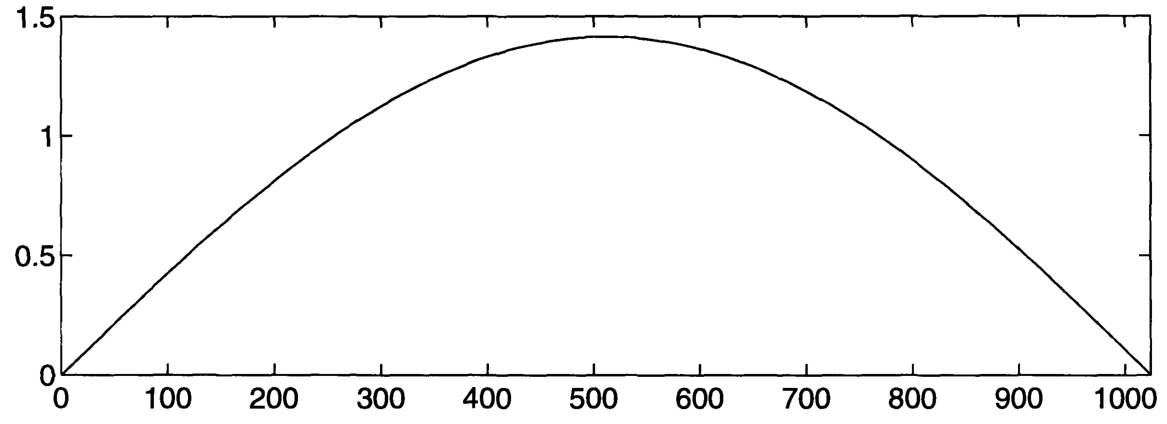


Figure 2-1: Raised Cosine Window

the specifics of the MIT Audio Coder (MIT-AC) as a general example of a transform coder.

2.1.1 Windowing

Given digital audio (for example, 16 bits/sample at 48 kHZ sampling rate), the first step is to window the signal. MIT-AC multiplies a section of digitized audio with the 1024-point raised-cosine window shown in figure 2-1. The particular transform used by MIT-AC requires each windowed section to overlap with the two adjacent sections by 50% as demonstrated in figure 2-2. Therefore, while each processed segment is 1024 samples in length, the effective processing rate is only 512 samples per segment.

2.1.2 Spectral Analysis

MIT-AC uses a single-side band, critically sampled filterbank, known in the literature as Time-Domain Aliasing Cancellation (TDAC) transform [8], to analyze a windowed segment of audio. The TDAC (or any other transform) analyzes the audio segment by transforming the time domain signal into transform domain coefficients. This represents the signal as a linear sum of weighted basis functions, where the basis functions represent the different frequencies of the filter bank, and the transform

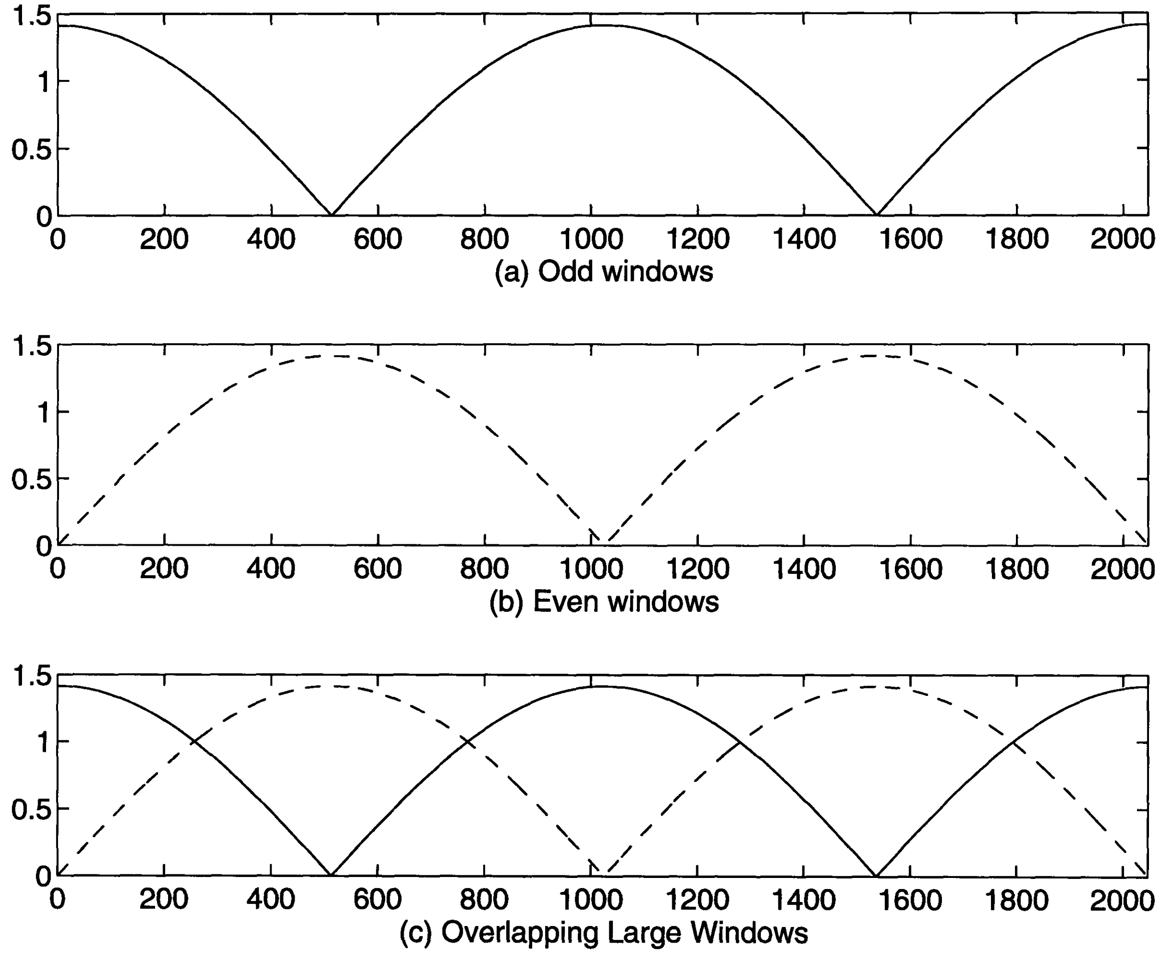


Figure 2-2: Overlapping Transform Windows

coefficients represent the weightings for these basis functions. It is important to note that the basis functions span the length of the window. In other words, if the window is 1024 samples long, then each basis function will also be 1024 samples long. It will be shown that this contributes to the formation of pre-echoes.

2.1.3 Psycho-Acoustic Modeling and Quantization

The spectral analysis transforms the signal into the frequency domain where the coder uses psycho-acoustic models to determine levels of quantization. Psycho-acoustic experiments [10, 9] have shown that a loud tone at any particular frequency tends to mask noise in neighboring frequencies within the same critical band. This stems from the physical workings of the inner ear. Critical bands are approximately logarithmic in length and correspond to lengths of the basilar membrane in the inner ear. A tone at a particular frequency will stimulate nerve endings at a specific location along the membrane. In this way, the membrane acts as a spectral analyzer. As the tone increases in volume, neighboring nerve endings, to a lesser extent, are also stimulated. This means that a loud tone at a given frequency tends to mask noise at near-by frequencies. The coder takes advantage of this fact by first dividing the spectrum into frequency bands which closely approximate the critical bands of the ear, and then calculating the effects of masking using the largest coefficient within each band.

There exists a curve that describes the "just audible" threshold of noise as a function of frequency [11]. To this basic level, the coder adds the calculated effects of masking to produce a noise floor like the one shown in figure 2-3. This curve represents the maximum amount of noise that will be inaudible to the average human ear as a function of frequency. By allocating an increasing number of bits to encode a coefficient, the quantization noise at that frequency is reduced. The coder allocates bits to the different transform coefficients in an attempt to minimize the audible quantization noise by closely matching or remaining under this noise floor. Under

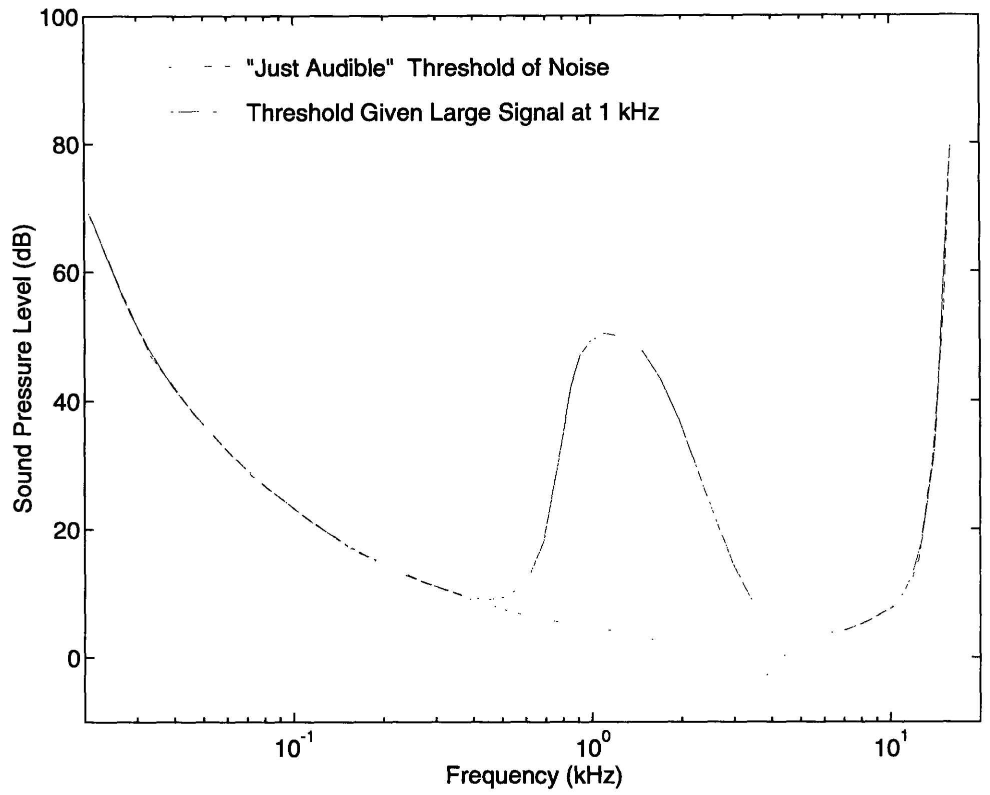


Figure 2-3: Noise Floors: "Just-Audible" Threshold; Masking Due to Large Signal at 1 kHz. Adopted from [5].

certain circumstances, the quantization noise which is introduced at this stage, even if it remains below the noise floor, can lead to an audible artifact called a pre-echo.

The transform domain representation of the signal covers a wide dynamic range. High frequency coefficients tend to be small compared with low frequency coefficients. Quantizing all the coefficients on the same scale would require a large number of bits for any reasonable degree of accuracy. Fortunately, the magnitude of the coefficients varies slowly with respect to frequency. Within a critical band, the coder determines the largest coefficient and scales all the other coefficients in the band with respect to this largest coefficient. This helps maximize the effectiveness of allocating another bit to the quantization of a coefficient. For a given number of bits, a large coefficient will be quantized with a large absolute error. However, a large coefficient represents a large amplitude contribution at that frequency. This means that more noise will be masked by the large signal, so it is usually acceptable that more quantization noise is present at this frequency. Unfortunately, in a pre-echo situation this no longer holds true and the fact that the quantization noise is proportional to the signal strength contributes to the audibility of the artifact.

2.2 Factors Which Contribute to Pre-Echo

In order to create a pre-echo, the conditions must be just right. The audio signal must be fairly non-stationary with respect to the window length and must be energetic enough to create a large absolute level of quantization noise.

2.2.1 Quantization

As discussed earlier, the quantization algorithm takes advantage of psycho-acoustic models to hide noise within the existing signal. These models divide the frequency spectrum into critical bands, and calculate a noise floor based on the largest coefficient within each band. After calculating the noise floor, the coder allocates bits to the transform coefficients. The more bits allocated to a transform coefficient, the more accurately the coefficient is coded. The error between the magnitude of the transform coefficient and its quantized representation contributes to the quantization noise. The total quantization noise is the linear sum of weighted basis functions, where the weightings are the quantization errors of the different coefficients. Therefore, since the basis functions span the length of the window, the quantization noise also spans the entire length of the window. This becomes a problem when the audio signal becomes highly non-stationary.

As mentioned in the previous section, the coefficients in a critical band are scaled to the largest coefficient within that band. This means that for a given level of quantization (e.g. 3 bits per coefficient), a large coefficient will have a larger quantization error than a small coefficient. So large energy signals will have more quantization noise than small energy signals. This should not be a problem since large energy signals can mask more noise than small energy signals. However, under certain circumstances, this too can contribute to a pre-echo.

2.2.2 Window Length

The choice of window length involves a trade-off between temporal and frequency resolution. A longer window offers better frequency resolution at the cost of reduced temporal resolution. The idea is to find a window length that produces the required frequency resolution without sacrificing the temporal resolution. Furthermore, it is convenient if the window is a power of two in length. This facilitates the transform calculation which uses a Fast-Fourier Transform (FFT) algorithm.

Frequency Resolution

The length of the MIT-AC window was originally chosen to be 1024 samples in order to allow the use of psycho-acoustic modeling. With a 1024-point window and a sampling frequency of 48 kHz, the spectral analysis has a frequency resolution of approximately 46.875 Hz and a range of 24 kHz. The high frequency resolution is necessary to effectively shape the quantization noise so as to be inaudible. A shorter window results in poorer frequency resolution which makes it difficult to accurately model the effects of tonal masking. The problem is that the critical bands at the lower frequencies are only 100 Hz wide. That means that there are only 2 coefficients per critical band for the first couple of bands. Using a shorter window could mean that one coefficient would represent more than one critical band. At that point, it is impossible to calculate inter-band masking and the coding advantage gained from using psycho-acoustic models is lost. On top of that, the coding of shorter windows is less efficient due to some fixed overhead. For these reasons, it can be more desirable to use longer windows for transform coding.

Temporal Resolution

While a long window may be necessary for frequency resolution, it does not always offer the necessary temporal resolution. In the absence of quantization, the coder offers perfect reconstruction of the signal, regardless of the window length. In that case, there is no reason not to use a longer window. However, with quantization, the use of a shorter window length can become useful. In order to take advantage of tonal masking of quantization noise, the coder assumes the audio signal varies slowly in character with respect to the window length. The longer the window, the less likely this assumption will remain true. This is important because the basis functions of the transform span the length of the window. This means that the quantization noise also spans the length of the window. If the audio signal is highly non-stationary across the window, the noise may become unmasked in low energy sections. The shorter the window, the more stationary the signal will be over that window. Furthermore, the quantization noise within that window will be shorter in duration.

2.2.3 Waveform

A pre-echo usually occurs when a quiescent passage is disturbed by a sharp transient. Examples of such an event include a chime or a castanet being struck or a guitar string being plucked. Figure 2-4 shows the waveform of a castanet being played. Such a waveform, in conjunction with the window length and the quantization, leads to a pre-echo. To see how this happens, let us see what happens when the coder processes this passage.

The encoder windows this segment of audio in a manner similar to that shown in figure 2-5. Part of the segment contains a very low energy signal while the rest of the segment contains a very energetic signal. A fairly sharp transition exists between the low and high energy sections. As mentioned before, the signal, when analyzed, is broken down into a linear sum of basis functions. The frequency spectrum is divided into critical bands and a noise floor is calculated. Up until this point no information

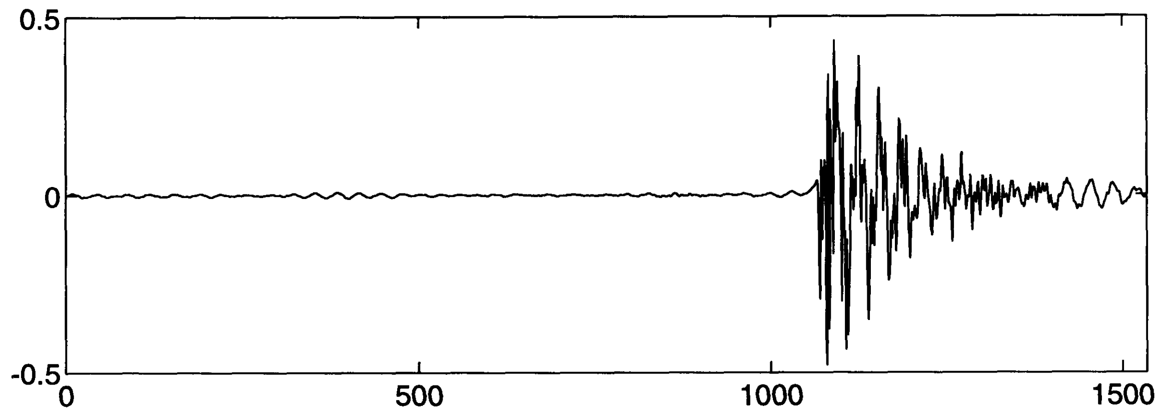


Figure 2-4: Original "Castanet" Waveform

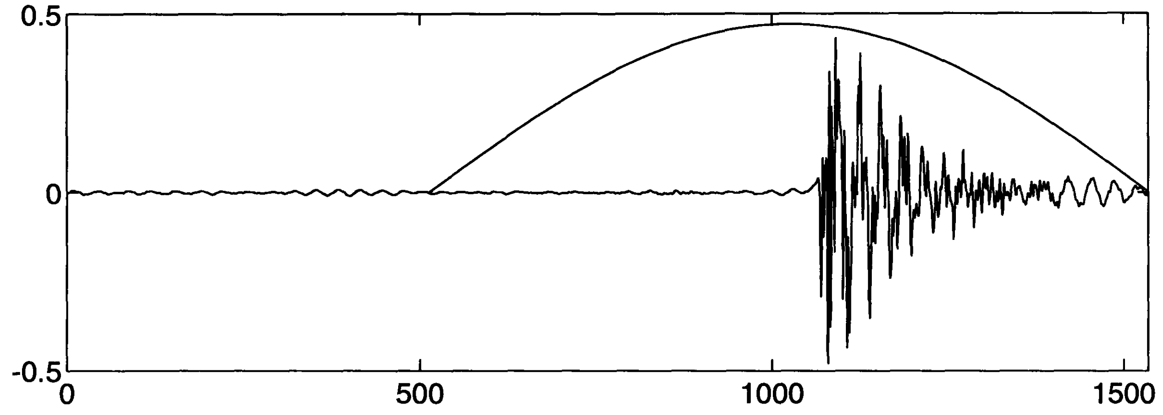


Figure 2-5: Possible Windowing of "Castanet" Waveform

has been lost. Given the transform coefficients, the decoder could exactly reconstruct the original signal.

Using the noise floor as a guide, the coder allocates bits to the coefficients. The coefficients are then scaled and quantized. Here is where the quantization noise is added to the signal. The decoder performs an inverse transform on the quantized coefficients in order to synthesize a signal similar to the original. The processed signal differs from the original by the quantization noise. As already mentioned, the quantization noise spans the length of the window and is scaled to the energy of the original signal strength. Since the segment shown in figure 2-4 has a large signal strength due to the transient, the noise is also going to be fairly large. Normally, the signal will be pseudo-stationary across the window, and the noise will be masked by the original signal. However, in this case, the noise will only be masked in the large signal portion of the window, while the noise in the quiescent portion will add an audible artifact to the leading edge of the transient. This pre-echo often manifests as a short hiss preceding the attack, or as a dulling of the attack itself.

Chapter 3

Pre-Echo Reduction

Psycho-acoustic experiments have demonstrated a temporal masking effect in which a large signal will mask noise both preceding and following the signal. The preceding masking time is limited to approximately 5 ms [3, 7]. Noise which occurs less than 5 ms before a loud sound will be masked to some extent. If the sound is loud with respect to the noise, then the noise will be inaudible. So if the quantization noise, which gives rise to a pre-echo, can be shortened to less than 5 ms, the pre-echo will become inaudible.

3.1 Background

The MIT-AC pre-echo reduction algorithm draws on work previously done trying to optimize the temporal/frequency resolution trade-off. While such work does not aim to address the pre-echo artifact in particular, it does attempt to deal with a general problem of which pre-echo is a more specific case. One approach is to use shorter windows which trade frequency resolution for temporal resolution. Another approach is to use multiple window sizes, choosing an appropriate length for each segment of audio.

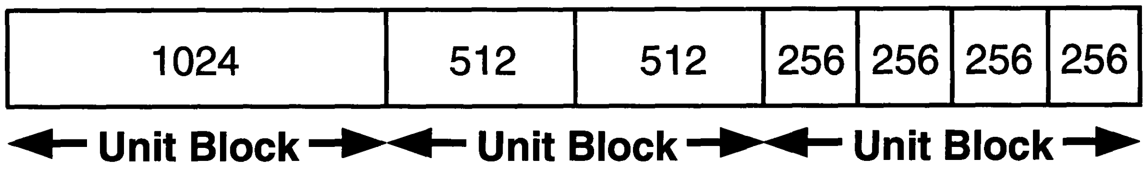


Figure 3-1: Block Lengths for ABS Algorithm

3.1.1 Short Windows

The shorter the window, the better the temporal resolution. Sharp features in the time domain are well preserved using such a window. This is useful in the case of a pre-echo since the pre-echo is a blurring of a sharp temporal feature. However, windows that are too short are fairly unattractive for audio coding. There are two main reasons for this. First of all, for a given bit rate, the coder can do a better job coding a long segment versus a short segment. Secondly, the frequency resolution of a short window is poor compared to the resolution of a long window. At discussed earlier, it becomes difficult to effectively apply psycho-acoustic masking models at low frequency resolutions.

3.1.2 Adaptive Block Size

In Adaptive Block Size (ABS) coding, each section of audio is processed using several different window lengths and the resulting waveforms are compared on the basis of some metric, such as the Signal to Noise Ratio (SNR). The waveform with the "best" metric is chosen.

For example, a coder might use three block sizes (window lengths), 1024, 512, and 256. For computational purposes, the blocks are all powers of two in length. The unit block length is 1024 meaning that 1024 samples are processed at a time. Therefore, a unit block consists of one 1024-point block, or two 512-point blocks, or four 256-point blocks as illustrated in figure 3-1.

ABS has the advantage of finding the appropriate time/frequency trade-off for each segment of audio. Segments with rapid time-varying features will have a larger snr when processed with shorter windows than with longer windows. Therefore, shorter windows will be chosen for the processing. Segments which are more stationary will benefit from the coding gain to be had using longer windows. The resulting increase in snr will cause the coder to choose long windows for processing.

Unfortunately, using ABS coding has a couple of drawbacks. For one thing, it involves an increased number of calculations. For each window length used, the coder must go through all the coding steps normally associated with one window length. An ABS coder using four different window lengths involves approximately four times the number of calculations that a fixed window length coder needs. This can become prohibitively expensive in terms of computation cycles, requiring faster processors or multiple processors in order to run in real-time.

3.2 MIT-AC Solution

Using a modified ABS algorithm, the MIT-AC takes advantage of the psycho-acoustic phenomenon of temporal masking in order to hide pre-echoes. Experiments predict that low level noise, which precedes a large signal by less then approximately 5 ms, will be masked by the signal. A window 5 ms in length or shorter guarantees that the quantization noise for any particular audio segment will be limited in duration to less than 5 ms. In order to maintain the high coding gain offered by the use of frequency masking models, MIT-AC uses long windows for most of the processing, switching to shorter windows only when a sharp transient is detected.

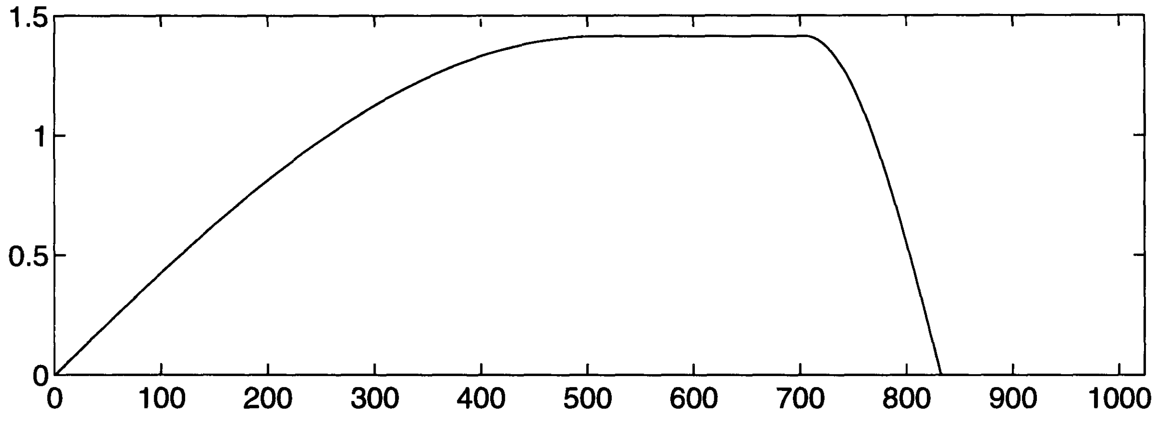
3.2.1 Temporal Masking

Psycho-acoustic experiments have demonstrated that the human ear exhibits temporal as well as frequency masking [3, 7, 10]. A large amplitude tone burst masks noise preceding as well as following the tone. However, the level of noise which remains masked in the backward masking case falls off rapidly when the noise precedes the tone by more than 5 ms. By contrast, noise which follows a tone burst is masked for a much longer duration. Typically, forward masking will hide a significant level of noise for up to 200 ms following a tone burst. So, even though a sharp attack produces quantization noise both preceding and following the attack, forward masking is guaranteed to render any trailing noise inaudible due to the long masking duration. This is why transform coders do not have to worry about a corresponding "post-echo" artifact.

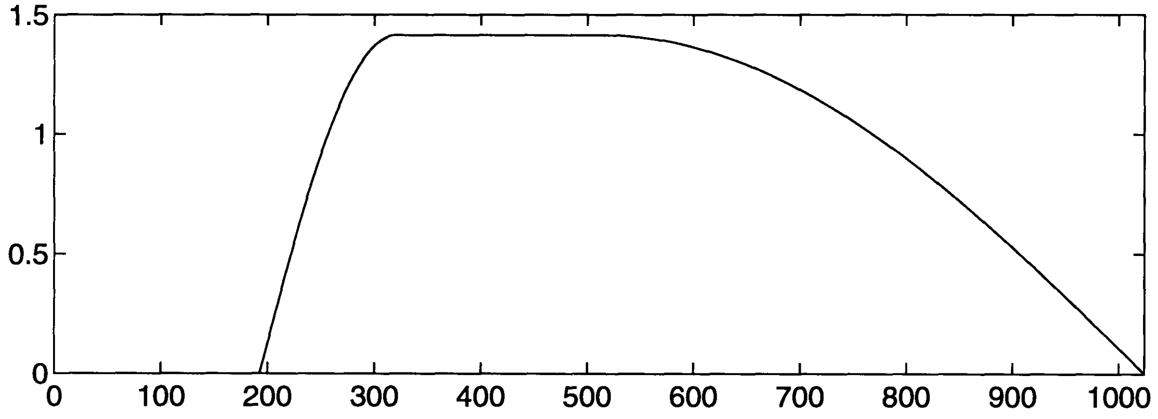
3.2.2 Adaptive Window Length Algorithm

A shorter (256-point) raised-cosine window was designed. For a 48 kHz sampling rate, the window is approximately 5 ms in length. Therefore, any pre-echo that occurs will be limited in duration to less than 5 ms, and should thus be inaudible. However, as mentioned in the discussion on window length, the short window can not be used in general due to its poor frequency resolution. If audio is coded using only short windows as opposed to long windows, the quality of the processed version is significantly degraded.

However, the short window is useful for limiting the duration of pre-echoes. Therefore, an adaptive window length scheme was proposed and implemented on the real-time system. The coder uses the longer 1024-point window by default, but switches to the shorter 256-point window when a potential pre-echo situation arises. This requires the use of transition windows as shown in figure 3-2. The transition windows were designed to easily fit into the existing algorithm. For this reason, they are 1024 samples long. Four short windows also have a periodicity of 1024 samples. This allows the coder to always processes the signal at the same rate, which eliminates the need for complex buffering at the input (and output). The windows fit together as shown in figure 3-3. Figure 3-4 shows a few of the schemes that could have been used to transition to and from shorter windows. However, none of these maintains a constant input-samples-per-frame processing rate, and would thus require



(a) Transition from Long to Short Windows



(b) Transition from Short to Long Windows

Figure 3-2: Transition Windows

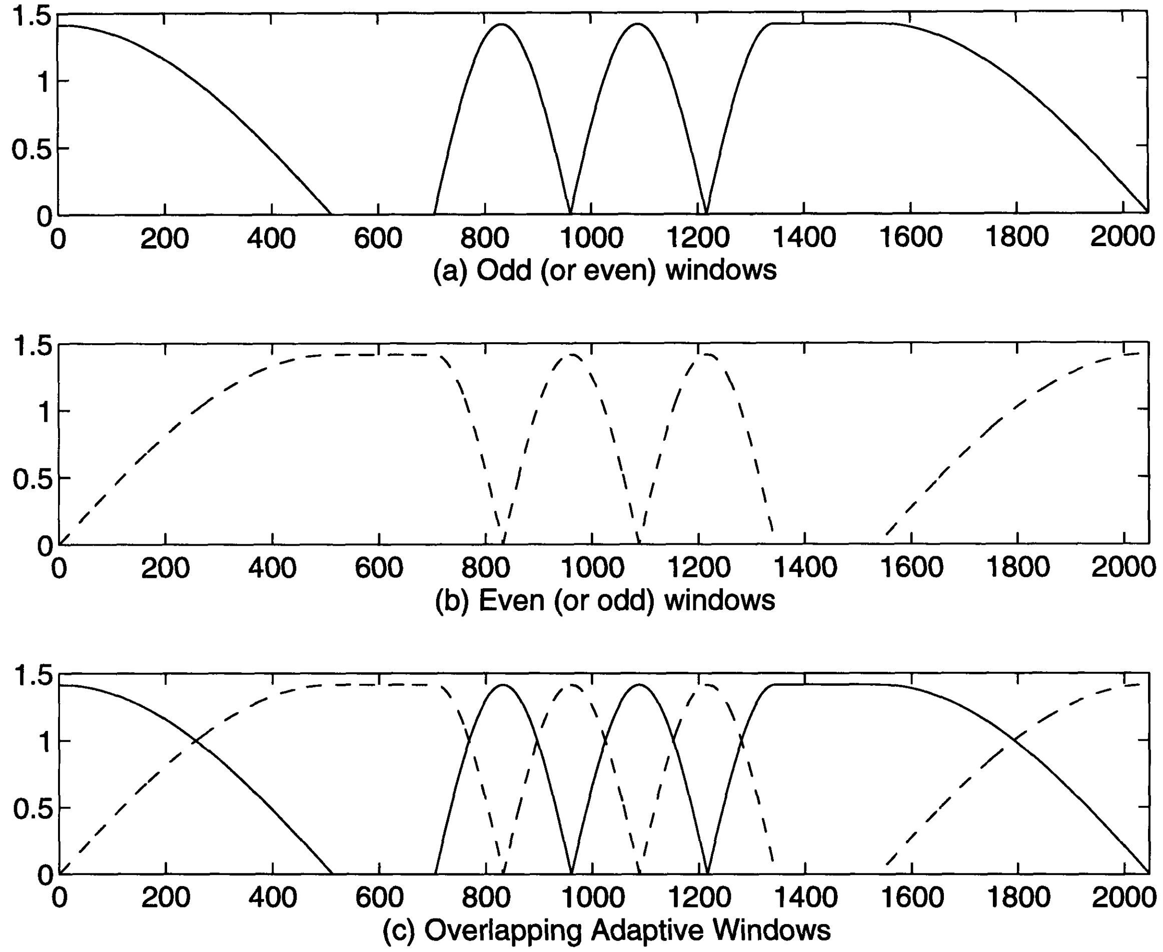
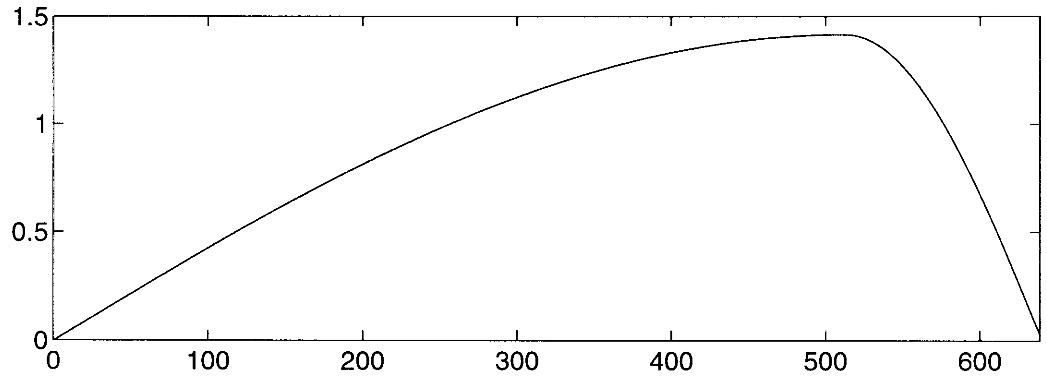


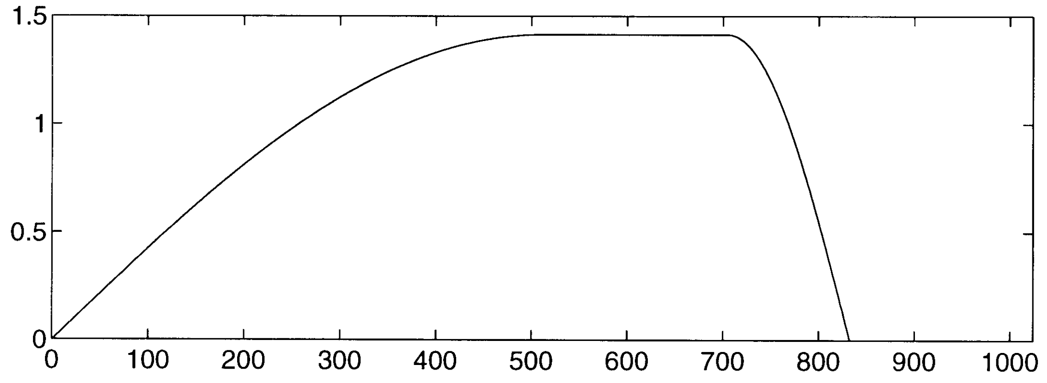
Figure 3-3: Overlapping Windows: Transition to Short Windows and Back

complex buffering and substantial modification of the code structure. Furthermore, the window lengths were chosen to be a power of two which allows the use of an FFT algorithm in calculating the transform.

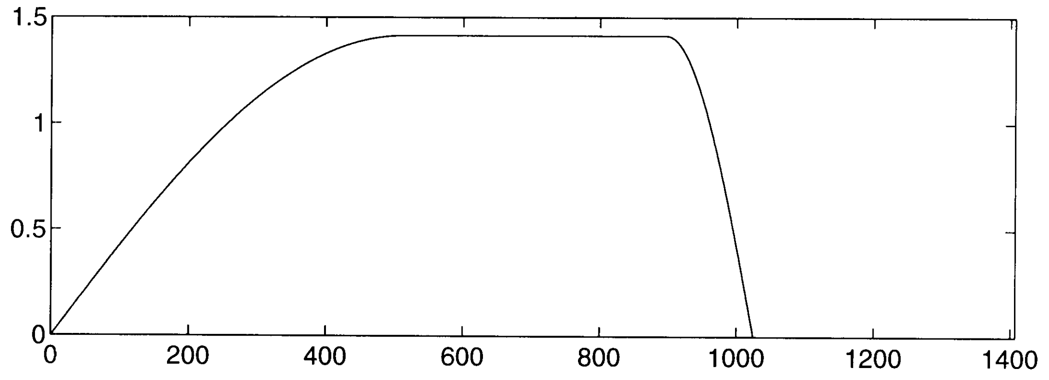
The same long window algorithm is used to code each of the four short windows. Each short segment is windowed and transformed. The effects of inter-band masking are ignored since they can not be accurately modeled at this frequency resolution. Instead, bit allocation and quantization are simply based on the "just audible threshold" noise floor. Since a segment that would normally be processed with a long window is instead processed with four short windows, approximately one-fourth the bits are given to each. After each short window is processed individually, the information



(a) 640-Point Transition Window



(b) 1024-Point Transition Window



(c) 1408-Point Transition Window

Figure 3-4: Transition Windows of Various Lengths

from all four is packed into one frame and transmitted to the decoder.

3.2.3 Detector

In order to use an adaptive window length scheme, the coder must use some criterion to decide which window length is appropriate. The Adaptive Block Size algorithm described at the beginning of the chapter, if applied to the MIT-AC, would require coding each segment of audio with both long and short windows and then comparing the two processed results. The processed segment with less audible noise would be chosen. This method turned out to be too computationally expensive to run on the real-time system. Therefore, a different approach was used. A detector was designed to examine the incoming audio signal for transients which could potentially lead to pre-echoes. Upon detecting a transient, the coder transitions to short windows, precluding any possible pre-echo from occurring.

Since the MIT-AC was already running a fairly computationally intensive algorithm, the detector needed to remain simple. The detector looks ahead for potential pre-echo situations, situations where a fairly silent passage is followed by a sharp transient. While not all such cases will necessarily lead to a pre-echo, many cases that do lead to a pre-echo, fall into this category. The detector implemented looks for a sudden, large increase in the variance. It does this by dividing the signal into blocks, calculating the variance within each block and comparing the variance with the variance from the preceding blocks. If the variance increases significantly over a short time period, then the detector instructs the coder to transition to short windows in order to reduce the pre-echo.

3.2.4 Problem With Detector

During testing, it became apparent that there were special cases which the detector did not catch. In particular, a sharp, high frequency transient added to a low frequency signal did not trigger the detector, yet still produced a pre-echo. An example of such a signal might be a kettle drum and a chime being played at the same time. The detector calculates the variance in an attempt to locate a transient, but the variance remains fairly constant since the large amplitude, low frequency kettle drum tends to dominate the waveform. However, the chime produces a high frequency transient which gives rise to a pre-echo during the coding process. The human ear is more sensitive to high frequency transients than to low frequency transients, so the pre-echo is still audible over the kettle drums.

3.2.5 Filtering

The detector was modified to trigger in such a case. The signal is high pass filtered before the variance is calculated. Any sharp transient will contain significant high frequency components which are preserved after filtering. However, low frequency signals are filtered out, making it easier to detect transients.

The high pass filter used is shown in figure 3-5. It was empirically derived and is limited to 5 taps due to limited real-time processing power. Simulations show that more accurate results can be obtained using filters with a larger number of taps. Even with 5 taps, the effect of filtering is readily apparent. The filtered waveform in figure 3-6 shows a significant increase in the variance over a short time span, whereas the unfiltered signal does not. Without filtering, the detector would miss this transient, and would code this segment using a long window, which could create a pre-echo.

3.3 Implementation Issues

There were a couple of implementation issues that had to be dealt with, in order to incorporate the adaptive window length scheme into the real-time MIT-AC system. Careful thought went into the design of the short window and the transition windows in order to maintain a steady processing bit-rate. Micro-code had to be optimized in

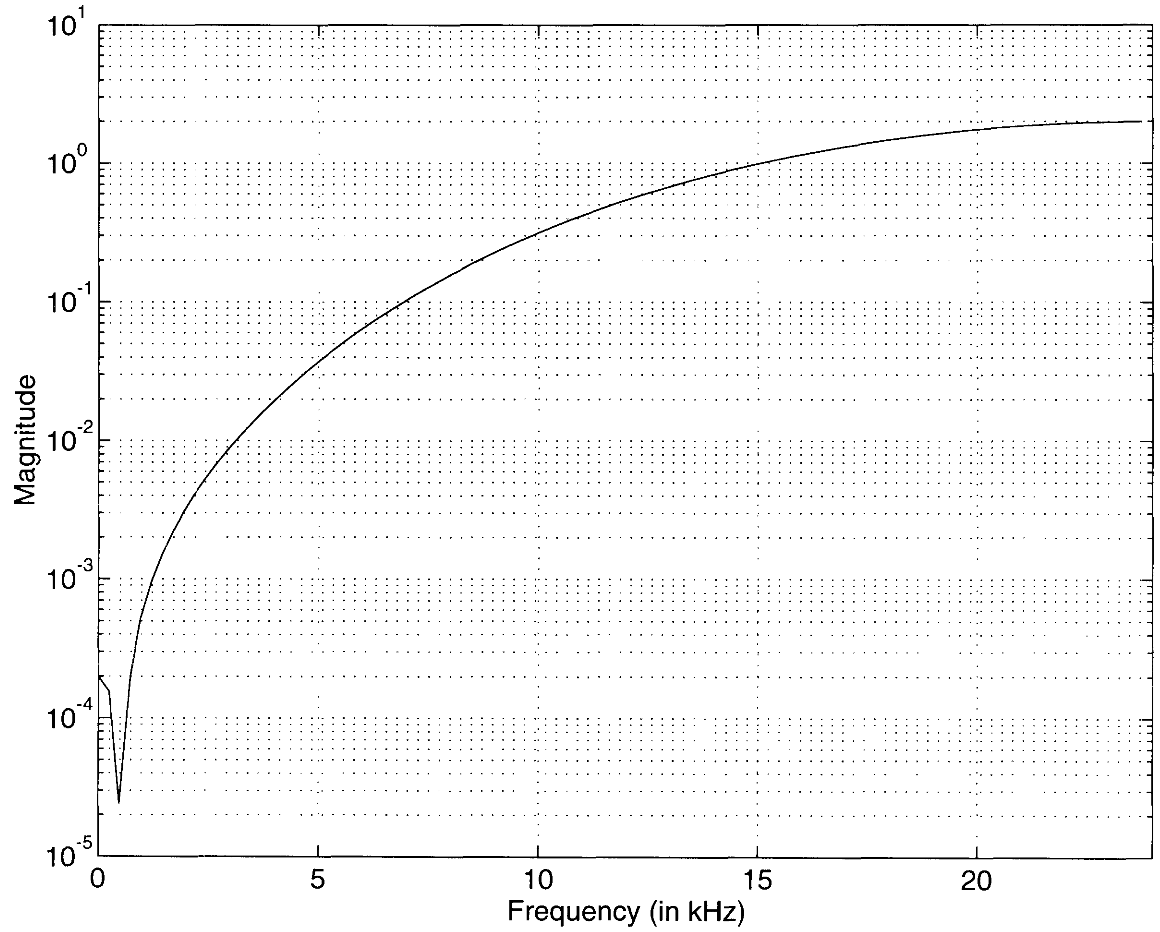
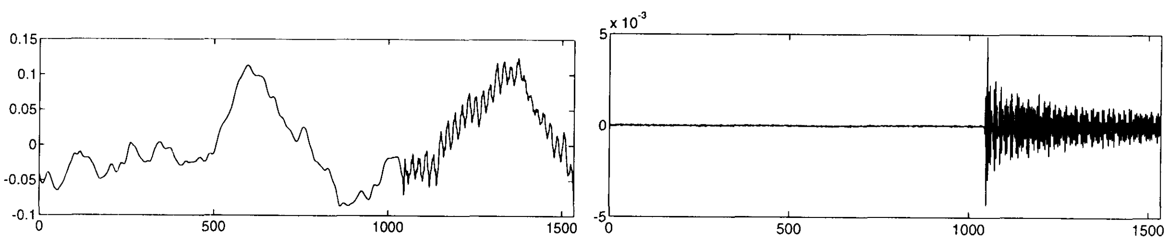


Figure 3-5: High Pass Filter



(a) Original Waveform

(b) Filtered Waveform

Figure 3-6: Comparison Between Filtered and Unfiltered Versions of "Kettle Drum & Chime" Waveform

order to free up more processing time for the adaptive window algorithm. Compromises had to be made in terms of detector complexity and filter length. These were issues that had little or nothing to do with the theory behind the coding, but that impacted the MIT-AC algorithm from an implementation standpoint.

3.3.1 Real-Time System

The real-time system uses Motorola DSP chips (MM96002) on PC boards made by Ariel. These boards plug into a host PC. The PC is used to down-load programs and supply power to the boards, as well as monitor the results of processing. All of the actual algorithm work takes place on the plug-in boards. As such, the algorithm is limited in complexity by the memory and processing capabilities of the boards and not the PC. Since the original algorithm (using only long windows) already used most of the processing power of the DSP chip, it was important to incorporate the adaptive window length algorithm with as little added complexity as possible.

3.3.2 Window Lengths

The original algorithm processed 1024 samples at a time while buffering 512 new samples per frame. This means that two blocks of samples, one old and one new, are processed together. The old block is discarded, the new block becomes the old block, and another 512 samples are buffered in to become the new block. This buffering reflects the fact that each window is 1024 samples long and overlaps adjacent windows by 50%. Figure 3-7 shows that window 1 uses block A and block B while window 2 uses block B and block C.

Ideally, changing to different window lengths will not affect this buffering routine. Extra computation time would be required to break apart the 512 blocks and reconstruct blocks of arbitrary size. Therefore, the new algorithm should still process 1024 samples per frame and should advance 512 new samples per frame. Using 256-point windows, this is a fairly straightforward extension. Figure 3-8 demonstrates this

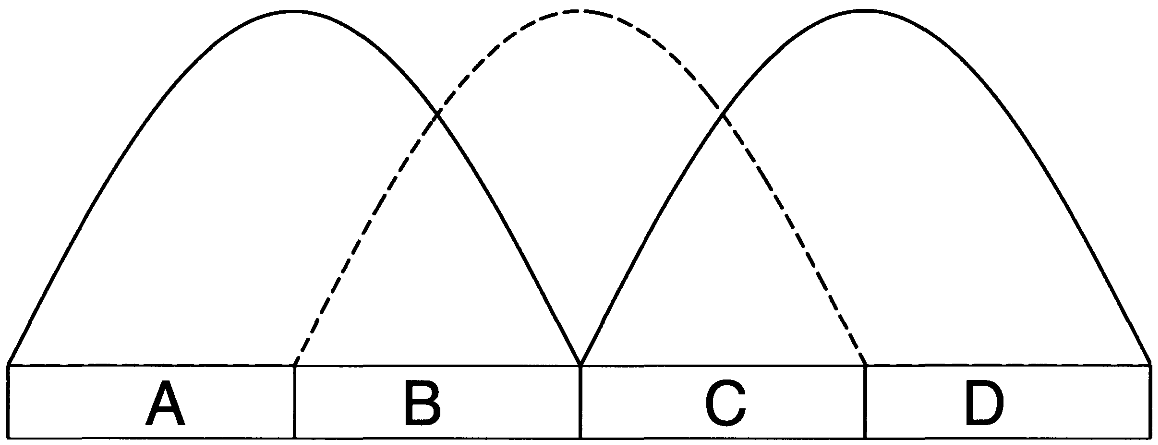


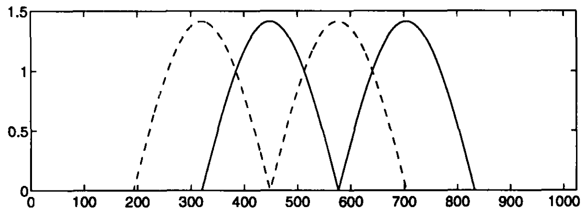
Figure 3-7: Processing with 512 Samples Per Block

scheme using only short windows. As can be seen, each frame covers 1024 samples and advances by 512 samples per frame. This requires processing 4 short windows per frame. Note that even though not all 1024 samples are processed per frame (the points at the beginning and end of the frame are zeroed out), these samples are covered by windows in adjacent frames.

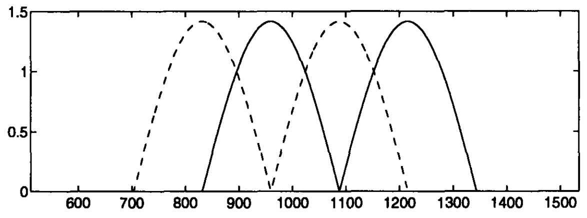
This scheme can be substituted for the long window scheme without changing the buffering, but in order to transition between long and short windows, special "transition windows" are needed. In order to conform to the constraints of the TDAC transform, the transition windows need to overlap adjacent windows properly. Figure 3-4 shows some possible transition windows. However, only one of these, window (b), fits in with the scheme chosen for the short windows. Note that the transition window is 1024 samples in length. This means that the transition window can be substituted for the long window without changing any of the remaining coding steps. Figures 3-9 and 3-10 demonstrate the transition to and from short windows.

3.3.3 Micro-code Optimization

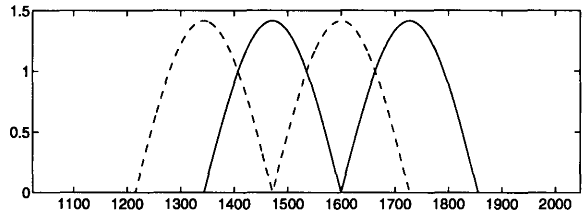
The adaptive window length algorithm required more processing time than the original single-window-length algorithm. When the new code was down-loaded and run



(a) Frame 1

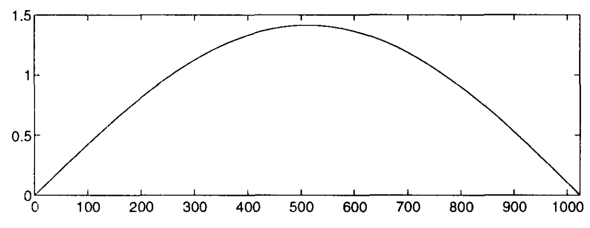


(b) Frame 2

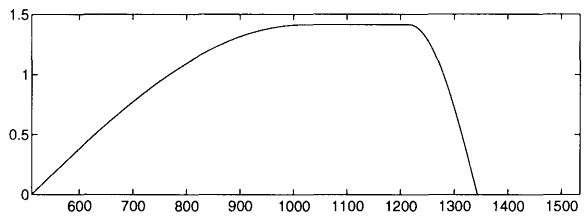


(c) Frame 3

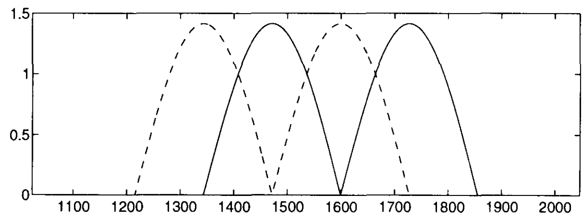
Figure 3-8: Frame by Frame Windowing: Using Only Short Windows



(a) Frame 1

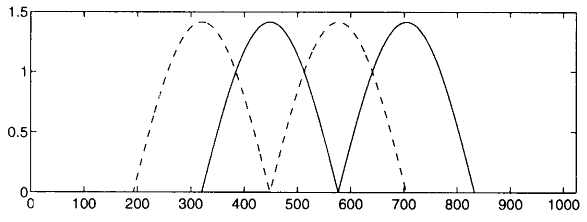


(b) Frame 2

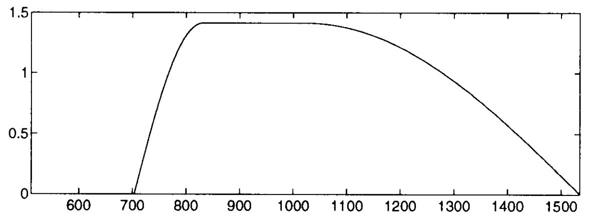


(c) Frame 3

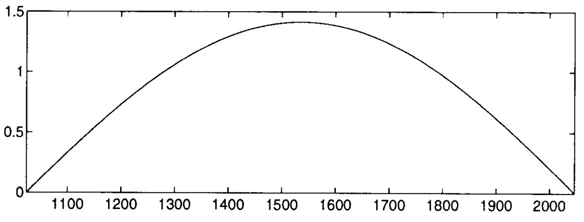
Figure 3-9: Frame by Frame Windowing: Transitioning to Short Windows



(a) Frame 1



(b) Frame 2



(c) Frame 3

Figure 3-10: Frame by Frame Windowing: Transitioning from Short Windows

on the real-time hardware, it became apparent that the encoder occasionally used up more processing time than allotted. Every 10.66 ms 512 new samples are buffered in, ready to be processed. Without some dynamic buffering scheme, each and every window needs to be processed in 10.66 ms or less. Otherwise, the encoder goes over real-time, gets out of synch with the input data flow, and transmits errors to the decoder.

So in order to run the new code on the real-time system, optimizations had to be made with respect to speed. Several sections of code were targeted for optimization. These were often fairly simple routines or loops that were called on repeatedly. Compact micro-code was written to replace the less efficient code compiled down from C. This was enough to allow the new algorithm to run without going over real-time.

3.3.4 Filter Length

By the time the detector was modified to include the filtering, all the obvious, and easy optimizations to the code had already been made. Saving any more cycles would have been a difficult and time consuming task. Due to lack of time, the filtering was implemented as well as possible given the constraints of the processing time available. This meant limiting the filter to 5 coefficients in length. While this reduced the effectiveness of the filter, the end result was still good enough, and the savings in programming time large enough to justify this trade-off.

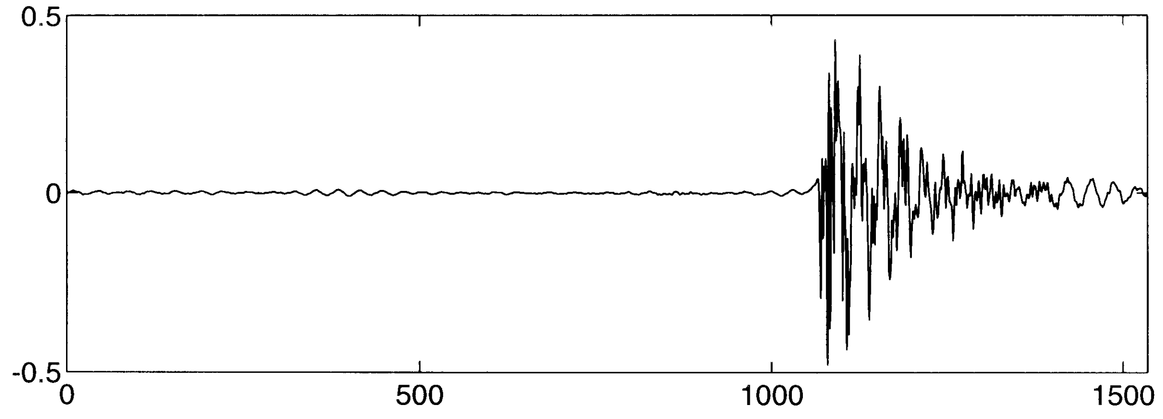
Chapter 4

Results

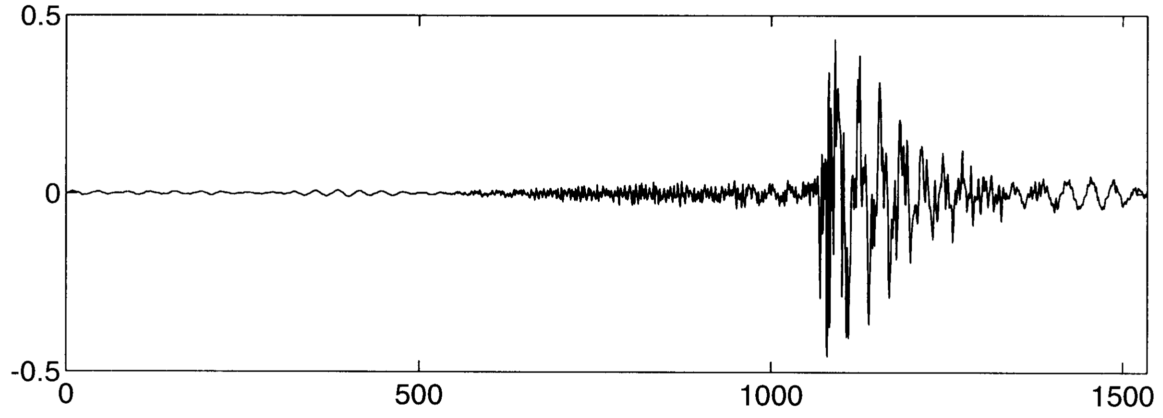
The algorithm was simulated in C and various waveforms were analyzed in Matlab. Furthermore, modifications were made to the real-time system to implement the transient detector and the adaptive window length selection. The results of both the simulation and the real-time system indicate that the algorithm successfully reduces the duration of pre-echoes to the point where temporal masking makes them inaudible. The detector triggers more often than necessary, but in the instances where it does over-trigger, there are no noticeable degradations.

4.1 Simulation Results

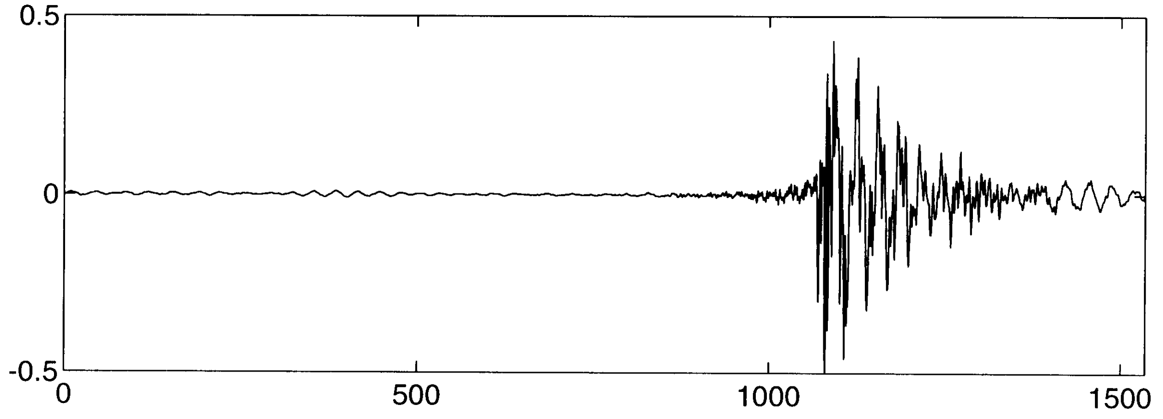
Figure 4-1 shows some of the results of simulations. The quantization noise is noticeably shorter in duration when the coder uses short windows. Figure 4-2 shows a segment of audio, and the processed versions (using both long and short windows). The waveforms on the right show the filtered version of each. The filtering reveals the high-frequency pre-echo and the reduction of the pre-echo using short windows. Once again, notice that the noise is shorter in duration. Using high-pass filtering, the encoder is able to detect and adapt to potential pre-echo situations even when the sound consists of a high frequency transient hidden under a larger amplitude signal.



(a) Original Waveform

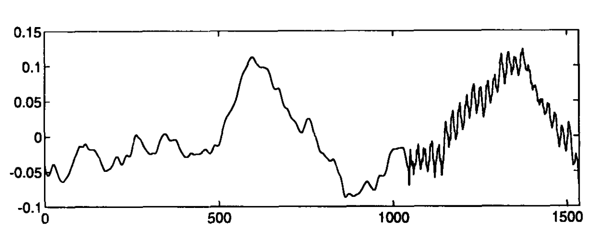


(b) Waveform Processed with Long Windows

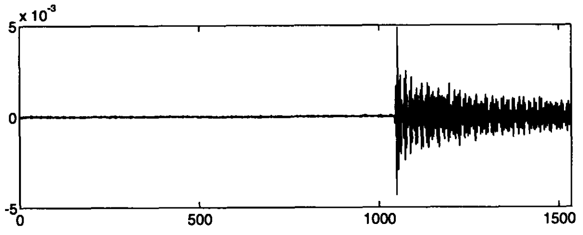


(c) Waveform Processed with Adaptive Windows

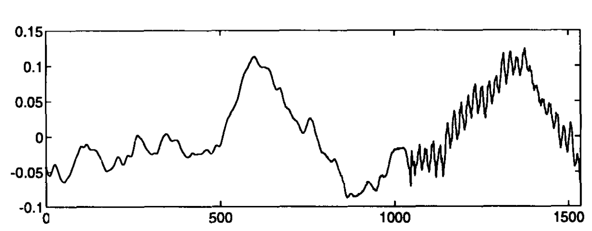
Figure 4-1: Simulation Results Processing "Castanet" Waveform



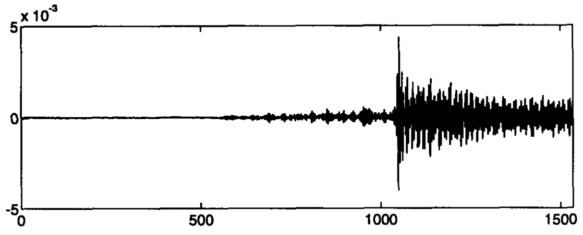
(a) Original Waveform



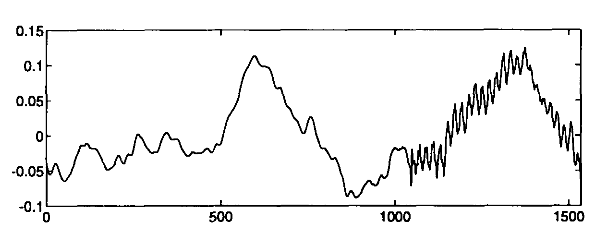
(b) Filtered Version of (a)



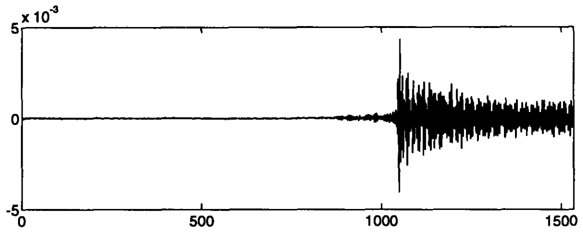
(c) Waveform Processed with Long Windows



(d) Filtered Version of (c)



(e) Waveform Processed with Adaptive Windows



(f) Filtered Version of (e)

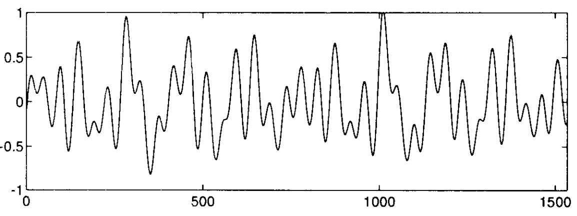
Figure 4-2: Simulation Results Processing "Kettle Drum & Chime" Waveform

4.2 Real-Time Results

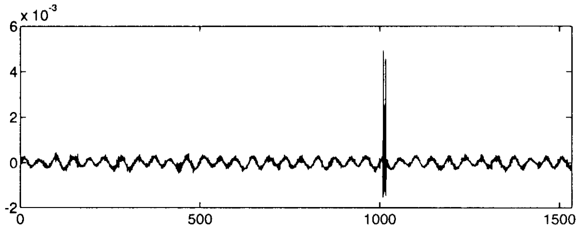
The real-time system was modified to implement the algorithm. Audio material, known to have previously created pre-echoes on the fixed-window algorithm, was run through the adaptive window system. Pre-echoes were no longer evident or were greatly reduced in passages where previously they had been fairly pronounced. This perceptual testing involved several hours of listening to critical audio material to evaluate the effectiveness of the algorithm as well as to make sure that the coder was not introducing any unforeseen artifacts. Various other audio materials were also selected and used to exercise the system. In all cases, back to back comparison of the original and the coded versions demonstrated a high degree of perceptual fidelity.

4.3 Detector Over-Triggering

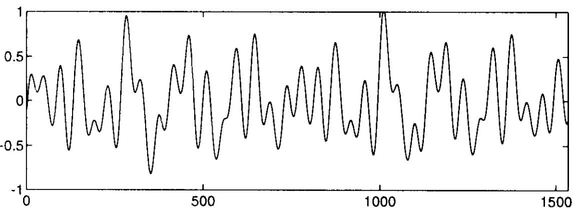
The detector sometimes triggers during passages where one would not expect a pre-echo to develop. However, even though the detector falsely triggers, the subsequent transition to short windows does not seem to adversely affect the perceptual quality of the signal in these few cases. Figures 4-3 through 4-5 show some example waveforms of cases where the detector triggers unnecessarily. Once again, the original signal is shown on top, followed by the version coded with long windows and then the version coded with adaptive windows. The filtered versions of each are shown on the right. Looking at these pictures, several things become obvious. The filtered version of the original waveform clearly shows a high frequency transient. So in this respect, the detector is actually doing the correct thing by flagging this as a potential pre-echo problem. In fact, looking at the filtered version of the waveform processed with long windows, the quantization noise is quite apparent. When processed with long windows, no artifact is audible because the quantization noise is either below the absolute threshold of hearing or because other components in the signal are loud enough to mask out the noise. So while the short windows do not really hide noise



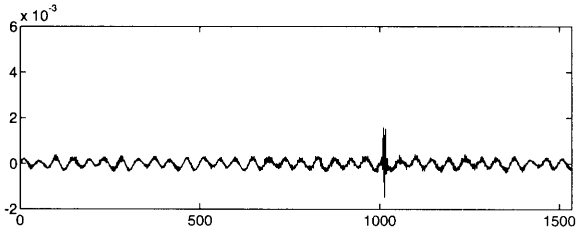
(a) Original Waveform



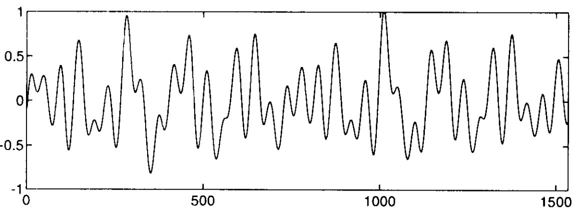
(b) Filtered Version of (a)



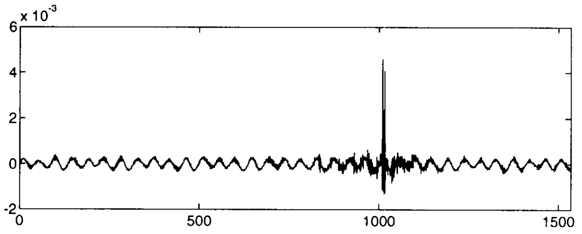
(c) Waveform Processed with Long Windows



(d) Filtered Version of (c)



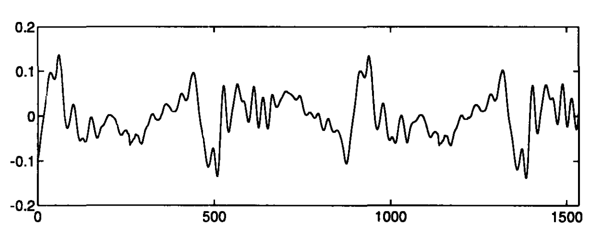
(e) Waveform Processed with Adaptive Windows



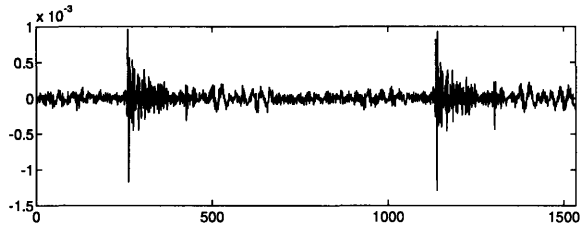
(f) Filtered Version of (e)

Figure 4-3: Simulation Results Processing "Chime" Waveform

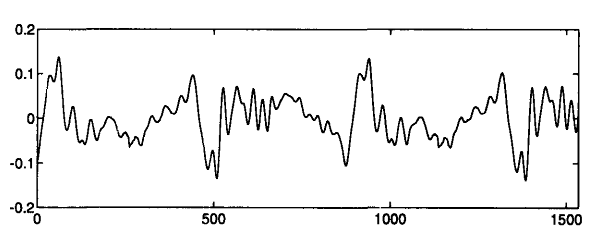
(the noise would have already been inaudible), they also do not add any new artifacts, and the perceptual audio quality is maintained.



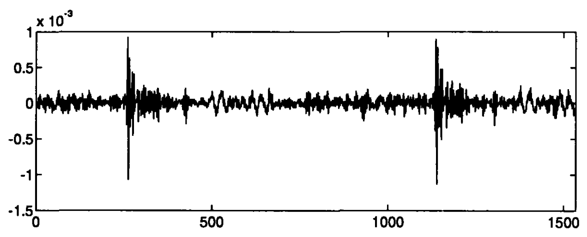
(a) Original Waveform



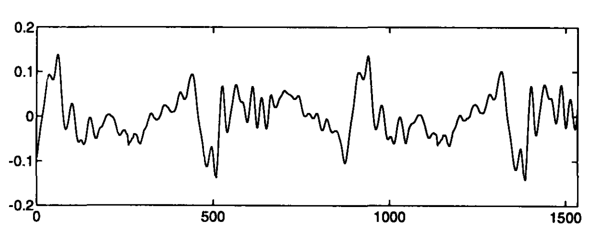
(b) Filtered Version of (a)



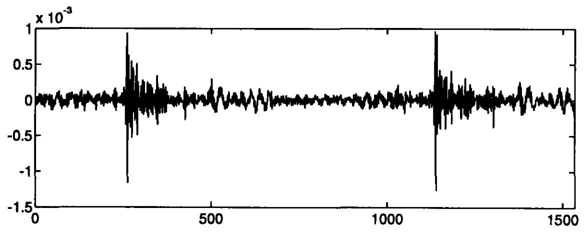
(c) Waveform Processed with Long Windows



(d) Filtered Version of (c)

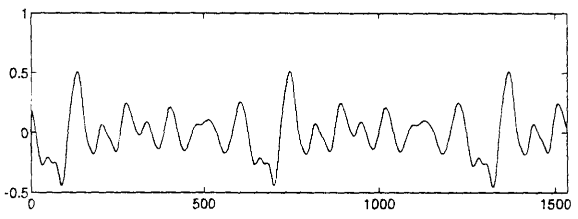


(e) Waveform Processed with Adaptive Windows

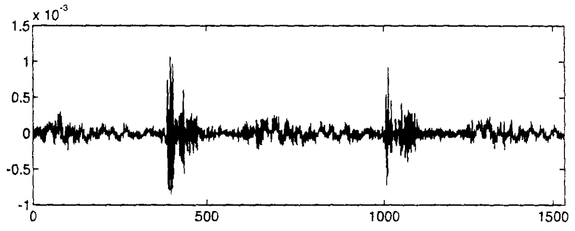


(f) Filtered Version of (e)

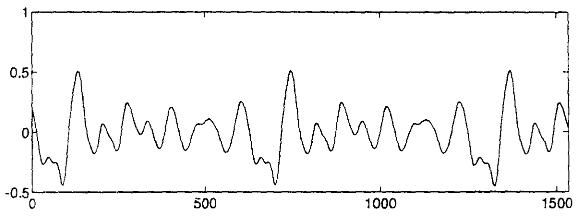
Figure 4-4: Simulation Results Processing "Oboe" Waveform



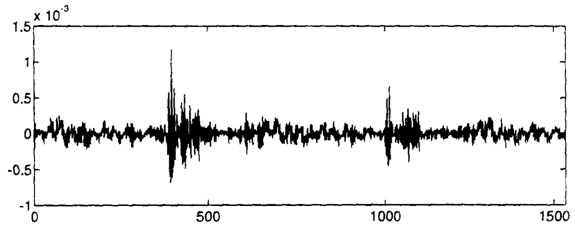
(a) Original Waveform



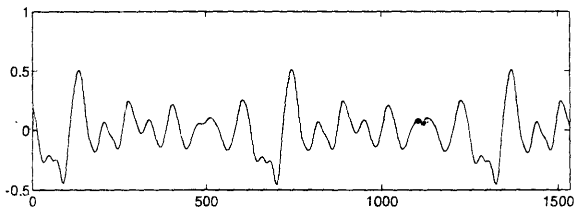
(b) Filtered Version of (a)



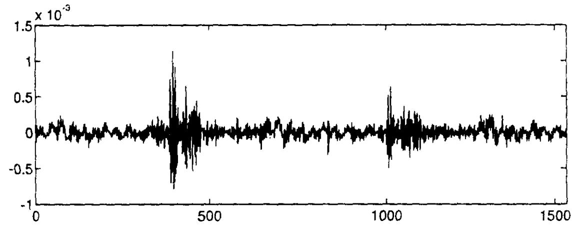
(c) Waveform Processed with Long Windows



(d) Filtered Version of (c)



(e) Waveform Processed with Adaptive Windows



(f) Filtered Version of (e)

Figure 4-5: Simulation Results Processing "Oboe2" Waveform

Chapter 5

Conclusions

5.1 Summary

Transient waveforms, window length, and psycho-acoustic bit allocation and quantization interact to produce pre-echoes in transform coders. Given this type of transform coding and the desire to achieve high compression ratios, there are a limited number of approaches that can be used to deal with pre-echoes. It is clearly unrealistic to constrain the input audio to pseudo-stationary waveforms, and the coder needs to take advantage of psycho-acoustic modeling in order to maintain low bit rates. Therefore, the transform window length is the easiest factor to modify.

While short transform windows are effective in limiting the audibility of pre-echoes, their poor frequency selectivity makes them unsuitable for applications that require high perceptual quality at low bit rates. Therefore, an adaptive window algorithm was developed to combine the advantages of long and short windows. Long windows have the frequency selectivity needed to effectively employ psycho-acoustic modeling. In the presence of a highly transient waveform, the coder is able to transition to short windows which limit the pre-echo to the realm of inaudibility.

In order for the adaptive window algorithm to be effective, the coder must be able to detect potential pre-echo situations. To this end, a detector was designed to search for transients. The detector calculates the variance of a small section of audio and compares it to the variance of preceding sections. If the variance suddenly increases by a large amount, the detector concludes it has found a transient. High pass filtering allows the detector to look for sharp transients even in the presence of large amplitude, low frequency noise.

The transient detector and adaptive window length algorithm were implemented on the MIT-AC real-time system. Critical listening tests confirmed the elimination of audible pre-echo artifacts. The coder switched to short windows during transients and pre-echoes which had been previously noted at these locations were no longer audible. There are situations where no pre-echoes were previously noted in a passage, but the detector now instructs the coder to switch to short windows. However, in these cases, there is still no audible degradation of the signal. The MIT-AC thus demonstrates that a computationally inexpensive adaptive window algorithm can effectively deal with pre-echo artifacts.

5.2 Future Work

5.2.1 Detector Improvements

Simulations indicate that better detectors can be designed using a larger number of coefficients for the high pass filter. More work can be done in the area of filter design as well as other methods of transient detection. In the real-time system, lack of more processing time precludes implementing a larger filter. Therefore, work can be done on further code optimizations with the goal of freeing up processing power to allow the use of a more complex detector.

5.2.2 Signal Representation

As described earlier, the representation of the signal in the transform domain is very important to the coding process. The length of the window helps determine the frequency resolution of the representation. More importantly, the type of transform used determines the actual representation of the signal. For instance, coders using the Hierarchical Lapped Transform (HLT) do not suffer from the same pre-echo problem that most transform coders encounter [6]. Switching to such a representation requires more processing power, but excludes the occurrence of a pre-echo. Work could be done trying to implement different transform representations on a real-time system.

5.2.3 Buffer Feedback

A more complex buffering system could also help reduce pre-echoes. Buffer feedback allows the coder to assign a variable number of bits for use in coding a frame. By using fewer bits to code an "easy frame", more bits can be used to code a "difficult frame". Using buffer feedback, the coder could allocate a larger number of bits to a frame which covers a sharp transient. The increased number of bits would help code the signal more accurately and would thus help reduce any pre-echo.

Bibliography

[1] Karlheinz Brandenburg et al. Fast signal processor encodes 48 khz/16 bit audio into 3 bit in real time. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2528–2531, 1988.

[2] Ernst Eberlein, Heinz Gerhäuser, and Stefan Krägeloh. Audio codec for 64 kbit/sec (ISDN channel) — requirements and results. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1105–1108, 1990.

[3] L. L. Elliott. Backward masking: Monotic & dichotic conditions. *Journal of the Acoustical Society of America*, 34:1108–1115, 1962.

[4] James D. Johnston. Perceptual transform coding of wideband stereo signals. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1993–1996, 1989.

[5] Y. Mahieux and J. P. Petit. Transform coding of audio signals at 64 kbits/s. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 0518–0522, 1990.

[6] Henrique S. Malvar. Extended lapped transforms: Properties, applications, and fast algorithms. *IEEE Transactions on Signal Processing*, 40(11):2703–2714, November 1992.

[7] J. M. Pickett. Backward masking. *Journal of the Acoustical Society of America*, 31:1613–1615, 1959.

[8] John P. Princen and Alan Bernard Bradley. Analysis/synthesis filter bank design based on time domain aliasing cancellation. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-34(5):1153–1161, October 1986.

[9] M. R. Schroeder, B. S. Atal, and J. L. Hall. Optimizing digital speech coders by exploiting masking properties of the human ear. *Journal of the Acoustical Society of America*, 66(6):1647–1652, December 1979.

[10] J. A. Tobias. *Foundations of Modern Auditory Theory*, volume 1. Academic Press, Inc., 1970.

[11] E. Zwicker and H. Fastl. *Psychoacoustics: Facts and Models*. Springer-Verlag, 1990.