Bit Rate Scalability in Audio Coding

Simon E. M. Plain

Department of Electrical and Computer Engineering
McGill University
Montréal, Canada

April 2000

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Engineering.

Abstract

In recent years, audio coding has proved to be of great importance for applications such as mobile communication and multimedia systems. Our research has focused on adapting a low bit rate/fixed bit rate audio coder to handle signals at a wide range of bit rates and sampling rates. Bit rates as low as 4 kbps and sampling rates between 4 kHz and 16 kHz were incorporated. Further, the coder can change bit rates during transmission in order to adapt to the available channel bandwidth. The coder was implemented in the C programming language and its performance evaluated. It was found that the quality of the reproduced audio for this coder requires some improvement, but the scalability performs smoothly, with transitions causing no artifacts, and reasonable loss of quality at the lowest bit rates.

Sommaire

Récemment, le codage sonore est devenu très important pour les applications telles que la communication mobile et le multimédia. Nous avons adapté un codeur sonore avec un taux de bit bas et fixe pour le rendre capable d'accepter des signaux avec une grande variété de taux d'échantillonage et capable d'envoyer des signaux à plusieurs taux de bit. Des taux d'échantillonnage entre 4 kHz et 16 kHz et des taux de bit aussi bas que 4 kbps ont été incorporés. De plus, le codeur est capable de changer les taux de bit pendant la transmission afin de s'adapter à la largeur de bande passante disponible. Le codeur a été mis en application en utilisant la programmation C et ses performances ont été évaluées. On a constaté que la qualité de l'acoustique reproduite par ce codeur exige une certaine amélioration, mais la scalabilité s'exécute sans à-coup, avec des transitions ne causant aucune erreur perceptible, et une perte acceptable de qualité aux plus bas taux.

Acknowledgments

I would like to thank my supervisor, Professor Peter Kabal, for his direction and advice throughout the course of this thesis. I am also very grateful to Hossein Najafzadeh-Azghandi for working with me and guiding me through his coder. Finally I would like to thank Khaled El-Maleh and the other graduate students at the Telecommunications and Signal Processing laboratory for interesting discussions and assistance with various issues.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Audio Coding

In the modern world we find ourselves relying more and more on less and less bandwidth for our communications. Wireless and Internet communications in particular have forced us to innovate towards reduction of bandwidth usage while maintaining a reasonable level of speech or audio reproduction. Digital Signal Processing (DSP) techniques can be used to decrease the redundancy and irrelevancy contained in an audio signal.

A great deal of DSP research has focused on digitally encoded speech for quality improvements and bandwidth reduction. More and more however, research is turning to the more general problem of audio compression, or coding.

Audio coding is an important step towards delivering a high quality communications experience. Traditional speech coders generally perform poorly when applied to any sort of music or music combined with speech. This can cause audible artifacts for the listener when, for example, he or she is on hold with music playing over a phone, or during a multimedia Internet broadcast.

Generally speaking, high compression speech coders these days use a production based model to minimize the information transmitted. For example, most voiced speech (e.g., vowels) contains important information such as the *fundamental frequency* of the speaker and its corresponding *formants* [1]. Exploiting knowledge of the characteristics of speech has allowed researchers to drastically reduce the information required to adequately represent it in digital form. Audio information, however, is too complex to encode based on its source, since there is such a wide variation in the sources for audio signals. Therefore, with audio coders, we instead try to model the human hearing system to eliminate inaudible portions of the signal. It is also possible for some speech coders to take advantage of the human hearing system, but either due to relatively little gain in quality for the amount of processing which must be done, or because the structure of the speech coder does not allow direct application of a hearing model, few popular speech coders do so.

The most famous audio codecs today are arguably those standardized by the Moving Picture Experts Group (MPEG). They have so far produced two standards which include audio codecs, MPEG-1 and MPEG-2. MPEG-4, as of this writing, is in development but nearing completion. MPEG-1 has drawn significant public attention for its high quality wide bandwidth compression. MPEG-2, especially with its Advanced Audio Coding (AAC) profile, has improved on this performance [2, p. 789]. MPEG-4 will include a wide array of low bit rate audio codecs as well as a revised version of AAC.

1.2 Bit Rate and Sampling Rate Scalability

When designing a digital coder, we must at some point decide how much bandwidth we are willing or able to occupy with our encoded signal. This translates directly to the number of bits per unit of time we are allowed to send. In fixed rate coders, the number of bits to be used is rigid. The same amount of bandwidth is used for every time frame. This contrasts with coders which can change the number of bits per unit of time they use.

There are several ways in which coders can implement possible bit rate changes. First, coders can use a different bit rate each time it is used, without changing mid-transmission. We will call these coders *bit rate switchable.* Second, a coder could be able to dynamically alter its bit rate so that the minimum number of bits is used to deliver a specified level of quality. This we will call *adaptive bit rate mode.* Third, a coder which can change its top bit rate during transmission in order to adapt to the available channel bandwidth we will call *bit rate adjustable.* Finally, when a coder can do all three of these operations, we will call it *bit rate scalable.*

An audio coder which is bit rate switchable allows us to move one step closer to a *universal coder.* This would be of great advantage in terms of using a single codec for connections of differing speeds. Users with greater bandwidth could enjoy the higher fidelity of high bit rates, while the low bandwidth users would be able to communicate with decent quality.

Adaptive bit rate mode coders can use the available bandwidth more efficiently than fixed rate coders. They are sometimes avoided due to hardware limitations. For example, in a channel with a fixed bandwidth, if the portion of a signal requiring significantly more bits than average was being coded, the channel could be overloaded. This requires that buffers be constructed for the additional bits [3, p. 632].

A bit rate adjustable coder could be very useful in many applications. For example, during an Internet broadcast there are many reasons why the bandwidth available to a user would decrease, and it would likely be preferable to simply lower the quality of the audio for that user than to stop the broadcast entirely.

A very general audio coder must also be able to handle multiple sampling rates. The decision of what sampling rate to use for a transmission affects the quality of the reproduced signal, as well as how many bits are required to represent that higher bandwidth signal. Most coders work either at a single fixed sampling rate, or have specified sampling rates at which they can operate. The coder discussed herein is able to operate at any sampling rate between 4 kHz and 16 kHz, and could be altered to function well for almost any reasonable sampling rate. This allows us to use a single codec for many sampling rates and therefore a wider range of applications.

1.3 Scope of Our Research

The overriding goal of our research is to create an audio coder which can be used in as many situations as possible. At high bit rates (e.g., 64 kbps per channel), codecs which can represent music and speech without artifacts have already been developed, therefore we concentrate our efforts on a coder which can adequately represent speech and audio at low bit rates (less than 16 kbps). In the course of our research we look at existing coders as well as their approach to bit rate scalability.

A previously invented coder designed to operate at a fixed 8 kbps bit rate and 8 kHz sampling rate, which codes audio and speech data well [4], is adapted to operate at many bit rates and sampling rates. We also enable it to switch between bit rates midstream. The resulting coder, along with the original coder, is simulated in software using the C programming language, and tested.

This thesis will first present fundamental information on audio coding practices. Second, existing audio codecs will be discussed along with their bit rate scalability. Next, we present the details of the audio coder we developed. Finally, we discuss the performance of our coder as well as where it could likely be improved.

Chapter 2

Audio Coding Basics

2.1 Time to Frequency Transformations

The most basic form of digital representation of a signal is called Pulse Code Modulation (PCM). This representation is formed by taking a *sample* of an analog (continuous in time) signal at regular time intervals. The rate at which these samples are taken is called the sampling rate. The input to our coder is assumed to be in PCM form, so we refer to our input data as samples. We should note that on the receiving end of a coder, the signal will be output to PCM in order to be converted back to a continuous signal for the listener to hear.

Almost all high compression audio coders are frequency domain coders so that they can take advantage of frequency characteristics of the ear. Frequency domain coders analyze the frequency properties of the signal in order to determine what parts are redundant. Frequency domain audio codecs operate on frames of data, the length of which depends on the intent of the coder and the type of signal. Long frames tend to allow better frequency resolution, and short frames give better time resolution while *smearing* frequency information. So for example if we wanted to analyze a sharp transition in a sound, we would want to use a short window. This would allow us to analyze this data without polluting the analysis with information unrelated to the transient. It would also unfortunately give poor frequency resolution. For a long tone however, a longer frame (e.g., 30 ms) would naturally be better so the frequencies present could be determined more accurately, and also because we would not need a great deal of time resolution.

In order to determine the frequency characteristics, we must apply a time to frequency mapping function. For digital signals, the most common mapping is the Discrete Fourier Transform (DFT),

$$X_{\mathrm{DFT}}(k) = \sum_{n=0}^{N-1} x(n) e^{\frac{-j2\pi kn}{N}}, \tag{2.1}$$

where n is the sample index and N is the number of samples in the transform. Early audio coders often used the DFT for its well understood properties as well as for the many fast implementations which have been developed for it (Fast Fourier Transform, or FFT). DFT coefficients are rarely directly encoded in audio coders today though, mainly because use of the DFT produces what are called *block edge effects*. Block edge effects are caused because the DFT operates on individual frames of signal data. When that data is transformed, quantized, then transformed back to the time domain, the beginning and ending samples of that block of data are often not coordinated with the preceding and subsequent blocks. This causes an audible periodic noise.

In order to deal with block effects, another transform, the Modified Discrete Cosine Transform (MDCT) has had widespread adoption in audio coders,

$$X(k) = 2 \sum_{n=0}^{N-1} x(n) \cos\left(\frac{2\pi}{N}\left(n + \frac{\frac{N}{2}+1}{2}\right)\left(k + \frac{1}{2}\right)\right), \quad k = 0, \ldots, \frac{N}{2} - 1. \tag{2.2}$$

The MDCT is a transform whose frames overlap by 50%. This deals with the problem of block edge effects [5, p. 23], and the MDCT retains the feature of critical sampling (producing one coefficient per input sample), since it only has half as many frequency coefficients as it has samples in its frame. The primary disadvantage of the MDCT is the less obvious analysis of its coefficients. As a result of this, audio coders often still use the DFT to more accurately apply their hearing model.

The coefficients $X(k)$ and $X_{\mathrm{DFT}}(k)$ are often referred to as *spectral* coefficients.

The diagram in Figure 2.1 represents the basic form of a transform coder. The darker lines indicate frequency information being passed in parallel between sections.

2.2 Windowing

When analyzing frames of data we must decide not only how long a frame of data we want, but also how that frame needs to be modified for clearer analysis. When we modify the

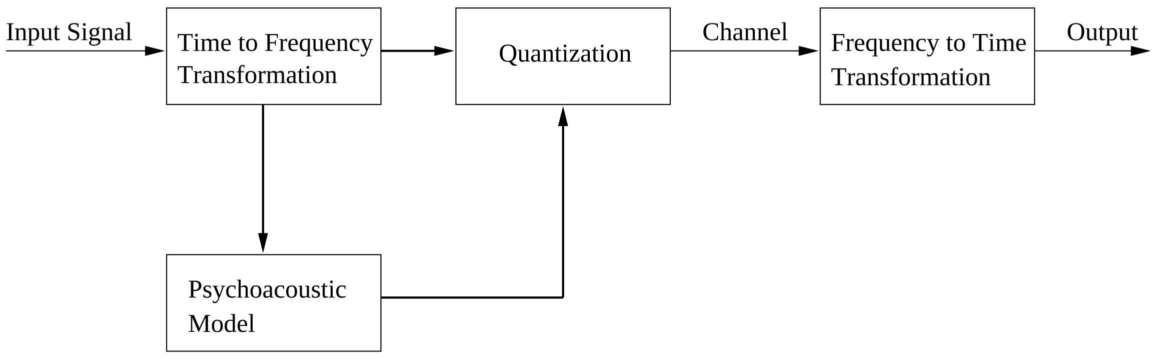
Input Signal
Time to Frequency Transformation
Quantization
Channel
Frequency to Time Transformation
Output
Psychoacoustic Model

Fig. 2.1 Basic structure of a transform audio coder

shape of an input frame it is called applying a window to that frame. If we were to take a frame of PCM sound data and perform frequency analysis directly on that frame without modifying it, we say we are applying a rectangular window. The main problem with using a rectangular window is that there is usually some loss of accuracy in the analysis. This is because at the edge of each frame the periodic parts of the signal are cut off which can lead to distortion of the frequency analysis. To correct this, most windows emphasize the mid-frame samples while degrading the edge samples.

We also want to take advantage of the fact that using the MDCT we can perform perfect reconstruction. This happens when we are able to obtain the identical signal after applying an MDCT and then an inverse MDCT (IMDCT) to it. For perfect reconstruction to be possible however, our window must have the property that the sum of the product of overlapped windows is a constant [6].

One popular window is the half-sine window:

$$w(n) = \sin\left(\frac{\pi n}{N-1}\right), \quad n = 0, \ldots, N-1, \tag{2.3}$$

where N is the length of the window. This window satisfies the above criterion and performs well for signals with dense harmonics. Other windows with greater rejection of widely spaced frequency components have also been used in audio coding.

2.3 Psychoacoustics

In order to find the redundancies and irrelevances in sound, a great deal of research has been put into building a mathematical model of human hearing. Considerable success has been achieved and we are able to effectively eliminate parts of a digital signal with no or very little effect on the perceived sound [7]. This achievement is in spite of great difficulties. Since we cannot observe hearing anatomy while it is active, we must rely on psychological hearing tests, which are subjective.

The human inner ear receives audio frequencies as physical vibrations at different locations on the basilar membrane [1]. When a sound reaches its corresponding location, it causes hair cells on the basilar membrane to vibrate, which in turn causes neurons to fire.

Table 2.1 Critical band frequencies (approximate)

Band	Frequency Range	Band	Frequency Range
0	0 – 50	14	1735 – 1970
1	50 – 95	15	1970 – 2340
2	95 – 140	16	2340 – 2720
3	140 – 235	17	2720 – 3280
4	235 – 330	18	3280 – 3840
5	330 – 420	19	3840 – 4690
6	420 – 560	20	4690 – 5440
7	560 – 680	21	5440 – 6375
8	680 – 800	22	6375 – 7690
9	800 – 940	23	7690 – 9375
10	940 – 1125	24	9375 – 11625
11	1125 – 1265	25	11625 – 15375
12	1265 – 1500	26	15375 – 20250
13	1500 – 1735		

The ear performs an effect similar to a set of bandpass filters on the sound. We call the bandwidth of those “filters” *critical bands.* The effect of the critical bands is that a constant volume sound will seem louder if it spans the boundary between two critical bands than it would were it entirely contained within one critical band [1]. When we discuss perceptual characteristics of the ear, we usually use the Bark unit, where one Bark corresponds to the width of one critical band.

Knowledge of critical bands has led to the discovery that if two sounds both have components in one critical band and if one is a certain amount louder than the other, the quieter one will be imperceptible. Physically, this corresponds to the hair cells in the particular location being overstimulated and therefore unable to respond to lower magnitude vibrations. This phenomenon is called *simultaneous masking* and is the primary effect in the hearing system taken advantage of by audio coders. This is done by calculating a *masking threshold* based on the characteristics of the incoming signal, and making sure any error in representation is below this level, thereby rendering the error noise inaudible.

The calculation of the masking threshold is influenced by the character of the sound. For instance, a sound which has tonelike characteristics in a critical band is more able to mask noiselike distortion than vice versa.

Another characteristic of our hearing system which can be taken advantage of is the absolute threshold of hearing. This is the minimum loudness a sound must have at each frequency for it to be audible. An approximation to it is given by the following equation, [8]

$$T_q(f) = 3.64(\frac{f}{1000})^{-0.8} - 6.5e^{-0.6(\frac{f}{1000}-3.3)^2} + 10^{-3}(\frac{f}{1000})^4. \tag{2.4}$$

This can be taken advantage of in a similar way to the masking threshold: any sound components below this level can be discarded. There is a fundamental difficulty with using an absolute threshold of hearing however, since we do not necessarily know what amplification level the output signal will be played back at. Given enough amplification, the absolute threshold can be overcome at any frequency. In order to take advantage of this property of human hearing, we must therefore assume that the sound will be played back at a level which is equal to or less than the level of the original sound.

Another form of auditory masking is *temporal masking*. When a loud sound is immediately (between 50 ms and 200 ms) followed by a quieter sound, the second sound can be masked by the previous sound. To some degree also, a quiet sound followed very closely (5 ms) by a loud sound can be masked. The former effect is called forward masking, the other backward masking. In audio coding, because of its limited effect, backward masking is often disregarded. Overall, temporal masking creates less effective masking than simultaneous masking, and so in lower complexity coders is sometimes disregarded entirely.

Pre-echo is a phenomenon which occurs when a section of relative silence is followed by a section of audible sound within one frame. When the masking threshold of this frame is

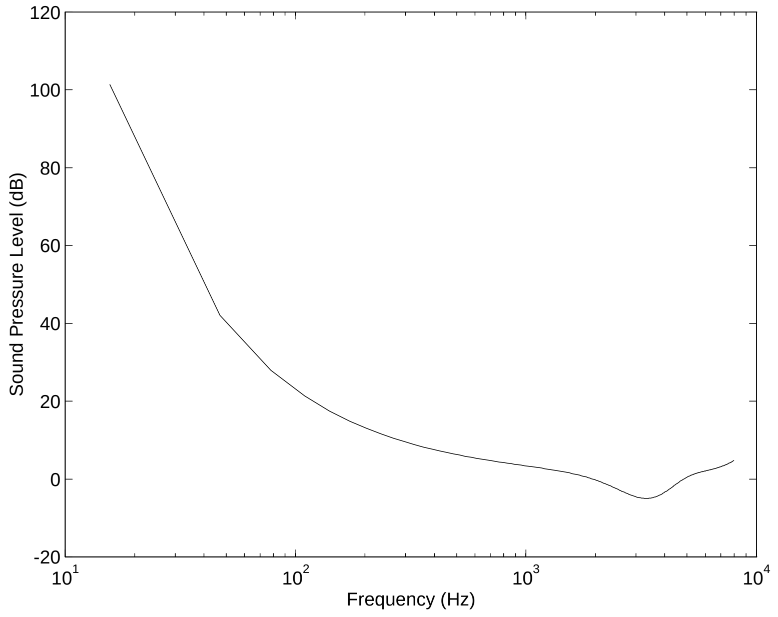

Fig. 2.2 Absolute threshold of hearing model

calculated, the later sound will cause the threshold to be raised above the level of the earlier silence. When calculations are subsequently made about what noise can be tolerated, it will be done such that the noise will likely be audible during the period of silence. Several methods have been proposed to deal with this problem, the most common one being a reduction in window length around transients, attempting to reduce the amount of data over which the masking threshold is incorrectly calculated. When a transient is followed by a period of silence, the error in masking threshold is less important due to the effects of forward temporal masking.

2.4 Quantization

The quantization stage of an audio coder is where bits are assigned to represent the data. Two basic methods of this are scalar quantization and vector quantization.

In scalar quantization, a quantizer has a set of scalar values, each one assigned a bit representation. The nearest scalar value to the actual value we want to represent is the quantized value. These scalar levels can be arranged in either a uniform fashion, i.e., uniformly distributed from the highest expected value to the lowest expected value, or nonuniformly distributed. Uniform quantizers allow the designer to designate a maximum value for the error of any quantized value, while non-uniform quantizers can give a significant increase in accuracy, especially when the statistics of the incoming signal are known.

Vector quantizers use a bit representation for several variables at once. The values are arranged in vectors of pre-specified lengths. A collection of representative vectors are stored in a *codebook*, and for each vector of variables, the codebook vector which best matches this input vector is chosen from the codebook. The representation of that code in terms of bits is then transmitted. This method can significantly increase coding gain at the expense of greater processing time in order to find the best vector, and memory for holding the possible vectors [3, p. 332].

Quantization can be done predictively or non-predictively. In scalar quantization, one predictive scheme would be to take the last value found and subtract it from the current value, and encode that value. When encoding slowly changing or static values (like the frequency representation of a tone) this method can achieve significant coding gains. When the difference between subsequent values is larger than the value itself however, a non-predictive scheme will give better performance. Non-predictive scalar quantization schemes simply encode the current value. Predictive vector quantization usually subtracts every value in the vector from the previous frame from their corresponding element in the vector for the current frame. More complex predictive encoders may use more than one previous frame which can improve the prediction.

2.5 Chapter Summary

In this chapter we discussed the theory required for understanding the building blocks of audio coders, as well as some basic digital signal processing theory.

We discussed time to frequency mappings which are used in most audio coders in order to take advantage of models of audio perception, where the two most important mappings, or transforms, are the Discrete Fourier Transform (DFT), and the Modified Discrete Cosine Transform (MDCT).

We then discussed some windows which are applied to each frame of input samples in order to improve frequency analysis. Also we noted the constrants on windows so that they can be used in conjunction with an MDCT and the MDCT will retain its characteristic of perfect reconstruction.

Psychoacoustic modeling is a very important element of most audio coders, because it can be used to greatly decrease the amount of audio information required to be transmitted. The two basic elements most often used are simultaneous masking and temporal masking, where simultaneous masking has the greater impact.

How we quantize of the set of information we want to send to the receiver greatly impacts the quality of the reconstructed signal. Two basic types of quantization used are scalar quantization and vector quantization.

Chapter 3

Existing Audio Coders With Alterable Bit Rates

3.1 MPEG-1

The MPEG-1 audio standard was the product of four years' work by audio experts in the Moving Picture Experts Group. It supports two channels, sampling rates 32 kHz, 44.1 kHz, and 48 kHz, and bit rates from 32–192 kbps for a monophonic signal. MPEG-1 audio is a bit rate switchable coder. The coder implements three levels or *layers* of compression. Each layer increases the quality at a given bit rate while adding to complexity and delay.

The overall structure of all three layers is as follows. First, a filter bank is applied to the input. In parallel with this, a psychoacoustic model is applied to the data. This model is used in a bit allocation block. These bits in turn are used to quantize the information from the filter bank.

3.1.1 Layer 1

For the first layer, the filter bank consists of 32 high order filters. The output of these filters is downsampled by a factor of 32. This splits up the input signal into 32 equal width subbands. This approach was taken despite several disadvantages:

1. There can be some aliasing between adjacent subbands.

2. The application of the filter bank is not a lossless transformation.

3. Equal width subbands do not represent the critical bands of the ear.

The first two disadvantages are a fairly minor source of error [9, p. 62], and the third is dealt with when using layer 3. Each filter produces a single sample for every 32 input samples. Once 12 samples are produced from each filter, a frame can be formed.

The psychoacoustic model for the encoder is not entirely specified by the MPEG-1 standard and is implementation specific. The standard does supply two example approaches for applying a model, both of which follow the same basic steps. First, the data is transformed to the frequency domain. In both examples an FFT is used. The spectral coefficients obtained are then arranged into critical bands, and the energy in each critical band calculated. Next, a decision as to whether each band is tonal or noiselike in nature is made. From the energies and the nature of the signal, a masking threshold is be calculated for each band and compared with the absolute threshold of hearing. Finally the Signal-to-Mask Ratio (SMR) for each band is calculated.

For each subband for which the SMR is positive, a normalizing scale factor is introduced. These scale factors are encoded as side information and allocated 6 bits each. These allow more efficient quantization of the values in each subband.

The bit allocation procedure first calculates the Mask-to-Noise Ratio (MNR) for each subband:

$$\mathrm{MNR} = \mathrm{SNR} - \mathrm{SMR}. \tag{3.1}$$

Then, the subband which has the lowest MNR has bits allocated to it. The MNR is calculated again for each subband with the new bit allocations and the process repeats until all available bits are allocated.

Each subband has 16 pre-calculated uniform quantizers from which the best quantizer can be chosen such that the lowest distortion is obtained when the values in that subband are quantized. The choice of quantizer is sent as 4 bits of side information for each subband. The maximum resolution of each quantizer in layer 1 is 15 bits.

3.1.2 Layer 2

Encoding using layer 2 of the MPEG-1 codec is very similar to the layer 1 encoding procedure. The end results of the changes are a small increase in complexity for a reduction in bit rate and improvement in quality.

The largest difference between the two layers is that in layer 2, instead of encoding 12 filtered samples per subband in each frame, 36 are encoded. These are 3 groups of 12 samples. This is advantageous because if the scale factors for these groups are similar then only one need be encoded. The scheme reserves the ability to encode up to all three of the scale coefficients in cases of great difference. Another advantage is that a longer window can be used for applying the psychoacoustic model, giving greater accuracy for slowly changing signals. Other than using a longer analysis window, the psychoacoustic model generally remains the same.

Bit allocation is also performed in a similar method to layer 1, except that it applies to subbands of 36 samples rather than 12. The resolution of the quantizers changes from 15 to 16 bits. In order to counteract this increase in bits, the number of quantizers to choose between decreases at higher subbands.

3.1.3 Layer 3

Layer 3 offers substantial improvements in quality for an equivalent bit rate from layers 1 and 2, again at a cost of complexity. This audio codec has found widespread use lately, particularly for the purpose of distributing high quality audio data over the Internet, where it is known as “mpeg 3” or “mp3”.

The filter bank behaves similarly to layer 2 except that each filter is followed by an MDCT transformation. The purpose of this is to give better frequency resolution for the subsequent masking calculations and bit allocation. Furthermore, in order to control pre-echo, two different window lengths can be used: a long window of 36 points or a short window of 12 points. Note that these are MDCT windows which are overlapped by 50%; so, respectively, 18 and 6 new points are added each frame. Transition windows are used between short and long windows. A mixed mode is also available, where long windows are used for the lowest two frequency subbands and short windows are used for the rest, due to the higher tendency of lower frequencies to be tonelike.

Secondly, instead of assigning scale factors to the respective filter subbands, MDCT coefficients are grouped roughly in terms of the critical bands of the ear and scale factors are calculated from these, called *scale factor bands*.

Other techniques used to improve coding efficiency in layer 3 include reduction in aliasing from the filter banks, entropy coding of data values, and use of non-uniform quantization.

Two changes in layer 3 impact the bit rate scalability of the codec. First, a different scheme for bit allocation is used. A nested loop is formed where the inner loop adjusts the shape of the quantizer to be used such that it fits the available number of bits. The outer loop then evaluates the distortion from that bit configuration, and if the distortion is too high, the scale factor band is amplified. Secondly, layer 3 makes use of a *bit reservoir* which allows it to allocate more bits to frames which need them and take bits from frames which do not.

3.2 MPEG-2 Advanced Audio Coding (AAC)

The goal of MPEG-2 audio coding was to create a coder which could produce transparent quality audio for five channels at low bit rates. Five channels were desired for playing audio in theatrical settings: so that sound can be played from several directions. When it was finalized in April 1997, independent testing determined that transparent quality was achieved at 320 kbps for five channels [2, p. 789].

MPEG-2 audio supports up to 48 channels, sampling rates between 8 kHz and 96 kHz, and bit rates up to 576 kbps per channel. Like MPEG-1 audio, MPEG-2 AAC has three profiles, but their purposes differ. The *Main* profile is the highest quality profile and consequently it requires the most computation. The *Low Complexity* (LC) profile and the *Scalable Sampling Rate* (SSR) profile require less computation, and the SSR profile splits up the signal so that different bit rates and sampling rates can be used by different decoders. So, we can say that the Main profile is bit rate switchable, while SSR is bit rate adjustable.

All three profiles follow the same scheme, with a few modifications for each. First, an MDCT is performed on the frame of input data. The length of the MDCT can either be “long” at 2048 samples or “short” at 256 samples. Short windows are used for transients to avoid the pre-echo problem. The window used is also switchable. For signals with dense frequency components, a half-sine window is used, but for signals with more distant components, a window called the *Kaiser-Bessel Derived* (KBD) window is used. The advantage of the KBD window is its greater rejection of outlying frequency components. In all cases of window switching, transition windows are used for smooth transition and to retain perfect reconstruction for the MDCT. After transformation, the MDCT coefficients are filtered by the *Temporal Noise Shaping* (TNS) tool. The goal of this tool is to further reduce pre-echo effects and better encode signals with stable pitch [2, p. 804–806]. For the LC and SSR tools, the TNS is order limited. The MDCT coefficients are finally grouped into 49 scale factor bands, roughly equivalent to the critical bands of the ear.

In parallel with the MDCT transform, a psychoacoustic model very similar to the one used in MPEG-1 is applied to find masking thresholds.

For long windows, and for only the Main profile, a prediction tool is then applied to the MDCT data. For each spectral coefficient up to 16 kHz, a predictor is implemented, based on the last two frames of data. The predictor is subtracted from the current value of the spectral coefficient and the error is quantized. First however, for each scale factor band the coder determines whether prediction creates greater distortion than without it. If it does, prediction is not applied for that scale factor band. Omission of prediction for certain bands requires side information to be sent to the decoder, so the coder further determines if the need for side information increases distortion more than prediction helps it. In this case, prediction is not performed at all.

Quantization in MPEG-2 AAC is performed with a desire to keep any distortion below the masking threshold calculated by the psychoacoustic model, while keeping the number of bits used below the average number permitted per frame. Like MPEG-1 Layer 3, a bit reservoir is implemented so that for frames which require fewer bits than the average, some bits can be deposited into the reservoir and for frames which require more bits, they may be available.

The MPEG-2 AAC standard does not specify how quantization should be performed, it only dictates the format of the bitstream which is output from the encoder. It does however give an example of how it can be done.

The main features of the example quantization scheme are the use of scale factors which can be used to amplify their respective bands, non-uniform quantization, and entropy coding for both scale factors and spectral coefficients. The example quantizer is implemented as a nested loop.

The inner loop alters the step size of the non-uniform quantizers such that all the coefficients in a scale factor can be encoded, then applies entropy coding to the quantized data. If this data fits within the bit limit for that frame, the inner loop is finished. Otherwise, the step size must be increased again so that fewer bits will be used.

The outer loop takes the configuration determined by the inner loop and calculates for each scale factor band whether the distortion is below the masking threshold. If any one band has distortion above the threshold, that band will be amplified. This amplification increases the *Signal-to-Noise Ratio* (SNR) of the band after quantization, at the expense of more bits. Side information detailing the level of amplification needs to be sent to the decoder so that it can be reversed at the output. If the outer loop finds that none of the scale factor bands requires amplification, or that they all do, the quantization procedure ends.

The SSR profile of MPEG-2 AAC requires several tools to be added to the coder. The primary difference is the addition of a *Polyphase Quadrature Filter* (PQF) bank. This filter bank splits the signal into four frequency bands of equal width. An MDCT is applied to the outputs of these filters, and the encoding proceeds as normal. This allows the decoder to ignore one or more of the filter banks so that the bit rate and sampling rate are limited to a level the decoder can handle. At the decoder, an inverse filter bank must be applied to recombine the signal.

3.3 MPEG-4

The aim of the MPEG-4 audio standard is to take different coding tools and integrate them into a coherent package. When completed, it will include two perceptual audio coders and a speech coder, as well as text-to-speech and MIDI capabilities. Because the standard is not yet complete, we will only discuss briefly the aspects of the coder relevant to our work.

The primary general audio coder in MPEG-4 Version 1 is the MPEG-2 AAC standard. As discussed in the previous section, this allows for high quality audio compression across a broad range of bit rates and sampling rates.

Some changes to the MPEG-2 AAC standard are being incorporated into the MPEG-4 standard.

One of the changes in the current draft is *Perceptual Noise Substitution*. Using this tool, medium-to-high frequency (above 4 kHz) scale factor bands are classified as being either noiselike or not, using the psychoacoustic model. If a scale factor band is classified as noiselike, the energy of that band is calculated and transmitted in place of the scale factor, and the coefficients in that band are set to zero. This energy is then used in the decoder to create noise in that scale factor band.

A second change of interest is the introduction of *Bit-Sliced Arithmetic Coding* (BSAC). This tool introduces more bit rate scalability by allowing a decoder to decode a 64 kbps stream using a minimum of 16 kbps and a step size of 1 kbps [10]. This flexibility comes at a price of increased side information.

The other general audio coder included is *TwinVQ* from NTT Human Interface Laboratories. This coder is focused on lower bit rates than MPEG-2 AAC, and is designed to allow portions of its bitstream to be disregarded by the decoder in order to implement both adjustable bit rate and sampling rate.

The TwinVQ coder uses four "layers" of encoding. The input to each layer is a signal with a 24 kHz sampling rate. The first of the four layers is used to encode the 0–8 kHz frequencies at a rate of 8 kbps. The other three 8 kHz wide layers can be placed at any frequency location such that the frequencies they cover are between 0 kHz and 12 kHz. Where two or more layers overlap, greater coding quality will be obtained due to the addition of more bits for those frequencies. Decoders will be able to ignore one or more of the layers, thereby decreasing the sampling rate and bit rate. Details of the TwinVQ coder can be found in [11].

MPEG-4 audio will also allow what is called *Tools for Large Step Scalability*, which is designed to allow several tools to combine such that high bandwidth decoders could use all the tools and lower bandwidth receivers would use only the "core" tools. For example, a speech coder could be used in combination with an MPEG-2 AAC stream [12].

3.4 Chapter Summary

In this chapter we discussed several of the more popular audio coders available or in development as of this writing: those produced by the Moving Picture Experts Group (MPEG).

The MPEG-1 audio standard in particular is well known for its widspread use in distributing audio files over the Internet. It consists of three different layers, with the first layer being the least complex and having the least quality at a given bit rate, while the third layer is the most complex with the highest quality at a given bit rate. MPEG-1 has a switchable bit rate, and has some adjustable bit rate features.

MPEG-2 Advanced Audio Coding (AAC) is less well known than MPEG-1, but has been evaluated as being higher quality at an equivalent bit rate. The two coders differ significantly in structure. For example, MPEG-1 uses a filter bank to divide the audio into 32 subbands, while AAC applies a time to frequency mapping directly on frames of data. AAC is also bit rate switchable and has some adjustable bit rate features. It also has the ability to send audio data at one bit rate and sampling rate and have it received by different receivers at several different bit rates and sampling rates.

MPEG-4 audio is still under development as of this writing, but shows promise of being quite scalable, incorporating an updated version of MPEG-2 AAC as well as various other tools for audio and speech coding.

Chapter 4

Audio Coder Description

4.1 Overview

Our audio coder is based on the work done by Najafzadeh-Azghandi and Kabal in Narrowband Perceptual Audio Coding [4] and in [13]. Their coder will be called the *original coder* in this text. The most significant alterations to the original coder have been made to incorporate bit rate and sampling rate scalability. The changes will be described where applicable, and summarized in the final chapter.

Our coder is a transform coder in the sense that it uses a time to frequency mapping and then encodes the frequency components obtained. These transform coefficients are used to obtain the masking threshold for each critical band for that frame. The gain of each critical band is obtained and encoded along with the perceptually important parts of the spectrum as a scale factor. Using the masking threshold and the bit rate of the current frame, bits are dynamically allocated to the different critical bands, which determines the accuracy of quantization for the coefficients in that critical band. Bits are also dynamically allocated for the scale factors. This is in contrast to the original coder which had a fixed number of bits assigned to scale factors.

The encoder will accept sampling rates up to and including 16 kHz. The original coder accepted only an 8 kHz sampling rate. The bit rates can be set in various ways. First, a constant bit rate from 4 kbps to approximately 16 kbps can be used. Secondly, bit rates can be changed midstream, to any bit rate accepted by the coder. Thirdly, the minimum bit rate required for good quality can be calculated and used on a frame by frame basis by the encoder. In this case only a maximum bit rate would be specified, though this too can

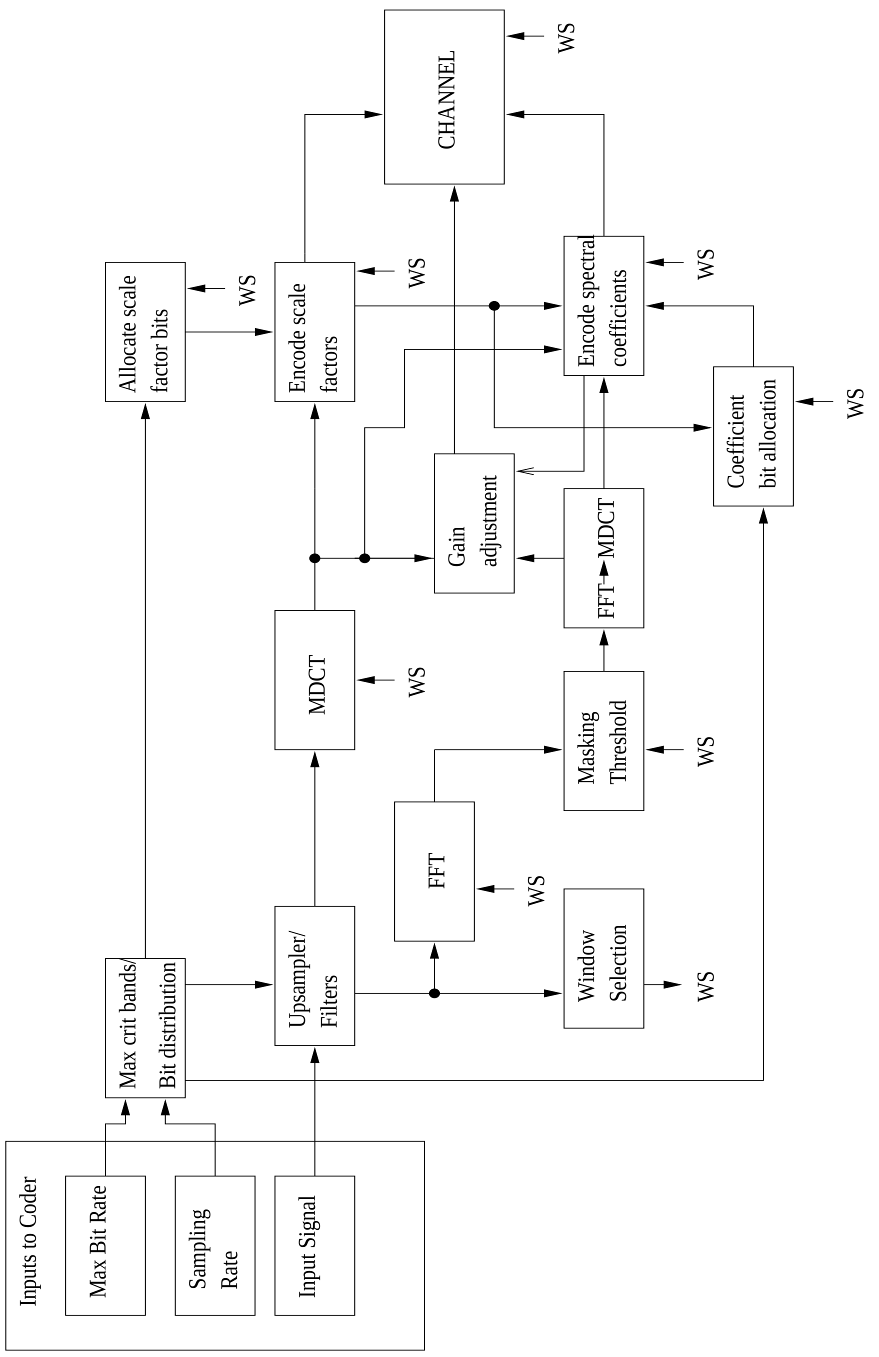

Fig. 4.1 Block diagram of the encoder

be adjusted midstream. This coder is therefore fully bit rate scalable. The original coder was not designed for a changing bit rate, and operated at a fixed 8 kbps.

4.2 Sampling Rate Change

The first step in the coder is that the input data is read in and upsampled to 16 kHz, which is the sampling frequency the rest of the coder operates at. The amount of data read in for each frame depends on the sampling rate of the input. It is calculated by multiplying the sampling rate of the input signal by desired length in time of the analysis window, in our case 31.5 ms.

Raising the sampling rate of a signal usually begins with inserting zeros between individual samples. For example, if upsampling by a factor of two, one would insert a single sample with a value of zero after each existing sample. The effect of this operation on the signal is that “images” of the base spectrum of the signal are duplicated in the higher frequencies which are now available due to the higher sampling rate. In order to remove these images, the signal is passed through a low-pass filter. The higher quality the filter, the more effectively the images are removed.

This form of upsampling is limited however, since we can only upsample by an integer factor. To solve this problem, the usual approach is to upsample the signal to a sampling rate which is divisible by an integer which will allow downsampling to the desired rate. For example, upsample a signal sampled at 8 kHz by 7 to 56 kHz, then downsample by 4 to 14 kHz. Downsampling is the removal of a number of samples corresponding to the factor we want to downsample by, so if we were downsampling by a factor of two, we would remove every second sample. Downsampling introduces distortion in the form of *aliasing*. Again, to avoid this distortion, a low-pass filter is usually employed. When doing the entire upsampling and downsampling operation, a single filter can be used for both. For more details on the basics of upsampling and downsampling see [14].

This upsampling and downsampling routine can however become impractical for many resampling operations. For example to upsample an 8001 Hz signal to 16 kHz would require a very large upsampling factor, which may exceed the amount of memory available for the operation. For this reason, for our coder we perform upsampling using resampling routines from [15].

In the case that 16 kHz divided by the input sampling rate can be expressed as a ratio of small integers, the routines perform the resampling operation as it is discussed above. In the more general case, such as upsampling 8001 Hz to 16 kHz, a different procedure is followed.

The approach used is, conceptually, to create a continuous time signal from our discrete time signal, and resample it. To create this signal, the input is first upsampled. The factor of upsampling is a parameter passed to the routine. The higher the upsampling factor, the more accurate our filtering is. Increasing the upsampling factor does not greatly affect processing time, but it does consume more memory. Next, an FIR filter is constructed by applying a Kaiser window [14] to an ideal lowpass filter response. This filter is implemented in polyphase form to avoid unnecessary computation [14, p. 794]. The upsampled signal is convolved with this filter to suppress the repeating spectrum created. The output of this is then convolved with a linear interpolation function

$$H(\omega) = (\frac{\sin(\omega)}{\omega})^2. \tag{4.1}$$

One effect of this filtering is to reduce the spectral images at multiples of the upsampled sampling rate of the original signal, which are left intact by the first filter. Second, the resulting signal is represented as a continuous time signal. This is accomplished by calculating the value between samples based on its “distance” from adjacent samples. We are therefore evaluating the value of the signal at non-integer sampling points. In this way we can resample this signal at any sampling rate to return it to a digital signal. All input signals are passed through this routine to bring them to our desired 16 kHz sampling rate.

Note that although our coder can operate on input signals with sampling rates as low as 4 kHz, such a low input sampling rate severely restricts the available audio frequencies and would most likely be judged unacceptable by listeners for both music and speech. The main purpose for this capability is for when the coder is operating in bit rate adjustable mode: during a time of extremely low bandwidth, perhaps due to other traffic, some (low quality) audio would still be transmitted to the receiver, which would likely be deemed preferable to receiving no audio at all.

Along with interpolation factor (number of zeros to insert between samples, plus one), we also tell the routines how complex we want our polyphase filter to be. For our coder we decided to use an interpolation factor which would raise the sampling rate to more than double our target 16 kHz. So for example a signal with sampling rate of 12 kHz would be

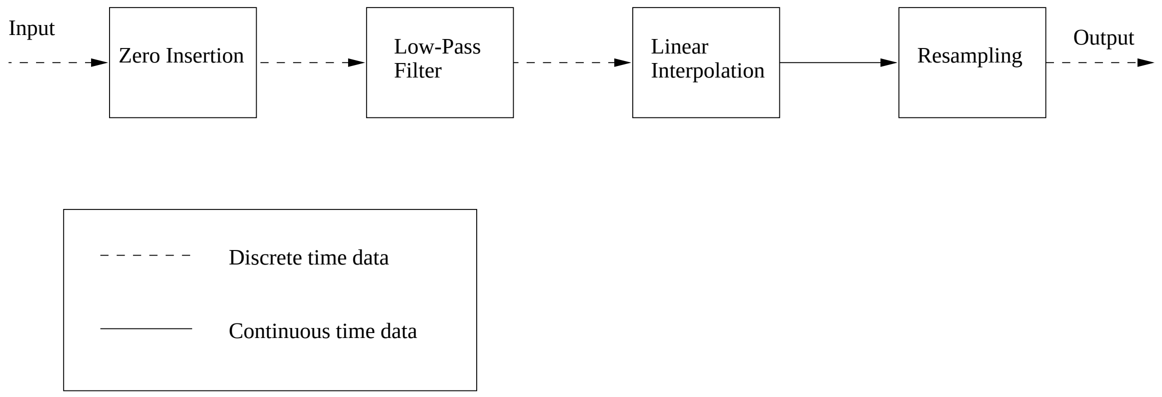

Fig. 4.2 General resampling operation

upsampled by a factor of 3 and an 8001 Hz sampling rate signal would be upsampled by a factor of 4. As explained earlier, this setting only affects memory use, so at the expense of more memory we could have a more accurate filter with very similar processing time. We settled on 40 dB of attenuation for the stopband of the filter because most masking threshold offsets are less than this amount. To reduce computation and delay, we also limited the number of coefficients for each polyphase filter to 10. Note that the full number of coefficients is over 10 multiplied by the interpolation factor, but only 10 are used for calculating each output value. We found this level of accuracy to give adequate quality.

The original coder only operated at a fixed 8 kHz and no sampling rate changes were performed.

The last procedure applied to the time domain input data before the time-to-frequency transform is a low-pass filtering according to the number of critical bands we are using for this particular frame. This is a set of pre-calculated 10th order Butterworth filters, one for each possible number of critical bands, made up of second order sections to reduce the effect of filter coefficient quantization errors. Since our coder only uses a limited number of critical bands for each frame of data, it is advisable to remove the portions of the signal which will be unused. The description of how the number of critical bands is chosen for each frame is described in a later section. The original coder used a fixed number of critical bands, and used a fixed low-pass filter.

4.3 Time To Frequency Transform

The time to frequency mapping is accomplished using an MDCT on each frame of data, where the frames are overlapped by 50%. In the section on masking, we will also perform, in parallel, an FFT on the same data in order to more accurately use our masking threshold model.

Our input signal, upsampled to 16 kHz is analyzed using a frame length of 510 data values for a long frame, or three frames of length 170 for short frames. The original coder used windows of length 240 and 80 respectively. We are using FFT techniques for our psychoacoustic analysis, so a frame length which is close to a power of 2 is advantageous.

The window we use is the half-sine window. This window has the property of perfect reconstruction with the MDCT which allows us to accurately determine the amount of noise we introduce in our coder since it will be entirely due to the internal quantization. It also has good frequency selectivity for dense frequency components.

In order to avoid difficulties with pre-echo and to increase our temporal resolution for transitional signals, the coder switches to shorter windows when a transition is detected. A transition is detected in the following manner. Because forward temporal masking is a longer lasting effect and would probably mask any noise caused by a descending transition, we want to concentrate on switching to short windows for rising transitions. To determine if a transition is occurring, energies are determined for 170 groups of 3 points in the input signal. Then we find

$$r = \max(\frac{e_{j+1} - e_j}{e_j}), \tag{4.2}$$

where e_j is the energy of the jth interval. If r is found to be above a threshold, it is determined that a transition is occurring. When this happens, a transition window is used. This transition window will be followed by frames of three short windows until it is determined that the signal is no longer in transition, at which point a second transition window will be applied before starting with long windows again. These transition windows are necessary to maintain perfect reconstruction in the MDCT transforms.

At this point in the coder, the number of critical bands we can take advantage of are calculated. This coder only encodes the audio frequencies which it can adequately represent given a bit rate, and given the sampling rate of the input signal. So, first we must determine the highest audible frequency which is possible to encode. This is taken from the sampling rate of the input signal. From the Sampling Theorem, we can theoretically recover audible frequencies from a digitally encoded signal up to half of the sampling rate of the signal [14]. So, for an 8 kHz signal, we can only possibly recover audible frequencies up to 4 kHz. In this way we can eliminate the audible frequencies which are irrelevant to the signal and which we do not need to encode. We describe this upper limit by assigning a maximum critical band above which we will not encode.

Next we need to use the available bits to determine how much of the available audio frequency range we are able to encode at a reasonable level of accuracy. We describe the number of audio frequencies which we need to encode by calculating the number of critical bands we are including in our encoding. This use of critical bands as an upper limit is a somewhat coarse method but significantly reduces the information we need to send to the receiver.

The number of critical bands to be used is initially set to the minimum number, 15, which approximately corresponds to 2 kHz of audio frequencies. The number of bits available for a window of data is determined by dividing the maximum current bit rate (in bits per second) by the number of frames per second.

This overall bit rate must be divided between the two quantization sections of the coder, the scale factors and the spectral coefficients. Bits are allocated between these two sections, and within these sections bit allocation occurs to further improve the efficiency of our bit use. These two further bit allocation procedures will be described in their respective sections. Note that the original coder only performed dynamic bit allocation for spectral coefficients, and its overall bit rate was fixed.

Of the available bits, a portion is allocated to scale factor encoding such that a minimum level of quality would be upheld. The rest of the bits are allocated to coefficient encoding (minus the small number of header bits). Bits are then iteratively added to the scale factor allocation and subtracted from the coefficients, until either the minimum level of acceptable coefficient bits are allocated, or a sufficient number of bits is allocated to both the scale factors and the spectral coefficients such that the addition of a critical band would allow a reasonable level of encoding quality to be retained. In this case, the number of critical bands would be increased by one, and the process would repeat.

If the coder is in adaptive bit rate mode, the number of bits used for scale factor encoding is fixed at a level approximated to produce a minimum level of quality. This number of bits does not change unless the maximum bit rate changes.

At this point we know how many critical bands we are going to use initially for encoding. The number of critical bands can change depending on the bit rate, so for example if the bit rate were to decrease by a certain amount, the critical band calculation would repeat and decrease the number of critical bands used when bits would be spread too thinly over the bandwidth allocated.

4.4 Masking

The coder employs both simultaneous and temporal masking to eliminate irrelevancies in the signal.

4.4.1 Simultaneous Masking

The simultaneous model we used is obtained from Johnston [16], Zwicker [17], and Schroeder *et al* [18].

First we perform an FFT on the frame of data. The resulting spectrum is used to calculate the energy in each of the critical bands,

$$G(i) = \sum_{k \epsilon \mathbf{B}} X_{\mathrm{DFT}}^2(k), \tag{4.3}$$

where $\mathbf{B}$ is the set of coefficients in critical band i. These energies are called the Bark spectrum. The Bark spectrum is then convolved with a spreading function. The purpose of the spreading function is to represent the effects of masking across critical bands [19]. The equation we use is taken from [18],

$$10 \log B(i) = 15.81 + 7.5(i + 0.474) - 17.5(1 + (i + 0.474)^2)^{\frac{1}{2}}, \tag{4.4}$$

where i is the critical band index and $B(i)$ is the spreading function. After the convolution, the resulting spread energies are converted to the log domain.

We now have the log-spread Bark spectrum for the signal. Next we calculate the tonality factor in each critical band. We know that when the portion of a signal in a particular critical band is noiselike in character, it is less effective for simultaneous masking in that band than a signal portion which is tonelike in character. Because of this, we calculate the tonality in each band in the following way, following [20].

The equation

$$\tilde{X}(i) = 2X(i)' - X(i)'' \tag{4.5}$$

gives us $\tilde{X}(i)$ which is a prediction of the critical band subvector for the current frame. We denote the subvectors of the frame previous to the current one with $'$, and two frames previous with $''$. The relative error in our prediction is given as

$$\delta = \frac{\|X(i) - \tilde{X}(i)\|}{\|X(i)\| + \|\tilde{X}(i)\|}. \tag{4.6}$$

From this, we assign a tonality coefficient

$$a = \min\left(1, \max\left(-0.3 - 0.43\log(\delta), 0\right)\right). \tag{4.7}$$

The tonality coefficient gives us the final value of the simultaneous masking coefficients (for a DFT spectrum),

$$m_{s_{\text{DFT}}} = \max\left(a(14.5 + i) + 5.5(1 - a), T_q\right), \tag{4.8}$$

where T_q is the absolute threshold of hearing given in Equation 2.4, and i is the critical band index.

4.4.2 Temporal Masking

The coder also takes advantage of forward temporal masking to increase coding efficiency. The model used in this coder is the one proposed in [21],

$$m_{t_{\text{DFT}}} = \alpha + \beta \exp\left(\frac{-f}{\gamma}\right), \tag{4.9}$$

gives the temporal masking in dB, where f is frequency in Hz. The other three variables are defined as

$$\begin{aligned}
\alpha &= 0.001L^2 + 0.2267L + 17.7142, \\
\beta &= -0.0047L^2 + 1.2256L - 24.32548, \\
\gamma &= -0.0002L^4 + 0.0546L^3 - 5.4685L^2 + 234.7411L - 3325.035,
\end{aligned} \tag{4.10}$$

where L is the sound level (dB) of the previous frame.

4.4.3 Total Masking Threshold

Finally the two masking thresholds are combined. Again following [21] we determine the combined threshold as

$$m_{\mathrm{DFT}} = \max((m_{s_{\mathrm{DFT}}}^{0.3} + m_{t_{\mathrm{DFT}}}^{0.3})^{\frac{1}{0.3}}, T_q). \tag{4.11}$$

Note that before the above equation we have converted both masking thresholds from logarithmic to linear.

To compare our masking threshold with the input MDCT coefficients we will use, we convert the masking coefficients to MDCT masking coefficients [13]. The relationship between DFT coefficients $X_{\mathrm{DFT}}(k)$ and MDCT coefficients $X(k)$ is

$$X(k) = \sqrt{\frac{2}{M}}|X_{\mathrm{DFT}}(k)| \cos\left(\frac{2\pi n_0(k+0.5)}{N} - \angle X_{\mathrm{DFT}}(k)\right), \tag{4.12}$$

where $n_0 = \frac{(M+1)}{2}$ and $M = \frac{N}{2}$. N is the number of time samples per input frame. The simultaneous masking threshold is finally calculated such that

$$m(k) = \frac{2}{M} m_{\mathrm{DFT}}(k) \cos^2\left(\frac{2\pi n_0(k+0.5)}{N} - \angle X_{\mathrm{DFT}}(k)\right), \tag{4.13}$$

and then converted to logarithmic form.

4.5 Scale Factor Encoding

In order to decrease the range of values the spectral coefficients occupy, we want to normalize them by scale factors. These encoded scale factors will need to be sent to the receiver so that the signal can be recovered. We find a scale factor for each critical band used in the coder.

We obtain the vector of unquantized scale factors by calculating the energy in dB for each critical band (Bark spectrum for MDCT coefficients),

$$O(i) = 10 \log_{10}\left(\sum_{k \epsilon \mathbf{B}} X(k)^2\right). \tag{4.14}$$

Due to the 50% overlap of MDCT windows and the similarity of successive frames for tonelike signals, we are able to use prediction to achieve some coding gain. Each frame of data sent includes a flag stating whether a predictive or non-predictive scheme is being used for the scale factors.

The decision whether or not to use a predictive scheme rests on a spectral distortion measure. First we find the average gain for the scale factors

$$G = \frac{1}{C}\sum_{i=1}^{C} O(i). \tag{4.15}$$

where C is the number of critical bands being used. This is then subtracted from each of the scale factors to give $O_n(i)$. The spectral distortion is given by

$$D = \sum_{i=1}^{C}(O_n(i) - O_n(i)')^2, \tag{4.16}$$

where $O_n(i)'$ denotes the normalized scale factor for the previous frame. If the spectral distortion D is found to be less than 6 dB the predictive scheme is used, otherwise the non-predictive scheme is used.

The criterion used for finding the distortion for vector quantization calculations in the following discussion is the basic squared error,

$$d = (X^{(i)} - \tilde{X}^{(i)})^2, \tag{4.17}$$

summed over the length of the vector.

At this point bits must be allocated to the scale factor encoding sections. We will describe the bit allocation at the end of this section so that the encoding procedure can be discussed.

For both scale factor quantization schemes we perform several steps of quantization on the vector of scale factors. This is because a one step vector quantizer would have to be very large in order to accommodate as many bits as we are allocating to it. Such a large vector quantizer would not be feasible to implement because a search would take too long. Therefore, we perform several quantizations on the same vector in an attempt to achieve approximately the same accuracy as a single step vector quantizer with greatly reduced computation.

4.5.1 Predictive Scheme

The predictive scheme for the scale factors begins by encoding the gain G referred to above. It does this by subtracting it from the gain of the previous frame, and that difference in gain is quantized using a statistically based scalar quantizer, giving us $\hat{G}$. All of the individual critical band gains $O(i)$ are then normalized by this quantized gain giving $O_{\hat{n}}(i)$.

Next, a collection of *predictor matrices* are applied to the $O_{\hat{n}}(i)$ of the previous frame. The one that gives the best estimate of the current vector is chosen. Predictor matrices are designed to optimize prediction while keeping computation reasonable. A full discussion can be found in [4], but essentially they are matrices of C by C size, where only the main diagonal and the adjacent diagonals are nonzero. It was found that most of the other diagonals are close to zero after optimization and can be disregarded to reduce computation. The index of the matrix which gives the best prediction is encoded and sent to the receiver.

The vector which results from this prediction is then subtracted from the original $O_{\hat{n}}(i)$ to give $P(i)$, which is then vector quantized. This is then subtracted from $P(i)$, giving us another remainder, $Q(i)$, which is also vector quantized.

To summarize, the predictive scheme sends the following information to the receiver.

1. Quantized differential gain index
2. Index of the best predictor matrix
3. Index of first difference vector quantization
4. Index of second difference vector quantization

4.5.2 Non-Predictive Scheme

The non-predictive scheme works in a similar manner. First, the gain G is encoded non-differentially, which usually requires more bits for the same level of accuracy than predictive encoding does. Next, this quantized gain is subtracted from $O(i)$ to give $O_{\hat{n}}(i)$, and this vector is vector quantized. Now, predictor matrices are applied to the output of the vector quantizer to find which one best approximates $O_{\hat{n}}(i)$. Finally, the predicted vector is subtracted from $O_{\hat{n}}(i)$ and the result vector quantized.

So, the data sent in the non-predictive scheme is

1. Quantized gain index
2. Index of vector quantized $O_{\hat{n}}(i)$
3. Index of predictor matrix which best estimates $O_{\hat{n}}(i)$
4. Index of quantized difference vector

The previous discussions have been for the case of long windows (the most common situation). When the current window is short, the encoding performs identically except that the second stage of vector quantization is dropped in both the predictive and non-predictive case.

4.5.3 Scale Factor Bit Allocation

Bit allocation for the scale factor quantization is done by allocating a fraction of the available bits to each of the four parts of the encoding process. This ratio was found by evaluating quantization distortion in the scale factors for various configurations until a good compromise was found. In the predictive scheme, the gain, being differential, requires fewer bits than in the non-predictive scheme. We can also expect the predictor matrices to give us more useful results in the predictive method than in the non-predictive case. Therefore, more bits are allocated to the matrices in the predictive scheme and more bits are allocated to gain encoding in the non-predictive scheme. Approximately equal numbers of bits are assigned to each of the vector quantizations in both cases. Any remaining bits are added iteratively to each of the four sections.

4.6 Coefficient Encoding

Once we have the scale factors, we are able to encode the normalized MDCT coefficients.

To increase our coding gain, vector quantization is used for encoding the coefficients. Our MDCT transforms use 510 points, so we will be encoding up to 255 MDCT coefficients, depending on the bandwidth of our encoded signal. For vector quantization, anything above 10–12 elements is quite intensive in terms of memory and computation [3, p. 410]. This is because for each element we add to our vector, we must increase the size of our codebook in order to achieve the same accuracy. Therefore we need to reduce the dimensionality of the vectors to make the encoding computationally feasible. We split up the spectral coefficients into critical bands, because this will result in fewer coefficients in the vectors at lower frequencies, leading to more accurate encoding of those coefficients, given enough bits. This is done because our ears are more sensitive to low frequency information than high, so the higher coefficients can withstand the more noiselike representation which will result from slightly less accurate encoding. In some of the highest critical bands however, the number of coefficients can exceed 40, and so these vectors are further split into more workable vector lengths. The original coder, since it only used 17 critical bands, where the vector lengths are reasonable, did not need to split its critical bands.

In the case of short windows, many of the critical bands contain only a single coefficient. In order to increase the efficiency of the quantization therefore, the coefficients for short windows are split up into equally sized vectors.

Table 4.1 Division of spectral coefficients into critical bands

Band	Long Window	Short Window	Band	Long Window	Short Window
0	1		12	41 – 48	21 – 24
1	2 – 3	1	13	49 – 55	25 – 27
2	4	2	14	56 – 63	28 – 31
3	5 – 7	3	15	64 – 74	32 – 37
4	8 – 10	4 – 5	16	75 – 87	38 – 43
5	11 – 13	6 – 7	17	88 – 104	44 – 52
6	14 – 17	8	18	105 – 122	53 – 61
7	18 – 21	9 – 10	19	123 – 150	62 – 75
8	22 – 25	11 – 12	20	151 – 174	76 – 87
9	26 – 30	13 – 15	21	175 – 204	88 – 102
10	31 – 36	16 – 18	22	205 – 246	103 – 123
11	37 – 40	19 – 20	23	247 – 256	124 – 128

Now we must allocate the bits designated to coefficient encoding to each of the critical bands. Here, we can take advantage of the masking threshold we have calculated in order to give bits to the most perceptually important critical bands. In this way we can do our best to keep the noise introduced by quantization below the masking threshold and thus reduce the audibility of the noise.

The bit allocation begins by calculating the SMR for each critical band. The higher above the masking threshold the signal is in that critical band, the more bits it will require to eliminate audible quantization noise. The SMR is calculated by subtracting the masking threshold from the quantized Bark spectrum,

$$\mathrm{SMR}(i) = \hat{O}(i) - m(i), \tag{4.18}$$

where i corresponds to the critical band and both masking and Bark spectrum are in the log-domain. Now, bits are assigned according to the following formula [4]:

$$b(i) = \max\left(\frac{\mathrm{SMR}(i) b_T}{\lambda(i) \sum_{j \in \boldsymbol{\Omega}} \left(\frac{\mathrm{SMR}(j)}{\lambda(j)}\right)}, 0\right), \tag{4.19}$$

where $\boldsymbol{\Omega}$ is the set of critical bands with a positive SMR, and b_T is the total number of bits available for coefficient quantization. The value $\lambda(i)$ is an evaluation of how much the distortion of the overall encoding would be decreased by adding a bit to critical band i. This is calculated by forming a linear approximation to the slope of the rate-distortion curve. Points on the rate-distortion curve are found by encoding many vectors at each bit rate for a particular critical band and calculating their average distortion.

The $b(i)$ found from this procedure will have a fractional component. For each critical band the fraction is dropped. If the coder is operating in adaptive bit rate mode, this $b(i)$ will give the number of bits to be used for coefficient encoding for the current frame. Otherwise, the remainder of the bits will be assigned iteratively based on the Noise-to-Masking Ratio (NMR),

$$\mathrm{NMR}_i = \hat{O}(i) - m(i) - \lambda_i b_i. \tag{4.20}$$

The NMR tells us which critical band has the highest level of perceptually significant noise. In each iteration the NMR for each band is calculated and one bit is allocated to the band with the highest NMR.

To quantize the coefficients, the masking threshold is used to aid in accurately quantizing the perceptually important parts of the spectrum. To accomplish this, a specialized distortion measure is used for determining the vector to be chosen from the codebook,

$$d = \max\left(\frac{|X(i) - \hat{O}(i) X_c(i)|^2 - m(i)}{X(i)^2 + m(i)}, 0\right), \tag{4.21}$$

where $X_c(i)$ is the codebook vector of normalized coefficients. This error criterion makes sure that well masked coefficients are given less weight than others in terms of finding a good code vector. Coefficients which are completely masked can be completely disregarded.

In this way, codebook indices representing the MDCT coefficients are chosen for transmission.

4.7 Gain Adjustment

When the bit rate of our coder is approximately 8 kbps or less, we have very few bits to spare for the two major sections of the coder. When the bit rate is higher however, the level of accuracy we can obtain with our current scheme reaches a limit. Due to complexity considerations, we do not want to use codebooks which are very large for improving quantization accuracy. A better solution is to add features to the coder at higher bit rates.

One feature that we can perform is *gain adjustment* [4]. Gain adjustment will reduce errors in quantization by adding a final layer of improvement to the gain of each critical band. The scale factor adjustment for each critical band is calculated by optimizing the following equation [4]:

$$\rho_{\min} = \sum_{k=1}^{K} \max((X(k) - \rho\hat{X}(k))^2 - m(k), 0), \tag{4.22}$$

where ρ is called the *gain adjustment factor*, K is the number of coefficients in the current critical band, and $\hat{X}(k)$ is the subvector of quantized coefficients.

These gain adjustment factors can then be vector quantized and sent to the receiver and have been found to increase the quality of the output signal [4].

4.8 Codebook Creation

Our codebooks are created by taking a very large collection of test data and encoding it until test vectors are found. These test vectors are stored and used to create a codebook. Since we are using many differential operations in our coder, for example subtracting quantized values from unquantized values in the scale factor encoding, we must construct our codebooks in sequence. So when we have created our first codebook, we use that to generate test vectors for the next codebook, and so on.

The codebooks themselves are formed by applying Lloyd's algorithm [3, p. 367] to the test vectors. Lloyd's algorithm is an iterative optimization procedure which is generally used to improve an existing codebook. Ideally, we would want to create the codebook initially with a faster procedure like *pruning* [3, p. 359] and then apply Lloyd's algorithm to improve it.

Because our coder requires a codebook for each bit configuration, we need to reduce our use of memory. Therefore, we use *embedded codebooks.* Embedding codebooks is a technique which allows us to use a single codebook for various bit configurations. We do this by running a set of test data through the coder and seeing which code vectors are used most frequently. The code vectors are then ordered in descending order of most frequently accessed. This way, for bit rates lower than the maximum in a codebook, the coder will have a choice of the most popular vectors in that critical band. Clearly this is somewhat less optimal in terms of coding quality, but in our subjective testing with the original coder we found very little difference, while our memory requirements were reduced by 50%.

Table 4.2 Summary of data sent to the receiver

Encoder Section	Data Sent	Data Type
Preliminary	Bit rate change	Optional data
Transform	Window type	Flag
Masking	Predictive/Non-predictive	Flag
Gain Encoding	Mean scale factor gain	Scalar quantization
Gain Encoding	Scale factor prediction matrix	Prediction Matrix
Gain Encoding	First scale factor index	Vector quantization
Gain Encoding	Second scale factor index	Vector quantization
Coefficient Encoding	Coefficient vectors	Vector quantization

4.9 Decoder

The decoder for our audio coder consists of looking up vector quantizer indices in codebooks to reconstruct the MDCT coefficients and then applying an IMDCT on the coefficients.

The scale factors are reconstructed first, using a method depending on whether predictive or non-predictive quantization was used.

For predictive quantization, the decoder has preserved the mean gain and normalized quantized scale factors from the previous frame. It looks up the differential gain from the quantizer index and adds it to the previous mean gain. Then it uses the prediction matrix chosen by the encoder on the previous vector of scale factors to get the set of predicted scale factors. Next the two vector quantizations are looked up in their respective codebooks and added to the predicted scale factors. Finally, the mean gain is added to each of the scale factors.

In the non-predictive case, the mean gain is looked up directly from the quantizer index. The first vector quantization is then retrieved from its codebook, and the predictor matrix applied to this vector. Next, the second vector quantization is added to the result, and finally the mean gain is added.

The spectral coefficients are found by retrieving the index supplied by the encoder to look up the result in each codebook corresponding to the correct critical band. Each value in the vector looked up is multiplied by the scale factor for that critical band.

Lastly, these spectral coefficients are sent to an inverse MDCT to be converted back to PCM, ready for conversion to analog in order for the audio to be listened to.

4.10 Chapter Summary

In this chapter we have described the audio coder which we created by adapting a low bit rate, 8 kHz sampling rate audio coder for a wide range of bit rates and sampling rates.

First we discussed the method we chose for upsampling input signals. In order to make our coder very scalable, we desired a universal sampling rate converter. For this reason we chose the routines discussed in the chapter.

Next we gave the details of the time to frequency transformations we employed, as well as the window type and length applied to each frame of data.

We discussed how the available bits are divided between the two major quantization stages: scale factors and scalar coefficients. The bits allocated to each of these sections are further allocated within these sections.

In order to apply our psychoacoustic model, we generate a masking threshold for each critical band, based on both simultaneous and temporal masking. This set of masking thresholds is used in both the spectral coefficient and scale factor quantization sections.

Scale factor encoding is performed in several stages in order to decrease complexity while keeping quantization accuracy high. Bit allocation is performed between these stages in order to efficiently use the bits available for the scale factors. Based on the differences between frames, the quantization can take either a predictive or non-predictive form.

Spectral coefficient quantization is performed by vector quantizing each set of coefficients in its corresponding critical band. Bit allocation is performed in this section, dividing the bits between the critical bands, giving more bits to the more perceptually significant bands.

Finally, the procedure for creating codebooks for vector quantization, as well as the details of decoding the audio on the receiver side, are discussed.

Chapter 5

Evaluation of Our Approach

5.1 Coder Sections

5.1.1 Sampling Rate Scalability

The choice of the audio routines in [15] for our resampling operations was made due to their generality. The sampling frequency of any input signal can be changed to 16 kHz. We found with our chosen constraints on the interpolating filter there was no noticeable difference in the sound quality of the upsampled signal, while saving considerably on computation from the default (conservative) settings.

Table 5.1 Interpolating filter constraints

	Our coder	Defaults
Oversampling	To above 32 kHz	Factor of 24
FIR Filter Coefficients	10	68
Stopband Attenuation (dB)	40	80

Our second filtering operation, based on the number of critical bands we are using, removes unwanted frequencies from the signal, while adding fairly little complexity (an approximate 2% increase in overall encoding time). We did not detect any artifacts due to changing filters during transmission while testing.

We tested the sampling rate scalability by passing a segment of audio, which began with frequency components up to 8 kHz, through the coder in four different configurations:

1. Audio bandwidth 4 kHz, bit rate 8 kbps
2. Audio bandwidth 4 kHz, bit rate 16 kbps
3. Audio bandwidth 8 kHz, bit rate 8 kbps
4. Audio bandwidth 8 kHz, bit rate 16 kbps

We chose these cases in order to be able to clearly detect the effect of bit rate scalability on the range of audible frequencies. Also, we want to ensure that as the input sampling rate is increased, given high enough bit rate, the audio quality increases due to the presence of higher audio frequencies. When we compared the audio quality of the first two cases, we found the quality to be very similar. This is most likely because we have a limited number of critical bands in which the coder can allocate bits due to the lower audio bandwidth. Therefore, as more bits are allocated to those critical bands, the top level of quality possible given our codebooks is reached, and the quality can not improve any further. The third case also had very similar audio quality to the first case. This is because with a limited number of bits available, only a limited number of critical bands are allowed for bit allocation. At 8 kbps, the number of critical bands used is the same as the number used for the first case, which is very close to 4 kHz audio bandwidth. In the final case, higher frequency components could be detected from the output of the decoder, due to higher critical bands being included in the encoding, which improved the quality of the sound.

5.1.2 Transform and Window

The choice of using MDCT as a transform was a natural one, for the reasons discussed in Section 2.1: its critical sampling, reduction of block edge effects, and the availability of fast algorithms for it. The main disadvantage of using the MDCT over the FFT is that it is less clear how frequencies map onto MDCT coefficients. It is for this reason that we use an FFT for calculating our psychoacoustic model.

The choice of using a half-sine window was also an obvious one, as it is a very common window used in audio coding where an MDCT is employed. Some audio coders, such as MPEG-2 AAC, switch between the half-sine window and another window, with better rejection of outlying frequencies. We performed some informal testing using a Kaiser-Bessel Derived window in our coder and noticed little difference for our audio quality target.

Therefore, the added complexity of a window shape switching algorithm is probably not worthwhile for our coder.

5.1.3 Masking

Our masking threshold model was subjected to testing by transforming a signal using an MDCT, the masking threshold calculated, and any MDCT coefficients below the masking threshold were set to zero, and an IMDCT applied to the result. We were unable to detect any difference between the input and the output of this process.

One problem with our masking threshold is our use of the absolute threshold of hearing. Refer to Figure 2.3 for the model we used. In conditions where the level of playback can not be controlled, our calculations could be counterproductive. What is more, our model is designed to follow the threshold of hearing of a young listener, but the threshold may be substantially different for an older listener, so that playback level would be difficult to compensate for. At many frequencies however, this threshold gives us a considerably higher masking threshold than the simultaneous and temporal masking calculations do.

5.1.4 Quantization

For quantization, since we did not implement other quantization schemes, we evaluate the performance by determining how well the schemes are suited to bit rate scalability. The two sections of quantization, scale factors and spectral coefficients, use different methods for their quantization.

The scale factor quantization attempts to encode the entire vector of critical band scale factors together, applying different schemes to that vector. This method is taken from the original coder, and performs well there, but for bit rate scalability it has some limitations.

When we perform bit allocation, we must choose which of the layers of quantization is most important for the best quality encoding. It is not an easy task to determine on a frame by frame basis which of: average gain (differential or not), predictor matrix, first vector quantization, and second quantization, deserves more bits, as our psychoacoustic model does not help us.

For our spectral coefficient quantization, the system we use is quite well suited to bit rate scalability due to our ability to allocate bits based on a psychoacoustic model. Also, the distortion measure we use allows us to shape the audible quantization noise to follow the distribution of energy inside a critical band [4].

Our bit allocation scheme for spectral coefficients, using SMR, was chosen over other allocation schemes, such as Energy-based Bit Allocation (EBA) [4], following the analysis in the same article.

Our codebook creation has been performed using Lloyd's algorithm based on test vectors created from test audio data selected from speech, music, and combined speech and music. More sets of test audio data would have been advantageous to experiment with, but processing time for the creation of codebooks is significant and limited the amount of experimentation it was feasible to do. We do not create a "starting" codebook before using Lloyd's algorithm to optimize it, as discussed in Section 4.8.

5.2 Overall

The primary goal of our research has been accomplished, that the coder described in [4] has been adapted for use with a wide range of sampling rates and has been made bit rate scalable.

The final audio quality of the coded and decoded audio is not yet usable, with artifacts audible at all bit rates. In [4], the quality of the original coder was compared to G.729, a speech coder. The authors found that their original coder had much higher quality for music signals, and slightly lower quality for speech signals. They also compared the original coder to the 8 kbps Real Audio coder, a commercial audio coder, and again found their coder had higher quality. Our coder, as tested, has lower audio quality than the original coder we implemented, at the same bit rate. All evaluation of our coder was performed informally, by the author.

There are several possible reasons why our coder has significant audio artifacts. We believe that our codebooks are the primary culprit for the following reasons. The training data used to create the codebooks for use with the coder may not be optimal, since a simple selection of audio, speech, and audio with speech was employed. Also, the reordering done to create embedded codebooks placed frequently used vectors at the beginning of the codebook, but these vectors were chosen based on the use of the entire vector of scale factors, which is not usually the case. So the first few elements in the scale factor vector could be quite suboptimal. This may make the use of embedded codebooks for the current scale factor scheme unwise, despite their great savings in terms of memory required for the codebooks. Finally, it could also be, as discussed in [4] for the original coder, that harmonics are not being tracked accurately enough, and some kind of pitch measure would decrease distortion.

On the other hand, the difference in quality between the highest bit rate and lowest bit rate is reasonable. The largest difference in audio quality is the loss of higher frequencies due to the cutoff of higher critical bands when fewer bits are available. Also, when the coder is directed to change bit rates during transmission, the transition is smooth with no artifacts. The transition is, however, noticeable. If possible therefore, a gradual change would likely be less unsettling to a listener. This may not always be possible due to constraints on the amount of time available for a transition.

Since the coder is implemented in the C programming language, we are able to find a rough approximation of the time required for encoding. To test, we ran a 500 second audio file through the encoder on a 450 MHz Pentium II equipped computer, with no processor specific optimizations. Note that except for the use of an FFT algorithm for the psychoacoustic model, very few implementation decisions were made to reduce processing time, e.g., codebook structure.

Table 5.2 Encoder processing time

Maximum Bit Rate (bps)	Seconds of Processing Per Second of Audio
4000	0.6288
8000	0.8891
16000	2.6355
Original coder:	
8000	0.5679

5.3 Chapter Summary

In this chapter we discussed the degree of success we obtained in adapting the 8 kbps original coder for sampling rate and bit rate scalability. We analyzed the changes we made in order to determine what changes were most effective and which areas of the coder could be improved.

We were satisfied by the performance of our sampling rate scalability, in terms of the resampling routines as well as in our bit allocation schemes.

For our bit allocation schemes we found some performed well while others could profit from further attention. Specifically the scale factor quantization scheme was deemed not satisfactory from the bit allocation point of view.

The final audio quality was deemed not yet adequate for artifact free communication. However, any change in bit rate during transmission was accompanied by a corresponding increase or decrease in the quality of the audio, and introduced no additional artifacts.

Chapter 6

Summary and Future Work

6.1 Summary of Our Work

The purpose of our research was to look at bit rate scalability in audio coders, implement an existing coder, and adapt it for bit rate scalability.

The low bit rate audio coder detailed in [4] was well suited for the incorporation of bit rate scalability. We adapted this coder for bit rate scalability and sampling rate scalability, and implemented it in C in order to judge whether the design was feasible in terms of complexity.

The first change which needed to be made was the ability to handle input signals of differing sampling rates. To this end, audio resampling routines from [15] were incorporated, with limitations on the quality of the interpolating filter used, for purposes of complexity. These routines are able to upsample any signal from its original sampling rate to the standard 16 kHz which the coder uses. This upsampling does not increase the rate at which we need to send data to the receiver because a subset of quantized information is used for signals with sampling rates less than 16 kHz, based on the number of critical bands required to represent the audio bandwidth present in the signal.

As well as allowing different sampling rates for our input signal, we incorporated various methods of specifying bit rates. The first, bit rate switchability, allows for different maximum bit rates to be used for an input signal. Second, an adaptive bit rate allows a signal to use the minimum number of bits for adequate quality, changing bit rate on a frame by frame basis. A maximum bit rate must still be specified in order to establish a target level of quality. Third, bit rate adjustability allows the maximum bit rate to be altered during transfer, which can be applied to both switchable and adaptive bit rate modes.

Incorporation of bit rate scalability required several changes to the coder, primarily consisting of dividing bits between the two quantization sections: scale factors and spectral coefficients, and by incorporating a bit allocation scheme for the scale factor quantization section. The spectral coefficient quantization section already contained a good bit allocation scheme.

The number of critical bands we can encode at an adequate level of quality is refined using the number of bits available for each frame of data. A minimum number of bits is determined for quantization of both scale factors and scalar coefficients based on how many critical bands we are using. Then, an upper level is determined so that when there are sufficient bits for both quantizations, more critical bands are incorporated, up to a maximum number which are useful given the sampling rate of the input signal. This procedure determines the division of bits between scale factors and scalar coefficients as well as the number of critical bands to use.

Since we will be unable to use a portion of the available audio frequencies if we have insufficient bits to do so, we incorporated a low-pass filter so that high frequency components in the highest critical bands do not cause audible artifacts when they are decoded. Filters were generated for each of the possible maximum critical bands from number 15, corresponding to a 4 kHz sampling rate, to number 23, corresponding approximately to a 16 kHz sampling rate.

The scale factor quantization section of the coder was altered so that bits are allocated to the different layers of quantization on a frame by frame basis. This was done by determining which types of quantization were most effective in reducing distortion and weighting the allocation of bits towards those quantization types. Both the non-predictive and predictive schemes use a scalar quantization for average gain, a predictor matrix, and two vector quantizers (one in the case of short windows). Equal numbers of bits are assigned to each of the vector quantizers. In the predictive case, more bits are assigned to predictor matrices, and in the non-predictive case more bits are assigned to the gain quantization.

Other than these modifications, many sections of the coder had to be altered to take into account a changing number of critical bands. For example, after a change in the number of critical bands, the spreading function needs to be recalculated, the low-pass input filter must be changed to the new filter, and we need to make use of the non-predictive scheme for scale factors.

6.2 Future Work

- In the upsampling section, we could limit input sampling rates to integer divisions of 16 kHz. In this case we could use simpler upsampling techniques to upsample to 16 kHz, instead of upsampling to above 32 kHz then downsampling to 16 kHz, saving on some computation. Also, two changes to the filter based on critical bands could be made. First, it could be removed entirely without too great a decrease in quality, though as discussed previously, the savings in computation would not be large. Second, it could be used to replace the interpolating filter during upsampling, if we were limiting the input sampling rates as above.

- The length of the short windows we use was chosen to have a length of approximately 10 ms. It may be, however, that shorter windows, i.e., more short windows per long window, would give better coding of transients, or that fewer short windows per long window would give better coding gain at relatively little loss in transient coding. This is an area in which more research could be done.

- We have tried constructing the psychoacoustic model using MDCT coefficients instead of FFT coefficients and found very little difference in audio quality, with a significant decrease in complexity (approximately 7%). If other areas of the coder were optimized, that decrease in complexity could become even more significant, if further testing showed a negligible difference in quality.

- Due to the problems with the use of an absolute threshold of hearing model where playback level cannot be controlled, in these conditions a replacement model or complete removal of the model may be necessary.

- The first step of quantization for our coder is the division of bits between the scale factors and the spectral coefficients. The ratios we use are optimized to give the same division used by the original coder at 8 kbps operation, so more work needs to be done to make sure bits are allocated efficiently at higher and lower bit rates. If too many bits are allocated to one section or the other, bits can be wasted. Alternatively, an entirely new scheme could be developed for the division of bits, where the number of bits allocated to the two quantization stages is based on the psychoacoustic model.

- One way we could more accurately allocate bits to the scale factors would be to divide the scale factor vector into smaller vectors and allocate bits based on masking threshold. This would cost some efficiency, since smaller vectors give us a less optimal vector quantization, but it would be helpful to break up the vector for performance considerations as well, since up to 23 elements can be in the vector. For higher bit rates, it may even be worthwhile to move to scalar quantization of the scale factors, to increase accuracy at the expense of more bits. If we wanted to increase the scalability of our coder even more, extra scale factors could be introduced so that only relevant portions of critical bands are being sent to the receiver. If these approaches do not prove to be feasible, some improvements to the ratios used for dividing bits between quantization layers would aid in bit allocation performance.

- Concerning the spectral coefficients, if we wished to increase the scalability of our coder, following our discussion for the scale factors, we could divide up the coefficients into smaller groups, corresponding to the number of scale factors we have. This would allow greater flexibility in terms of adding audio bandwidth at an increase in bit rate.

- The codebooks we use for both quantizations are embedded codebooks. This reduces memory needed, but we could also decrease the searching time of our codebooks by structuring them for this purpose. Several methods of structuring a codebook for increased speed of searching can be found in [3], some of which are nearly as optimal as the complete search we do. Also, an improved method of generation for our codebooks would likely help, as we do not create a "starting" codebook before using Lloyd's algorithm to optimize it.

- The coder which we based our coder on, referred to as the original coder, was taken while it was still in progress, and it has continued to be improved over the time our coder was developed. Those improvements, if they are compatible with our bit rate and sampling rate scalability, could be added to our coder as well.

- One scalability feature which we have not implemented, and is becoming common in recent MPEG audio standards, is the ability to send information at a single rate and have several different decoders be able to receive lower sampling rate and lower bit rate versions of that signal. This feature would be especially useful in the case that a single source is *multicasting* to several destinations of differing bandwidth. The SSR profile of MPEG-2 AAC accomplishes this by preceding its MDCT with a filter bank of four filters. Each decoder can choose to decode up to the maximum number of outputs from those filters, depending on its bandwidth. Such a scheme could also be applied to this coder, at the expense of greater complexity.

References

[1] D. O'Shaughnessy, *Speech Communications Human and Machine.* IEEE Press, 2000.

[2] M. Bosi, K. Brandenburg, S. Quackenbush, L. Fielder, K. Akagiri, H. Fuchs, M. Dietz, J. Herre, G. Davidson, and Y. Oikawa, "ISO/IEC MPEG-2 Advanced Audio Coding," *J. Audio Eng. Soc.*, vol. 45, pp. 789–812, 1997.

[3] A. Gersho and R. Gray, *Vector Quantization and Signal Compression.* Kluwer Academic Publishers, 1992.

[4] H. Najafzadeh-Azghandi and P. Kabal, "Narrowband perceptual audio coding." Prepared for submission to *IEEE Trans. on Speech and Audio Processing*, 1999.

[5] H. Malvar, *Signal Processing with Lapped Transforms.* Artech House, 1992.

[6] K. Brandenburg and M. Bosi, "Overview of MPEG audio: Current and future standards for low-bit-rate audio coding," *J. Audio Eng. Soc.*, vol. 45, pp. 4–21, 1997.

[7] J. Tobias, *Foundations of Modern Auditory Theory.* New York and London: Academic Press, 1970.

[8] E. Terhardt, "Calculating virtual pitch," *Hearing Research*, pp. 155–182, 1979.

[9] D. Pan, "A tutorial on MPEG/Audio compression," *IEEE Multimedia*, vol. 2, pp. 60–74, 1995.

[10] "ISO/IEC JTC1/SC29/WG11 14496-3," 1997.

[11] A. Jin, T. Moriya, T. Norimatsu, M. Tsushima, and T. Ishikawa, "Scalable audio coder based on quantizer units of MDCT coefficients," *Proc. IEEE Int. Conf. on Acoustics, Speech, Signal Processing*, vol. 2, pp. 897–900, 1999.

[12] Fraunhofer-Gesellschaft, "MPEG 4 Audio Scalable Profile." `http://www.iis.fhg.de/amm/techinf/mpeg4/scalable.html`, 1999.

[13] H. Najafzadeh-Azghandi and P. Kabal, "Improving perceptual coding of narrowband audio signals at low rates," *Proc. IEEE Int. Conf. on Acoustics, Speech, Signal Processing*, pp. 913–916, 1999.

[14] J. Proakis and D. Manolakis, *Digital Signal Processing Principles, Algorithms, and Applications*. Prentice Hall, 1996.

[15] P. Kabal, "Audio file programs and routines." Can be obtained through `http://www.tsp.ece.mcgill.ca`, October 1996.

[16] J. Johnston, "Transform coding of audio signals using the perceptual noise criteria," *IEEE J. Selected Areas in Comm.*, vol. 6, pp. 314–323, 1988.

[17] E. Zwicker and T. Zwicker, "Audio engineering and psychoacoustics," *J. Audio Eng. Soc.*, vol. 39, pp. 115–126, 1991.

[18] M. Schroeder, B. Atal, and J. Hall, "Optimizing digital speech coders by exploiting masking properties of the human ear," *J. Acoust. Soc. Am.*, vol. 66, pp. 1647–1652, 1979.

[19] J. Johnston, "Estimation of perceptual entropy using noise masking criteria," *Proc. IEEE Int. Conf. on Acoustics, Speech, Signal Processing*, pp. 2524–2527, 1988.

[20] K. Brandenburg, G. Stoll, Y. Dehery, J. Johnston, L. Kerkhof, and E. Schroeder, "The ISO/MPEG audio codec: A generic standard for coding of high quality digital audio," *J. Audio Eng. Soc.*, vol. 42, pp. 780–791, 1994.

[21] G. Soulodre, *Adaptive Methods for Removing Camera Noise from Film Soundtracks*. PhD thesis, McGill University, 1998.

[22] T. Painter and A. Spanias, "A review of algorithms for perceptual coding of digital audio signals," *Int. Conf. on Digital Signal Processing (DSP)*, pp. 1–30, 1997.

[23] D. Pan, "Digital audio compression," *Digital Technical J.*, vol. 5, pp. 1–14, 1993.

[24] M. Dietz, J. Herre, B. Teichmann, and K. Brandenburg, "Bridging the gap: Extending MPEG audio down to 8 kbit/s," *102nd AES Convention*, vol. 42, pp. 780–791, 1997. Preprint 4508.