Embedded lossless audio coding using linear prediction and cascade coding

Kevin Adistambha
University of Wollongong

Embedded Lossless Audio Coding using Linear Prediction and Cascade coding

A thesis submitted in fulfillment of the

requirements for the award of the degree

Master of Engineering Research

From

The University of Wollongong

By

Kevin Adistambha

Bachelor of Science, Master of Internet Technology

School of Electrical, Computer

and Telecommunications Engineering

2005

Abstract

This thesis studies the techniques and feasibility of embedding a perceptual audio coder within a lossless compression scheme. The goal is to provide for two step scalability in the resulting bitstream, where both a perceptual version of the audio signal and a lossless version of the same signal are provided in the one bitstream.

The focus of this thesis is the selection of the perceptual coder to be used as the perceptual base layer and the techniques to be used to compress the lossless layer by using backward linear prediction followed by entropy coding. The perceptual base layer used is MPEG-4 AAC, chosen based on entropy measurements of the residual signal. Results of the work in this thesis show that the embedded lossless coding scheme could achieve an average compression ratio of only 6% larger compared to lossless only coding. Performing decorrelation on the AAC residual signal by means of backward linear predictive coding and measuring the entropy of the resulting LPC residual signal of various orders revealed that an 8% decrease in coding rate is achievable using 15^{th} order prediction.

Furthermore, this thesis also investigates an entropy coding technique known as cascade coding which is originally designed to compress hydroacoustic image data and is modified to compress audio data. Cascade coding is an entropy coding technique that uses multiple cascaded stages where each stage codes a specific range of integers and is used to perform entropy coding of the backward linear prediction residual signal. The cascade coding technique explored in this thesis includes using a frame based approach and trained codebooks.

Statement of Originality

This is to certify that the work described in this thesis is entirely my own, except where due reference is made in the text.

No work in this thesis has been submitted for a degree to any other university or institution.

Signed

Kevin Adistambha

6 December 2005

Acknowledgements

I would like to thank my supervisors Dr. Christian Ritz, Dr. Jason Lukasiak and Assoc. Prof. Ian Burnett. Their support and patience made this work possible. My never ending questions and problems don't seem to deter them to help me finish this thesis.

I would also like to thank my family and friends for their morale support and providing relief when the stress or boredom (or both) is too much for me to handle. Especially my parents, whose patience seems impossible to exhaust.

Last but not least, thanks to Ibanez guitars, Marshall amplification and Line 6 for providing me with much needed distractions.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AAC	Advanced Audio Coding
AAZ	Advanced Audio Zip
BPGC	Bit Plane Golomb Code
BSAC	Bit Sliced Arithmetic Coding
EBCOT	Embedded Block Coding with Optimize Truncation
EZW	Embedded Zerotree Wavelet
IID	Independent and Identically Distributed
JPEG	Joint Photographic Experts Group
LPC	Linear Predictive Coding
LSB	Least Significant Bit
LTP	Long Term Prediction
MAC	Monkey’s Audio Coder
MDCT	Modified Discrete Cosine Transform
MGE	Multigrid Embedding
MPEG	Motion Picture Experts Group
MSB	Most Significant Bit
PCM	Pulse Code Modulation

PDF	Probability Density Function
PNS	Perceptual Noise Substitution
SFM	Spectral Flatness Measure
SMR	Signal to Mask Ratio
SNR	Signal to Noise Ratio
SPIHT	Set Partitioning in Hierarchical Tree
TWINVQ	Transform-domain Weighted Interleave Vector Quantization

1 Introduction

The popularity of digital audio has increased significantly since the introduction of the compact disc. The ease of use and sound clarity afforded by the compact disc surpasses that of traditional analog audio technology [1].

Further advances in digital audio technology have led to the development of *perceptual coding* or *lossy coding*, with the implementation in the MPEG-1 Layer 3 standard (mp3) being highly successful in today's consumer market. Perceptual coding compresses a digital audio signal significantly by removing data inaudible to the human ear, hence the name *lossy*. The current state of the art in perceptual audio coding is MPEG-4 Advanced Audio Coding (AAC) [2], which offers many improvements over mp3, including smaller file size and higher quality audio.

Another branch of study in digital audio compression is *lossless coding*. Lossless coding aims to digitize an audio signal by exploiting redundancy, thus not losing any data in the process of encoding and decoding. Typical uses for lossless coding are for mastering and archival purposes, which cannot tolerate any reduction in audio quality. The disadvantage of lossless coding is a much bigger file size compared to perceptual coding.

Scalable lossless coding and embedded lossless coding are two paradigms in between the two extremes above, with two different approaches to combine lossy and lossless coding. Scalable lossless coding typically uses a new coder architecture to achieve fine granular scalability to bridge lossy and lossless coding. Embedded lossless coding, on the other hand, embeds a lossy signal inside a lossless signal by layering a lossless enhancement layer on top of the lossy layer. In this thesis, embedded lossless audio coding will be investigated.

1.1 Digital representation of audio data

There are two main steps required for the digitization of audio data, s*ampling* and *quantization*. Sampling refers to taking measurement of an audio signal in discrete time intervals. The sampling rate is a measurement that denotes how many times a second a given signal is sampled. The sampling theory widely used today was proposed by Harry Nyquist in 1928 [3]. It states that in order to accurately transform a sampled signal back into a continuous signal, the sampling rate has to be at least twice the bandwidth that can possibly occur in the input signal. For example, CD audio with a bandwidth of 22.05 kHz has to be sampled with a sampling rate of at least 44.1 kHz to avoid *aliasing*, or one representative for two different signals. This frequency is called the *Nyquist* or sampling *frequency*.

Quantization refers to approximating a continuous signal value using a fixed length representation using a binary number in the case of digital signal quantization [4]. The quantization resolution, or quantization level, is measured in *bits/sample*. For CD audio, samples are quantized using bits, hence the resulting data has a resolution of 16 bits/sample, which means that there are 65536 possible values that can be chosen for a given sample.

Both of these concepts make digital audio possible. By sampling and quantizing an analog audio signal, it is possible to reconstruct an approximated original analog signal by reversing the sampling and quantization process, as long as guidelines such as the Nyquist theorem are observed. However, it is not possible to perfectly reconstruct the original analog signal due to the loss of information during the sampling and quantization process.

1.2 Perceptual coding

While the human ear can generally perceive frequencies ranging from 20 Hz to 20 kHz [4], not all frequencies are perceived equally. Significant research has been performed in the field of *psychoacoustics*, or the research into human auditory perception. This research field forms the basis of digital coding of audio signals using a psychoacoustic phenomena called *masking*. This type of coder is known as *perceptual coder*.

Perceptual coding exploits the unequal human perception of auditory signals in various frequencies and the fact that frequencies can mask one another. Compression is achieved by removing parts of the audio frequencies not perceived by the human ear. [4]. Using psychoacoustic measurements in a coding environment has been highly successful, with compression ratios of 1:12 not uncommon, while maintaining near-transparent audio quality for stereophonic signals [2]. Near-transparent means that the resulting compressed signal has inaudible or barely audible distortion from the coding process, hence the perceptual quality of the compressed signal remains relatively identical to the original signal.

However, the disadvantage of perceptual coding is the removal of perceptually irrelevant data. While there are obvious benefits in reducing file size, for example, easier and faster transmission of audio data and reduced requirements for storage capacity, the data removed cannot be recovered by any means. This makes perceptual coding unsuitable for applications requiring bit-by-bit lossless quality of an audio signal. The term bit-by-bit lossless means that the signal is compressed faithfully in the sense that every bit of data is preserved and can be recovered exactly.

1.3 Lossless coding

The lossless coding paradigm compresses a signal by retaining all the information in the original signal without any loss. It achieves compression by removing redundancy and correlation from the signal and stores the signal in a more efficient manner so that the original signal can be recovered in the decoding process.

The main disadvantage of lossless audio compression is the resulting compressed file size. The best compression ratio achievable using lossless coders is currently around 1:2, a significant difference in compression performance compared to perceptual coding. However, for applications requiring strict data integrity, lossless coding is the only compression method acceptable. Lossless coding of audio data will be discussed in more detail in Chapter 2.

1.3.1 Scalable lossless coding

Between the two extremes of perceptual and lossless coding detailed in sections 1.2 and 1.3 respectively, significant research has been performed in developing a scalable

coder that merges the two conflicting paradigms. There are two approaches to achieving scalability in a lossless coder with different grains of scale: by using a single coder that provides perceptual or signal-to-noise ratio scalability up to a pure lossless signal, and by using an existing perceptual audio coder to provide the perceptual layer and using another coder to compress the difference between the perceptual coder and the original signal. The first method is called *scalable to lossless coding*, which could provide fine granular scalability that can provide many bitrates adjustable by 1 bit/second (called *fine granular scalability*). The second approach is called *embedded lossless coding*.

1.3.2 Approaches to scalable to lossless audio coding

Fine granular scalability is achieved by typically using an entirely new coder architecture based on either an image coding method [5], integer transform [6], or modifying an existing architecture to achieve scalability such as used in MPEG-4 Scalable Lossless Coding [7, 8].

Reference [8] implements an embedded paradigm with a lossy base layer based on TwinVQ and a scalable lossless enhancement layer encoded with the *Bit Sliced Arithmetic Coding* (BSAC) entropy coding technique, which is also used in MPEG-4 scalable audio coding standard [2].

Reference [6] expands the idea further by using a coder inspired by the MPEG-4 AAC encoder by replacing the *Modified Discrete Cosine Transform* (MDCT) unit with an integer approximation of the MDCT, called the *Integer MDCT* (IntMDCT) [9]. The resulting lossless IntMDCT coefficients are then sorted according to their perceptual relevance and coded using a BSAC method, hence achieving fine grain scalability. In order to reconstruct the resulting bitstream according to perceptual significance, the decoder requires side information in the form of the original signal's masking curve.

Fine granularity is desired in order for one to customize an audio stream according to specific bandwidth requirements. If a perceptual signal is needed, then decoding a low bitrate stream will provide the signal required. Whereas, if a lossless signal is needed, then decoding all the available bits will yield a lossless signal. Generally, a scalable coder can achieve qualities ranging from a low quality signal comparable to voice coding or even less for previewing purposes or transmission in a low bandwidth

environment to a full bit-by-bit lossless version for archival purposes, adjustable up to 1 bit/second increase/decrease in bitrate [10].

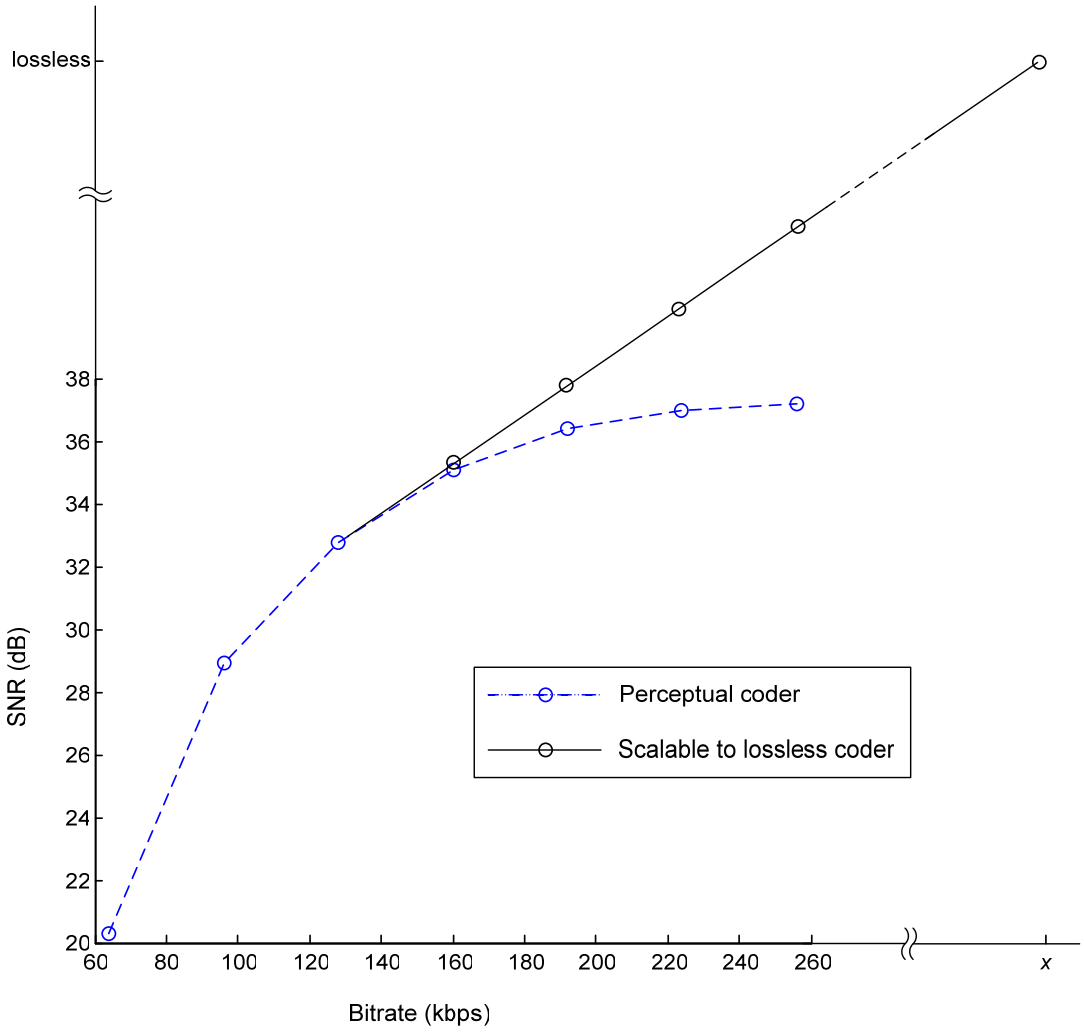


Figure 1.1. Example of perceptual coding vs. objective coding SNR function.

A graphical example of scalable to lossless scalability in terms of signal-to-noise ratio which is the ratio of the power of the original signal and the power of the residual signal is shown in Figure 1.1. Note that the objective scalable to lossless coder mentioned in Figure 1.1 is a bit-by-bit scalable coder. In Figure 1.1, it is shown that the SNR function of a perceptually coded signal has a knee point at around 128 kbps (which is consistent with the SNR function of an AAC coder, further detailed in Chapter 3) and the SNR function of an objectively scalable coded signal increases until lossless representation of the original signal is achieved at *x* kbps, where the scalable to lossless coder is designed with the objective that the bitrate *x* where lossless compression is achieved, the bitrate of the coder is less than PCM bitrate. The lossless coder SNR curve

shown in Figure 1.1 is an extrapolation from the perceptual coder knee point at 128 kbps, thus Figure 1.1 serves to illustrate the goal of a scalable to lossless coder. A scalable to lossless coder could be designed so that for rates less than 128 kbps, a perceptual coder is utilized, while for higher bitrates an objectively scalable coder is utilized hence the encoded signal displays an increasing SNR until lossless compression is achieved.

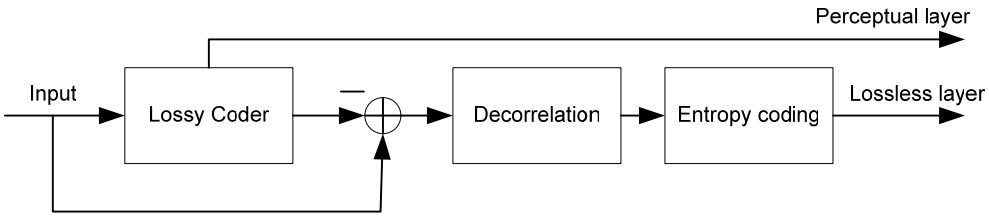


Figure 1.2. Block diagram of an embedded lossless coder.

The embedded coding methodology described in this thesis embeds a perceptually transparent quality bitstream layer inside a lossless quality layer in order to facilitate both perceptual and lossless paradigms by combining both technologies together by using a separate coder for each layer. The approach used in this work is to use an embedded lossless coding method that provides two step scalability, the first step being the perceptual coding layer using a standard perceptual audio coder, and the second step being the lossless enhancement layer, hence providing a lossless bitstream by combining the two steps. Thus this work is similar to [7, 8]. The block diagram for the coder examined in this work is depicted in Figure 1.2.

1.4 Motivation

Currently, scalable lossless coding has received much recent attention. This thesis aims to examine the feasibility of embedded lossless coding using currently available techniques in lossy and lossless audio coding. This technique of embedded coding is desired to offer a lossless coder using a perceptual coder already widely available, thus retaining compatibility with existing high quality perceptual audio coders. The perceptual layer of the resulting bitstream can then be decoded independently, yielding a perceptually coded version of the audio signal which has the advantage of a much lower transmission bandwidth requirement compared to the full losslessly coded signal. If a lossless signal is required, both layers can be decoded.

1.5 Thesis outline

Chapter 2 will provide a literature review of techniques concerning perceptual coding and lossless coding of audio data. Chapter 3 will provide an examination of using MPEG-4 AAC lossy coder to act as the base layer in an embedded coder. Chapter 4 will examine linear prediction as a decorrelation method of an AAC residual signal and will examine entropy coding techniques for storing and transmitting the resulting LPC residual signal. Chapter 5 will provide conclusions and future work.

1.6 Contributions of this Thesis

Contributions of this thesis include the following:

1. A study of a perceptual coder residual signal and the optimum base layer bitrate for an embedded lossless coder as shown in the block labeled "Lossy coder" in Figure 1.2. The analysis performed in Chapter 3 shows that the optimal perceptual coder bitrate for an embedded lossless coding application is 96 kbps mono using the MPEG-4 AAC perceptual coder, where the overall embedded lossless coding bitrate results in only 6% increase over the bitrate used by a state of the art lossless only coder.

2. A study of backward LPC processing of an AAC residual signal using a large database of audio data to identify an appropriate decorrelation method for the block labeled "Decorrelation" in Figure 1.2. Chapter 4 shows that using 15^{th} order backward adaptive linear prediction results in an overall 8% decrease in the coding rate of the AAC residual signal. Orders beyond 15 do not increase the coding rate significantly.

3. An investigation into the cascade coding technique [11], originally designed to compress hydroacoustic image data, for audio compression purposes. The cascade coding technique is compared to Rice coding, an existing popular approach to perform entropy coding of audio data.

4. New design of a cascade coder by training the cascade coder for use with audio signals. Modifications of the cascade coder paradigm to employ a frame adaptive, codebook based method.

5. Part of the work performed in this thesis has lead to the publications [12] and [13]

2 Entropy and decorrelation of audio signals

2.1 Introduction

This chapter provides a literature review of methods concerning the amount of information in a given signal (the entropy of a signal) and discusses how the entropy of a signal is related to its lossless compression rate using various compression techniques. This chapter also discusses ways to reduce the entropy of a signal by removing the correlation between samples in a signal, hence lowering its entropy. This process of removing correlation also implies the need to recover the correlation in order to reconstruct the compressed signal without loss of information and will be discussed in this chapter.

2.2 Signal entropy and coding theory

2.2.1 Entropy theory

Information entropy is a term used to describe the amount of information in a signal. The lower the entropy value, the more predictable a signal. From a coding perspective, the lower the entropy, the smaller the resulting binary representation will be since there is no need to send a predictable stream of data.

Claude Shannon formulated the theorem for entropy of a signal encoded into binary format, using bits/sample as a measurement method [14].

$$H = -\sum_{x} p_x \log_2 p_x \tag{2.1}$$

In (2.1), H is the entropy of the signal in bits/sample and p is the probability of a symbol occurring in a signal. H denotes the theoretical minimum bits/sample required to code a given data stream into binary [4]. It is not possible to code a given data stream into binary at any bitrate lower than H.

A property every entropy coder has to have is *unique decipherability*, that is, every symbol that is translated into binary has to be able to be translated back to its original symbol. Any entropy coder that can translate a stream of symbols into its binary representation is not very useful without the ability to decipher the binary stream back to its original symbols.

A common way of achieving unique decipherability is by using *prefix codes* or *prefix-free codes* [14]. For these codes, no code is a prefix of another code, which will create confusion in the decoding process. For example, if a codebook contains [A='1' and B='11'], and if a given binary stream is '1111', does the original symbol consists of four As or two Bs? Without the prefix codes property, the decoder cannot know for certain.

While it is possible to code a given stream of symbols at its *H* value, the decoder needs to know what symbol each binary value corresponds to. Hence, coding at the *H* value requires a codebook of the alphabet sent to the receiver, which in turn increases the data transimitted. Also, the unique decipherability of a binary stream has to be taken into account along with the codebook, otherwise a decoder can easily make errors in decoding.

Since a practical binary coder has to take into account many variables such as *unique decipherability* of a binary stream and implementation details such as the codebook required to decode the signal, oftentimes it is not possible or practical to code a signal into binary at a rate given by *H* [15].

2.2.2 Entropy coding methods

There are two general categories for entropy coding: *lossless source coding* which requires transmission of the signal statistics for maximum efficiency and *universal lossless source coding* which does not require prior knowledge of signal statistics to allow decoding [15]. Universal lossless source coders are designed with an assumption of the signal statistics in mind or by employing a dictionary based approach. Examples of universal lossless coders include *Rice coding* for coding signals with a Laplacian distribution and *cascade coding* for signals with a *stepwise distribution*. Rice coding and cascade coding techniques will be described in more detail in sections 2.2.2.4 and 2.2.2.5 respectively. Generally, lossless source coding is more flexible since it does not

make any assumption on the signal. Examples of lossless source coders include *Huffman coding* and *arithmetic coding* which will be described in sections 2.2.2.1 and 2.2.2.3. However, it is a requirement to send a codebook describing the signal statistics to the receiver as overhead which is a disadvantage of this approach.

2.2.2.1 Huffman coding

Introduced by D. A. Huffman in 1952 [15], Huffman coding is an efficient entropy coding method based on prior knowledge of the statistics of the data, therefore it can be categorized as a lossless source coder. Huffman codes have the properties of assigning a short code to the most occurring symbol and a longer code to the least occuring symbol, also it is a *prefix code* as described in Section 2.2.1. As it was the first successful entropy coding method, any coder that has both properties is generalized as a Huffman coder, although the algorithm for code generation could be radically different than Huffman's. Huffman codes are created by forming a tree of symbol probabilities, such that the most probable symbol is assigned the shortest code and the least probable symbol is assigned the longest code. An example of a Huffman code is depicted in Figure 2.1.

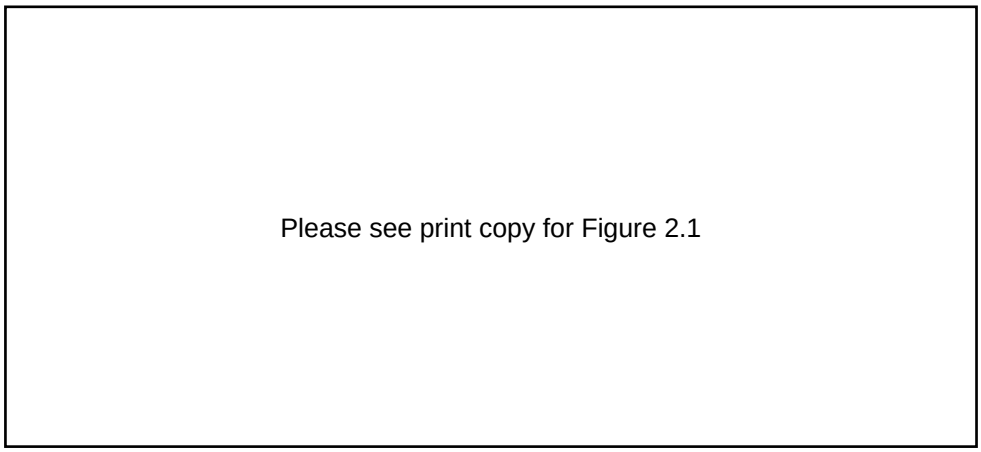


Figure 2.1. Example of Huffman coding

In Figure 2.1, a Huffman tree is created for source signal with an alphabet of size 5 (A, B, C, D, E) with symbol probabilities of 0.4, 0.3, 0.1, 0.1 and 0.1 respectively. The Huffman algorithm creates the tree by adding two of the lowest probabilities, and the sum of the two probabilities is then counted as one in the next additive process. Thus for the above example, nodes D and E combine their probabilities in the first step resulting in a joint probability of 0.2. The next two lowest probabilities are then 0.2 and 0.1. The

process continues until all symbols are accounted for and the sum of all probabilities is 1. Next, the upper and lower branches of the tree are labeled with 0 and 1. The resulting Huffman code can then be read from the tree by following the branches from the root.

With the most probable symbol assigned the shortest code, and decreasing probability symbols are assigned an increasing length of binary representation of the symbol. The average codeword length for a signal with alphabet and probabilities in the example is then: (0.4 × 1) + (0.3 × 2) + (0.1 × 3) + (0.1 × 4) + (0.1 × 4) = 2.1 bits. The entropy for such a signal is ≈2.05. It is shown that for this example, the Huffman code could encode a sequence at a rate within 0.05 bits/sample to its entropy.

However due to the fact that Huffman coding is a lossless source coding technique, to decode the Huffman encoded sequence, the decoder requires the Huffman codebook that describes the codewords and its corresponding symbols. Therefore, although the resulting encoded signal is efficient, the need to send the codebook to the receiver has to be taken into account. This codebook will grow in size as the size of the alphabet increases, hence Huffman coding's efficiency is negated when the alphabet size is large, unless the codebook is stored in both the encoder and the decoder for coding a fixed signal statistics which made both the encoder and decoder unsuitable to code signals with different statistics.

Another disadvantage of Huffman coding is that since Huffman coding is designed to encode a signal into its binary representation, the resulting Huffman code for each symbol is required to have an integer number of bits (since a non-integer number of bits representation is not possible). For example, equation (2.1) depicts the average number of bits/sample required to code a sequence of symbols with their respective probabilities. By removing the averaging operation from (2.1), the optimal number of bits required to code a symbol is then shown in equation (2.2).

$$h_x = -\log_2(p_x) \tag{2.2}$$

Where h_x denotes the optimal number of bits required to code a given symbol with probability p_x.

Therefore, if the probability of a symbol is 0.3, then the optimal number of bits required to code the symbol is $-\log_2(0.3) = 1.7370$ bit. Huffman coding requires the symbol in question to be coded using 2 bits. The problem becomes more apparent for symbols having high probability. For example if one symbol probability is 0.95, the symbol theoretically requires $-\log_2(0.95) = 0.0740$ bit. Huffman coding requires the symbol to be coded using 1 bit, resulting in a binary stream which is ≈14 times larger than theoretically necessary. Arithmetic coding, which is detailed in section 2.2.2.3 was designed without this limitation.

Some techniques such as MPEG-4 AAC employ Huffman coding, but work around the Huffman disadvantage of requiring a codebook by predetermining Huffman codebooks which are designed based on various collected signal statistics [2]. The disadvantage of this approach is that the Huffman codebook might not be optimal for the input signal.

2.2.2.2 Lempel-Ziv

Introduced by Abraham Lempel and Jacob Ziv in 1977 (LZ77), the Lempel-Ziv algorithm does not require prior analysis of the signal probability statistics and thus can be categorized as a *universal lossless coding scheme* [15]. The algorithm was refined in 1978 (LZ78) and improved upon by Welch in 1984 (LZW). The original LZ77 remains the most popular, implemented in gzip [16] and winzip [17]. For the purpose of this literature review, the operating principles of LZ77 is explained in detail below.

Position	Current symbol	Matches	Output
1	A	-	0,0,A
2	B	-	0,0,B
5	C	AB	2,2,C
8	A	BC	2,2,A
11	<EOF>	BC	3,2,<EOF>

Table 2.1. Example of LZ77 encoding.

LZ77 compressed output consists of three symbols: *pointer to previous instance* which is a number that denotes how far back does the current symbol already occurred in the signal, *length to copy* which is the length of the match in the previous instance and *current symbol* which is the current symbol. The efficiency of LZ77 depends upon

using the *previous instance* and *length to copy* as long a section as possible. For example, encoding the message 'ABABCBCABC' using LZ77. The encoding process is as shown in Table 2.1.

For decoding, LZ77 simply reconstructs the data using the pointers and length to copy. There is no statistical analysis involved, but since the compression performance of LZ77 depends upon the previously occurring data, increasing the window length of data to be searched by the algorithm can increase the likeliness of a match occurring in a longer stream of symbols, hence increasing the compression performance. Of course, increasing the window size has the disadvantage of increasing the complexity of the search algorithm.

Although Lempel-Ziv algorithm is widely used for compressing textual data which is what it was designed for, it has not been able to achieve the same level of performance for compressing audio data [18]. Hence no audio coder used Lempel-Ziv algorithm for entropy coding.

2.2.2.3 Arithmetic coding

Arithmetic coding, like Huffman coding, is a high efficiency entropy coder requiring prior knowledge of signal statistics [15]. The principle of operation is one of continually dividing an interval between 0 and 1 by symbol probabilities until there exists exactly one number that uniquely identifies the signal.

Symbol	Probability	Interval
A	0.3	$0 \leq i_A < 0.3$
B	0.4	$0.3 \leq i_B < 0.7$
C	0.3	$0.7 \leq i_C < 1$

Table 2.2. Example of symbol probabilities and interval values for arithmetic coding.

For example, encoding the symbol stream 'ABABCBCABC'. Establishing the symbol probabilities and dividing the probabilities with the interval $0 \leq i < 1$ results in Table 2.2, which shows an arithmetic coding codebook with i_A, i_B and i_C denotes the intervals for symbols A, B and C respectively.

Symbol	Symbol interval	Output interval
A	$0 \leq i_A < 0.3$	$0.0 \leq i_O < 0.3$
B	$0.3 \leq i_B < 0.7$	$0.09 \leq i_O < 0.21$
A	$0 \leq i_A < 0.3$	$0.090 \leq i_O < 0.126$
B	$0.3 \leq i_B < 0.7$	$0.1008 \leq i_O < 0.1152$
C	$0.7 \leq i_C < 1$	$0.11088 \leq i_O < 0.11520$
B	$0.3 \leq i_B < 0.7$	$0.112176 \leq i_O < 0.113904$
C	$0.7 \leq i_C < 1$	$0.1133856 \leq i_O < 0.1139040$
A	$0 \leq i_A < 0.3$	$0.11338560 \leq i_O < 0.11354112$
B	$0.3 \leq i_B < 0.7$	$0.113432256 \leq i_O < 0.113494464$
C	$0.7 \leq i_C < 1$	$0.1134758016 \leq i_O < 0.1134944640$

Table 2.3. Example of arithmetic coding.

By recursively limiting the interval of successively occuring symbols in the input signal, the arithmetic coding steps are then depicted in Table 2.3.

The lower bound of the final output interval (0.1134758016 in the example) uniquely identifies the message ‘ABABCBCABC’. The decoding process works in reverse, by recursively decoding the symbols based on the number transmitted by the encoder starting from the most significant digit first. It is obvious from the example in Table 2.3 that arithmetic coding aims to code *sequences of symbols* together and assign an arithmetic code to the whole sequence, instead of assigning a predetermined code per symbol as in the case of Huffman coding.

To perform arithmetic coding with binary codes, the fractional interval limits of arithmetic coding are replaced with binary sequences. For example, given an interval of $0 \leq x < 1$, the binary representation assuming 16 bits are used is $0000 \leq x <$ FFFF. Division of intervals as in the example in Table 2.3 are performed using exactly the same procedure.

Arithmetic coding can achieve a rate closer to the entropy of the input signal than Huffman coding by grouping symbols together, for example one arithmetic code can represent a sequence of two symbols occurring in sequence, instead of only one arithmetic code per symbol. The advantage of arithmetic coding compared to Huffman coding is that there is no need to build the complete codebook [14].

As in the Huffman coding example of 2.2.2.1, the disadvantage of arithmetic coding is the requirement to send the codebook to the receiver, thus making it unsuitable for large alphabet inputs, such as audio signals.

2.2.2.4 Rice coding

Introduced by Robert F. Rice for NASA, Rice coding is a family of entropy coders designed for transmitting data in a space environment [19];. it is further detailed in [20] and [18], Rice coding performs compression close to the entropy of a signal with a *probability density function* (p.d.f.) with zero mean and descending probabilities of larger magnitude values, comparable to the shape of a Laplacian distribution. Rice coding does not require prior knowledge of signal statistics, although the assumption of a Laplacian distribution of the signal must hold for optimal performance

Rice coding consists of a two part symbol, with the first symbol comprising of a *unary code* and with the 2nd symbol being a simple binary code separated by a binary 1. A Unary code, as opposed to a binary code, employs only one symbol, for example 0. Thus, a unary code for the number 4 will be '0000'

The efficiency of Rice coding depends upon the length of each symbol. To determine the length of each, a *k-value* is first calculated. The *k-value* is

$$k = \log_2(\bar{s}) \tag{2.3}$$

Where in (2.3), $\bar{s}$ is the mean of the absolute value of the signal.

Using the *k*-value, approximately 50% of the values are coded as a straight binary form without any unary codes, given the distribution is Laplacian [20]. Any value larger than *k* is then partly coded using binary and partly coded using unary. While this may create a long code due to the unary part, the probability of a long code is small due to the assumption of a Laplacian distribution for the symbol values.

The three parts comprising a Rice code with input I_i are shown in Equations (2.4), (2.5) and (2.6).

$$S_i = \begin{cases} 0\ , & I_i \geq 0 \\ 1\ , & I_i < 0 \end{cases} \tag{2.4}$$

$$B_i = binary[\ \mathrm{mod}(\ |I_i|\ ,\ 2^k)\ ] \tag{2.5}$$

$$U_i = unary\left\lfloor \frac{|I_i|}{2^k} \right\rfloor \tag{2.6}$$

Where S_i, B_i and U_i are the sign bit, binary symbol and unary symbol respectively, $\mathrm{mod}(m,n) = m - n\left\lfloor \frac{m}{n} \right\rfloor$, $|x|$ denotes the absolute value of x, and $\lfloor x \rfloor$ is the largest integer where $\lfloor x \rfloor \leq x$. The function *binary* in (2.5) performs conversion from integer to binary using exactly k bits, and the function *unary* in (2.6) converts an integer value into its unary value.

The output code is [S_i U_i '1' B_i] for the i[th] input, with the length of B_i of exactly k bits. In other words, equation (2.5) takes the k least significant bits of the binary representation of I_i and equation (2.6) converts the most significant bits of I_i not encoded in B_i into a unary representation.

For example, given the input of -17 and k=4:

1. Using (2.4): $-17 < 0,\ \therefore S_i = 1$

2. Using (2.5): $B_i = binary[\ \mathrm{mod}(|-17|, 2^4)\ ]$
   $B_i = binary[1]$
   $B_i = 0001$

3. Using (2.6): $U_i = unary\left\lfloor \frac{|-17|}{2^4} \right\rfloor$
   $U_i = unary\lfloor 1.0625 \rfloor$
   $U_i = 0$

4. Combining 1,2 and 3, the resulting code is '1 0 1 0001'.

In some implementations, the output code is reversed, e.g. [S_i B_i U_i '1'], with the separator bit '1' acting as a flag that signifies the end of a symbol. The resulting code length, however, remains constant. In a computing implementation, Rice coding is very efficient operation-wise due to the fact that (2.5) can be implemented as a *bit mask* with *k* ones, and (2.6) can be implemented as *bit right shift* and conversion to unary. The relatively expensive parts of Rice coding are the calculation of the *k* value and the unary conversion.

Rice coding is widely used in lossless audio coding applications which employ predictive coding, such as [20], [21] and [22]. It is also used in the recent MPEG-4 specification for Audio Lossless Coding standard (MPEG-4 ALS) [23]. Its advantage is its efficiency in coding a Laplacian like distribution of values, which is generally the distribution of a prediction residual. This fact is observed in [20].

2.2.2.5 Cascade coding

Cascade coding was first proposed by Lixue Wu et al in [11] as a method of lossless coding of hydroacoustic data. It is further generalized in [24] as a method for coding large dynamic range integer sequences. Cascade coding is designed to perform most efficient on a zero mean integer sequence with steeply descending order of probability that is defined as a *stepwise distribution* [24]. An advantage of cascade coding is low complexity consisting of mainly addition and subtractions [24].

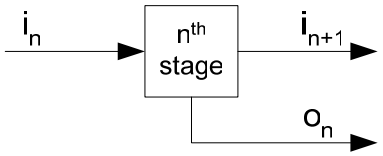


Figure 2.2. The nth stage of a cascade coder.

The operation of a cascade coder at stage *n* is shown in Figure 2.2, where:

$$i_{n+1} = i_n - \left(2^{b_n} - 1\right) \tag{2.7}$$

$$o_n = \begin{cases} 0, & i_n > 2^{b_n} - 1 \\ i_n, & i_n \le 2^{b_n} - 1 \end{cases} \tag{2.8}$$

With *i* as the input and *o* as the output of the nth stage.

A cascade coder consists of a series of stages with a specific bit allocation for each of the stages. If the input integer overflows a given stage, the output of the current stage is set to zero and the residual of the integer is then cascaded into the next stage until the whole integer is coded. A stage is allocated *n* bits, and each stage can code a maximum integer magnitude of 2^n-1. An output of all zeros for a stage denotes that the input overflows the current stage ((2.7) and $o_n = 0$ in (2.8)). The allocation of the first stage in a cascade coder includes one bit reserved for the sign bit. In other words, the input integer is successively subtracted by the largest magnitude of integer that can be coded in each of the stages until the whole integer are coded (until $o_n = i_n$ in (2.8)).

Decoding works in reverse. If a stage output is all zeros, then the maximum value of 2^n-1 of the current stage is added to the output of the next stage until all the stages are accounted for (until $o_n \neq 0$ in (2.8)). If the output of all stages are zeros, the input integer is the cumulative sum of the maximum magnitude of all the stages.

The efficiency of cascade coding depends upon the fact that it is designed to operate on a zero mean, steeply descending distribution of integer values such that the first stage only is used more often than multiple stages.

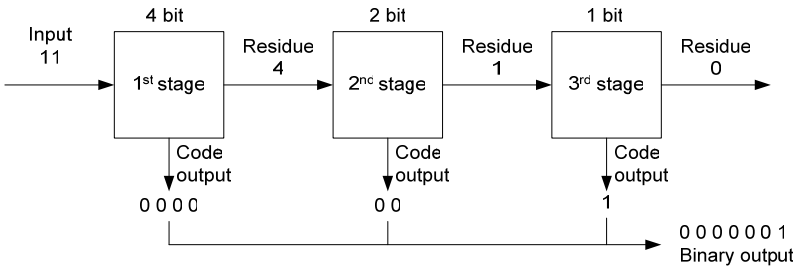


Figure 2.3. Example of a cascade coder.

Figure 2.3 shows an example of cascade coding with three stages using a bit allocation of [4 2 1] and input integer 11. The output of the first and second stage are zeros which includes the sign bit which is also zero, and the output of the third stage is 1. The resulting bitstream is ‘0000 00 1’.

2.2.3 Entropy coding techniques comparison

This section has discussed entropy coding techniques of Huffman coding, Lempel-Ziv coding, arithmetic coding, Rice coding and cascade coding. Huffman and arithmetic

are categorized as universal lossless coding schemes that require prior knowledge of signal statistics, whereas Rice and cascade are lossless coding schemes that were designed for a specific signal statistics. Lempel-Ziv can be categorized as a universal lossless source coding requiring no prior knowledge of signal statistics, although it is designed for signals with repetitive symbols in mind, such as textual data. While universal lossless coding schemes are more robust in handling any type of signal statistics, if assumptions can be made on the input signal statistics then the use of Rice and cascade coding is justified, with the added advantage of not requiring a dictionary to be sent to the receiver.

While Rice coding has been popular with many lossless audio coders such as [20-23], the low complexity and implementation simplicity promised by cascade coding has not been exploited thoroughly in literature.

2.3 Scalable entropy coding techniques

Scalable lossless coding is a method of coding a stream of data that is able to scale from a rough approximation of the original signal up to a lossless version. As described in Chapter 1, there are two coding methods associated with scalable coding: *fine grain scalability* having the characteristic of small step scalability down to 1 bit/second and *embedded coding* having the characteristic of large step scalability with two versions of the decoded signal: perceptual and lossless.

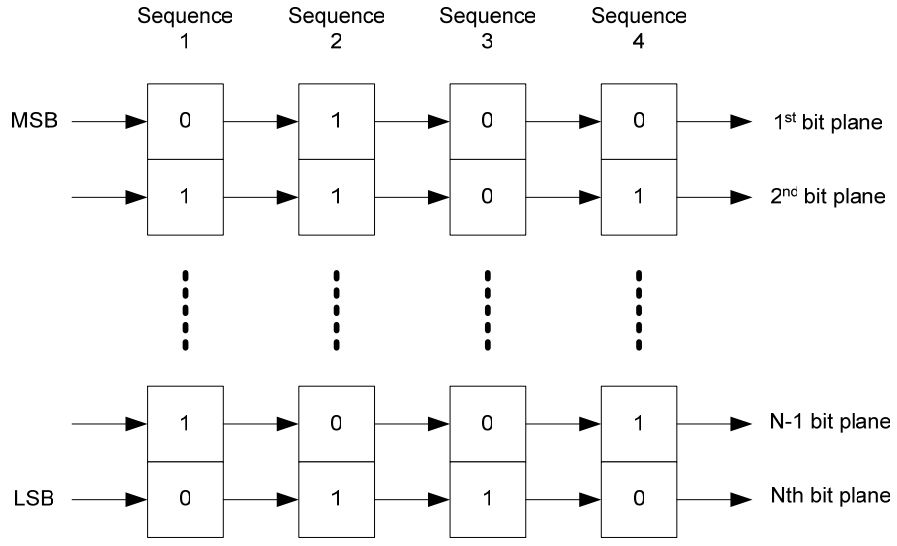


Figure 2.4. General operation of the bit plane coding scheme.

2.3.1 Fine grain scalability via bit plane coding

In general, fine grain scalability is achieved via a *bit plane coding* or *bit slice* scheme, and is depicted in Figure 2.4 and (2.9).

$$c(k) = s\sum_{i=1}^{M} b(i,k)\,2^{i} \tag{2.9}$$

In (2.9), *c(k)* is the k^{th} symbol, *s* is the sign of the symbol with $s \in \{-1,1\}$, *b(i,k)* is the bit plane symbol with $b(i,k) \in \{0,1\}$, and *M* is the total number of bit planes with *i* denoting the bit plane order from the *most significant bit* (MSB) to the *least significant bit* (LSB). Hence, by transmitting the symbol sequence bit plane by bit plane, reconstruction in the receiver can progress from the MSB to the LSB of each symbol with the effect of the receiver receiving a rough approximation of the signal by decoding the first few bit planes up to the completely lossless signal when all bit planes are decoded.

Bit plane coding schemes are popular with fine grain scalability lossless audio coding methods, such as *Bit Plane Golomb Code* (BPGC) which is used in the AAZ scheme and is being standardized as the MPEG-4 Scalable to lossless (MPEG-4 SLS) standard [25] and *Bit Sliced Arithmetic Coding* (BSAC) which is used in MPEG-4 AAC as a scalable entropy coding method [2]. Both techniques employ bit plane coding to replace the Huffman based entropy coding method.

2.3.1.1 BSAC and BPGC

In audio coding, BSAC and BPGC methods exhibit some similarities: both use arithmetic coding in a bit plane coding environment. The BPGC technique further expands the idea by restricting the input signal statistics having a Laplacian distribution and employing some statistical model describing each bit plane [26].

The BSAC method is utilized in MPEG-4 as a method of entropy coding to encode the MDCT coefficients of the MPEG-4 AAC encoder with the property of scalability of up to 1 kbps/channel [27]. In order to achieve scalability, BSAC adopts the approach of using many enhancement layers, with each enhancement layer adding a 1 kbps/channel to the resulting bitstream. Each enhancement layer contains one bit plane or *bit slice* of the arithmetic code sequences, with the bit planes arranged from the MSB to the LSB of

the arithmetic code sequence. Hence, the first bit plane (enhancement layer) contains only some of the most significant bits of all arithmetic code sequences. With more enhancement layers decoded by the receiver, the decoded signal then receives finer detail in the form of more of the least significant bits from subsequent enhancement layers.

Both methods have been examined for use in the MPEG-4 AAC based scalable to lossless coder in [6, 25, 27], primarily by utilizing an MPEG-4 AAC compatible bitstream as the base layer and employing BSAC or BPGC scalable entropy coding methods to achieve fine grain scalability in the lossless enhancement layer. The base layer is constructed by approximating the output of the MDCT unit of the MPEG-4 AAC encoder, thus a modification of the AAC encoder is required.

2.3.1.2 Other coding schemes

While audio based encoding primarily utilizes a *Modified Discrete Cosine Transform* (MDCT) which will be explained in detail in Chapter 3, wavelet transform based signal decomposition has been used extensively in scalable image coding techniques including *Embedded Zerotree Wavelet* (EZW) [28], *Set Partitioning in Hierarchical Trees* (SPIHT) [29], *Multigrid Embedding* (MGE) [30] and *Embedded Block Coding with Optimized Truncation* (EBCOT) [31] which is the basis for the JPEG2000 standard. Recently, however, the EZW, the SPIHT and the EBCOT methods have been explored for use in audio environment in [32], [5] and [33]. Fine grain scalability, however, is not the focus of this thesis and was not explored in detail and is reserved for future work in this area.

2.3.2 Embedded coding

Embedded coding used in this thesis utilizes two layers of quality, *perceptual layer* and *lossless layer.* The difference being that the lossless layer is not scalable by itself and the perceptual layer is created using standard perceptual coder with no modification, thus giving two-step scalability in the final bitstream. Since no modification is performed on the actual perceptual layer audio coder, the work in this thesis concentrates on the selection of the perceptual coder to be used and compression techniques for the lossless layer.

2.4 Decorrelation techniques in lossless audio coding

It is well known that audio signals have a significant amount of correlation between samples [18]. In order to minimise the compression rate, a decorrelation step has to be performed prior to the entropy coding stage that serves to transform the original audio signal into a signal with a lower entropy. While the perceptual coder of an embedded audio coder removes some correlation in the original signal, it is shown in Chapter 4 that further decorrelation is necessary.

For lossless audio coding purposes, the predominant technique is predictive coding based on past samples of audio data, which is used in several variations in lossless audio coders such as [20-23, 34]. Predictive coding aims to predict the current sample using a linear combination of past samples. The prediction error is then transmitted to the decoder, which then can reconstruct the original signal using the prediction model used in the encoder and taking into account the prediction error of the current sample.

2.4.1 Linear predictive coding

The Linear Predictive Coding (LPC) method predicts the value of the present sample using values of past samples. LPC has been studied and used extensively in the field of speech coding [15, 35]. The premise is that in a speech signal, short term correlation between samples in a frame is exploitable to represent the speech signal with a lower entropy signal denoted as the *LPC residual*, which is the difference between the predicted sample value and the actual value. This premise relies on the assumption that a speech signal is relatively stationary over a short period of time.

The sample prediction and recovery technique using LPC is described in equations (2.10) - (2.14).

$$\hat{S}(n) = \sum_{i=1}^{N} a_i \, S(n-i) \tag{2.10}$$

$$E(n) = S(n) - \hat{S}(n) \tag{2.11}$$

Thus, combining (2.10) and (2.11) we arrive at (2.12).

$$E(n) = S(n) - \sum_{i=1}^{N} a_i \, S(n-i) \tag{2.12}$$

Where *S(n)* is the n[th] signal sample, $\hat{S}(n)$ is the predicted sample, *E(n)* is the difference (error or residual) of the signal sample and the predicted sample, a_i is the i[th] LPC coefficient for the current frame and *N* is the prediction order. The coefficients *a* are typically calculated using an autocorrelation method and solved using the recursive *Levinson Durbin algorithm* [15].

If a signal is predictable, the LPC residual signal has a lower entropy and lower magnitude compared with the input signal. The encoder then transmits the LP coefficients *a* and the LP residual signal *E* to the receiver, which then performs the reverse process to obtain the original signal *S* with:

$$S(n) = E(n) + \hat{S}(n) \tag{2.13}$$

Hence, combining (2.10) and (2.13) we arrive at:

$$S(n) = E(n) + \sum_{i=1}^{N} a_i \, S(n-i) \tag{2.14}$$

2.4.1.1 Forward linear prediction

The most common LPC implementation involves calculating the coefficients *a* using the current frame in equation (2.10) by minimizing the error between *S(i)* and $\hat{S}(i)$, hence ensuring the prediction model adapts with the signal characteristic appropriately. The implementations of lossless audio coding in [18, 20-23] are using the forward LPC paradigm to perform decorrelation before entropy coding.

2.4.1.2 Backward linear prediction

Another paradigm involving LPC is called *backward lpc*, in which the coefficients *a* are not transmitted to the receiver. The receiver decodes the signal by calculating the LPC coefficients using previously decoded signal samples. This approach has the advantage of not requiring bitrate overhead for transmitting the coefficients, but has the disadvantage of a mismatching prediction model for a given frame, thus resulting in sub optimal performance compared to forward LPC [36]. However, if adjacent frames are

relatively similar, the disadvantage of lower prediction gain could be countered by increasing the prediction order at the expense of increased complexity [34, 36]. Reference [34] identifies that the advantage of increasing prediction order without increasing bitrate overhead could result in relatively comparable performance to forward LPC.

2.4.2 SHORTEN linear predictor

The lossless coder SHORTEN [20] aims to perform linear prediction on audio signals using a lower complexity technique than the LPC method. While LPC is known to be able to lower the entropy of a signal by a significant amount, the complexity required to perform LPC might not be desirable for some purposes such as the implementation on low cost hardware. The SHORTEN coder replaces the LPC process with a greatly simplified curve-fitting predictor that attempts to fit a p^{th} order polynomial to the last p data points.

$$\hat{s}_0(n) = 0 \tag{2.15}$$

$$\hat{s}_1(n) = s(n-1) \tag{2.16}$$

$$\hat{s}_2(n) = 2s(n-1) - s(n-2) \tag{2.17}$$

$$\hat{s}_3(n) = 3s(n-1) - 3s(n-2) + s(n-3) \tag{2.18}$$

With error signals:

$$e_0(n) = s(n) \tag{2.19}$$

$$e_1(n) = e_0(n) - e_0(n-1) \tag{2.20}$$

$$e_2(n) = e_1(n) - e_1(n-1) \tag{2.21}$$

$$e_3(n) = e_2(n) - e_2(n-1) \tag{2.22}$$

The SHORTEN decorrelation algorithm calculates all four predictions using equations (2.15) to (2.18) for a given audio data frame and chooses the best performing one in terms of the smallest error. The algorithm has low complexity compared to LPC,

while LPC customarily has to perform Levinson Durbin recursion on the autocorrelation coefficients to obtain the prediction model, the SHORTEN algorithm consists primarily of addition and subtraction for all four defined orders of prediction.

It is shown that while the prediction gain of the SHORTEN predictor is comparable to the LPC approach of low orders, the performance at higher orders is inferior. However, the lower complexity afforded by the SHORTEN predictor yields a satisfactory performance. The SHORTEN predictor is also implemented in [18].

2.4.3 Non-linear prediction

Non-linear prediction predicts the next sample using a non-linear combination of past samples. It is noted in [37] that one of the reasons LPC performs well is its inherent non-linearity by adapting the prediction coefficients (*a(i)* in equation (2.10)) for each frame of an input signal. Non-linear prediction generally takes the form of equation (2.23).

$$x(n) = a\left(x(n-1), x(n-2), \ldots, x(n-N), e(n)\right) \tag{2.23}$$

Where in (2.23), *a(·)* refers to a non-linear function based on combination of past samples *x(n-1), x(n-2), …, x(n-N)* and *e(n)* is a white noise signal where *n* is the current sample and *N* is the prediction order. It is important to distinguish the parameter *a(·)* in (2.23) and a_i in (2.10), where the former is a non-linear function and the latter is a weighting coefficient, calculated by minimizing the error between the current sample and the predicted sample.

Due to the difficulty in finding the solution to *a(·)* that minimizes the error between the current sample and the predicted sample, non-linear prediction has not been very popular although it was researched quite extensively in the literature as evident in [37].

A non-linear prediction technique is currently implemented in Monkey's Audio [22] as a decorrelation method (combined with a fixed predictor) prior to entropy coding. In Monkey's Audio, *a(·)* is approximated using a neural network approach. It is observed that not many lossless audio coders explore non-linear prediction as a decorrelation technique. Most lossless audio coders, including the new MPEG-4 ALS [23], implement LPC.

2.4.4 Lossless audio decorrelation techniques comparison

This section has summarized decorrelation techniques via prediction either by linear combination or non-linear combination of past samples. In lossless audio coding, LPC and its derivatives are the predominant technique used in [20-23, 34] while non-linear prediction is used in [22]. In [20-23], forward LPC or its modification is used, while backward LPC is used in [34]. Non-linear prediction has not been used extensively in audio coding, due to the difficulty in approximating the prediction function.

While the popularity of forward LPC is overwhelming in the literature of audio coding, the use of backward LPC for lossless decorrelation technique for audio has not been researched extensively due to backward LPC's disadvantage of a mismatching prediction model. However, in [34] it was shown that given a high enough LPC order, the performance of forward and backward LPC converges. The advantage of not requiring side information of backward LPC thus simplifies the resulting compressed bitstream. Hence, backward LPC was chosen for the coder developed in this thesis.

2.5 Chapter summary

This chapter has discussed audio coding techniques. In summary, there are two different paradigms of audio coding: perceptual coding and lossless coding. Perceptual coding exploits the fact that the human ear has limited and non-uniform sensitivity across frequencies and thus removes perceptually irrelevant data, whereas lossless coding exploits the fact that audio signals have high correlation between samples and thus removes redundant data. The goal of both types of coders are similar, that is, by lowering the entropy of the signal either by removing certain parts or reducing correlation, compression is achieved.

To combine these two approaches, the scalable to lossless coding paradigm allows one to choose between the two approaches with high flexibility. However a scalable approach requires the development, implementation and distribution of an entirely new coder or modifications of existing approaches.

The embedded lossless coding technique aims to combine perceptual and lossless paradigms by using existing perceptual coders that are already widely distributed, thus

has the advantage of wider potential distribution and better acceptance. Embedded lossless coding will be described in more detail in the remaining chapters of this thesis.

3 Perceptual base layer

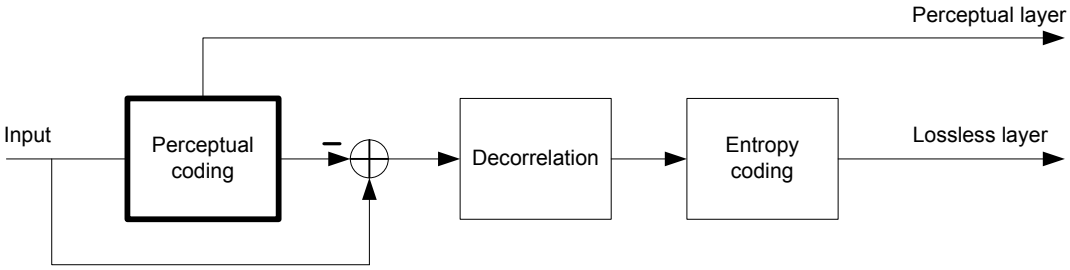


Figure 3.1. Block diagram of the embedded lossless coder.

3.1 Overview of the embedded lossless coder

This chapter describes the techniques relevant to the block labelled “Perceptual coding” in Figure 3.1, and thus contributes to the output labelled as “Perceptual layer” in the resulting coder. The goal of the perceptual layer is to achieve relative perceptual transparency, i.e. the perceptually coded signal has a sonic quality that is virtually indistinguishable from the original as perceived by the human ear. The perceptually encoded signal is then decoded and subtracted from the original signal, creating an error signal denoted as *residual signal*. The residual signal essentially contains the quantization noise that the perceptual coder attempts to hide under the psychoacoustic masking threshold. An analysis of the residual characteristic is therefore an indirect measurement of the psychoacoustic characteristics of the underlying perceptual coder [12].

Since the objective is to achieve the lowest overall file size for the resulting embedded lossless bitstream, an analysis of the characteristics of the residual signal with respect to the changing parameters and type of the perceptual coder needs to be performed.

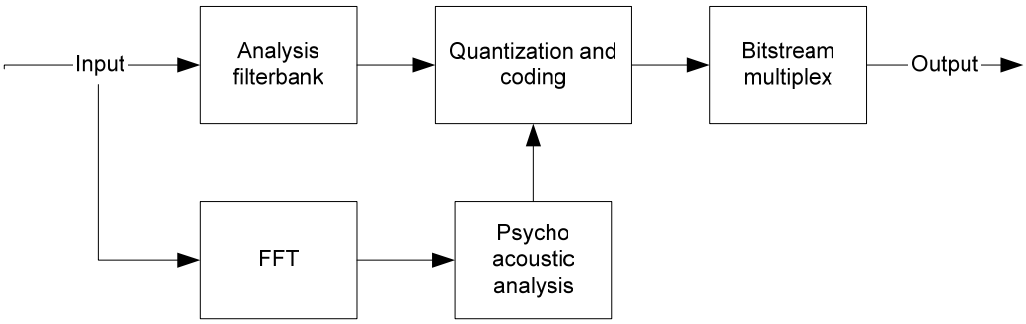


Figure 3.2. General block diagram of a perceptual coder.

Section 3.2 will describe the general operating procedures of perceptual coders as well as specific details of the perceptual coders tested. Section 3.3 will describe the test procedures performed. Section 3.4 will detail the analysis performed on the resulting residual signal. Section 3.5 will compare two methods of achieving an embedded lossless coder. Section 3.6 will then summarizes the analysis results and concludes this chapter.

3.2 Overview of the perceptual coder

Most perceptual coders operate using the same basic principles: by windowing the input signal and performing a transform such as *Modified Discrete Cosine Transform* (MDCT) [38] on the windowed signal. The MDCT, which is the core of most perceptual coders, is a critically sampled transform with windows that are overlapping by 50% and with reconstruction by an *overlap and add* operation. *Critical sampling* means that the number of frequency domain coefficients which is the result of the transform, is equal to the number of samples in the time domain. Overlapping subsequent windows by 50% has the desired effect of eliminating the artifact that could occur in switching between one frame and the next due to the discontinuity in the transform coefficients between frames. In the event of a *transient* or sudden change of amplitude in the signal, most perceptual coders adjust the windowing function using a smaller window ("short window") to achieve good time resolution and thus minimize the problem of a *pre-echo* artifact [39], where the amplitude of a time domain signal is spread throughout the window in the resulting frequency domain representation, thus creating an audible artifact. Otherwise, a "long window", typically 3-8 times the length of the short window, is used to achieve good frequency resolution. The resulting MDCT coefficients are then quantized using psychoacoustic criteria and the bitrate available to the coder,

such that the quantization noise is kept below a masking threshold. The resulting, perceptually weighted, quantized MDCT coefficients are then multiplexed by side information required to reverse transform the coefficients correctly. The general block diagram for a perceptual coder can be observed in Figure 3.2.

Since coders are generally designed with a specific bitrate in mind, good performance at a certain bitrate does not necessarily translate to better performance overall at other bitrates.

3.2.1 MPEG-1 Layer III

MPEG-1 Layer III, also known as MP3, utilizes subband coding along with the MDCT transform. The input signal is first decimated into 32 equal subbands with the long window size of 36 samples before being transformed into 18 spectral lines by the MDCT, totalling 576 MDCT coefficients for one frame of audio data. For transient handling, the window length is 12 samples, transformed into 6 MDCT coefficients per short window, totalling 192 MDCT coefficients per short window. Every subband is then perceptually weighted using a psychoacoustic measurement before Huffman encoding and multiplexing with side information.

MP3 is inherently a *variable rate coder*, where the size of an encoded frame can vary from one frame to the next. To achieve this functionality, MP3 uses a technique called the *bit reservoir*. If a given frame does not require the full bandwidth for encoding, the leftover bits are allowed to be saved for future frames where the allocated bandwidth may not be sufficient to encode a frame without distortion. Hence the overall bitrate of an MP3 encoded file may be lower than specified, although the use of the bit reservoir is regulated so that the overall average bits per frame does not exceed the allocated maximum.

3.2.2 MPEG-4 AAC

MPEG-4 AAC is essentially an identical coder to that described in the MPEG-2 AAC specification. The MPEG-4 version adds some new coding tools specifically designed for low bitrate encoding such as vector quantization via *Transform-domain Weighted Interleave Vector Quantization* (TwinVQ) , *Perceptual Noise Substitution* (PNS) and *Long Term Prediction* (LTP) [2].

The principles of operation of an AAC coder, are different to the MP3 coder. The AAC coder does not decimate the input signal into subbands before the MDCT transformation. The size of a long window in AAC is 2048 samples, which are transformed into 1024 MDCT coefficients. For the short window, the window length is 256 samples, which results in 128 MDCT coefficients. Note that the frequency domain representation of AAC in the long window is significantly better than in MP3 (1024 coefficients in AAC vs. 576 in MP3). Hence, AAC has a better frequency resolution during long window compared to MP3. During transient periods, AAC has fewer MDCT coefficients per short window (128 coefficients in AAC vs. 192 in MP3) and therefore in a transient period AAC has better time resolution compared to MP3 which contributes to better transient representation and less pre- and post-echo artifacts occurring [3].

3.2.3 Ogg Vorbis

Ogg Vorbis [40] is an open source high quality perceptual audio coder that is designed to be fully extendable into the future. Whereas in MPEG standards the decoder is strictly defined with all the predefined Huffman codebook and framesize convention, Vorbis does not define any codebook or framesize. The Vorbis format is essentially free form, with the decoder required to be able to decode a partial bitstream using any framesize with the only requirement being that all side information be placed before any audio data is sent. Due to the designed extensibility of Vorbis, the side information required to decode the bitstream is relatively large compared to MPEG standards and hence a Vorbis decoder cannot arbitrarily decode part of the bitstream without first decoding the side information.

The principles of operation are similar to MPEG-2/4 AAC specification, with the input signal transformed into the frequency domain using an MDCT which are then perceptually weighted using psychoacoustic measures. However, Vorbis defines two methods of entropy coding that is to be used simultaneously within a bitstream: after MDCT transformation, the coarse representation of the spectral values are Huffman coded (called the “Floor value”), with the fine structure representation of the audio data (called the “residue”) encoded using multi-stage vector quantization. Due to the designed extensibility, none of the Huffman or VQ codebooks are predefined, hence the full codebook needs to be sent to the receiver as part of the side information.

3.3 Description of the test procedure

Genre	Count
Ambient	3
Blues	1
Classical	5
Electronic	35
Jazz	2
Pop	31
Rock	11
Total	88

Table 3.1. Genre breakdown of test signals

The test signals used were obtained from Q-Music [41], and consist of 88 music files in the genres of electronic, pop, rock, classical, ambient, jazz and blues. The genre breakdown is as shown in Table 3.1.

The signals are sampled at 44.1 KHz with 16 bit quantization stereo. The length of the files vary from 9 seconds to 7 minutes 10 seconds, with an average of 2 minutes 52 seconds. For the experiment, all the signals were reduced to mono by taking the left channel and discarding the right channel. This is done to simplify the experiment, since in [18] it was found that the correlation between channels are minimal and therefore a multichannel lossless coder generally compresses each channel independently.

Four perceptual coder implementations were tested: Nero MPEG-4 AAC coder [42], PsyTel AAC coder [43], LAME MPEG-1 Layer III coder [44] and Ogg Vorbis coder [40]. The four coders were chosen to include the widest range of perceptual base layer possibilities, with an MPEG-4 AAC coder, popular MPEG-1 Layer III coder and a popular open source perceptual coder. The tests were performec with bitrates of 64, 96, 128, 192, 160, 224 and 256 kbps. The encoding profile used in MPEG-4 encoder is the *low complexity* profile (MPEG-4 AAC LC). Since the goal of this work is lossless coding using a perceptually lossless base layer coding, low bitrate coding tools in MPEG-4 AAC specification are not used in this work.

To analyze the residual signal, three methods of measurement were used: the entropy, measured using expression (3.24), the Signal to Noise Ratio (SNR), measured

using expression (3.25) and the bitrate of the residual signal compressed using existing lossless coders, measured using expression(3.26).

$$H = -\sum_{x} p_x \log_2 p_x \tag{3.24}$$

$$SNR(dB) = 10 \log_{10} \frac{\sum_{i=1}^{N} S_i^{\,2}}{\sum_{i=1}^{N} E_i^{\,2}} \tag{3.25}$$

$$B = 16 \times \frac{C}{S} \tag{3.26}$$

In (3.24), H is the entropy and x is the number of samples with respective symbol probabilities p. In (3.25) S denotes the original signal and E denotes the error or residual with the measured SNR using a decibel scale measurement. In (3.26), B denotes the bitrate of the compressed signal assuming the original signal was encoded using 16 bitrate, with C and S representing the filesize of the compressed signal and the filesize of the original signal, respectively.

The first analysis method, entropy calculation, is used in this thesis as a measurement of the theoretical minimum bitrate required to code a given signal into its binary representation with the assumption that the source signal is *independent and identically distributed* (i.i.d.). Hence, the entropy metric is the theoretical coding lower bound on a source signal without any decorrelation. Note that the entropy metric does not consider the decodability of the signal, i.e. the size of the overhead information required to decode the signal correctly is not included in the calculation.

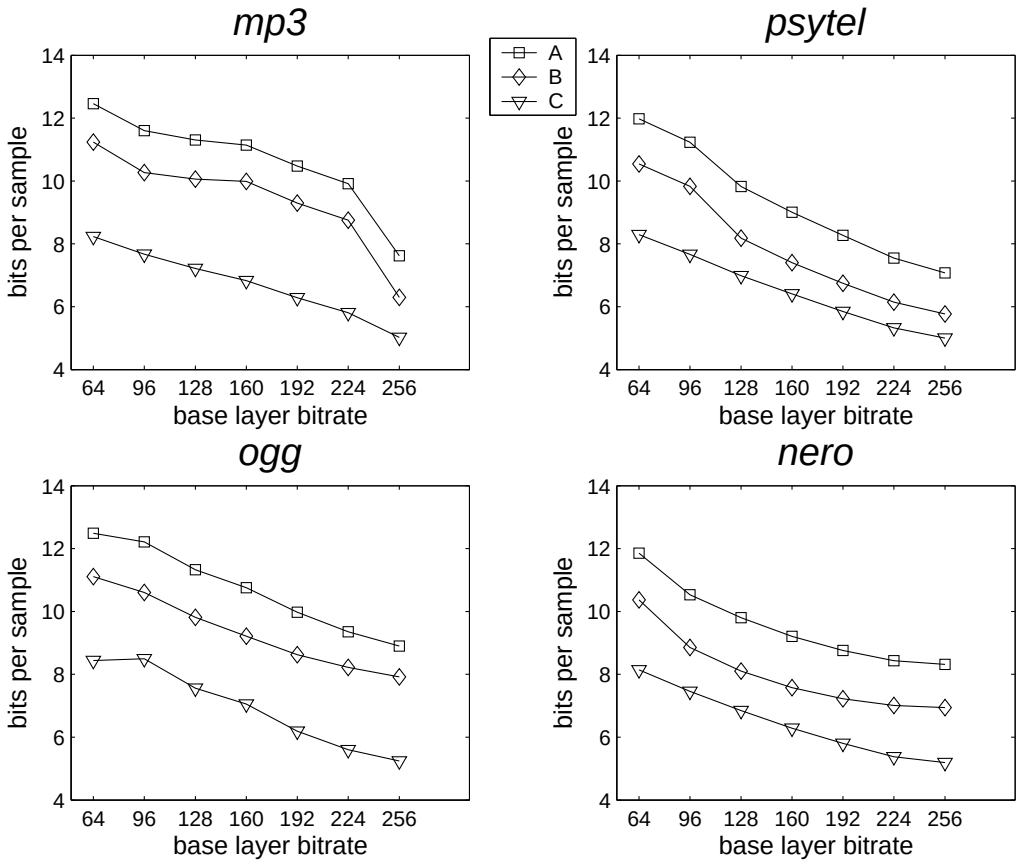


Figure 3.3. Entropy and compression of the residual signal using gzip and Monkey's Audio. Line A represents residual compressed with gzip, line B represents residual entropy and line C represents the residual compressed with Monkey's Audio.

The second analysis method used in this experiment calculates the Signal to Noise Ratio (SNR) between the residual and the original signal so as to measure the level of quantization noise present in the residual signal. SNR is used instead of Signal to Mask Ratio (SMR) [39] because lossless coding is not concerned with perceptual masking. The SNR measurement is therefore used in this thesis to show the objective scalability of the base layer coders across tested bitrates.

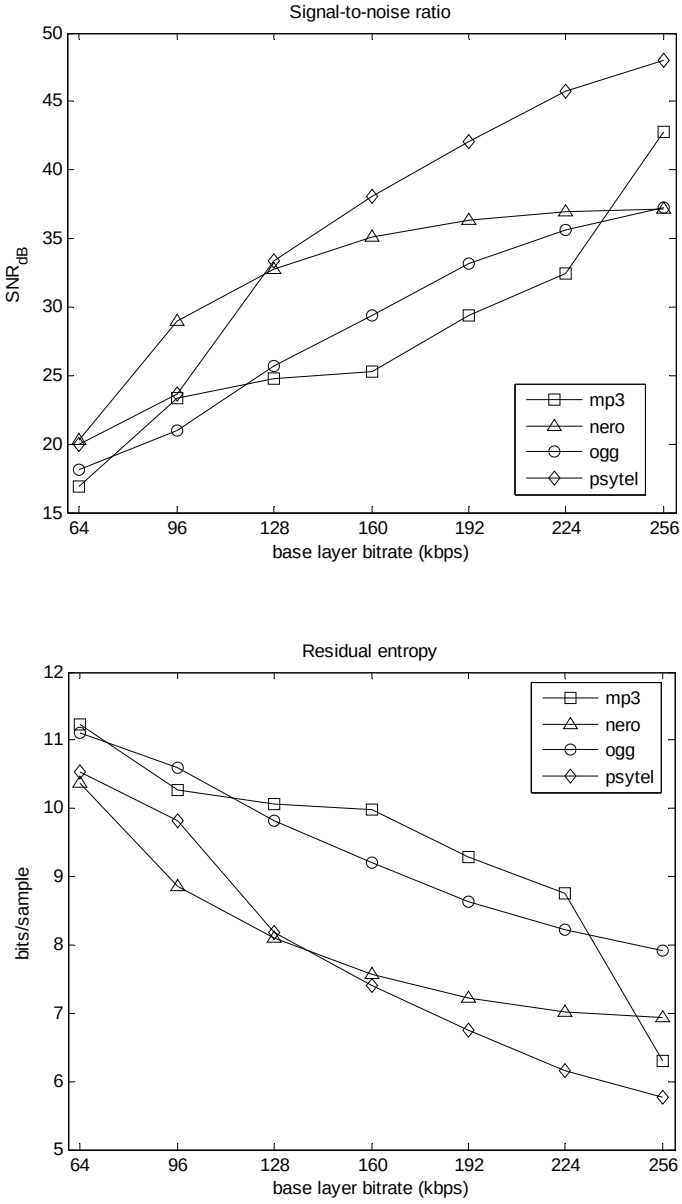


Figure 3.4. Signal-to-noise ratio and entropy of the signal.

A compression of the residual is used as the third analysis method to measure the performance of real world compression algorithms compared with the two aforementioned objective measurements. Using the two different lossless compression techniques is beneficial for comparing the performance of two different methods applied to the residual signal, one which involves decorrelation and one without. Gzip was chosen as a widely available general purpose compression algorithm employing the well known LZ77 algorithm [14], and Monkey's Audio [22] represents the state of the art in audio lossless coding, which employs a decorrelation step in the coding process.

3.4 Residual entropy, compression and SNR results

3.4.1 Entropy and compression of the residual

In Figure 3.3, it is shown that across all tested perceptual coders, the entropy of the residual signal has a descending trend when the perceptual coder bitrate is increased. This is due to the characteristic of the residual signal, which consists of quantization noise resulting from the perceptual coding of the original signal. As the perceptual coder bitrate is increased, the quantization noise present in the residual signal will then have lower variance and dynamic range, which results in reduction in the residual entropy. Note that the measurement metric used in Figure 3.3 is bits/sample in order to decouple them from the sampling rate of the signal.

The bitrate of the residual signal compressed with Monkey's Audio has a consistently lower value compared to the entropy of the residual signal itself. This is expected, since Monkey's Audio decorrelates its input signal before any entropy coding step. This also means that the signal coded by Monkey's Audio's entropy coding step is not the same signal as the signal coded by gzip. Also, the entropy measurement assumes the *independent identically distributed* (i.i.d.) property, which is clearly not the case here. Note that the lowest compressed residual bitrate is achieved in all coders when the perceptual coder bitrate is the highest. This is expected, since at the highest perceptual coder bitrate, the quantization noise in the residual will have the lowest variance and dynamic range compared to all other bitrates.

Hence, while direct comparison of signals coded by Monkey's Audio and gzip may not be valid, the average bitrate result from Monkey's Audio gives an indication of how much correlation still exist in the residual signal, despite the residual containing quantization errors and masked frequencies. Hence, this indicates that correlation may still be present in the residual signal.

Coder	Total lossless bitrate	Residual SNR
Nero AAC, 64 kbps	9.6 bits/sample	20.28 dB
Nero AAC, 96 kbps	9.64 bits/sample	28.93 dB
MP3, 64 kbps	9.68 bits/sample	16.92 dB
Psytel AAC, 64 kbps	9.74 bits/sample	19.94 dB
Nero AAC, 128 kbps	9.75 bits/sample	32.77 dB
Psytel AAC, 96 kbps	9.84 bits/sample	23.6 dB
MP3, 96 kbps	9.85 bits/sample	23.33 dB
Psytel AAC, 128 kbps	9.9 bits/sample	33.33 dB
Ogg Vorbis, 64 kbps	9.9 bits/sample	18.14 dB
Nero AAC, 160 kbps	9.92 bits/sample	35.1 dB

Table 3.2. Comparison of total lossless bitrate and residual signal-to-noise ratio with residual compressed using Monkey's Audio.

Figure 3.5 shows that the lowest overall bitrate can be achieved by employing Nero AAC as the perceptual coder and employing Monkey's Audio to losslessly compress the residual signal, with the bitrate for the Nero AAC set at 96 kbps mono.

3.4.2 Signal to noise ratio

Figure 3.4 depicts the signal-to-noise ratio compared to the entropy of the signal of various base layer bitrates of the tested coders. As expected, as the base layer bitrate is increased, the SNR of the residual signal decreases, as there is less quantization error present in the residual signal. The entropy of the residual also decreases due to the reduced variance present in the quantization noise as the perceptually coded signal is closer to the original signal.

Note that the SNR curve mimics the inverted shape of the entropy curve in Figure 3.4. This suggests that the SNR function is related to the inverse of the entropy function for the residual of a perceptual coder, hence we can estimate the SNR of a given residual by its entropy and vice versa.

3.4.3 Total Lossless bitrate comparison

The total bitrate of the embedded coder described in this thesis is found by adding the bitrate of the perceptual layer with the bitrate of the residual signal, which is

calculated with the three methods as described in section 3.3. The overall result is shown in Figure 3.5.

Note that even though the residual bitrate has a decreasing trend either by compressing it or calculating its entropy value as shown in Figure 3.3, the total bitrate curve in Figure 3.5 is more irregular in its shape. Figure 3.5 therefore suggests that there is no definite pattern in the residual characteristic with respect to the perceptual coder bitrate and every perceptual coder has a different pattern based on its perceptual model.

It can be seen in Figure 3.5 that the curves for gzip and the entropy measurements follows a general shape while the curve for Monkey's Audio does not, due to additional decorrelation performed by Monkey's Audio. However, all curves show an increasing trend as the base layer bitrate is increased, except for the lowest Nero AAC bitrate and the highest MP3 bitrate, which can be attributed to the coders being tuned to perform maximally under certain bitrates. The absolute lowest overall total bitrate assuming Monkey's Audio is used to compress the residual is ≈9.6 bits/sample, which is exhibited in Table 3.2. This bitrate is achieved when Nero AAC is used as the perceptual coder using 64 kbps bitrate and Monkey's Audio is used to compress the residual signal. However, in Table 3.2 it is also evident that if Nero AAC is operated using 96 kbps, the total lossless bitrate is increased by only ≈0.04 bits/sample, while the signal-to-noise ratio between the original signal and the residual signal is increased by ≈8.65 dB which suggests that the perceptual quality of the base layer is increased significantly with minimal impact on the total lossless bitrate. Based on the observation in Table 3.2 and sections 3.4.1, 3.4.3 and Figure 3.5, subsequent work in this thesis then utilizes Nero AAC operated with 96 kbps mono as the perceptual base layer.

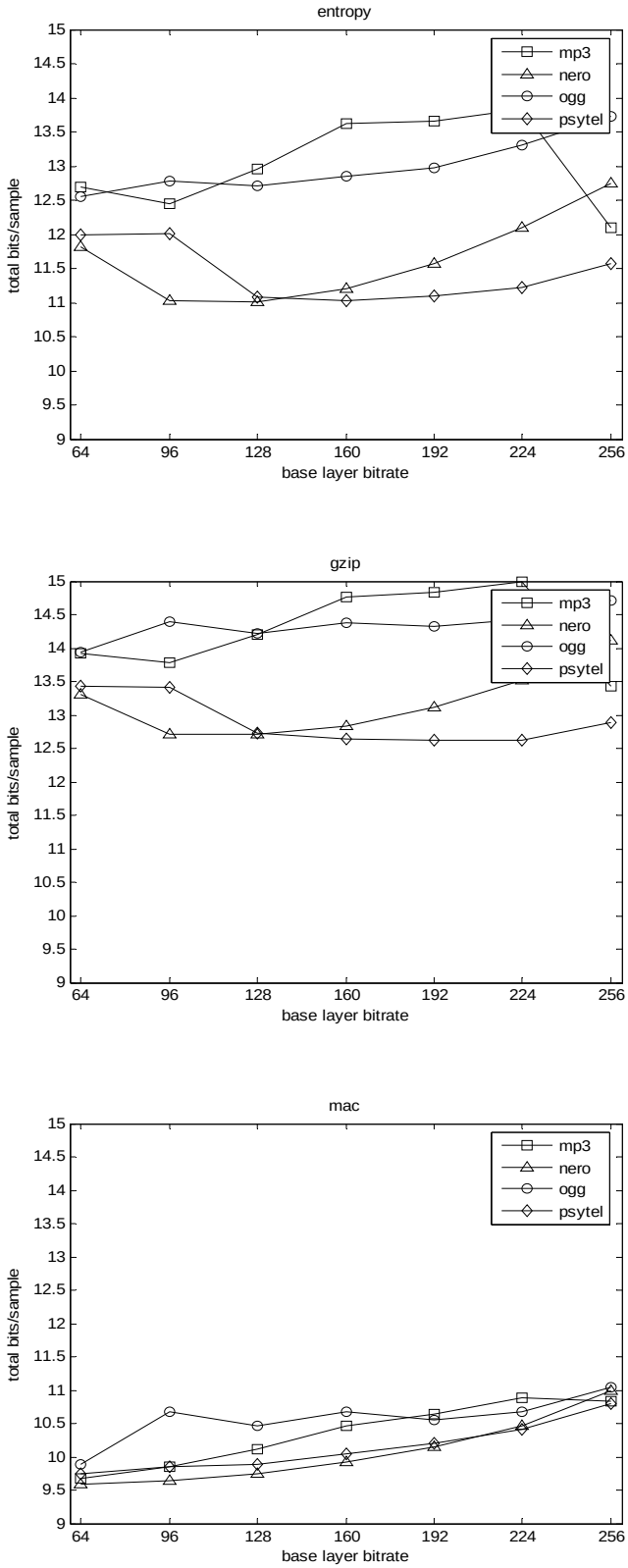


Figure 3.5. Total bitrate (perceptual layer + lossless layer) comparison.

3.5 Comparison of embedded lossless coding methods and lossless only coding

Nero AAC bitrate	Lossless + perceptual	Perceptual + residual
64 kbps	16.0%	5.9%
96 kbps	24.0%	6.4%
128 kbps	32.0%	7.6%
160 kbps	40.0%	9.4%
192 kbps	48.1%	12.0%
224 kbps	56.1%	15.4%
256 kbps	64.1%	21.3%

Table 3.3. Percentage of filesize increase of two embedded lossless coding methods.

There are two methods for achieving an embedded lossless coding. The first method depicted in Figure 3.1 is achieved by compressing the perceptual layer residual signal and by combining the two layers to create a lossless bitstream with a perceptual bitstream embedded. The second method is to create a lossless compressed version of the original signal and multiplexing the perceptual version of the signal to create the embedded lossless bitstream. While the second method might not appear valid, it is less complex than the first method due to the lack of need in the encoder to create the residual signal and the lack of need in the decoder to decode both perceptual and lossless enhancement layers to obtain the lossless version of the signal.

Table 3.3 shows the percentage of overall filesize increase using Monkey's Audio as the lossless coder and Nero AAC encoder as the perceptual coder. The column marked as Lossless+perceptual shows the percentage of filesize increase by combining the Monkey's Audio lossless compressed version and adding an AAC compressed perceptual layer. The column marked Perceptual+residual shows the percentage of filesize increase by perceptually coding the original signal and appending the Monkey's Audio compressed residual signal.

In Table 3.3, it is shown that the Lossless+perceptual method total filesize increases quickly as the perceptual base layer bitrate is increased, while the total filesize increase for the lossy+residual method increases much less. Combined with the results in section 3.4.3 which show that the lowest overall bitrate for an embedded lossless coder using

Nero AAC and Monkey's Audio is 96 kbps mono, the percentage of filesize increase is only 6.37% compared to lossless only coding using Monkey's Audio.

3.6 Chapter summary

This chapter has discussed the feasibility of using AAC as the perceptual layer of an embedded lossless coder and the existence of correlation in the AAC residual signal. Section 3.4.1 found that correlation still exist in the residual signal since compression of the residual using Monkey's Audio yields a lower bitrate compared to the entropy of the residual. Sections 3.4.2 and 3.4.3 has discussed that the best performing AAC base layer bitrate in terms of total filesize and signal-to-noise ratio is 96 kbps mono. Section 0 has discussed the feasibility of using a perceptual coder as the base layer and lossless compression of the residual signal results in only ≈6% compared to lossless only coding of the original signal. The rest of this thesis will then utilize the perceptual coding and lossless compression of the residual paradigm using 96 kbps mono as the perceptual layer bitrate.

4 Decorrelation and entropy coding

4.1 Decorrelation of the AAC residual

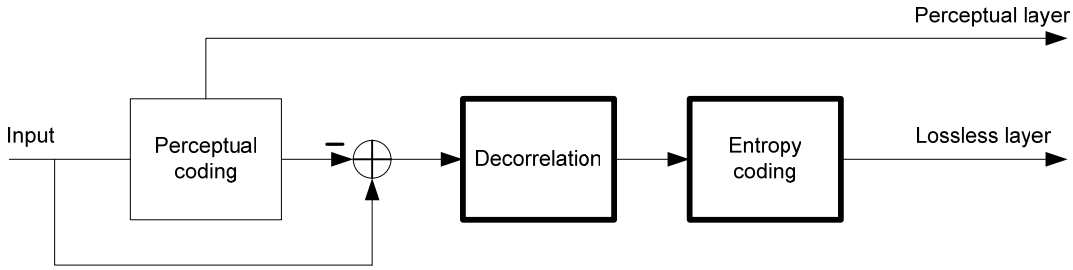


Figure 4.1. Block diagram of the embedded lossless coder.

4.1.1 Overview of decorrelation methods in lossless audio coding

This chapter describes the techniques for the block labeled as “Decorrelation” and thus contributes to the output labeled as “Lossless layer” in Figure 4.1. Since the perceptual coder described in Chapter 3 aims to represent the input signal in a perceptually transparent manner, some correlation might still exist in the AAC residual signal in the form of masked frequencies with periodic components. Therefore, to enhance the performance of the subsequent entropy coding stage, further decorrelation of the AAC residual signal might be required.

As mentioned in Chapter 2, several decorrelation techniques are available: using a perceptual coder as a decorrelation method (which has been performed in Chapter 2), using a form of *Linear predictive coding* (LPC), using a simplified linear predictive scheme like the SHORTEN decorrelation method or using a non-linear decorrelation technique. All of the techniques mentioned have certain advantages and disadvantages. Primarily the balance between prediction gain, complexity and overhead information size needs to be found. *Prediction gain* is the signal-to-noise ratio of the original signal and the prediction residual, *complexity* is the amount of processing power required to

perform prediction and *overhead information* is the information that needs to be sent to the receiver in order to perform reverse prediction and thus recover the original signal.

The prediction gain of forward adaptive LPC is the primary reason for its popularity in state of the art lossless audio coder such as MPEG-4 ALS [23] and FLAC [21]. Non-linear decorrelation techniques are not as popular due to the difficulty in finding the optimal non-linear function and the complexity in doing so. However, a non-linear predictor is implemented in Monkey's Audio [22] which is considered the best performing lossless audio coder currently available. Although both of these techniques exhibit high prediction gain, both requires overhead information in the form of LPC coefficients needed to reconstruct the signal in the receiver in the case of forward adaptive LPC and the parameters of the non-linear function in the case of non-linear prediction be sent to the receiver to correctly decode the signal. Backward adaptive LPC, on the other hand, does not require overhead information sent to the receiver with the disadvantage of lower prediction gain.

4.1.1.1 Decorrelation via backward linear predictive coding

The backward adaptive LPC decorrelation technique performs LPC analysis in the previous frame and applies the LPC coefficients on the current frame, whereas forward adaptive LPC performs both LPC analysis and processing on the current frame (both LPC techniques were discussed in detail in Chapter 2). Hence the prediction model might not be the optimal for any frame. However, the LPC coefficients required to synthesize the current frame can be obtained from the previously decoded frame, hence backward predictive LPC does not require the LPC coefficients be sent to the receiver as side information. It is shown in [34] that the problem of mismatching prediction models of backward adaptive LPC can be countered by increasing the prediction order. Since backward adaptive LPC does not require side information to be sent, increasing the prediction order only result in higher complexity without increasing the size of the bitstream.

The LPC process is described in Chapter 2 and shown in equations (4.27), (4.28) and (4.29) with the LPC prediction, error or residual and synthesis, respectively for

completeness. The goal is to minimize the error *E(n)* by predicting the signal *S(n)* using $\hat{S}(n)$.

$$\hat{S}(n) = \sum_{i=1}^{N} a_i S(n-i) \tag{4.27}$$

$$E(n) = S(n) - \hat{S}(n) \tag{4.28}$$

$$S(n) = \hat{S}(n) + E(n) \tag{4.29}$$

Note that the LPC process uses a floating point operation by design, which creates rounding errors upon resynthesizing to 16 bit values which constitute the input audio signal samples. To avoid the rounding problem which will create a non perfect reconstruction upon LPC synthesis, a rounded backward adaptive LPC which modifies equations (4.28) and (4.29) into equations (4.30) and (4.31) is used in this thesis.

$$E(n) = S(n) - fix\left(\hat{S}(n)\right) \tag{4.30}$$

$$S(n) = fix\left(\hat{S}(n)\right) + E(n) \tag{4.31}$$

In equations (4.30) and (4.31), *fix(·)* denotes the rounding of floating point value to the nearest 16 bit integer toward zero. This restricts the LPC residual *E(n)* to 16 bit integer values as well, hence allowing the LPC synthesis process in (4.31) to achieve perfect reconstruction of the original signal *S(n)* by effectively eliminating rounding errors.

$$A(x) = \begin{cases} 0.54 - 0.46\cos\left(\dfrac{2\pi x}{a-1}\right), & 0 \le x < 0.95N,\ a = 1.9N \\ \cos\left(\dfrac{2\pi x}{0.2N-1} + \dfrac{\pi}{2}\right), & 0.95N \le x \le N-1 \end{cases} \tag{4.32}$$

Equation (4.32) describes the hybrid windowing function, where *A(x)* is the left half of the Hamming window for 95% of the window length *f* where *a* is the length of a complete Hamming window. The remaining 5% of the hybrid window consists of a quarter period of a cosine function *C(x)*.

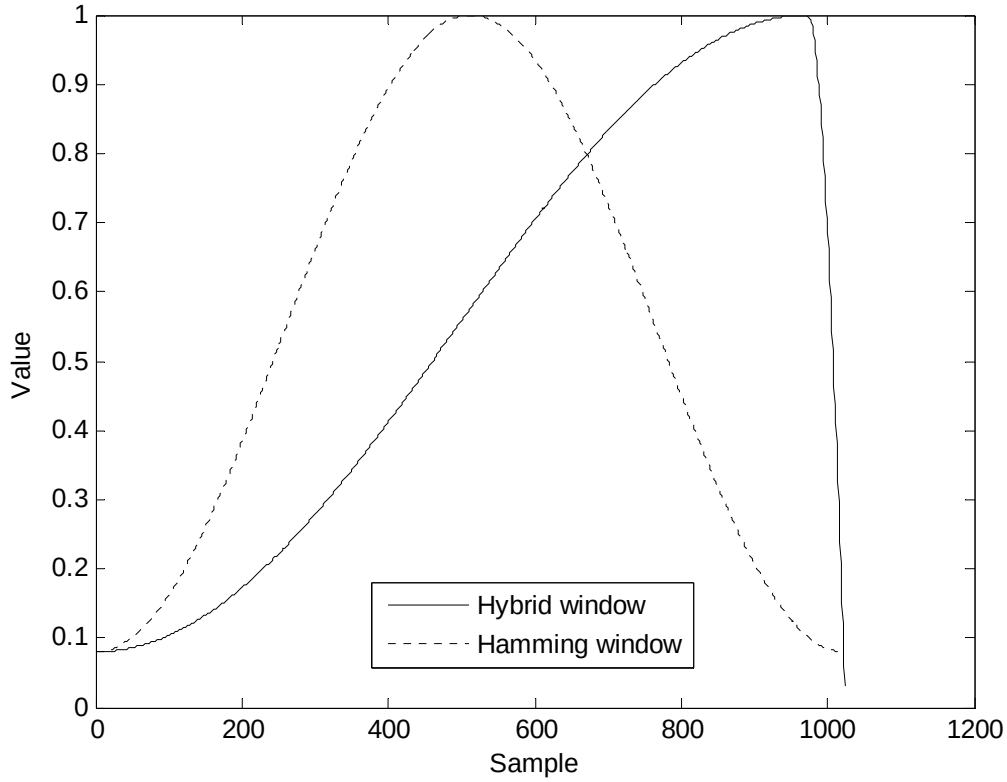


Figure 4.2. The windowing function used in the backward LPC process.

The frame length used in LPC analysis is 1024 samples, which was chosen to coincide with the frame length of the AAC coder. The windowing function used in the analysis process is depicted in and equation (4.32), which is denoted as the "Hybrid window" derived from the windowing function described in [34]. In the example in the window length is 1024 samples, which constitutes of 973 samples of Hamming window and 51 samples of cosine window. To minimize the impact of mismatching prediction model of backward adaptive LPC, the LPC process is performed using an overlapping window with a ¼ frame overlap or 256 samples in the case of a frame size of 1024 samples, and four updates of LPC coefficients per frame. The use of the hybrid window results in some prediction gain and more noise-like LPC residual signal compared to Hamming window, evident in Figure 4.3 and Figure 4.4.

To assess the performance of backward adaptive LPC, two measurement metrics are used: the *LPC gain* which is described in equation (4.33) and the *Spectral flatness measure* (SFM) which is described in equation (4.34).

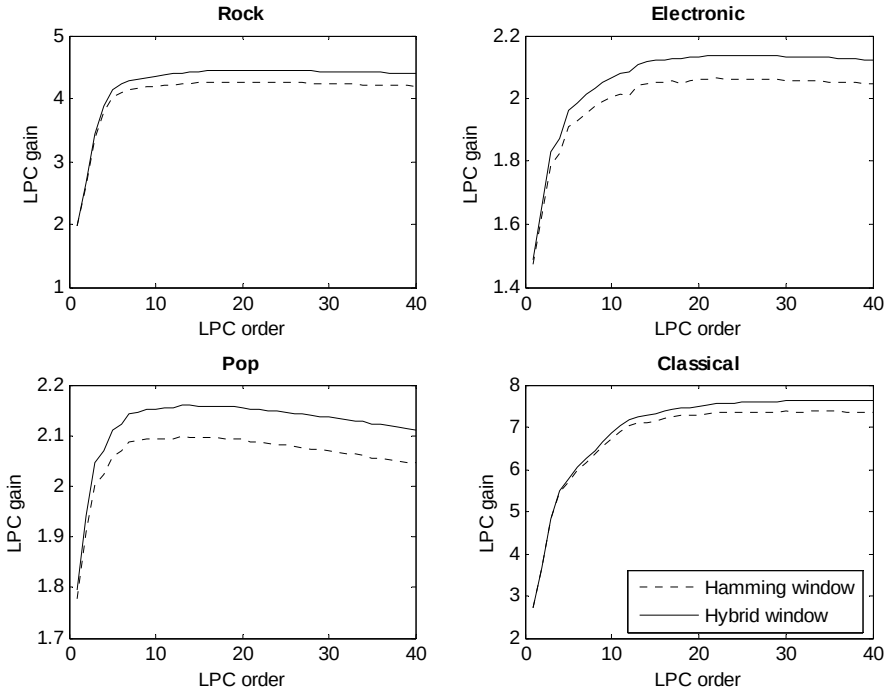


Figure 4.3. LPC gain plot for backward LPC of order 1-40 using Hamming window and Hybrid windows.

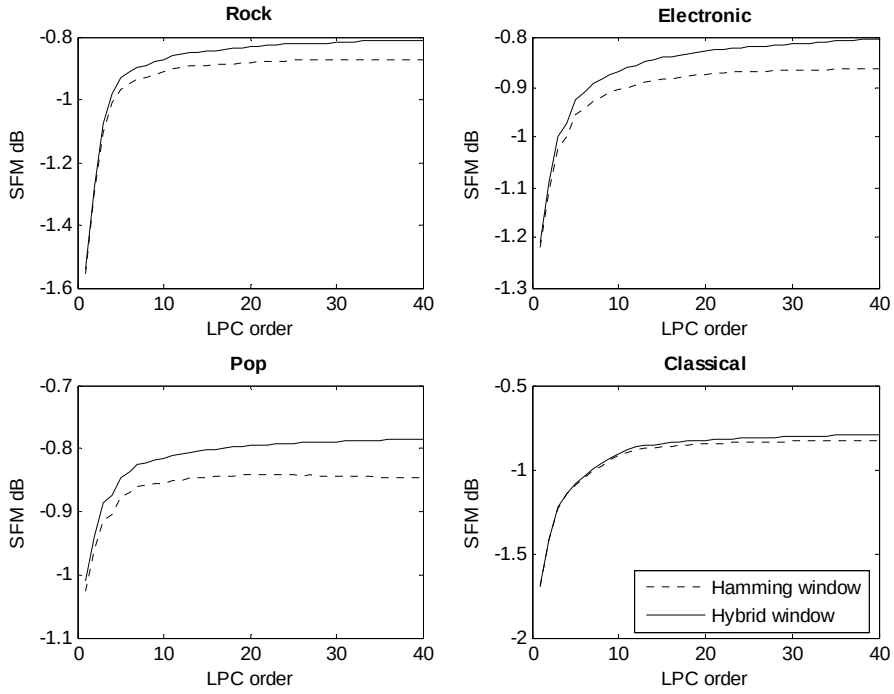


Figure 4.4. SFM measurements for the LPC residual signal of order 1-40 using Hamming window and Hybrid window.

$$P = \frac{\sum_{n=1}^{N} S_n^{\,2}}{\sum_{n=1}^{N} E_n^{\,2}} \tag{4.33}$$

$$SFM_{dB} = 10\log_{10}\left(\frac{Gm}{Am}\right) \tag{4.34}$$

Where in (4.33) P is the prediction gain, S is the original signal samples and E is the residual signal samples. In (4.34) Gm and Am are the geometric mean and arithmetic mean of the Fourier transform, or frequency domain representation, of the LPC residual signal.

Equation (4.33) is the signal-to-noise ratio of the original signal vs. the LPC residual signal, and (4.34) measures how close a signal is to white noise. SFM result of close to 0 dB means that the signal is very noise-like, whereas SFM result of about -60 dB means that the signal is very tone-like. Applied to the LPC residual signal, both metrics aim to describe how predictable is the residual signal. Hence, high prediction gain indicates low power residual and a predictable signal. An SFM measure near 0 dB indicates that the signal is spectrally flat and close to white noise, which is a desirable characteristic of an LPC residual.

The test signal database was extracted from the Q-Music database [41] used in Chapter 3, consisting of 20 files in the genre of classical, electronic, pop and rock categories. Each categories consists of five signals each, with each signal is of thirty seconds in length. The test signals are downmixed to mono by taking the left channel and discarding the right channel. All signals are quantized with 16 bits and sampled at 44.1 KHz. The signals are then encoded using Nero AAC encoder [42] and decoded back to PCM audio using 96 kbps as the AAC bitrate.

4.1.2 Decorrelation results

The LPC gain measurement result is depicted in Figure 4.3. The gain results show that classical music, which is known to have strong tone-like signals, have the most gain

compared to other genres with knee point at around LPC order 15-20. Genres rock and pop are quite similar with a knee point at around LPC order 5-10, and the electronic music genre has a knee point at around LPC order 15.

The "whiteness" of the LPC residual signal can be observed in Figure 4.4, where the SFM results show that the LPC residuals are already close to white noise prior to LPC processing. However, a small order of LPC processing performed on the residual signal does improve the "whiteness" of the signal if the hybrid window described in section 4.1.1.1 is used instead of Hamming window. The SFM plots of Figure 4.4 show similar knee points to those in Figure 4.3, which suggests that using backward LPC of order 15 on the AAC residual signal provides a good prediction gain for all of the genres tested.

4.1.3 Statistical characteristics of the AAC residual

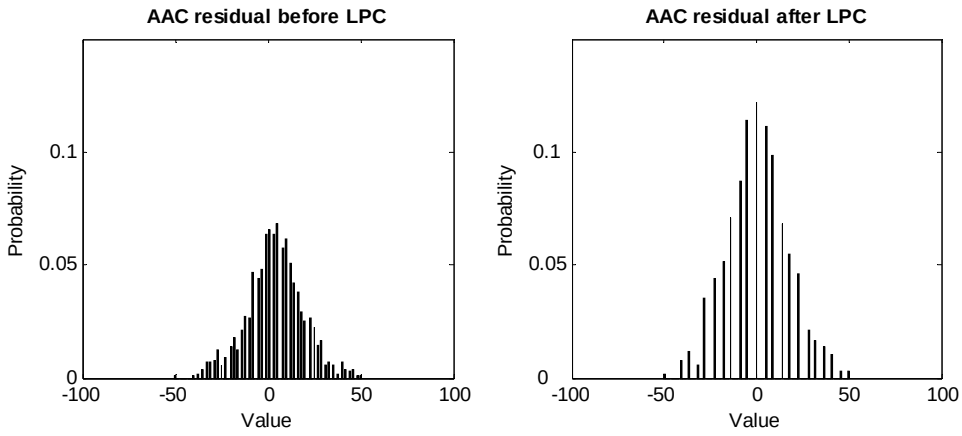


Figure 4.5. One frame of an AAC residual signal histogram before and after order 15 LPC processing.

As outlined in works such as [20, 45], the *probability density function* (p.d.f) of the residual of an LPC process follows the shape of the p.d.f. of a Laplacian distribution, with the Laplacian defined in (4.35).

$$p(x) = \frac{1}{\sqrt{2}\sigma} e^{\frac{-\sqrt{2}}{\sigma}|x|} \tag{4.35}$$

Where $|x|$ is the absolute value of x and σ^2 is the variance of the distribution.

The Laplacian p.d.f. has the characteristic of zero-mean distribution of values, with the probability of a high magnitude value occurring is considerably smaller than the probability of a small magnitude value occurring. This type of distribution function is perfectly suited for Rice coding, where the most frequently occurring values in the signal are coded using *k+1* bits. The proof that this Rice coding method is optimal was presented in [20]

As a comparison for signal characteristic before and after LPC, in Figure 4.5, the AAC residual before LPC has a mean value of 0.332 whereas the AAC residual after 15th order LPC processing has a mean value of 0.108. In addition, the probability of a zero magnitude has increased. Hence, it is shown that the LPC process moves more values in the residual toward zero and increases the probability of a low magnitude values occurring.

4.2 Entropy coding

4.2.1 Overview of entropy coding methods

This section describes the techniques for performing entropy coding or noiseless coding in the block marked “Entropy coding” in the embedded lossless coder block diagram in Figure 4.1, and thus is the final step for the output marked “Lossless layer”. The goal for this step is to transform the LPC residual signal from the decorrelation step in Chapter 4 into a binary representation understood by a computer without loss of information for transmission or storage purposes. The coding step is performed by utilizing the statistics of the decorrelated signal from the previous step in order to create a binary stream using as few bits as possible. The two entropy coding methods examined in this chapter and described in Chapter 2 are Rice coding and cascade coding. The test database used and decorrelated in Chapter 4 is then entropy coded using both methods described in Sections 4.2.2 and 4.2.3.

4.2.2 Entropy coding via Rice coding

In Chapter 2 it was described that the resulting binary code from a Rice coder is obtained by concatenating the results of equations (4.36), (4.37)and (4.38) with a separating binary symbol ‘1’ between the results of (4.37)and (4.38). Equation (4.39)

governs the exact length of the result from (4.37). Rice coding is suitable for coding a geometric distribution of symbols, which is generally the distribution shape of an LPC residual signal, as shown in [20] and discussed in Chapter 2.

$$S_i = \begin{cases} 0\ , & I_i \geq 0 \\ 1\ , & I_i < 0 \end{cases} \tag{4.36}$$

$$B_i = binary[\ \mathrm{mod}(\ |I_i|\ ,\ 2^k)\ ] \tag{4.37}$$

$$U_i = unary\left\lfloor \frac{|I_i|}{2^k} \right\rfloor \tag{4.38}$$

$$k = \log_2(\bar{s}) \tag{4.39}$$

The Rice coding parameter is adapted per frame of the encoded signal in the form of the *k-value*, which is the only parameter needed to decode the Rice coded sequence correctly. The maximum size of the side information can then be calculated by assuming that only signed 16 bit integers are to be coded. Hence, the maximum k-value is then $\log_2(2^{15}) = 15$ and can be represented with 4 bits per frame.

4.2.3 Entropy coding via cascade coding

4.2.3.1 Cascade coding methodology

In Chapter 2, it was described that cascade coding is a multi-stage binary coder, with the output of the n^{th} stage shown in equations (4.41) and the value passed onto the next stage shown in equation (4.40). Cascade coding is suitable for coding a sequence of symbols having a stepwise distribution and its advantage is its low complexity by operating mainly by using addition and subtraction [24].

$$i_{n+1} = i_n - \left(2^{b_n} - 1\right) \tag{4.40}$$

$$o_n = \begin{cases} 0, & i_n > 2^{b_n} - 1 \\ i_n, & i_n \leq 2^{b_n} - 1 \end{cases} \tag{4.41}$$

4.2.3.2 Frame adaptive cascade coding

According to [24], a curve-fitting technique could be used to select the optimal cascade coding bit allocation per stage. However, curve-fitting as proposed in [24] will only yield the optimal performance for a stationary signal, i.e. the statistics of the signal are unchanging throughout the whole data stream. While this assumption might hold true for some signals, it is not the case in audio signals where the statistics of the signal might change over a short duration. Hence, for maximum coding efficiency, adapting the cascade bit allocation per frame is required.

File no.	Without frame adaptive	With frame adaptive
4	10.0	6.3
14	7.0	4.7
16	7.0	5.3
Average	**8.0**	**5.4**

Table 4.1. Bits/sample comparison of cascade coding with and without frame adaptive method.

Table 4.1 shows the benefit of the frame adaptive cascade coding method by comparing compression results for three signals comprising of 430 frames (440320 samples) with a cascade coder that is using the frame adaptive method and cascade coder that is adapting to the whole signal by calculating the dynamic range of the entire signal and choosing the cascade coder bit allocation based on that dynamic range. For example, by referring to Table 4.2, if the maximum magnitude of integer in a given frame is 7, then a cascade coder bit allocation that is able to perform encoding of that dynamic range is then chosen to encode the current frame (which is entry #2 in Table 4.2). It is evident that using the frame adaptive cascade coding method yields a significantly higher compression rate, with 5.4 bits/sample on average compared to 8.0 bits/sample without the frame adaptive method.

4.2.3.3 Codebook based cascade coding

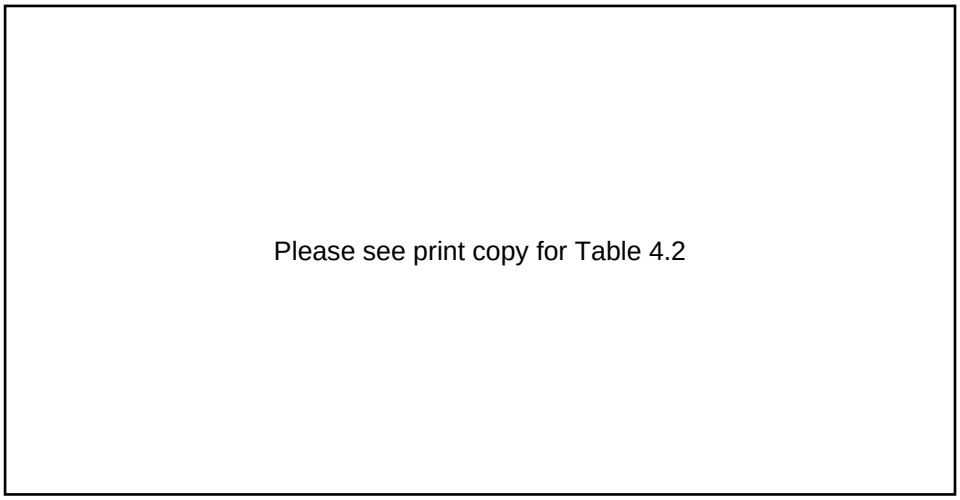
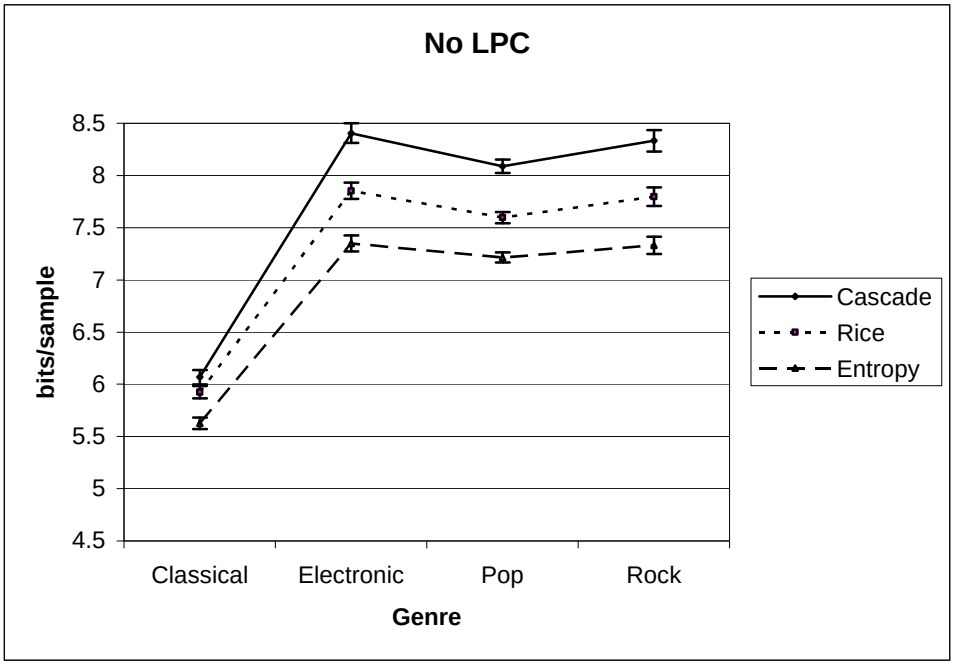


Table 4.2. The 15 entry codebook for cascade coding, based on [24].

Although adapting the cascade bit allocation is required in order to obtain optimal efficiency, implementing a curve-fitting method for each frame might defeat the advantage of cascade coding in regards to low complexity since the curve-fitting method described in Section 4.2.3.2 has to be performed for every frame.

Reference [24] details the optimal bit allocation for a cascade coder given Gaussian distributed random integer sequences with various magnitudes. Motivated by this work, a codebook based approach is proposed in this thesis, where in this approach each frame is coded using one of the entries in the codebook. The codebook entry selection is then based on the dynamic range of the input frame. Such approach would provide frame adaptation without the complexity required to compute a new bit allocation for each frame during encoding and simplify the implementation of the encoder and decoder.



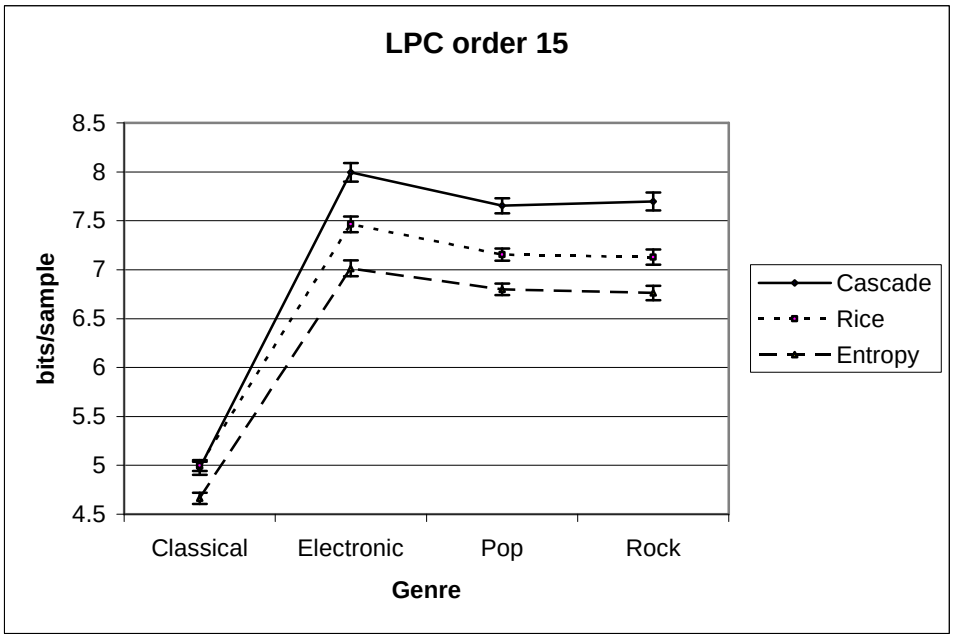


Figure 4.6. 95% confidence interval plot for cascade coding, Rice coding and entropy values, using no LPC and LPC order 15 on the residual signal.

The codebook entries consist of the description of the bit allocation and the maximum integer magnitude that is possible to be coded given the bit allocation. This table was presented in [24] and is shown in Table 4.2 and generated using the exhaustive search optimization method described in [11] for a Gaussian distributed

integer sequence. The codebook entry to be used for a given input signal is then chosen based on the maximum integer magnitude present in the signal. This is because although cascade coding is inherently a variable length signal, there is a limitation on the integer magnitude that can be coded, as the number of cascade stages is not infinite. The maximum magnitude to be coded is the sum of the magnitude that can be wholly coded by each stage, by taking into account that the first stage includes the sign bit. For example, if the bit allocation is {3 3 2}, then the overall magnitude is $(2^2-1) + (2^3-1) + (2^2-1) = 3 + 7 + 3 = 13$. Therefore, the allocation {3 3 2} can code signed integer sequences ranging from -13 to 13.

By implementing a codebook approach in cascade coding, the side information required to signal the bit allocation to the decoder is then simply the codebook entry number. In contrast, without using a codebook approach, side information is required to describe all the bit allocations for all the stages, resulting in large side information with variable length, since cascade coding by design can use any number of stages.

The exact length of the side information required to be sent to the receiver depends upon the size of the codebook used. In the current implementation of a 15 entry codebook, the side information size is 4 bits.

4.2.3.4 Codebook base cascade compression results and discussion

As a study on how cascade coding performs compared to another entropy coding method, notably Rice coding, the bit allocations in [24] were designated as the cascade coding codebook in this work, and is shown in Table 4.2.

Note that the codebook shown in Table 4.2 is not optimal due to the limitation of magnitude. The maximum possible signal magnitude to be coded with this codebook is 602, whereas the maximum possible magnitude of a 16 bit audio signal is 32768. The results in Table 4.3 provides a direct comparison between the performance of cascade coding and Rice coding by compressing only the frames that are applicable to cascade coding and is used as an indication of the performance of codebook based cascade coding.

Genre	File no.	No LPC		LPC order 15	
		Cacsade	Rice	Cascade	Rice
	7	7.5	7.1	5.5	5.4
	14	4.7	4.8	4.3	4.5
Classical	15	6.9	6.6	6.3	6.1
	16	5.3	5.3	3.2	3.6
	17	6.1	6.0	5.5	5.4
Average		**6.1**	**5.9**	**5.0**	**5.0**
95% confidence		**±0.07**	**±0.06**	**±0.07**	**±0.06**
	1	6.6	6.2	6.2	5.9
	3	9.5	8.8	9.3	8.6
Electronic	6	9.1	8.4	8.6	8.0
	11	7.9	7.4	7.5	7.0
	19	9.5	8.8	9.1	8.5
Average		**8.5**	**7.9**	**8.1**	**7.6**
95% confidence		**±0.09**	**±0.08**	**±0.09**	**±0.08**
	8	8.1	7.6	7.4	6.9
	9	9.1	8.4	8.9	8.1
Pop	12	6.5	6.2	5.7	5.5
	13	8.5	8.0	8.4	7.8
	18	8.7	8.1	8.4	7.8
Average		**8.2**	**7.7**	**7.8**	**7.2**
95% confidence		**±0.06**	**±0.05**	**±0.08**	**±0.06**
	2	9.3	8.6	9.1	8.2
	4	6.3	5.9	5.9	5.5
Rock	5	9.4	8.7	8.4	8.0
	10	9.3	8.7	8.1	7.5
	20	9.5	8.9	8.5	7.9
Average		**8.8**	**8.2**	**8.0**	**7.4**
95% confidence		**±0.1**	**±0.09**	**±0.09**	**±0.08**
Overall average		**7.9**	**7.4**	**7.2**	**6.8**

Table 4.3. Results of cascade and Rice coding with the entropy of the AAC residual signal, with and without LPC . Figures are in bits/sample.

The results presented in Table 4.3 show that LPC processing appears to offer an overall benefit for multiple genres tested. The entropy of the AAC residual signal are reduced by approximately 8%, on average, after LPC processing. This decrease is entropy rate is also reflected by the coding rate of cascade and Rice coding, which also decreases by about 8% if LPC processing of order 15 is utilized.

From the results in Table 4.3, it is evident that LPC processing is most beneficial for the genres of classical and rock music, where the reduction in the entropy rates are 0.9 and 0.7 bits/sample for respective genre. The entropy reduction due to LPC processing of order 15 for the genres of electronic and pop are 0.3 and 0.4 bits/sample, respectively. Note that there are no statistical difference between coding with the cascade coding method or the Rice coding method in the classical music genre.

Statistical results of 95% confidence intervals for each genre as shown in Table 4.3 show that the statistical significance of the results are within one decimal place.

4.3 Cascade Codebook Design and Testing

In order to optimize cascade coding for audio residual signals investigated in this work, a new bit allocation codebook may be needed, since the codebook in Section 4.2.3.3 is optimized for Gaussian signals and not for residual signal resulting from the LPC process described in Section 4.1. This codebook was trained and tested using the techniques described below.

4.3.1 Cascade coding codebook training algorithm

An iterative search method based on systematic reduction of all bit allocations possible for a given input signal is described in [24]. A similar algorithm is adopted here for codebook training and is described as follows:

1. Set the initial allocation as {2 1 1 1…}. The number of one bit stages must be sufficient to encode the maximum magnitude of the input signal. Note that while the first stage is initially {2}, it is actually {1} since the first stage of a cascade coder also includes the sign bit.

2. Calculate the resulting compression rate using the bit allocation specified in step 1 for the entire training data.

3. Increase the bit allocation by 1 until ceil($\log_2$(K+1)) where K is the maximum integer magnitude in the input signal and compute the compression result for each allocation.

4. Find the two lowest compression rate results from step 3. For the corresponding set of allocations, repeat the algorithm from step 2 and optimizing for the next stage.

5. Continue until all possible combinations of bit allocations and number of stages for the given input signal are tested.

This algorithm is applied separately to train each codebook entry, corresponding to signals with different dynamic ranges. The approach adopted here differs to that described in [11, 24] in that it measures the resulting compression rate for the training data at step 2 rather than estimating it based on signal statistics.

Codebook entry	Magnitude range	Number of frames in the Training database
1	$s \leq 8$	505
2	$8 < s \leq 16$	904
3	$16 < s \leq 32$	728
4	$32 < s \leq 64$	836
5	$64 < s \leq 128$	1091
6	$128 < s \leq 256$	1323
7	$256 < s \leq 512$	1416
8	$s > 512$	1774

Table 4.4. Proposed cascade coder eight-entries codebook.

The codebook entries are limited to 8 entries instead of 16 to simplify the computational process and to determine whether the performance of an 8 entry codebook is comparable to the performance of a 16 entry codebook used in section 4.2.3.3. This new proposed codebook is shown in Table 4.4.

Note that the new codebook increases by powers of 2 for each entry instead of an arbitrary increase in magnitude as described in [24]. This is done to ensure that low magnitude values are coded with finer granularity compared to higher magnitude values, since low magnitude values in audio signals and LPC residual signals occur more frequently than high magnitude values.

4.3.2 Training database

A *Training database* was formed from the database used for LPC processing analysis described in Section 4.1 and was divided into frames of length 1024 samples. The frames are categorized according to the codebook in Table 4.4.

To facilitate training, the Training database is divided into two unequal parts: the training portion which comprised of 80% of the total number of frames in a given codebook entry, and the initial testing portion which comprised of the remaining 20%.

The test portion in the Training database is used to analyze the performance of the cascade coder bit allocation trained for each codebook entry.

4.3.3 Cascade coder codebook training results

Entry	Bit Allocation
1	[2 1 1 1 1 1 1 1 1]
2	[3 2 1 1 1 1 1 1 1 1 1 1 1]
3	[4 3 3 2 2 1 1 1 1 1 1]
4	[6 4 3 3 1 1 1 1 1]
5	[7 5 4 4 1 1 1 1 1]
6	[8 6 6 1 1 1 1]
7	[9 8 1 1 1]
8	[10 9 1 1 1 15]

Table 4.5. Result of the new proposed codebook training and testing.

	Code Entry	Dyn. Range	entropy (bits/sample)	compression (bits/sample)	efficiency (%)	n (samples)
Training portion	1	8	3	3.1	96.8	413696
	2	16	4	4.1	97.6	740556
	3	32	4.8	5	96	596377
	4	64	5.8	6.1	95.1	684851
	5	128	6.7	7.1	94.4	893747
	6	256	7.7	8.1	95.1	1083801
	7	512	8.6	9.1	94.5	1159987
	8	1024	9.5	10.1	94.1	799539
	8	2048	10	10.8	92.6	1223884
	8	32768	10.4	14	74.3	1453260
Testing portion	1	8	3.1	3.2	96.9	103424
	2	16	3.9	4.1	95.1	185140
	3	32	4.7	4.9	95.9	149095
	4	64	5.7	6.1	93.4	171213
	5	128	7	7.3	95.9	223437
	6	256	7.8	8.2	95.1	270951
	7	512	8.7	9.2	94.6	289997
	8	1024	9.4	10.2	92.2	199885
	8	2048	9.5	10.5	90.5	305972
	8	32768	9.5	14	67.9	363316

Table 4.6. Result of the codebook entry 8 training using the test portion of the Training database.

The overall training results are shown in Table 4.5 as the bit allocation per codebook entry. Table 4.6 lists the resulting entropy, compression rate and number of samples per

codebook entry, for both the training and initial test portions of the Training database. It is evident in Table 4.6 that the efficiency of the trained codebook is quite high, averaging 95%. The particular result of codebook entry 8 in Table 4.6 also shows that the training strategy detailed in Section 4.3.3.1 results in good performance over the samples in the testing portion of Training database.

4.3.3.1 Codebook entry 8 training

Since codebook entry 8 in Table 4.4 includes high magnitude values beyond 512 up to the maximum value able to be represented by a 16 bit integer (32768), training the codebook entry 8 by itself might not yield the best encoded bits/sample since the bit allocation has to accommodate a high dynamic range of signals. The high dynamic range of frames in the codebook entry 8 category is apparent in the histogram plot of the samples involved in the frames of codebook entry 8 as shown in Figure 4.7.

The histogram plot of codebook entry 8 sample values in Figure 4.7 shows that the majority of samples in the frames that are included in codebook 8 category are less than 1000. In this case, values of more than 1000 account for approximately 3% of the total number of samples and can be considered as outliers.

In order to investigate this concern, frames with magnitudes of the two next power of 2 from codebook 7, which are 1024 and 2048, are also trained. The resulting bit allocation from both training will then be modified with an allocation of 15 bits as the last cascade stage. This two separate training sessions essentially trains a cascade coder which concentrates on magnitudes up to 1024 and 2048, and uses a stage of 15 bits to encode values larger than their respective coding limits.

The motivation for this training approach is to prevent high magnitude outliers with low occurrence from influencing the training as a whole thus lowering the overall efficiency of the trained bit allocation. By removing the outliers from consideration, the training algorithm can focus on the high occurrence values with lower magnitudes instead.

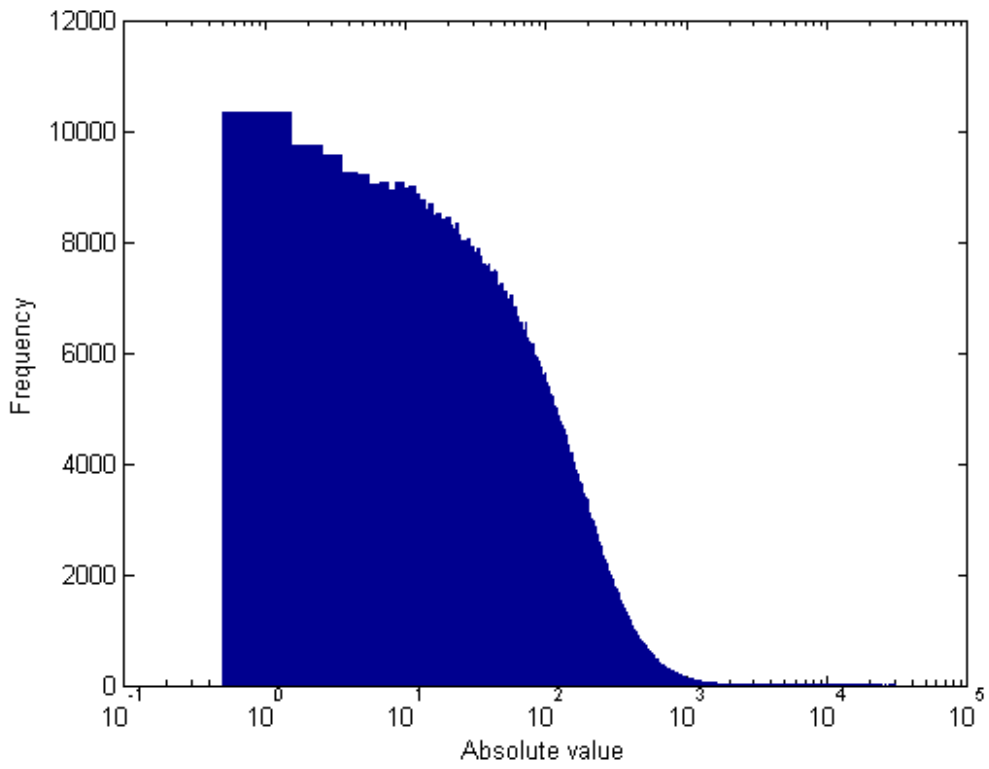


Figure 4.7. Histogram plot of the absolute values of samples in the codebook 8 frames.

The result of codebook entry 8 training is shown in Table 4.6, where the resulting bit allocation is tested using the test portion of the Training database. In Table 4.6 it is shown that limiting the training to magnitude 32768 results in 14 bits/sample compression rate using the test portion, which suggests that the resulting bit allocation is sub-optimal. By limiting the training to magnitudes of 1024 and 2048, significant performance improvement is gained, where the performance of limiting the training to magnitude 1024 shows a better performance over magnitude 2048. Hence, based on the results shown in Table 4.6, the codebook entry 8 will subsequently use the bit allocation obtained by training on samples with a maximum magnitude of 1024 and an extra 15 bit stage to code any samples beyond this range.

4.3.4 Test database coding result

Genre	File id	Cascade		Rice		MAC bits / sample
		bits / sample	confidence interval	bits / sample	confidence interval	
	A	6.6	0.06	6.4	0.05	6.1
Classical	B	6.5	0.1	6.3	0.09	6.2
	C	6.1	0.16	5.9	0.14	5.8
Average		**6.4**	**0.07**	**6.2**	**0.06**	**5.8**
	D	8.5	0.17	7.9	0.13	7.7
Electronic	E	8.9	0.08	8.5	0.08	8.4
	F	8.1	0.07	7.7	0.06	7.6
Average		**8.5**	**0.07**	**8**	**0.06**	**7.6**
	G	6.7	0.14	6.4	0.12	6.2
Pop	H	9.9	0.15	9.1	0.11	8.9
	I	9.6	0.12	9.1	0.1	9
Average		**8.7**	**0.11**	**8.2**	**0.09**	**9**
	J	7.7	0.21	7.2	0.19	7
Rock	K	9.3	0.19	8.7	0.12	8.6
	L	7.2	0.11	6.9	0.09	6.7
Average		**8.1**	**0.11**	**7.6**	**0.09**	**6.7**
Average		**7.9**	**0.05**	**7.5**	**0.04**	**6.7**

Table 4.7. Test database coding result, arranged by signal.

Codebook entry	Cascade		Rice		Total samples
	bits/sample	confidence interval	bits/sample	confidence interval	
1	2.8	0.1	3.3	0.08	21504
2	4.3	0.1	4.4	0.07	67584
3	5.1	0.03	5	0.02	590848
4	6.2	0.01	6	0.02	659456
5	7.1	0.01	6.8	0.02	940032
6	8.1	0.01	7.7	0.02	1193984
7	9.1	0.01	8.6	0.03	904192
8	10.8	0.1	9.9	0.05	892928
	7.9	**0.05**	**7.5**	**0.04**	**5270528**

Table 4.8. Test database coding result, arranged by codebook entry.

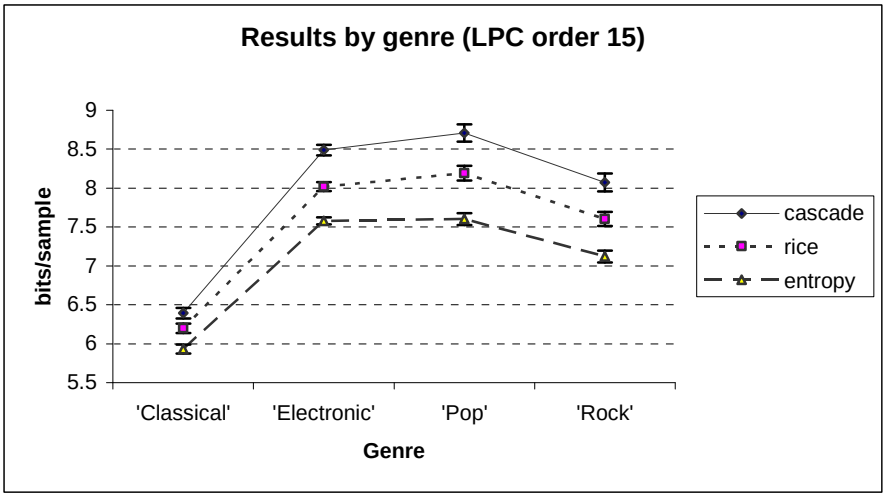


Figure 4.8. 95% confidence interval plot of test database coding. Based on the data shown in Table 4.7.

4.4 Cascade Codebook Performance and Discussion

To analyse the overall compression performance of the trained codebook based cascade coder a test database was created using samples that were separate to the Training database. The samples were obtained from LPC residuals derived using the techniques outlined in Section 4.1 in a similar way to the signals generated for the Training database. This database was used to perform an evaluation on the performance of the trained codebook-based cascade coder applied to different music genres.

Table 4.7 and Table 4.8 show the results for compressing the test database using the trained codebooks described in Section 4.3, where the results are arranged in codebook entries and arranged by files, respectively. The plot of the results is shown in Figure 4.8. In Table 4.7 it is shown that the overall average cascade coding compression rate is 7.9 bits/sample, with ±0.05 bits/sample in 95% confidence interval. Note that the compression performance is not equal across genres, where the classical genre exhibits ≈21%-26% lower compression rate compared to other genres. The results of Rice coding shows ≈0.4 bits/sample lower compression rate compared to cascade coding, consistent across all genres tested, although the compression rate of the classical genre compared to other genres has a slightly higher percentage point (≈18%-23%) from the cascade coding results. The 95% confidence interval for Rice coding is however slightly smaller compared to cascade coding (0.04 bits/sample for Rice compared to 0.05

bits/sample for cascade). The compression rate of Monkey's Audio is the best overall, with ≈1.2 bits/sample difference to cascade coding and ≈0.8 bits/sample difference to Rice coding. Since the Monkey's Audio coder was operated directly on the AAC residual signal, it is reasonable to assume that the decorrelation step used by Monkey's Audio more effectively lowers the entropy of the AAC residual signal compared to backward LPC process. However, the higher compression rate of Monkey's Audio in the Pop genre compared to both cascade and Rice coding suggests that the performance of Monkey's Audio decorrelation step is not consistent. This can also be examined from the inconsistent difference between the compression rate of Monkey's Audio and Rice coding, where the difference varies between 0.9 bits/sample lower for Monkey's Audio to 0.8 bits/sample higher for Monkey's Audio compared to Rice coding.

In Table 4.8, comparing cascade coding and Rice coding results based on the cascade coder codebook entries indirectly provides a measure of performance of both coders for various frame magnitudes. Note that for codebook entries 1 and 2 (low magnitude values), the performance of cascade coding is better than Rice coding, where cascade coding performs slightly worse on codebook entry 3 and subsequent codebook entries. Table 4.8 shows that for low magnitude frames, cascade coding consistently performs better than Rice coding.

It is clear from the results that the cascade coder performance is relatively unchanged relative to the entropy of the signal and the compression rate results of Rice coder, as shown in the Gaussian codebook results in Table 4.3 and the new audio specific codebook in Table 4.7. Comparing average results from both codebooks shows that Rice coding compression rate is consistently 0.4 bits/sample lower compared to cascade coding, even if the codebook used in the cascade coder different. Note that the results of Table 4.3 and Table 4.7 cannot be compared directly. This is because the higher compression rates in Table 4.7 are the result of coding integer magnitudes up to 32768 (which is the limit of a 16 bit integer), whereas the results in Table 4.3 is the result of coding up to magnitude 602 (which is the limit of an approximately 9.2 bit integer). It is interesting to observe that the new audio specific codebook can maintain a comparable performance to the Gaussian codebook even though the magnitude codeable by the Gaussian codebook is less than 2% of the magnitude codeable by the new

codebook, and the new codebook could achieve this comparable performance using only 8 codebook entries as opposed to 15 entries used in the Gaussian codebook.

The consistently higher bit rate of cascade coding could be attributed to the training algorithm outlined in Section 4.3.1 due to the fact that the codebooks in Table 4.2 and Table 4.4 are generated using the same algorithm, albeit with different probability density functions for the training signal.

An explanation of the poorer performance of cascade coding when applied to genres other than classical could be explained by referring to the entropy of the resulting signals. For classical signals, the entropy is much lower than for the other genres. In [24], the performance (in terms of compression rate) of the designed cascade coders (which are used here) significantly decrease as the entropy of the samples increase. Hence, the designed cascade coders are suboptimal for higher entropy signals.

From Table 4.3, the compression rate of Monkey's Audio (MAC) is the lowest of all the methods tested, although the entropy of the decorrelated AAC residual signal is below the Monkey's Audio compression rate. Without using backward LPC however, the entropy of the AAC residual is higher than the bits/sample result of Monkey's Audio. Table 4.3 shows that using 15^{th} order backward LPC results in consistently ≈8% lower coding rate compared to no LPC processing. Although on average using Monkey's Audio results in lower coding rate, the inconsistent results in Table 4.7 suggests that using Monkey's Audio decorrelation technique on an AAC residual signal which it was not designed for could potentially result in data expansion, although the parameters for the Monkey's Audio decorrelation technique performances is beyond the scope of this thesis and provided only for comparative purposes only.

The similar compression results and 95% confidence intervals of the Gaussian optimized codebook in Table 4.3 and the audio signal trained codebook in Table 4.7 show the current performance limit of cascade coding, where in Table 4.8 it is shown that cascade coding outperforms Rice coding for low dynamic range, low entropy signals. This similar result is encouraging, noting that the Gaussian codebook is a 16 entries codebook, whereas the audio optimized codebook contains only 8 entries.

The performance of cascade coding compared with Rice coding is limited when the signals to be coded are high dynamic range, high entropy signals. The training algorithm does not appear to be optimal for designing a cascade coder for encoding such signals, thus new training methods or algorithms need to be investigated.

4.5 Chapter summary

This chapter has discussed the lossless compression benefit of performing backward adaptive LPC on the residual of an AAC signal. While performing forward adaptive LPC may have the benefit of higher prediction gain, the need to send overhead information to the receiver may limit the prediction order due to the increasing size of the subsequent overhead information. Through testing, it was found that backward adaptive LPC of order 15 ensures that the prediction gain is maximum for all musical genres in the test database. It was also found that an order 15 backward LPC analysis results in an LPC residual signal probability distribution where there is a higher probability of low magnitude symbols compared to the AAC residual signal. Overall, backward adaptive LPC resulted in ≈8% decrease in coding rate compared to the coding rate without LPC processing. This decrease is consistent using both cascade and Rice coders.

This chapter has compared the compression performance of Rice coding, and cascade coding, with and without LPC processing. It was found that performing LPC processing on the AAC residual signal using order 15 generally decreases the entropy of the signal and also the coding rate for both coders by 8%, which suggests that the decrease is consistent between any entropy coder.

Finally, this chapter proposed a cascade coding codebook based on training using audio signals generated from the residual of an LPC processed AAC residual signal. The performance of this new codebook is then compared to that of the Gaussian codebook proposed in [24]. Section 4.3 shows that although the new codebook is smaller (8 entries compared to 15 entries), the performance of the new codebook is similar or better to that of the Gaussian codebook.

It was also noted that the cascade coding codebook utilized in this work is sub-optimal for coding high entropy signals, hence the lagging performance of cascade coding

compared to Rice coding by 0.4 bits/sample on average. However, the performance of cascade coding is better compared to that of Rice coding for low entropy signals.

5 Conclusion

Embedded lossless coding aims to embed a lossless enhancement layer on top of a perceptual layer, and thus achieves scalability in the bitstream by providing two-step scalability: perceptually near-lossless signal and bit-by-bit lossless signal. As opposed to the fine grain scalability approach, the embedded lossless approach could use an existing standard audio coder for the perceptual layer and a suitable entropy coding technique for the lossless enhancement layer.

Chapters 1 and 2 have provided the introduction and review of audio coding techniques relevant to the work performed in this thesis. Chapter 2 specifically has provided background information on information theory, perceptual audio coding techniques and lossless audio coding techniques.

Chapter 3 has tested four perceptual audio coders to identify which coder and which method is suitable for the perceptual base layer in the embedded lossless coder. It was found that the MPEG-4 AAC perceptual coder implemented by Nero AG [42] operating at 96 kbps mono achieved the best signal-to-noise ratio of the residual signal compared to the original signal for a given lossless compression rate. It was also found that operating an embedded paradigm by using the Nero AAC coder at 96 kbps with the residual compressed using Monkey's Audio results in only an approximate 6% increase in the bitrate compared to the original signal when compressed using Monkey's Audio only.

Chapter 4 has shown that the residual of a perceptual base layer, particularly the residual of the Nero AAC coder, still contains correlation between samples. In order to facilitate the entropy coding stage of the lossless enhancement layer, decorrelation is thus necessary. In this chapter it was found that operating a 15^{th} order backward linear prediction process on the AAC residual signal results in 8% decrease of coding rate for

all musical genres tested. Also, it was found that operating backward linear prediction results in a lower entropy signal, thus an easier to code signal. Backward linear prediction has the advantage of requiring no side information to perform the reverse process. A rounded backward linear prediction was used in this thesis to ensure that the reverse LPC process is lossless.

Chapter 4 also provided a comparison of performance between two entropy coding methods: Rice coding [19] and cascade coding [24]. Rice coding is the de-facto standard for compressing integer sequences with a Laplacian probability distribution function (p.d.f.) and is used in the state of the art lossless audio compression techniques such as [20-22, 46]. Two modifications were made to cascade coding in order to make it more suitable to perform entropy coding for audio signals: frame adaptive cascade coding and codebook-based cascade coding. Furthermore, a new cascade coder designed specifically for encoding audio signals was proposed. It was found that although the performance of a frame-adaptive codebook-based cascade coder lagged 0.4 bits/sample compared to Rice coding, cascade coding shows promising results as the proposed audio specific coder design performs better than Rice coding for low magnitude frames. These results, and the low complexity advantage show that cascade coding is an attractive entropy coder for use in lossless audio coding.

5.1 Limitations of techniques

Rice coding is a highly popular and efficient entropy coding method for lossless audio coding, and the cascade coding technique shows promising results as described in Chapter 4. However, the most obvious limitation of these two coding schemes is that both are Huffman codes that require an integer number of binary symbols per input symbol. This seriously limits the performance of both methods if the signal to be coded involves a very high probability symbol, as described in Chapter 2.

5.2 Future work

For the decorrelation step, it is worthwhile investigating the performance of nonlinear prediction when applied to an AAC residual signal, as is evident in the performance of Monkey's Audio detailed in Chapter 3 (which uses a nonlinear predictor

as described in Chapter 2). The 8% decrease of coding rate achieved in this work could potentially be increased.

For the entropy coding step, applying arithmetic coding might improve the entropy coding performance in the case of a high probability symbol and enables the entropy coder to achieve fine grain scalability in the bitstream. This would essentially perform bit plane coding, but instead of using a bitplane method on the arithmetically coded frame as in the BSAC method, cascade coding is used instead. Fine grain scalability could be potentially achieved by arranging the cascade coding stages according to the bit plane encoding paradigm. This will effectively minimize the limitation of cascade coding which requires an integer number of binary symbol and allows partial decoding of the bitstream.

For the cascade coding technique, a new codebook specifically designed for use with audio signals (besides the one proposed in this thesis) and an improved training algorithm are required for further improvement in compression performance. This codebook could be designed by applying the curve-fitting method described in [24] on a large database of audio signals. Another possible codebook generation technique might involve the use of a genetic algorithm to identify the optimal bit allocation. Increasing the size of the codebook to accommodate more signal statistics could also be explored, obviously with the increase of side information needing to be taken into account. From the structure of cascade coding, it is possible to modify the technique to allow for partial decoding to achieve scalability. For example, a bitstream can have only the first stage decoded to show the general features of the signal, with each additional stage adding the detail.

References

[1] K. C. Pohlmann, *The compact disc : a handbook of theory and use / Ken C. Pohlmann*. Madison, Wis. :: A-R Editions,, c1989.

[2] J. Herre and H. Purnhagen, "General audio coding," in *The MPEG-4 Book*, F. Pereira and T. Ebrahimi, Eds.: IMSC Press, 2003, pp. 487-544.

[3] K. C. Pohlmann, *Principles of digital audio / Ken C. Pohlmann*, 4th ed ed. New York ; London :: McGraw-Hill,, c2000.

[4] M. Bosi, *Introduction to digital audio coding and standards / Marina Bosi, Richard E. Goldberg*. Boston :: Kluwer Academic Publishers,, c2003.

[5] M. Raad, A. Mertins, and I. Burnett, "Scalable to lossless audio compression based on perceptual set partitioning in hierarchical trees (PSPIHT)," presented at Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on, 2003.

[6] R. Geiger, A. Herre, G. Schuller, and T. Sporer, "Fine grain scalable perceptual and lossless audio coding based on IntMDCT," presented at Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on, 2003.

[7] R. Yu, X. Lin, S. Rahardja, and C. C. Ko, "A fine granular scalable perceptually lossy and lossless audio codec," presented at Multimedia and Expo, 2003. ICME '03. Proceedings. 2003 International Conference on, 2003.

[8] T. Moriya, N. Iwakami, A. Jin, and T. Mori, "A design of lossy and lossless scalable audio coding," presented at Acoustics, Speech, and Signal Processing, 2000. Proceedings. (ICASSP '00). 2000 IEEE International Conference on, 2000.

[9] R. Geiger, J. Herre, J. Koller, and K. Brandenburg, "IntMDCT - a link between perceptual and lossless audio coding," presented at Acoustics, Speech, and Signal Processing, 2002. Proceedings. (ICASSP '02). IEEE International Conference on, 2002.

[10] M. Raad, *Scalable and perceptual audio compression*, PhD thesis, University of Wollongong,2002

[11] L. Wu, A. Zielinski, and J. S. Bird, "Lossless compression of hydroacoustic image data," *Oceanic Engineering, IEEE Journal of*, vol. 22, pp. 93-101, 1997.

[12] K. Adistambha, C. H. Ritz, J. Lukasiak, and I. S. Burnett, "An investigation into embedded audio coding using an AAC perceptually lossless base layer," presented at Speech Science and Technology Conference 2004, Sydney, Australia, 2004.

[13] K. Adistambha, C. H. Ritz, and J. Lukasiak, "Embedded lossless audio coding using linear prediction and cascade coding," in *Submitted to the 8th International Symposium on DSP and Communication Systems (DSPCS 2005)*. Australia, 2005.

[14] K. Sayood, *Introduction to data compression*, 2nd ed: Morgan Kaufmann Publishers, 2000.

[15] J. D. Gibson, *Digital compression for multimedia : principles and standards*. San Francisco, Calif.: Morgan Kaufmann Publishers, 1998.

[16] The gzip home page, available: http://www.gzip.org, Accessed: 3 December 2005, Last updated: 27 July 2003.

[17] Winzip - the zip file utility for windows, available: http://www.winzip.com, Accessed: 3 December 2005, Last updated: 2 November 2005.

[18] M. Hans and R. W. Schafer, "Lossless compression of digital audio," *Signal Processing Magazine, IEEE*, vol. 18, pp. 21-32, 2001.

[19] R. F. Rice, "Lossless coding standards for space data systems," presented at Signals, Systems and Computers, 1996. 1996 Conference Record of the Thirtieth Asilomar Conference on, 1996.

[20] T. Robinson, "SHORTEN: simple lossless and near-lossless waveform compression," Cambridge University Engineering Department CUED/F-INFENG/TR.156, 1994.

[21] Free Lossless Audio Codec, available: http://flac.sourceforge.net, Accessed: 3 December 2005, Last updated: 3 November 2005.

[22] Monkey's Audio: A fast and powerful lossless audio compressor, available: http://www.monkeysaudio.com, Accessed: 3 December 2005, Last updated: 29 April 2004.

[23] T. Liebchen, Y. A. Reznik, T. Moriya, and D. T. Yang, "MPEG-4 audio lossless coding," presented at 116th AES Convention, Berlin, Germany, 2004.

[24] L. Wu, A. Zielinski, and J. S. Bird, "A cascade coding technique for compressing large-dynamic-range integer sequences," in *Digital Signal Processing*, vol. 14: Elsevier, 2004, pp. 54-71.

[25] R. Yu, X. Lin, S. Rahardja, and C. C. Ko, "A scalable lossy to lossless audio coder for MPEG-4 lossless audio coding," presented at Acoustics, Speech, and

Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on, 2004.

[26] R. Yu, C. C. Ko, S. Rahardja, and X. Lin, "Bit-plane Golomb coding for sources with Laplacian distributions," presented at Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on, 2003.

[27] S.-H. Park, Y.-B. Kim, S.-W. Kim, and Y.-S. Seo, "Multi-layer bit-sliced bit rate scalable audio coding," presented at 103rd AES Convention, New York, 1997.

[28] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]*, vol. 41, pp. 3445-3462, 1993.

[29] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 6, pp. 243-250, 1996.

[30] T.-H. Lan and A. H. Tewfik, "Multigrid embedding (MGE) image coding," presented at Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on, 1999.

[31] D. Taubman, "High performance scalable image compression with EBCOT," *Image Processing, IEEE Transactions on*, vol. 9, pp. 1158-1170, 2000.

[32] J. Yan, Z. Dong, and W. Dou, "A scalable digital audio encoder based on embedded zerotree wavelet," presented at Communication Technology Proceedings, 2003. ICCT 2003. International Conference on, 2003.

[33] Z. Wang and F. Lin, "A new and efficient audio compression based on EBCOT and RVLC," presented at Signal Processing, 2002 6th International Conference on, 2002.

[34] J.-L. Garcia, P. Gournay, and R. Lefebvre, "Backward linear prediction for lossless coding of stereo audio," presented at 116th AES Convention, Berlin, Germany, 2004.

[35] J. D. Markel, 1943-, *Linear prediction of speech / J. D. Markel, A. H. Gray, Jr*. Berlin ; New York :: Springer-Verlag,, 1976.

[36] J. H. Chen, "Low-delay coding of speech," in *Speech Coding and Synthesis*, W. B. Kleijn and K. K. Paliwal, Eds.: Elsevier, 1995, pp. 209-256.

[37] G. Kubin, "Nonlinear processing of speech," in *Speech Coding and Synthesis*, W. B. Kleijn and K. K. Paliwal, Eds.: Elsevier, 1995, pp. 557-610.

[38] T. Sporer, K. Brandenburg, and B. Edler, "The use of multirate filter banks for coding of high quality digital audio," presented at 6th European signal processing conference (EUSIPCO), Amsterdam, 1992.

[39] T. Painter and A. Spanias, "Perceptual coding of digital audio," *Proceedings of the IEEE*, vol. 88, pp. 451-515, 2000.

[40] Vorbis.com - Open, Free Audio, available: http://www.vorbis.com/, Accessed: 3 December 2005, Last updated: 2 December 2005.

[41] Q-Music royalty free music, available: http://www.q-music.co.uk, Accessed: 3 December 2005, Last updated: 2 December 2005.

[42] Nero AG homepage, available: http://www.nero.com, Accessed: 3 December 2005, Last updated: 2 December 2005.

[43] PsyTel AAC encoder, available: http://www.rarewares.org/aac.html, Accessed: 3 December 2005, Last updated: 2 March 2002.

[44] LAME mp3 encoder, available: http://lame.sourceforge.net/, Accessed: 3 December 2005, Last updated: 16 September 2005.

[45] Y. A. Reznik, "Coding of prediction residual in MPEG-4 standard for lossless audio coding (MPEG-4 ALS)," presented at Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). 2004 IEEE International Conference on, 2004.

[46] T. Liebchen, "An introduction to MPEG-4 audio lossless coding," presented at Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on, 2004.