HIGH FIDELITY MULTICHANNEL AUDIO COMPRESSION

by

Dai Yang

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(ELECTRICAL ENGINEERING)

August 2002

Dedication

Dedicated with love to

my husband *Ruhua He*,

my son *Joshua S. He*,

my parents *Junhui Yang* and *Zongduo Dai.*

Acknowledgements

I would like to express my most profound graditude to my advisor Dr. C.-C. Jay Kuo, who has been a tireless and enthusiastic source of good ideas. I have benefited greatly from Dr. Kuo's extensive knowledge, his invaluable experience in research and his decent personality. And I will continue to benefit from this throughout my entire life. Dr. Kuo has been a strong influence in both my educational and personal pursuits and I will forever be grateful to him for sharing his successful career with me.

My sincere gratitude also goes to my co-advisor Dr. Chris Kyriakakis for his guidance and support throughout my thesis-building period. Without these, it would be impossible for me to finish my Ph.D. program.

I wish also to thank Dr. Hongmei Ai for her valuable discussions on my research, her help and encouragement with my thesis, and most importantly, the friendship that we have shared throughout my graduate study at USC.

My thanks goes out to all group members under Dr. Kuo's guidance. Post doctors and fellow students have helped me with useful discussions and entertaining conversations over years. My graduate study in this group was greatly enriched by their accompany, and it is my privilege to have spent time with them.

Finally, I would like to thank my husband Ruhua. His unconditional support and love make it possible for me to finish my program on time. I also wish to thank my parents and parents-in-law for their delight in helping taking care of my son during my research period.

Contents

List of Tables

List of Figures

Abstract

With the popularization of high-quality audio, there is an increasing demand for an effective compression and transmission technology. Despite of the success of current perceptual audio coding techniques, Some problems still remain open and need improvement. This dissertation addresses two system issues that may arise in practice in high-fidelity audio coding technology: (i) an inter-channel redundancy removal approach designed for high-quality multichannel audio compression and (ii) a progressive syntax-rich multichannel audio coding algorithm and its error-resilient codec design.

The first contribution of this dissertation is to develop a Modified Advanced Audio Coding with Karhunen-Loève Transform (MAACKLT) algorithm. In MAACKLT, we exploit the inter-channel redundancy inherent in most multichannel audio sources, and prioritize the transformed channel transmission policy. Experimental results show that, compared with MPEG AAC (Advanced Audio Coding), the MAACKLT algorithm not only reconstructs better quality of multichannel audio at a regular low bit rate of 64 kbit/s/ch but also achieves coarse-grain quality scalability.

The second contribution of this dissertation is the design of a Progressive Syntax-Rich Multichannel Audio Coding (PSMAC) system. PSMAC inherits the efficient

inter-channel de-correlation block in the MAACKLT algorithm while adding a scalable quantization coding block and a context-based QM noiseless coding block. The final bitstream generated by this multichannel audio coding system provides fine-grain scalability and three user-defined functionalities which are not available in other existing multichannel audio codecs. The reconstructed audio files generated by our proposed algorithm achieve an excellent performance in a formal subjective listening test at various bit rates. Based on the PSMAC algorithm, we extend its error-free version to an error-resilient codec by re-organizing the bitstream and modifying the noiseless coding modules. The performance of the proposed algorithm has been tested under different error patterns in WCDMA channels using several single channel audio materials. Our experimental results show that the proposed approach has excellent error resiliency at a regular user bit rate of 64 kbit/s.

Chapter 1

Introduction

1.1 Motivation and Overview

Ever since the beginning of the twentieth century, the art of sound coding, transmission, recording, mixing, and reproduction has been constantly evolving. Starting from the monophonic technology, technologies on multichannel audio have been gradually extended to include stereophonic, quadraphonic, 5.1 channels, and more. Compared with traditional mono or stereo audio, multichannel audio provides end users with a more compelling experience and becomes more and more appealing to music producers. As a result, an efficient coding scheme is needed for multichannel audio's storage and transmission, and this subject has attracted a lot of attention recently.

Among several existing multichannel audio compression algorithms, Dolby AC-3 and MPEG Advanced Audio Coding (AAC) are the two most prevalent perceptual digital audio coding systems. Dolby AC-3 is the third generation of digital audio

compression system from Dolby Laboratories, and has been adopted as the audio standard for High Definition Television (HDTV) systems. It is capable of providing indistinguishable audio quality at 384 kbit/s for 5.1 channels [A/5]. AAC is currently the most powerful multichannel audio coding algorithm in the MPEG family. It can support up to 48 audio channels and provide perceptually lossless audio at 320 kbit/s for 5.1 channels [BB97]. In general, these low bit rate multichannel audio compression algorithms not only utilize transform coding to remove statistical redundancy within each channel, but also take advantage of the human auditory system to hide lossy coding distortions.

1.1.1 Redundancy Inherent in Multichannel Audio

Despite the success of AC-3 and AAC, not much effort has been made in reducing inter-channel redundancy inherent in multichannel audio. The only technique used in AC-3 and AAC to eliminate redundancy across channels is called "Joint Coding", which consists of Intensity/Coupling and Mid/Side(M/S) stereo coding. Coupling is adopted based on the psychoacoustic evidence that, at high frequencies (above approximately 2kHz), the human auditory system localizes sound primarily based on envelopes of critical-band-filtered signals that reach human ears, rather than signals themselves [Dav93, TDD+94]. M/S stereo coding is only applied to lower frequency coefficients of Channel-Pair-Elements (CPEs). Instead of direct coding of original signals in the left and right channels, it encodes the sum and the difference of signals in two symmetric channels [BBQ+96, JF92].

Our experimental results show that high correlation is very likely to be present between every pair of channels besides CPE in all frequency regions, especially for those multichannel audio signals that are captured and recorded in a real space [YAKK00b]. Since neither AAC nor AC-3 exploits this property to reduce redundancy, none of them can efficiently compress this kind of multichannel audio content. On the other hand, if the input multichannel audio signals presented to the encoder module have little correlation between channels, the same bit rate encoding would result in higher reconstructed audio quality. Therefore, a better compression performance can be achieved if inter-channel redundancy can be effectively removed via a certain kind of transform together with redundancy removal techniques available in the existing multichannel audio coding algorithms. One possibility to reduce the cross-channel redundancy is to use inter-channel prediction [Fuc93] to improve the coding performance. However, a recent study [KJ01] argues that this kind of technique is not applicable to perceptual audio coding.

1.1.2 Quality-Scalable Single Compressed Bitstream

As the world is evolving into the information era, media compression for a pure storage purpose is far less than enough. The design of a multichannel audio codec which takes the network transmission condition into account is also important. When a multichannel audio bitstream is transmitted through a heterogeneous network to multiple end users, a quality-scalable bitstream would be much more desirable than the non-scalable one.

The quality scalability of a multichannel audio bitstream makes it possible that the entire multichannel sound can be played at various degrees of quality for end users with different receiving bandwidths. To be more precise, when a single quality-scalable bitstream is streamed to multiple users over the Internet via multicast, some lower priority packets can be dropped, and a certain portion of the bitstream can be transmitted successfully to reconstruct different quality multichannel sound according to different users' requirement or their available bandwidth. This is called the multicast streaming [WHZ00]. With non-scalable bitstreams, the server has to send different users different unicast bitstreams. This is certainly a waste of resources. Not being considered for audio delivery over heterogenous networks, the bitstream generated by most existing multichannel audio compression algorithms, such as AC-3 or AAC, is not scalable by nature.

1.1.3 Embedded Multichannel Audio Bitstream

Similar to the quality-scalable single audio bitstream mentioned in the previous section, the most distinguishable property of an embedded multichannel audio compression technique lies in its network transmission applications. In the scenario of audio coding, the embedded code contains all lower rate codes "embedded" at the beginning of the bitstream. In other words, bits are ordered in importance, and the decoder can reconstruct audio progressively. With an embedded codec, an encoder can terminate the encoding at any point, thus allowing a target rate or a distortion metric to be met exactly. Typically, some target parameters, such as the bit

count, is monitored in the encoding process. When the target is met, the encoding simply stops. Similarly, given a bitstream, the decoder can cease decoding at any point and produce reconstructions corresponding to all lower-rate encodings. The property of being able to terminate the encoding or decoding of an embedded bitstream at any specific point is extremely useful in systems that are either rate- or distortion-constrained [Sha93].

MPEG-4 version-2 audio coding supports fine grain bit rate scalablility [PKKS97, ISOb, ISOg, ISOh, HAB+98] in its Generic Audio Coder (GAC). It has a Bit-Sliced Arithmetic Coding (BSAC) tool, which provides scalability in the step of 1 kbit/s per audio channel for mono or stereo audio material. Several other scalable mono or stereo audio coding algorithms [ZL01, VA01, SAK99] were proposed in recent years. However, not much work has been done on progressively transmitting multichannel audio sources. Most existing multichannel audio codecs, such as AAC or AC-3, can only provide fixed-bit-rate perceptually loseless coding at about 64 kbit/s/ch. In order to transfer high quality multichannel audio through a network of a time-varying bandwidth, an embedded audio compression algorithm is highly desirable.

1.1.4 Error-Resilient Scalable Audio Bitstream

Current coding techniques for high quality audio mainly focus on coding efficiency, which makes them extremely sensitive to channel errors. A few bit errors may lead to a long period of error propagation and cause catastrophic results, including making the reconstructed sound file with un-acceptable perceptual quality and the crash

of the decoder. The desired audio bitstream transmitted over the network should be the one with error resiliency, which means that the audio bitstream should be designed to be robust to channel errors, *i.e.* make the error impact to be as small as possible. During the period when packet are corrupted or lost, the decoder should be able to perform error concealment and allow the output audio to be reproduced at acceptable quality. Compared with existing work on image, video or speech coding, the amount of work on error-resilient audio is relatively small. Techniques on robust coding and error concealment for compressed speech signals have been discussed for years. However, since speech and audio signals have different applications, straightforwardly applying these techniques to audio does not generate satisfactory results in general.

1.2 Contributions of the Research

Based on the current status of multichannel audio compression discussed in previous sections, we propose three audio coding algorithm which are all build upon the MPEG Advance Audio Coding's (AAC) basic coding structure in this dissertation. The first one is called Modified Advance Audio Coding with Karhunen-Loève Transform (MAACKLT). The second one is called Progressive Syntax-Rich Multichannel Audio Codec (PSMAC). And the third one is called Error-Resilient Scalable Audio Coding (ERSAC). Major contributions of this dissertation are summarized below.

1.2.1 Inter-Channel Redundancy Removal Approach

As mentioned in Section 1.1.1, not much effort has been made in reducing inter-channel redundancy inherent in multichannel audio sources. In our research, we carefully observe the inter-channel correlation present in multichannel audio materials and propose an effective channel redundancy removal approach. Specific contributions along this direction are listed below.

- **Observation of channel correlation**

  Based on our observation, most of the multichannel audio materials of interest exhibit two types of channel correlation. The first type only shows high correlation between CPEs, but little correlation between other channel pairs. The other type shows high correlation among all channels.

- **Proposal of an effective inter-channel redundancy removal approach**

  An inter-channel de-correlation method via KLT is adopted in the pre-processing stage to remove the redundancy inherent in original multichannel audio signals. Audio channels after KLT show little correlation between channels.

- **Study of efficiency of the proposed inter-channel de-correlation method**

  Experimental results show that the KLT pre-processing approach not only significantly de-correlates input multichannel audio signals but also considerably compacts the signal energy into the first several eigen-channels. This provides strong evidence of KLT's data compaction capability. Moreover, the energy compaction efficiency increases with the number of input channels.

- **Frequency domain KLT versus time domain KLT**

  It is observed that applying KLT to frequency domain signals achieves a better performance than directly applying KLT to time domain signals as shown by experimental results in Section 2.8. Thus, intra-channel signal de-correlation and energy compaction procedures should be performed after time-domain signals are transformed into the frequency domain via MDCT in the AAC encoder.

- **Temporal adaptive KLT**

  Multichannel audio program often comprises of different periods, each of which has its unique spectral signature. In order to achieve the highest information compactness, the de-correlation transform matrix must adapt to the characteristics of different periods. Thus, a temporal-adaptive KLT method is proposed, and the trade-off between the adaptive window size and the overhead bit rate is analyzed.

- **Eigen-channel compression**

  Since signals in de-correlated eigen-channels have different characteristics from signals in original physical channels, MPEG AAC coding blocks are modified accordingly so that they are more suitable to compress audio signals in eigen-channels.

1.2.2 Audio Concealment and Channel Transmission Strategy for Heterogeneous Network

Based on results of our KLT pre-processing approach, the advantage of this method is further explored. Once signals in original physical channels are transformed into independent eigen-channels, the energy accumulates much faster with the number of channel increases. This implies, when transmitting data of a fixed number of channels with our algorithm, more information content will be received in the decoder side and better quality of the reconstructed multichannel audio can be achieved. Possible channel transmission and recovery strategies for MPEG AAC and our algorithm are studied and compared. The following two contributions have been made in this work.

- **Channel importance sequence**

  It is desirable to re-organize the bitstream such that bits of more important channels are received at the decoder side first for audio decoding. This should result in best audio quality given a fixed amount of received bits. According to the channel energy and, at the same time, considering the sound effect caused by different channels, the channel importance for both original physical channels and KL-transformed eigen-channels is studied. A channel transmission strategy is determined according to this channel importance criterion.

- **Audio concealment and channel scalable decoding**

  When packets belonging to less important channels are dropped, an audio concealment strategy must be enforced in order to reconstruct a full multichannel audio. A channel-scalable decoding method based on this audio concealment strategy for bitstreams generated by AAC and the proposed algorithm is proposed. Experimental results show that our algorithm has a much better scalable capability and can reconstruct multichannel audio of better quality, especially at lower bit rates.

1.2.3 Quantization Efficiency for Adaptive Karhunen-Loève Transform

The quantization method for the Karhunen-Loève Transform matrix and the discussion on the temporal adaptive KLT method in the MAACKLT algorithm are relatively simple. Some questions arise in improving the efficiency of the quantization scheme. For example, can we improve the coding performance by reducing the overhead involved in transmitting the KL transform matrix? If the number of bits required to quantize each KL transform matrix is minimized, can we achieve much better inter-channel de-correlation efficiency if the KLT matrix is updated much more frequently? Having these questions in mind, we investigate the impact of different quantization methods and their efficiency for adaptive Karhunen-Loève Transform. The following two areas are addressed in this research.

- **Scalar quantizer versus the vector quantizer**

  The coding efficiency and the bit requirement of the scalar quantizer and the vector quantizer are carefully analyzed. Although a scalar quantizer applied to the KL transform matrix gives much better inter-channel de-correlation efficiency, vector quantization methods that have a smaller bit requirement achieve a better performance in term of the final MNR values of the reconstructed sound file.

- **Long versus short temporal adaptive period for KLT**

  We study how to choose a moderately long temporal adaptive period so that the optimal trade-off between the inter-channel de-correlation efficiency and the overhead bit rate can be achieved.

1.2.4 Progressive Syntax-Rich Multichannel Audio Codec Design

Being inspired by progressive image coding and the MPEG AAC system, a novel progressive syntax-rich multichannel audio compression algorithm is proposed in this dissertation. The distinctive feature of the multichannel audio bitstream generated by our embedded algorithm is that it can be truncated at any point and still reconstruct a full multichannel audio, which is extremely desirable in the network transmission. The novelty of this algorithm includes the following.

- **Subband selection strategy**

  A subband selection strategy based on the Mask-to-Noise Ratio (MNR) is proposed. An empirical MNR threshold is used to determine the importance of a subband in each channel so that the most sensitive frequency region can be reconstructed first.

- **Layered coefficient coding**

  A dual-threshold strategy is adopted in our implementation. At each layer, the MNR threshold is used to determine the subband significance, the coefficient magnitude threshold is used to determine coefficient significance. According to these selection criteria, within each selected subband, transformed coefficients are layered quantized and transmitted into the bitstream so that a coarse-to-fine multiprecision representation of these coefficients can be achieved at the decoder side.

- **Multiple context lossless coding**

  A context-based QM coder is used in the lossless coding part in the proposed algorithm. Six classes of contexts are carefully selected in order to increase the coding performance of the QM coder.

- **Three user-defined profiles**

  Three user-defined profiles are designed in this codec. They are MNR progressive, random access and channel enhancement. With these profiles, PSMAC algorithm provide end users versatile functionalities.

1.2.5 Error-Resilient Scalable Audio Coding

In order to improve the performance of the PSMAC algorithm when its bitstream is transmitted over erroneous channels, we extend its error-free codec to an error-resilient scalable audio coding (ERSAC) over WCDMA channels by re-organizing the bitstream and modifying the noiseless coding part. The distinctive features of the ERSAC algorithm are presented below.

- **Unequal error protection**

  Compared with the equal error protection scheme, the unequal error protection method gives higher priority to critical bits and, therefore, better protection. It offers an improved perceived signal quality at the same channel-signal-to-noise ratio.

- **Adaptive segmentation**

  In order to minimize the error propagation effect, the bitstream is dynamically segmented into several variable length segments such that it can be re-synchronized at the beginning of each segment even when error happens. Within each segment, bits can be independently decoded. In this way, errors can be confined to one segment, and will not propagate and affect the decoding of neighboring segments.

- **Frequency interleaving**

  To further improve the error-resilience, bits belong to the same time period but a different frequency region are divided into two groups and sent in different

packets. Therefore, even when the packets that contain bits for the header or the data part of the base layer are corrupted, not all information for the same time position is lost so that the end user is still able to reconstruct a poorer version of the sound with some frequency component missing.

1.3 Outline of the Dissertation

This dissertation consist of several chapters and they are organized as follows. Chapter 2 and Chapter 3 are devoted to the Modified Advance Audio Coding with Karhunen-Loève Transform (MAACKLT) algorithm. Chapter 2 presents the inter-channel redundancy removal approach and the channel-scalable decoding method, while Chapter 3 studies the quantization and adaptation properties of the Karhunen-Loève Transform (KLT) employed in MAACKLT algorithm. Chapter 4 describes the Progressive Syntax-Rich Multichannel Audio Codec (PAMAC). Chapter 5 extends the work done in Chapter 4 to an error robust codec design. Finally, all main results achieved in this thesis are summarized in Chapter 6. Some related research background knowledge is included in Appendices.

Chapter 2

Inter-Channel Redundancy Removal and Channel-Scalable Decoding

2.1 Introduction

In this chapter[1], we present a new algorithm called MAACKLT, which stands for Modified Advanced Audio Coding with Karhunen-Loève Transform (KLT). In MAACKLT, a 1-D temporal-adaptive KLT is applied in the pre-processing stage to remove inter-channel redundancy. Then, de-correlated signals in the KL transformed channels, called eigen-channels, are compressed by a modified AAC main profile encoder module. Finally, a prioritized eigen-channel transmission policy is enforced to achieve quality scalability.

In this work, we show that the proposed MAACKLT algorithm provides a coarse-grain scalable audio solution. That is, even if packets of some eigen-channels are

[1]Part of this chapter represents works published before, see [YAKK00b, YAKK00a, YAKK02c]

dropped completely, a slightly degraded yet full-channel audio can still be reconstructed in a reasonable fashion without any additional computational cost.

To summarize, we focus on two issues in this research. First, the proposed MAACKLT algorithm exploits inter-channel correlation existing in audio data to achieve a better coding gain. Second, it provides a quality-scalable multichannel audio bitstream which can be adaptive to networks of time-varying bandwidth. The rest of this chapter is organized as follows. Section 2.2 summarizes the inter-channel de-correlation scheme and its efficiency. Section 2.3 discusses the temporal adaptive approach. Section 2.4 describes the eigen-channel coding method and its selective transmission policy. Section 2.5 demonstrates the audio concealment strategy at the decoder end when the bitstream is partially received. The system overview of the complete MAACKLT compression algorithm is provided in Section 2.6. The computational complexity of MAACKLT is compared with that of MPEG AAC in section 2.7. Experimental results are shown in Section 2.8. Finally, concluding remarks are given in Section 2.9.

2.2 Inter-Channel Redundancy Removal

2.2.1 Karhunen-Loeve Transform

For a given time instance, removing inter-channel redundancy would result in a significant bandwidth reduction. This can be done via an orthogonal transform $MV = U$. Among several commonly used transforms, including the Discrete Cosine

Transform (DCT), the Fourier Transform (FT), and the Karhunen-Loeve Transform (KLT), the signal-dependent KLT is adopted in the pre-processing stage because it is theoretically optimal in de-correlating signals across channels. Figure 2.1 illustrates how KLT is performed on multichannel audio signals, where the columns of the KL transform matrix is composed by eigenvectors calculated from the covariance matrix associated with original multichannel audio signals.

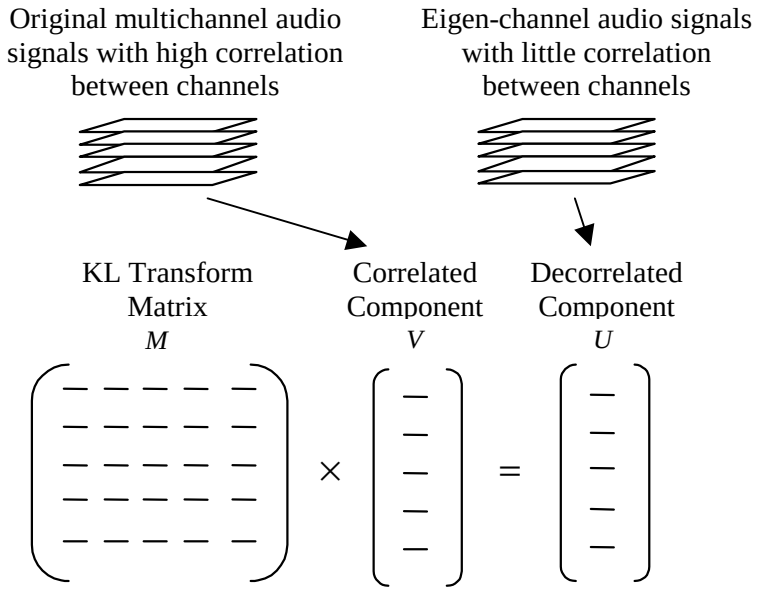


Figure 2.1: Inter-channel decorrelation via KLT.

Suppose that an input audio signal has n channels. Then, we can form an $n \times n$ KL transform matrix M composing of n eigenvectors of the cross-covariance matrix associated with these n channels. Let $V(i)$ denote the vector whose n elements are the i^{th} sample value in channel $1, 2, \ldots, n$, i.e.

$$V(i) = [x_1, x_2, \ldots, x_n]^T, \quad i = 1, 2, \ldots, k,$$

where x_j is the i^{th} sample value in channel j ($1 \leq j \leq n$), k represents the number of samples in each channel, and $[*]^T$ represents the transpose of $[*]$. The mean vector μ_V and covariance matrix C_V are defined as

$$\begin{aligned} \mu_V &= E[V] = \frac{\sum_{i=1}^{k} V(i)}{k}, \\ C_V &= E[(V-\mu_V)(V-\mu_V)^T] = \frac{\sum_{i=1}^{k}[V(i)-\mu_V][V(i)-\mu_V]^T}{k}. \end{aligned}$$

The KL transform matrix M is $M = [m_1, m_2, \ldots, m_n]^T$, where m_1, m_2, ..., m_n are eigenvectors of C_V . The covariance of KL transformed signals is

$$\begin{aligned} E[(U-\mu_U)(U-\mu_U)^T] &= E[(MV-M\mu_V)(MV-M\mu_V)^T] \\ &= ME[(V-\mu_V)(V-\mu_V)^T]M^T \\ &= MC_VM^T \\ &= \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}, \end{aligned}$$

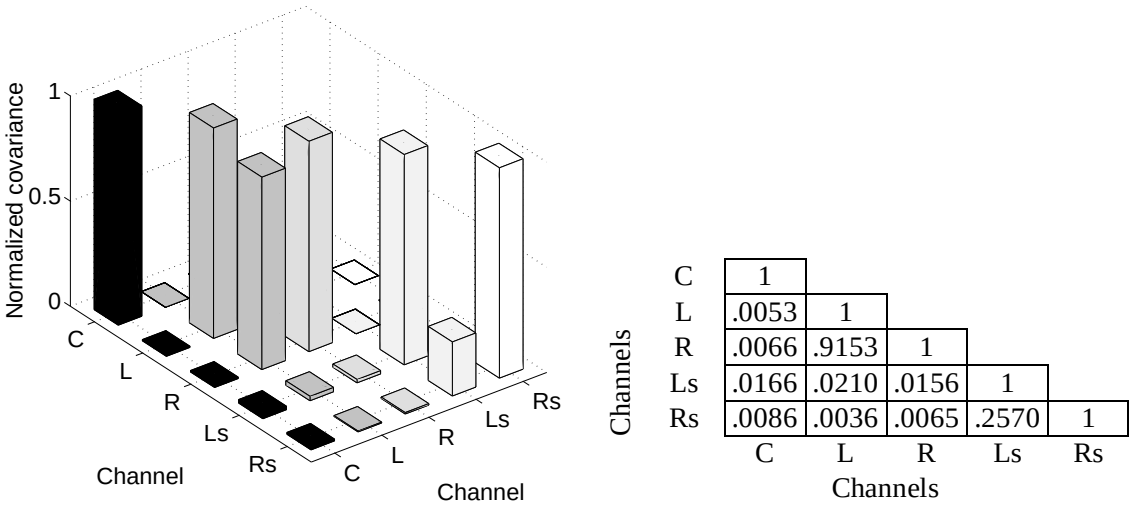
where λ_1, λ_2, ..., λ_n are eigenvalues of C_V. Thus, the transform produces statistically de-correlated channels in the sense of having a diagonal covariance matrix for transformed signals. Another property of KLT, which can be used in the reconstruction of audio of original channels, is that the inverse transform matrix of M is equal to its transpose. Since C_V is real and symmetric, the matrix formed by

normalized eigenvectors are orthonormal. Therefore, we have $V = M^T U$ in reconstruction. From KL expansion theory [Hay96], we know that selecting eigenvectors associated with the largest eigenvalues can minimize the error between original and reconstructed channels. This error will go to zero if all eigenvectors are used. KLT is thus optimum in the least-square-error sense.

2.2.2 Evidence for Inter-Channel De-Correlation

Multichannel audio sources can be roughly classified into three categories. Those belonging to class I are mostly used in broadcasting, where signals in one channel may be completely different from the other. Either broadcasting programs are different from channel to channel, or the same program is broadcast but in different languages. Samples of audio sources in class I normally contain relatively independent signals in each channel and present little correlation among channels. Therefore, this type of audio sources will not fall into the scope of high quality multichannel audio compression discussed here.

The second class of multichannel audio sources can be found in most film soundtracks, which are typically in the format of 5.1 channels. Most of this kind of program material has a symmetry property among CPEs and presents high correlation in CPEs, but little correlation across CPEs and SCEs (Single Channel Elements). Almost all existing multichannel audio compression algorithms such as AAC and Dolby AC-3 are mainly designed to encode audio material that belongs to this category. Figure 2.2 shows the normalized covariance matrix generated from one sample



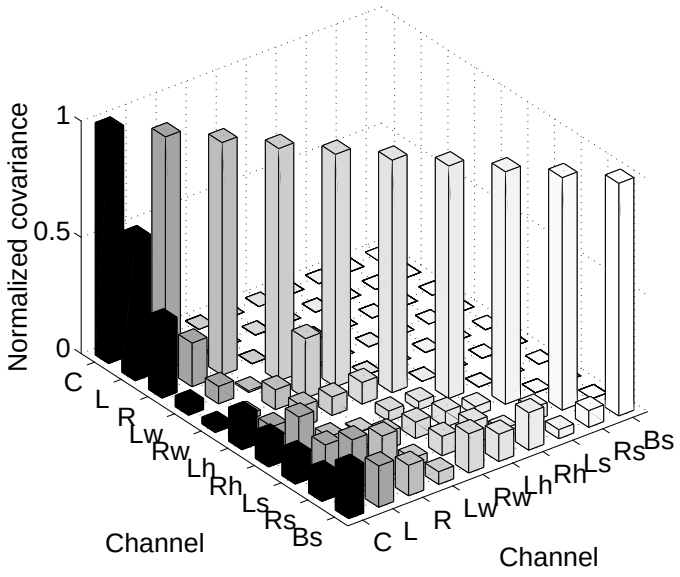
Channels	C	L	R	Ls	Rs
C	1				
L	.0053	1			
R	.0066	.9153	1		
Ls	.0166	.0210	.0156	1	
Rs	.0086	.0036	.0065	.2570	1

Channels

Figure 2.2: Absolute values of elements in the lower triangular normalized covariance matrix for 5-channel "Herre".

audio of class II, where the normalized covariance matrix is derived from the cross-covariance matrix by multiplying each coefficient with the reciprocal of the square root of the product of their individual variance. Since the magnitude of non-diagonal elements in a normalized covariance matrix provides a convenient and useful method to measure the degree of inter-channel redundancy, it is used as a correlation metric throughout the paper.

A third emerging class of multichannel audio sources consists of material recorded in a real space with multiple microphones that capture acoustical characteristics of that space. Audio of class III is becoming more prevalent with the introduction of consumer media such as DVD-Audio. This type of audio signals has considerably larger redundancy inherent among channels especially adjacent channels as graphically shown in Figure 2.3, which corresponds to the normalized covariance matrix



Channels	C	L	R	Lw	Rw	Lh	Rh	Ls	Rs	Bs
C	1									
C	.6067	1								
R	.3231	.1873	1							
Lw	.0484	.0745	.0064	1						
Rw	.0298	.0236	.0854	.2564	1					
Lh	.1464	.0493	.0887	.0791	.0921	1				
Rh	.1235	.1655	.0310	.0031	.0439	.0356	1			
Ls	.1260	.1189	.0407	.0724	.0705	.0671	.0384	1		
Rs	.1148	.2154	.1827	.0375	.0826	.0952	.0147	.0570	1	
Bs	.2156	.1773	.1305	.0540	.1686	.1173	.1606	.0383	.0714	1

Channels

Figure 2.3: Absolute values of elements in the lower triangular normalized covariance matrix for 10-channel "Messiah".

derived from a test sequence named "Messiah". As shown in the figure, a large degree of correlation is present between not only CPEs (e.g. left/right channel pair and left-surround/right-surround channel pair) but also SCE (e.g. the center channel) and any other channels.

The work presented in this research will focus on improving the compression performance for multichannel audio sources that belong to classes II and III. It will be demonstrated that the proposed MAACKLT algorithm not only achieves good

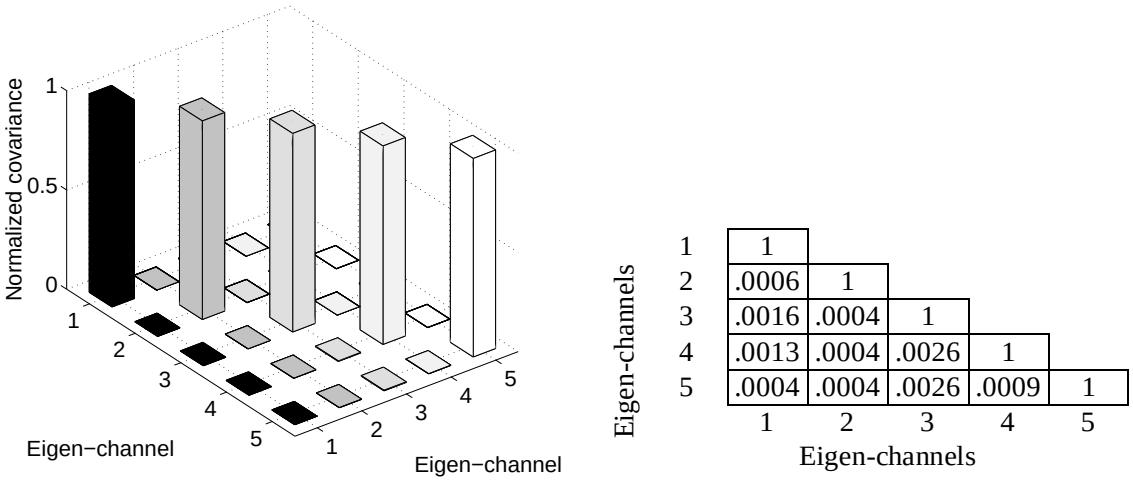
results for class III audio sources, but also improves the coding performance to a certain extent for class II audio sources compared with original AAC.

Two test data sets are used to illustrate the de-correlation effect of KLT. One is a class III 10-channel audio called "Messiah" [2], which is a piece of classical music recorded live in a concert hall. They were obtained from signals mixed from 16 microphones placed in various locations in the hall. Another one is a class II 5-channel music called "Herre" [3], which was used in MPEG-2 AAC standard (ISO/IEC 13818-7) conformance work. These test sequences are chosen because they contain a diverse range of frequency components played by several different instruments so that they are very challenging for inter-channel de-correlation and subsequent coding experiments. In addition, they provide good samples for result comparison between original AAC and the proposed MAACKLT algorithm.

Figures 2.4 and 2.5 show absolute values of elements in the lower triangular part of the normalized cross-covariance matrix after KLT for 5-channel set "Herre" and 10-channel set "Messiah". These figures clearly indicate that KLT method achieves a high degree of de-correlation. Note that the non-diagonal elements are not exactly zeros because we are dealing with an approximation of KLT during calculation. We predict that by removing redundancy in the input audio with KLT, a much better coding performance can be achieved by encoding each channel independently, which will be verified in later sections.

[2]The 10 channels include Center (C), Left (L), Right (R), Left Wide (Lw), Right Wide (Rw), Left High (Lh), Right High (Rh), Left Surround (Ls), Right Surround (Rs) and Back Surround (Bs)

[3]The 5 channels include C, L, R, Ls, and Rs.

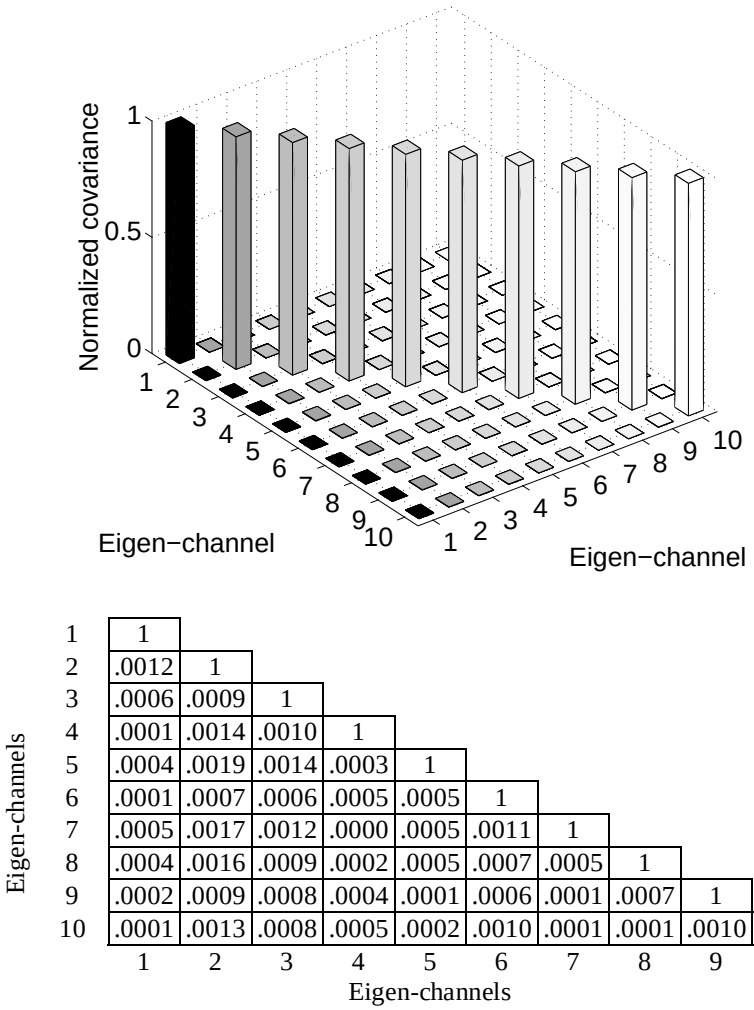


Eigen-channels	1	2	3	4	5
1	1				
2	.0006	1			
3	.0016	.0004	1		
4	.0013	.0004	.0026	1	
5	.0004	.0004	.0026	.0009	1

Figure 2.4: Absolute values of elements in the lower triangular normalized covariance matrix after KLT for 5-channel "Herre".

2.2.3 Energy Compaction Effect

The KLT pre-processing approach not only significantly de-correlates the input multichannel audio signals but also considerably compacts the signal energy into the first several eigen-channels. Figures 2.6 (a) and (b) show how energy is accumulated with an increased number of channels for original audio channels and de-correlated eigen-channels. As clearly shown in these two figures, energy accumulates much faster in the case of eigen-channels than original channels, which provides a strong evidence of data compaction of KLT. It implies that, when transmitting data of a fixed number of channels with the proposed MAACKLT algorithm, more information content will be received at the decoder side, and better quality of reconstructed multichannel audio can be achieved.



Eigen-channels	1	2	3	4	5	6	7	8	9	10
1	1									
2	.0012	1								
3	.0006	.0009	1							
4	.0001	.0014	.0010	1						
5	.0004	.0019	.0014	.0003	1					
6	.0001	.0007	.0006	.0005	.0005	1				
7	.0005	.0017	.0012	.0000	.0005	.0011	1			
8	.0004	.0016	.0009	.0002	.0005	.0007	.0005	1		
9	.0002	.0009	.0008	.0004	.0001	.0006	.0001	.0007	1	
10	.0001	.0013	.0008	.0005	.0002	.0010	.0001	.0001	.0010	1

Eigen-channels

Figure 2.5: Absolute values of elements in the lower triangular normalized covariance matrix after KLT for 10-channel "Messiah".

Another convenient way to measure the amount of data compaction can be obtained via eigenvalues of the cross-covariance matrix associated with the KL transformed data. In fact, these eigenvalues are nothing else but variances of eigen-channels, and the variance of a set of signals reflects its degree of jitter, or the information content. Figures 2.7 (a) and (b) are plots of variances of eigen-channels

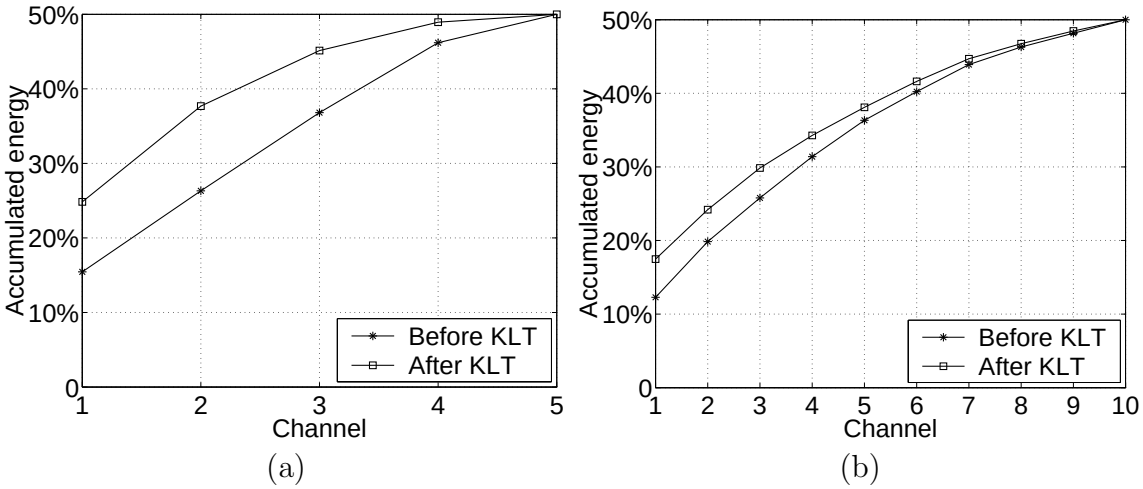


Figure 2.6: Comparison of accumulated energy distribution for (a) 5-channel "Herre" and (b) 10-channel "Messiah".

associated with the "Messiah" test set consisting of 10 and 5 channels, respectively. As shown in figures, the variance drops dramatically with the order of eigen-channels. The steeper the variance drop is, the more efficient the energy compaction is achieved. These experimental results also show that the energy compaction efficiency increases with the number of input channels. The area under the variance curve reflects the amount of information to be encoded. As illustrated from these two figures, this particular area is substantially much smaller for the 10-channel set than that of the 5-channel set. As the number of input channels decreases, the final compression performance of MAACKLT tends to be more influenced by the coding power of the AAC main profile encoder.

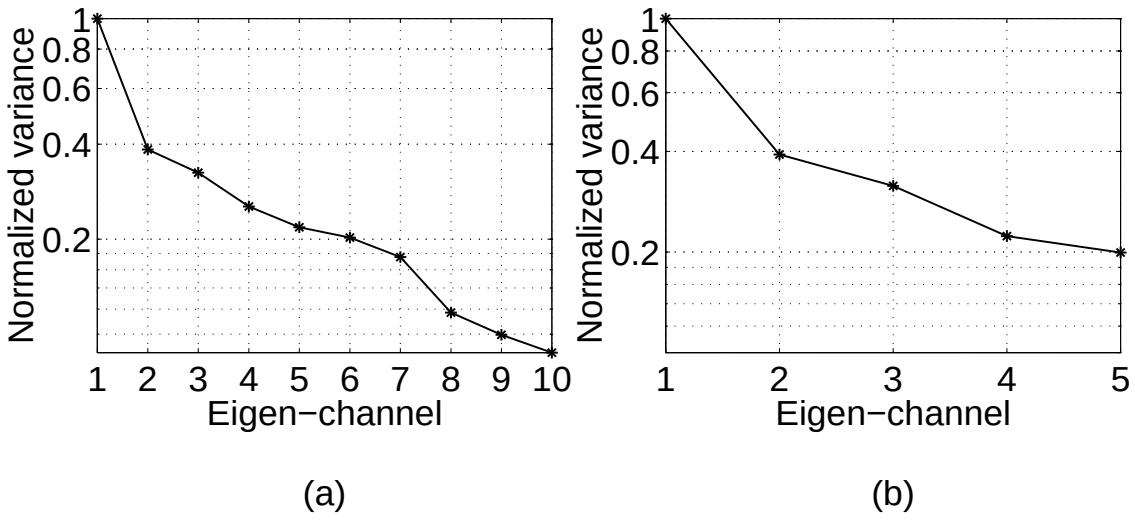


Figure 2.7: Normalized variances for (a) 10-channel "Messiah", and (b) 5-channel "Messiah", where the vertical axis is plotted in the log scale.

2.2.4 Frequency-Domain versus Time-Domain KLT

In all previous discussion, we considered only the case of applying KLT to time-domain signals across channels. However, it is also possible to apply the inter-channel de-correlation procedure after time-domain signals are transformed into the frequency-domain via MDCT (Modified Discrete Cosine Transform) in the AAC encoder.

One frame of the audio signal from the center channel of "Herre" in the frequency-domain and in the time-domain are shown in Figures 2.8 (a) and (b), respectively. The energy compaction property can be clearly seen from the simple comparison between the time-domain and the frequency-domain plots. Generally speaking, applying KLT to frequency-domain signals achieve a better performance than directly applying KLT to time-domain signals. In addition, a certain degree of delay and

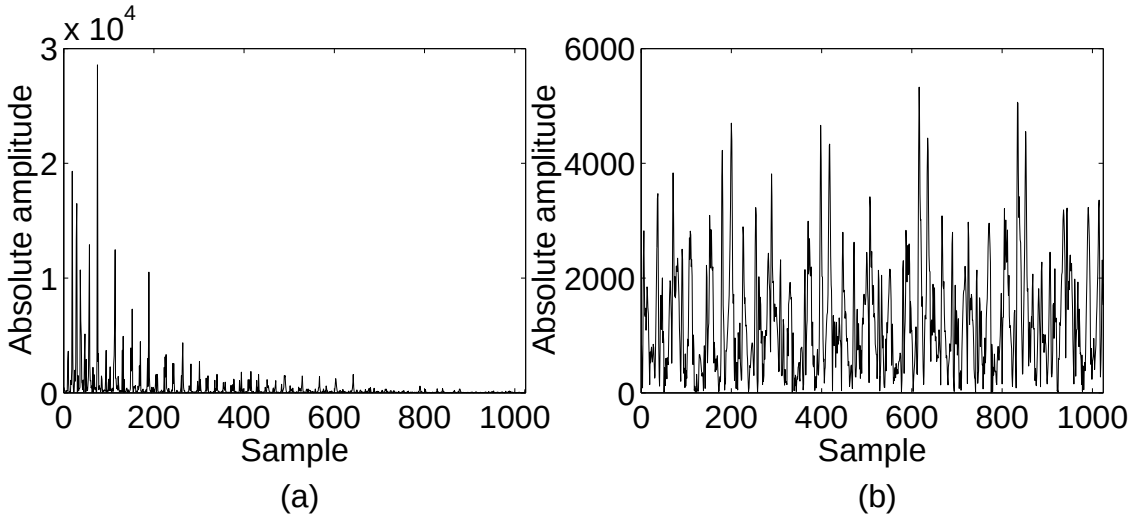


Figure 2.8: (a) Frequency-domain and (b) time-domain representations of the center channel from "Herre".

reverberant sound copies may exist in time-domain signals among different channels, which is especially true for class III multichannel audio sources. The delay and reverberation effects affect the time-domain KLT's de-correlation capability, however, they may not have that much impact on frequency-domain signals. Figures 2.9 and 2.10 show absolute values of off-diagonal non-redundant elements for normalized covariance matrices generated from frequency- and time-domain KL transforms with test audio "Herre" and "Messiah", respectively. Clearly, the frequency-domain KLT has a much better inter-channel de-correlation capability than that of the time-domain KLT. This implies that applying KLT to frequency-domain signals should lead to a better coding performance, which will be verified by experimental results shown in Section 2.8. And any results discussed hereafter will focus on frequency-domain KLT method unless otherwise mentioned.

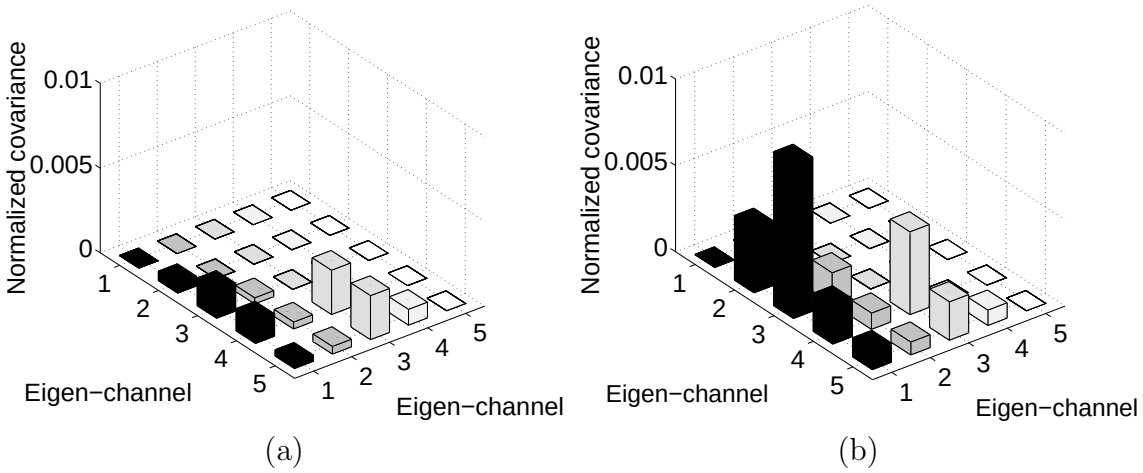
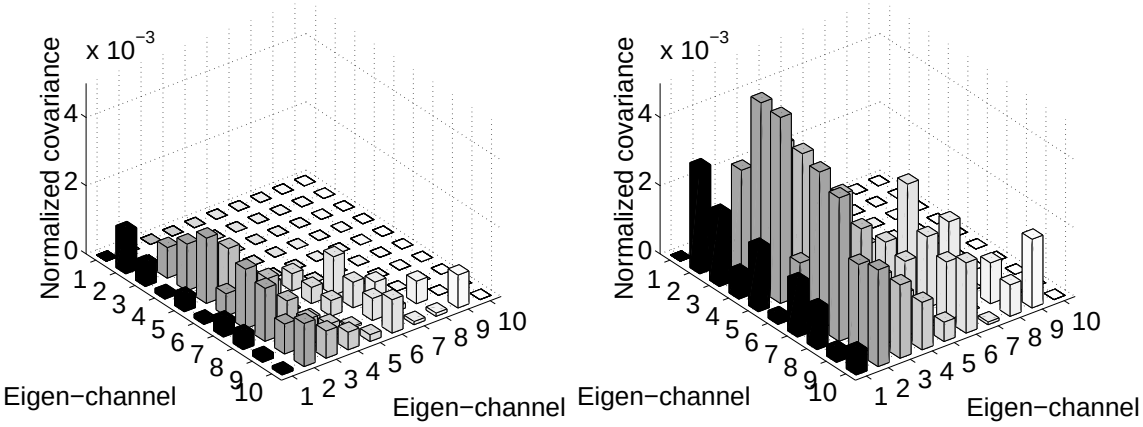


Figure 2.9: Absolute values of off-diagonal elements for the normalized covariance matrix after (a) frequency-domain and (b) time-domain KL transforms with test audio "Herre".

2.3 Temporal Adaptive KLT

A multichannel audio program comprises of different periods, each of which has its unique spectral signature. For example, a piece of music may begin with a piano preclude followed by a chorus. In order to achieve the highest information compactness, the de-correlation transform matrix should be adaptive to the characteristics of different periods. In this section, we present a temporal-adaptive KLT approach, in which the covariance matrix (and, consequently, the corresponding KL transform matrix) is updated from time to time. Each adaptive period is called a "block".

Figure 2.11 shows the variance of each eigen-channel of one non-adaptive and two temporal-adaptive approaches for test set "Messiah". Compared with the non-adaptive method, the adaptive method achieves a smaller variance for each eigen-channel. Furthermore, the shorter the adaptive period, the higher inter-channel de-correlation is achieved. The only drawback of the temporal-adaptive approach over



(a) (b)

Figure 2.10: Absolute values of off-diagonal elements for the normalized covariance matrix after (a) frequency-domain and (b) time-domain KL transforms with test audio "Messiah".

the non-adaptive approach goes to the overhead bits, which have to be transmitted to the decoder so that the multichannel audio can be reconstructed to its original physical channels. Due to the increase of the block number, the shorter the adaptive-period is, the larger the overhead bit rate is. The trade-off between this "block" size and the overhead bit rate will be discussed below.

Since the inverse KLT has to be performed at the decoder side, the information of the transform matrix should be included in the coded bitstream. As mentioned before, the inverse KLT matrix is the transpose of the forward KLT matrix, which is composed by eigenvectors of the cross-covariance matrix. To reduce the overhead bit rate, elements of the covariance matrix are included in the bitstream instead of those of the KLT matrix since the covariance matrix is real and symmetric and we only have to send the lower (or higher) triangular part that contains non-redundant

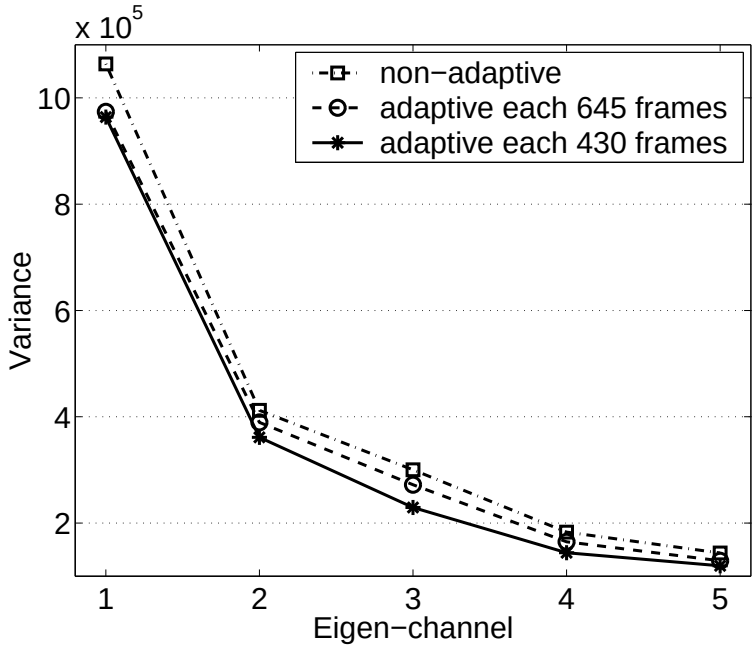


Figure 2.11: De-correlation efficiency of temporal adaptive KLT.

elements. As a result, the decoder also has to calculate eigenvectors of the covariance matrix before the inverse KLT can be performed.

Only one covariance matrix has to be coded for the non-temporal-adaptive approach. However, for the temporal-adaptive approach, every covariance matrix must be coded for each block. Assume that n channels are selected for simultaneous inter-channel de-correlation, and the adaptive period is K seconds, i.e. each block contains K seconds of audio. The size of the covariance matrix is $n \times n$, and the number of non-redundant elements is $n \times (n+1)/2$. In order to reduce the overhead bit rate, the floating-point covariance matrix is quantized to 16 bits per element. Therefore,

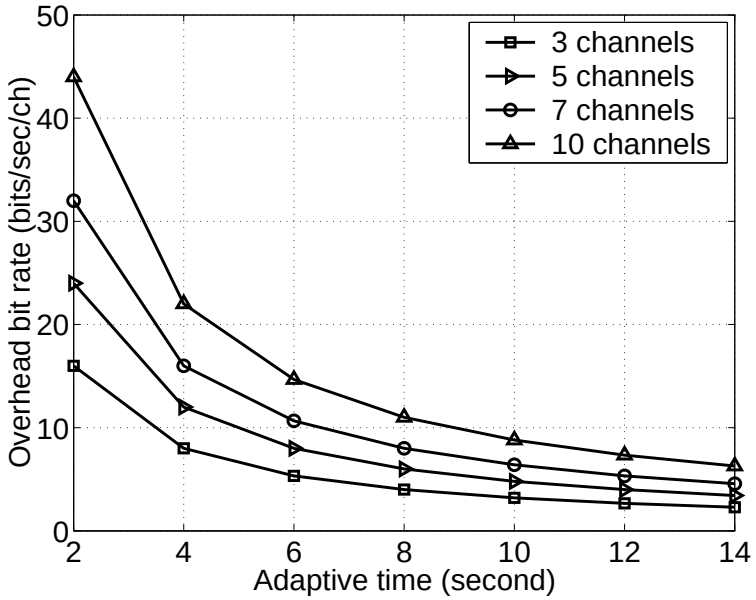


Figure 2.12: The overhead bit rate versus the number of channel and the adaptive period.

the total bit requirement for each covariance matrix is $8n \times (n+1)$ bits, and the overhead bit rate $r_{overhead}$ is

$$r_{overhead} = \frac{8n \times (n+1)}{nK} = \frac{8(n+1)}{K} \tag{2.1}$$

in bit per second per channel (bit/s/ch). The above equation suggests that the overhead bit rate increases approximately linearly with the number of channels. The overhead bit rate is, however, inversely proportional to the adaptive time (or the block size).

Figure 2.12 illustrates the overhead bit rate for different channel numbers and block sizes. The optimal adaptive time is around 10 seconds. At this block size, inter-channel redundancy can be efficiently removed with a reasonable cost of overhead

bits. From this figure, we know that a 10-second block only generates less than 10 bit/s/ch overhead bit rate. Compared with the 64 kbit/s/ch typical bit rate, this overhead is so small that it can be neglected.

2.4 Eigen-Channel Coding and Transmission

2.4.1 Eigen-Channel Coding

The main profile of the AAC encoder is modified to compress audio signals in de-correlated eigen-channels. The detailed encoder block diagram is given in Figure 2.13, where the shaded parts represent coding blocks that are different from the original AAC algorithm.

The major difference between Figure 2.13 and the original AAC encoder block diagram is the KLT block added after the filter bank. When the original input signals are transformed into frequency domain, the cross-channel KLT are performed to generate the de-correlated eigen-channel signals. Masking thresholds are then calculated based on the KL transformed signals in the perceptual model. The KLT related overhead information is sent into the bitstream afterwards.

The original AAC is typically used to compress class II audio sources. Its M/S stereo coding block is specifically used for symmetric CPEs. It encodes the mean and difference of CPEs instead of two independent SCEs, which reduces redundancy existing in symmetric channel pairs. In the proposed algorithm, since inter-channel de-correlation has been performed in an earlier stage and audio signals after KLT are

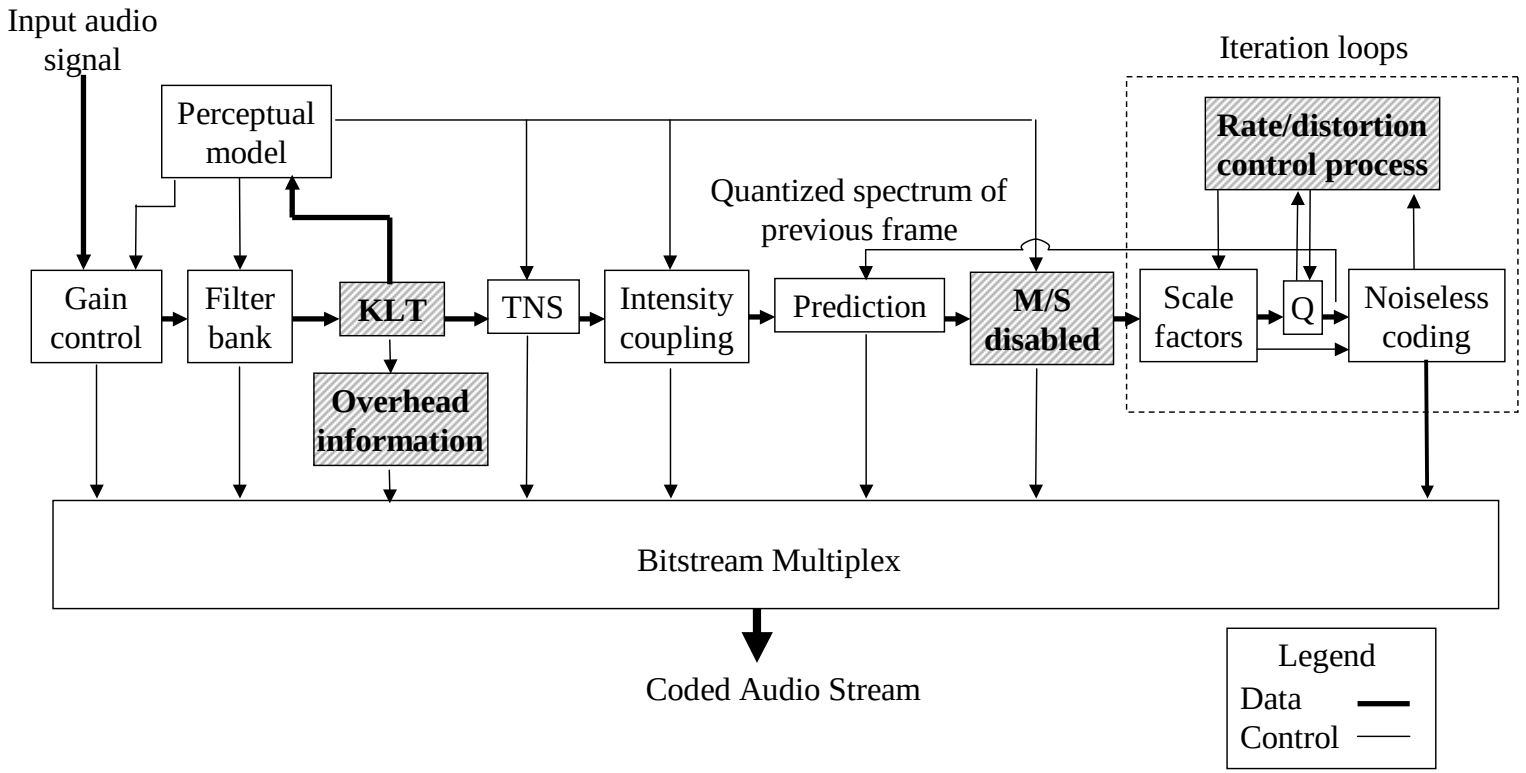


Figure 2.13: The modified AAC encoder block diagram.

from independent eigen-channels with little correlation between any channel pairs, the M/S coding block is no longer needed. Thus, the M/S coding block of the AAC main profile encoder is disabled.

The AAC encoder module originally assigns an equal amount of bits to each input channel. However, since signals into the iteration loops are no longer the original multichannel audio in the new system, the optimality of the same strategy has to be investigated. Experimental results indicate that the compression performance will be strongly influenced by the bit assignment scheme for de-correlated eigen-channels.

According to the bit allocation theory [GG91], the optimal bit assignment for identically distributed normalized random variables under the high rate approximations while without nonnegativity or integer constraints on the bit allocations is

$$b_i = \bar{b} + \frac{1}{2}\log_2 \frac{\sigma_i^2}{\rho^2}, \tag{2.2}$$

where $\bar{b} = \frac{B}{k}$ is the average number of bits per parameter, k is the number of parameters, and $\rho^2 = (\prod_{i=1}^{k} \sigma_i^2)^{\frac{1}{k}}$ is the geometric mean of the variances of the random variables. It is verified by experimental data that the normalized probability density functions of signals in eigen-channels are almost identical. They are given in Figures 2.14 and 2.15. This optimal bit allocation method is adopted for rate/distortion control processing when coding eigen-channel signals.

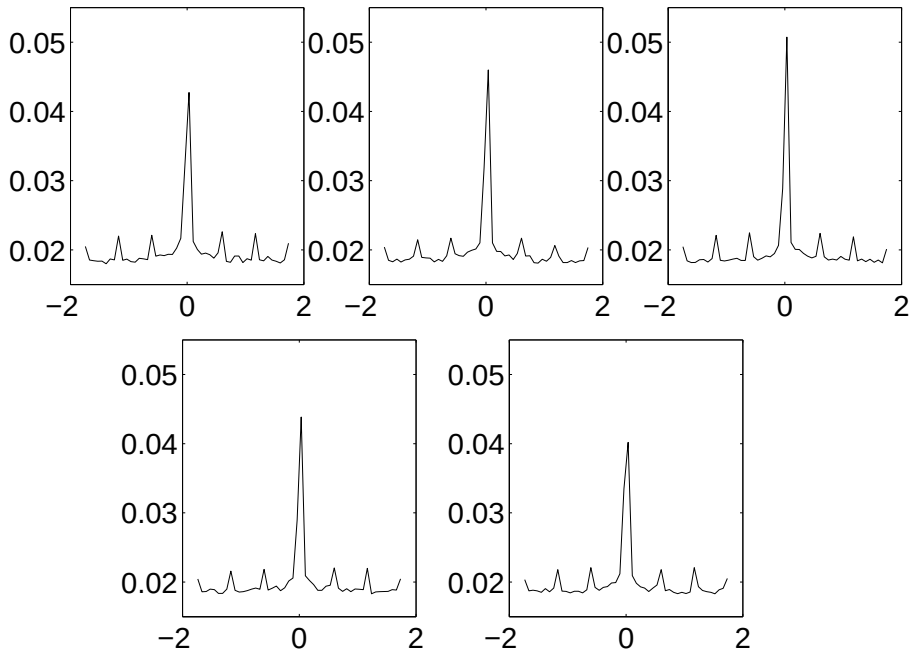


Figure 2.14: The empirical probability density functions of normalized signals in 5 eigen-channels generated from test audio "Herre".

2.4.2 Eigen-Channel Transmission

Figures 2.6 (a) and (b) show that the signal energy accumulates faster in eigen-channel form than original multichannel form. This implies that, with a proper channel transmission and recovery strategy, transmitting the same number of eigen-channels and of original multichannels, the eigen-channel approach should result in a higher quality reconstructed audio since more energy is transmitted.

It is desirable to re-organize the bitstream so that bits of more important channels can be received at the decoder side first for audio decoding. This should result in the best audio quality given a fixed amount of received bits. When this re-organized audio bitstream is transmitted over a heterogeneous network, for those users with

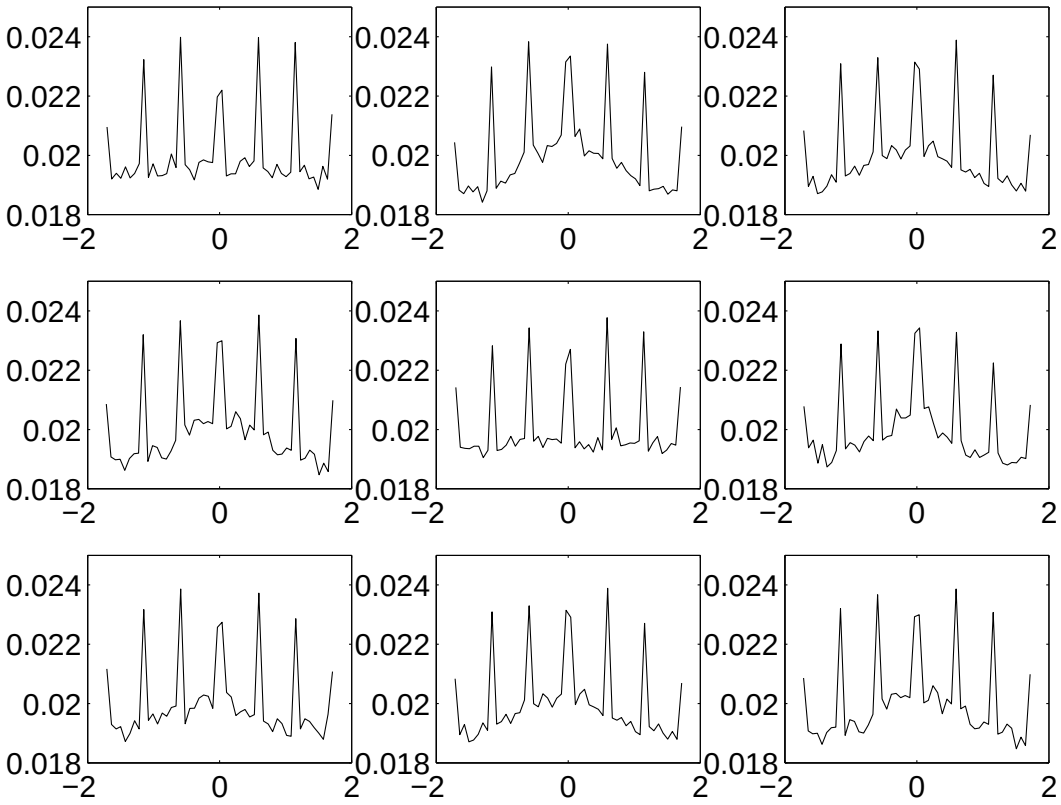


Figure 2.15: The empirical probability density functions of normalized signals in the first 9 eigen-channels generated from test audio "Messiah".

a limited bandwidth, the network can drop packets belonging to less important channels.

The first instinct about the metric of channel importance would be the energy of the audio signal in each channel. However, this metric does not work well in general. For example, for some multichannel audio sources, especially those belonging to class II, since they are re-produced in a music studio artificially, the side channel which normally does not contain the main melody may even has a larger energy than the center channel. Based on our experience with multichannel audio, loss or significant distortion of the main melody in the center channel would be much more

annoying than loss of melodies in side channels. In other words, the location of channels also plays an important role. Therefore, for a regular 5.1 channel configuration, the order of channel importance from the largest to the least should be:

1. Center channel,
2. L/R channel pair,
3. Ls/Rs channel pair,
4. Low frequency channel.

Between channel pairs, their importance can be determined by their energy values. This rule is adopted in experiments below.

After KLT, eigen-channels are no longer the original physical channels, and sounds in different physical channels are mixed in every eigen-channel. Thus, spatial dependency of eigen-channels is less trivial. We observe from experiments that although it is true that one eigen-channel may contain sounds from more than one original physical channel, there still exists a close correspondence between eigen-channels and physical channels. To be more precise, audio of eigen-channel 1 would sound similarly to that of the center channel, audio of eigen-channels 2 and 3 would sound similarly to that of the L/R channel pair etc. Therefore, if eigen-channel 1 is lost in transmission, we would end up with a very distorted center channel. Moreover, it happens that, sometimes, eigen-channel 1 may not be the channel with a very large energy and could be easily discarded if the channel energy is adopted as the metric of channel importance. Thus, the channel importance of eigen-channels

should be similar to that of physical channels. That is, eigen-channel 1 corresponding to the center channel, eigen-channel 2 and 3 corresponding to the L/R channel pair, eigen-channel 4 and 5 corresponding to the Ls/Rs channel pair. Within each channel pair, the importance is still determined by their energy values.

2.5 Audio Concealment for Channel-Scalable Decoding

Consider the scenario that an AAC-coded multichannel bitstream is transmitted in a heterogeneous network such as the Internet. For end-users who do not have enough bandwidth to receive full channel audio, some packets have to be dropped. In this section, we consider the bitstream of each channel as one minimum unit for audio reconstruction. When the bandwidth is not sufficient, we may drop bitstreams of a certain number of channels to reduce the bit rate. It is called channel-scalable decoding, which has an analogy in MPEG video coding, i.e. dropping B frames while keeping only I and P frames.

For an AAC channel pair, the M/S stereo coding block will replace low frequency coefficients in symmetric channels to be their sum and difference at the encoder, i.e.

$$spec_l[i] \leftarrow (spec_l[i] + spec_r[i])/2, \tag{2.3}$$

$$spec_r[i] \leftarrow (spec_l[i] - spec_r[i])/2, \tag{2.4}$$

where $spec_l[i]$ and $spec_r[i]$ are the i^{th} frequency-domain coefficient in the left and right channels of the channel pair, respectively.

The intensity coupling coding block will replace high frequency coefficients of the left channel with a value proportional to the envelope of the sound signal in the symmetric channel, and set the value of right channel high frequency coefficients to zero, i.e.

$$spec_l[i] \leftarrow (spec_l[i] + spec_r[i]) \times \sqrt{\frac{E_l[sfb]}{E_s[sfb]}}, \tag{2.5}$$

$$spec_r[i] \leftarrow 0, \tag{2.6}$$

where $E_l[sfb]$, $E_r[sfb]$ and $E_s[sfb]$ represent, respectively, energy values of the left channel, the right channel and the sum of left and right channels of the subband that sample i belongs to. Values of $\frac{E_l[sfb]}{E_r[sfb]}$ are sent to the bitstream as scaling factors.

At the decoder end, the low frequency coefficients of the left and right channel are reconstructed via

$$spec_l[i] \leftarrow spec_l[i] + spec_r[i], \tag{2.7}$$

$$spec_r[i] \leftarrow spec_l[i] - spec_r[i]. \tag{2.8}$$

For high frequency coefficients, audio signals in the left channel will remain the same as they are received from the bitstream, while those in the right channel will be reconstructed via

$$spec_r[i] = scale \times spec_l[i], \tag{2.9}$$

where *scale* is the inverse of the scaling factor.

When packets of one channel of a channel pair are dropped, we drop frequency coefficients of the right channel while keeping all other side information including scaling factors. Therefore, what we receive at the decoder side are just coefficients in the left channel. For low frequency coefficients, they correspond to the mean value of the original frequency coefficient in the left and right channels. For high frequency coefficients, they correspond to the energy envelope of the symmetric channel. That it, we have

$$spec_l[i] \rightarrow (spec_l[i] + spec_r[i])/2, \tag{2.10}$$

$$spec_r[i] \rightarrow 0, \tag{2.11}$$

for the low frequency part and

$$spec_l[i] \rightarrow (spec_l[i] + spec_r[i]) \times \sqrt{\frac{E_l[sfb]}{E_s[sfb]}}, \tag{2.12}$$

$$spec_r[i] \rightarrow 0, \tag{2.13}$$

for the high frequency part.

Note that since scaling factors are contained in the received bitstream, reconstruction of high frequency coefficients in the right channel will remain the same as the original AAC when data of all channels are received. Therefore, only low frequency coefficients in the right channel need to be recovered. The strategy used to reconstruct these coefficients is just to let values of right channel coefficients equal to values of received left channel coefficients. This is nothing else but the mean value of coefficients in the original channel pair, i.e.

$$spec_r[i] = spec_l[i] \rightarrow (spec_l[i] + spec_r[i])/2. \tag{2.14}$$

Audio concealment for the proposed eigen-channel coding scheme is relatively simple. All coefficients in dropped channels will be set to 0, then a regular decoding process is performed to reconstruct full multichannel audio. For the situation where packets of two or more channels are dropped, the reconstructed dropped channel may have a much smaller energy than other channels after inverse KLT. In order to get better reconstructed audio quality, an energy boost up process can be enforced so that the signal in each channel will have a similar amount of energy.

To illustrate that the proposed algorithm MAACKLT has a better quality-degradation property than AAC (via a proper audio concealment process described in this section), we perform experiments with lossy channels where packets are dropped in a coded bitstream in Section 2.8.

2.6 Compression System Overview

The block diagram of the proposed compression system is illustrated in Figure 2.16. It consists of four modules: (1) data partitioning, (2) Karhunen-Loève transform, (3) dynamic range control, and (4) the modified AAC main profile encoder. In the data partitioning module, audio signals in each channel are partitioned into sets of non-overlapping intervals, i.e. blocks. Each block contains K frames, where K is a pre-defined value. Then, data in each block are sequentially fed into the KLT module to perform inter-channel de-correlation. In the KLT module, multichannel block data are de-correlated to produce a set of statistically independent eigen-channels. The KLT matrix consists of eigenvectors of the cross-covariance matrix associated with the multichannel block set. The covariance matrix is first estimated and then quantized into 16 bits per element. The quantized covariance coefficients will be sent to the bitstream as the overhead.

As shown in Figure 2.1, eigen-channels are generated by multiplication of the KLT matrix and the block data set. Therefore, after the transform, the sample value in eigen-channels may have a larger dynamic range than that of original channels. To avoid any possible data overflow in the later compression module, data in eigen-channels are rescaled in the dynamic range control module so that the sample value input to the modified AAC encoder module does not exceed the dynamic range of that in regular 16-bit PCM audio files. This rescaling information will also be sent to the bitstream as the overhead.

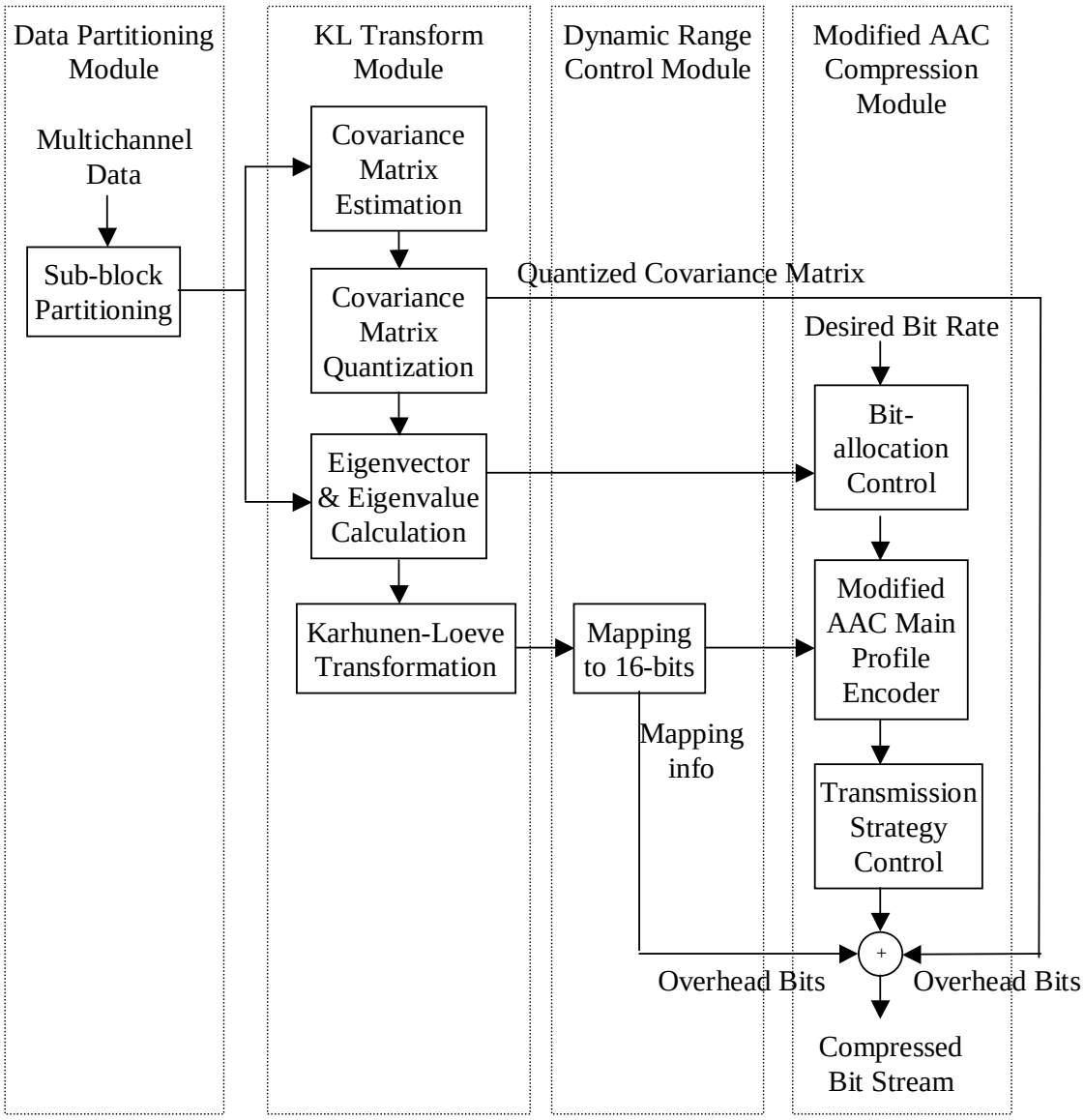


Figure 2.16: The block diagram of the proposed MAACKLT encoder.

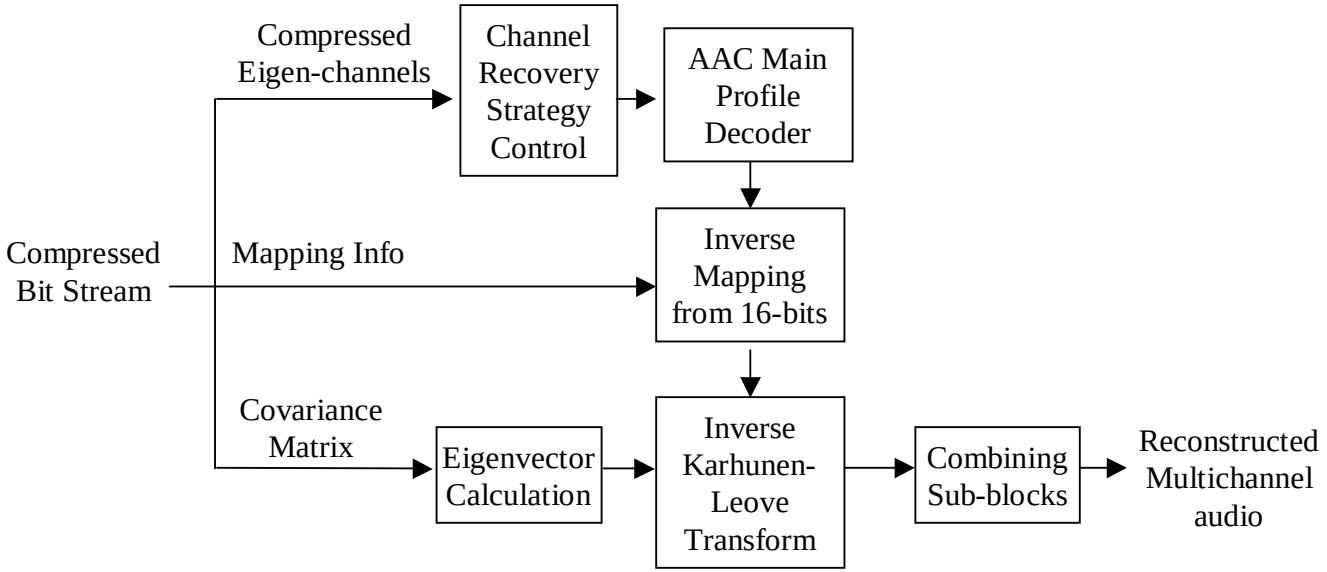


Figure 2.17: The block diagram of the proposed MAACKLT decoder.

Signals in de-correlated eigen-channels are compressed in the next module by a modified AAC main profile encoder. The AAC main profile encoder is modified in our algorithm so that it is more suitable in compressing the audio signal in eigen-channels. To enable channel-scalability, a transmission strategy control block is adopted in this module right before the compressed bitstream is formed.

The block diagram of the decoder is shown in Figure 2.17. The mapping information and the covariance matrix together with the coded information for eigen-channels are extracted from the received bitstream. If data of some eigen-channels are lost due to the network condition, the eigen-channel concealment block will be enabled. Then, signal values in eigen-channels will be reconstructed by the AAC main profile decoder. The mapping information is used to restore from a 16-bit dynamic range of the decoded eigen-channel back to its original range. The inverse KLT matrix can be calculated from the extracted covariance matrix via transposing its eigenvectors. Then, inverse KLT is performed to generate the reconstructed multichannel block set. These block sets are finally combined together to produce the reconstructed multichannel audio signals.

2.7 Complexity Analysis

Compared with the original AAC compression algorithm, the additional computational complexity required by the MAACKLT algorithm mainly comes from the KLT

Table 2.1: Comparison of computational complexity between MAACKLT and AAC

Encoding					
Time used (seconds)	MAACKLT		AAC	Extra	
				Time	Percent
Messiah	1-sec AP	344.28		26.15	8.2%
Messiah	5-sec AP	340.43		22.30	7.0%
Messiah	10-sec AP	339.44		21.31	6.7%
Messiah	NonA	337.62	318.13	19.49	6.1%
Herre	NonA	112.15	101.23	10.92	10.8%
Decoding					
Time used (seconds)	MAACKLT		AAC	Extra	
				Time	Percent
Messiah	1-sec AP	16.92		4.62	37.6%
Messiah	5-sec AP	16.04		3.74	30.4%
Messiah	10-sec AP	15.60		3.30	26.8%
Messiah	NonA	14.66	12.30	2.36	19.2%
Herre	NonA	2.75	2.42	0.33	13.6%

pre-processing module, which includes generation of the cross-covariance matrix, calculation of its eigenvalues and eigenvectors, and matrix multiplication required by KLT.

Table 2.1 illustrates the running time of MAACKLT and AAC for both the encoder and the decoder at a typical bit rate of 64 kbit/s/ch, where "n-sec AP" means the MAACKLT algorithm with a temporal adaptive period of n seconds while "NonA" means a non-adaptive MAACKLT algorithm. The input test audio signals are 20-second 10-channel "Messiah" and 8-second 5-channel "Herre". The system used to generate the above result is a Pentium III 600 PC with 128M RAM.

These results indicate that the coding time for MAACKLT is still dominated by the AAC compression and de-compression part. When the optimal 10-second

temporal adaptive period is used for test audio "Messiah", the additional KLT computational time is less than 7% of the total encoding time at the encoder side while the MAACKLT algorithm only takes about 26.8% longer than that of the original AAC at the decoder side. The MAACKLT algorithm with a shorter adaptive period will take a little bit more time in encoding and decoding since more KL transform matrices are need to be generated. Note also that we have not made any attempt to optimize our experimental codes. A much lower amount of encoding/decoding time of MAACKLT is expected if the source code for the KLT pre-processing part is carefully re-written to optimize the performance.

In channel-scalable decoding, when packets belonging to less important channels are dropped during transmission in the heterogeneous network, the audio concealment part adds a negligible amount of additional complexity in the MAACKLT decoder. The decoding time remains about the same as that of regular bit rate decoding at 64 kbit/s/ch when all packets are received at the decoder side.

2.8 Experimental Results

2.8.1 Multichannel Audio Coding

The proposed MAACKLT algorithm has been implemented and tested under the PC Windows environment. We supplemented an inter-channel redundancy removal block and a channel transmission control block to the basic source code structure of MPEG-2 AAC [ISOa, ISOc]. The proposed algorithm is conveniently parameterized

to accommodate various input parameters, such as the number of audio channels, the desired bit rate and the window size of temporal adaptation, etc.

We have tested the coding performance of the proposed MAACKLT algorithm by using three 10-channel set audio data "Messiah", "Band" and "Herbie" and one 5-channel set audio data "Herre" at a typical rate of 64 kbit/s/ch. Test materials "Messiah" and "Band" are class III audio files, while "Herbie" and "Herre" are class II audio files. Figures 2.18 (a) and (b) show the mean Mask-to-Noise-Ratio (MNR) comparison between the original AAC[4] and the MAACKLT scheme for the 10-channel set "Herbie" and the 5-channel set "Herre", respectively. The mean MNR values in these figures are calculated via

$$\text{mean MNR}_{subband} = \frac{\sum_{channel} \text{MNR}_{channel,subband}}{\text{number of channels}}. \tag{2.15}$$

The mean MNR improvement shown in these figures are calculated via

$$\text{mean MNR improvement} = \frac{\sum_{subband}(\text{mean MNR}^{\text{MAACKLT}}_{subband} - \text{mean MNR}^{\text{AAC}}_{subband})}{\text{number of subbands}}. \tag{2.16}$$

Experimental results shown in Figure 2.18 (a) and (b) are generated by using the frequency-domain non-adaptive KLT method. These plots clearly indicate that MAACKLT outperforms AAC in the objective MNR measurement for most subbands and achieves mean MNR improvement of more than 1 dB for both test audio.

[4]All audio files generated by AAC in this section are processed by the AAC main profile encoder and decoder.

It implies that, compared with AAC, MAACKLT can achieve a higher compression ratio while maintaining similar indistinguishable audio quality. It is worthwhile to mention that no software optimization has been performed for any codec used in this section and all coding blocks adopted from AAC have not been modified to improve the performance of our codec.

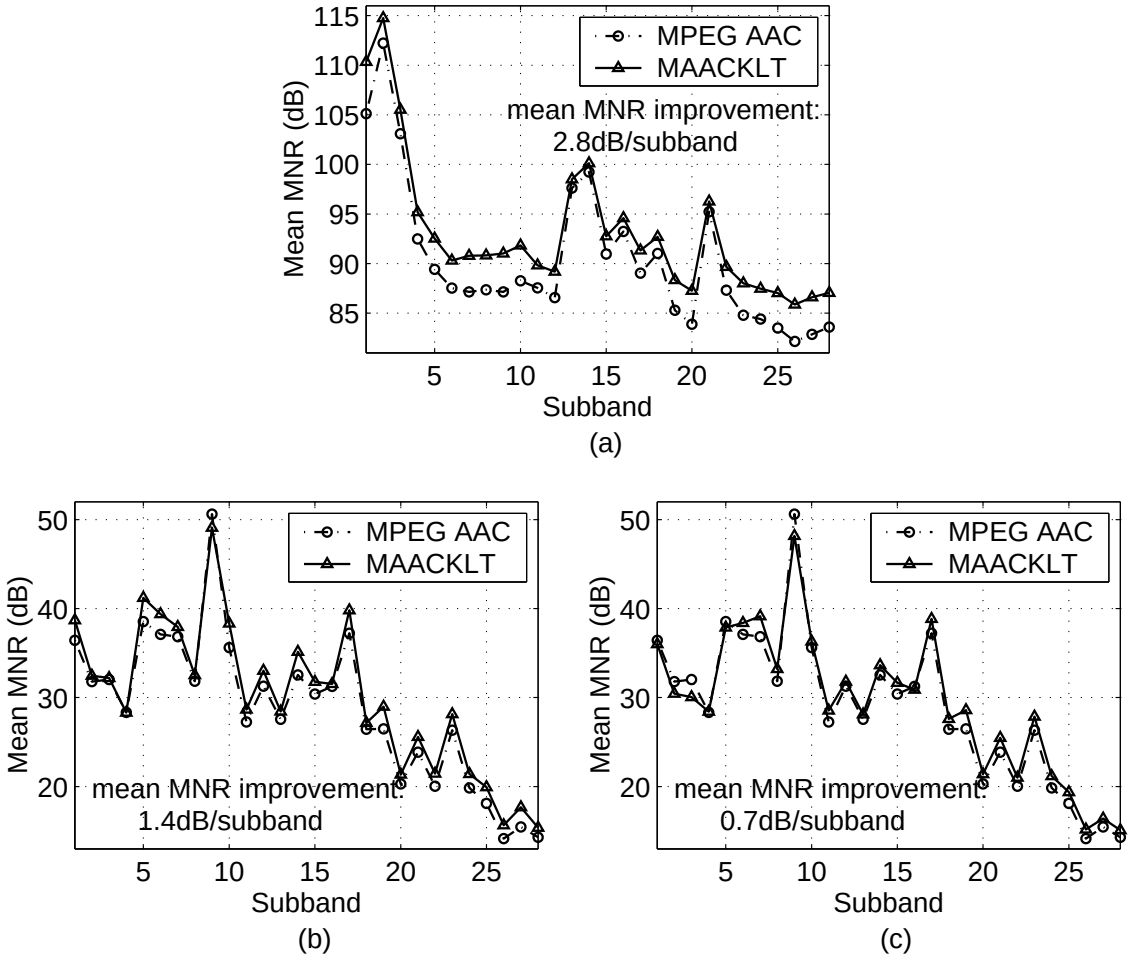


Figure 2.18: The MNR comparison for (a) 10-channel "Herbie" using frequency-domain KLT (b) 5-channel "Herre" using frequency-domain KLT (c) 5-channel "Herre" using time-domain KLT.

Figure 2.18 (c) shows the mean MNR comparison between AAC and MAACKLT with the time-domain KLT method using 5-channel set "Herre". Compared with

the result shown in Figure 2.18 (b), we confirm that frequency-domain KLT achieves a better coding performance than time-domain KLT.

The experimental result for the temporal-adaptive approach for 10-channel set "Messiah" is shown in Figure 2.19. This result verifies that a shorter adaptive period de-correlates the multichannel signal better but sacrifices the coding performance by adding the overhead in the bitstream. On the other hand, if the covariance matrix is not updated frequently enough, inter-channel redundancy cannot be removed to the largest extent. As shown in the figure, to compromise these two constraints, the optimal adaptive period for "Messiah" is around 10 seconds.

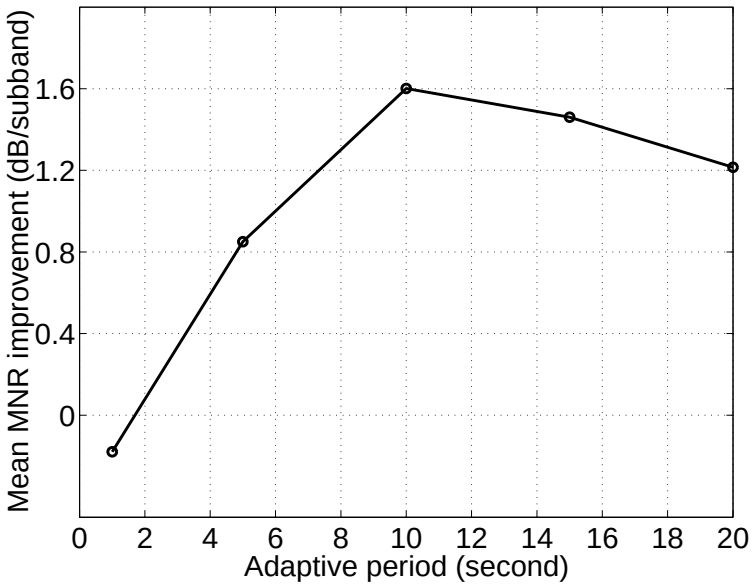


Figure 2.19: The mean MNR improvement for temporal-adaptive KLT applied to the coding of 10-channel "Messiah", where the overhead information is included in the overall bit rate calculation.

2.8.2 Audio Concealment with Channel-Scalable Coding

As described in Section 2.5, when packets of one channel from a channel pair are lost, we can conceal the missing channel at the decoder side. Experimental results show that the quality of the recovered channel pair with the AAC bitstream is much worse than that of the MAACKLT bitstream when it is transmitted under the same network condition.

Take the test audio "Herre" as an example. If one of the L/R channel pair is lost, the reconstructed R channel using the AAC bitstream has obvious distortion and discontinuity in several places while the reconstructed right channel by using the MAACKLT bitstream has little distortion and is much more smoother. If one of the Ls/Rs channel pair is lost, the reconstructed Rs channel using the AAC bitstream has larger noise in the first one to two seconds in comparison with that of MAACKLT. The corresponding MNR values are compared in Figures 2.20 (a) and (b) when AAC and MAACKLT are used and missing channels are concealed, when packets of one channel from L/R and Ls/Rs channel pairs are lost. We see clearly that MAACKLT achieves better MNR values than AAC for about 2 dB per subband for both cases.

For a typical 5.1 channel configuration, when packets of more than two channels are dropped, which implies that at least one channel pair's information is lost, some lost channel can no longer be concealed from the received AAC bitstream. In contrast, the MAACKLT bitstream can still be concealed to obtain a full 5.1 channel

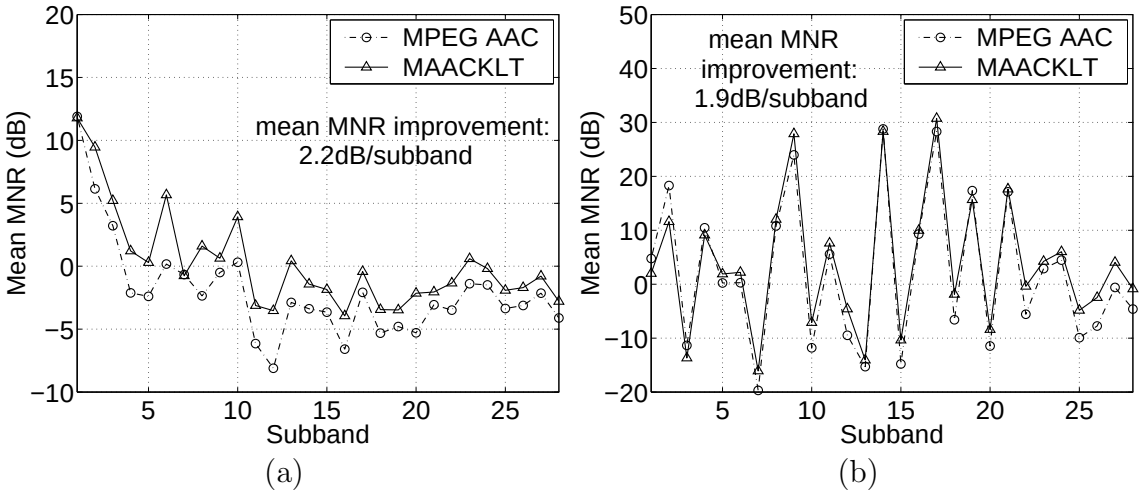


Figure 2.20: MNR comparison for 5-channel "Herre" when packets of one channel from the (a) L/R and (b) Ls/Rs channel pairs are lost.

audio with poorer quality. Although the recovered channel pairs do not sound exactly the same as the original ones, a reconstructed full multichannel audio would give the listener a much better acoustical effect than a three- or mono-channel audio.

Take the 5-channel "Messiah", which include C, L, R, Ls and Rs channels, as an example. At the worst case, when packets of four channels are dropped and only data of the most important channel are received at the decoder side, the MAACKLT algorithm can still recover 5-channel audio. Compared with the original sound, the recovered Ls and Rs channels lost most of the reverberant sound effect. This is because inverse KLT can only recover those information in the received channels. Since eigen-channel 1 does not contain much reverberant sound, the MAACKLT decoder can hardly recover these reverberant sound effects in the Ls and Rs channels.

Similar experiments were also performed by using test audio "Herre". However, the advantage of MAACKLT over AAC is not as obvious as test audio "Messiah".

The reason can be easily found out from the original covariance matrix as shown in Figure 2.2. It indicates that little correlation exists between SCE and CPE for class II test audio such as "Herre". Thus, once one CPE are lost, little information can be recovered from other CPEs or SCEs.

2.8.3 Subjective Listening Test

In order to further confirm the advantage of the proposed algorithm, a formal subjective listening test according to ITU recommendations [111, 128a, 128b] was conducted in an audio lab to compare the coding performance of the proposed MAACKLT algorithm and that of the MPEG AAC main profile codec. At the bit rate of 64 kbit/s/ch, the reconstructed sound clips are supposed to have the indistinguishable quality as the original ones, which means that the difference between MAACKLT and AAC would be small enough such that non-professionals can hardly tell. Therefore, instead of inviting a large number of non-expert listeners, four well-trained professionals participated in the listening test [128b]. During the test, for each test sound clips, subjects listened to three versions of the same sound clips, i.e. the original one followed by two processed ones (one by MAACKLT and one by AAC in random order), subjects were allowed to listen to these files as many times as possible until they were comfortable to give scores to the two processed sound files for each test material.

The five-grade impairment scale given in Recommendation ITU-R BS. 1284 [128a] was adopted in the grading procedure and utilized for final data analysis. Four multi-channel audio materials, i.e. "Messiah", "Band", "Herbie" and "Herre", are all used in this subjective listening test. According to ITU-R BS. 1161-1 [111], audio files selected for listening test only contains short durations (10 to 20 seconds long), so all test files coded by MAACKLT are generated by non-adaptive frequency-domain KLT method.

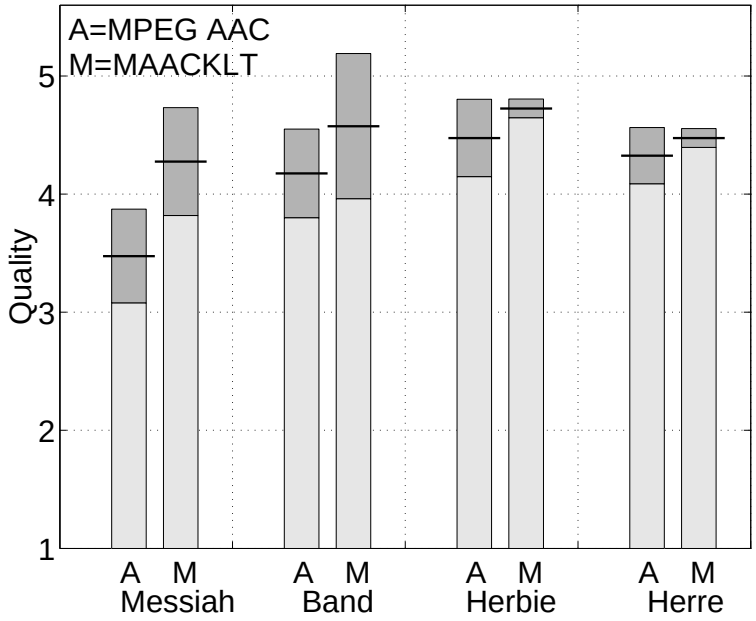


Figure 2.21: Subjective listening test results.

Figure 2.21 shows the listening test results, where bars represent the score given to each test material coded at 64 kbit/s/ch. The dark shaded area on the top of each bar represents the 95% confidence interval, where the middle line shows the mean value and the other two lines at the boundary of the dark shaded area represent the

upper and lower confidence limits [RADH87]. It is clear from Figure 2.21 that the proposed MAACKLT algorithm outperforms MPEG AAC in all four test materials.

2.9 Conclusion

We presented a new channel-scalable high-fidelity multichannel audio compression scheme called MAACKLT based on the existing MPEG-2 AAC codec. This algorithm explores the inter- and inner-channel correlation in the input audio signal and allows channel-scalable decoding. The compression technique utilizes KLT in the pre-processing stage to remove the inter-channel redundancy, then compresses the resulting relatively independent eigen-channel signals with a modified AAC main profile encoder module, and finally uses a prioritized transmission policy to achieve quality scalability. The novelty of this technique lies in its unique and desirable capability to adaptively vary the characteristics of the inter-channel de-correlation transform as a function of the covariance of a short period of music and its ability to reconstruct different quality audio with single bitstream. It achieves a good coding performance especially for the input audio source whose channel number goes beyond 5.1. In addition, it outperforms AAC according to both objective (MNR measurement) and subjective (listening) tests at the typical low bit rate of 64 kbit/s/ch while maintaining a similar computational complexity for both encoder and decoder modules. Moreover, compared with AAC, the channel-scalable property of MAACKLT

enables users to conceal full multichannel audio of reasonable quality without any additional cost.

Chapter 3

Adaptive Karhunen-Loève Transform and its Quantization Efficiency

3.1 Introduction

Based on today's most distinguished multichannel audio coding system, a Modified Advanced Audio Coding with Karhunen-Loève Transform (MAACKLT) method is proposed to perceptually losslessly compress a multichannel audio source in Chapter 2. This method utilizes the Karhunen-Loève Transform (KLT) in the preprocessing stage for the powerful multichannel audio compression tool, i.e. MPEG Advanced Audio Coding (AAC), to remove inter-channel redundancy and further improve the coding performance. However, as described in Chapter 2, each element of the covariance matrix, from which the KLT matrix is derived, is scalar quantized to 16 bits. This results in 240 bits overhead for each KL transform matrix for typical 5 channel audio contents. Since the bit budget is the most precious resource in the coding technique, every effort must be made to minimize the overhead due to

the additional pre-processing stage while maintaining a similar high-quality coding performance. Moreover, the original MAACKLT algorithm did not fully explore the KLT temporal adaptation effect.

In this research, we investigate the KLT de-correlation efficiency versus the quantization accuracy and the temporal KLT adaptive period. Extensive experiments on the quantization of the covariance matrix by using scalar and vector quantizers have been performed. Based on these results, the following two interesting points are concluded.

- Coarser quantization can dramatically degrade the de-correlation capability in terms of the normalized covariance matrix of de-correlated signals. However, the degradation of decoded multichannel audio quality is not as obvious.

- Shorter temporal adaptation of KLT will not significantly improve the de-correlation efficiency while considerably increase the overhead. Thus, a moderately long adaptation time is a good choice.

It is shown in this work that, with vector quantization, we can reduce the overhead from more than 200 bits to less than 3 bits per KL transform while maintaining comparable coding performance. Even with scalar quantization, a much lower overhead bit rate can still generate decoded audio with comparable quality. Our experimental results indicate that although a coarser quantization of the covariance matrix gives a poorer de-correlation effect, reduction of bits in the overhead is able

to compensate this degradation to result in a similar coding performance in terms of the objective MNR measurement.

The rest of this chapter[1] is organized as follows. In section 3.2 we introduce vector quantization and its application to the MAACKLT algorithm. In sections 3.3 and section 3.4 we explore how the quantization method and the temporal adaptive scheme affect the KLT de-correlation efficiency and the coding performance by applying scalar and vector quantizers to encode the KLT matrix with a range of different bit rates. In section 3.5 we examine computational complexity issues. Some experimental results are presented in section 3.6. Finally concluding remarks are given in section 3.7.

3.2 Vector Quantization

The MAACKLT algorithm described in Chapter 2 dealt only with scalar quantization of the covariance matrix. If the input audio material is short or if the KLT matrix is updated more frequently, the overhead that results from transmitting the covariance matrix will increase significantly, which will degrade the coding performance of the MAACKLT algorithm to a certain extent. To alleviate this problem, we have to resort to a look-up-table (LUT) approach. Here, a stored table of precalculated covariance matrices is searched to find the one that approximates the estimated covariance matrix of the current block of the input audio. This approach

[1]Part of this chapter represents work published before, see [YAKK01a]

yields a substantial savings in the overhead bit rate since only pointers to the table, instead of the entire covariance matrix itself, will be transmitted to the receiver.

Vector quantization (VQ) [GG91, PA93, Equ89] provides an excellent choice to implement the LUT idea. By vector quantization, we identify a set of possible vectors both at the encoder and the decoder side. They are called the codebook. The VQ encoder pairs up each source vector with the closest matching vector (i.e. "codeword") in the codebook, thus "quantizing" it. The actual encoding is then simply a process of sequentially listing the identity of codewords that match most closely with vectors making up the original data. The decoder has a codebook identical to the encoder, and decoding is a trivial matter of piecing together the vectors whose identity have been specified. Vector quantizers in this work consider the entire set of non-redundant elements of each covariance matrix as an entity, or a vector. The identified codebook should be general enough to include the characteristics of different types of multichannel audio sources. Since VQ allows for direct minimization of the quantization distortion, it results in smaller quantization distortion than scalar quantizers (SQ) at the same bit rate. In other words, VQ demands a smaller number of bits for source data coding while keeping the quantization error similar to that achieved with scalar quantizer.

Four different five-channel audio pieces (each containing center (C), left (L), right (R), left surround (Ls) and right surround (Rs) channels), are used to generate more than 80,000 covariance matrices. Each covariance matrix is treated as one training

vector $\mathbf{X}$, which is composed of fifteen non-redundant elements of the covariance matrix as shown below.

$$\mathbf{X} = \begin{matrix} x_1 & & & & \\ x_2 & x_3 & & & \\ x_4 & x_5 & x_6 & & \\ x_7 & x_8 & x_9 & x_{10} & \\ x_{11} & x_{12} & x_{13} & x_{14} & x_{15} \end{matrix},$$

where $x_1, x_2, \cdots, x_{15}$ are elements in the lower triangular part of the covariance matrix. During the codebook generation procedure, the Generalized Lloyd Algorithm (GLA) was run on the training sequence by using the simple square error distortion measurement, i.e.

$$d(\mathbf{X}, Q(\mathbf{X})) = \sum_{i=1}^{15} [\mathbf{X} - Q(\mathbf{X})]^2,$$

where $Q(\mathbf{X})$ represents the quantized value of $\mathbf{X}$. The same distortion measurement is used with the full searching method during the encoding procedure.

3.3 Efficiency Of KLT De-Correlation

The magnitudes of non-diagonal elements in a normalized covariance matrix provide a convenient metric to measure the degree of inter-channel correlation. The

normalized covariance matrix is derived from the cross-covariance matrix by multiplying each coefficient with the reciprocal of the square root of the product of their individual variance, i.e.

$$C_N(i,j) = \frac{C(i,j)}{\sqrt{C(i,i) \times C(j,j)}},$$

where $C_N(i,j)$ and $C(i,j)$ are elements of the normalized covariance matrix and the cross-covariance matrix in row i and column j, respectively.

Tables 3.1 and Table 3.2 show the absolute values of non-redundant elements (i.e. elements in only the lower or the upper triangle) of the normalized covariance matrix calculated from original signals and KLT de-correlated signals respectively, where no quantization is performed during the KLT de-correlation. From these tables, we can easily see that KLT reduces the inter-channel correlation from around the order of 10^{-1} to the order of 10^{-4}.

Table 3.1: Absolute values of non-redundant elements of the normalized covariance matrix calculated from original signals.

1				
5.36928147e-1	1			
3.26056331e-1	1.02651220e-1	1		
1.17594877e-1	8.56662289e-1	5.12340667e-3	1	
7.46899187e-2	1.33213668e-1	1.15962389e-1	6.55651089e-2	1

In order to investigate the de-correlation efficiency affected by various quantization schemes, a sequence of experiments, including SQ and VQ with a different

Table 3.2: Absolute values of non-redundant elements of the normalized covariance matrix calculated from KLT de-correlated signals.

1				
1.67971275e-4	1			
2.15059591e-4	1.01530173e-3	1		
4.19255484e-4	4.03864289e-4	2.56863610e-4	1	
3.07486032e-4	4.23535476e-4	3.48484672e-4	5.20389082e-5	1

Table 3.3: Absolute values of non-redundant elements of the normalized covariance matrix calculated from scalar quantized KLT de-correlated signals.

1				
1.67971369e-4	1			
2.15059518e-4	1.01530166e-3	1		
4.19255341e-4	4.03863772e-4	2.56863464e-4	1	
3.07486076e-4	4.23536876e-4	3.48484820e-4	5.20396538e-5	1

number of bits per element/vector, was performed. Table 3.3 shows the absolute values of non-redundant elements of the normalized covariance matrix calculated from KLT de-correlated signals, where each element of the covariance matrix is scalar quantized into 32 bits. Compared with Table 3.2, values in Table 3.3 are almost identical to those in Table 3.2 with less than 0.0001% distortion per element. This suggests that, with 32 bits per element scalar quantizer, we can almost faithfully reproduce the covariance matrix with negligible quantization error.

Figures 3.1 and Figure 3.2 illustrate how de-correlation efficiency and the corresponding overhead changes with SQ and VQ, respectively. It is observed that simple Mean Square Error (MSE) measurement is not a good choice when evaluating the

de-correlation efficiency. A better measure is the average distortion D, which is the summation of magnitudes of lower triangular elements of the normalized covariance matrix, i.e.

$$D = \sum_{i=2}^{N} \sum_{j=1}^{i-1} |C_N(i,j)|, \tag{3.1}$$

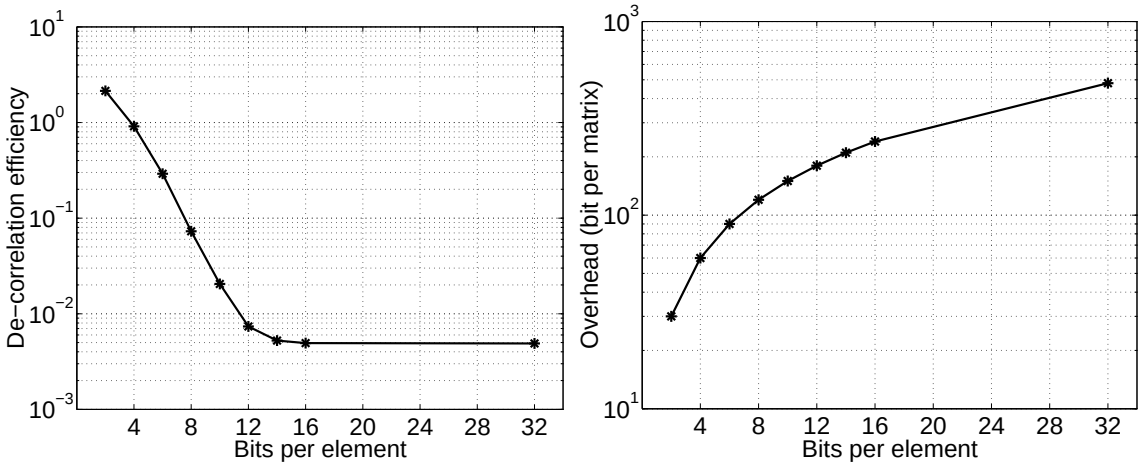
where C_N is the normalized covariance matrix of signals after KLT de-correlation. The overhead in terms of bits per KLT matrix is calculated via

$$OH_s = B_s \times N, \tag{3.2}$$

$$OH_v = B_v, \tag{3.3}$$

where B_s denotes the number of bits per element for SQ, N is the number of non-redundant elements per matrix, and B_v denotes the number of bits per codeword for VQ (recall that one KLT matrix is quantized into one codeword). For 5 channel audio material, N is equal to 15.

Figure 3.1 (a) suggests that there is no significant degradation in de-correlation efficiency when the number of bits is reduced from 32 bits per element to 14 bits per element. However, further reduction in the number of bits per element will result in a dramatic increase of distortion D given in Equation (3.1). From Equation (3.2), we know that the overhead increases linearly as the number of bits per element increases with a gradient equals to N. This is confirmed by Figure 3.1 (b), in which the overhead is plotted as a function of the number of bits per element in the logarithmic scale. Compared with Figure 3.1 (a), we observe that the overhead OH

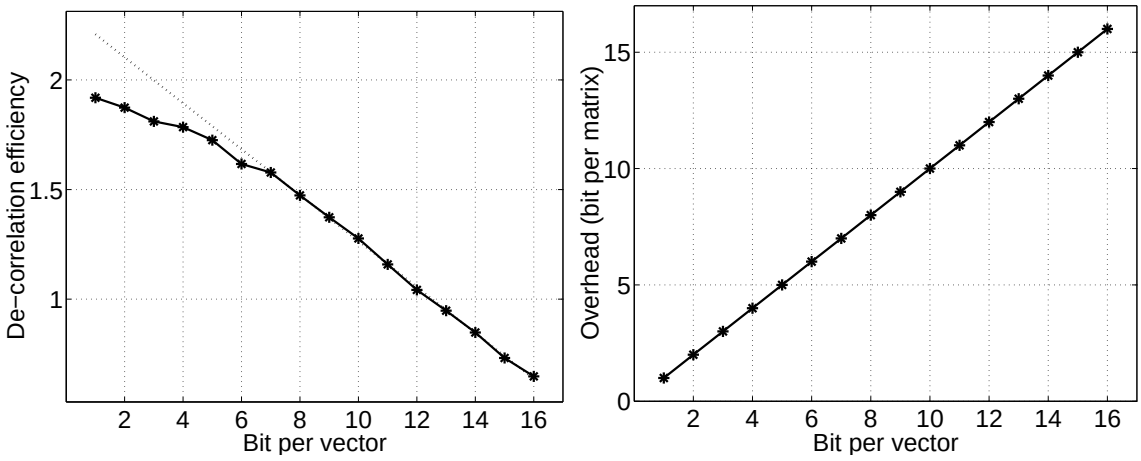


(a) (b)

Figure 3.1: (a) The de-correlation efficiency and (b) the overhead bit rate versus the number of bits per element in SQ.

increases much more rapidly than the decrease of the distortion D when the number of bits per element increases from 14 to 32. It indicates that when transmitting the covariance matrix with a higher bit rate, the improvement of de-correlation efficiency is actually not sufficient to compensate the loss due to a higher overhead rate.

The minimum number of bits per element for SQ is 2, since we need one bit for the sign and at least one bit for the absolute value for each element. To further reduce the number of bits per element can be achieved by using VQ. Figure 3.2 (a) illustrates that the average distortion D increases almost linearly when the number of bit per vector decreases from 16 bits per vector to 7 bits per vector, and then slows down when the bit per vector further decreases. Compared with Figure 3.1, it is verified that VQ results in smaller quantization distortion than SQ at any given bit rate. Figure 3.2 (b) shows how the overhead varies with the number of bits per



(a) (b)

Figure 3.2: (a) The de-correlation efficiency and (b) the overhead bit rate versus the number of bits per vector in VQ.

covariance matrix. Note that VQ reduces the overhead bit rate more than a factor of N (which is 15 for 5 channel audio) with respect to SQ.

Tables 3.4 and Table 3.5 show the average distortion D, the overhead information and the average MNR value for SQ and VQ, respectively, where the average MNR value is calculated as below:

$$\text{mean MNR}_{subband} = \frac{\sum_{channel} \text{MNR}_{channel,subband}}{\text{number of channels}}, \tag{3.4}$$

$$\text{average MNR} = \frac{\sum_{subband} \text{mean MNR}_{subband}}{\text{number of subband}}. \tag{3.5}$$

The test audio material used to generate results in this section is a 5 channel performance of "Messiah" with the KLT matrix updated every one second. As shown in Table 3.4, a fewer number of bits per element results in a higher distortion in

Table 3.4: De-correlation results with SQ.

bit/element	D	Overhead (bit/matrix)	Ave MNR (dB/sb)
2	2.14	30	N/A[a]
4	9.13e-1	60	56.56
6	2.91e-1	90	56.37
8	7.29e-2	120	56.02
10	2.05e-2	150	56.08
12	7.36e-3	180	56.00
14	5.24e-3	210	55.93
16	4.93e-3	240	55.91
32	4.89e-3	480	55.84

[a]Using 2 bits per element, which quantizes each element into values of either 0 or ± 1, leads to problems in later compression steps.

exchange of a smaller overhead and, after all, a larger MNR value. Thus, although a smaller number of bits per element of the covariance matrix results in a larger average distortion D, the decrease of the overhead bit rate actually compensates this distortion and improves the MNR value for the same coding rate.

A similar argument applies to the VQ case as shown in Table 3.5 with only one minor difference. That is, the MNR value is nearly a monotonic function for the SQ case while it moves up and down slightly in a local region (fluctuating within 1 dB per subband) for the VQ case. However, the general trend is the same, i.e. a larger overhead in KLT coding degrades the final MNR value. We also noticed that even when using 1 bit per vector, vector quantization of the covariance matrix still gives good MNR results.

Table 3.5: De-correlation results with VQ.

bit/vector	D	Overhead (bit/matrix)	Ave MNR (dB/sb)
1	1.92	1	56.73
2	1.87	2	56.61
3	1.81	3	56.81
4	1.78	4	56.87
5	1.73	5	56.12
6	1.62	6	56.23
7	1.58	7	56.88
8	1.47	8	56.96
9	1.37	9	56.42
10	1.28	10	55.97
11	1.16	11	56.08
12	1.04	12	56.28
13	0.948	13	55.83
14	0.848	14	55.72
15	0.732	15	56.19
16	0.648	16	55.87

Our conclusion is that it is beneficial to reduce the overhead bit rate used in the coding of the covariance matrix of KLT, since a larger overhead has a negative impact on the rate-distortion tradeoff.

3.4 Temporal Adaptation Effect

A multichannel audio program in general comprises of several different periods, each of which has its unique spectral signature. For example, a piece of music may begin

with a piano preclude followed by a chorus. In order to achieve the highest information compactness, the de-correlation transform matrix must adapt to the characteristics of different sectoins of the program material. The MAACKLT algorithm utilizes a temporal-adaptive approach, in which the covariance matrix is updated frequently. On one hand, the shorter the adaptive time, the more efficient the inter-channel de-correlation mechanism. On the other hand, since the KLT covariance matrix has to be coded for audio decoding, a shorter adaptive time contributes to a larger overhead in bit rates. Thus, it is worthwhile to investigate the tradeoff so that a good balance between this adaptive time and the final coding performance can be reached.

In Figure 3.3, we show the magnitude of the lower triangular elements of the normalized covariance matrix calculated from de-correlated signals by using different adaptive periods, where no quantization has been applied yet. These figures suggest that there is no significant improvement of the de-correlation efficiency when the KLT adaptive time decreases from 10 seconds to 0.05 second. As the overhead dramatically increases with the shorter adaptive time, the final coding performance may be degraded. In order to find the optimal KLT adaptive time, a thorough investigation is performed for both SQ and VQ.

First, let us look at how adaptive time affects the overhead bit rate. Suppose n channels are selected for simultaneous inter-channel de-correlation, the adaptive

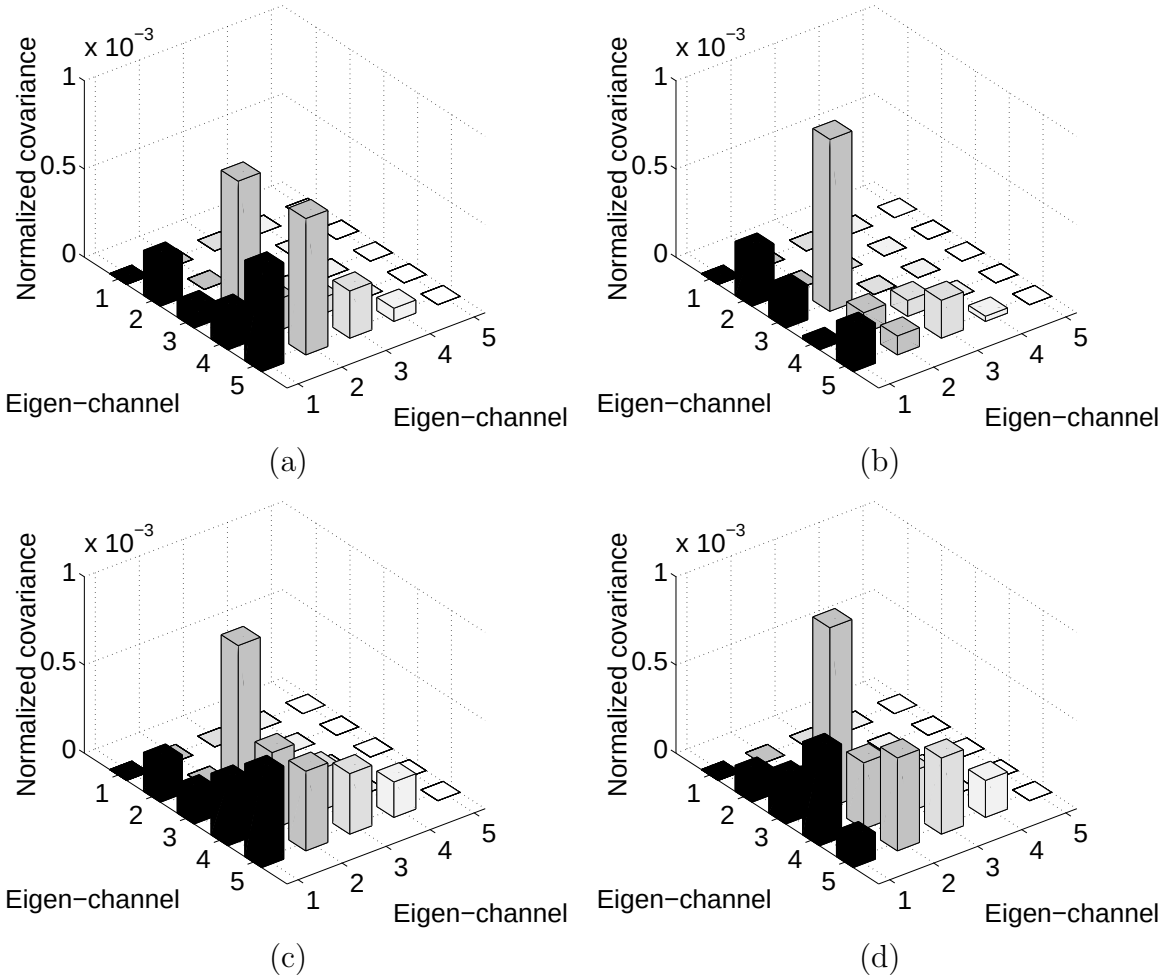


Figure 3.3: The magnitude of the lower triangular elements of the normalized covariance matrix calculated from de-correlated signals, where the adaptive time is equal to (a) 0.05, (b) 0.2, (c) 3, and (d) 10 seconds.

time is K seconds, i.e. each sub-block contains K seconds of audio, and M bits are transmitted to the decoder for each KL transform. The overhead bit rate $r_{overhead}$ is

$$r_{overhead} = \frac{M}{nK} \tag{3.6}$$

in bits per second per channel (bit/s/ch). This equation suggests that the overhead bit rate increases linearly with the number of bits used to encode and transmit the KL

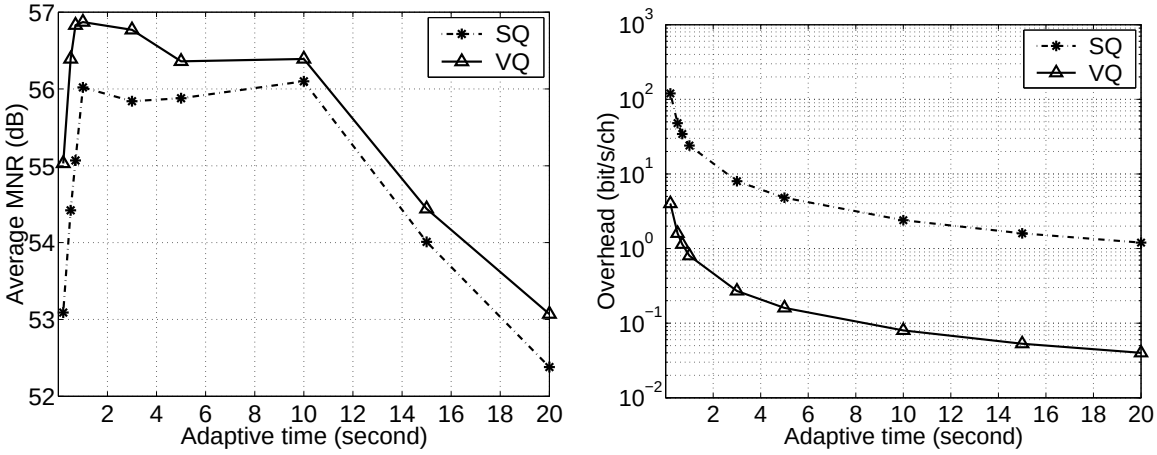
transform matrix. The overhead bit rate is, however, inversely proportional to the number of channels and the adaptive time. If SQ is used in the encoding procedure, each non-redundant element has to be sent. For n channel audio material, the size of the covariance matrix is $n \times n$, and the number of non-redundant elements is $n \times (n+1)/2$. If B_s bits are used to quantize each element, the total bit requirement for each KLT is $n(n+1)B_s/2$. Thus the overhead bit rate $r^{SQ}_{overhead}$ for SQ is equal to

$$r^{SQ}_{overhead} = \frac{(n+1)B_s}{2K}.$$

The overhead bit rate $r^{vQ}_{overhead}$ for VQ is simpler. It is equal to

$$r^{VQ}_{overhead} = \frac{B_v}{nK},$$

where B_v represent the number of bits used for each KLT covariance matrix.



(a) (b)

Figure 3.4: (a) Adaptive MNR results and (b) adaptive overhead bits for SQ and VQ for 5-channel Messiah.

The average MNR value (in dB) and the overhead bit rate (in the logarithm scale) versus the adaptive time for both SQ and VQ are shown in Figures 3.4 (a) and (b), respectively. The test material is 5-channel "Messiah", with 8 bits per element for SQ and 4 bits per vector for VQ for KLT de-correlation. The total coding bit rate (including bits for the overhead and the content) is kept the same for all points in the two curves in Figure 3.4 (a). We have the following observations from these figures.

First, for the SQ case, the average MNR value remains about the same with less than 0.3 dB variation per subband when the adaptive time varies from 1 to 10 seconds. However, when the adaptive time is decreased furthermore, the overhead effect starts to dominate, and the average MNR value decreases dramatically. On the other hand, when the adaptive time becomes larger than 10 seconds, the average MNR value also decreases, which implies that the coding performance degrades if the KLT matrix is not updated frequently enough. For VQ, the changing pattern of the average MNR value versus the adaptive time is similar as that of SQ. However, compared with the scalar case, the average MNR value starts to degrade earlier at about 5 seconds. This is probably due to the effect that VQ gives less efficient de-correlation, so that more frequent adaptation of the KLT matrix will generate a better coding result. As shown in Figure 3.4 (a), it is clear that the average MNR generated by using VQ is always better than that of SQ and the difference becomes significant when the overhead becomes the dominant factor for KLT adaptive time less than 1 second.

3.5 Complexity Analysis

The main concern of a VQ scheme is its computational complexity at the encoder side. For each D dimension vector, we need $O(DS)$ operations to find the best matched codeword from a codebook of size S using the full search technique. For a n channel audio, each covariance matrix is represented by a vector of dimension $n(n+1)/2$. Thus, for each KLT, we need $O(n^2S)$ operations. While for the scalar case, to quantize each element requires $O(1)$ operations, and for each covariance matrix, we need $O(n^2)$ operations. Suppose that the input audio is of L seconds long, and the KLT matrix is updated each K seconds. Then, there will be totally $\lceil L/K \rceil$ covariance matrices to be quantized[2]. This means we need

$$\begin{aligned} O(\lceil L/K \rceil n^2) &= O(Ln^2/K) \quad \text{for scalar quantization,} \\ O(\lceil L/K \rceil n^2 S) &= O(Ln^2 S/K) \quad \text{for vector quantization,} \end{aligned}$$

operations.

Thus, for a given test audio source, the complexity is inversely proportional to KLT adaptive time for either quantization scheme. For VQ, the complexity is also proportional to the codebook size. Compared with SQ, VQ requires more operations by a factor of S. To reduce the computational complexity, we should limit the codebook size and set the KLT adaptation time as long as possible while keeping the desired coding performance.

[2]where $\lceil * \rceil$ represents the smallest integer which is greater than or equal to $*$.

Experimental results shown in Section 3.3 suggests that a very small codebook size is usually good enough to generate the desired compressed audio. By choosing a small codebook size and keeping the KLT adaptation time long enough, we do not only limit the additional computational complexity, but also save the overhead bit requirement. At the decoder side, VQ demands just a table look up procedure and its complexity is comparable to that of SQ.

3.6 Experimental Results

We tested the modified MAACKLT method by using two five-channel audio sources "Messiah" and "Ftbl" at different bit rates varying from a typical rate of 64 kbit/s/ch to a very low bit rate of 16 kbit/s/ch. Figures 3.5 and Figure 3.6 show the mean MNR comparison between SQ and VQ for test audio "Messiah" and "Ftbl", respectively, where the KLT matrix adaptation time is set to 10 seconds. The mean MNR values in these figures are calculated by Equation (3.4). In order to show the general result of scalar and vector cases, a moderate bit rate (i.e. 8 bits per element for SQ and 4 bits per vector for VQ) is adopted here. From these figures, we see that, compared with SQ, VQ generates comparable mean MNR results at all bit rates, and VQ even outperforms SQ at all bit rates for some test sequence such as "Messiah".

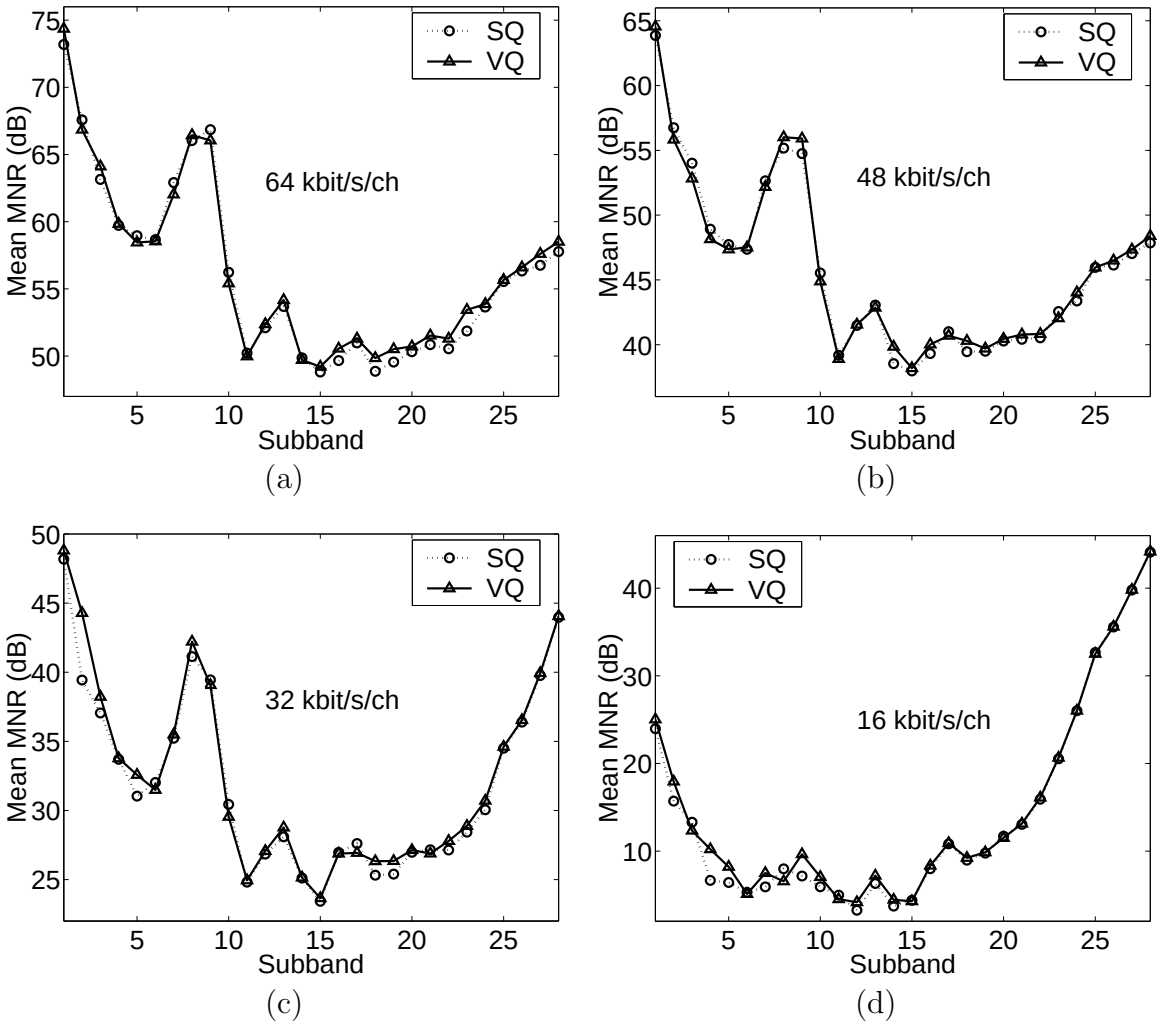


Figure 3.5: MNR result using test audio "Messiah" coded at (a) 64 kbit/s/ch, (b) 48 kbit/s/ch, (c) 32 kbit/s/ch, and (d) 16 kbit/s/ch.

3.7 Conclusion

To enhance the MAACKLT algorithm proposed earlier, we examined the relationship between coding of the KLT covariance matrix with different quantization methods, KLT de-correlation efficiency and the frequency of KLT information update extensively in this research. In specific, we investigated how different quantization method affects the final coding performance by using objective MNR measurements. It is

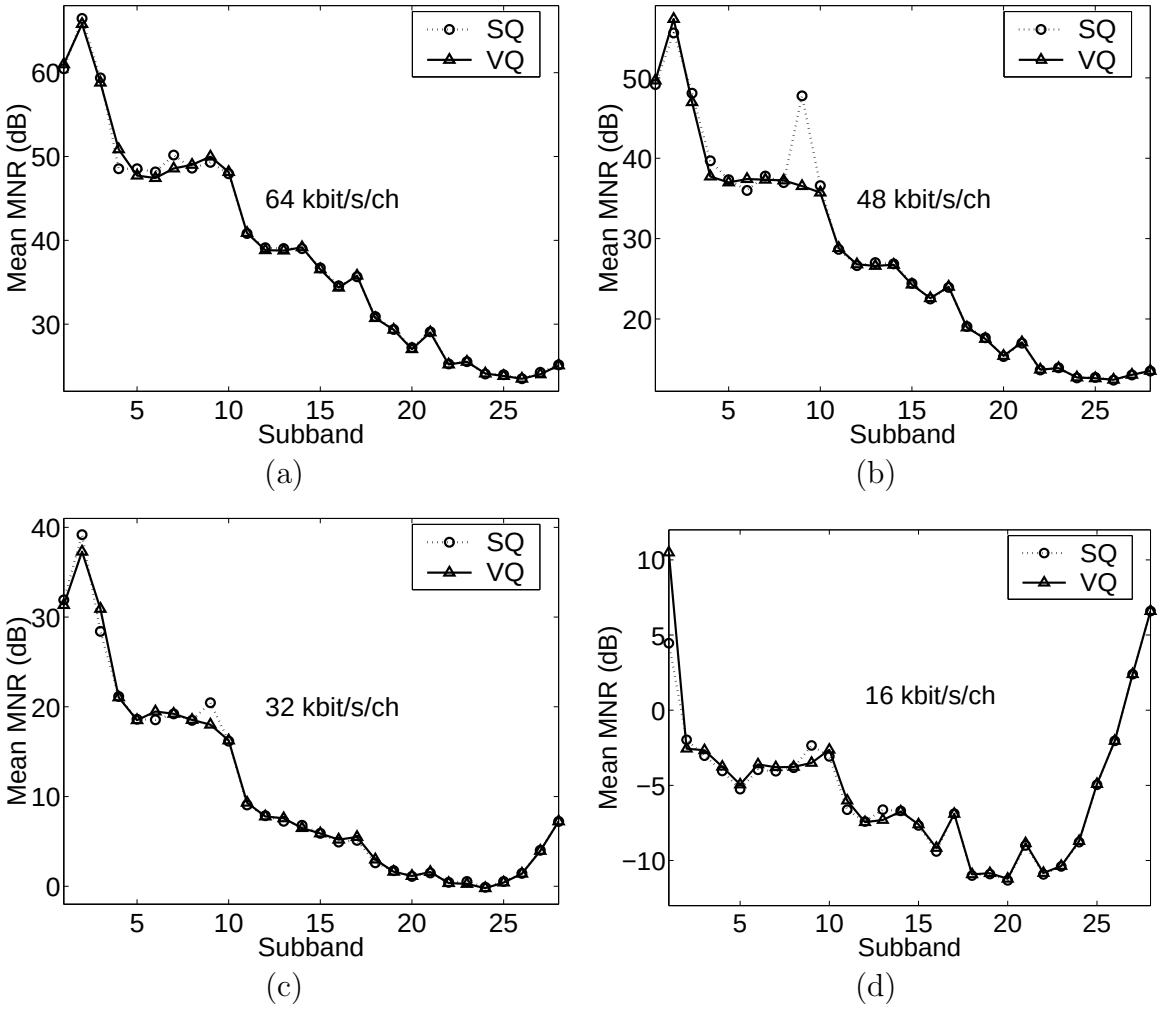


Figure 3.6: MNR result using test audio "Ftbl" coded at (a) 64 kbit/s/ch, (b) 48 kbit/s/ch, (c) 32 kbit/s/ch, and (d) 16 kbit/s/ch.

demonstrated that to reduce the overhead bit rate generally provides a better trade-off for the overall coding performance. This can be achieved by adopting a smallest possible bit rate to encode the covariance matrix together with moderately long KLT adaptation period to generate the desired coding performance. Besides, a small codebook size in VQ do not increase the computational complexity significantly.

Chapter 4

Progressive Syntax-Rich Multichannel Audio Codec

4.1 Introduction

Most of today's multichannel audio codecs can only provide bitstreams with a fixed bit rate, which is specified during the encoding phase. When this kind of bitstream is transmitted over variable bandwidth networks, the receiver can either successfully decode the full bitstream or ask the encoder site to re-transmit a bitstream with a lower bit rate. The best solution to address this problem is to develop a scalable compression algorithm, which is able to transmit and decode the bitstream with a bit rate that can be adaptive to a dynamically varying environment (e.g. the instantaneous capacity of a transmission channel). This capability offers a significant advantage in transmitting contents over channels with a variable channel capacity or connections for which the available channel capacity is unknown at the time of encoding. To achieve this goal, a bitstream generated by scalable coding schemes

consists of several partial bitstreams, each of which can be decoded on their own in a meaningful way. In this way, transmission and decoding of a subset of the total bitstream will result in a valid decodable signal at a lower bit rate and quality.

MPEG-4 version-2 audio coding supports fine grain bit rate scalablility [PKKS97, ISOb, ISOg, ISOh, HAB+98] in its Generic Audio Coder (GAC). It has a Bit-Sliced Arithmetic Coding (BSAC) tool, which provides scalability in the step of 1 kbit/s per audio channel for mono or stereo audio material. Several other scalable mono or stereo audio coding algorithms [ZL01, VA01, SAK99] were proposed in recent years. However, not much work has been done on progressively transmitting multichannel audio sources. In this work, we propose a progressive syntax-rich multichannel audio codec (PSMAC) based on MPEG AAC. In PSMAC, the inter-channel redundancy inherent in original physical channels is first removed in the pre-processing stage by using the Karhunen-Loève Transform (KLT). Then, most coding blocks in the AAC main profile encoder are employed to generate spectral coefficients. Finally, a progressive transmission strategy and a context-based QM coder are adopted to obtain the fully quality-scalable multichannel audio bitstream. The PSMAC system not only supports fine grain bit rate scalability for the multichannel audio bitstream, but also provides several other desirable functionalities, such as random access and channel enhancement, which have not been supported by other existing multichannel audio codecs.

Moreover, compared with the BSAC tool provided in MPEG-4 version-2 and most of other scalable audio coding tools, a more sophisticated progressive transmission strategy is employed in PSMAC. PSMAC does not only encode spectral coefficients from MSB to LSB and from low to high frequency so that the decoder can reconstruct these coefficients more and more precisely with an increasing bandwidth as the receiver collects more and more bits from the bitstream, but also utilize the psychoacoustic model to control the subband transmission sequence so that the most sensitive frequency area is more precisely reconstructed. In this way, bits used to encode coefficients in those non-sensitive frequency area can be saved and used to encode coefficients in the sensitive frequency area. Compared with the algorithm without this subband selection strategy, a perceptually more appealing audio can be reconstructed, especially at very low bit rate such as 16 kbit/s/ch. The side information required to encode the subband transmission sequence is nicely handled in our implementation so that the overall overhead will not have significant impact on the audio quality even at very low bit rates. Note that Shen *et al.* [SAK99] proposed a subband selection rule to achieve progressive coding. However, Shen's scheme demands a large amount of overhead in coding the selection order.

Experimental results show that, when compared with MPEG AAC, the decoded multichannel audio generated by the proposed PSMAC's MNR progressive mode has comparable quality at high bit rates, such as 64 kbits/s/ch or 48 kbits/s/ch and much better quality at very low bit rates, such as 32 kbits/s/ch or 16 kbits/s/ch. We also demonstrate that our PSMAC codec can provide better quality of single channel

audio when compared with MPEG-4 version-2 generic audio coder at several different bit rates.

The rest of this chapter[1] is organized as follows. Sections 4.2 gives an overview of the proposed syntax-rich design. Section 4.3 and Section 4.4 describe how the progressive quantization is employed in our system. Section 4.5 discusses some implementation issues. Section 4.6 illustrates the complete compression system. Some experimental results are shown in Section 4.7. Finally, conclusion remarks are given in Section 4.8.

4.2 Progressive Syntax-Rich Codec Design

In the proposed progressive syntax-rich codec, the following three user defined profiles are provided.

1. MNR Progressive:

   If this flag is on, it should be possible to decode the first N bytes of the bitstream, where N in term of bit rate is a user specified value or a value that the current network parameters allowed.

2. Random Access:

   If this flag is present, the codec will be able to independently encode a short period of audio more precisely than other periods. It allows users to randomly access a certain part of audio that is more of interest to end users.

[1]Part of this chapter represents work published before, see [YAKK01b, YAKK02a, YAK02]

3. Channel Enhancement:

   If this flag is on, the codec will be able to independently encode an audio channel more precisely than other channels. Either these channels are more of interest to end users or the network situation does not allow the full multi-channel audio bitstream to be received in time.

The MNR progressive profile is the default mode. For the other two profiles, i.e. random access mode and channel enhance mode, the MNR progressive feature is still provided as a basic functionality and decoding of the bitstream can be stopped at any arbitrary point. With these three profiles, the codec provides a versatile functionality that is desired in a variable bandwidth network condition with different user access bandwidth.

4.3 Scalable Quantization and Entropy Coding

The major difference between the proposed progressive audio codec and other existing non-progressive audio codecs such as AAC lies in the quantization module and the entropy coding module. The dual iteration loop used in AAC to calculate the quantization step size for each frame's and each channel's coefficients is replaced by a progressive quantization block. The huffman coding module used in the AAC to encode quantized data is replaced by a context-based QM coder. They will be explained in detail below.

4.3.1 Successive Approximation Quantization (SAQ)

The most important component of the quantization module is called successive approximation quantization (SAQ). The SAQ scheme, which is adopted by most embedded wavelet coders for progressive image coding, is crucial to the design of embedded coders. The motivation for successive approximation is built upon the goal of developing an embedded code that is in analogy to find an approximation of binary-representation to a real number [Sha93]. Instead of coding every quantized coefficient as one symbol, SAQ processes the bit representation of coefficients via bit layer sliced in the order of their importance. Thus, SAQ provides a coarse-to-fine, multiprecision representation of the amplitude information. The bitstream is organized such that a decoder can immediately start reconstruction based on the partial received bitstream. As more and more bits are received, more accurate coefficients and higher quality multichannel audio can be reconstructed.

4.3.1.1 Description of the SAQ Algorithm

SAQ sequentially applies a sequence of thresholds $T_0, T_1, \ldots, T_{N+1}$ for refined quantization, where these thresholds are chosen such that $T_i = T_{i-1}/2$. The initial threshold T_0 is selected such that $|C(i)| < 2T_0$ for all transformed coefficients in one subband, where $C(i)$ represents the i^{th} spectral coefficient in the subband. To implement SAQ, two separate lists, the dominant list and the subordinate list, are maintained both at the encoder and the decoder sides. At any point of the process, the dominant list contains the coordinates of those coefficients that have not

yet been found to be significant. While the subordinate list contains magnitudes of those coefficients that have been found to be significant. The process that updates the dominate list is called the significant pass, and the process that updates the subordinate list is called the refinement pass.

In the proposed algorithm, SAQ is adopted as the quantization method for each spectral coefficient within each subband. This algorithm, is listed below.

Successive Approximation Quantization (SAQ) Algorithm

1. Initialization:

   For each subband, find out the maximum absolute value $C_{\max}$ for all coefficients $C(i)$ in the subband, and set the initial quantization threshold to be $T_0 = C_{\max}/2 + BIAS$, where $BIAS$ is a small constant.

2. Construction of the significant map (significance identification):

   For each $C(i)$ contained in the dominant list, if $|C(i)| \geq T_k$, where T_k is the threshold of the current layer (layer k), add i to the significant map, remove i from the dominant list and encode it with $'1s'$, where $'s'$ is the sign bit. Moreover, modify the coefficient's value to

$$C(i) \leftarrow \begin{cases} C(i) - 1.5 \times T_k, & \forall C(i) > 0 \\ C(i) + 1.5 \times T_k, & \text{otherwise} \end{cases}$$

3. Construction of the refinement map (refinement):

   For each $C(i)$ contained in the significant map, encode the bit at layer k with a refinement bit $'D'$ and change the value of $C(i)$ to

$$C(i) \leftarrow \begin{cases} C(i) - 0.25 \times T_k, & \forall C(i) > 0 \\ C(i) + 0.25 \times T_k, & \text{otherwise} \end{cases}$$

4. Iteration:

   Set $T_{k+1} = T_k/2$ and repeat Steps 2-4 for $k = 0, 1, 2, \ldots$

4.3.1.2 Analysis of Error Reduction Rates

The following two points have been observed before [K*et al.*97]:

- The coding efficiency of the significant map is always better than that of the refinement map at the same layer.
- The coding efficiency of the significant map at the k^{th} layer is better than that of the refinement map at the $(k-1)^{th}$ layer.

In the following, we would like to provide a formal proof by analyzing the error reduction capability due to the significant pass and the refinement pass, respectively.

First, let us consider the error reduction capability for the bit-layer coding of coefficient $C(i)$, $\forall i$, in the significant pass. Since the sign of each coefficient will be coded separately, we will assume $C(i) > 0$ below without loss of generality. Suppose that $C(i)$ becomes significant at layer k. This means $T_k \leq C(i) < T_{k-1} = 2T_k$ and

its value is modified accordingly. Then, error reduction Δ_1 due to the coding of this bit can be found as

$$\Delta_1 = C(i) - |C(i) - 1.5 \times T_k|.$$

Note that, at any point of the process, the value of $|C(i)|$ is nothing else but the remaining coding error. Since $T_k \leq C(i) < 2T_k$, $-0.5T_k < C(i) - 1.5T_k \leq 0.5T_k$, we have $|C(i) - 1.5T_k| \leq 0.5T_k$. Consequently,

$$\Delta_1 = C(i) - |C(i) - 1.5 \times T_k| \geq 0.5T_k.$$

Now, let us calculate the error reduction for the bit-layer coding of coefficient $C(j), \forall j$, in the refinement pass. Similar to the previous case, we assume $C(j) > 0$. At layer k, suppose $C(j)$ is being refined, and its value is modified accordingly. The corresponding error reduction is

$$\Delta_2 = C(j) - |C(j) - 0.25 \times T_k|.$$

Two cases have to be considered:

1. If $C(j) \geq 0.25T_k$,

$$\Delta_2 = C(j) - C(j) + 0.25T_k = 0.25T_k.$$

2. If $C(j) < 0.25T_k$,

$$\Delta_2 = C(j) + C(j) - 0.25T_k = 2C(j) - 0.25T_k < 0.5T_k - 0.25T_k = 0.25T_k.$$

Thus, we conclude that

$$\Delta_2 = C(j) - |C(j) - 0.25 \times T_k| \leq 0.25T_k < 0.5T_k \leq \Delta_1.$$

Thus, the error reduction for significant pass is always greater than that of the refinement pass at the same layer.

Similarly, at layer $(k-1)$, the error reduction for coefficient $C(j), \forall j$, caused by the refinement pass is

$$\Delta_3 = C(j) - |C(j) - 0.25 \times T_{k-1}| \leq 0.25T_{k-1} = 0.5T_k \leq \Delta_1,$$

which demonstrates that error reduction in the significant pass at layer k is actually greater than or equal to that of the refinement pass at layer $(k-1)$.

According to the above analysis, a refinement-significant map coding is proposed and adopted in our progressive multichannel audio codec. That is, the transmission of k^{th} refinement map of subband i is followed immediately by the transmission of $(k+1)^{th}$ significant map of subband i.

4.3.1.3 Analysis of Error Bounds

Suppose the i^{th} coefficient $C(i)$ has a value $T_0/2^{R+1} \leq |C(i)| < T_0/2^R$. Then, its binary representation can be written as

$$\begin{aligned} C(i) &= \text{sign} \times [a_0(\frac{T}{2^0}) + a_1(\frac{T}{2^1}) + a_2(\frac{T}{2^2}) + \ldots] \\ &= \text{sign} \times \sum_{k=0}^{\infty} a_k(\frac{T}{2^k}), \end{aligned}$$

where $T = T_0/2^{R+1}$, T_0 is the initial threshold, and a_0, a_1, a_2, ... are binary values (either 0 or 1).

In the SAQ algorithm, $C(i)$ is represented by:

$$\begin{aligned} C(i) &= \text{sign} \times [1.5a_0(\frac{T}{2^0}) + 0.5b_1(\frac{T}{2^1}) + 0.5b_2(\frac{T}{2^2}) + \ldots] \\ &= \text{sign} \times [1.5a_0(\frac{T}{2^0}) + 0.5\sum_{k=1}^{\infty} b_k(\frac{T}{2^k})], \end{aligned}$$

where a_k and b_k are related via

$$a_k = 0.5(b_k + 1), \ \forall k = 1, 2, 3, \ldots,$$

or

$$b_k = \begin{cases} 1, & a_k = 1, \\ -1, & a_k = 0, \end{cases} \quad \forall k = 1, 2, 3, \ldots$$

Based on the first $M+1$ bits $a_0, a_1, a_2, \ldots, a_M$, the reconstructed value $R_1(i)$ by using the binary representation is

$$\begin{aligned} R_1(i) &= \text{sign} \times [a_0(\frac{T}{2^0}) + a_1(\frac{T}{2^1}) + a_2(\frac{T}{2^2}) + \ldots + a_M\frac{T}{2^M}] \\ &= \text{sign} \times [\sum_{k=0}^{M} a_k(\frac{T}{2^k})]. \end{aligned}$$

Based on the first $M+1$ bits $a_0, b_1, b_2, \ldots, b_M$, the reconstructed value $R_2(i)$ by using SAQ is

$$\begin{aligned} R_2(i) &= \text{sign} \times [1.5a_0(\frac{T}{2^0}) + 0.5b_1(\frac{T}{2^1}) + 0.5b_2(\frac{T}{2^2}) + \ldots + 0.5b_M\frac{T}{2^M}] \\ &= \text{sign} \times [1.5a_0(\frac{T}{2^0}) + 0.5\sum_{k=1}^{M} b_k(\frac{T}{2^k})]. \end{aligned}$$

Thus, the error introduced by the binary representation for this coefficient is

$$\begin{aligned} E_1(i) &= |C(i) - R_1(i)| = |\sum_{k=M+1}^{\infty} a_k\frac{T}{2^k}| \\ &\leq \sum_{k=M+1}^{\infty} \frac{T}{2^k} = \frac{T}{2^M}. \end{aligned}$$

Similarly, the error introduced by SAQ for this coefficient is

$$\begin{aligned} E_2(i) &= |C(i) - R_2(i)| = |0.5 \times \sum_{k=M+1}^{\infty} b_k\frac{T}{2^k}| \\ &\leq 0.5\sum_{k=M+1}^{\infty} \frac{T}{2^k} = \frac{T}{2^{M+1}}. \end{aligned}$$

We conclude that the upper bound of both error $E_1(i)$ caused by the binary representation and error $E_2(i)$ cause by SAQ are decaying exponentially when the incoming number M of bits is increasing linearly.

4.3.2 Context-based QM coder

The QM coder is a binary arithmetic-coding algorithm designed to encode data formed by a binary symbol set. It was the result of the effort by JPEG and JBIG committees, in which the best features of various arithmetic coders are integrated. The QM coder is a lineal descendent of the Q-coder, but significantly enhanced by improvements in the two building blocks, i.e. interval subdivision and probability estimation [PM93]. Based on the Bayesian estimation, a state-transition table, which consists of a set of rules to estimate the statistics of the bitstream depending on the next incoming symbols, can be derived. The efficiency of the QM coder can be improved by introducing a set of context rules. The QM arithmetic coder achieves a very good compression result if the context is properly selected to summarize the correlation between coded data.

Six classes of contexts are used in the proposed embedded audio codec as shown in Figure 4.1. They are the general context, the constant context, the subband significance context, the coefficient significance context, the coefficient refinement context and the coefficient sign context. The general context is used in the coding of the configuration information. The constant context is used to encode different

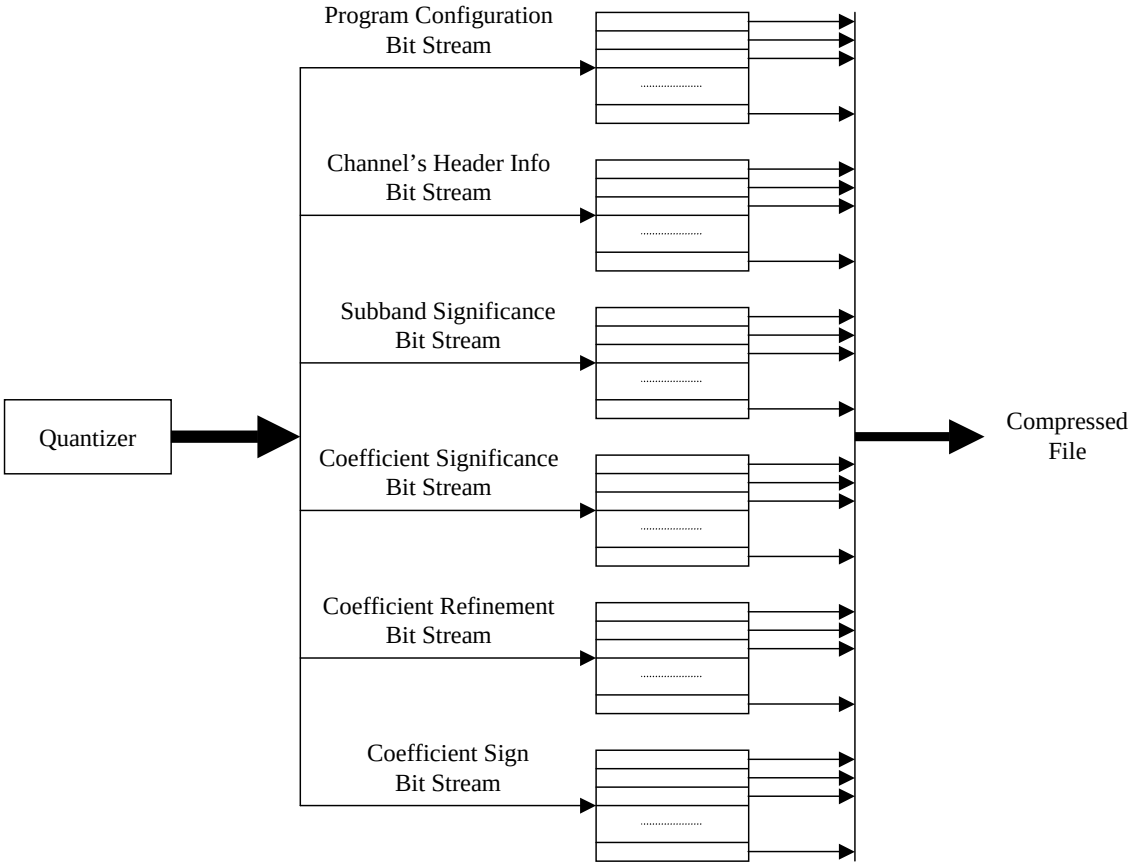


Figure 4.1: The adopted context-based QM coder with six classes of contexts.

channel's header information. As their names suggest, the subband significance context, the coefficient significance context, the coefficient refinement context and the coefficient sign context are used to encode the subband significance, coefficient significance, coefficient refinement and coefficient sign bits, respectively. These contexts are adopted because different classes of bits may have different probability distributions. In principle, separating their contexts should increase the coding performance of the QM coder.

4.4 Channel and Subband Transmission Strategy

4.4.1 Channel Selection Rule

In the embedded multichannel audio codec, we should put the most important bits (in the rate-distortion sense) to the cascaded bitstream first so that the decoder can reconstruct the optimal quality of multichannel audio given a fixed number of bit received. Thus, the importance of channels should be determined for an appropriate ordering of the bitstream. For the normal 5.1 channel configuration, it was observed in Chapter 2 that the channel importance will be eigen-channel 1, followed by eigen-channels 2 and 3, and then followed by eigen-channels 4 and 5. Between each channel pair, the importance is determined by their energy. This policy is used in this paper.

4.4.2 Subband Selection Rule

In principle, any quality assessment of an audio channel can be either performed subjectively by employing a large number of expert listeners or done objectively by using an appropriate measuring technique. While the first choice tends to be an expensive and time-consuming task, the use of objective measures provides quick and reproducible results. An optimal measuring technique would be a method that produces the same results as subjective tests while avoiding all problems associated with the subjective assessment procedure. Nowadays, the most prevalent objective measurement is the Mask-to-Noise-Ratio (MNR) technique, which was first introduced by Brandenburg [Bra87] in 1987. It is the ratio of the masking threshold with

respect to the error energy. In our implementation, the masking is calculated from the general psychoacoustic model of the AAC encoder. The psychoacoustic model calculates the maximum distortion energy which is masked by the signal energy, and outputs the Signal-to-Mask-Ratio (SMR).

A subband is masked if the quantization noise level is below the masking threshold so the distortion introduced by the quantization process is not perceptible to human ears. As discussed earlier, SMR represents the human auditory response to the audio signal. If SNR of an input audio signal is high enough, the noise level will be suppressed below masking threshold and the quantization distortion will not be perceived. Since SNR can be easily calculated by

$$SNR = \frac{\sum_i |S_{\text{original}}(i)|^2}{\sum_i |S_{\text{original}}(i) - S_{\text{reconstruct}}(i)|^2},$$

where $S_{\text{original}}(i)$ and $S_{\text{reconstruct}}(i)$ represent the i^{th} original and the i^{th} reconstructed audio signal value, respectively. Thus, MNR is just the difference of SNR and SMR (in dB), or

$$SNR = MNR + SMR.$$

A side benefit of the SAQ technique is that an operational rate vs. distortion plot (or, equivalently, an operational rate vs. the current MNR value) for the coding algorithm can be computed on-line.

The basic ideas behind choosing the subband selection rules are simple. They are:

1. The subband with a better rate deduction capability should be chosen earlier to improve the performance.

2. The subband with a smaller number of coefficients should be chosen earlier to reduce the computational complexity, if the rate reduction performances of two subbands are close.

The first rule implies that we should allocate more bits to those subbands with larger SMR values or smaller MNR values. In other words, we should send out bits belonging to those subbands with larger SMR or smaller MNR values first. The second rule tells us how to decide the subband scanning order. As we know about the subband formation in MPEG AAC, the number of coefficients in each subband is non-decreasing with the increase of the subband number. Figure 4.2 shows the subband width distribution used in AAC for 44.1 kHz and 48 kHz sampling frequencies and long block frames. Thus, a sequential subband scanning order from the lowest number to the highest number is adopted in this work.

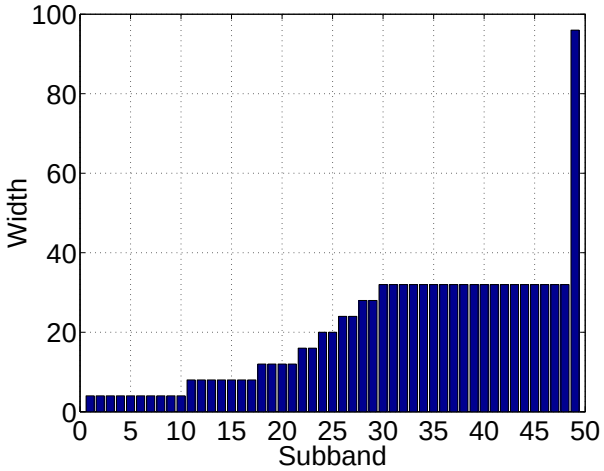


Figure 4.2: Subband width distribution.

In order to save bits, especially at very low bit rates, only information corresponding to lower subbands will be sent into the bitstream. And when number of layer increases, more and more subbands will be added. Figure 4.3 shows how subbands are scanned for the first several layers. At base layer, priority are given to lower frequency signals, so only subbands numbered upto L_B will be scanned. As enhance layers' information goes into the bitstream, the subband scanning upper limit, i.e. L_{E1}, L_{E2} and L_{E3}, increases, until this limit reach the effective psychoacoustic upper bound of all subbands N. In our implementation, $L_{E3} = N$. Which means after the third enhance layer, all subband will be scanned. Here subband scanning upper limits L_B, L_{E1} and L_{E2} are empirically determined values which compromise the coding efficiency with the coding performance.

In PSMAC, a dual-threshold coding technique is proposed in the progressive quantization module. One is the MNR threshold, which is used in subband selection. The other is the magnitude threshold, which is used for coefficients' quantization in each selected individual subband. A subband which has its current MNR value smaller than the current MNR threshold is called significant subband. Similar as the SAQ process for coefficient quantization, two lists, i.e. the dominant subband list and the subordinate subband list, are maintained in the encoder and the decoder sides. The dominant subband list contains the indices of those subbands that have not become significant, and the subordinate subband list contains the indices of those subbands that have already become significant. The process that updates the

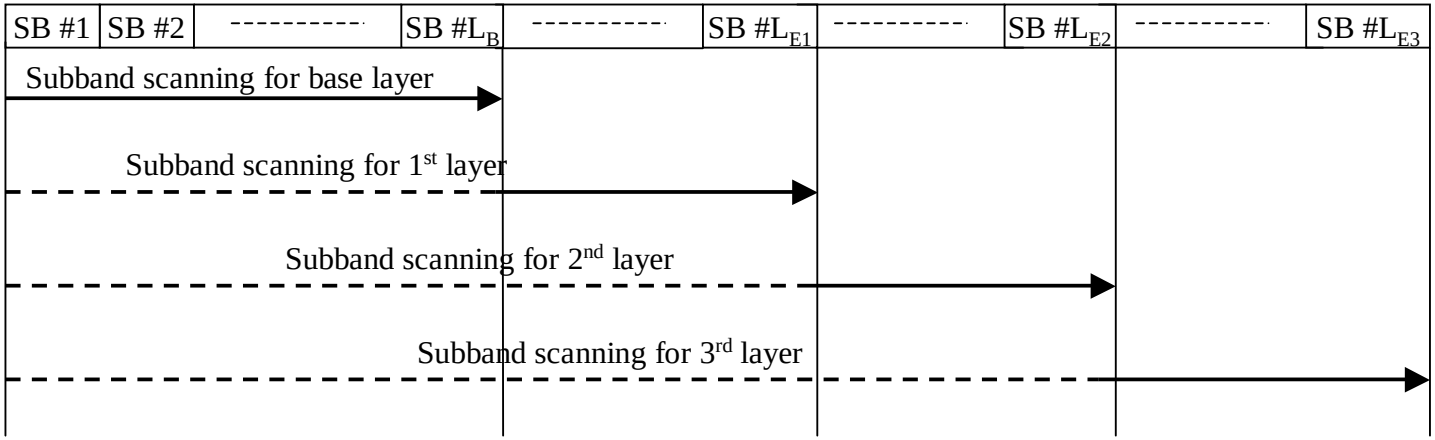


Figure 4.3: Subband scanning rule, where the solid line with arrow means all subbands inside this area are scanned, and the dashed line means only those non-significant subbands inside the area are scanned.

subband dominant list is called the subband significant pass, and the process that updates the subband subordinate list is called the subband refinement pass.

Different coefficient magnitude thresholds are maintained in different subbands. Since we would like to deal with the most important subbands at first and get the best result with only a little information from the resource. Moreover, since sounds in different subbands have different sensibilities to human ears according to the psychoacoustic model, it is worthwhile to consider each subband independently instead of all subbands in one frame simultaneously.

We summarize the subband selection rule below.

Subband Selection Procedure

1. MNR threshold calculation

   Determine empirically the MNR threshold value $T_{i,k}^{\mathrm{MNR}}$ for channel i at layer k. Subbands with smaller MNR value at the current layer are given higher priority.

2. Subband dominant pass

   For those subbands that are still in the dominant subband list, if subband j in channel i has the current MNR value $\mathrm{MNR}_{i,j}^{k} < T_{i,k}^{\mathrm{MNR}}$, add subband j of channel i into the significant map, remove it from the dominant subband list, send 1 to the bitstream, indicating this subband is selected. Then, do coefficient SAQ for this subband. For subbands that have $\mathrm{MNR}_{i,j}^{k} \geq T_{i,k}^{\mathrm{MNR}}$, send 0 to the bitstream, indicating this subband is not selected at this layer.

3. Subband refinement pass

   For subband already in the subordinate list, do coefficient SAQ.

4. MNR values update

   Re-calculate and update MNR values for selected subbands.

5. Repeat Steps 1-4 until the bitstream meets the target rate.

4.5 Implementation Issues

4.5.1 Frame, subband or channel skipping

As mentioned earlier, each subband has its own initial coefficient magnitude threshold. This threshold has to be included in the bitstream as the overhead so that the decoder can start to reconstruct these coefficients once the layered information is available. In our implementation, the initial coefficient magnitude threshold $T_{i,j}(0)$ for channel i and subband j will be truncated to the nearest power of 2 that is no smaller than $C_{i,j}^{max}$, i.e.

$$T_{i,j}(0) = 2^{p_{i,j}},\ p_{i,j} = \lceil \log_2 C_{i,j}^{max} \rceil,$$

where $C_{i,j}^{max}$ is the maximum magnitude for all coefficients in channel i and subband j.

In order to save bits, the maximum power $p_i^{max} = max(p_{i,j}), \forall j$, for all subbands in channel i will be included in the bitstream at the first time when channel i is

selected. A relative value of each subband's maximum power, i.e. the difference $\Delta p_{i,j} = p_i^{max} - p_{i,j}$ between p_i^{max} and $p_{i,j}$, will be included in the bit stream at the first time when the selected subband becomes significant.

For a frame with its maximum value $C_{i,j}^{max}$ equal to 0, i.e. $max(C_{i,j}^{max}) = 0, \forall j$, which means all coefficients in channel i in this frame have value 0, then a special indicator will be set to let the decoder know it should skip this frame. Similarly, if $C_{i,j}^{max}$ has value 0, another special indicator is set to tell the decoder that it should always skip this subband. In some cases when the end user is only interested in reconstructing some channels, channel skipping can also be adopted.

4.5.2 Determination of the MNR threshold

At each layer, the MNR threshold for each channel is determined empirically. Two basic rules are adopted when calculating this threshold.

1. The MNR threshold should allow a certain number of subbands to pass at each layer.

   Since the algorithm sends 0 to the bit stream for each un-selected subband which is still in the significant subband list, if the MNR threshold is so small that it allows too few subbands to pass, too many overhead bits will be generated. As a result, this will degrade the performance of the progressive audio codec.

2. Adopt a maximum MNR threshold.

   If the MNR threshold calculated by using the above rule is greater than a predefined maximum MNR threshold T_{max}^{MNR}, then the current MNR threshold for channel i at k^{th} layer $T_{i,k}^{\mathrm{MNR}}$ will be set to T_{max}^{MNR}. This is based on the assumption that a higher MNR value does not provide higher perceptual audio quality perceived by the human auditory system.

4.6 Complete Algorithm Description

The block diagram of a complete encoder is shown in Figure 4.4. The perceptual model, the filter bank, the temporal noise shaping (TNS), and the intensity blocks in our progressive encoder are borrowed from the AAC main profile encoder. The inter-channel redundancy removal procedure via KLT is implemented after the input audio signals are transformed into the MDCT domain. Then, a dynamic range control block follows to avoid any possible data overflow in later compression stages. Masking thresholds are then calculated in the perceptual model based on the KL transformed signals. The progressive quantization and lossless coding parts are finally used to construct the compressed bitstream. The information generated at the first several coding blocks will be sent into the bitstream as the overhead.

Figure 4.5 provides more details of the progressive quantization block. The channel and the subband selection rules are used to determine which subband in which channel should be encoded at this point, and then coefficients within this selected

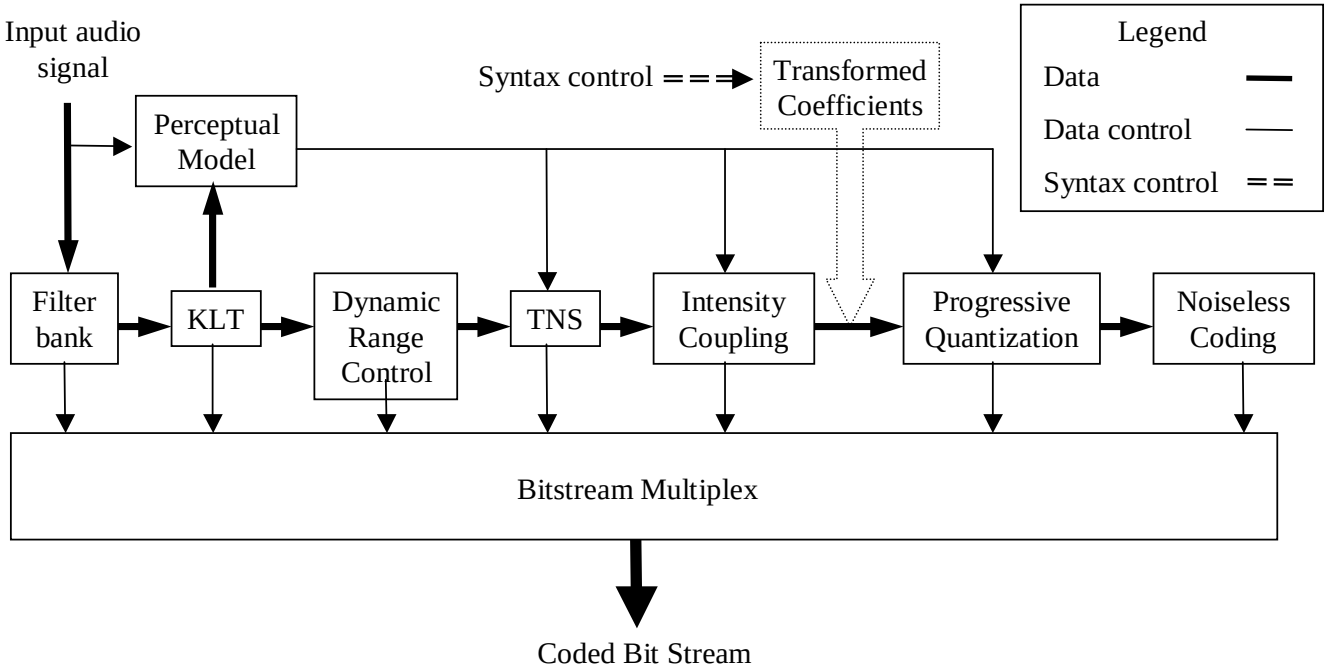


Figure 4.4: The block-diagram of the proposed PSMAC encoder.

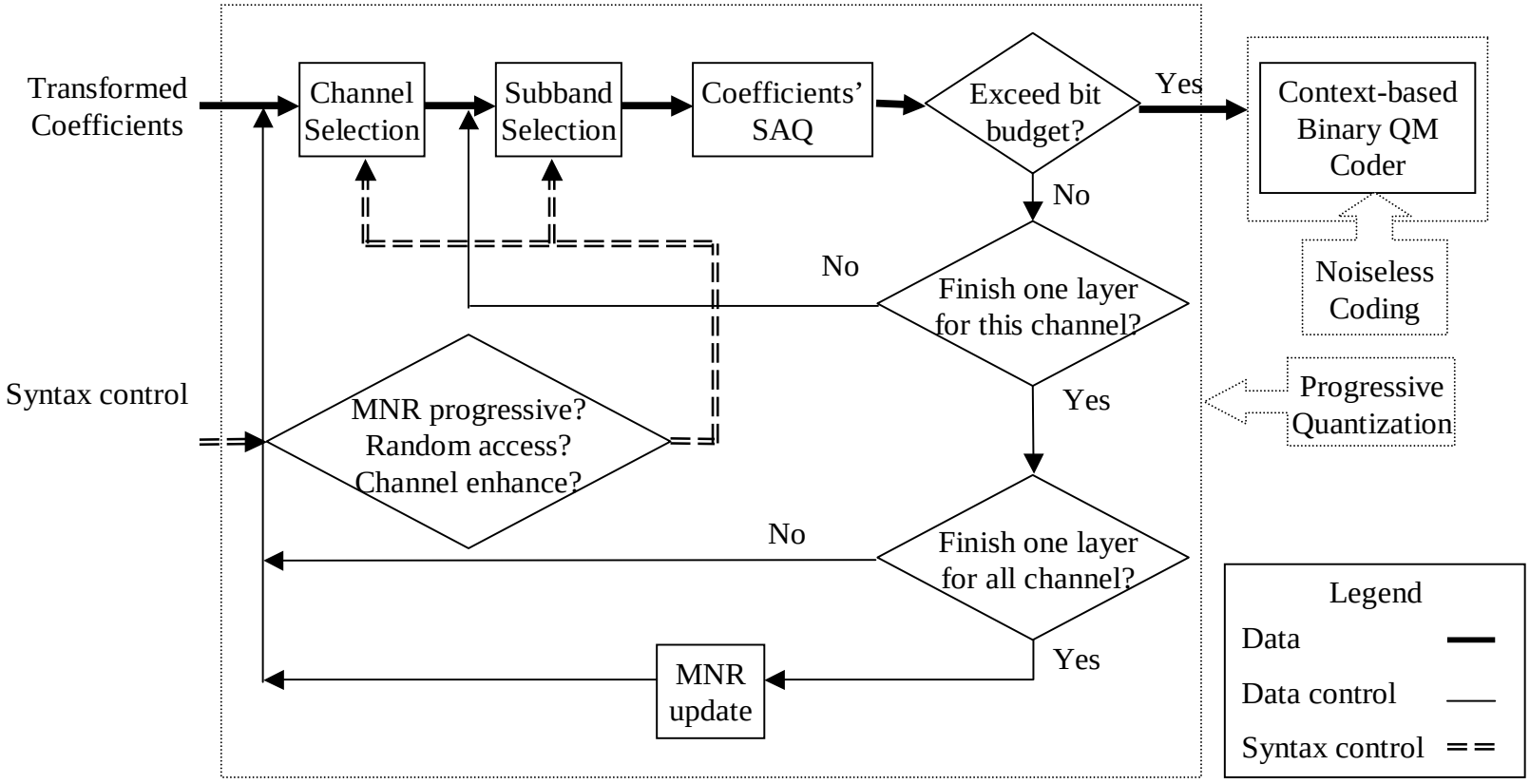


Figure 4.5: Illustration of the progressive quantization and lossless coding blocks.

subband will be quantized via SAQ. The user defined profile parameter is used for the syntax control of the channel selection and the subband selection. Finally, based on several different contexts, the layered information together with all overhead bits generated during previous coding blocks will be losslessly coded by using the context-based QM coder.

The encoding process performed by using the proposed algorithm will stop when the bit budget is exhausted. It can cease at any time, and the resulting bitstream contains all lower rate coded bitstreams. This is the so-called fully embedded property. The capability to terminate the decoding of an embedded bitstream at any specific point is extremely useful in systems that are either rate-constrained or distortion-constrained.

4.7 Experimental Results

The proposed PSMAC system has been implemented and tested. The basic audio coding blocks [ISOc] inside the MPEG AAC main profile encoder, including the psychoacoustic model, filter bank, temporal noise shaping and intensity/coupling, are still adopted. Furthermore, an inter-channel removal block, a progressive quantization block and a context-based QM coder block are added to construct the PSMAC audio codec. Two kinds of experimental results are shown in this sections. They are results measured by objective metric, i.e. Mask-to-Noise Ratio (MNR), and results measured in subjective metric, i.e. listening test score. It is worthwhile to mention

that no software optimization has been performed in any codec we used and the coding blocks adopted from AAC has not been modified to improve the performance of our codec. Moreover, test audio that has the worst performance generated by the MPEG reference software has not been selected in the experiment.

4.7.1 Results using MNR measurement

Two multichannel audio materials are used in this experiment to compare the performance of the proposed PSMAC algorithm with MPEG AAC's [ISOc] main profile codec. One is a one-minute long ten-channel [2] audio material called "Messiah", which is a piece of classical music recorded live in a real concert hall. Another one is an eight-second long five-channel [3] music called "Herre", which was used in the MPEG-2 AAC standard (ISO/IEC 13818-7) conformance work.

4.7.1.1 MNR Progressive

The performance comparison of MPEG AAC and the proposed PSMAC for the normal MNR progressive mode are shown in Table 4.1. The average MNR shown in the table is calculated by Equation 3.4 and 3.5

Table 4.1 shows the MNR values to compare the performance of the non-progressive algorithm AAC and the proposed PSMAC algorithm when working in the MNR progressive profile. Values in this table clearly show that our codec outperforms AAC

[2]including center, left, right, left surround, right surrond, back surround, left high, right high, left wide and right wide.

[3]including center, left, right, left surround and right surrond.

Table 4.1: MNR comparison for MNR progressive profiles

Bit rate (bit/s/ch)	Average MNR values (dB/subband/ch)			
	Herre		Messiah	
	AAC	PSMAC	AAC	PSMAC
16k	-0.90	6.00	14.37	21.82
32k	5.81	14.63	32.40	34.57
48k	17.92	22.32	45.13	42.81
64k	28.64	28.42	54.67	47.84

for both testing materials at lower bit rates and only has a small performance degradation at higher bit rates. In addition, the bitstream generated by MPEG AAC only achieves an approximate bit rate and is normally a little bit higher than the desired one while our algorithm achieves a much more accurate bit rate in all experiments carried out.

4.7.1.2 Random Access

The MNR result after the base layer reconstruction for the random access mode by using the test material "Herre" is shown in Table 4.2. When listening to the reconstructed music, we can clearly hear the quality difference between the enhance period and the rest of the other period. The MNR value given in Table 4.2 verifies the above claim by showing that the mean MNR value for the enhanced period is about 10 dB per subband better than the rest of other periods. It is common that we may prefer a certain part of a music to others. With the random access profile, the user can individually access a period of music with better quality than others when the network condition does not allow a full high quality transmission.

Table 4.2: MNR comparison for Random Access and Channel Enhancement profiles

Random Access		Channel Enhancement			
		Enhanced channel		Other channels	
other area	enhanced area	w/o enhance	w/ enhance	w/o enhance	w/ enhance
3.99	13.94	8.42	19.23	1.09	-2.19

4.7.1.3 Channel Enhancement

The performance result using test material "Herre" for the channel enhancement mode is also shown in Table 4.2. Here, the center channel has been enhanced with enhancement parameter 1 and coded at bit rate of 16 kbit/s/ch. Since we have to separate the quantization and the coding control of the enhanced physical channel, as well as to ease the implementation, KLT is disabled in the channel enhancement mode. Compared with the normal MNR progressive mode, we find that the enhanced center channel has an average of more than 10 dB per subband MNR improvement, while the quality of other channels is only degraded by about 3 dB per subband. When subjectively listen to the reconstructed audio, the one with the center channel enhanced has a much better performance and is more appealing, compared with the one without channel enhancement at the very low bit rate of 16 kbit/s/ch. This is because the center channel of "Herre" contains more musical information than other channels and a better reconstructed center channel will give listeners with better overall quality, which is basically true for most multichannel audio materials. Therefore, this experiment suggests that with a narrower bandwidth, audio generated by the channel enhancement mode of the PSMAC algorithm can provide the

user a more compelling experience with either a better reconstructed center channel or a channel which is more interesting to a particular user.

4.7.2 Subjective Listening Test

In order to further confirm the advantage of the proposed algorithm, a formal subjective listening test according to ITU recommendations [111, 128a, 128b] was conducted in an audio lab to compare the coding performance of the proposed PSMAC algorithm and that of the MPEG AAC main profile codec. Same group of listeners as described in Section 2.8.3 participated in the listening test. During the test, for each test sound clips, subjects listened to three versions of the same sound clips, i.e. the original one followed by two processed ones (one by PSMAC and one by AAC in random order), subjects were allowed to listen to these files as many times as possible until they were comfortable to give scores to the two processed sound files for each test material.

The five-grade impairment scale given in Recommendation ITU-R BS. 1284 [128a] was adopted in the grading procedure and utilized for final data analysis. Besides "Messiah" and "Herre", another two 10-channel audio materials, i.e. "Band" and "Herbie", are added in this subjective listening test. According to ITU-R BS. 1161-1 [111], audio files selected for listening test only contains short durations, i.e. 10 to 20 seconds long.

Figure 4.6 shows the score given to each test material coded at four different bit rates during the listening test for multi-channel audio materials. The solid vertical

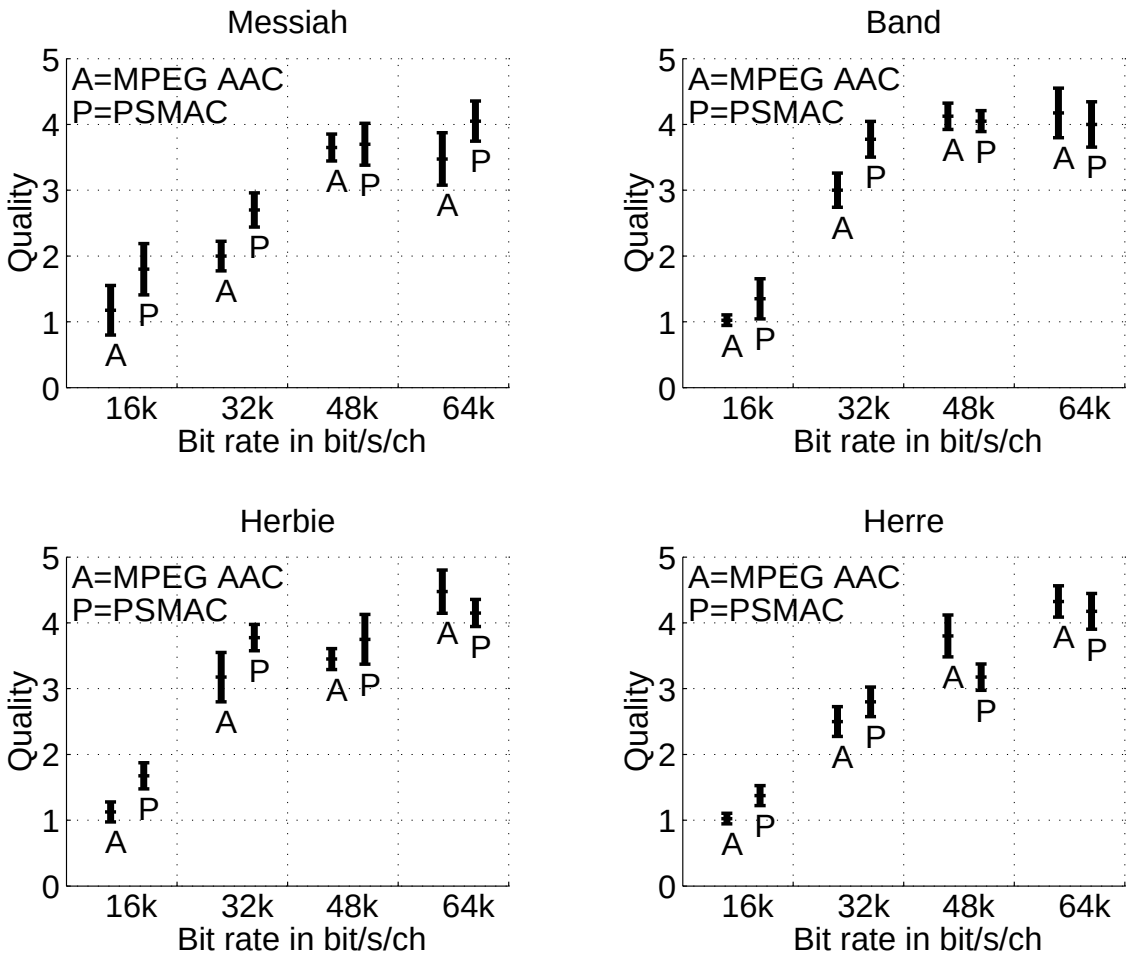


Figure 4.6: Listening test results for multi-channel audio sources

line represents the 95% confidence interval, where the middle line shows the mean value and the other two lines at the boundary of the vertical line represent the upper and lower confidence limits [RADH87]. It is clear from Figures 4.6 that at lower bit rate, such as 16 kbit/s/ch or 32 kbit/s/ch, the proposed PSMAC algorithm outperforms MPEG AAC in all four test materials; while at higher bit rate, such as 48 kbit/s/ch or 64 kbit/s/ch, PSMAC achieves comparable or a little degraded subjective quality when compared with MPEG AAC.

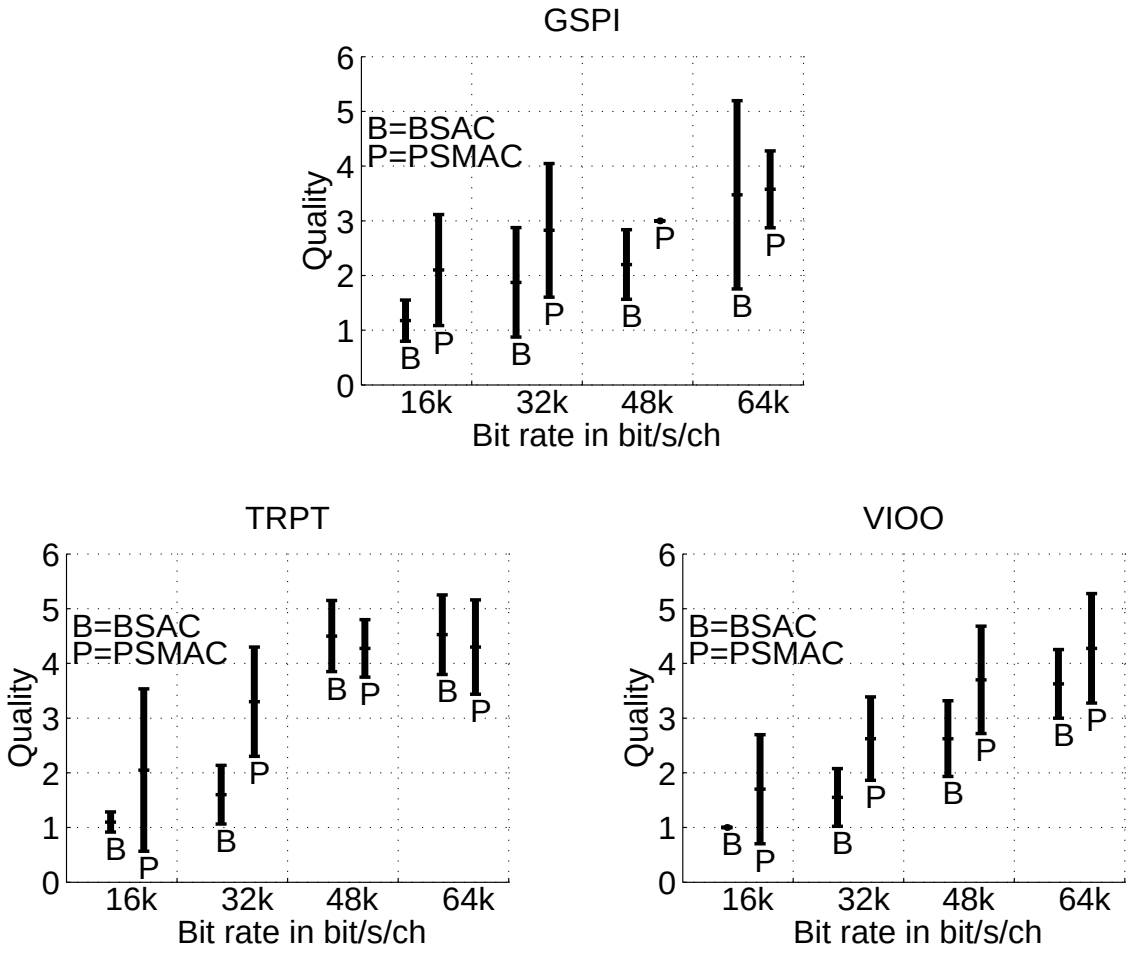


Figure 4.7: Listening test results for single channel audio sources. The cases where no confidence intervals are shown correspond to the situation when all four listeners happened to give the same score to the given sound clip.

To demonstrate that the PSMAC algorithm achieves excellent coding performance even for single channel audio files, another listening test for mono sound is carried out as well. Three single channel test audio materials, called "GSPI", "TRPT" and "VIOO", are used in this experiment. Here we are comparing the performance between standard fine-grain scalable audio coder provided by MPEG-4 BSAC [ISOb, ISOh] and the single channel mode of the proposed PSMAC algorithm.

Figure 4.7 shows the listening test result for single channel audio materials. From this figure, we can clearly see that at lower bit rates, e.g. 16 kbit/s/ch and 32 kbit/s/ch, our algorithm generates better sound quality for all test sequences. At higher bit rates, e.g. 48 kbit/s/ch and 64 kbit/s/ch, our algorithm outperforms MPEG-4 BSAC for two out of three test materials and is only a slightly worse for the "TRPT" case.

4.8 Conclusion

A progressive syntax-rich multichannel audio coding algorithm is presented in this research. This algorithm utilizes KLT in the pre-processing stage to remove inter-channel redundancy inherent in the original multichannel audio source. Then, rules for channel selection and subband selection are developed and the SAQ process is used to determine the importance of coefficients and their layered information. At the last stage, all information is losslessly compressed by using the context-based QM coder to generate the final multichannel audio bitstream.

The distinct advantages of the proposed algorithm over most existing multichannel audio codecs not only lie in its progressive transmission property which can achieve a precise rate control, but also in its rich-syntax design. Compared with the new MPEG-4 BSAC tool, PSMAC provides a more delicate subband selection strategy such that the information, which is more sensitive to the human ear, is

reconstructed earlier and more precisely at the decoder side. It was shown by experimental results that PSMAC has a comparable performance as non-progressive MPEG AAC at several different bit rates when using the multichannel test material, while it achieves better reconstructed audio quality than MPEG-4 BSAC tools when using single channel test materials. Moreover, the advantage of the proposed algorithm over the other existing audio codec is more obvious at lower bit rates.

Chapter 5

Error-Resilient Design

5.1 Introduction

High quality audio communication becomes an increasingly important part of the global information infrastructure. Compared with speech, audio communication requires a much more volume of data being transmitted in a timely manner, and a highly efficient compression scheme for the storage and transmission of audio data are critical. Extensive research on audio coding has been conducted in both academia and industry for years. Several standards, including AC-3, MPEG-1, MPEG-2, MPEG-4 [BB97, A/5, ISOa, ISOe, ISOf], have been established in the past decade. Earlier standards, *e.g.* MPEG-1, MPEG-2 or AC-3 were primarily designed for coding efficiency and they only allow a fixed bit rate coding structure. These algorithms are not ideal for audio delivery over noisy wireless IP networks with a time-varying bandwidth, since they do not take error resilience and VBR traffic into consideration.

Recent technological developments have led to several mobile systems aiming at personal communications services (PCS), supporting both speech and data transmission. Mobile uses usually communicate over wireless links characterized by lower bandwidths, higher transmission error rates, and more frequent disconnections in comparison to wired networks. To transmit high quality audio through an IP network with VBR (variable bit rates) traffic, a scalable audio compression algorithm, which is able to transfer audio signals from coarse to fine qualities progressively, is desirable. However, to achieve a good coding gain, most existing scalable techniques adopt variable-length coding in their entropy coding part, which makes the entire bitstream susceptible to channel noise. The traditional channel coding scheme only protects bits equally, without giving important bits higher protection, which results in a situation that a small bit error rate may lead to reconstructed audio with annoying distortion or even unacceptable perceptual quality. Since most audio compression standards and network protocols, such as the MPEG-4 version 2 audio codec, were designed for wired audio communications, they would not be effective if straightforwardly applied to the wireless case. A scalable bitstream with joint source-channel coding would be truly needed in a wireless audio streaming system.

MPEG-4 version 2 supports audio fine-grain scalability and error-robust coding [ISOe, ISOf]. Its error resilient AAC coder does not have the progressive property. Its BSAC utilizes Segmented Binary Arithmetic (SBA) coding to avoid error propagation within spectral data. However, this feature alone is not sufficient to protect the audio data in an effective manner over the wireless channel. Compared to

work on error-resilient image/video coding, the number of papers on error resilient audio coding is relatively small. Data partitioning and reversible variable length codes were adopted by Zhou *et al.* in [ZZXZ01] to provide the error-resilient feature to the scalable audio codec in [ZL01]. Base on the framework in [ZZXZ01], Wang *et al.* incorporated an unequal error protection scheme in [WZZZ01]. In Chapter 4, we proposed a progressive high quality audio coding algorithm, which has been shown to outperform MPEG-4 version 2's scalable audio codec. In this work, we extend the error-free progressive audio codec to an error-resilient scalable audio codec (ERSAC) by re-organizing the bitstream and modifying its noiseless coding part. The proposed error-resilient scalable audio codec actually uses the MPEG Advanced Audio Coding (AAC) as the baseline together with an error robust scalable transmission module, which is specifically designed for WCDMA channels.

In the proposed ERSAC codec, a dynamic segmentation scheme is first used to divide the audio bitstream into several variable-length segments. In order to achieve good error resiliency, the length of each segment is adaptively determined by the characteristics of WCDMA channels. The arithmetic coder and its probability table are re-initialized at the beginning of each segment, so that synchronization can be achieved at the decoder side even when error occurs. Bits within each segment are ordered in such a way that more important bits are placed near the synchronization point. In addition, an unequal error protection scheme is adopted to improve robustness of the final bitstream, where Reed-Solomon codes are used to protect data bits, and the parameters of each Reed-Solomon code is determined by the WCDMA

channel condition. Moreover, a frequency interleaving technique is adopted when data packetization is performed so that the frequency information belongs to the same period is sent in different packets. In this way, even if some packets belongs to the header or the base layer are corrupted, we still can hear a poorer quality period of sound with some frequency component lost (unless packets corresponding to the same period of sound are corrupted at the same time). We test the performance of our algorithm using several single-channel audio materials under different error patterns of WCDMA channels. Experimental results show that the proposed approach has excellent error resiliency at a regular user bit rate of 64 kb/s.

The rest of this chapter[1] is organized as follows. Some characteristics of the WCDMA channel are summarized in Section 5.2. The layered audio coding structure is described in Section 5.3. Section 5.4 explains the detailed error-resilient technique in the proposed algorithm. Some experimental results are shown in Section 5.5, and concluding remarks are given in Section 5.6. Some discussions and future work directions are addressed in Section 5.7.

5.2 WCDMA Characteristics

The third generation (3G) mobile communication systems have been designed for effective wireless multimedia communication [HT01]. The WCDMA (wideband Direct-Sequence Code Division Multiple access) technology has been adopted by the

[1]Part of this chapter represents work published before, see [YAKK02b]

UMTS standard as the physical layer for air interface. WCDMA has the following characteristics.

- WCDMA is designed to be deployed in conjunction with GSM.
- The chip rate of 3.84 Mcps used leads to a carrier bandwidth of approximately 5 MHz.
- WCDMA supports highly variable user data rates; in other words, the concept of obtaining Bandwidth on Demand (BoD) is well supported.
- WCDMA supports two basic modes of operation: Frequency Division Duplex (FDD) and Time Division Duplex (TDD).
- WCDMA supports the operation of asynchronous base stations.
- WCDMA employs coherent detection on uplink and downlink signals based on the use of pilot symbols or common pilot.
- The WCDMA air interface has been crafted in such a way that advanced CDMA receiver concepts can be deployed by the network operator as a system option to increase capacity and/or coverage.

Two reference error-resilient simulation studies [ITU98, ITU99] for the characterization of the radio channel performance of the 1.9 GHz WCDMA air interface were recently carried out by the ITU-Telecommunications Standardization Sector. In the study of 1998, which is referred to as study #1, only six simulation results for fixed data bit rate of 64 kb/s were obtained. In the study of 1999, which is

referred to as study #2, simulation results were extended to four different data bit rates, including 32 kb/s, 64 kb/s, 128 kb/s and 384 kb/s. Study #2 also replaced convolutional codes by turbo codes so that an even better channel performance can be achieved. In this work, we only consider error-resilient coding for single-channel audio coded at 64 kb/s. The main characteristics of all error patterns corresponding to 64 kb/s contained in two studies are listed in Table 5.1. Each error pattern file is of 11,520,000 bit long corresponding to 3 minutes of data transmission. The bit error is a binary one. Within a byte the least significant bit is transmitted first.

Table 5.1: Characteristics of WCDMA error patterns.

Study #	File #	File name	Mobile speed (km/h)	Average BER (b/s)
1	0	wcdma-64kb-005hz-4	3	8.2e-5
1	1	wcdma-64kb-070hz-4	40	1.2e-4
1	2	wcdma-64kb-211hz-4	120	9.4e-5
1	3	wcdma-64kb-005hz-3	3	1.4e-3
1	4	wcdma-64kb-070hz-3	40	1.3e-3
1	5	wcdma-64kb-211hz-3	120	9.7e-4
2	6	wcdma_64kb_50kph_7e-04	50	6.6e-4
2	7	wcdma_64kb_50kph_2e-04	50	1.7e-4
2	8	wcdma_64kb_3kph_5e-04	3	5.1e-4
2	9	wcdma_64kb_3kph_2e-04	3	1.6e-4
2	10	wcdma_64kb_3kph_7e-05	3	7.2e-5
2	11	wcdma_64kb_3kph_3e-06	3	3.4e-6
2	12	wcdma_64kb_50kph_6e-05	50	6.0e-5
2	13	wcdma_64kb_50kph_3e-06	50	3.4e-6

5.3 Layered Coding Structure

5.3.1 Advantages of the Layered Coding

The most popular and efficient way to construct a scalable bitstream is to use the layered coding structure. When the information from the first and the most important layer called the base layer is successfully retrieved from the bitstream at the decoder side, a rough outline of the entire sound file can be recovered. When the information from more and more higher level layers, called enhancement layers, are successfully retrieved from the bitstream, the sound file with better and better quality can be reconstructed. When the bitstream is transmitted over error-prone channels, such as the wired and/or wireless IP networks, the advantage of the layered coded bitstream is more notable than the fixed rate bitstreams. For a fixed rate bitstream, when an error occurs during transmission, the decoder can only reconstruct the period before the error and the period after the decoder regain the synchronization caused by the error. The resulting sound file may contain the lost period of several milli-seconds to several seconds long, depending on how long the synchronization is regained at the decoder site. If the bitstream cannot be re-synchronized, the data after the error may be completely lost, which results in a partially reconstructed sound file. However, when an error occurs during transmission of a layered coded bitstream, unless the error occurs in the base layer, the decoder can still recover the sound file, but has sound quality degradation in enhancement layers for some period of time. Experiments suggest that, even containing poorer quality for some period of time,

the reconstruct sound file of full length would give listeners better sensation than a sound file with some completely lost periods.

5.3.2 Main Features of Scalable Codec

The major difference between the proposed scalable codec design and the traditional fixed bit rate codec lies in the quantization module and the entropy coding module. The ERSAC algorithm inherit the basic idea of the progressive quantization and context-based QM coder in PSMAC algorithm to achieve the fine-grain bit rate scalability. In order to classify and protect bit according to their importance, bits are re-ordered so that bits which belong to the same priority are grouped together for easier protection.

Compared with PSMAC algorithm, the major modification in the progressive quantization module lies in how to transmit those subband significant bits. In PSMAC, bits which indicate the subband significance are sent together with the coefficient bits, while in the ERSAC algorithm, these bits are sent in the header. In PSMAC the threshold used to determine the subband significance is MNR values and they are updated after each coding layer, while in ERSAC, SMR are adopted for determination of the subband significance in every layer. Thus the subband selecting sequence might not be the same when using PSMAC and ERSAC algorithm for some input audio files. However, experiments show that the perceptual quality of the reconstructed sound files is quite similar when adopting these two slightly different subband selection rules.

In ERSAC, at layer i, $i \leq 3$, an empirical threshold T_i based on the Signal-to-Mask Ratio (SMR) is calculated. Only those subbands whose SMR values are greater or equal to T_i will be selected and become significant. At the next layer, *i.e.* layer $i+1$, the SMR values of newly included subbands together with the remaining non-significant subbands in previous layers will be compared with T_{i+1}, and an updated significant subband list will be created.

Figure 5.1 provides an example to show how subbands are selected from layer 0 to layer 3. To better illustrate this procedure, L_0, L_1, L_2 and L_3 are set to 6, 10, 14 and 18, respectively, in this example. If the bit budget is not exhausted after the 3rd enhancement layer, more layers can be encoded. All subbands will be included in the significant list from this point on. At the encoder, the subband significance information is included in the header, where a binary digit "1" represents a significant subband and "0" represents a non-significant subband. Thus, in the example given in Figure 5.1, the subband significance information bits should be

$$0101111011011001100101011.$$

Whenever the encoder finds a significant subband, it visits coefficients inside this subband, and performs progressive quantization on coefficients and then do the entropy coding. Here, we adopt the Successive Approximation Quantization (SAQ) scheme to quantize the magnitude of coefficients and the context-based QM coder

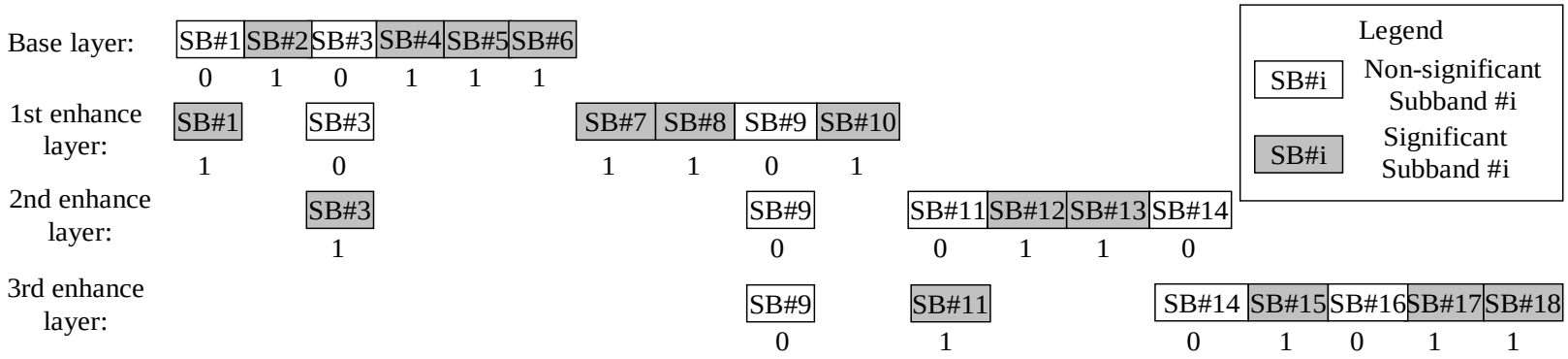


Figure 5.1: A simplified example of how subbands are selected from layers 0 to 3.

to noiseless code all generated bits. Detailed description of coefficients' SAQ and context-based QM coder can be found in Chapter 4.

5.4 Error-Resilient Codec Design

5.4.1 Unequal Error Protection

When a bitstream is transmitted over the network, errors may occur due to channel noise. Even with channel coding, errors can still be found in received audio data. In particular, for highly compressed audio signals, a small bit error rate can lead to highly annoying or perceptually unacceptable distortion. Instead of demanding an even lower bit error rate (BER), which is expensive to achieve in a wireless environment, the use of joint source and channel coders have been studied in [YFT+99, SS99, WZZZ01] and shown to be promising in achieving good perceptual quality without complex processing.

Most coded audio bitstreams contain certain bits that are more sensitive to transmission errors than others in audio reconstruction. Unequal error protection (UEP) offers a mechanism to relate the transmission error sensitivity to the error protection capability. An UEP system typically has the same average transmission rate as a corresponding equal error protection (EEP) system but offers an improved perceived signal quality at the same channel signal to noise ratio. In order to prevent the complete loss of transmitted audio, the critical bits need to be well protected from channel errors. Examples of critical bits include the headers and the most

significant bits of source symbols. The loss of header's information usually leads to catastrophic sound distortion while an error in the most significant bit results in higher degradation than that of others. Thus, high-priority bits need to be protected using channel coding or other methods. But the redundancy due to channel coding reduces compression efficiency. A good tradeoff between rates of the source coder and the channel coder has to be considered.

The study of error-correcting codes began in late 1940's. Among several error correcting codes [MS77, LDJC83], such as Hamming, BCH, cyclic and Reed-Muller codes, the Reed-Solomon code is chosen in our implementation because of its excellent performance on correcting burst errors, which is the most common case in wireless channel transmission. Reed-Solomon codes are block-based error correcting codes with a wide range of applications in digital communications and storage. The number and the type of errors that can be corrected by Reed-Solomon codes depend on code parameters. For a fixed number of data symbols, codes that can detect and correct a smaller number of bit errors has smaller parity check symbols, thus producing smaller redundancy bits. In this work, data in the compressed audio bitstreams are protected according to their error sensitivity classes. Experimental results suggest that errors in both headers and the base layer lead to unacceptable reconstructed audio quality, while the same amount of errors in the enhancement layers results in less perceptual distortion with the number of the enhancement layer goes higher. Therefore, bits in the header and the base layer are given the same highest priority, bits in the first enhancement layer are given the moderate priority, and

bits in the second and higher enhancement layers are given the lowest priority during the error protection procedure.

To further determine the error correcting capability of Reed-Solomon codes used for each error sensitivity class, more detailed analysis is performed on all WCDMA error patterns. Since the Reed-Solomon code is a byte-based error correcting code, the mean and the standard deviation of the byte error rate are calculated for each error file. Files with similar byte error rate characteristics are combined to one group. All fourteen error patterns are finally divided into four groups, whose virtual mean and standard deviation values are then empirically determined. Based on these virtual statistical data, the target error correcting capability of the Reed-Solomon code is finally calculated by the following formula

$$etarget(g,c) = mean_{byte}(g) + f_{byte}(g,c) \times std_{byte}(g), \tag{5.1}$$

where $etarget(g,c)$, $mean_{byte}(g)$, $std_{byte}(g)$ are the target error correcting ability (in percentage), the virtual mean and the virtual standard deviation of the byte error rate for group g and error sensitive class c, respectively, and $f_{byte}(g,c)$ is a function of group number g and error sensitivity class number c.

5.4.2 Adaptive Segmentation

Although the arithmetic coder has excellent coding efficiency and is adopted by almost all layered-coded source coding techniques, the arithmetic coder together with

other variable-length codes are known to be highly susceptible to channel errors due to the synchronization loss at the decoder side, which leads to error propagation, the loss of some source symbols and even the crash of the decoder. To prevent these undesirable results and, at the same time, to consider the redundancy generated by the UEP scheme, an adaptive segmentation strategy is developed in this work. That is, the generated bitstream is partitioned into several independent segments. By "independent", we mean that the entropy coding module at the decoder side can independently decode each segment. This can be achieved as follows. At the beginning of each segment, the arithmetic coder is restarted, its probability tables are refreshed, and some stuffing bits are appended at the end of each segment so that each segment is byte-aligned. In this way, several independent synchronization points are provided in the bitstream and errors can only propagate until the next synchronization point, which means that errors will be confined to their corresponding segments and will not affect the decoding of other segments.

The determination of the segment length is another issue to be addressed. Since the arithmetic coder has to be restarted and flushed for each segment, segments with a length too small will considerably degrade the entropy coding efficiency. Thus, a good tradeoff between the coding performance and the error resilient capability should be studied. The use of the bit error rate as a parameter to determine the segment length provides an intuitive and straight-forward solution. However, after some exploration, we find out that the error distribution pattern should also be taken into consideration.

Experimental results show that error files with a similar bit error rate may have a quite different error distribution pattern. Let us introduce a concept called the error occurrence period, which is defined as the length (in bits) between two neighboring errors. Here, it is assumed that there are at least eight or more free bits between these two neighboring errors. The average error occurrence period and its standard deviation are calculated for each error pattern file. Files with the similar characteristics are grouped together. Then, the virtual mean and the virtual standard deviation value of the error occurrence period are empirically determined for each group. Finally, the segment length is calculated via

$$seglen(g) = mean_{occur}(g) + f_{occur}(g) \times std_{occur}(g), \tag{5.2}$$

where $seglen(g)$, $mean_{occur}(g)$ and $std_{occur}(g)$ are the segment length, the virtual mean and the virtual standard deviation of the error occurrence period for group g, respectively, and $f_{occur}(g)$ is a function of group number g. Note that the group number g in Eq (5.2) may not be the same as that in Eq (5.1).

5.4.3 Frequency Interleaving

Traditionally, all bits belonging to the same time position are packed together and sent into the bitstream. When errors happen in the global header or the data part of the base layer, the corresponding period of sound data cannot be reconstructed. Instead, it may have to be replaced by silence or other error concealment technology.

Simple experiments show that substituting the corrupted period with silence in a reconstructed sound file generates an unpleasant sound effect. So far, there is no effective error concealment technology to recover the lost period in audio. In order to improve the performance under this situation, a novel frequency interleaving method is proposed and incorporated in the ERSAC algorithm. With this new method, bits corresponding to different frequency components in the same time position are divided into two groups. Then, even if some packets are corrupted during transmission, the decoder can still be able to reconstruct a poorer quality version of the sound with some frequency component missing.

Figure 5.2 depicts a simple example on how the frequency interleaving is implemented in ERSAC. In this figure, only significant subbands for each layer in Figure 5.1 are shown. Adjacent significant subbands are divided into different groups. Bits belonging to a different group will not be included in the same packet. By this way, bits in different subbands, which correspond to different frequency interval, are interleaved so that a perceptually better decoded sound file can be reconstructed when some packets are corrupted.

To show the advantage of the frequency interleaving method, a simple experiment is carried out. During the experiment, we artificially corrupt two packets for each test sound file. Both packets belong to the data part of the base layer in the bitstream. The corresponding time positions of these packets are chosen such that one packet contains information for a smooth period and the other one contains information for a period with rapid melody variations. With one packet lost in the base

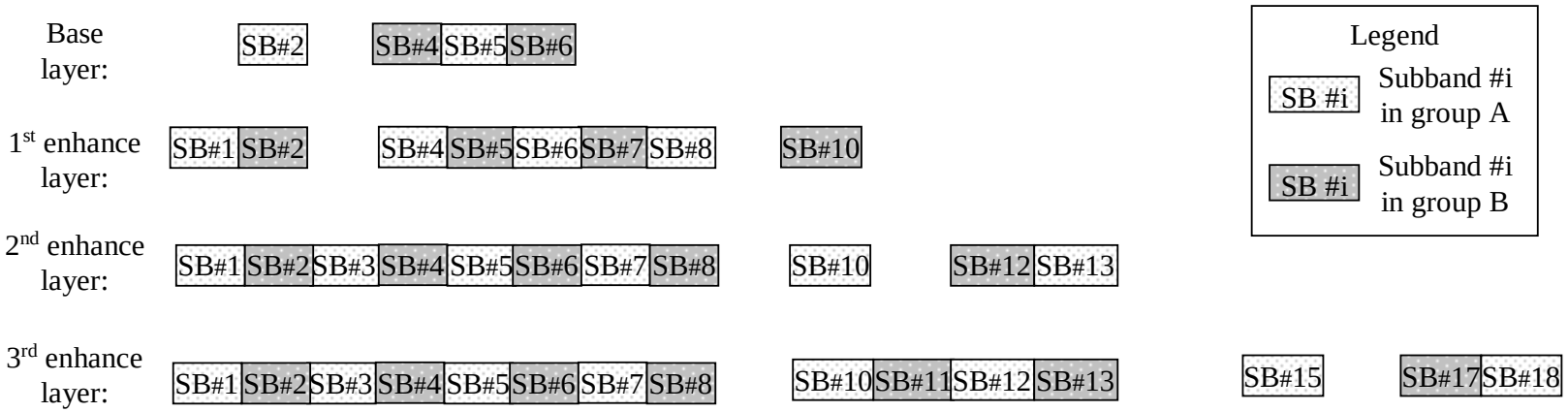


Figure 5.2: Example of frequency interleaving.

layer, coefficients corresponding to certain frequency areas cannot be reconstructed. Therefore, the reconstructed sound clips may contain a period with defects. However, the degree of the perceptual impairment differs a lot from sample to sample. Table 5.2 lists the experiment results for the frequency interleaving method.

Table 5.2: Experimental results of the frequency interleaving method.

Affected area	Input file		
	VIOO	TRPT	GSPI
Smooth area	Can only be noticed when listened really carefully	Can be noticed but not annoying	Can be noticed and is a little annoying
Area with rapid melody variations	hardly noticeable	hardly noticeable	can be noticed but not annoying

We see from this table that, for some input sound files, such as the one named "VIOO", users can hardly detect the defect of the reconstructed file. For some other input sound files, such as the one named "TRPT", users are able to catch the defect in the smooth period, but the perceptual impairment is in the level of "perceived but not annoying". For input sound files like "GSPI", which has a wide range of frequency components, users can easily detect the defect which may be somewhat annoying. On the other hand, if no frequency interleaving is enforced, when corrupted packets resides in the header or the data part of the base layer, no information for the corresponding time period can be recovered and can only be played back by silence if there is no concealment technique involved. We can decisively conclude that a sound file with a sudden silence period inserted is much

more annoying than the one constructed by the proposed frequency interleaving method.

5.4.4 Bitstream Architecture

The bitstream architecture of the proposed algorithm is illustrated in Figure 5.3. The entire bitstream is composed of a global header and several layered information. Bits contained by lower layers represent information of perceptually more important coefficients' values. In other words, the bitstream contains all lower bit rate codes at the beginning of the bitstream so that it can provide different QoS to different end-users. Let us look at the details of each layer. Within each layer, there are many variable-length segments, and each segment can be independently decoded. At the beginning of each segment, there is a segment header. These segment header bits are utilized to indicate the synchronization point. The data part within segments are partitioned into several packets. One packet is considered as a basic unit input into the Reed-Solomon coding block, where parity check bits are appended after data bits. At the end of each segment, there are some stuffing bits so that the whole segment is byte-aligned.

5.4.5 Error Control Strategy

When the end user receives a bitstream, the Reed-Solomon decoder is employed to detect and correct any possible errors occurred during channel transmission. Once

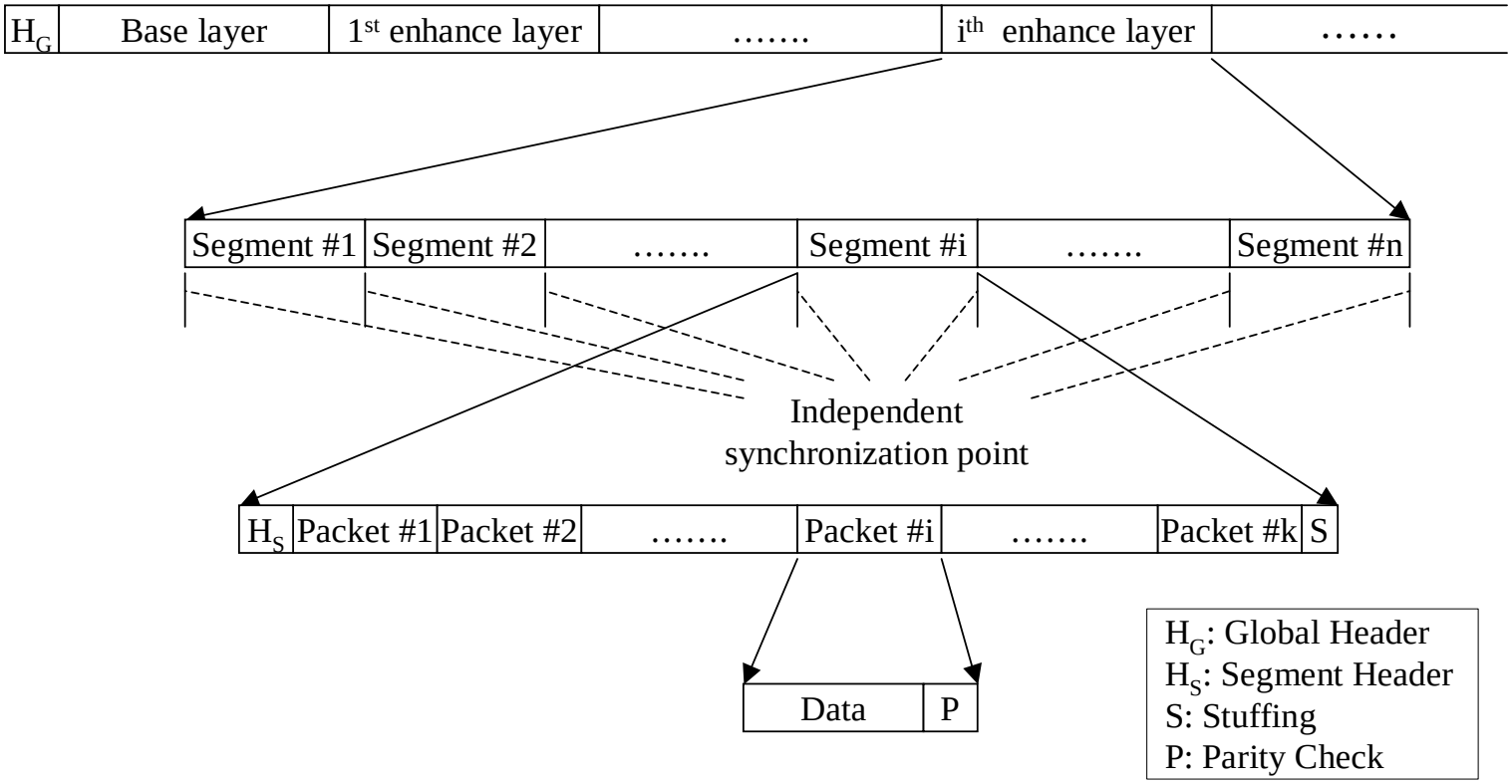


Figure 5.3: The bitstream architecture.

an uncorrectable error is detected, its position information, such as the layer number, the segmentation number and the packet number, will be marked. If this uncorrectable error occurs in the global header or the data part of the base layer, all bits belonging to the corresponding time position will not be reconstructed and this period of sound will be replaced with silence. Some error concealment strategy may be involved to make the final audio file sound more smoothly. However, if the uncorrectable error occurs in the data part of any other enhancement layers, all bits belonging to the corresponding time position will not be used to refine the spectral data so that this error will not cause unpleasant distortions.

Experimental results show that uncorrectable errors in layer two or higher have little impact on the final audio file. Normal listeners can hardly perceive any impairment, if not listening carefully. If these errors are in layer one, normal listeners may perceive a little but not annoying impairment in the final audio file. There is another type of error that happens to bits belonging to the segment header. When this type of error occurs, it may cause the decoder to lose synchronization and stop decoding earlier than expected. In this scenario, the error affects more frames which can be well beyond one specific segment and the resulting audio file may correspond to a lower rate reconstructed audio file with poorer quality.

5.5 Experimental Results

The proposed ERSAC system has been implemented and tested. The basic audio coding blocks [ISOc] of the MPEG AAC main profile encoder, including the psychoacoustic model, filter bank, and temporal noise shaping, are adopted to generate spectral data. An error-resilient progressive quantization block and a context-based QM coder block are added at the end to construct the error-robust scalable audio bitstream. Three single-channel sound files, *i.e.* GSPI, TRPT and VIOO, which are downloaded and processed from the MPEG Sound Quality Assessment Material, are selected to test the coding performance of the proposed algorithm. The Mask-To-Noise Ratio (MNR) values are adopted here as the objective sound quality measurement.

Figure 5.4 and Table 5.3 show the experimental results for three test material using different WCDMA error pattern file, where the mean MNR and the average MNR values are calculated by Equation 3.4 and 3.5.

Based on results shown in Figure 5.4 and Table 5.3, we have the following observations.

1. No error: there is no uncorrectable errors (in GSPI error pattern 0, 4, 5, 9, 10, 11, 12, 13; TRPT error pattern 0, 1, 2, 9, 10, 11, 12, 13; VIOO error pattern 0, 1, 10, 11, 13).

   This happens when either there is no error during the period when the bitstream is transmitted over the WCDMA channel or there are error but they

have been corrected by ERSAC's error detection scheme. Since more than half of experiment cases belong to this category, it shows the proposed ERSAC algorithm has an excellent error-resilient capability.

2. Error case 1: error occurs in the global header and the data part of the base layer (None is observed in our experiment).

   When this happens, the decoder has no way to reconstruct the affected period of the sound file. Then, this period will be error concealed by the repetition of data in the previous period.

3. Error case 2: error occurs in the segment header (None is observed in our experiment).

   When this happens, the decoder may lost synchronization and will not be able to continue decoding the proceeding bitstream, which means the decoder will stop refining all coefficients' values. If this happens in lower layers, *e.g.* layer 0 or layer 1, the reconstructed audio should have poor quality and the end user may easily perceive the distortion. However, if this happens in higher layers, *e.g.* layer 2 or higher, errors will not have big impact on the reconstructed sound file.

4. Error case 3: error occurs in the data part of layer 1 or higher (observed in all remaining cases).

   When this happens, the decoder will stop refining coefficients in the affected

period, and the reconstructed sound file has slightly degraded quality, which belongs to the perceptible distortion degree, but not annoying.

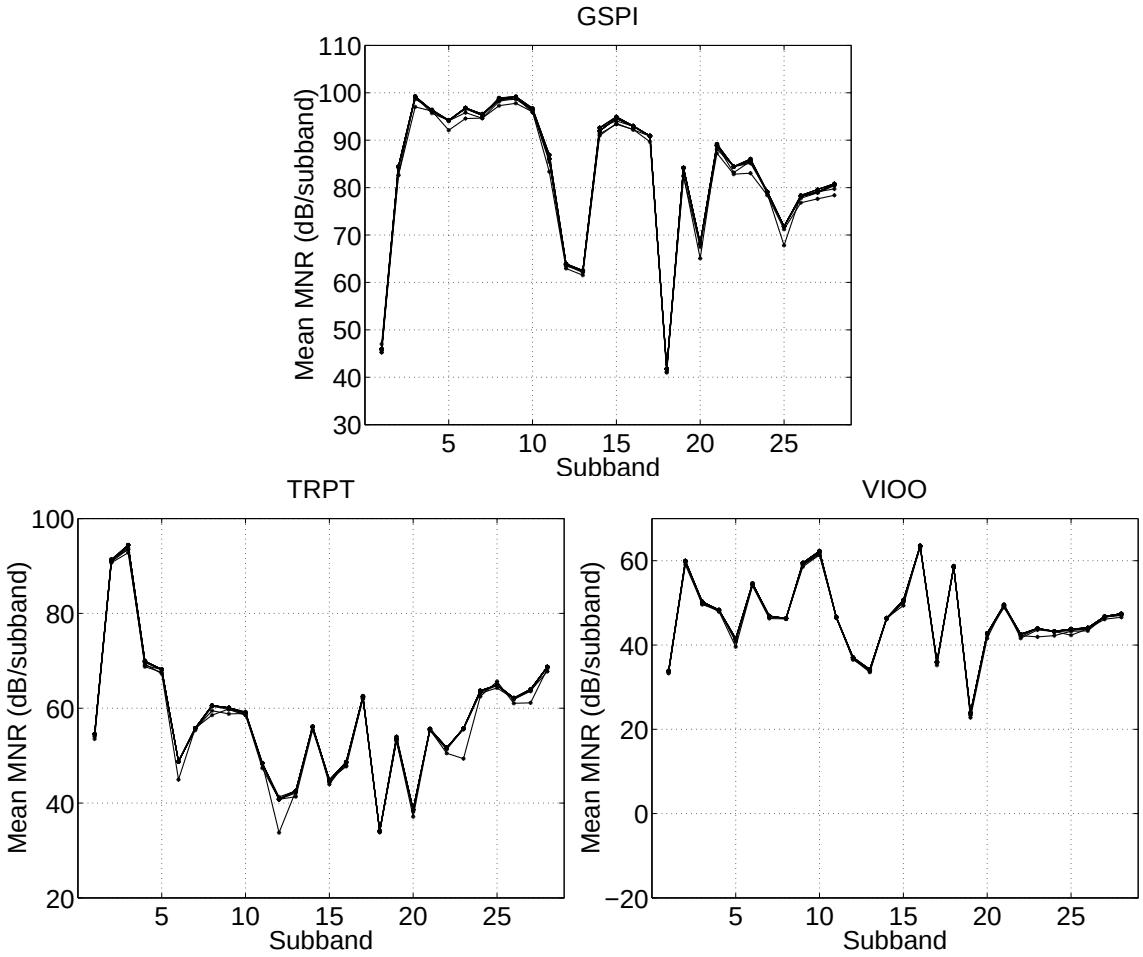


Figure 5.4: Mean MNR values of reconstructed audio files through different WCDMA channels.

5.6 Conclusion

We presented an error-resilient scalable audio coding algorithm, which is an extension of our previous work on progressive audio compression. Compared with other existing audio codecs, this algorithm not only preserves the scalable property, but

Table 5.3: Average MNR values of reconstructed audio files through different WCDMA channels.

Error pattern file #	Error pattern file name	Ave. MNR (dB/subband)		
		GSPI	TRPT	VIOO
0	wcdma-64kb-005hz-4	83.24	57.82	46.57
1	wcdma-64kb-070hz-4	82.97	57.82	46.57
2	wcdma-64kb-211hz-4	83.18	57.82	46.53
3	wcdma-64kb-005hz-3	83.25	57.64	46.25
4	wcdma-64kb-070hz-3	83.24	57.49	46.47
5	wcdma-64kb-211hz-3	83.24	57.77	46.43
6	wcdma_64kb_50kph_7e-04	82.72	57.24	46.38
7	wcdma_64kb_50kph_2e-04	83.36	57.80	46.31
8	wcdma_64kb_3kph_5e-04	81.96	56.82	46.23
9	wcdma_64kb_3kph_2e-04	83.39	57.86	46.45
10	wcdma_64kb_3kph_7e-05	83.39	57.86	46.61
11	wcdma_64kb_3kph_3e-06	83.24	57.82	46.57
12	wcdma_64kb_50kph_6e-05	83.39	57.86	46.58
13	wcdma_64kb_50kph_3e-06	83.24	57.82	46.57

also incorporate an error-robust scheme specifically designed for WCDMA channels. Based on the characteristics of the simulated WCDMA channel, a joint source-channel coding method was developed in this work. The novelty of this technique lies in its unique unequal error protection, adaptive segmentation coding structures and the freqeucny interleaving technique. Experimental results showed that the proposed ERSAC achieved a good performance using all simulated WCDMA error pattern files at a regular user bit rate of 64 kb/s.

5.7 Discussion and Future Work

5.7.1 Discussion

5.7.1.1 Frame Interleaving

Error-resilient algorithms designed for image or video codec normally contain a block interleaving procedure, where bits belonging to adjacent areas are packed separately so that any propagated error within packets will not affect a large area. This is done because human eyes are more sensitive to low frequency components while less to high frequency components when errors are spread to a larger area. Similarly, the frame interleaving technique can also be considered for error-resilient audio codec design. However, unlike image or video, experimental results show that human ears are capable in catching spreading impairment in sound files. In fact, a longer evenly distorted period is less annoying and more tolerable than an un-smoothly sound period with distortion frequently on and off. Therefore, no frame interleaving is adopted in the proposed algorithm, packets are just sent according to their original time sequence.

5.7.1.2 Error Concealment

Several error concealment techniques were proposed for speech communications [GLWW86, Jay81, WGDP88, VR93, RW85], such as SOLA, WSOLA, frame repetition and waveform substitution etc. However, these methods are not suitable for high quality audio because of different applications for speech and audio. The

main purpose of speech is communication while the main purpose of audio is entertainment. Thus, as long as people can understand, some noise in the background of speech is tolerable, which is certainly not the case for high quality audio. One common practice of existing speech error concealment methods is the addition of background noise. As a result, the error-concealed speech has good intelligibility while just having some additional noise in the background. However, adding noise in the audio file is normally un-tolerated, which makes none of available error concealment methods suitable for high quality audio.

5.7.2 Future work

In our current work, the ERSAC algorithm has only been implemented for single channel material, and it can be extended to accommodate stereo or even multichannel error-resilient codecs. Although only mono or stereo audio applications are needed in today's wireless communication systems, we can foresee the need of sending multichannel audio files over wired or wireless networks in the future. Thus, error-resilient multichannel audio coding is still a research topic. When input sound files with more than one channel are incorporated into the error-resilient codec, channel dependency should be taken into account, and could be utilized to develop an efficient error concealment strategy.

Chapter 6

Conclusion

The important results achieved in this dissertation are summarized as follows:

- **Modified Advance Audio Coding with Karhunen-Loève Transform (MAACKLT)**

  MAACKLT is a new quality-scalable high-fidelity multichannel audio compression algorithm based on MPEG-2 Advanced Audio Coding (AAC). The Karhunen-Loéve Transform (KLT) is applied to multichannel audio signals in the pre-processing stage to remove inter-channel redundancy. Then, signals in de-correlated channels are compressed by a modified AAC main profile encoder. Finally, a channel transmission control mechanism is used to re-organize the bitstream so that the multichannel audio bitstream has a quality scalable property when it is transmitted over a heterogeneous network. Experimental results show that, compared with AAC, the proposed algorithm achieves a better performance with the objective Mask-to-Noise-Ratio (MNR) measurement while maintaining a similar computational complexity at the regular bit rate

of 64 kbit/s/ch. When the bitstream is transmitted to narrow-band end users at a lower bit rate, packets of some channels can be dropped, and slightly degraded yet full-channel audio can still be reconstructed in a reasonable fashion without any additional computational cost.

- **Progressive Syntax-Rich Multichannel Audio Codec (PSMAC)**

  Based on AAC, we develop a progressive syntax-rich multichannel audio codec in this dissertation. It not only supports fine grain bit rate scalability for the multichannel audio bitstream, but also provides several other desirable functionalities, which are not available in other existing multichannel audio codecs. Moreover, compared with other existing scalable audio coding tools, a more sophisticated progressive transmission strategy is employed in PSMAC. A formal subjective listening test shows that the proposed algorithm achieves excellent performance at several different bit rates when compared with MPEG AAC using multichannel test material and when compared with MPEG version-2's BSAC using single channel test materials.

- **Error-Resilient Scalable Audio Coding (ERSAC)**

  Inheriting the basic coding structure of PSMAC, the ERSAC algorithm extends the error-free progressive audio codec to an error-resilient scalable audio codec by re-organizing the bitstream and modifying the noiseless coding module. A progressive quantization, a dynamic segmentation scheme, a frequency interleaving technique and an unequal error protection scheme are adopted in

the proposed algorithm to construct the final error robust layered audio bit-stream. The performance of the proposed algorithm is tested under different error patterns of WCDMA channels with several test audio materials. Our experimental results show that the proposed approach achieves excellent error resilience at a regular user bit rate of 64 kb/s.

Bibliography

[111] Recommemdation ITU-R BS. 1116-1. *Methods for the Subjective Assessment of Small Impairments in Audio Systems Including Multichannel Sound Systems.*

[128a] Recommemdation ITU-R BS. 1284. *Methods for the subjective assessment of sound quality – general requirments.*

[128b] Recommemdation ITU-R BS. 1285. *Pre-Selection Methods for the Subjective Assesment of Small Impairments in Audio Systems.*

[A/5] ATSC Document A/52. *Digital Audio Compression Standard (AC-3).*

[BB97] K. Brandenburg and M. Bosi. ISO/IEC MPEG-2 Advanced Audio Coding: Overview and applications. In *AES 103rd convention*, AES preprint 4641, New York, September 1997.

[BBQ+96] M. Bosi, K. Brandenburg, S. Quackenbush, L. Fielder, H. Fuchs K. Akagiri, M. Dietz, J. Herre, G. Davidson, and Y. Oikawa. ISO/IEC MPEG-2 Advanced Audio Coding. In *AES 101st convention*, AES preprint 4382, Los Angeles, November 1996.

[Bla83] J. Blauert. *Spatial Hearing.* MIT Press, 1983.

[Bra87] K. Brandenburg. Evaluation of quality for audio encoding at low bit rates. In *AES 82nd convention*, AES preprint 2433, London, 1987.

[Dav93] M. Davis. The AC-3 multichannel coder. In *AES 95th convention*, AES preprint 3774, New York, October 1993.

[Equ89] W. H. Equitz. A new vector quantization clustering algorithm. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(10), October 1989.

[Fuc93] H. Fuchs. Improving joint stereo audio coding by adaptive inter-channel prediction. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 39–42, 1993.

[GG91] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression.* Kluwer Academic, 1991.

[GL83] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Baltimore, MD: Johns Hopkins Univ. Press, 1983.

[GLWW86] D. J. Goodman, G. B. Lockhart, O. J. Wasem, and W.-C. Wong. Waveform substitution techniques for recovering missing speech segments in packet voice communications. *IEEE Transaction on Acoustics, Speech and Signal Processing*, 34(6), December 1986.

[HAB+98] J. Herre, E. Allamanche, K. Brandenburg, M. Dieta, B. Teichmann, B. Grill, A. Jin, T. Moriya, N. Iwakami, T. Norimatsu, M. Tsushima, and T. Ishikawa. The integrated fileterbank based scalable MPEG-4 audio coder. In *AES 105th convention*, AES preprint 4810, San Francisco, CA, September 26–29 1998.

[Hay96] S. Haykin. *Adaptive Filter Theory*. Prentice Hall, third edition, 1996.

[HJ96] J. Herre and J. Johnston. Enhancing the performance of perceptual audio coders by using temporal noise shaping (TNS). In *AES 101st convention*, AES preprint 4384, Los Angeles, CA, November 1996.

[HT01] H. Holma and A. Toskala. *WCDMA for UMTS, Radio Access for Third Generation Mobile Communications*. Wiley, revised edition, 2001.

[ISOa] ISO/IEC JTC1/SC29/WG11 N1650. *IS 13818-7 (MPEG-2 Advanced Audio Coding, AAC)*.

[ISOb] ISO/IEC JTC1/SC29/WG11 N2205. *Final Text of ISO/IEC FCD 14496-5 Reference Software*.

[ISOc] ISO/IEC JTC1/SC29/WG11 N2262. *ISO/IEC TR 13818-5, Software Simulation*.

[ISOd] ISO/IEC JTC1/SC29/WG11 N2425. *MPEG-4 Audio verification test results: Audio on Internet*.

[ISOe] ISO/IEC JTC1/SC29/WG11 N2503. *Information Technology – Coding of Audio-Visual Objectis – Part 3. ISO/IEC IS 14496-3:1999*.

[ISOf] ISO/IEC JTC1/SC29/WG11 N2803. *Information Technology – Coding of Audio-Visual Objects – Part 3: Audio Amendment 1: Audio Extensions. ISO/IEC 14496-3:1999/AMD 1:2000*.

[ISOg] ISO/IEC JTC1/SC29/WG11 N2803. *Text ISO/IEC 14496-3 Amd 1/FPDAM*.

[ISOh] ISO/IEC JTC1/SC29/WG11 N4025. *Text of ISO/IEC 14496-5:2001*.

[ITU98] ITU-T SG-16. *WCDMA Error Patterns at 64kb/s*, June 1998.

[ITU99] ITU-T SG-16. *WCDMA Error Patterns*, January 1999.

[Jay81] N. S. Jayant. Effects of packet losses in waveform coded speech and improvements due to an odd-even sample-interpolation procedure. *IEEE Transaction on Communications*, 29(2), February 1981.

[JF92] J. Johnson and A. Ferreira. Sum-difference stereo transform coding. In *IEEE ICASSP*, pages 569–571, 1992.

[JHDG96] J. Johnson, J. Herre, M. Davis, and U. Gbur. MPEG-2 NBC audio - stereo and multichannel coding methods. In *AES 101st convention*, AES preprint 4383, Los Angeles, CA, November 1996.

[K*et al.*97] C.-C. Jay Kuo and *et al.* *Multithreshold wavelet codec (MTWC)*, November 1997. Doc. N. WG1N665.

[KJ01] S. Kuo and J. D. Johnston. A study of why cross channel prediction is not applicable to perceptural audio coding. *IEEE Signal Processing Letters*, 8(9), September 2001.

[LDJC83] S. Lin and Jr D. J. Costello. *Error Control Coding: Fundamentals and Applications*. Prentice Hall, Englewood Cliffs, NJ, 1983.

[Lee99] J. Lee. Optimized quadtree for karhunen-loève transform in multispectral image coding. *IEEE Transactions on Image Processing*, 8(4):453–461, April 1999.

[Mat] SQAM Sound Quality Assessment Material. *http://www.tnt.uni-hannover.de/project/mpeg/audio/sqam/*.

[MS77] F. J. Macwilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, Netherlands, 1977.

[PA93] K. K. Paliwal and B. S. Atal. Efficient vector quantization of LPC parameters at 24 bits/frame. *IEEE Transactions on Speech and Audio Processing*, 1(1), January 1993.

[PB86] J. Princen and A. Bradley. Analysis/syhthesis filter bank design based on time domain aliasing cancellation. *IEEE transaction on acoustics, speech, and signal processing*, ASSP-34(5), October 1986.

[PFTV92] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C, the Art of Scientific Computing*. Cambridge University Press, second edition edition, 1992.

[PKKS97] S. Park, Y. Kim, S. Kim, and Y. Seo. Multi-layer bit-sliced bit-rate scalable audio coding. In *AES 103rd convention*, AES preprint 4520, New York, NY, September 26–29 1997.

[PM93] W. Pennebaker and J. Mitchell. *JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold, New York, 1993.

[PS00] T. Painter and A. Spanias. Perceptual coding of digital audio. *Proceedings of the IEEE*, 88(4), April 2000.

[RADH87] Jr. R. A. Damon and W. R. Harvey. *Experimental Design ANOVA, and Regression*. Harper & Row, New York, 1987.

[RW85] S. Roucos and A. Wilgus. High quality time-scale modification of speech. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 493–496, 1985.

[SAK99] Y. Shen, H. Ai, and C.-C. Kuo. A progressive algorithm for perceptual coding of digital audio signals. In *the 33rd Annual Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, Octorber 1999.

[Sha93] J. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE transaction on signal processing*, 41(12), December 1993.

[SS99] D. Sinha and C.-E.W. Sundberg. Unequal error protection methods for perceptual audio coders. In *1999 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2423–2426, Phoenix, AZ, March 15–19 1999.

[STR95] J. Saghri, A. Tescher, and J. Reagan. Practical transform coding of multispectral imagery. *IEEE Signal Processing Magazine*, 12(1):32–43, January 1995.

[TDD+94] C. Todd, G. Davidson, M. Davis, L. Fielder, B. Link, and S. Vernon. AC-3: Flexible perceptual coding for audio transmission and storage. In *AES 96th convention*, AES preprint 3796, Amsterdam, February 1994.

[VA01] M. S. Vinton and E. Atlas. A scalable and progressive audio codec. In *IEEE ICASSP 2001*, Salt Lake City, Utah, USA, May 2001.

[VR93] W. Verhelst and M. Roelands. An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 554–557, 1993.

[WGDP88] O. J. Wasem, D. J. Goodman, C. A. Dvorak, and H. G. Page. The effects of waveform substitution on the quality of PCM packet communications. *IEEE Transaction on Acoustics, Speech and Signal Processing*, 36(3), March 1988.

[WHZ00] D. Wu, Y. T. Hou, and Y. Zhang. Transporting real-time video over the internet: Challenges and approaches. *Proceedings of the IEEE*, 88(12):1855–1877, December 2000.

[WV91] R. Waal and R. Veldhuis. Subband coding of stereophonic digital audio signals. In *IEEE ICASSP*, pages 3601–3604, 1991.

[WZZZ01] G. Wang, Q. Zhang, W. Zhu, and J. Zhou. Channel-adaptive error protection for scalable audio over channels with bit errors and packet erasures. In *IEEE Globecom'01*, November 2001.

[YAK02] D. Yang, H. Ai, and C.-C. Kuo. Progressive multichannel audio codec (PMAC) with rich features. In *International Conference on Acoustics, Speech, and Signal Processing*, Orlando, FL, May 13–17 2002.

[YAKK00a] D. Yang, H. Ai, C. Kyriakakis, and C.-C. Kuo. An explorition of Karhunen-Loève transform for multichannel audio coding. In *Proc. SPIE on Digital Cinema and Microdisplays*, volume 4207, pages 89–100, Boston, MA, November 5–8 2000.

[YAKK00b] D. Yang, H. Ai, C. Kyriakakis, and C.-C. Kuo. An inter-channel redundancy removal approach for high-quality multichannel audio compression. In *AES 109th convention*, AES preprint 5238, Los Angeles, CA, September 2000.

[YAKK01a] D. Yang, H. Ai, C. Kyriakakis, and C.-C. Kuo. Adaptive Karhunen-Loève transform for enhanced multichannel audio coding. In *Proc. SPIE on Mathematics of Data/Image Coding, Compression, and Encryption IV*, volume 4475, pages 43–54, San Diego, CA, July 29–August 3 2001.

[YAKK01b] D. Yang, H. Ai, C. Kyriakakis, and C.-C. Kuo. Embedded high-quality multichannel audio coding. In *Conference on Media Processors, Part of the Symposium on Electronic Imaging 2001*, San Jose, CA, January 21–26 2001.

[YAKK02a] D. Yang, H. Ai, C. Kyriakakis, and C.-C. Kuo. Design of progressive syntax-rich multichannel audio codec. In *Proc. SPIE*, pages 121–132, San Jose, CA, January 2002.

[YAKK02b] D. Yang, H. Ai, C. Kyriakakis, and C.-C. Kuo. Error-resilient design of high fidelity scalable audio coding. In *Proc. SPIE on Digital Wireless Communications IV*, volume 4740, Orlando, FL, April 1–5 2002.

[YAKK02c] D. Yang, H. Ai, C. Kyriakakis, and C.-C. Kuo. High fidelity multichannel audio coding with Karhunen-Loève transform. Submitted for second-round review, February 2002.

[YFT+99] C. W. Yung, H. F. Fu, C. Y. Tsui, R. S. Cheng, and D. George. Unequal error protection for wireless transmission of mpeg audio. In *ISCAS'99 Proc. of the 1999 IEEE International Symposium on Circuits and Systems*, pages 342–345, Orlando, FL, May 30–June 2 1999.

[ZF90] E. Zwicker and H. Fastl. *Psychoacoustics, facts, and models*. Springer-Verlag, Berlin, 1990.

[ZL01] J. Zhou and J. Li. Scalable audio streaming over the internet with network-aware rate-distortion optimization. In *IEEE International Conference on Multimedia and Expo 2001*, Tokyo, Japan, August 2001.

[ZZXZ01] J. Zhou, Q. Zhang, Z. Xiong, and W. Zhu. Error resilient scalable audio coding (ERSAC) for mobile applications. In *Proc. Multimedia Signal Processing Workshop*, Cannes, France, October 2001.

Appendix A

Descriptive Statistics and Parameters

A.1 Mean

One of several items of interest in a set of observations is a measure of the central or the representative overall value. While such values as the mode, the median, and the midpoint of the range of values are occasionally considered, the most commonly used measure is the average, generally referred to as the arithmetic mean, or simply the mean. The mean is obtained by dividing the total of all observations by the number of observations as given below

$$\bar{y} = \frac{\sum_i Y_i}{n} = \frac{Y.}{n}, \tag{A.1}$$

where $\bar{y}$ is the common symbol for the mean, and n is the number of observations in the sample, and $Y.$ represents the sum of the observations. Table A.1 list one sample data. The mean for the data in this table is

$$\bar{y} = \frac{Y.}{n} = \frac{1795}{4} = 448.75,$$

and is an estimate of the unknown population parameter μ.

Table A.1: Weaning weights of four charolais steers (in pounds)

steer number	weight
1	$420 = Y_1$
2	$480 = Y_2$
3	$430 = Y_3$
4	$465 = Y_4$
sum=1795=$\sum_i Y_i = Y.$	

A.2 Variance

When dealing with biological data, one is continually confronted with variability among observations. While various measures of this variability are used, the one of primary interest in statistical analysis is known as the *variance*. The variance in a sample or a group of observations is measured as the sum of the squares of the deviations from the sample mean divided by one fewer than the number of observations. It has been shown that the division by the total number of observations leads to a biased estimate of the population variance, while the division by one fewer

than the total number of observations leads to an unbiased estimate. The formula for calculating the variance in any sample set of data can be written

$$s^2 = \frac{\sum_i Y_i - \bar{y}^2}{n-1}, \tag{A.2}$$

where s^2 is the symbol used for the variance of the observations in the sample and n is the number of observations. The value s^2 is an estimate of the population variance, a parameter designated σ^2. The deviation $Y_i - \bar{y}$ is often written y_i, leading to the frequent use of the symbol $\bar{Y}$ for the mean, since $\sum_i y_i = 0$.

Table A.2: Variance calculation using deviations from the mean

Y_i	$Y_i - \bar{y}$	$(Y_i - \bar{y})^2 = y_i^2$
420	-28.75	826.5625
480	31.25	976.5625
430	-18.75	351.5625
465	16.25	264.0625
$Y_\cdot = 1795$	$\sum_i Y_i - \bar{y} = 0.00$	$\sum_i Y_i - \bar{y}^2 = 2418.7500 = \sum_i y_i^2$
$\bar{y} = \frac{Y_\cdot}{n} = \frac{1795}{4} = 448.75$		$s^2 = \frac{\sum_i Y_i - \bar{y}^2}{n-1} = \frac{\sum y_i^2}{n-1} = \frac{2418.7500}{3} = 806.25$

To calculate the variance of observations in Table A.1, we present Table A.2 which shows operations involved in calculating the variance using (A.2). The variance of 806.25 is an estimate of the population variance σ^2. While calculating the variance using (A.2) is relatively easy with only a few observations, it is quite cumbersome

when a large number of observations are involved. It can be shown algebraically that

$$\frac{\sum_i Y_i - \bar{y}^2}{n-1} = \frac{\sum_i Y_i^2 - Y.^2/n}{n-1}. \tag{A.3}$$

The right-hand side of the equation, known as the "working formula," is easier to use with a large number of observations. By using (A.3) to calculate the variance of observations given in Table A.1, we have

$$\begin{aligned} s^2 &= \frac{\sum_i Y_i^2 - Y.^2/n}{n-1} = \frac{807,925 - (1795)^2/4}{4-1} \\ &= \frac{807,925.00 - 805,506.25}{3} = \frac{2418.75}{3} = 806.25. \end{aligned}$$

A.3 Standard Deviation

While it is mathematically convenient to use squared deviations as a measure of dispersion or variation about the mean, it is normal to think of this variation in terms of the original values. We can merely take the square root of the variance to return to the original scale of measurement. The square root of the variance is known as the *standard deviation* and is expressed as

$$s = \sqrt{\frac{\sum_i Y_i^2 - Y.^2/n}{n-1}}, \tag{A.4}$$

where s is the symbol for the standard deviation of a sample from the population and is an estimate of the parameter σ. For the example under discussion, we have

$$s = \sqrt{806.25} = 28.39. \tag{A.5}$$

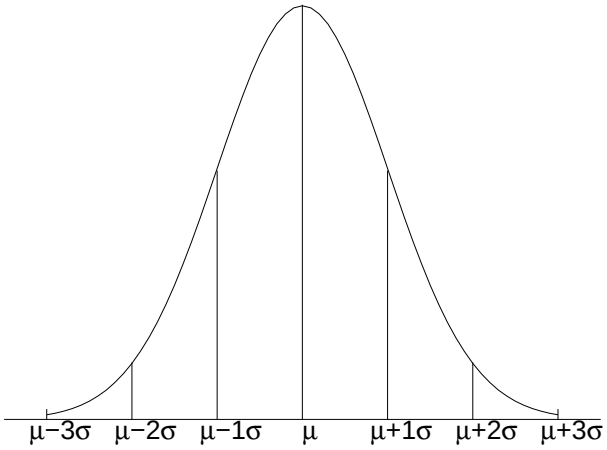


Figure A.1: Areas of the normal curve.

If the data of interest follow the normal curve, the population of observations would be visually described by the normal curve as shown in Figure A.1. That is, if one were able to plot the values of all individuals in a population, their frequencies would follow a bell-shaped curve, or distribution, with a clustering of observations not greatly different in size than the mean and a reduction in numbers of observations as the size of deviation from the mean increases in either direction. Statistical theory tells us that the area under the curve included by $\mu \pm 1\sigma$ would include 68.26 percent of the variates or observations, $\mu \pm 2\sigma$ would include 95.46 percent of the observations, and $\mu \pm 3\sigma$ would include 99.73 percent of the values. It can also be stated that 50 percent of the values would fall between $\mu \pm 0.674\sigma$, 95 percent of the values would

fall between $\mu \pm 1.965\sigma$, and 99 percent of the values would fall between $\mu \pm 2.576\sigma$. Assuming that the estimates of the population standard deviation and the mean from our sample are the population parameters (i.e., $\sigma = 28.38$ and $\mu = 448.75$), we would expect 50 percent of the weaning weights of Charolais steers to fall between 429.62 and 467.88 lb, 95 percent of the weaning weights to fall between 392.96 and 504.54 lb, and 99 percent to fall between 375.62 and 521.88 lb. It should be noted that four observations provide an extremely small number from which to make such estimations.

A.4 Standard Error of the Mean

If a number of samples are drawn from a population, the mean of each of the samples could be calculated, and we would find that we had variability among these sample means just as we had variability among individual observations in a single sample. If a large number of sample means were calculated, the mean of these sample means would be an estimate of the parameter μ. The means would also be distributed in a normal curve similar to that of the original population. However, the variation among the means would be smaller than σ_y since the mean of a sample will deviate less from the overall mean than will some members of the sample. The variance among a group of sample means would be measured as

$$s_{\bar{y}}{}^2 = \frac{\sum_{i=1}^{k} (\bar{y}_i - \bar{y}.)^2}{k-1}, \tag{A.6}$$

where ${s_{\bar{y}}}^2$ represents the variance of a group of sample means, $\bar{y}_i$ represents an individual sample mean, $\bar{y}.$ represents the mean of all sample means, and k is the number of means included. The value ${s_{\bar{y}}}^2$ would be an estimate of the parameter ${\sigma_{\bar{y}}}^2$.

A common situation arises when we have only one sample mean and wish to estimate the variance to be expected in a distribution of several means. This variance is estimated as

$$ {\sigma_{\bar{y}}}^2 = \frac{s^2}{n}, \tag{A.7} $$

where s^2 is the variance calculated among the observations within the sample, and n is the number of observations in the sample. The square root of the variance of the means, $s_{\bar{y}}$, would be the standard deviation of the mean. However, the standard deviation of a statistic such as a mean, whether calculated as the square root of (A.6) or estimated as the square root of (A.7), is normally referred to as the standard error of the statistic, and the *standard error of the mean* in this case. The term standard deviation is usually reserved to describe the variation among individual observations. In our sample, the standard error of the mean would be

$$ s_{\bar{y}} = \frac{s}{\sqrt{n}} = \frac{28.39}{\sqrt{4}} = 14.20. $$

It can be seen that the standard error of the mean is inversely related to the square root of the sample size. As the sample size increases, the standard error of the mean decreases. The standard error of the mean serves the same purpose for

the distribution of sample means as does the standard deviation for a distribution of individual observations. The standard error of the mean is an estimate of the parameter $\sigma_{\bar{y}}$ and a range of $\mu \pm 1\sigma_{\bar{y}}$ includes 68.26 percent of a population of means and a range of $\mu \pm 1.96\sigma_{\bar{y}}$ includes 95 percent of means. The standard error of the mean is used frequently in statistics to indicate what amount of variation would be expected with continued sampling of a population of means.

A.5 Confidence Interval

The standard error of a statistic is used frequently to develop what is termed a confidence interval. A *confidence interval* is a range between upper and lower limits, which is expected to include the true population value of a parameter at a selected level of probability. The upper and lower limits are referred to as a *confidence limits* and are a function of a t value for a given level of probability and the standard error of the statistic.

The necessary t values are found in Table A.3 and are derived from the t distribution developed by William S. Gossett (Student, 1908) and perfected by R. A. Fisher (1926). W. S. Gossett (1876-1937) was a brewer and statistician who published under the name of Student. R. A. Fisher (1890-1962) was one of the pioneers

in the field of statistics and made outstanding contributions in a great many areas of statistical theory and application. Let us define the statistic

$$t = \frac{\bar{y} - \mu}{s_{\bar{y}}}, \tag{A.8}$$

where $\bar{y}$, $s_{\bar{y}}$ and μ are the sample mean, the standard deviation and the parameter, respectively. The curve for the t distribution is symmetric, and as the degrees of freedom increase, the t distribution comes closer to the normal curve in form. Values for degrees of freedom of ∞ are those of the normal distribution.

The t distribution has found great utility in statistical procedures, and its application with regard to statistics other than the mean, and their standard errors, can be found in [RADH87].

If we wished to develop the 95 percent confidence interval about the mean of a set of observations, we would calculate the confidence limits as

$$\text{Lower confidence limit} = -t_{0.05} s_{\bar{y}}, \tag{A.9}$$

$$\text{Upper confidence limit} = +t_{0.05} s_{\bar{y}}, \tag{A.10}$$

where the t value is found in the table of the t distribution. *i.e.* Table A.3, under the column headed 0.05 level of probability and in the row for $n-1$ degrees of freedom,

where n is the number of observations in the sample. The confidence interval about the population mean can then be written

$$\bar{y} - t_{0.05} s_{\bar{y}} \leq \mu \leq \bar{y} + t_{0.05} s_{\bar{y}}. \tag{A.11}$$

The 95 percent confidence interval for the mean if the four observations in Table A.1 would be from

$$448.75 - (3.182)(14.20) \text{ to } 448.75 + (3.182)(14.20)$$

or

$$403.57 \text{ to } 493.93.$$

Calculation of this interval leads to the statement that we feel 95 percent confident that the population mean μ lies between 403.57 and 493.93 lb. The interval is commonly written

$$\bar{y} \pm t_{0.05} s_{\bar{y}}, \tag{A.12}$$

which in our example would be

$$448.75 \pm 45.18.$$

Table A.3: Distribution of t: two-tailed tests.

	Probability of a larger value of t, sign ignored								
df	0.500	0.400	0.300	0.200	0.100	0.050	0.020	0.010	0.001
1	1.000	1.376	1.963	3.078	6.314	12.706	31.821	63.657	636.619
2	0.816	1.061	1.386	1.886	2.920	4.303	6.965	9.925	31.598
3	0.765	0.978	1.250	1.638	2.353	3.182	3.541	5.841	12.941
4	0.741	0.941	1.190	1.533	2.132	2.776	3.747	4.604	8.610
5	0.727	0.920	1.156	1.476	2.015	2.571	3.365	4.032	6.859
6	0.718	0.906	1.134	1.440	1.943	2.447	3.143	3.707	5.959
7	0.711	0.896	1.119	1.415	1.895	2.365	2.998	3.499	5.405
8	0.706	0.889	1.108	1.397	1.860	2.306	2.896	3.355	5.041
9	0.703	0.883	1.100	1.383	1.833	2.262	2.821	3.250	4.781
10	0.700	0.879	1.093	1.372	1.812	2.228	2.764	3.169	4.587
11	0.697	0.876	1.088	1.363	1.796	2.201	2.718	3.106	4.437
12	0.695	0.873	1.083	1.356	1.782	2.179	2.681	3.055	4.318
13	0.694	0.870	1.079	1.350	1.771	2.160	2.650	3.012	4.221
14	0.692	0.868	1.076	1.345	1.761	2.145	2.624	2.977	4.140
15	0.691	0.866	1.074	1.341	1.753	2.131	2.602	2.947	4.073
16	0.690	0.865	1.071	1.337	1.746	2.120	2.583	2.921	4.015
17	0.689	0.863	1.069	1.333	1.740	2.110	2.567	2.898	3.965
18	0.688	0.862	1.067	1.330	1.734	2.101	2.552	2.878	3.922
19	0.688	0.861	1.066	1.328	1.729	2.093	2.539	2.861	3.883
20	0.687	0.860	1.064	1.325	1.725	2.086	2.528	2.845	3.850
21	0.686	0.859	1.063	1.323	1.721	2.080	2.518	2.831	3.819
22	0.686	0.858	1.061	1.321	1.717	2.074	2.408	2.819	3.792
23	0.685	0.858	1.060	1.319	1.714	2.069	2.500	2.807	3.767
24	0.685	0.857	1.059	1.318	1.711	2.064	2.492	2.797	3.745
25	0.684	0.856	1.058	1.316	1.708	2.060	2.485	2.787	3.725
26	0.684	0.856	1.058	1.315	1.706	2.056	2.479	2.779	3.707
27	0.684	0.855	1.057	1.314	1.703	2.052	2.473	2.771	3.690
28	0.683	0.855	1.056	1.313	1.701	2.048	2.467	2.763	3.674
29	0.683	0.854	1.055	1.311	1.699	2.045	2.462	2.756	3.659
30	0.683	0.854	1.055	1.310	1.697	2.042	2.457	2.750	3.646

Appendix B

Karhunen-Loève Expansion

B.1 Definition

Let the M-by-1 vector $\mathbf{u}(n)$ denote a data sequence drawn from a wide-sense stationary process of zero mean and correlation matrix $\mathbf{R}$. Let $\mathbf{q}_1$, $\mathbf{q}_2$, ..., $\mathbf{q}_M$ be eigenvectors associated with the M eigenvalues of the matrix $\mathbf{R}$. Vector $\mathbf{u}(n)$ may be expanded as a linear combination of these eigenvectors as follows:

$$\mathbf{u}(n) = \sum_{i=1}^{M} c_i \mathbf{q}_i \tag{B.1}$$

The coefficients of the expansion are zero-mean, uncorrelated random variables defined by the inner product

$$c_i(n) = \mathbf{q}_i^H \mathbf{u}(n), \ i = 1, 2, \ldots, M. \tag{B.2}$$

The representation of random vector $\mathbf{u}(n)$ described by (B.1) and (B.2) is the discrete-time version of the *Karhunen-Loève expansion*. In particular, (B.2) is the analysis part of the expansion in that it defines $c_i(n)$ in terms of the input vector $\mathbf{u}(n)$. On the other hand, (B.1) is the "synthesis" part of the expansion in that it reconstructs the original input vector $\mathbf{u}(n)$ from $c_i(n)$. Given the expansion of (B.1), the definition of $c_i(n)$ in (B.2) follows directly from the fact that the eigenvectors $\mathbf{q}_1$, $\mathbf{q}_2$, ..., $\mathbf{q}_M$ form an orthonormal set, assuming they are all normalized to have the unit length. Conversely, this same property may be used to derive (B.1), given (B.2).

B.2 Features and Properties

The coefficients of the expansion are random variables characterized by the following properties:

$$E[c_i(n)] = 0, \ i = 1, 2, \ldots, M, \tag{B.3}$$

and

$$E[c_i(n)c_j^*(n)] = \begin{cases} \lambda_i, & i = j \\ 0, & i \neq j \end{cases} \tag{B.4}$$

Equation (B.3) states that all coefficients of the expansion have zero mean, which follows directly from (B.2) and the fact that random vector $\mathbf{u}(n)$ is itself assumed to have the zero mean. Equation (B.4) states that coefficients of the expansion

are uncorrelated and that each one of them has a mean-square value equal to its respective eigenvalue. The second equation can be easily derived as below:

$$\begin{aligned} E[c_i(n)c_j^*(n)] &= E[(\mathbf{q}_i^H\mathbf{u}(n))(\mathbf{q}_j^H\mathbf{u}(n))^H] \\ &= \mathbf{q}_i^H E[\mathbf{u}(n)\mathbf{u}(n)^H]\mathbf{q}_j \\ &= \mathbf{q}_i^H\mathbf{R}\mathbf{q}_j \\ &= \lambda_j\mathbf{q}_i^H\mathbf{q}_j \\ &= \begin{cases} \lambda_i, & i = j \\ 0, & i \neq j \end{cases} \end{aligned}$$

For a physical interpretation of the Karhunen-Loève expansion, we may view eigenvectors $\mathbf{q}_1$, $\mathbf{q}_2$, ..., $\mathbf{q}_M$ as coordinates of an M-dimensional space. Random vector $\mathbf{u}(n)$ can be represented by the set of its projections $c_1(n)$, $c_2(n)$, ..., $c_M(n)$ onto these axes, respectively. Moreover, we deduce from (B.1) that

$$\sum_{i=1}^{M} |c_i(n)|^2 = |\mathbf{u}(n)|^2 \tag{B.5}$$

where $|\mathbf{u}(n)|$ is the Euclidean norm of $\mathbf{u}(n)$. That is, coefficient $c_i(n)$ has an energy equal to that of the observation vector $\mathbf{u}(n)$ measured along the i^{th} coordinate. Naturally, this energy is a random variable whose mean value equals the i^{th} eigenvalue, as shown by

$$E[|c_i(n)|^2] = \lambda_i,\ i = 1, 2, \ldots, M. \tag{B.6}$$

This result follows directly from (B.2) and (B.4).

Appendix C

Psychoacoustics

Starting from the first generation of PCM/DPCM coding to the second generation of perceptual audio coding, psychoacoustics plays an important role in reducing the bit rate in audio compression. Most current audio coders achieve compression by exploiting the fact that "irrelevant" signal information is not detectable by even a well trained or sensitive listener. Irrelevant information is identified during signal analysis by incorporating several psychoacoustic principles in the coder design such as the absolute hearing threshold, simultaneous masking and temporal masking, etc. [PS00]. This section reviews some fundamental knowledge of psychoacoustic principles that are commonly used in perceptual audio coding.

C.1 Hearing Area

The hearing area is a plane in which audible sounds can be displayed. In its normal form, the hearing area is plotted with the frequency on a logarithmic scale as the abscissa, and the sound pressure level in dB on a linear scale as the ordinate. This

means that two logarithmic scales are used because the level is related to the logarithm of the sound pressure. The critical-band rate may also be used as the abscissa. This scale is more relevant to features of our hearing system than the frequency.

The usual display of the human hearing area is shown in Fig. C.1. On the right, ordinate scales are either the sound intensity in Watt per square meter (W/m^2) or the sound pressure in Pascal (Pa). The sound pressure level is given for a free-field condition relative to 2×10^{-1} Pa. The sound intensity level is plotted relative to 10^{-12} W/m^2. A range of about 15 decades in the intensity or 7.5 decades in the sound pressure (corresponding to a range of 150 dB in the sound pressure level) is encompassed by the ordinate scale. As to the abscissa, we must realize that our hearing organ produces sensations for pure tones within three decades in the frequency ranging from 20 Hz to 20 kHz. The actual hearing area represents that range, which lies between the threshold in quiet (the limit towards low levels) and the threshold of pain (the limit towards high levels). These thresholds are given in Fig. C.1 as solid and broken lines, respectively. These limits hold for pure tones in the steady state condition, i.e. for tones lasting longer than about 100 ms.

If speech is resolved into spectral components, the region it normally occupies can also be illustrated in the hearing area. In Fig. C.1, the range encompassed by speech sounds is indicated by the area hatched from the top left to the bottom right starting near 100 Hz and ending near 7 kHz. The area as indicated hold for normal speech, for example, as delivered in a small lecture hall. The components of music encompass a larger distribution in the hearing area as given in Fig. C.1

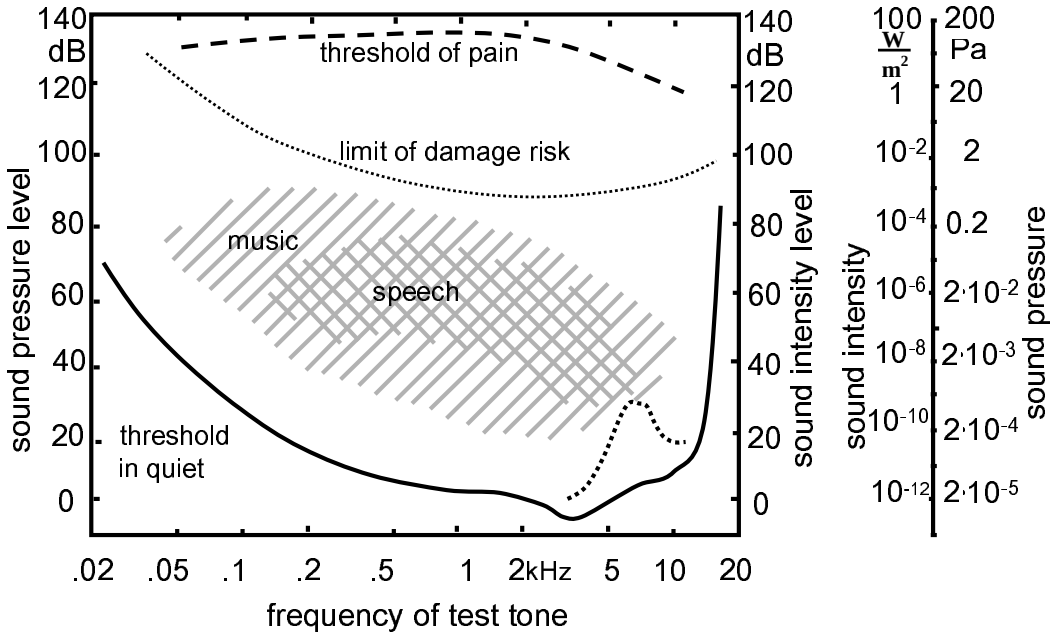


Figure C.1: Illustration of the hearing area, i.e. the area between the threshold on quiet and the threshold of pain. Also indicated are areas encompassed by music and speech, and the limit of damage risk. The ordinate scale is not only expressed in the sound pressure level but also in the sound intensity. The dotted part of the threshold in quiet stems from subjects who frequently listen to very loud music.

by different hatching. It starts at low frequencies near 40 Hz and reaches about 10 kHz. Including pianissimo and fortissimo, the dynamic range of music starts at sound pressure levels below 20 dB and reaches levels in excess of 95 dB. Extreme and rare cases are ignored for spectral distributions of music and speech displayed. It can be seen, however, that both areas are well above the threshold in quiet, which will be explained in more detail later.

The threshold in quiet is a function of the frequency, where the sound pressure level of a pure tone is just audible. This threshold can be measured quite easily by experienced or inexperienced subjects. The reproducibility of the threshold in quiet for a single subject is high and lies normally within ±3 dB.

The frequency dependence of the threshold in quiet can be measured precisely and quickly by Békésy tracking. A recording produced in this manner is shown in Fig. C.2. A whole recording from low to high frequencies lasts about 15 minutes. In order to show the reproducibility of such a tracking of threshold in quiet, two trackings, one with an upward sweep and another with a downward sweep in the frequency, are shown in Fig. C.2 for a frequency range between 0.3 and 8 kHz. Excursions of the zigzag reach as much as 12 dB (i.e. about ±6 dB). The middle of this zigzag curve is defined as the threshold in quiet.

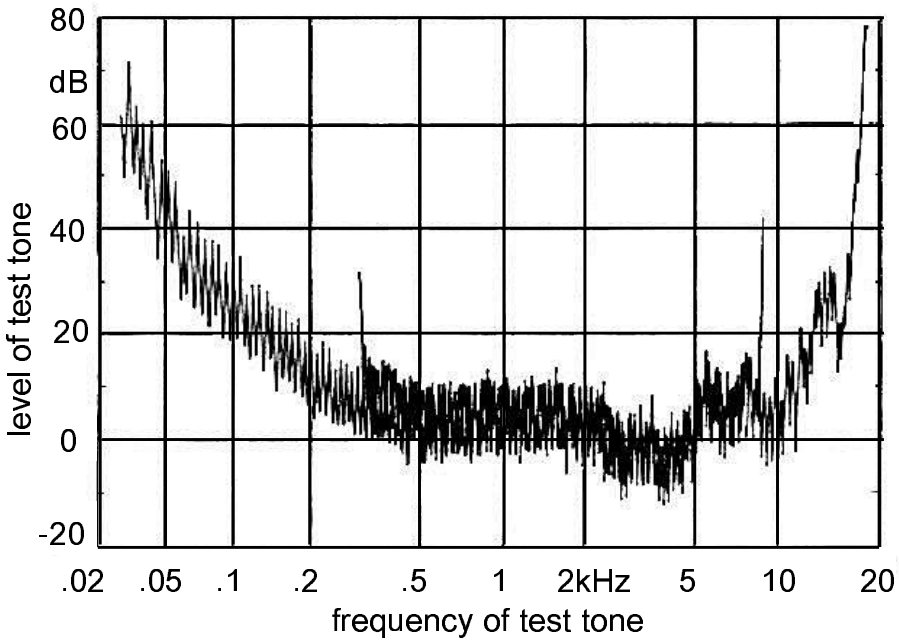


Figure C.2: Illustration of the threshold in quiet, i.e. the just-noticeable level of a test tone as a function of its frequency, registered with the method of tracking. Note that the threshold is measured twice between 0.3 and 8 kHz.

C.2 Masking

Masking plays a very important role in everyday life. For a conversation on pavements of a quiet street, for example, little speech power is necessary for speakers to understand each other. However, if a loud truck passes by, the conversation is severely disturbed. By keeping the speech power constant, one person can no longer hear from the other. There are two ways to deal with the masking phenomenon. We can either wait until the truck passed and then continue our conversation or raise our voice to produce more speech power and greater loudness. Our partner can hear the speech sound again. Similar effects take place in most pieces of music. One instrument may be masked by another if one of them produces high levels while the other remains faint. If the loud instrument pauses, the faint one becomes audible again. These are typical examples of simultaneous masking. To measure the effect of masking quantitatively, the masked threshold is usually determined. The masked threshold is the sound pressure level of a test sound (usually a sinusoidal test tone), necessary to be just audible in the presence of a masker. The mask threshold, in all but a very few special cases, always lies above the threshold in quiet. It is identical with the threshold in quiet when frequencies of the masker and the test sound are very different.

If the masker is increased steadily, there is a continuous transition between an audible (unmasked) test tone and one that is totally masked. This means that, besides total masking, partial masking also occurs. Partial masking reduces the

loudness of a test tone but does not mask the test tone completely. This effect often takes place in conversations.

Masking effects can be measured not only when masker and test sounds are present simultaneously, but also when they are not simultaneous. In the latter case, the test sound has to be a short burst or sound impulse which is present before the masker stimulus is switched on. The masking effect produced under these conditions is called the pre-stimulus masking, shorted to "premasking". This effect is not very strong. However, if the test sound is present after the masker is switched off, then a quite pronounced effect may occur. Since the test sound is present after the termination of the masker, the effect is called the post-stimulus masking, shorted to "postmasking".

C.2.1 Masking of Pure Tones

Several different masking effects are studied in Zwicker's book. We will discuss the masking effect of pure tones by complex tones here, since this is the most common situation occurring in music.

Most instrumental sounds in music are composed of a fundamental tone and many harmonics. The difference in timbre produced by different musical instruments depends on the frequency spectra of their harmonics. Whereas a flute produces primarily one signal component (i.e. the fundamental), a trumpet produces many harmonic partials and therefore elicits a much broader masking effect than a flute.

Fig. C.3 shows thresholds of pure tones, masked by a complex tone composed of a 200 Hz fundamental frequency and nine higher harmonics, all with the same amplitude but random in the phase. The mask thresholds are given for sound pressure levels of 40 dB and 60 dB of each partial. On the logarithmic frequency scale, the distance between partials is relatively large at low frequencies, but becomes very small between the ninth and tenth harmonic. Accordingly, dips between harmonics become smaller and smaller with the increasing frequency of the test tone. In the frequency range between 1.5 Hz and 2 Hz, the maxima and minima can hardly be distinguished. At frequencies above the last harmonic (in our case 2 Hz), the mask thresholds are flatter towards higher frequencies at higher levels of the masking complex. At frequencies one to two octaves above the highest spectral component, mask thresholds approach the threshold in quiet. In music, many complex tones, each composed of many harmonics, are used at the same time. This means that the corresponding masking effect can be assumed to produce shapes similar to those outlined in Fig. C.3. However, the minima between lines become even smaller because the density of lines is higher.

It should be noted that non-random phase conditions of components lead to temporal envelopes of the sound that can be described as impulsive. Consequently, temporal effects in masking may become a crucial factor in determining the mask threshold. Effects of this kind are discussed in the following section.

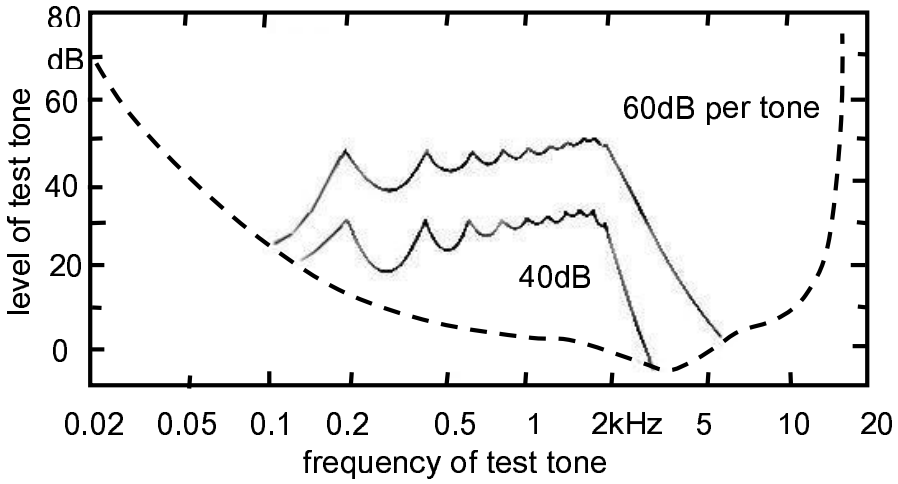


Figure C.3: The level of test tone masked by ten harmonics of 200 Hz as a function of the frequency of the test tone. Levels of the individual harmonics of an equal size are given as the parameter.

C.2.2 Temporal Effects

Masking in steady-state conditions with long-lasting test and masking sounds, was described in previous sections. However, the transmission of information in music or speech implies a strong temporal structure of the sound. Loud sounds are followed by faint sounds and vice versa. In speech, vowels generally represent the loudest parts whereas consonants are relatively faint. A plosive consonant is a typical example of a sound that is often masked by a preceding loud vowel. The effect occurs not only because of the reverberation of the room in which speech is received, but also in free-field conditions, because of the temporal effects of masking which characterize our hearing system.

To measure these effects quantitatively, maskers of a limited duration are given and masking effects tested with short test-tone bursts or short pulses. Further, the short signal is shifted in time relative to the masker as illustrated in Fig. C.4, where

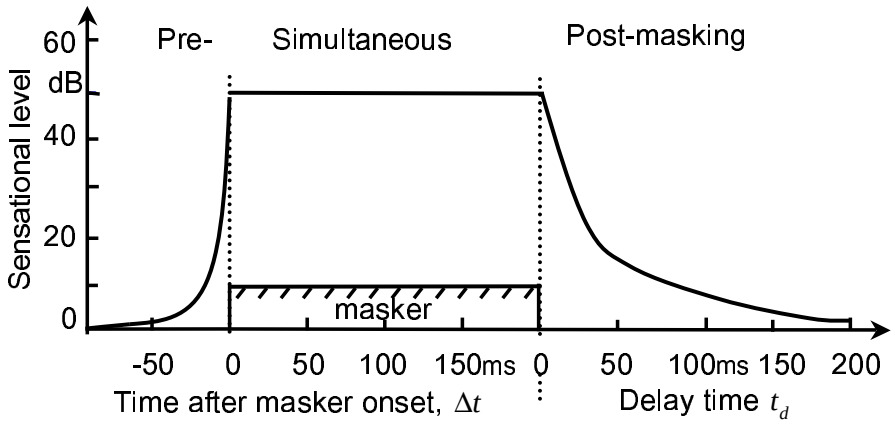


Figure C.4: Schematic drawing to illustrate and characterize regions within which premasking, simultaneous masking and postmasking occur. Note that postmasking uses a different time origin than premasking and simultaneous masking.

a 200 ms masker masks a short tone burst with as small a duration as possible and negligible in relation to the duration of the masker. In such a case, it is advantageous to use two different time scales. At first, the value Δt corresponds to the time relative to the onset of the masker – both positive and negative values exist. The second time scale starts at the end of the masker. This time is often called delay time and indicated by t_d. It is convenient to use as the ordinate not the sound pressure level of the test-tone burst, but the level above the threshold of this sound. This level is referred to as the sensation level. Three different temporal regions of masking relative to the presentation of the masker stimulus can be differentiated. Premasking takes place during that period of time before the masker is switched on. In this period, negative values of Δt apply. The period of pre-stimulus masking is followed by simultaneous masking when the masker and test sound are presented simultaneously. In this condition, Δt is positive. After the end of the masker, post-stimulus masking, normally called postmasking, occurs. During the time scale given

by positive delay time, t_d, the masker is not physically existent; nevertheless, is still produces masking.

The effect of postmasking corresponds to a decay in the effect of the masker and is more or less expected. Premasking, however, represents an effect that is unforeseen because it appears during a time before the masker is switched on. This does not mean, of course, that our hearing system is able to listen into the future. Rather, the effect is understandable if one realizes that each sensation – including premasking – does not exist instantaneously, but requires a build-up time to be perceived. If we assume a quick build-up time for loud maskers and a slower build-up time for faint test sounds, then we can understand why premasking exists. The time during which premasking can be measured is relatively short and lasts only about 20 ms. Postmasking, on the other hand, can last longer than 100 ms and ends after about a 200 ms delay. Therefore, postmasking is the dominant non-simultaneous temporal masking effect.

Appendix D

MPEG Advanced Audio Coding

MPEG AAC (Advanced Audio Coding) is the newest and most powerful member of the MPEG family for high-quality, digital audio compression. AAC allows for a flexible selection of operating modes and configurations. Applications of MPEG AAC range from low bit-rate Internet audio to multichannel broadcasting services. The review presented here is mostly based on the MPEG-2 AAC document ISO/IEC 13818-7 [ISOa]. The following sections describe the main feature of the AAC multichannel coding system.

D.1 Overview of MPEG-2 Advanced Audio Coding

Efficient audio coding removes redundant information from audio signals. Correlations between audio samples and statistics of sample's representation are exploited in order to remove redundancy. Frequency-domain and time-domain masking properties of the human auditory system [ZF90] are also exploited in order to remove

imperceptible signal content (irrelevancies). The frequency content of the audio signal is subdivided by means of a filter bank into subbands which are approximations of human auditory critical bands. The data rate reduction is achieved by quantizing the spectrum of the time signal according to perceptual models and may include a noiseless coding process. The steps to carry out these processes, as will be fully described in the following sections, lead to basic structure of the MPEG-2 AAC system as shown in Figures D.1 and D.2.

In order to allow the tradeoff between quality and memory/processing power requirements, the AAC system offers three profiles: the Main Profile, the Low Complexity (LC) Profile, and the Scalable Sampling Rate (SSR) Profile, where the main profile is the highest quality profile.

1. Main Profile

   In this configuration, the AAC system provides the best audio quality at any given data rate. With exception of the preprocessing tool, all parts of AAC tools may be used. The memory and the processing power required in this configuration are higher than those required in the low complexity profile configuration. It should also be noted that a main profile AAC decoder can decode a low-complexity-profile encoded bit stream.

2. Low Complexity Profile

   In this configuration, the prediction and preprocessing tools are not employed and the TNS order is limited. While the quality performance of the LC profile

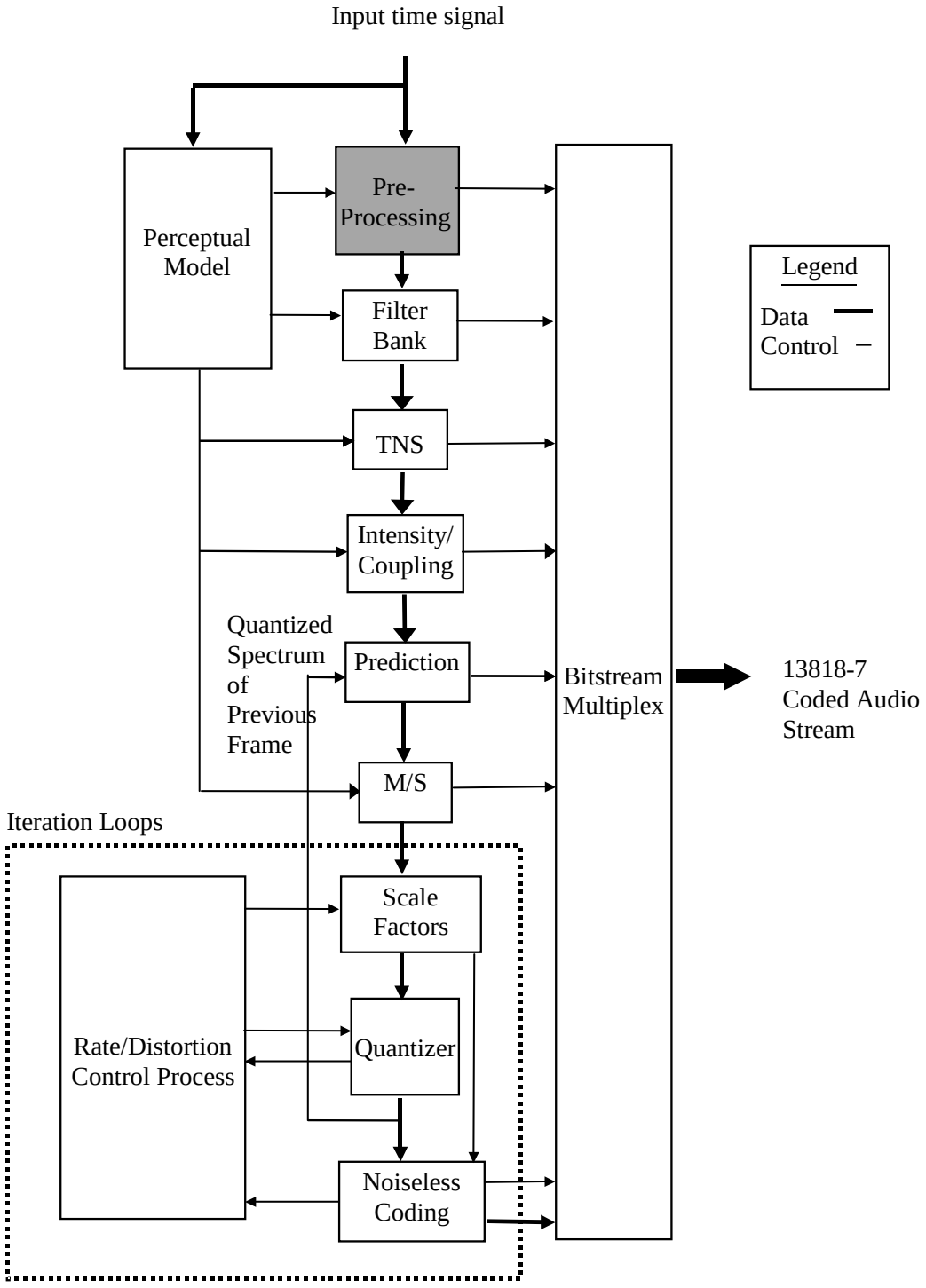


Figure D.1: The block diagram of the AAC encoder.

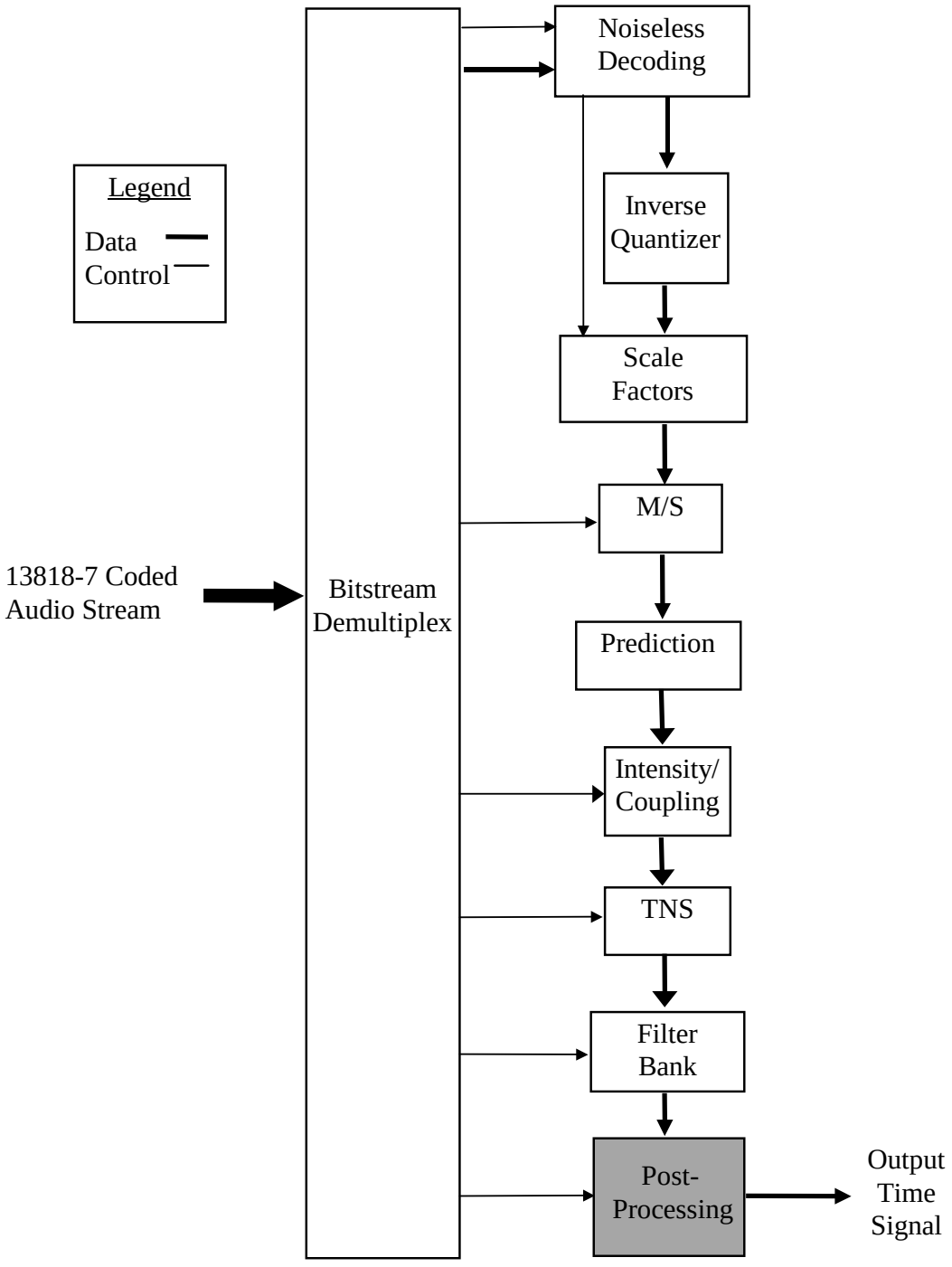


Figure D.2: The block diagram of the AAC decoder.

is very high, the memory and the processing power requirements are considerably reduced in this configuration.

3. Scalable Sampling Rate Profile

   In this configuration, preprocessing tools are required. They include a polyphase quadrature filter (PQF), gain detectors and gain modifiers. The prediction module is not used in this profile, and the TNS order and the bandwidth are limited. The SSR profile has a lower complexity than that of the main profile or the LC profile, and it can provide a frequency scalable signal.

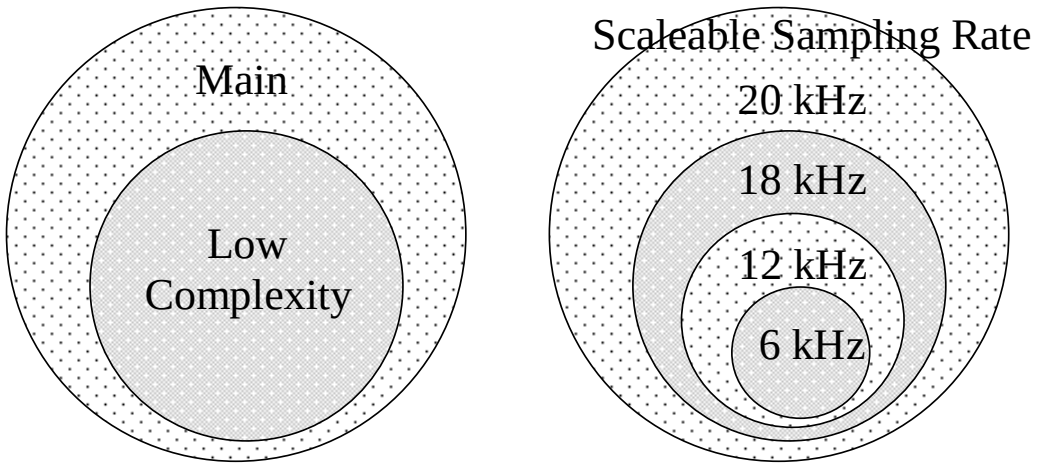


Figure D.3: Three AAC profiles.

D.2 Preprocessing

The preprocessing block is only used in the SSR profile. It consists of a polyphase quadrature filter (PQF), gain detectors and gain modifiers. The PQF has four unique bandwidth outputs. At the sampling rate of 48 kHz, it can provide four

output bandwidths at 24 kHz, 18 kHz, 12 kHz and 6 kHz. Gain detectors produce the gain control data compliant with the 13818-7 syntax. This information consists of the number of gain data, the index of position and the index of level. The gain control is applied in order to suppress pre-echo. The amplitudes of each PQF band are controlled independently by gain detectors, and gain control can be applied in conjunction with all types of window sequences. The time resolution of gain control is approximately 0.7 ms at the 48 kHz sampling rate. The step size of gain control is 2^n where n is an integer between -4 and 11. The signal can be amplified or attenuated by gain control. Gain modifiers control the gain of each PQF band. The modification function (MOD) is calculated by gain control data decoding and gain control function setting processes. Gain controlled signals are derived by applying MOD to the corresponding signal bands.

D.3 Filter Bank

A fundamental component of the MPEG AAC audio coder is the conversion of time domain signals at the input of the encoder into an internal time-frequency representation and its reverse process in the decoder. This conversion is done by a forward modified discrete cosine transform (MDCT) in the encoder, and an inverse modified discrete cosine transform (IMDCT) in the decoder. MDCT and IMDCT employ a technique called time domain aliasing cancellation (TDAC). Additional

information about the TDAC and the window-overlap-add process can be found in [PB86].

The analytical expression for MDCT can be written as

$$X_{ik} = \frac{1}{N} \sum_{n=0}^{N-1} x_{in} \cos\left(\frac{2\pi}{N}(n + n_0)(k + \frac{1}{2})\right), \quad 0 \leq k \leq N - 1.$$

Since sequence X_{ik} is odd-symmetric, coefficients from 0 to $N/2 - 1$ uniquely specify the transform. The analytical expression of IMDCT is

$$x_{in} = \sum_{k=0}^{N-1} X_{ik} \cos\left(\frac{2\pi}{N}(n + n_0)(k + \frac{1}{2})\right), \quad 0 \leq n \leq N - 1,$$

where

$$\begin{aligned} n &= \text{sample index}, \\ N &= \text{transform block length}, \\ i &= \text{block index}, \\ n_0 &= (N/2 + 1)/2. \end{aligned}$$

In the encoder, this processing takes the appropriate block of time samples, modulates them by an appropriate window function, and performs MDCT to ensure good frequency selectivity for the filter band. Each block of input samples is overlapped by 50% with the immediately preceding block and the following block. The length N of the transform block can be set either 2048 or 256 samples.

Because the window function has a significant effect on the filter-bank frequency response, the MPEG AAC filter bank has been designed to allow a change in the window shape to best adapt to input signal conditions. The shape of the window is varied simultaneously in the encoder and the decoder to allow the filter bank to efficiently separate spectral components of the input for a wider variety of input signals.

The use of 2048-sample time-domain transform improves coding efficiency for signals with complex spectra, but may create problems for transient signals. Quantization errors extending more than a few milliseconds before a transient event are not effectively masked by the transient itself. This leads to a phenomenon called pre-echo in which the quantization error from one transform block is spread in time and becomes audible. Long transforms are inefficient for coding signals which are transient in nature. Transient signals are best encoded with relatively short transform lengths. Unfortunately, short transforms produce inefficient coding of steady-state signals due to poorer frequency resolution.

MPEG AAC circumvents this problem by allowing the block length of the transform to vary as a function of signal conditions. Signals that are short-term stationary are best accommodated by the long transform, while transient signals are generally reproduced more accurately by short transforms. The transition between long and short transforms is seamless in the sense that aliasing is completely canceled in the absence of transform coefficient quantization.

D.4 Temporal Noise Shaping

A novel concept in perceptual audio coding is represented by the temporal noise shaping (TNS) tool of AAC [HJ96]. This tool is motivated by the fact that, despite the advanced state of today's perceptual audio coders, the handling of transient and pitched input signals still presents a major challenge. This is mainly due to the problem of maintaining the masking effect in the reproduced audio signal. In particular, coding is difficult because of temporal mismatch between the masking threshold and the quantization noise (also known as the pre-echo problem).

The TNS technique permits the coder to exercise a control over the temporal fine structure of quantization noise even within a filter-bank window. The concept of this technique can be outlined as follows.

- Time/frequency duality considerations.

  The concept of TNS uses the duality of time and frequency to extend known coding techniques by a new variant. It is well known that signals with an "un-flat" spectrum can be coded efficiently either by directly coding spectral values ("transform coding") or by applying predictive coding methods to time signals. Consequently, the corresponding dual statement relates to the coding of signals with an "un-flat" time structure, i.e. transient signals. Efficient coding of transient signals can thus be achieved by either directly coding time domain values or by employing predictive coding methods to spectral data. Such a predictive coding of spectral coefficients in the frequency domain in fact

constitutes the dual concept to the intra-channel prediction tool as described in Section D.6. While intra-channel prediction over time increases coder's spectral resolution, prediction over frequency enhances its temporal resolution.

- Noise shaping by predictive coding.

  If an open-loop predictive coding technique is applied to a time signal, the quantization error in the final decoded signal is known to be adapted in its Power Spectral Density (PSD) to the PSD of the input signal. Dual to this, if predictive coding is applied to spectral data over frequency, the temporal shape of the quantization error signal will be adapted to the temporal shape of the input signal at the output of the decoder. This effectively puts quantization noise under the actual signal and avoids problems of temporal masking, either in transient or pitched signals. This type of predictive coding of spectral data is referred to as the Temporal Noise Shaping (TNS) method.

Since the TNS preprocessing can be applied to either the entire spectrum, or only a part of the spectrum, time-domain noise control can be applied in any necessary frequency-dependent fashion. In particular, it is possible to use several predictive filters operating on distinct frequency (coefficient) regions.

D.5 Joint Stereo Coding

For further enhancement of its capabilities, MPEG-2 AAC includes two well-known techniques for joint stereo coding of signals: Mid/Side (M/S) stereo coding (also

know as "sum/difference coding") and intensity stereo coding. Both joint coding strategies can be combined by selectively applying them to different frequency regions. By using M/S stereo coding, intensity stereo coding, and L/R (independent) coding as appropriate, it is possible to avoid expensive overcoding due to Binaural Masking Level Depression, which account for noise imaging. Also, it achieve a significant saving in data rate very frequently. The concept of joint stereo coding in MPEG-2 AAC is discussed in detail in [JHDG96].

D.5.1 M/S Stereo Coding

M/S stereo coding is used to control the imaging of coding noise, as compared to the imaging of the original signal. In particular, this technique is capable of addressing the issue of "Binaural Masking Level Depression" [Bla83], where a signal at lower frequencies (below 2 kHz) can show up to 20 dB difference in masking thresholds depending on the phase of the signal and noise present (or lack of correlation in the case of noise). A second important issue is that of high-frequency time-domain imaging on transient or pitched signals. In either case, the properly coded stereo signal can require more bits than two transparently coded monophonic signals.

In MPEG-2 AAC, M/S stereo coding is applied within each channel pair of the multichannel signal, i.e. between a pair of channels that are arranged symmetrically on the left/right listener axis. In this way, imaging problems due to spatial unmasking are avoided to a larger degree.

M/S stereo coding can be used in a flexible way by selectively switching in time (on a block-by-block basis), as well as in frequency (on a scale-factor-band by scale-factor-band basis) [JF92]. The switching state (M/S stereo coding "on" and "off") is transmitted to the decoder as an array of signaling bits ("ms_used"). This can accommodate short time delays between L and R channels, and still accomplish both image control and signal-processing gain. While the amount of time delay allowed is limited, the time delay is greater than the interaural time delay and allows for control of most critical image issues [JF92].

D.5.2 Intensity Stereo Coding

The second important joint stereo coding strategy for exploiting inter-channel irrelevancy is the well known generic concept of *intensity stereo coding* [WV91]. This idea has been widely utilized in the past for stereophonic and multichannel coding under various names such as dynamic crosstalk and channel coupling. Intensity stereo coding exploits the fact that the perception of high frequency sound components mainly relies on the analysis of their energy-time envelopes [Bla83]. Thus, it is possible for certain types of signals to transmit a single set of spectral values shared among several audio channels with virtually no loss in sound quality. The original energy-time envelopes of coded channels are preserved approximately by means of a scaling operation such that each channel signal is reconstructed with its original level after decoding.

MPEG-2 AAC provides two mechanisms for applying generic intensity stereo coding.

- The first one is based on the "channel pair" concept as used for M/S stereo coding and implements an easy-to-use straight-forward coding concept that covers most of normal needs without introducing noticeable signaling overhead into the bit stream. For simplicity, this mechanism is referred to as the AAC intensity stereo coding tool. While the intensity stereo coding tool only implements joint coding within each channel pair, it may be used for coding of both two-channel as well as multichannel signals.

- The second one, which is a more sophisticated mechanism, is not restricted by the channel pair concept and allows better control of coding parameters. This mechanism is called the AAC coupling-channel element. It provides two functionalities: First, coupling channels may be used to implement generalized intensity stereo coding, where channel spectra can be shared across channel boundaries, including sharing among different channel pairs. The second functionality of the coupling channel element is to perform a downmix of additional sound objects into the stereo audio so that e.g. a commentary channel can be added to an existing multichannel program ("voice-over"). Depending on the profile, certain restrictions apply regarding consistency between coupling channels and target channels in terms of the window sequence and window shape parameters.

Thus, MPEG-2 AAC provides appropriate coding tools for many types of stereophonic audio from traditional two channel recordings to 5 to 7 channel surround sound material.

D.6 Prediction

Prediction is used to improve redundancy reduction. It is especially effective in handling stationary parts of a signal which are the most demanding parts in terms of the required data rate. Because a short window in the filter bank is used to handle signal changes (i.e. non-stationary signal characteristic), prediction is only used for long windows.

For each channel, prediction is applied to spectral components resulting from the spectral decomposition of the filter bank. For each spectral component up to 16 kHz, there is one corresponding predictor, resulting in a bank of predictors, where each predictor exploits the auto-correlation between spectral component values of consecutive frames.

The overall coding structure by using a filter bank with high spectral resolution implies the use of backward adaptive predictors. In this structure, predictor coefficients are calculated from preceding quantized spectral components in the encoder as well as in the decoder, to achieve high coding efficiency. No additional side information is needed for the transmission of predictor coefficients - as would be required for forward adaptive predictors. A second order backward-adaptive lattice structure

predictor is used for each spectral component so that each predictor is working on the spectral component values of two preceding frames. Predictor parameters are adapted to current signal statistics on a frame-by-frame base by using an LMS adaptation algorithm. If prediction is activated, the quantizer is fed with a prediction error instead of the original spectral component, resulting in a coding gain.

In order to guarantee that prediction is only used if it results in a coding gain, an appropriate predictor control is required and a small amount of predictor control information has to be transmitted to the decoder. For predictor control, predictors are grouped into scale factor bands. The predictor control information for each frame is determined in two steps. First, it is determined for each scale factor band whether or not prediction gives a coding gain, and all predictors belonging to a scale factor band are switched on/off accordingly. Then, it is determined whether the overall coding gain by prediction in the current frame compensates at least the additional bits needed for the predictor side information. Only in this case, prediction is activated and the side information is transmitted. Otherwise, prediction is not used in the current frame and only one signaling bit is transmitted.

D.7 Quantization and Coding

D.7.1 Overview

While all preceding steps perform some kind of preprocessing of the audio data, the real data rate reduction is done during the quantization process. The primary goal of

the module is to quantize the spectral data in such a way that the quantization noise fulfills the demands of the psychoacoustic model. At the same time, the number of bits needed to code this quantized spectrum must be below a certain limit, which is normally the average number of bits that is available for a block of audio data. This value depends on the sampling frequency and the desired data rate. In the AAC coder, a bit reservoir permits a variable bit distribution between consecutive audio blocks on a short-time basis. There are two constraints in this process: to fulfill the demands of the psychoacoustic model and to keep the number of needed bits below a certain number. Thus, the following two problems of the quantization process have to be addressed. What to do when the demands cannot be fulfilled with the available number of bits? What should be done if not all bits are needed to meet the demand?

The strategy to optimize the quantization process is not standardized, the only requirement is that the produced bit stream is compliant with the syntax as described in [ISOa]. One possible strategy is to use two nested iteration loops as described in this section. One important issue is the tuning between the psychoacoustic model and the quantization process, which can be viewed as one of the secretes of audio coding, since it requires a lot of experience and know-how.

The main features of the AAC quantization and coding process are:

- Non-uniform quantization.
- Huffman coding of spectral values with different tables.

- Noise shaping by amplification of groups of spectral values (the so-called scale factor bands). The information about the amplification is stored in scale factors.

- Huffman coding of differential scale factors.

D.7.2 Non-Uniform Quantization

The main advantage of a non-uniform quantizer is the built-in noise shaping depending on the amplitude of coefficients. The increase of the signal-to-noise ratio with an increasing signal energy is much lower than that of a linear quantizer. The range of quantized values is limited to ±8191. The Quantizer_stepsize represents the global quantizer step size. Thus, the quantizer can be changed in steps of 1.5 dB.

D.7.3 Coding of Quantized Spectral Values

Quantized coefficients created by the quantizer are coded by using Huffman Codes. A highly flexible coding method allows the use of several Huffman tables for one spectrum. Two and four-dimensional tables with and without the sign are available. The lossless coding process is described in detail in Section D.8. To calculate the number of bits needed to code a spectrum of quantized data, the coding process has to be performed, and the number of bits needed for the spectral data and the side information have to be accumulated.

D.7.4 Noise Shaping

The use of a non-uniform quantizer is not sufficient to fulfill the psychoacoustic demands. An additional method to shape the quantization noise is required. AAC uses the individual amplification of groups of spectral coefficients, the so-called scale factor bands. To be able to fulfill the requirements as efficiently as possible, it is desirable to shape the quantization noise in units similar to the critical bands of the human auditory system. Since the AAC system offers a relatively higher frequency resolution for long blocks of 23.43 Hz/line at the 48 kHz sampling frequency, it is possible to build groups of spectral values which reflect the bandwidth of critical bands very closely. Figure 4.2 shows the width of scale factor bands for long blocks at 44.1 kHz or 48 kHz sampling frequency. Note that the width of the scale factor bands is limited to 32 coefficients except for the last scale factor band. The total number of scale factor bands is 49 for long blocks.

The AAC system offers the possibility to individually amplify scale factor bands in a step of 1.5 dB. Noise shaping is achieved because amplified coefficients have larger amplitudes. Therefore, they will generally exhibit a higher signal-to-noise ratio after quantization. On the other hand, larger amplitudes normally need more bits to be coded, i.e. the bit distribution between scale factor bands is changed implicitly.

Amplification has to be performed in the decoder. For this reason, the amplification information, which is stored in the scale factors (in units of 1.5dB steps), has to be transmitted to the decoder.

D.7.5 Iteration Process

The decision on which scale factor band has to be amplified is, within certain limits, up to the encoder. Thresholds calculated by the psychoacoustic model are the most important criteria, but not the only ones, since the number of bits that can be used is limited. As already mentioned above, the iteration process described here is just one method to perform noise shaping. This method is however known to produce very good audio quality. Two nested loops, the so-called inner and outer iteration loops are used to determine optimal quantization. The description given here has been simplified to facilitate the understanding of the process.

The task of the inner iteration loop is to change the quantizer step size until the given spectral data can be coded within the number of available bits. For this purpose, an initial quantizer step size is chosen, the spectral data are quantized and the number of bits necessary to code the quantized data is counted. If this number if higher than the number of available bits, the quantizer step size is increased and the whole process is repeated.

The task of the outer iteration loop is to amplify spectral coefficients in all scale factor bands in a way such that the demands of the psychoacoustic model are fulfilled as much as possible.

1. At the beginning, no scale factor band is amplified.

2. The inner loop is called.

3. For each scale factor band, distortion caused by the quantization is calculated.

4. The real distortion is compared with the allowed distortion calculated by the psychoacoustic model.

5. If this result is the best result so far, it is stored. This is needed, since the iteration process does not necessarily converge.

6. Scale factor bands with a real distortion higher than the allowed distortion are amplified. At this point, different methods can be applied to determine the scale factor bands to be amplified.

7. If all scale factor bands have been amplified, the iteration process stops. The best result is restored.

8. If there is no scale factor band with a real distortion above the allowed distortion, the iteration process stops as well.

9. Otherwise, the process is repeated with new amplifications.

Some other conditions which cause termination of the outer iteration loop are not mentioned above. Since amplified parts of the spectrum need more bits for coding while the number of available bits is constant, the quantization step size has to be changed in the inner iteration loop to decrease the number of used bits. This mechanism moves bits from spectral regions where they are not needed to spectral regions where they are needed. This is also the reason that the result after an amplification in the outer loop may be worse than before. The best result should be stored after the termination of the iteration process.

Quantization and coding of short blocks is similar to those for long blocks. However, grouping and interleaving have to be taken into account. Both mechanisms are described in detail in Section D.8.

D.8 Noiseless Coding

The input to the noiseless coding module is the set of 1024 quantized spectral coefficients. As a first step a method of noiseless dynamic range compression may be applied to the spectrum. Up to four coefficients can be coded separately as magnitudes in excess of one, with a value of ± 1 left in the quantized coefficient array to carry the sign. The clipped coefficients are coded as integer magnitudes and an offset from the base of the coefficient array to mark their location. Since the side information for carrying clipped coefficients costs some bits, this noiseless compression is applied only if it results in a net saving of bits.

D.8.1 Sectioning

The noiseless coding module segments the set of 1024 quantized spectral coefficients into sections so that a single Huffman codebook is used to code each section. For reasons of coding efficiency, section boundaries can only be at scale factor band boundaries so that, for each section of the spectrum, one must transmit the length of the section, in scale factor bands, and the Huffman codebook number used for the section.

Sectioning is dynamic. It typically varies from block to block so that the number of bits needed to represent the full set of quantized spectral coefficients is minimized. This is done by using a greedy merge algorithm starting from the maximum possible number of sections, each of which uses the Huffman codebook with the smallest possible index. Sections are merged if the resulting merged section results in a lower total bit count, with merges that yield the greatest bit count reduction done first. If the sections to be merged do not use the same Huffman codebook, then the codebook with the higher index must be used.

Sections often contain only coefficients whose value is zero. For example, if the audio input is band limited to 20 kHz or lower, then the highest coefficients are zero. Such sections are coded with Huffman codebook zero, which is an escape mechanism that indicates that all coefficients are zero and it does not require that any Huffman codewords be sent for that section.

D.8.2 Grouping and Interleaving

If the window sequence is eight short windows, then the set of 1024 coefficients is actually a matrix of 8 by 128 frequency coefficients representing the time-frequency evolution of the signal over the duration of eight short windows. Although the sectioning mechanism is flexible enough to efficiently represent the 8 zero sections, *grouping* and *interleaving* provide greater coding efficiency. As explained earlier, coefficients associated with contiguous short windows can be grouped such that they share scale factors among all scale factor bands within the group. In addition,

coefficients within a group are interleaved by interchanging the order of scale factor bands and windows. To be more specific, it is assumed that, before interleaving, the set of 1024 coefficients c are indexed as

$$c[g][w][b][k],$$

where

g is the index on groups,

w is the index on windows within a group,

b is the index on scale factor bands within a window,

k is the index on coefficients within a scale factor band

and the right-most index varies most rapidly.

After interleaving, coefficients are indexed as

$$c[g][b][w][k].$$

This has the advantage of combining all zero sections due to band-limiting within each group.

D.8.3 Scale Factors

The coded spectrum uses one quantizer per scale factor band. The step size of each of these quantizers is specified as a set of scale factors and a global gain that normalizes these scale factors. In order to increase compression, scale factors associated with scale factor bands that have only zero-valued coefficients are not transmitted. Both the global gain and scale factors are quantized in 1.5dB steps. The global gain is coded as an 8-bit unsigned integer and the scale factors are differentially encoded relative to the previous scale factor (or global gain for the first scale factor) and then Huffman coded. The dynamic range of the global gain is sufficient to represent full-scale values from a 24-bit PCM audio source.

D.8.4 Huffman Coding

Huffman coding is used to represent n-tuples of quantized coefficients, with the Huffman code drawn from one of 11 codebooks. The spectral coefficients within n-tuples are ordered (low to high) and the n-tuple size is two or four coefficients. The maximum absolute value of the quantized coefficients that can be represented by each Huffman codebook and the number of coefficients in each n-tuple for each codebook is shown in Table D.1. There are two codebooks for each maximum absolute value, with each representing a distinct probability distribution function. The best fit is always chosen. In order to save on codebook storage (an important consideration in a mass-produced decoder), most codebooks represent unsigned values. For these

Codebook index	n-Tuple size	Maximum absolute value	Signed values
0		0	
1	4	1	yes
2	4	1	yes
3	4	2	no
4	4	2	no
5	2	4	yes
6	2	4	yes
7	2	7	no
8	2	7	no
9	2	12	no
10	2	12	no
11	2	16(ESC)	no

Table D.1: Huffman codebooks used in AAC.

codebooks, the magnitude of coefficients is Huffman coded and the sign bit of each non-zero coefficient is appended to the codeword.

Two codebooks require special note, i.e. codebook 0 and codebook 11. As mentioned previously, codebook 0 indicates that all coefficients within a section are zero. Codebook 11 can represent quantized coefficients that have an absolute value greater than or equal to 16. If the magnitude of one or both coefficients is greater than or equal to 16, a special *escape coding* mechanism is used to represent those values. The magnitude of coefficients is limited to no greater than 16 and the corresponding 2-tuple is Huffman coded. The sign bits, as needed, are appended to the codeword.