New Approaches to Fine-Grain Scalable Audio Coding

Mahmood Movassagh

Department of Electrical & Computer Engineering
McGill University
Montreal, Canada

December 2015

Research Thesis submitted to McGill University in partial fulfillment of the requirements for the degree of PhD.

In memory of my mother whom I lost in the last year of my PhD studies

To my father who has been the greatest support for my studies in my life

Abstract

Bit-rate scalability has been a useful feature in the multimedia communications. Without the need to re-encode the original signal, it allows for improving/decreasing the quality of a signal as more/less of a total bit stream becomes available. Using scalable coding, there is no need to store multiple versions of a signal encoded at different bit-rates. Scalable coding can also be used to provide users with different quality streaming when they have different constraints or when there is a varying channel; i.e., the receivers with lower channel capacities will be able to receive signals at lower bit-rates. It is especially useful in the client-server applications where the network nodes are able to drop some enhancement layer bits to satisfy link capacity constraints.

In this dissertation, we provide three contributions to practical scalable audio coding systems. Our first contribution is the scalable audio coding using watermarking. The proposed scheme uses watermarking to embed some of the information of each layer into the previous layer. This approach leads to a saving in bitrate, so that it outperforms (in terms of rate-distortion) the common scalable audio coding based on the reconstruction error quantization (REQ) as used in MPEG-4 audio. Another contribution is for the scalable coding based on bit-plane coding (BPC). Considering the properties of the residual signal, core-based bit-plane probabilities are provided for MPEG-4 audio scalable to lossless coding (SLS), which matches the quantization and coding performed in the core layer. Simulations show that proper consideration of the core layer parameters improves the bit-plane probabilities estimation compared to the existing method.

Perhaps the most important contribution is presented lastly, which is a very fine-grain scalable coding approach by designing a scalable entropy coding using a trellis-based optimization. In the proposed scheme, by constructing an entropy coding tree where the internal nodes can be mapped into the reconstruction points, the tree can be pruned at internal nodes to control the rate-distortion (RD) performance of the encoder in a fine-grain manner. A set of metrics and a trellis-based approach is proposed so that an appropriate path is generated on the RD plane. The results show the proposed method outperforms the scalable audio coding performed based on reconstruction error quantization as used in practical systems, e.g. in scalable advanced audio coding (S-AAC).

Sommaire

Scalabilité du débit binaire a été une caractéristique efficace en communication multimédia. Sans le besoin de ré-encoder le signal original, elle permet l'amélioration/la baisse de la qualité d'un signal dès qu'un plus/moins flux numérique total est disponible. Grâce au codage scalable, il n'est pas nécessaire de stocker de multiples versions d'un signal codé à des débits binaires différents. Le codage scalable peut également être utilisé à offrir aux utilisateurs différentes qualités de flux de bits lorsqu'ils ont des contraintes différentes ou lorsqu'il y a un canal variable, c.-à-d., les récepteurs ayant des capacités de canal inférieures seront capables de recevoir des signaux à des flux numériques plus bas. Ceci est particulièrement utile dans les applications client-serveur dans lesquelles les nœuds de réseaux sont capables de réduire certaines bits de couches d'amélioration afin de satisfaire aux contraintes de capacités de liaison.

Dans cette thèse, on fourni trois contributions pratiques au système de codage scalable. Notre première contribution est le codage scalable à l'aide de filigrane numérique. Le schéma proposé utilise le filigrane numérique afin d'intégrer les informations de chaque couche à la couche précédente. Cette approche permet d'économiser des débits binaires afin qu'elle puisse surpasser (en terme de taux de distorsion), le codage audio scalable commun basé sur la reconstitution d'erreur de quantification (REQ), tel qu'utilisé en audio MPEG-4. Une autre contribution est en codage scalable basé sur le codage par plan-de-bits (BPC). En tenant compte des propriétés du signal résiduel, les probabilités basées sur les noyaux de plans de bits sont fournies pour les audio MPEG-4 scalable aux codages sans pertes (SLS), qui correspond à la quantification et codage réalisé dans la couche centrale. Des simulations démontrent qu'une considération appropriée des paramètres de la couche centrale améliore l'estimation de probabilités du plan binaire comparé à la méthode existante. 9 Peut-être notre plus importante contribution est présentée en dernier lieu, celle de l'approche de codage scalable à grains très fins en concevant un codage entropique utilisant une optimisation à base de treillis. Dans le plan proposé, en construisant un arbre de codage entropique, où les noeuds internes peuvent être définis en points reconstitués, l'arbre peut être taillé en noeuds internes afin de contrôler la performance du taux de distorsion (RD) de l'encodeur d'une manière de grains très fins. Un ensemble de métriques et une approche à base de treillis sont proposés afin qu'un parcours adéquat est généré au niveau du taux de distorsion. Les résultats démontrent que la méthode proposée surpasse

le codage audio scalable réalisé basé sur la reconstitution d'erreur de quantification tel qu'utilisée dans des systèmes pratiques, p. ex. dans le codage audio scalable avancé (S-AAC).

Acknowledgments

I would like to express my immense gratitude to my supervisor, Prof. Peter Kabal for all of his support, guidance and insight during my PhD studies. I had the honor to be his last student, obtaining the opportunity to work with one of the most experienced and globally recognized experts in our field. It was not only his keen intellect and deep knowledge that helped me throughout the course of my PhD, but also his character and morality which I learned a lot from. His inspiration will remain with me for the rest of my life.

Also, I would like to thank Dr. Richard Rose and Dr. Philippe Depalle, my PhD advisory committee members, for their constructive comments and suggestions. I am also grateful to my colleagues at the TSP lab of McGill university, specially Joe and Sina for their companionship.

And my deepest gratitude is for my family, specially my parents, for their endless support in my life. My father, together with my late beloved mother were the greatest support for my studies, from the elementary school to the PhD degree.

Contents

List of Figures

List of Tables

List of Acronyms

3GPP	3rd Generation Partnership Project
AAC	Advanced Audio Coding
AC-3	Audio Codec 3
ADPCM	Adaptive Differential PCM
ALAC	Apple Lossless Audio Codec
ANMR	Average Noise-to-Mask Ratio
ASAC	Analysis/Synthesis Audio Codec
ATRAC	Adaptive Transform Acoustic Coding
BC/LSF	Backward Compatible/Low Sampling Frequency
BPC	Bit-Plane Coding
BPGC	Bit-Plane Golomb Code
BPP	Bit-Plane Probability
BSAC	Bit-Sliced Arithmetic Coding
CB	Critical Band
CELP	Code Excited Linear Prediction
ClipBPP	BPP Considering Clipping
CoreBPP	Core-Based BPP
DCT	Discrete Cosine Transform
DCT-IV	Discrete Cosine Transform type-IV
DFT	Discrete Fourier Transform
DPCM	Differential PCM
EC	Entropy Coding
ECSQ	Entropy-Constrained Scalar Quantizer
ESC	Enhanced Scalable Coder

EZK	Embedded Zerotree coding developed at King's college
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FLAC	Free Lossless Audio Codec
GAC	General Audio Coder
GQMF	Generalized QMF
HAS	Human Auditory System
HILN	Harmonic and Individual Lines Plus Noise
IntMDCT	Integer MDCT
KBD	Kaiser-Bessel Designed
LMS	Least Mean Square
LP	Linear Prediction
LPC	Linear Predictive Coding
LPCM	Linear PCM
LSB	Least Significant Bit
LSF	Lower Sampling Frequency
LTP	Long-Term Prediction
MDCT	Modified Discrete Cosine Transform
MIDI	Musical Instrument Digital Interface
MP3	MPEG-1 Layer 3
MPEG	Moving Picture Experts Group
MPEG-1	MPEG Phase 1
MPEG-2	MPEG Phase 2
MPEG-4	MPEG Phase 4
MSB	Most Significant Bit
MSE	Mean Square Error
MSLS	Modified SLS
MUSHRA	Multiple Stimuli with Hidden Reference and Anchor
NBC	Non-Backward Compatible
NE-QIM	Nearest Even QIM
NE-Odd	Nearest Odd QIM
NMR	Noise-to-Mask Ratio
NMT	Noise Masking Tone

NS-AAC	Non-Scalable AAC
NSR	Noise-to-Signal Ratio
PCM	Pulse Code Modulation
PEAQ	Perceptual Evaluation of Audio Quality
PNS	Perceptual Noise Substitution
PR	Perfect Reconstruction
QIM	Quantization Index Modulation
QMF	Quadrature Mirror Filter
RD	Rate-Distortion
REQ	Reconstruction Error Quantization
S-AAC	Scalable AAC
SFB	Scalefactor Band
SLS	Scalable to Lossless Coding
SMR	Signal to Mask Ratio
SNR	Signal to Noise Ratio
SPIHT	Set Partitioning in Hierarchial Trees
SPL	Sound Pressure Level
SQ	Scalar Quantization
STFT	Short Time Fourier Transform
TDA	Time Domain Aliasing
TDAC	Time Domain Aliasing Cancelation
TMN	Tone Masking Noise
TNS	Temporal Noise Shaping
TrelSEC	Trellis-Based Scalable Entropy Coding
Twin VQ	Transform domain Weighted Interleave Vector Quantization
UTQ	Uniform Threshold Quantizer
VPCM	Vector Pulse Code Modulation
VQ	Vector Quantization
WMSE	Weighted Mean Square Error
WSAC	Watermarking-based Scalable Audio Coding

Chapter 1

Introduction

The utility of audio and video compression (coding) is well known. For instance, the MPEG coding standards as efficient multimedia compression tools for use in personal computers and portable devices. You barely find someone who has not heard the term 'MP3' (MPEG-1 layer 3 [1]), which is one of the most popular audio compression standards available today. Many of us listen to MP3 music files regularly. MPEG-4 Advanced Audio Coding (AAC) [2, 3], Dolby Digital AC3 [4], and ATRAC [5] are among the other common audio coding standards. Other than the storage advantages, compression has a specific importance in telecommunications and multimedia streaming where there are bandwidth constraints. Compression provides low-bitrate coded version of an audio signal which can be used for transmissions over low-bandwidth channels, to meet the network constraints in digital streaming and etc.

By audio, we refer to all types of sounds around us. This can be human voice (speech), music, sounds in the nature, etc. The audio compression techniques are normally categorized into two different groups: model-based and waveform-based methods. In the model-based techniques the compression is performed based on the properties of the audio source. For example, in the case of speech, the source can be modeled as two parts: The vocal cords which generate periodic signals account for the pitch and intonation of a voice; and the vocal tract which determines the spectral resonances (formants) of the phonemes. If the information related to the first part is removed, the remaining part could be still understandable. This removal can lead to a compression ratio around $1000 : 1$. Note that, in practice the first part is coded as well, but at a very low bitrate.

Similarly there are model-based music coding techniques. The MIDI standard which is basically a music control protocol can also be considered as a music coder. In this standard, the three main parameters related to a musical note are stored which are: name of the note (equivalent to its perceived pitch), the duration and the intensity. Therefore, the information related to the musical instruments is removed. In fact, just the instrument's name is stored as a code. Again removing this high-rate information gives a great amount of compression. The removed information is related to the harmonics which determine an instrument's timbre. The instrument's sound is imitated by the MIDI decoder software or device at the time of playing a MIDI file (General MIDI).

On the other hand, there are waveform-based compression techniques which are used for all types of audio signals. In these techniques the compression is performed based on the properties of the generated waveform. The goal here is to have a compressed audio with an acceptable perceptual quality. In these methods, the properties of the human auditory system (HAS) are considered. These coders which use these properties are called perceptual coders. While most of the efficient speech coders are model-based, in general audio coding is mainly waveform-based. This dissertation focuses on the general audio coding techniques. This type of audio coding is also referred to as wide-band audio coding, since it covers all types of audio from speech to music. Speech is a narrow-band signal compared to the music which has a bandwidth more than 20 kHz. Although the human ear is not able to hear the frequencies components above this limit and they are normally discarded.

Not all audio coding techniques incur a loss of information of a digitized signal. In fact, there are also two major categories for audio coding in this aspect: Lossy coding and lossless coding. In lossy coding some of the information of the digitized audio signal is discarded. This information is normally what is found redundant in terms of audibility. A psychoacoustic model is used to determine to what degree the resulting distortion is important perceptually. On the other hand, there is lossless audio coding in which there is no loss and the original signal can be perfectly reconstructed at the decoder. Obviously, there is a tradeoff between the reconstructed audio quality and the bitrate required for these types of audio coding. The compression ratio is normally much higher in the lossy techniques while the audio quality is sacrificed.

1.1 Scalable Audio Coding

One specific type of audio coding which has become very useful during the past several years is the scalable audio coding providing bit-rate scalability. Bit-rate scalability is a desired feature in multimedia communications. Without the need to re-encode the original signal, it allows for improving/decreasing the quality as more/less of a total bit stream becomes available. This way there is no need to store multiple versions of a signal encoded at different bit-rates. Scalable coding can also be used to provide users with different quality streaming when they have different constraints or when there is a varying channel; i.e. the receivers associated with lower channel capacities will be able to receive signals at lower bit-rates.

Scalable coding is especially useful in client-server applications where the network nodes are able to drop higher layer packets to satisfy link capacity constraints. It can also provide robustness to packet loss for transmission over packet networks. In such systems, a very robust channel coding is performed for the core bitstream so that all the receivers can receive it almost without any loss, and the rest of the bitstream is sent using normal channel coding. Therefore if the packets are lost from this part of the bitstream, the signal can be still reconstructed from the core bitstream.

In scalable coding systems typically there is a base layer which gives the core bitstream, and there are enhancement layers which form the rest of the bitstream [6]. The base layer bitstream is used to reconstruct the original input audio with tolerable audible artifacts. The enhancement layers on the other hand, can be used (if received) to enhance the quality of the signal reconstructed from the base layer[1].

Scalability can also be viewed in other ways. Signal-to-noise ratio (SNR), noise-to-mask ratio[2] (NMR), and the audio bandwidth are among the other measures considered in a scalable coding system. In SNR scalability the signal-to-noise ratio of the reconstructed signal is improved by decoding the enhancement layers bitstream, in NMR scalability there is the same enhancement strategy for NMR instead of SNR and in bandwidth scalability the base layer gives a minimum bandwidth of the signal and using the enhancement layers the bandwidth can be increased. The latter is sometimes referred to as sample rate scalability

[1] Scalable coding is also sometimes refereed to as successive refinement or multi-layer coding in the literature.

[2] The mask refers to the masking threshold obtained from a psychoacoustic model in the perceptual coding systems. We will discuss it in detail in Chapter 3.

since sampling frequency for a signal is directly related to its bandwidth. Using all the above measures finally lead to the scalability in bitrate which is of the special interest in a communication system.

There has been research going on in scalable coding of speech and audio signals. Scalable audio coding using wavelet transform [7], bit-plane based scalable coding [8–10] and fine-grain scalable coding [11], [12] are among the important research topics in scalable audio coding. Wavelet transform provides good means of scalability because it gives "approximation" and "detail" coefficients which form a scalable representation of the signal. However, it has not found its way into standardized practical coders. Bit-plane scalable coding systems work based on grouping the encoded quantized coefficients in order of the bit importance (ranging form MSB to LSB) to form a scalable pattern. For example in bit-sliced arithmetic coding (BSAC) technique [10] the quantized spectral coefficients are grouped into slices of bit-planes and each of these bit slices are coded using arithmetic coding[3]. Fine-grain scalability techniques try to enhance the quality of the base signal as a series of small steps.

Scalable audio coding has been standardized by MPEG and there has been recent research to improve upon the standard. Scalable coding of stereophonic audio [13], compander domain approach to scalable coding [14], conditional enhancement layer quantization [15], cross-layer rate-distortion optimization [16] and joint optimization of the perceptual core and the enhancement layers [17, 18], are among these contributions.

In [13] predicting one channel from the other in both base layers and enhancement layers leads to higher coding efficiency. In [14] it is shown that the typical scalable coding system based on the reconstruction error quantization (REQ) is asymptotically optimal if the coding is performed in the companding domain (for the compressed signal). The proposed scheme in [15] exploits the statistical dependence of the enhancement-layer signal on the base-layer quantization parameters. An optimization technique is proposed in [16] for the advanced audio coding (AAC) framework wherein a cross-layer iterative optimization is performed to select the encoding parameters for each layer with a conscious accounting of rate and distortion costs in all layers, which allows for a trade-off between performances at different layers. An encoding process is proposed in [17] in which by setting special parameters a trade-off is provided between the perceptual quality of the AAC core and

[3]Arithmetic coding is one of the common variable-length coding methods. It is explained in detail in Appendix B.

the enhancement layers in MPEG-4 scalable AAC (S-AAC) and scalable-to-lossless coding (SLS).

There have been also other research activities in most of which the above main ideas have been adopted such as applying the compander domain approach to AAC [19], conditional enhancement-layer quantizer applied to AAC [20], and efficient scalable coding of stereophonic audio by conditional quantization where the ideas of stereophonic scalable coding and conditional scalable coding is applied to a single coder at the same time [21].

In this dissertation, we present some contributions to scalable coding schemes which improve upon the practical coders such as MPEG-4 audio. In the next section we will summarize them.

1.2 Summary of Contributions

Among the existing scalable coders there is one particular scalable coding system which is of practical interest and is the core of MPEG-4 scalable coding [22]. This coder is based on reconstruction error quantization (REQ). In such a scalable coding system, the audio signal is quantized by an optimal quantizer designed for a minimum average bitrate with acceptable distortion (base layer) and there are additional layers which enhance the quality of the base layer output signal by refining the quantization quality: The quantized signal is subtracted from the original signal to form the reconstruction error signal and this error signal (also referred to as the residual signal) again is quantized in the enhancement layer. At the receiver side, the output of enhancement layer can be added to the base layer output and hence an enhanced reconstruction of the original signal is achieved. This enhancement process can be repeated by having successive enhancement layers.

There is a recent alternative to the scalable audio coding in MPEG-4 audio. This coding system which has a different scheme is called Scalable-to-Lossless (SLS) coding. The system uses a specific bit-plane coding (BPC) called bit-plane Golomb code (BPGC) [8,23]. BPGC provides lower computational complexity compared to the REQ and can be considered as another option for scalable coding in MPEG-4 audio.

It has been shown that, there is a considerable performance gap between MPEG-4 scalable coding schemes and a non-scalable AAC coder operating at the same bitrate [24]. This sub-optimality results in part from the fact that in this system the layers are coded independently and the mutual information between them is not considered. This leads to

a scalability penalty which becomes bigger as the number of layers increases.

In this dissertation we provide three different contributions to the practical scalable audio coding systems mentioned above. Our first contribution is the scalable audio coding using watermarking[4]. In this method we propose using a technique, Quantization Index Modulation, borrowed from watermarking. Using this technique some of the information of each layer output is embedded (watermarked) into the previous layer. This approach leads to a saving in bitrate while keeping the distortion almost unchanged. This makes the scalable coding system more efficient in terms of Rate-Distortion when more than one layer is received most of the time. The results show that the proposed method outperforms the scalable audio coding based on REQ.

The next contribution is for the BPC-based scalable coding used in MPEG-4 audio. Considering the properties of the residual signal, core-based bit-plane probabilities are provided for MPEG-4 Audio Scalable to Lossless Coding (SLS), which matches the quantization and coding performed in the core layer. Using the same strategy, new probabilities are obtained to consider the clipping effect in bit-plane coding of an unbounded signal. Simulations show that considering the core layer parameters improves the bit-plane probabilities estimation.

Our last contribution which is perhaps the most important one is presented in Chapter 6, which is a very fine-grain scalable coding approach by designing a scalable entropy coding using a trellis-based optimization. The sub-optimality of a scalable coder compared to a non-scalable coder results in part from the independent coding of the layers in these systems. One of the aspects which plays a role in this sub-optimality is the entropy coding. In practical audio coding systems including MPEG Advanced Audio Coding (AAC), the transform domain coefficients are quantized using an entropy-constrained quantizer. In MPEG-4 Scalable AAC (S-AAC), the quantization and coding is performed separately at each layer. The entropy coding adds an overhead for each layer which is larger at lower quantization resolutions in the case of Huffman coding. This overhead which becomes larger as the number of layers increases, could result in a difference of tens of kbps in bitrate of the overall coder. In fact, there is a trade off between the overall overhead and fine-grain scalable coding in which more layers are required and the bitrate (hence the quantization resolution) per layer is small.

In the proposed scheme, a scalable coding for audio signals is provided where the entropy

[4]Watermarking refers to embedding additional information into a signal. We will discuss it in Chapter 4.

coding of a single quantizer is made scalable. By constructing a Huffman-like coding tree where the internal nodes can be mapped to the reconstruction points, the tree can be pruned at the internal nodes to control the rate-distortion (RD) performance of the encoder in a fine-grain manner. A set of metrics and a trellis-based approach is proposed to create a coding tree so that an appropriate path is generated on the RD plane. The results show the proposed method outperforms the scalable audio coding performed based on reconstruction error quantization as used in practical systems, e.g. in scalable advanced audio coding (S-AAC).

The above contributions have been reported in the following papers:

• M. Movassagh, P. Kabal, "Fine-Grain Scalable Audio Coding by Trellis-Based Optimized Scalable Entropy Coding", *Submitted to IEEE Trans. Audio, Speech and Language Processing.*

• M. Movassagh, P. Kabal, "New Bit-Plane Probability Calculations for Scalable to Lossless Audio Coding", *Proc. IEEE ICASSP* (Florence, Italy), pp. 3675-3679, May 2014 [25].

• M. Movassagh, P. Kabal, "Scalable Audio Coding Using Watermarking", *Proc. IEEE Int. Conf. Multimedia and Expo* (San Jose, CA), 5 pp., July 2013 [26].

• M. Movassagh, J. Thiemann, and P. Kabal, "Joint Entropy-Scalable Coding of Audio Signals", *Proc. IEEE ICASSP* (Kyoto, Japan), pp. 2691-2694, March 2012 [27].

1.3 Organization

This thesis is organized as follows. We will provide a background on audio coding tools, techniques and scalability in Chapter 2. A review of the common audio coding tools is given, optimal quantization issues is discussed, practical scalable audio coding systems are explained and at the end of the chapter, MPEG-4 audio coding and scalability is reviewed. Chapter 3 provides a brief introduction to psychoacoustic principles and reviews two important psychoacoustic models. The first contribution, scalable audio coding using watermarking is presented in Chapter 4. New bit-plane probability calculations for SLS coding system is provided in Chapter 5 and finally a fine-grain scalable audio coding scheme by trellis-based optimized entropy coding is provided in Chapter 6. Lastly, Chapter 7 concludes and summarizes the thesis.

Chapter 2

Background on Audio Coding and Scalability

The main goal in audio coding is to compress an audio signal by removing the redundant or/and irrelevant information from it. The redundant information refers to the information without which the original signal can be perfectly reconstructed after the coding process. An example of a process which creates redundant information could be sampling a signal in a rate higher than twice its bandwidth (the Nyquist rate). Also, an example of an important process for removing redundant information could be the entropy coding which we will discuss in detail in this chapter. The irrelevant information is unique information which is required for reconstructing the original signal waveform, however is not perceptible. For audio signals it is the information which is not audible. There are several common techniques which are used to remove these types of information from an audio signal which we will discuss briefly in the following sections.

In this chapter, first we will review briefly some of the common audio coding tools which are used in most practical encoders. The first section is followed by some discussion regarding the optimal quantization issues. The typical scalable audio coding systems are discussed in Section 2.3 and the last section covers scalable/audio coding in the state of the art MPEG-4 audio as one the most successful audio coding standards.

2.1 Review of Common Audio Coding Tools

2.1.1 Quantization

Quantization is a process for obtaining the digital or binary representation of a signal. It can be applied to the time-domain samples or the frequency coefficients[1] of a signal during the coding process. There are different types of quantizers with different properties. They are typically different in being memoryless or with memory, the distribution of the quantization intervals (uniform or non-uniform quantizers) and the dimensions of the quantization which gives a scalar or vector quantizer. In the following we will briefly discuss these different quantizers.

Uniform Quantization

In uniform quantization the range of the input is divided into equal-sized intervals where the number of these intervals is given as $M = 2^B$ where B is the number of bits required for the binary representation of the quantizer output (in fixed-length coding). All the input values within an interval are quantized to a reconstruction point. This point can be in the middle of the interval or not.

The main measure for evaluating the performance of a quantizer is the quantization error which is defined as the mean square error (MSE) between the input and output:

$$\text{MSE} = \int_x (x - Q(x))^2 f(x)\, dx \tag{2.1}$$

where x is the input, $Q(x)$ is its quantized value (output) and $f(x)$ is the PDF of the input. This could also be expressed in the form of

$$\text{MSE} = \sum_i p_i e_i, \tag{2.2}$$

where p_i is the probability of $x \in X_i$ and e_i is the conditional error variance in each interval

[1] In the rest of this dissertation we may refer to the time-domain samples and the frequency-domain coefficients briefly as samples or coefficients, respectively.

i:

$$e_i = E[e^2|X_i] = \int_{x \in X_i} (x - \hat{x}_i)^2 \frac{f(x)}{p_i} dx. \tag{2.3}$$

where $\hat{x}_i = Q(x)|_{x \in X_i}$.

In a specific case where the quantizer intervals are very small, the quantization error can be well modelled as an independent noise with uniform pdf and variance of $\sigma_e^2 = \frac{\Delta^2}{2}$ where $\Delta = \frac{2X_{max}}{M}$ (X_{max} and $-X_{max}$ being the quantizer limits). The SNR of the quantizer in dB then becomes

$$\text{SNR}_{\text{dB}} = 10 \log_{10}(\frac{\sigma_x^2}{\sigma_e^2}) = 10 \log_{10}(\frac{12\sigma_x^2 2^{2B}}{4X_{max}^2}) \tag{2.4}$$

Considering $X_{max} = 4\sigma_x$ which is a good practical choice (e.g. it leads to quantizing 99.6% of speech without clipping), we get

$$\text{SNR}_{\text{dB}} = 6B - 7.2 \tag{2.5}$$

For instance, to get a 60 dB SNR for speech (good speech), 11 bits are required. This leads to a bitrate of 88 kbps for 8 kHz sample rate. In general, the SNR of a uniform scalar quantizer (USQ) can be approximated as [28]

$$\text{SNR}_{\text{dB}} = 6.02B + \alpha \tag{2.6}$$

where the normal range of alpha is about $\alpha \in [-10, 10]$.

Non-Uniform Quantization

By looking at the MSE equation (2.1), it becomes intuitive that, if we make the quantization more precise in the regions where the signal is more probable (by decreasing the interval width) the average quantization error becomes smaller. For example, for the audio signals which can be well modelled by a Laplacian source, the probability is larger for smaller values. Therefore, by making the intervals smaller around zero the quantizer performance increases. This is done in a non-uniform quantizer. In fact, for a given quantization resolution (where the number of intervals is fixed), the optimal quantizer which gives the

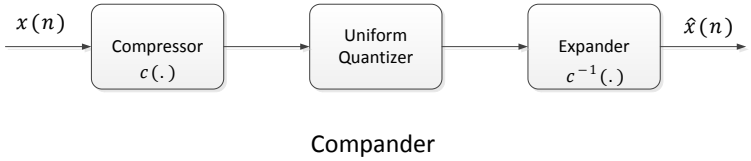


Fig. 2.1 Block diagram of a compander. At the encoder, the compressor compresses the input to get a signal with uniform distribution. The new signal is quantized by a uniform quantizer. At the decoder, the reverse function of compression i.e. expansion undoes the compression.

smallest MSE, is a non-uniform quantizer for which the probability in each region is the same [29]. This again implies that: the higher probability region, the smaller intervals.

In practice a non-uniform quantization is performed in two steps: 1) Non-linear scaling 2) Uniform quantization. The scaling process is performed by the so-called "Compander" (Fig. 2.1) which comes from the words "Compression" and "Expansion". At the encoder side, a compressor compresses the input to get a signal with uniform distribution. Then in the next step, the new signal is quantized by a uniform quantizer. At the decoder, the reverse function of compression (expansion function) is applied to the reconstructed signal to undo the compression. Two common companders for speech are the μ-Law and A-Law companders which are used in the United States and Europe respectively. As another example of companding for audio we can refer to the AAC compression function which is given by

$$y = c(x) = x^{\frac{3}{4}}. \tag{2.7}$$

This compression applies a very slight modification to the input signal. Figure 2.2 shows a plot of this function. In fact, this quantizer was basically designed to be a specific uniform quantizer called uniform threshold quantizer (UTQ). We will talk about UTQ in more detail in Section 2.2.

For audio in general, a Laplacian[2] well modelles the signal's sample statistics. The companding functions can be calculated from the pdf of the input for the case of resolution-constrained quantization and they are related by [30]

$$\frac{dc(x)}{dx} = \Delta\Lambda(x) \tag{2.8}$$

[2]The Laplace distribution is a two-sided exponential distribution.

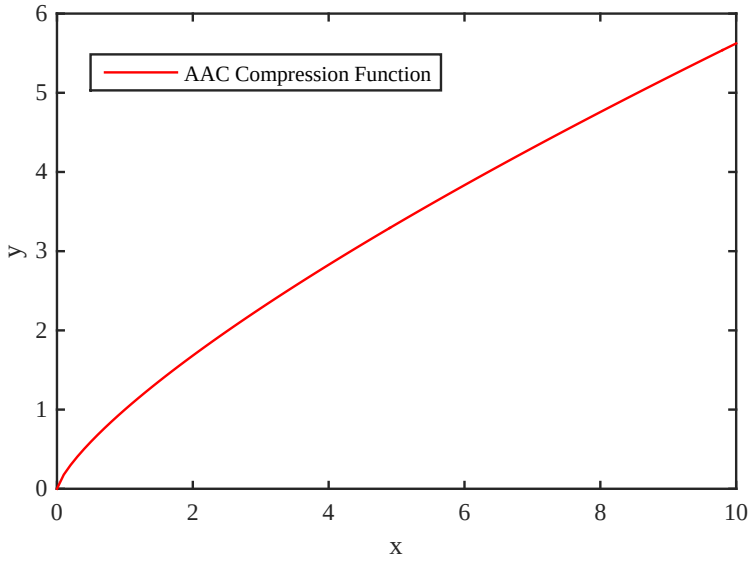


Fig. 2.2 AAC compression function. The function applies a slight modification to the input signal and is given by $y = c(x) = x^{\frac{3}{4}}$.

where $c(x)$ is the compression function, Δ is the step size of the new uniform quantizer and $\Lambda(x)$ is the level density function of a non-uniform quantizer which is obtained from its input pdf (see Section 2.2).

Adaptive Quantization

In practice an audio signal has a time-dependant pdf. Therefore, using even a non-uniform quantizer may lead to a large quantization error. One solution to this problem is using an adaptive quantizer. In such a quantizer the step sizes of the quantizer are adjusted adaptively depending on the local statistics.

Vector Quantization (VQ)

When we quantize a sequence of samples/coefficients, there is correlation between them . If they are quantized independently, this correlation is not considered. In a vector quantizer (VQ) a set of samples are quantized at the same time in a multi-dimensional quantizer. This way, the quantization performance improves considerably. In other words, for a given quantization error, using a vector quantizer decreases the number of bits required or vice versa. For an N_d-Dimensional VQ where there are N_v vectors in its codebook, the number

of bits per sample is (fixed-length coding)

$$B = \frac{1}{N_d} \log_2 N_v. \tag{2.9}$$

In practice, the quantizer vectors and the indices assigned to them are stored in a table called *codebook* and both the encoder and decoder have access to it. A Voronoi (nearest-neighbor) search of the codebook is performed at the encoder and the index of the vector is chosen from the codebook. The key issues in VQ are the design and search of the codebook and there are several design techniques for them [31–34]. For a given rate, the memory and the computational complexity grows exponentially with dimension N_d for an unstructured VQ.

2.1.2 Entropy Coding

The simplest way to obtain the signal's binary representation from the quantizer output, is to assign the same number of bits to all the intervals in the quantizer. This is how we obtain an B-bit representation for a quantizer with $N_I = 2^B$ intervals (fixed-length coding). However, this this may be inefficient. Shannon, in the mathematical theory of communication shows that, the minimum average number of bits required to encode a source (or a sequence of symbols), is the entropy of the source [35]. The entropy of source X is normally denoted by $H(X)$ and is defined as

$$H(X) = -\sum_i p_i \log_2 p_i, \tag{2.10}$$

where p_i is the probability of the i^{th} possible symbol that can be generated from the source. When a base-2 logarithm is used as above, the entropy is specified in bits. Here, in audio coding we always consider entropy in bits. It is clear from the above equation that entropy is never negative because $p_i \in (0, 1]$.

The entropy is also sometimes referred to as a *measure of uncertainty* because it shows the uncertainty of a random variable before observation. It can also be interpreted as the amount of information a source has which is directly related to its uncertainty. In fact, zero entropy indicates no uncertainty and high entropy shows strong uncertainty. As a specific case, it is good to show the entropy of a source with uniform distribution. Assuming the

source generates N_s different symbols with the same probability, from (2.10) we get

$$H(X) = -\sum_i \frac{1}{N_s} \log_2 \frac{1}{N_s} = \log_2 N_s. \tag{2.11}$$

This is the entropy of a uniformly distributed source which has the maximum possible entropy among all the sources with N_s symbols. This is because the pdf of this source does not give any priority to any symbol in terms of probability and hence it has the maximum uncertainty. The entropy of such a source increases with N_s.

The average codeword assigned to a source symbols in the general case of "variable-length coding" is obtained by

$$L_c = E[l] = \sum_i p_i l_i, \tag{2.12}$$

where l_i is the codeword length assigned to the i^{th} symbol. As mentioned above, the entropy provides a bound on the average code length of a source. Comparing (2.10) and (2.12), this statement can be also interpreted as: the best codeword length assigned to a symbol with probability p is $-\log_2 p$. The closer a code is to this bound, the better code it is. In general, it is the source-coding theorem which says [35, 36]:
The uniquely decodable code that minimizes the average codeword length, L_c, satisfies

$$H(X) \leq L_c < H(X) + 1. \tag{2.13}$$

Now see (2.11) again. When we we assign the same number of bits to a quantizer output ($B = log_2 N_I$), we are assuming that there is no information about the pdf of the source (uniform distribution with maximum entropy). That is why this is an inefficient way to obtain the binary representation.

In entropy coding, the coding algorithm takes into account the unequal probabilities of the source symbols. This means that, the goal in this coding is to generate codewords for which the average code length is as close to the entropy as possible. Two commonly used entropy coding algorithms in practical coding systems are the Huffman coding and the arithmetic coding. In the next subsections we we will discuss them briefly.

It is useful to mention here that there are some code properties that are usually required in the practical coding systems. For instance, the codes have to be uniquely decodable

(each symbol has to be assigned a unique code) and prefix-free[3]. In a code stream there is normally no boundaries between the successive codewords. Therefore, it is very important that the codes are prefix-free. This means that, no code should be a prefix of another code. The necessary condition for a code to be prefix-free is given by the Kraft inequality [36]:

$$K_I = \sum_i 2^{-l_i} l_i \leq 1, \tag{2.14}$$

where l_i is the codeword length of the i^{th} symbol.

Huffman Coding

Huffman coding is a common entropy coding technique used in the practical systems. It is known to give the shortest average codeword length for a set of finite-alphabet symbols with known probabilities [37]. It also generates an optimal prefix code (having the prefix property). It is widely used because of both its efficiency and simplicity.

The main idea in Huffman coding is to create a binary tree of the symbols. We start from the two source symbols which have the minimum probabilities. These symbols represented by some nodes (leaf nodes) are merged into a new symbol (new node) with a new probability which is sum of the old probabilities. One of these two branches is assigned 0 and the other one 1. This procedure is repeated until we get a full binary tree ending in the root node. The codeword assigned to each leaf node (original symbols) is the binary sequence from the root node to that leaf node. We will talk about Huffman coding in more detail in Chapter 6.

Arithmetic Coding

While Huffman coding is optimal, it has some drawbacks. Huffman coding performs best when the probabilities of the symbols are known in advance and the signal is stationary for a certain amount of time. In cases where the statistics of the signal changes by time, it is not easy to implement an adaptive Huffman coding. Arithmetic coding is another commonly used entropy coding which is more efficient for coding long sequences of source symbols [38, 39]. In Huffman coding, the encoding table grows exponentially with the sequence length, while in arithmetic coding the computational effort is linear with the

[3]The prefix-free property requires that no code can be a prefix of another code.

number of symbols in a sequence. Arithmetic coding has also the advantage of adaptivity as the symbol probabilities change.

In arithmetic coding the interval $[0, 1)$ is partitioned into cells. Each cell is associated with a source symbol and the size of the cell is proportional to the probability of the symbol. This partitioning is repeated for each cell according to the sequence of symbols. In the end, the truncated binary representation of the last cell's midpoint is considered the codeword for the whole sequence. We talk about arithmetic coding in more detail in Appendix B and we will use it in Chapter 5.

Run-Length Coding

Run-Length coding is useful when there are long sequences of identical symbols. Instead of sending all these symbols, the number of identical symbols is sent. Usually, if this coding (if used) acts as the first step of lossless coding. Then, the resulting stream of numbers is coded using either Huffman or arithmetic coding.

Rate-distortion Theory

The entropy coding gives the most efficient i.e. the minimum average bitrate for a given source with given distortion. However, in source coding we are interested in both bitrate and the distortion. For instance in audio coding, we need to know what the audio quality is for a specific bitrate. While the entropy coding is a lossless process, the preceding stages in a practical coding system are usually lossy, e.g. the quantization.

The rate-distortion theory is about the trade-off between the distortion and the bitrate in source coding by providing some bounds. It specifies the lowest rate possible for a given distortion or the lowest distortion for a given rate. These bounds are expressed by the rate-distortion function which is defined as

$$R(D) = \min_i R_i(D) \tag{2.15}$$

where D is the distortion, between the reconstructed and original signal, and the minimization is over all coding possibilities i. As an example, we can refer to the rate-distortion function for a Gaussian density source. If the distortion is defined as the mean squared

error (MSE), the rate-distortion function becomes [40]

$$R(D) = \frac{1}{2}\log(\frac{\sigma_x^2}{D}) \tag{2.16}$$

Here, we are interested in a rate-distortion function for quantization and entropy coding. Since the entropy coding is performed after quantization, we would like to know what the minimum quantization distortion is that we can get for a given quantization output entropy. In other words, we would like to know the best quality we can get for a given bitrate. This leads to a specific quantization called entropy-constrained quantization. In Section 2.2 we will discuss this type of quantization as opposed to the resolution-constrained quantization. In the first one, the number of quantization levels is not fixed since we just care about the entropy. The resolution-based quantization is the case where we want to minimize the quantization distortion for a given number of quantization intervals which is also referred to as the quantization resolution.

Perceptual Entropy

As explained above, in information theory the entropy gives the minimum possible bitrate to code a source with the ability of reconstructing it without any distortion. The distortion considered here is normally defined as the MSE between the original and the reconstructed signal. In perceptual coding of audio, another specific entropy is defined and used which is called perceptual entropy. In this type of entropy, instead of MSE, the audibility of the resulting distortion is considered. The perceptual entropy is defined as [41]

$$PE = \frac{1}{N_f}\sum_{f_l}^{f_h}\max(0, \log_2\frac{|S(f)|}{M(f)}) \tag{2.17}$$

where f_l is the lowest frequency in the desired frequency range (e.g. a band), f_h is the highest frequency in that range, S(f) is the magnitude of the frequency component f, and $M(f)$ is the masking threshold at f. Although there is not a complete theory about the perceptual entropy yet, (2.17) is useful as a perceptual entropy measure.

2.1.3 Prediction

There is correlation between audio signal samples/coefficients in both time and frequency domain. This means we can have a good estimate of the current sample by predicting it from the previous samples. By subtracting the estimated value from the original value we get a signal referred to as the prediction error which has a much smaller dynamic range compared to the original signal. This means that, for a given distortion, we can encode the residual signal with fewer bits compared to encoding the original signal.

Note that, prediction in the time domain and coding in the frequency domain are directly related. When there is correlation between the signal samples in the time domain, we get a colored power spectrum of the signal in the frequency domain. Whereas, if there is no correlation in the time domain, the spectrum will be flat or white. There is no advantage in coding a white-spectrum signal in the frequency domain. However, coding in the frequency domain has some other advantages. For instance, the perceptual coding can be performed using the frequency masking thresholds given by a psychoacoustic model. We will talk about the frequency-domain coding later.

The most commonly used prediction in speech and audio coding is the linear prediction (LP) where a linear combination of the previous samples gives the predicted signal. The residual signal in the z-transform domain can be expressed as

$$A(z) = 1 - \sum_{i}^{L} a_i z^{-i} \tag{2.18}$$

where $A(z)$ is called the LP analysis filter and L is the order of the predictor. The LP coefficients are calculated by using a correlation matrix which is formed based on the statistics of the signal. Two common techniques for obtaining the LP coefficients are the so called "autocorrelation method" and the "covariance method'. For fast processing, the iterative algorithms like the Levinson algorithm are used in practice.

The linear predictive (LP) coding is specially very popular for speech coding. In the model-based speech coding where there is excitation and vocal tract modeling, LP can be used to well model the vocal tract. By passing the speech signal through the LP analysis filter we get the excitation signal. The LP coefficients and the excitation can be coded and sent separately. This leads to a great amount of compression.

There is short-term prediction and long-term prediction. The short-term prediction is

basically the one described above and is performed over blocks of samples. In long-term prediction, the main goal is to capture the periodicity in an audio signal. In this type of prediction there are normally two parameters: the delay and the gain parameter. The analysis filter of long-term prediction can be expressed as

$$A_L(z) = 1 - a_\tau z^{-\tau} \tag{2.19}$$

where τ and a_τ are the delay and the gain parameters respectively.

In the following we will briefly refer to two important and common coding systems with the prediction core.

Differential PCM (DPCM)

As mentioned before there is a great amount of correlation between an audio signal samples. For example for speech the normalized correlation coefficient is usually in the range of 0.8 to 0.9. This implies that, range of values for the difference between two successive samples is much smaller than that of each individual sample. Therefore, if we quantize the difference values instead of the individual values, we can save a lot in the bitrate. In differential PCM, the difference between a current sample and a predicted estimate of the sample is quantized. The prediction is normally based on the weighted average of the previous samples i.e. a linear prediction. The prediction makes the range of quantizer input even smaller compared to DPCM (which can also be considered as a one-step prediction). This is because the estimated values are very close to the actual values and hence their difference has a very small range, unless the prediction fails (impulsive error).

In DPCM there is a local decoder inside the encoder that generates the estimated signal $\hat{x}(n)$. This is because the decoder has only access to the quantized error $e_q(n)$ and not the actual error $e(n)$. The prediction is performed on the previous estimated signal samples and the predicted value is subtracted from the original signal $x(n)$ to form the error signal which is the quantizer input. At the decoder a similar prediction process is performed to give the estimated signal.

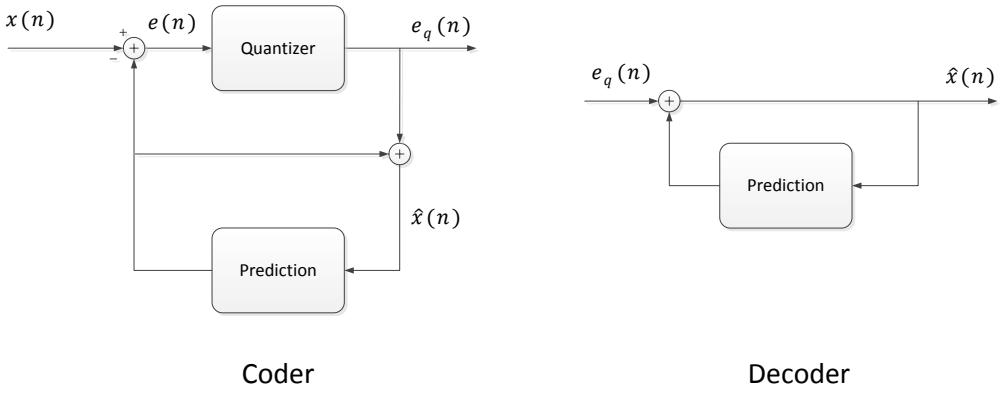


Fig. 2.3 Block diagram of DPCM codec. The prediction is performed on the previous estimated signal samples $\hat{x}(n)$. There is a local decoder inside the encoder which gives the estimated signal using the quantized error $e_q(n)$.

Adaptive Differential PCM (Adaptive DPCM)

We can take advantage of the adaptive coding again in a DPCM. The adaptiveness can be for both the quantization and the prediction parts. In an adaptive DPCM, typically there is a "scale factor generator" (for adaptive quantization) and an "adaptive predictor".

2.1.4 Sinusoidal Coding

Sinusoidal coding is a parametric audio coding where the audio signal is modeled by a sum of sinusoids called "partials" with time-varying amplitude, frequency and phase[4]. This model is expressed as [42, 43]

$$x(n) \approx \sum_{k=1}^{K} A_k(n) \cos(\omega_k(n)n + \phi_k(n)), \tag{2.20}$$

where $A_k(n)$, $\omega_k(n)$ and $\phi_k(n)$ are the magnitude, frequency and phase of the k^{th} sinusoid at time n. These parameters are typically obtained by either the peak picking technique in the short time Fourier transform domain (STFT) [42, 44] or by MSE minimization of the parameters using the analysis-by-synthesis technique [45–47].

Sinusoidal coding provides high quality audio coding at low rates. and has been incorpo-

[4]Sinusoidal coding is presented here briefly as part of a general review of the common audio coding schemes. It is not dealt with in this dissertation.

rated as part of the MPEG-4 parametric audio coding toolset. One of the other important features provided by sinusoidal coding is that the decoder will be able to perform the pitch and time-scale modifications by changing the parameters. For example, by changing the pitch of a speech or vocal signal, a low-pitch (typically male) voice can be changed to a high-pitch (typically female) voice or vice versa. Or by time-scale modifications, the speed and duration of a musical/audio signal is changed without changing its pitch properties.

In MPEG-4 audio, sinusoidal coding is performed using the analysis/synthesis audio codec (ASAC) which was proposed by Edler *et al.* [22]. ASAC uses an iterative analysis-by-synthesis algorithm for the estimation of the k^{th} partial. Then at each iteration, the most perceptually significant sinusoid remaining in the synthesis residual is identified, which is the component with the most unmasked energy. Therefore, the decoder will be able to reconstruct the signal in a perceptual scalable manner. Although other than its common uses, sinusoidal coding provides scalability as well, it has not been used as a scalable coder in practice.

ASAC provides improved quality for coding nonharmonic tonal signals, but not for pure speech compared to the standard speech coders [22, 48]. There is another sinusoidal codec called which improved the performance of ASAC in some aspects. This coder is called "harmonic and individual lines plus noise" (HILN) which is similar to the ASAC but with some modifications. HILN has specifically improved performance for harmonic audio signals. Also, it has been popular in low-rate internet audio streaming. HILN was accepted in MPEG-4 audio as a low rate parametric audio coder [49].

2.1.5 Filter Banks and Subband Coding

Filter banks are a set of filters which decompose a signal into several frequency bands. Since each filter output has a reduced bandwidth compared to the original signal, down-sampling is applied to each output. When this is done for all the filters, it is called "critical downsampling" where the total number of samples coming out of the filter bank is equal to the number of samples going into it.

A filter bank can be categorized into a low frequency resolution or a high frequency resolution type. In the first type the filter outputs are used directly. For example 32 filters decomposing the signal into 32 frequency bands (as used in MP3). In the high frequency resolution type, a frequency transform is applied to each filter output or to the original

signal directly. For example, in MP3 audio coding, MDCT transform is applied to the filter bank (for an audio frame of 1024 samples) and 512 frequency lines (coefficients) are obtained. The coders which use the first type of filter banks are usually called "sub-band coders" [50, 51] whereas the coders using the second type are called "transform coders" which we will talk about in Section 2.1.6.

One important property for a filter bank is the so called "perfect reconstruction". This means that, after the signal is decomposed into the frequency bands using the "analysis filters", the original signal can be perfectly reconstructed using the "synthesis filters". This is when there is no other sources of coding noise (e.g. quantization noise). This is important since this way analysis-synthesis process itself does not add distortion during the coding process. Almost all the filter banks used in the practical systems have this property.

The analysis/synthesis filter bank has a baseband impulse response which is multiplied by sinusoids with different frequencies to form the impulse response of the other filters. The convolution process in practice consists of buffering and multiplication by this impulse response which is equivalent to "windowing". That is why the baseband impulse response is also referred to as the "prototype window".

The filter bank can also be designed in a way that the different frequency bands have different bandwidths. In audio coding, usually the higher bandwidth is assigned to the higher frequency bands. This is to match the critical band structure of the human auditory system (HAS) which is more sensitive to the lower frequency components.

QMF Filter banks

One of the common filter banks used in audio coding is the one based on the quadrature mirror filters (QMF) which are part of many coding systems [52,53]. These filters are often used in a tree structure and the filtering process is performed in binary stages. At each stage, the signal can be decomposed into a low frequency and a high frequency band. By repeating this procedure the signal is decomposed exponentially in the frequency domain (see Fig. 2.4). The QMF banks can also be realized as a polyphase factorization [54].

The QMF filter banks have some drawbacks. First of all, a perfect reconstruction (PR) is not possible for them, although with anti-aliasing techniques the resulting noise can be decreased to some extent. The aliasing effect can be canceled by designing the analysis and

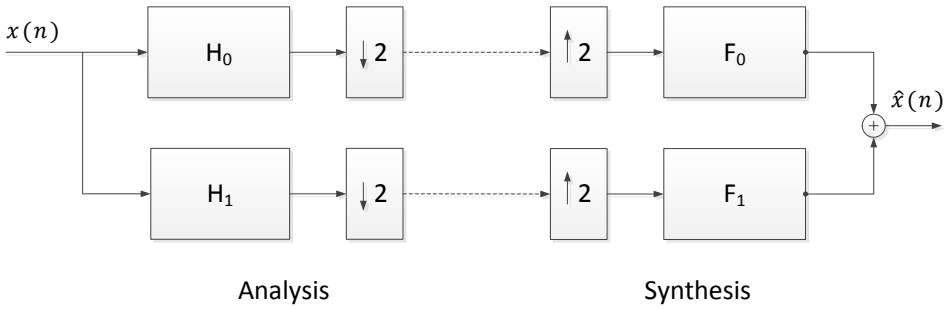


Fig. 2.4 A two-band QMF filterbank. The analysis and synthesis filters have the mirror property: $F_0(z) = H_1(-z)$ and $F_1(z) = -H_0(-z)$.

synthesis filters that have the "mirror" properties:

$$\begin{aligned} F_0(z) &= H_1(-z) \\ F_1(z) &= -H_0(-z) \end{aligned} \tag{2.21}$$

In addition to the PR problem, the QMF filter banks have also a large delay and high computational complexity e.g. compared to DFT.

Polyphase Filter Banks

Polyphase filter banks for audio coding have been introduced in [55]. These filter banks combine the filter design flexibility of the generalized QMF filters (GQMF) with low computational complexity. In MP3 standard this filter bank is used. Using this polyphase filter bank, the signal is decomposed into equal frequency bands. As mentioned before, in MP3 a prototype window is designed which is the core of the filter banks and is used in all the analysis/synthesis filters. These filters have relatively low computational complexity. The main drawback for them is that the signal can only be decomposed into equal frequency bands.

Hybrid Filter Banks

When two or more different filter banks are used together to increase the performance of the system, they are referred to as the hybrid filter banks [56]. For example in MP3, after the polyphase filters, MDCT is applied to each filter output to increase the frequency

resolution. The quantization and coding is then applied to the MDCT coefficients. This way, different frequency resolution can be obtained from different frequency bands.

2.1.6 Transform Coding

In audio coding based on transform coding, one of the unitary transforms e.g. FFT, DFT, DCT is used for time/frequency analysis of the audio signal. Applying the transforms to a frame/block of the signal, gives a set of frequency coefficients. These coefficients are then used in the other stages of the coder, mainly in the quantization/coding and the psychoacoustic model. Usually, the transform coders perform higher-resolution frequency analysis of the signals compared to the subband coders in which the signal is decomposed into coarse frequency subbands. Although, there are overlaps between these two common coding categories. Many transform coding schemes have been proposed so far and some of them have been standards for a long time including the MPEG standards.

MDCT

One of the most common transforms used in the current practical audio coding systems is the so called Modified Discrete Cosine Transform (MDCT). The MDCT is a lapped orthogonal block transform which can be considered as a cosine-modulated filter bank [57, 58]. It has several important properties that made it very popular. MDCT has the perfect reconstruction (PR), linear phase response, critical sampling, low complexity and several other properties [59, 60]. Also, a single FIR filter can be used in MDCT for both analysis and synthesis. One of the other important simplicity features for MDCT is that, it gives real coefficients (as opposed to the complex numbers given by DFT).

The MDCT is defined as [61]

$$X(k) = \sum_{n=0}^{2N-1} x(n) h_k(n) \tag{2.22}$$

where N is the block size and $h_k(n)$ is the MDCT analysis/synthesis filter impulse response given by

$$h_k(n) = \sqrt{\frac{2}{N}} \cos[\frac{\pi(2n + N + 1)(2k + 1)}{4N}]. \tag{2.23}$$

Note that the the number of coefficients obtained (N) is half the block size ($2N$). The block

advance however, is N samples. This means, there is 50% overlap between the successive blocks. The inverse MDCT is performed simply by applying the same analysis function to the transform coefficients:

$$x(n) = \sum_{n=0}^{N-1} X(k)h_k(n). \tag{2.24}$$

When performing the inverse transform, N spectral coefficients give $2N$ time-domain samples. These samples are overlapped and added with the $2N$ samples obtained for the previous block.

Typically a window function is applied to the block of signal before taking the transform. Therefore, (2.22) cane be written as

$$X(k) = \sum_{n=0}^{2N-1} x(n)w(n)h_k(n), \tag{2.25}$$

where $w(n)$ is the windowing function[5]. Several windowing functions have been proposed and used for the block transforms including the MDCT. In general, for the orthogonal transforms and to meet the time domain aliasing cancelation (TDAC) in the lapped transforms, the two following conditions must be met:

$$w(2N-1-n) = w(n), \tag{2.26}$$

and

$$w(n)^2 + w(n+N)^2 = 1. \tag{2.27}$$

One commonly used windows in audio coding is the "sine window". The sine window for a block of N samples is defined as

$$w(n) = \sin[(n+\frac{1}{2})\frac{\pi}{2M}] \tag{2.28}$$

where M is the window size. The sine window has several important properties specifically for MDCT. Using the sine window the DC energy is concentrated in a single transform

[5]In the practical coding systems the signal is processed block by block. This finite-block processing corresponds to windowing a signal (a rectangular window in its simplest form) before applying the block transforms such as DFT and DCT . Since the windowing affects the spectral properties of the signal, windows with different properties are used in practice for different purposes.

coefficient and the filter bank achieves 24 dB side lobe attenuation. Also, the sine window makes MDCT asymptotically optimal in terms of coding gain for a lapped transform [57].

There is also another window which is not as commonly used as the sine window, but is used in some important coders e.g. in AAC. The window is called the Kaiser-Bessel Derived (KBD) window which was originally proposed by Dolby Laboratories. The window is defined as

$$w(n) = \sqrt{\frac{\sum_{j=0}^{n} v_j}{\sum_{j=0}^{M} v_j}} \tag{2.29}$$

where $v(n)$ is the Kaiser-Bessel kernel. The KBD window trades off a better stop-band attenuation (more than 40dB) for a wider pass-band compared to the sine window.

Integer MDCT

Integer MDCT (IntMDCT) provides an invertible integer-to-integer implementation of MDCT [62, 63]. This means, given integer input samples, the IntMDCT coefficients will have integer values as well. From these integer-valued coefficients, the original signal can be reconstructed without any loss. IntMDCT provides an approximation of MDCT with a high accuracy [64]. Figure 2.5 shows the difference between MDCT and IntMDCT spectrum values in dB (as Noise-to-Signal ratio NSR). The upper two curves are for the MDCT and IntMDCT (and overlap) and the lower plot shows the difference. As can be seen, the difference appears as a noise floor (around -120dB) which is much lower than the error resulting from perceptual coding [64]. Therefore, IntMDCT can be efficiently used in transform coding.

The MDCT can be performed by using DCT-IV, which is applied to a N-sample block instead of $2N$ [62]. This is done by first applying the so called time domain aliasing (TDA) to the original block in the time domain, which gives N new samples. The DCT-IV is then applied to get the MDCT coefficients. For the symmetric windows meeting the conditions in (2.26) and (2.27), the TDA matrix can be decomposed into "Givens rotations". The Givens rotations themselves can be decomposed into the "lifting" matrices. Using rounding in the lifting matrices provides an integer-to-integer implementation which is fully invertible.

Since IntMDCT is a lossless transform process, it is specially very useful in lossless and scalable-to-lossless audio coding (SLS) which we will discuss in Chapter 5. We will also

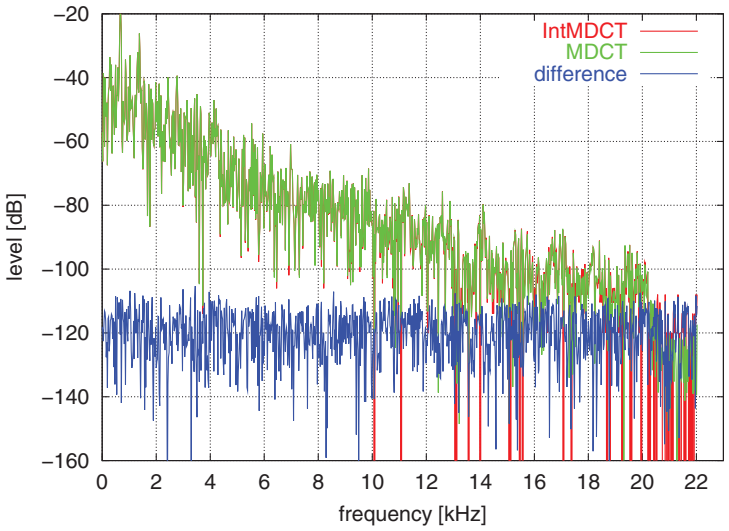


Fig. 2.5 Comparing MDCT and IntMDCT (from [64]). The upper two curves are for the MDCT and IntMDCT (and overlap) and the lower plot shows the difference NSR all in dB. The noise floor is much lower than the error resulting from perceptual coding.

talk about IntMDCT in more detail in that chapter.

Window/Block Switching

In transform coding including the schemes using MDCT and IntMDCT, different block sizes are used to meet the tradeoff between the time and frequency resolutions. For instance, in AAC there are the 'Long' and 'Short' blocks/windows and there are two transition windows between these two which are called the 'Start' and the 'Stop' windows. The start window is for transition from a long block to a short block and the stop window is for the opposite case. Figure 2.6 shows the typical block switching in AAC with the transition from long to short blocks and then back to the long blocks using these four types of windows [65].

Other than the block switching, there is also window switching where you can switch from one window type another one. In AAC there is window switching between the sine and the KBD windows. Consequently, there are two transition windows for this switching.

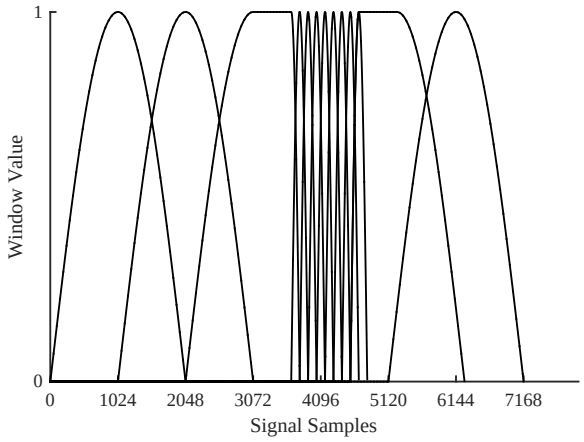


Fig. 2.6 Block switching in AAC. Long, Start, Short and Stop windows are respectively used in this figure.

2.1.7 Perceptual Audio Coding

Most practical audio coding systems use perceptual coding which gives much higher compression ratio for a given audio quality, compared to a non-perceptual coder. This is because in these systems, coding is performed based on the properties of the human auditory system (HAS). Such coders attempt to remove as much perceptually irrelevant information as possible for a given bit budget. In fact, removing parts of the signal which is not audible may introduce high distortion in terms of MSE, but it does not change its perceptual quality.

Figure 2.7 shows the simplified block diagram of a typical perceptual audio codec. In such a system's encoder, the signal is decomposed into several frequency bands. This decomposition is normally based on the critical band structure of the HAS and is performed using a filter bank including transforms. For instance in AAC when the sample rate is 48 kHz, the signal is decomposed into 49 non-overlapped subbands known as scale factor bands. The subband structure is such that, narrower frequency bands are assigned to the lower-frequency critical bands and vice versa.

For each frequency band, a masking threshold is calculated using a perceptual model which is most often referred to as the psychoacoustic model. The psychoacoustic model is the most important part of such a coder that makes it different from a non-perceptual coder. The masking thresholds determine the maximum noise energy introduced by quantization which will not be audible. Based on these limits on the quantization noise, it is decided

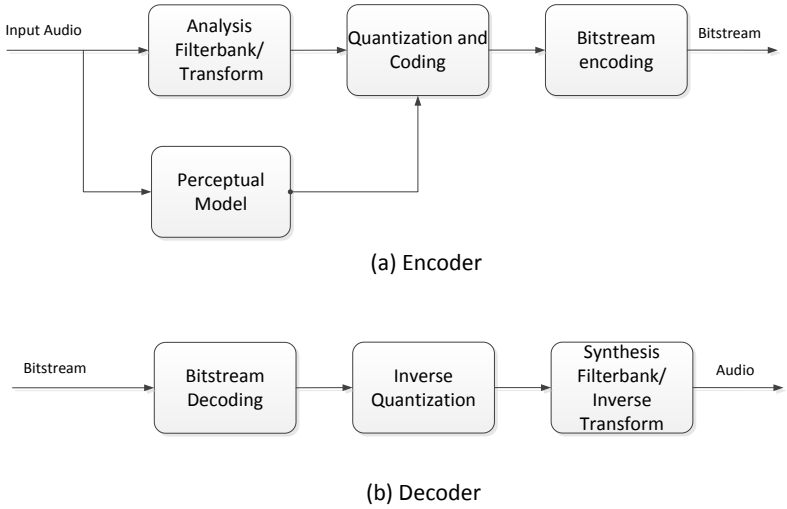


Fig. 2.7 Simplified block diagram of a typical perceptual codec. Coding is performed based on the properties of the HAS using a perceptual model normally referred to as the psychoacoustic model. The decoder does not need the psychoacoustic model for decoding the signal.

that how many bits can be assigned to each frequency band. The fewer the number of bits used, the higher the quantization noise and vice versa. Then based on the available bits, each frequency band is quantized and coded, normally using entropy coding. The coded subbands then are put in a frame or packet and sent/stored.

The decoder on the other hand, does not need to have access to the psychoacoustic model. It just receives the bit stream and decodes it to regenerate the quantized frequency components. Then frequency-to-time domain process is applied to get the reconstructed signal. If the coding is performed based on the subband coding, the filter outputs are just added together to give the input signal, as done in MPEG-1 layers 1 and 2. Current state of the art perceptual coders use transform coding. AAC for example uses the MDCT to get the frequency coefficients. These coefficients are then grouped into the 49 subbands as mentioned above.

2.1.8 Commonly-Used Audio Coding Standards and Formats

As mentioned before, audio coding techniques can be categorized into three major classes: digitized audio, lossless coding and lossy coding. For each of these major categories there have been several standards and formats. In the following, some of the most common audio

standards and formats will be discussed.

Digitized Audio

Even if no compression is required, the original audio needs to be coded to be sent or stored. The most common way to encode the original audio into the digitized audio is by using the pulse code modulation (PCM). There is also linear pulse code modulation (LPCM) where only a uniform quantizer is used, while in PCM in general, non-uniform quantization (e.g. using the $A-Law$ and $\mu-Law$ companders) can also be used. Audio files in '.wav', '.au' and '.aiff' formats are coded using PCM or LPCM.

Lossless Coding

As discussed before, in lossless coding the audio signal is compressed without loosing any information. This means that the original digitized audio can be perfectly reconstructed. Some of the most popular lossless coding formats are the "Free Lossless Audio Codec" (FLAC), "Apple Lossless Audio Codec" (ALAC) and the Monkey's Audio. FLAC and Monkey's audio file formats are ".flac" and ".ape". ALAC is stored in MP4 (MPEG-4) container with the file format of ".m4a". Although with lossless coding there is the advantage of reconstructing the original audio without any loss, its compression ratio is much smaller than for lossy compression techniques. For example a Monkey's Audio file is about 35 times as large as a relatively high quality 192 kbit/s bitrate MP3 file.

Lossy Coding

Lossy compression of audio is performed using perceptual coding considering the properties of the HAS, providing a much higher compression ratio compared to the lossless coding techniques. A variety of techniques have been proposed and used sofar. Some of the most popular lossy coding formats are the ".mp3" (MPEG-1 layer 3), ".aac" (MPEG-2/4 AAC), ".3gp" from the Third Generation Partnership Project (3GPP) and '.wma" (Windows Media Audio). State of the art MPEG-4 AAC is one the most successful audio coding standards available today. We will talk about this standard in more detail in Section 2.4.

2.2 Notes on Optimal Quantization

2.2.1 Resolution-Constrained Quantization

Constrained Resolution is a special case of quantization where we limit number of quantization levels to a specific number (say N) and find the best level density or reconstruction point density function $\Lambda(x)$ that minimizes the distortion. For this case it is shown that in the high-rate case the minimum MSE distortion is achieved when the reconstruction point density function is [66, 67]:

$$\Lambda(x) = N \frac{f_X(x)^{\frac{1}{3}}}{\int_{\Re} f_X(x)^{\frac{1}{3}} \mathrm{d}x} \tag{2.30}$$

And if we consider $R = \log_2(N)$ as the rate then the rate-distortion function is given by:

$$R = \frac{1}{2}\log_2(12D) + \frac{3}{2}\log_2(\int_{\Re} f_X(x)^{\frac{1}{3}} \mathrm{d}x) \tag{2.31}$$

In a special case where the input signal has a Gaussian pdf with a variance equal to unity the rate-distortion function becomes:

$$R = \frac{1}{2}\log_2(\frac{\sigma^2}{D}) + \frac{1}{2}\log_2(\frac{2\pi 3^{\frac{3}{2}}}{12}) \tag{2.32}$$

which is $\frac{1}{2}\log_2(\frac{2\pi 3^{\frac{3}{2}}}{12}) \approx 0.72$ bits above the rate-distortion function given by Shannon (see 2.16).

2.2.2 Entropy-Constrained Quantization

In most audio coding systems entropy coding is performed after quantization. Entropy coding tries to encode the quantization outputs based on the entropy of the output. The resulting average bitrate is bounded by that entropy, which means a practical coding system can not achieve a bitrate lower than the entropy of the output signal. So it is obvious that the desired quantizer in this case is what gives the minimum output entropy for a given distortion. In other words, the optimization criterion becomes finding level density function that minimizes the distortion for a given entropy constraint [68]. In high-rate theory, this

optimization leads to the following function for reconstruction point density [68]:

$$\Lambda(x) = 2^{R-h(X)} \tag{2.33}$$

which means for entropy-constrained quantization the optimal quantizer is a uniform quantizer (the reconstruction point density is constant) and the constant step size is:

$$\Delta = \frac{1}{\Lambda(x)} = 2^{-R+h(X)} \tag{2.34}$$

The rate-distortion function for this quantizer is given by:

$$R = h(X) - \frac{1}{2}\log_2(12D) \tag{2.35}$$

And for a Gaussian input signal with the variance equal to the unity it becomes:

$$R = \frac{1}{2}\log_2(\frac{1}{D}) + \frac{1}{2}\log_2(\frac{2\pi e}{12}) \tag{2.36}$$

Which is $\frac{1}{2}\log_2(\frac{2\pi e}{12}) \approx 0.25$ bits above the rate-distortion function given by Shannon and has a rate advantage of about 0.5 bit compared to the resolution-constrained quantizer.

2.2.3 Uniform Threshold Quantizer (UTQ)

Although it has been shown (mentioned in the previous section) that in high-rate theory for the entropy-constrained quantization the optimal quantizer has a uniform distribution, the reconstruction points within the intervals are shown not to be in the middle for the optimal quantizer. Also there is a "dead zone" in the quantizer which means all the intervals in the quantizer have the same width except the one including zero (the results are for a symmetric zero-mean input). These results have been proved for exponential signals [69] specially Laplacian signals which are good estimations for audio signal distributions. In such a quantizer which is called Uniform-Threshold Quantizer with Dead-Zone or briefly Uniform-Threshold Quantizer (UTQ), the reconstruction points are not in the middle of the intervals and their positions are shown by an offset parameter which is the distance between the reconstruction point and the beginning threshold of the intervals. The offset is the same for all the intervals and for a zero mean Laplacian signal with scale parameter

equal to unity it is obtained by:

$$\alpha = 1 - \frac{\Delta e^{-\Delta}}{1 - e^{-\Delta}} \tag{2.37}$$

where Δ is the quantizer step size of the quantizer (except for the dead-zone area) which in turn is obtained from some other calculations [69]. For the dead-zone the reconstruction point is at zero.

2.3 Scalable Audio Coding

In general, the scalable audio coding schemes that are used in the practical coders can be divided into two main categories. The first one, which is the most common scalable coding technique is a multi-layer coding where the audio signal is coded in different layers. In this scheme, each layer refines the coding error resulted from the preceding layers. This is performed by re-quantizing the quantization reconstruction error from the previous layer. This technique is also referred to as the reconstruction error quantization (REQ).The second scalable coding scheme is called "bit-plane coding" (BPC). In the following we will discuss these two techniques.

2.3.1 Scalable Coding Based on Reconstruction Error Quantization (REQ)

Introduction to REQ Scalable Coding System

Figure 2.8 shows the block diagram of a multi-layer salable audio coder based on REQ. In the first layer of such a scalable coding system, the audio signal is quantized by a quantizer set to give an acceptable distortion. The distortion should be tolerable by average listeners. The quantizer is normally designed based on the entropy-constrained rate-distortion optimization. This part of scalable coder is called the base layer because it produces the core bit stream, a basic representation of the original signal with an acceptable amount of distortion.

On the other hand there are enhancement layers which enhance the quality of the base layer output signal by refining the quantization quality as follows. In the next step, the quantized signal is subtracted from the original signal to form the reconstruction error signal. This signal is quantized in another layer called enhancement layer. The output in this layer gives the enhancement information, since by adding this quantized error signal

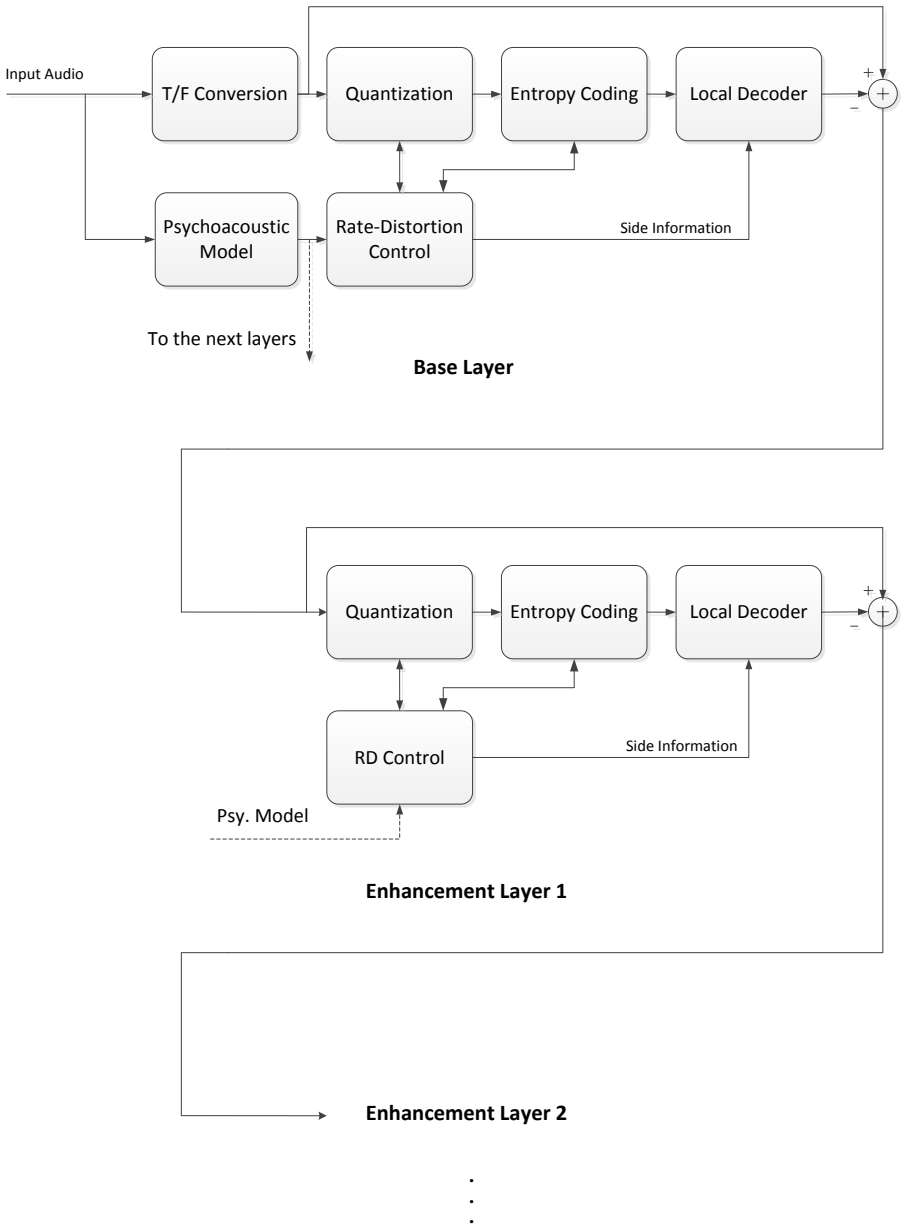


Fig. 2.8 Block diagram of scalable audio coding based on REQ. The base layer gives a minimum bit rate with acceptable distortion. The coded audio is then subtracted from the original input using a local decoder. The residual signal is passed to the next layer where it is encoded again and forms the first enhancement layer. This enhancement procedure is repeated to as many as layers required.

to the base layer output we can have an enhanced reconstruction for the original signal. Still we do not get perfect reconstruction but we can repeat the enhancement process by using successive enhancement layers, each of which quantizing the error signal, i.e. the difference between the overall reconstructed signal resulting from all the previous layers and the original signal. The larger the number of enhancement layers we have, the better the reconstructed signal.

Optimality Issues for REQ Scalable Coding

Consider again the simplified block diagram of the REQ system in Fig. 2.9. It has been shown that in general scalable audio coding systems (including REQ based coders) which minimize a non-MSE measure underperform compared to an equivalent non-scalable coder in terms of rate-distortion performance. It is also possible that for the audio signals a weighted mean square error (WMSE[6]) is used as the distortion metric for encoder optimization:

$$D = \int_x (x - Q(x))^2 w(x) f(x)\, \mathrm{d}x \tag{2.38}$$

where $w(x)$ is the weighting function.

For the special case of MSE, where there is no weighting, REQ based scalable coding

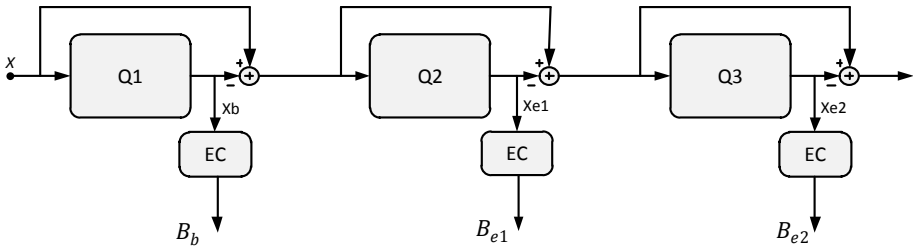


Fig. 2.9 Simplified block diagram of salable audio coding based on REQ.

achieves the operational RD bound of an optimal entropy-constrained non-scalable quantizer. In fact REQ is optimal when the optimal quantizers for both the base layer and subsequent enhancement layers are uniform [30]. In general for an entropy-constrained

[6]One typical WMSE example for audio signals is the Noise-to-Mask ratio (NMR = $\frac{N}{M}$) where $\frac{1}{M}$ can be considered as the weight.

quantization, where entropy coding is performed after quantization, the optimal high resolution scalar quantization (SQ) is given by the following formula in terms of the weight function [70]:

$$\Lambda(x) = \frac{\sqrt{w(x)}}{\Delta} \tag{2.39}$$

where $\Lambda(x)$ is the level density function or reconstruction point density function of the quantizer and Δ is obtained from the following equation:

$$\log(\Delta) = h(X) - R + \frac{1}{2}E[\log(w(x))] \tag{2.40}$$

and the Rate-Distortion function for this quantizer is given by [70]:

$$D(R) = \frac{1}{12}2^{2(h(X)-R)+E[\log(w(x))]} \tag{2.41}$$

Given an optimal base layer, the high resolution RD performance of a scalable coder employing REQ is strictly worse than the above operational distortion-rate function of a non-scalable coder [30]. However, when our measure is MSE or equivalently the weighting function equals to unity, the optimal quantizer becomes a uniform quantizer and in this case REQ becomes optimal in terms of rate-distortion optimization.

REQ Scalable Coding for the Case of a Weighted MSE (WWMSE) Measure

The common measure used in audio coding is WMSE instead of MSE. Therefore we will not get the best performance by performing REQ on the original signal. For an optimal entropy coded SQ, the WMSE of the original signal is equivalent to MSE of the companded signal [30], if the compander function is related to the quantization level density function by:

$$\acute{c}(x) = \Delta\Lambda(x) \tag{2.42}$$

Note that, the the compander function here is not the same as typical compander function used in non-uniform quantizers. In non-uniform quantizers this function is obtained from pdf of the input signal. But here it is obtained from the weighting function of WMSE measure.

The above statement means that if REQ is applied to the companded signal instead of

the original signal we can achieve asymptotic optimality. Therefore, a uniform quantizer can always be used for entropy-constrained quantization of a signal in the optimal way: if the weighting function equals unity (MSE measure) uniform quantization is applied to the original signal otherwise it is applied to the companded signal. In fact when the weighting function equals unity the companding function is unity too. Following the above discussion and conditions, it is shown in [30] that in general for REQ we can write:

$$D_{ns}(R) = D_s(R_b + R_e) \tag{2.43}$$

where ns and s are subscripts for non-scalable and scalable, R_b and R_e are the entropies of the base layer and enhancement layer outputs in a scalable coder and $R = R_b + R_e$ is the entropy of an equivalent non-scalable coder output. This says if we have a multi-layer scalable coder of Figure 4.1 in which each layer has its own output entropy, the resulting distortion of the overall coder can asymptotically reach the distortion of a non-scalable coder in which entropy of the quantization output equals the sum of entropies of the layers in the scalable coder.

To summarize our discussion we can say: The overall rate-distortion function of a REQ scalable coder and a non-scalable coder are asymptotically the same, when uniform quantizers are used for them or equivalently when the MSE measure is used for the distortion minimization.

The optimization criteria considered in the whole above statements was the rate-distortion relation. This function is the relation between the distortion and the entropy of quantizer outputs in the coders. However because entropy coding is performed for output of each quantizer we have an increase in rates. For example if we are using Huffman coding we have the following relation between entropy and bitrate of each quantizer output:

$$R \leq B \leq R + 1 \tag{2.44}$$

where B is the bitrate after Huffman coding.

Now consider the REQ scalable coder of 4.1. In such a coder, Huffman coding is performed independently for quantizer output of each layer. The above relation holds for

each of layer:

$$R_b \leq B_b \leq R_b + 1$$
$$R_{ei} \leq B_{ei} \leq R_{ei} + 1 \tag{2.45}$$

where the indices b and ei are used for base layer and ith enhancement layer.
So we can write the following relation for the overall coder:

$$R_b + R_{e1} + R_{e2} + ... \leq B_b + B_{e1} + B_{e2} + \cdots \leq R_b + R_{e1} + R_{e2} + \cdots + N \tag{2.46}$$

where N is the number of layers.
As it can be seen the bounds in the inequality is loosened by the number of layers N compared to that of an equivalent non-scalable coder in which $R = R_b + R_{e1} + R_{e2} + ...$.
This means that although the overall rate of the scalable coder and the rate of non-scalable coder are the same in terms of entropy, they are not the same in terms of bitrate which is the final parameter we are interested in for scalable bitrate coding. This happens because entropy coding of the layer outputs in scalable coding are done independently. This is an important factor for the sub-optimality of a scalable coder compared to a non-scalable coder.

2.3.2 Bit-Plane Coding (BPC)

In Bit-Plane coding (BPC), the binary representation of the input signal is sequentially scanned and coded from the most significant bit (MSB) to the least one (LSB). The coding is performed for each bit-plane separately using an entropy coding technique. The k^{th} bit-plane refers to the k^{th} bits of a sequence of signal samples/coefficients. BPC provides lower computational complexity compared to the REQ technique since it only needs a single scan of the coefficients in a frame. At the decoder, the signal can be fully or partially reconstructed depending on how many bit-planes it receives. Thus, a scalable coding scheme is provided by BPC: the more bit-planes decoded, the higher quality signal can be reconstructed.

Coding schemes including the bit-sliced arithmetic coding (BSAC) [71] with the improved performance using SPIHT[7] [72, 73], ESC [74], EZK [75] are examples of BPC tech-

[7]Set Partitioning in Hierarchial Trees. In SPIHT, the wavelet coefficients are organized in a hierarchial

niques that can be used for scalable audio and image coding. Techniques like SPIHT are mostly used in image coding. For scalable video coding, typically there are three types of scalability: temporal, spatial and quality scalability and the current standards support combinations of them [76–78]. Usually, the scalable image coding techniques, including the BPC techniques, are also applied to the second and the third types of scalable video coding [79].

For audio, we can refer to the Bit-Plane Golomb code (BPGC) [23, 80], as a specific case of BPC. It has been shown that, the Golomb code is optimal for a sources with geometrical distribution [81]. It was also shown that, BPGC provides a fair approximation of an entropy-constrained scalar quantizer (ECSQ) for Laplacian sources. That is why it has become a good replacement for the other BPC techniques for audio coding. BPGC is now the BPC technique used in MPEG-4 audio scalable to lossless audio coding (SLS). In Chapter 5 we will talk about the BPGC in more detail.

2.4 MPEG-4 Audio Coding and Scalability

2.4.1 A Brief Introduction to the Advanced Audio Coding (AAC)

Advanced Audio Coding (AAC) first was introduced in MPEG-2 Audio. The first MPEG-2 audio coding system was MPEG-2 BC/LSF [82] which worked very similar to MPEG-1 layer 3 (MP3) but employed multichannel bitstream format as well. This type of MPEG-2 coder was compatible with the previous versions of MPEG audio coders. It also supported lower sampling frequencies. BC in MPEG-2 BC/LSF corresponds to the backward compatibility of MPEG-2 towards MPEG-1, and the extension of sampling frequencies to lower ranges (16, 22.05, and 24 kHz) is denoted by LSF (lower sampling frequencies). However, because of the backwards compatibility constraint of the MPEG-2 BC/LSF, it was not practical for coding 5-channel audio at rates below 640 kb/s. MPEG started research on a more advanced coding system but not necessarily backward-compatible. This led to the Non-backward Compatible/Advanced Audio Coding NBC/AAC algorithm or briefly AAC [2]. This audio coding system could produce an acceptable quality at 320 kb/s for five full-bandwidth channels.

tree structure where each parent is associated with a set of children. The coding efficiency is mainly achieved by scanning the tree nodes in a specific order from the parents to the children.

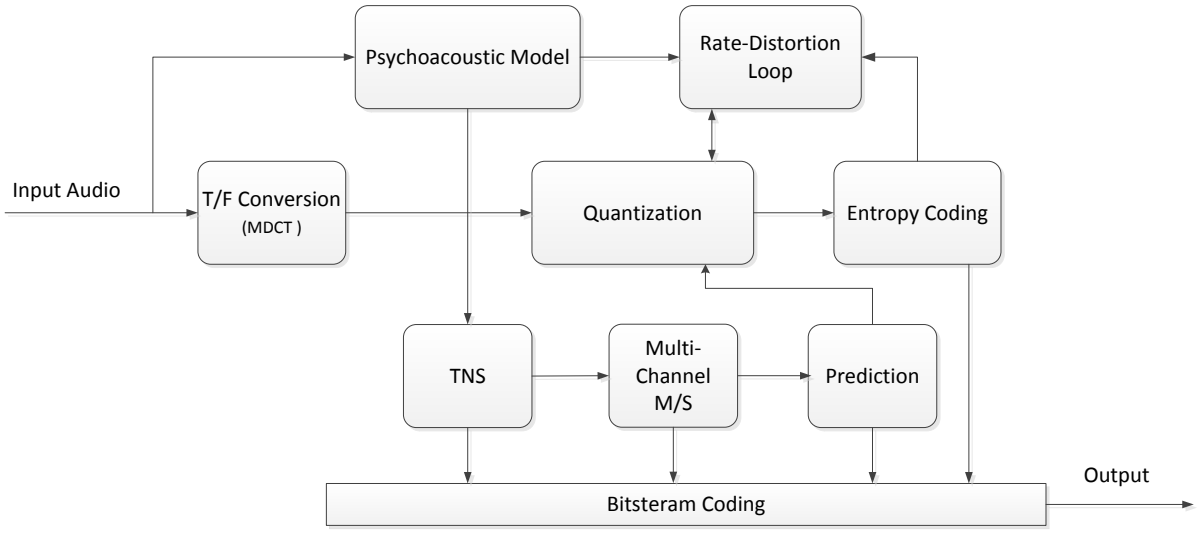


Fig. 2.10 Advanced Audio Coder (AAC) introduced in MPEG-2.

AAC (See figure 2.10) is similar to MP3 and MPEG-2 BC/LSF in that it uses a MDCT filter bank which gives the spectral coefficients and a psychoacoustic model which is basically the same model used in MP3. It also supports multichannel coding of MPEG-2 BC/LSF. However, it incorporates new audio coding tools which were not used in the previous standards and make it a more efficient coder. These include window shape adaptation, spectral coefficient prediction, temporal noise shaping (TNS), and sample-rate (bandwidth) and bit-rate scalability [83]:

AAC allows for using two types of windows, a sine window and a Kaiser-Bessel designed (KBD) window. The window shape adaptation is used to optimize filter-bank frequency selectivity. For example when a narrow pass-band is more beneficial than strong stop-band attenuation the sine window can be selected, but a Kaiser-Bessel designed (KBD) window is selected in the opposite case. Spectral coefficient prediction (over time) is used in AAC for the coefficients that are below 16 kHz. This improves the coding efficiency. A second-order lattice predictor is used and updated on each frame using a backward adaptive least mean square (LMS) algorithm. The TNS module is used to control the pre-echo effect[8] and the sample-rate scalability allows working with different sample-rates and hence coding audio signals with different bandwidths.

[8]Pre-echo occurs in frame-based systems when the distortion increases prior to the onset of the signal.

The bit-rate scalability which is of our special interest is discussed in a separate section in this chapter.

2.4.2 Overview of the MPEG-4 Audio Coding System

The current MPEG-4 audio coding standard specifies coding of natural and synthetic sources. Very low bit rates can be achieved through the use of synthetic speech and music such as text-to-speech and MIDI, while for higher bitrates "natural audio coding" is used providing integrated coding tools that make use of different signal models. Choosing the coding tools depend on the desired bit rate, bandwidth, complexity and quality.

Natural audio coding include a set of tools for coding of natural sounds at bit rates ranging from 200 b/s to 64 kb/s per channel and three distinct algorithms are integrated for that purpose: Parametric Coding, Code Excited Linear Prediction (CELP) Coding and Transform Coding.

The parametric coding is used for bit rates of 2–4 kb/s and 8kHz sampling rate together with 4–16 kb/s and 8 or 16 kHz sampling rates. For higher quality narrow-band and wide-band CELP can be used for 8 kHz and 16 kHz sampling rates respectively. And finally for general audio coding at bit rates above 16 kb/s General Audio coder (GA coder) is used which employs a time/frequency perceptual coder. This coder (GA) is basically MPEG-2 AAC coder with extensions for bit-rate scalability.

2.4.3 General Audio Coder

Figure 2.11 shows a block diagram of the General Audio Coder in MPEG-4 audio. As mentioned above the coder is built based on MPEG-2 AAC with some extended features. These features are highlighted in the figure and include: Perceptual Noise Substitution (PNS), Long-Term Prediction (LTP), Twin VQ coding and Scalability.

Perceptual Noise Substitution (PNS) It is used to model transform-coefficients in noise-like frequency subbands. Reduction in bitrate is achieved since only a parametric representation is required for each PNS subband rather than the full quantization and coding of the transform coefficients [83, 84].

Long-Term Prediction (LTP) Since the tonal signals are highly predictable LTP is used to increase coding efficiency of that type of signals. Speech pitch prediction techniques

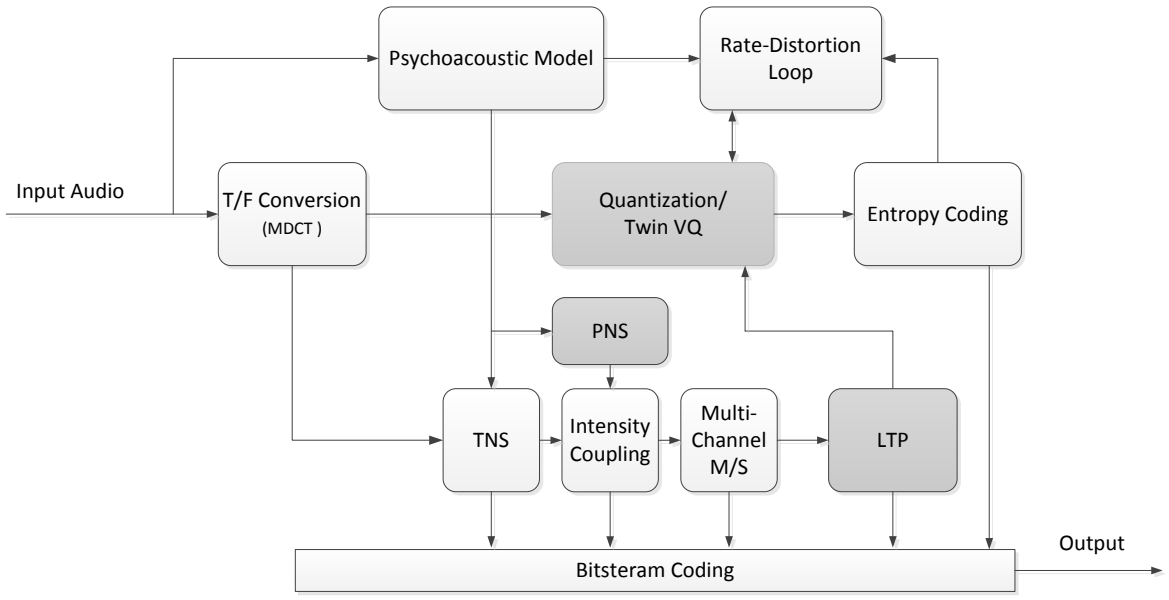


Fig. 2.11 MPEG-4 General Audio coder.

are used for this purpose but the only main difference between the common techniques in a speech coder and in MPEG-4 GA is that in the latter the LTP is performed in frequency domain while in speech coders it is performed in time domain. The complexity of MPEG-4 GA LTP scheme is considerably reduced compared to the MPEG-2 AAC prediction scheme [83, 85].

Twin VQ Twin VQ is an acronym of the Transform domain Weighted Interleave Vector Quantization [83]. The Twin VQ performs vector quantization of the transform coefficients based on perceptually weighted model. First the coefficients are flattened and normalized using LPC and Bark-Scale envelope coding, then they are interleaved to sub-vectors and finally they are quantized based on a perceptually weighted vector quantization [86–88].The Twin VQ provides high coding efficiencies at very low bit rates (6-8 kb/s) even for musical and tonal signals [83].

2.4.4 Rate-Distortion Loop in AAC

In AAC there is a nested loop which controls the bitrate and the distortion in each frame. The “rate loop” which is the inner loop, increases the quantizer step size for each scale-

factor band of the current frame (by decreasing the scalefactors[9]) and quantizes the input coefficients until the output quantized coefficients can be coded with the available number of bits for this frame. After completion of the inner loop, the "distortion loop" (the outer loop) checks the distortion of each scalefactor band and if the allowed distortion is exceeded (compared to the masking threshold calculated for that band), increases the scalefactor and calls the inner loop again.

2.4.5 Scalable Coding in MPEG-4 Audio

Scalable Advanced Audio Coding (S-AAC)

As mentioned before there are different types of scalability. The bit-rate scalability is the one used in MPEG-4 which is one of the main cores of that standard. Figure 2.12 shows the MPEG-4 bit-rate scalability scheme. This scalable coding scheme is based on Reconstruction Error Quantization (REQ). In such a scalable coder the first stage which is called the core layer or base layer encodes the input audio based on a lossy compression scheme. In the next stage which is called an enhancement layer an error signal is calculated by subtracting the reconstructed signal from the input signal where the signal reconstruction is performed by a local decoder. This error signal is encoded again to form the compressed residual. The above sequence of steps can be repeated to form as many enhancement layers as required. For example we can have a core layer which has 32 kb/s bitrate and we have two enhancement layers which use 16 kb/s and 8 kb/s. At the receiver side we can have reconstructed audio at 32 kbps, 48 kbps or 56 kbps.

The important thing to mention here is that the base layer bit stream should guarantee reconstruction of the original input audio with few audible artifacts which makes it acceptable in case of no enhancement layers are used. Additional enhancement layers can be added to increase the quality of the decoded audio.

Scalable-to-Lossless Coding (SLS)

There is another scalable coding scheme which is one of the extensions of the MPEG-4 scalable audio coding and is based on bit-plane coding [11]. In this technique, first the

[9]The scalefactor is a parameter in AAC quantization operation by changing of which, the quantization step size changes in each scalefactor band. We will discuss it in more detail in Chapter 5.

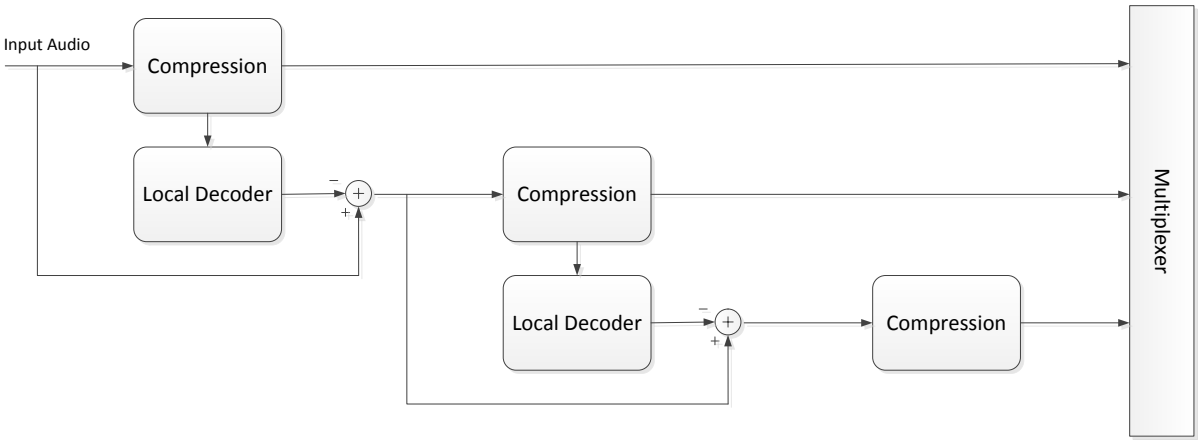


Fig. 2.12 MPEG-4 Scalable Coder.

quantized spectral coefficients are grouped into frequency groups. Then in each group bit planes of the coefficients are grouped into bit slices, in each of these bit slices there can be one or more bits of the coefficients. Then the bit slices are coded using arithmetic coding. The coded slices then can be sent in order of their significance (MSBs first) and this way a scalable coding pattern is created. By choosing appropriate number of bits for the slices scalability with steps of even 1kbps/channel can be provided. That is why the technique is called fine-grain scalability.

This scalable coding tool is called MPEG-4 scalable-to-lossless (SLS). It was released as a standard audio coding tool in June 2006 [12, 89] and there has been research on improving its performance [17, 18, 72, 90]. The latest update to SLS picks the so called bit-plane Golomb code (BPGC) as its BPC techniques. SLS has two modes: a perceptual-core mode and a non-core mode. In both of these modes integer MDCT (IntMDCT) is used instead of MDCT. We will discuss SLS in more detail in Chapter 5.

Chapter 3

Psychoacoustics and Practical Psychoacoustic Models

Psychoacoustics is a science that studies the way audio signals are perceived by Human Auditory System (HAS). The masking effect which is evaluated in this science, is used in audio compression as a very useful feature. Using this property, parts of an audio signal that are not audible could be removed without changing the perceptual quality of the original signal or creating audible artifacts. In the audio coding systems normally there are psychoacoustic models which simulate the HAS. These models calculate the masking thresholds for the different frequency bands of the input signal which are used by the other stages of the encoder.

In this chapter we will briefly discuss some important properties of the HAS and conclude the chapter by talking about two important practical psychoacoustic models.

3.1 Human Auditory System (HAS)

The HAS converts an acoustic signal to a neural signal and then sends it to the brain where it is perceived. The HAS consists of three main parts: The outer ear, the middle ear and the inner ear. Each of these plays a different role in processing the received sound (see Fig. 3.1).

The outer ear consists of the Pinna, Ear Canal, and the Eardrum. The main role of the outer ear is to receive the air pressure variations caused by a sound source and transmit it

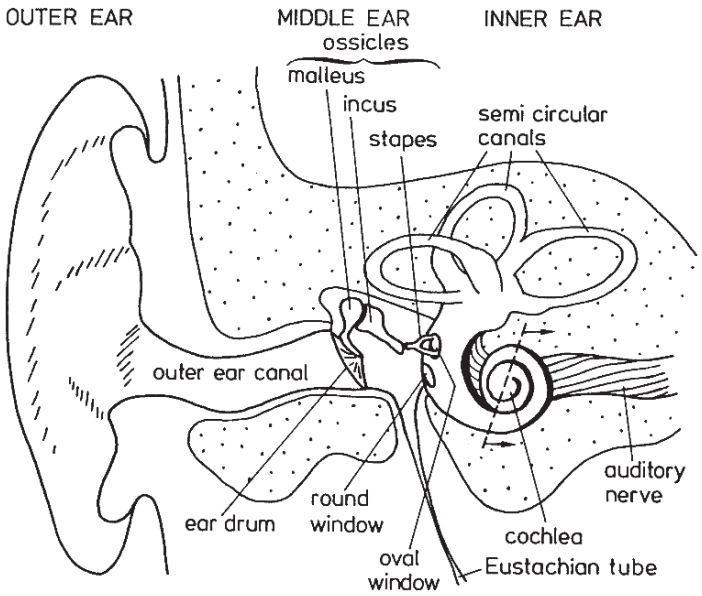


Fig. 3.1 Human Auditory System (HAS) (from [91]). The HAS consists of three main parts: The outer ear, the middle ear and the inner ear. The air pressure variation is received by the outer ear. The middle ear acts as an impedance matcher between the outer and the inner ear, and the inner ear converts the acoustic vibration to the neural information and sends it to the brain where the sound is perceived.

to the middle ear. The pinna acts as an acoustic antenna. Also, its special shape helps in recognizing the location and the direction of the sound source.

The ear canal conducts the received air pressure variations to the eardrum. The air canal can be modeled by an acoustic tube with a resonance at 3 kHz. When the eardrum receives these variations it starts to vibrate and the resulting vibrations are transferred to the middle ear.

The middle ear consists of three small bones. The main role of the middle ear is to transfer the sound vibrations from the outer ear to the inner ear. These vibrations are received by a liquid inside the inner ear. Therefore, this transfer is performed by the inner ear in an efficient way by matching the acoustic impedance of the air and the liquid impedance inside the inner ear.

The main part of the inner ear is called the Cochlea. It is a spiral-shaped cavity which makes 2.5 turns around its axis and is filled with a liquid. The cochlea has two ends. One end which receives the vibrations from the middle ear (at the oval window) is called the

base and the other end is called the apex. The length of the inner ear is about 3.5 cm and it is wider at the base (about 2 cm) compared to the apex. Inside the Cochlea there is a membrane which is called the Basilar Membrane and separates the two liquid-filled tubes of the Cochlea.

The vibrations and the pressure difference of the liquid in these tubes stimulates the Basilar Membrane so that it finds a peak of vibration at a specific point depending on the sound frequency. For the higher frequencies this point is closer to the base, while for the lower frequencies it is closer to the apex. This way a sound wave propagates through the basilar membrane. The neural receptors are connected to the different locations of the basilar membrane and are triggered by the physical movements on the membrane. Therefore, they are tuned to different frequency bands. This is how the perception information is sent to the brain for different frequencies.

The basilar membrane can be considered as a string with a decreasing tension moving from the base to the apex [92]. This tension changes by about four orders of magnitude. The wave propagation speed for a string is

$$c = \sqrt{\frac{T}{p_L}}, \tag{3.1}$$

where T is the string's tension and p_L is its linear density. The wavelength for the frequency component f is

$$c = \frac{1}{f}\sqrt{\frac{T}{p_L}} = \frac{c}{f} \tag{3.2}$$

The string impedance can be obtained by

$$z_0 = \sqrt{p_L T}, \tag{3.3}$$

and the wave power by

$$P_w = \frac{1}{2} z_0 \omega^2 A_w^2 = \frac{1}{2}\sqrt{p_L T}\,\omega^2 A_w^2, \tag{3.4}$$

where $\omega = 2\pi f$ and A_w is the wave amplitude.

When a wave component at frequency f propagates from the base to the apex, its wavelength decreases because of the tension decrease. To satisfy the requirement of the power constancy its amplitude increases. However, the propagation is lossy. The power dissipa-

tion depends on the frequency and increases with the amplitude. Therefore, a frequency-dependent peak of vibration occurs at a specific point of the basilar membrane.

The higher frequencies are more affected by the propagation loss and hence, their resonance point is closer to the base. On the other hand, for the low frequencies there is less power loss and therefore, they propagate more widely and peak at a point closer to the apex. About two thirds of the Cochlea vibrates for the low frequencies (one fourth of the audio bandwidth). This is why the spread of masking (as will be discussed later) toward higher frequencies is wider.

3.2 Critical Bands

As explained in the previous section, depending on the input audio frequency, a specific area along the basilar membrane has the maximum vibration. Therefore, the basilar membrane can be considered a spectrum analyzer or a filter bank which decomposes the audio signal into sub-bands of the frequency components. These bands are called the Critical Bands (CBs). The bandwidth of a critical band can be defined as the smallest frequency difference between two tones such that they can be heard as two different tones. The critical bandwidth can be approximated for an average listener by [91]

$$BW_c(f) = 25 + 75[1 + 1.4(\frac{f}{1000})^2]^{0.69} \tag{3.5}$$

where f is the frequency in Hz. A CB represents a constant physical distance along the basilar membrane. In fact, the width of a CB is corresponds to a distance along the basilar membrane of about 1.3 mm. This distance is not constant if you consider the corresponding filter bank bandwidths in Hz, but constant in another scale called Bark scale. A frequency component which is one CB apart from another one, has a distance of one Bark from another component. The relation between Bark and Hertz for a frequency component f can be approximated as [91]

$$z(f) = 13 \arctan(\frac{0.76f}{1000}) + 3.5 \arctan((\frac{f}{7500})^2) \tag{3.6}$$

where $z(f)$ is the frequency in Bark. Table 3.1 shows the frequency ranges of the discretized critical bands for the audio spectrum [91]. As mentioned before, the HAS acts as a filter

bank with critical bands. The center frequency of each band is also shown in the table. Although the frequency ranges follow a non-linear pattern in Hz, they are linear in the Bark scale. In other words, the critical bandwidths and the distance between the center frequencies is almost constant in Bark.

z (Bark)	f_l (Hz)	f_h (Hz)	BW (Hz)	f_c (Hz)
0	0	100	100	50
1	100	200	100	150
2	200	300	100	250
3	300	400	100	350
4	400	510	110	450
5	510	630	120	570
6	630	770	140	700
7	770	520	150	840
8	520	1080	160	1000
9	1080	1270	190	1170
10	1270	1480	210	1370
11	1480	1720	240	1600
12	1720	2000	280	1850
13	2000	2320	320	2150
14	2320	2700	380	2500
15	2700	3150	450	2900
16	3150	3700	550	3400
17	3700	4400	700	4000
18	4400	5300	900	4800
19	5300	6400	1100	5800
20	6400	7700	1300	7000
21	7700	9500	1800	8500
22	9500	12000	2500	10500
23	12000	15500	3500	13500
24	15500-			

Table 3.1 Frequency ranges and center frequencies for the discretized critical bands as a filter bank covering the audio spectrum [91]. The critical bandwidths and the distance between the center frequencies is almost constant in Bark.

3.3 Sound Pressure Level (SPL)

The human ear perceives the sound as the air pressure variation. Sound is usually expressed in terms of its intensity, i.e. the amount of energy which passes the surface unit in the time unit. The sound pressure level (SPL) is a quantity which is obtained from this intensity and is defined as [93]

$$\text{SPL} = 10\log_{10}(\frac{I}{I_0}) \tag{3.7}$$

where I is the sound signal intensity and I_0 is the reference sound intensity and is equal to $10^{-12}(Watt/m^2)$. Since the sound intensity is proportional to the squared air pressure, the SPL can be also expressed in terms of the magnitude of the air pressure variations [93]:

$$\text{SPL} = 10\log_{10}(\frac{P}{P_0})^2, \tag{3.8}$$

where $P_0 = 20\mu$ Pa is the threshold of hearing a tone at 2 kHz.

The SPL range which can be perceived by HAS is between 10^{-5} Pa to 10^2 Pa [93]. The SPL is expressed with a logarithmic scale and is in dB. Therefore, the dynamic range of the sound perceived by HAS is between 0 dB and 120 dB. In fact, 120 dB is the threshold of pain for the ear (on average, since it is different for different frequencies).

3.4 Absolute Threshold of Hearing

Absolute threshold of hearing or the threshold in quiet refers to the minimum power of a pure sound tone which can be heard in a noise-free environment. This threshold is different for different frequencies. This phenomena is related to the perception threshold of the neurons in HAS. All the components that are below this threshold are masked and can not be heard.

As can be seen in Fig. 3.2 the minimum threshold occurs at around 3 kHz. This is because the ear canal acts as an acoustic tube approximately 2.8 cm long with resonance. This tube exhibits a wide resonance peak for $2 - 5$ kHz. Also, it can be seen that the threshold goes relatively high for frequencies higher than 16 kHz and the frequencies above the 20 kHz are not audible even having a very high power.

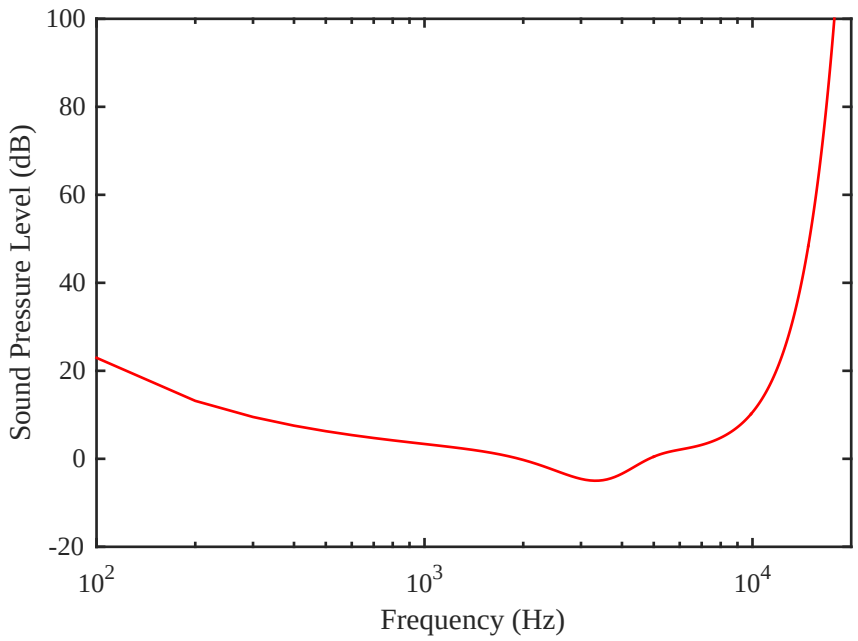


Fig. 3.2 Absolute threshold of hearing or the threshold in quiet. The threshold depends on the frequency and has a minimum around 3 kHz because of the ear canal's resonance. The threshold goes very high for high frequencies so that above the 20 kHz the sound becomes inaudible for the human ear.

The threshold in quiet can be approximated by the following formula [94]:

$$T_q(f) = 3.64(\frac{f}{1000})^{-0.8} - 6.5e^{-0.6(\frac{f}{1000}-3.3)^2} + 10^{-3}(\frac{f}{1000})^4, \tag{3.9}$$

where f is the frequency in Hz and T_q is the SPL of the threshold in dB.

3.5 The Masking Effect

Human Auditory System has special characteristics. One of its most important characteristics is the masking effect. In the masking effect one audio signal makes another signal inaudible. There are typically two types of masking [91]:

- Temporal masking
- Frequency masking (also called simultaneous masking)

3.5.1 Temporal Masking

In the temporal masking, one part of an audio signal (masker) makes the weaker parts inaudible which are present in a small time interval after or before the masker. There are also two types of temporal masking:

- Forward Masking
- Backward Masking

The forward masking occurs when the masker comes before the maskee in time, and in the backward masking the maskee comes before the masker. The forward masking can last about 300 to 500 ms, while in the backward masking the masker affects an interval of $1-20$ ms before it. In both cases the masking interval depends on the intensity and the duration of the masker.

3.5.2 Frequency Masking

In the frequency masking, an audio signal prevents a weaker signal (which is playing at the same time) to be heard. In fact, in this type of masking, the threshold of quiet for different frequencies changes in the presence of another signal. These signals can be a single frequency component or a set of components. Frequency masking can be divided into two main types of masking:

- A frequency component of the audio signal (called a tone) makes some weaker neighboring components inaudible: Tone Masking Noise (TMN).
- A set of frequency components included in a critical band of the audio signal makes some weaker neighboring components inaudible: Noise Masking Tone (NMT).

3.6 Spread of Masking

Spread of masking refers to the pattern a masker masks its neighboring frequencies. The spread of masking is different for a masker with different frequencies or different SPL and it depends directly on both of these parameters. Several functions have been proposed so far which model the spread of masking. Normally, analyzing the spread of masking is much easier in the Bark scale since this scale is very compatible with the auditory properties of

the HAS specifically the basilar membrane. In fact, using the Bark scale the frequency dependence of the spread of masking pattern and its modeling functions disappears.

The spread of masking can be modeled by an approximately triangular function with the slopes of +25 and −10 dB per Bark. An analytical expression for this function was given in as [95]

$$SF(x) = 15.81 + 7.5(x + 0.474) - 17.5\sqrt{1 + (x + 0.474)^2} \tag{3.10}$$

where x is the frequency distance in Bark and $SF(x)$ is the spread of masking in dB.

3.7 SMR and NMR

In audio coding based on the transform coding, typically a psychoacoustic model is used which gives the masking thresholds for different frequency bands. This threshold means that the signal components which have energies below this threshold are masked in that band by the other parts of the signal. This fact could be thought of in another way: if a noise is added to the signal and the noise energy is below the masking threshold, the noise becomes inaudible. This is the main reason for using a psychoacoustic model in audio coding which leads to a great amount of compression.

When the transform coefficients are quantized, the quantization error could be thought of a noise added to the original signal. Typically, the signal to noise ratio (SNR) is used to express the amount of noise compared to the signal energy:

$$\text{SNR} = 10 \log_{10} \frac{E_S}{E_N} \quad \text{dB}, \tag{3.11}$$

where E_S and E_N are the signal and noise energies respectively. In practical audio coders, each band of the signal is quantized separately. For each band these energies can be obtained by

$$E_S(n) = \sum_l c_n^2[l], \tag{3.12}$$

and

$$E_N(n) = \sum_l e_n^2[l], \tag{3.13}$$

where $c_n[l]$ is the l^{th} quantized coefficient index in band n, and $e_n[l]$ is the corresponding

quantization error. If the quantization resolution is chosen such that the resulting noise is smaller than the masking threshold, the noise can be inaudible. Therefore, together with the SNR, another ratio is defined and normally is used in audio coding called signal-to-mask ratio (SMR)

$$\text{SMR} = 10 \log_{10} \frac{E_S}{E_M} \quad \text{dB}, \tag{3.14}$$

where E_S and E_M are the signal and the mask energies respectively. This ratio could be interpreted as a perceptual SNR. In an audio coder, these ratios are calculated by the psychoacoustic model for each frequency band. Another commonly used ratio is the noise-to-mask ratio (NMR) which is defined as

$$\text{NMR} = 10 \log_{10} \frac{E_N}{E_M} \quad \text{dB}, \tag{3.15}$$

which can also be obtained by

$$\text{NMR} = \text{SMR} - \text{SNR} \quad \text{dB}. \tag{3.16}$$

This ratio says how strong the noise is compared to the masking threshold and hence how audible it is.

3.8 MPEG Psychoacoustic Models

A psychoacoustic model is a perceptual model based on HAS which gives the masking thresholds for different frequency bands, typically the critical bands. An audio encoder uses these thresholds during the coding process. In this section we will briefly discuss the psychoacoustic models which have been used in MPEG Audio standards. Two main psychoacoustic models have been developed and used in the MPEG Audio. In MPEG-1 layers I and II the psychoacoustic model 1 was used. For MPEG-1 layer III (known as MP3) a new psychoacoustic model was developed. This model is also the one used later in MPEG-2 Advance Audio coding (AAC) and MPEG-4 AAC with slight possible modifications.

3.8.1 MPEG Psychoacoustic Model 1

In MPEG psychoacoustic model 1 [1], the simultaneous masking threshold for an NMT, is calculated using the following spreading function:

$$v_{f(z_i,z_j)} = \begin{cases} 17dz - 0.4X(z_j) + 11 & -3 \leq dz < -1 \\ (0.4X(z_j) + 6)dz & -1 \leq dz < 0 \\ -17dz & 0 \leq dz < 1 \\ -17dz + 0.15(dz - 1)X(z_j) & 1 \leq dz < 8 \end{cases} \tag{3.17}$$

where z_j is the central frequency of the masker critical bank in Bark, z_i is the maskee in Bark and dz is the distance between the maskee and masker: $dz = z(f_{maskee}) - z(f_{masker})$. $X(z_j)$ is the sound pressure level of the masker in dB and $v_{f(z_i,z_j)}$ is the spreading function giving the masking effect of the critical band with central frequency of z_j on tone z_i. The masking threshold for the maskee z_i resulting from the masker z_j is obtained by

$$M(z_i, z_j) = X(z_j) + v_s(z_j) + v_f(z_i, z_j) \tag{3.18}$$

where

$$v_s(z_j) = -2.025 - 0.175z_j \tag{3.19}$$

is the self masking level of the masker. Finally the total masking threshold in dB for maskee z_i from all effective maskers is obtained by

$$M_T(z_i) = 10\log_{10}(10^{T_q(z_i)/10} + \sum_{j=1}^{N} 10^{M(z_i,z_j)/10}), \tag{3.20}$$

where $T_q(z_i)$ is the threshold in quiet for z_i. For maskers that are out of the specified distance range from maskee, the individual masking threshold is assumed to be -8dB.

3.8.2 MPEG Psychoacoustic Model 2

MPEG psychoacoustic model 2 is a more advanced model compared to its preceding and is the one used in MPEG-4 AAC. As noted in Section 3.5, there is a difference between the behaviors of a noise masking a tone and a tone masking a tone/noise. In MPEG psychoacoustic model 2 [1, 56], the tonality of a signal is estimated by a tonality index

which is a function of time and frequency. Using this index in the calculations leads to a generalized psychoacoustic model which integrates the two masking types into a single model. For this, a simple polynomial predictor was proposed and used [56]. This prediction is for the magnitude and phase of the frequency lines. Defining

$$\begin{aligned} r(t,\omega) &: \text{magnitude at time } t \text{ and frequency } \omega \\ \phi(t,\omega) &: \text{phase at time } t \text{ and frequency } \omega, \end{aligned} \tag{3.21}$$

The prediction is performed as

$$\begin{aligned} \hat{r}(t,\omega) &= r(t-1,\omega) + (r(t-1,\omega) - r(t-2,\omega)) \\ \hat{\phi}(t,\omega) &= \phi(t-1,\omega) + (\phi(t-1,\omega) - \phi(t-2,\omega)), \end{aligned} \tag{3.22}$$

where $\hat{r}$ and $\hat{\phi}$ are the predicted magnitude and phase respectively. Then a metric called the unpredictability is calculated. This metric is the Euclidean distance between the predicted and the actual values. It is also referred to as the 'chaos measure' and is defined as

$$c(t,w) = \frac{\text{dist}([\hat{r}(t,\omega), \hat{\phi}(t,\omega)], [r(t,\omega), \phi(t,\omega)])}{r(t,\omega) + \text{abs}[\hat{r}(t,w)]}. \tag{3.23}$$

Ideally, for a pure tone the prediction will be accurate and $c(t,\omega)$ becomes zero. On the other hand, if the signal is a noise, $c(t,\omega)$ can take values up to unity with a mean of 0.5. As a result, $c(t,\omega)$ can be limited to a range from a very small value like 0.05 (since a completely pure tone is very rare) to 0.5 representing a fully tonal and a noise-like signal respectively:

$$c_l(t,\omega) = \max\{0.05, \min[0.5, c(t,\omega)]\} \tag{3.24}$$

The chaos measure is mapped to the tonality measure using

$$v(t,\omega) = -0.43 \log_{10}(c_l(t,w)) - 0.299, \tag{3.25}$$

where $v(t,\omega)$ is the tonality index mentioned above and is used in the masking calculations explained below.

The frequency coefficients of the signal are grouped into the partitions with 1/3 critical band energy representation. For each of this partitions a different masking threshold is

calculated. The energy of a the signal in a partition can be obtained by

$$s_b = \sum_{\omega_l}^{\omega_h} r(\omega_i)^2 \tag{3.26}$$

where b is the partition index and ω_l and ω_h are the lowest and highest frequency in partition b. Then a weighted unpredictability c_b is calculated for each partition using the chaos measure described above as

$$c_b = \sum_{\omega_l}^{\omega_h} r(\omega_i)^2 c(\omega_i). \tag{3.27}$$

In the next step both s_b and c_b are convolved with the masking spreading function. The convolved c_b is then mapped to the tonality index t_b using (3.25). Then, the masking level is calculated using the tonality index and the convolved spectrum s_b. For this a signal to noise ratio is calculated for each threshold band as

$$\mathrm{SNR}_b = \max(minval_b, t_b \times TMN_b + (1 - t_n) \times NMT_b) \tag{3.28}$$

where NMT_b and TMN_b are estimates for the masking capabilities of tone masking noise and noise masking tone and $minval_b$ is a minimum value.

Now the threshold in quiet is taken into account and compared with the initial threshold for adjustment. Note that, the threshold in quiet is dependent on the sound pressure level of the audio output which is not known in advance. Therefore a normalization is required. In both psychoacoustic models 1 and 2 this is done by assuming that the LSB of the signal is audible. In other words, it is assumed that the LSB of a 4 kHz signal is associated with an SPL near 0 dB. This is because for a normal listener the threshold in quiet at 4kHz (which has approximately the minimum threshold) is about zero in SPL.

By doing the above adjustment, a preliminary threshold for each threshold band b of the current audio block is obtained. Then final masking threshold is calculated as

$$thr_b = \max(thrp_b, rpelev \times throld_b) \tag{3.29}$$

where thr_b is the final estimated masking threshold, $thrp_b$ is the preliminary threshold calculated for the current block, $throld_b$ is the preliminary threshold of the last block

(which is stored and used for the pre-echo control issues) and *rpelev* is a constant chosen to be 2.

These masking thresholds are obtained for each threshold band representing approximately 1/3 of a critical band. In practice, the signal is decomposed into several subbands. These subbands typically cover several threshold bands. In order to calculate the masking threshold for each subband, the thresholds obtained above are mapped to the spectral densities and then calculating the energy per subband gives the masking threshold for that band. The signal to mask ratio SMR is then obtained by

$$\mathrm{SMR}_n = 10 \log_{10}(\frac{S_n}{thr_n}) \tag{3.30}$$

where thr_n is the final calculated masking threshold for subband n and S_n is the signal energy in this subband.

Chapter 4

Scalable Audio Coding using Watermarking

4.1 Introduction

Figure 4.1 shows the simplified block diagram of the scalable audio coding based on REQ. In this system the signal is quantized by a quantizer designed for a minimum bit rate and acceptable distortion (the base layer). Enhancement layers improve the quality of the base layer signal, refining the quantization by subtracting the quantized signal from the original. This error signal is quantized, encoded and transmitted as the first enhancement layer. This enhancement step can be repeated, to form an ordered set of layers. From the base layer up, each additional layer that the receiver receives is used to refine the quality of the decoded signal.

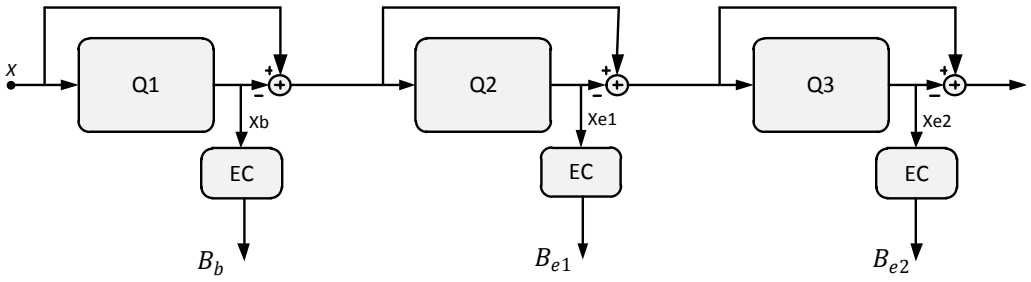


Fig. 4.1 Salable Audio Coding based on REQ. EC is entropy coding.

In terms of Rate-Distortion (RD) performance, REQ has been shown to be asymptotically optimal for the Mean Square Error (MSE) criterion [30]. It asymptotically achieves the performance of an equivalent non-scalable coding if the rate is measured by the entropy of resulting output symbols. However, in practical coding systems symbols need to be encoded in a bitstream using an entropy coding scheme, which adds an overhead for each layer.

In this chapter we propose a method that improves the performance of the scalable audio coding systems based on REQ. In this method we used the Quantization Index Modulation (QIM) which is a technique borrowed form watermarking [96]. Using this technique some of the information of each layer output is embedded (watermarked) in the previous layer. We will show that using this approach, a saving in bit rate is achieved while the distortion is not much affected. In the following section we will discuss AAC quantization and will address an issue which we will take advantage of in our method to improve the performance of the REQ system. Then we will give a brief introduction to QIM technique and finally we will introduce our proposed method Scalable Audio Coding using Watermarking which we will refer to by WSAC in the rest of the chapter.

The chapter is organized as follows. In the next section, we will discuss AAC quantization and its entropy properties. The quantization index modulation (QIM) technique is discussed in Section 4.3. Section 4.4 covers the proposed scalable audio coding using watermarking, the results are presented in Section 4.5 and Section 4.6 summarizes the chapter.

4.2 AAC and Uniform Threshold Quantization

The quantization formula which is used in AAC for quantizing the input MDCT coefficients is given by [3]

$$\begin{aligned} i_x &= \text{sgn(x)nint}(\frac{|x|^{0.75}}{\Delta} - 0.0946) \\ \hat{x} &= \text{sgn(x)}(\Delta|i_x|)^{\frac{4}{3}}, \end{aligned} \tag{4.1}$$

where Δ is the step size parameter, nint() and sgn() denote the nearest integer and signum functions and 0.0946 is the offset value which is also referred to as the magic number. This is a uniform threshold quantizer (UTQ) with a dead-zone around zero and with a

slight compression involved. The UTQ with dead-zone has been shown to be the optimal entropy-constrained quantizer for Laplacian sources in high rates [69]. In such a quantizer, reconstruction points are not in the middle of the quantizer intervals and there is a constant offset in each interval. However, the interval width (or the step size) of the quantizer is constant (uniform threshold) except for the zero interval (the dead-zone) which is larger than the other intervals.

In high-rate theory, for the case of constrained-entropy quantization the relation between the optimal uniform quantizer step size and its entropy is [29]

$$\Delta = 2^{-(R-h(X))}, \tag{4.2}$$

where Δ is the quantizer step size, R is the quantizer output entropy and $h(X)$ is the input signal differential entropy. This equation implies that when you double the resolution of the quantizer (by halving the step size), the output entropy will be increased by one bit. The above relation holds only with the high-rate assumptions where 1) quantization cells are small enough that the source density is constant within the cell 2) each re-construction point is located at the center of the cell and 3) $N \to \infty$. However, in a REQ scalable coding system a low resolution quantizer is used in each layer, e.g., a 4-layer system with 4-bit resolution quantizers. In the following we will obtain a general relation for the entropy of a UTQ and will show that the entropy increment becomes less than unity for the low-rate quantization.

Consider a Laplacian source which is a good approximation for the MDCT coefficients of an audio signal [97]. Our simulations also show that this is true, even for the slightly compressed MDCT coefficients (used in the AAC quantization (4.1)). In the rest of this chapter, by the input signal we mean the MDCT coefficients after this compression. For a Laplacian signal the pdf is

$$f_X(x) = \frac{1}{2}\lambda e^{-\lambda|x|}, \tag{4.3}$$

where $\lambda = \frac{\sqrt{2}}{\sigma}$ and σ is the standard deviation of the signal.

Now consider a UTQ with dead-zone. The probability of the input signal to be in each interval (for $i > 0$) is

$$p_i = \frac{1}{2}(e^{-\lambda t_i} - e^{-\lambda t_{i+1}}), \tag{4.4}$$

where t_i and t_{i+1} are the thresholds of the interval i and we have (except for the dead-zone)

$$t_{i+1} = t_i + \Delta \tag{4.5}$$

which gives

$$p_i = \frac{1}{2}(1 - e^{-\lambda\Delta})e^{-\lambda t_i} \tag{4.6}$$

and the probability of the dead-zone becomes

$$p_0 = (1 - e^{-\lambda T}), \tag{4.7}$$

where $T = \frac{\Delta}{2} + t_o$ (t_o is the offset value of the UTQ).

The entropy of the quantizer output then can be written as

$$\begin{aligned} R &= -\sum_i p_i \log_2(p_i) \\ &= -(1 - e^{-\lambda T}) \log_2(1 - e^{-\lambda T}) \\ &\quad - 2\sum_{i\geq 1} \frac{1}{2}(1 - e^{-\lambda\Delta})e^{-\lambda t_i} \log_2(\frac{1}{2}(1 - e^{-\lambda\Delta})e^{-\lambda t_i}) \\ &= (1 - e^{-\lambda T}) - (1 - e^{-\lambda\Delta})[\sum_{i\geq 1} e^{-\lambda t_i} \log_2(e^{-\lambda t_i}) + \\ &\quad (\log_2(1 - e^{-\lambda\Delta}) - 1)\sum_{i\geq 1} e^{-\lambda t_i}]. \end{aligned} \tag{4.8}$$

Since $t_1 = T$, we have

$$\begin{aligned} \sum_{i\geq 1} e^{-\lambda t_i} \log_2(e^{-\lambda t_i}) &= e^{-\lambda T} \log_2(e^{-\lambda T}) \\ &\quad + e^{-\lambda(T+\Delta)} \log_2(e^{-\lambda(T+\Delta)}) \\ &\quad + e^{-\lambda(T+2\Delta)} \log_2(e^{-\lambda(T+2\Delta)})) \\ &\quad + \dots \\ &= e^{-\lambda T} \log_2(e^{-\lambda T})(1 + e^{-\lambda\Delta} + e^{-2\lambda\Delta} + \dots) \\ &\quad - \lambda\Delta \log_2(e) e^{-\lambda T}(e^{-\lambda\Delta} + 2e^{-2\lambda\Delta} + \dots) \\ &= \frac{e^{-\lambda T} \log_2(e^{-\lambda T})}{1 - e^{-\lambda\Delta}} - \frac{\lambda\Delta \log_2(e) e^{-\lambda T} e^{-\lambda\Delta}}{(1 - e^{-\lambda\Delta})^2} \end{aligned} \tag{4.9}$$

and

$$\sum_{i\geq 1} e^{-\lambda t_i} = e^{-\lambda T}(e^{-\lambda\Delta} + e^{-2\lambda\Delta} + \dots) = \frac{e^{-\lambda T}}{(1-e^{-\lambda\Delta})}, \tag{4.10}$$

then

$$\begin{aligned} R = e^{-\lambda T} &+ \lambda \log_2(e) e^{-\lambda T}(T + \frac{\Delta e^{-\lambda\Delta}}{1-e^{-\lambda\Delta}}) \\ &- e^{-\lambda T}\log_2(1-e^{-\lambda\Delta}) \\ &- (1-e^{-\lambda T})\log_2(1-e^{-\lambda T}). \end{aligned} \tag{4.11}$$

The quantization loading factor L_f is defined as $L_f = x_m/\sigma$, where σ is the standard deviation of the input signal and x_m is the quantization limit which is for instance $2^{13}-1$ in AAC quantization. By properly choosing this loading factor, the probability of the signal being beyond the quantization limit becomes negligible. That is why we used the infinite summation formula for the above geometric series (in all resolutions the quantizer limit is fixed and that probability remains the same for a given L_f). Normally $L_f \geq 7$ is chosen for Laplacian signals to avoid the quantization overload [28]. In Fig. 4.2 the obtained entropy versus the quantization resolution is plotted for the AAC quantizer and for three different values of L_f. Note that changing the step size parameter Δ in (2.10) changes the quantizer resolution at the same time. For instance, if Δ changes from $\Delta = 1$ to $\Delta = 2$, the resolution of the quantizer is halved. In other words the resolution (number of levels) can be expressed by: $N = 2x_m/\Delta$. The horizontal axis in Fig. 4.2 is $\log_2(N)$.

It can be seen that for high resolutions the slope of the curves $\frac{\Delta R}{\Delta b}$tends to unity (as predicted by the high-rate equation), while for lower resolutions it is less than unity. This fact that in low resolutions the entropy increment becomes less than unity for doubling the resolution is the issue that we mentioned in the introduction section. We will take the advantage of that in our proposed coding system.

4.3 Quantization Index Modulation (QIM)

QIM is one of the techniques used in watermarking. Consider for instance the AAC quantizer in which the quantized values are integers (nint() function in (2.10)). The idea is to quantize a signal to the nearest-even or nearest-odd integer rather than to the nearest

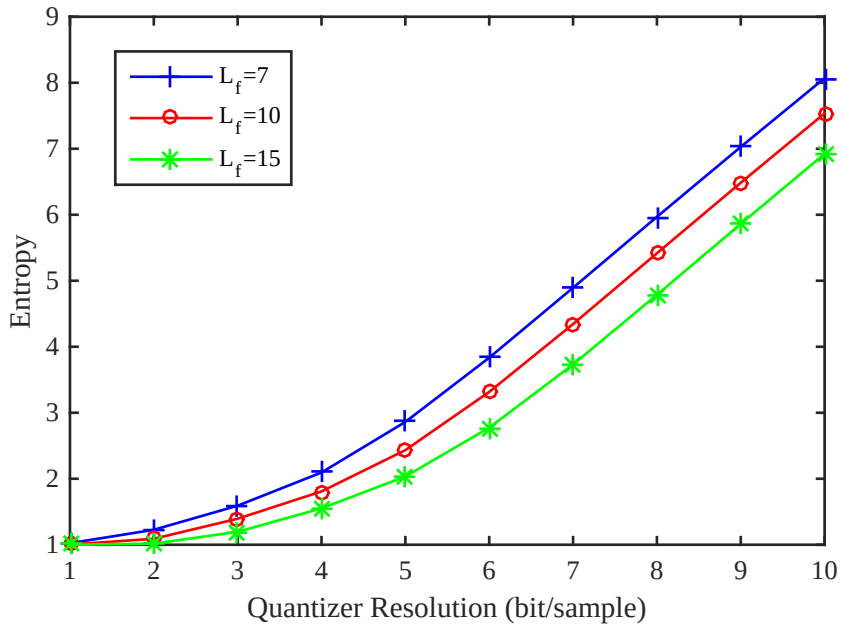


Fig. 4.2 Entropy of a UTQ versus quantization resolution for three values of input σ ($L_f = 7$, $L_f = 10$ and $L_f = 15$)

integer value. If the receiver receives an even value a 0 bit is decoded and otherwise a 1 bit is decoded. Therefore, QIM enables us to watermark one bit of data into the signal at the cost of higher distortion. Figure 4.3 shows the QIM based on the nearest-even integer quantization.

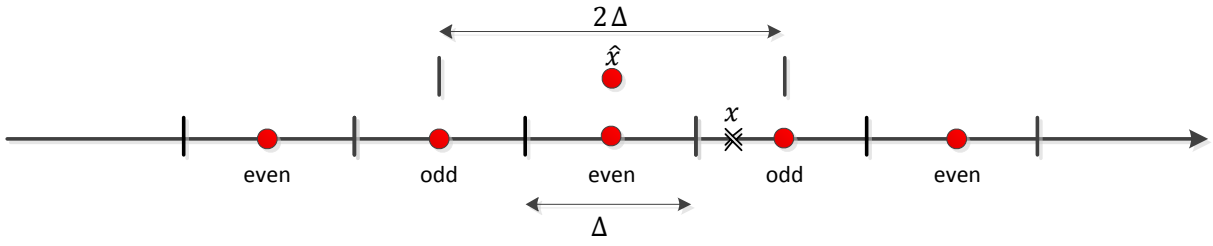


Fig. 4.3 Quantization Index Modulation and its rough equivalency to quantization with half resolution.

However, the point (which is desired here) is that if we first double the resolution of a quantizer and the use the QIM, that is approximately equivalent to using the original quantizer in terms of rate-distortion. Figure 4.4 shows the rate distortion performance of an AAC quantizer without and with using QIM denoted by UTQ and QIM respectively

($L_f = 7$). For each resolution, first the regular quantization was performed using UTQ, and then the resolution was doubled and QIM was applied to the new quantizer for a randomly generated binary watermark bitstream with $0.5 - 0.5$ probabilities.

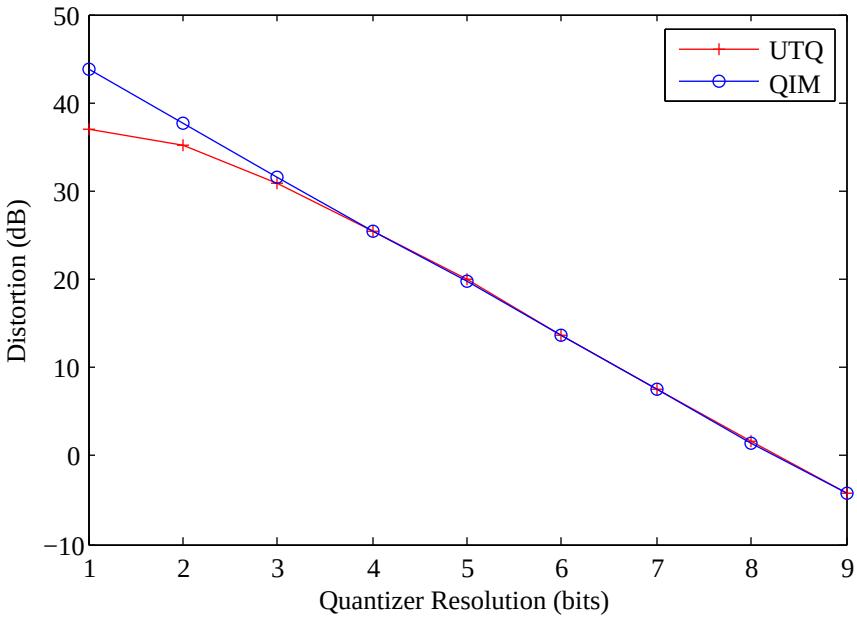


Fig. 4.4 Performance of UTQ and its equivalent QIM ($L_f = 7$)

As can be seen in the figure, the difference in performance of UTQ and its equivalent QIM is not noticeable except for the very low resolution. Appendix A shows that the probability of the watermarked bit directly affects the performance of the QIM. We will see later that, the overall RD performance is better at low resolutions, when we use QIM and consider the saving in bitrate.

Note that in performing QIM using the nearest even-integer function (NE-QIM) the dead-zone covers one interval, whereas in the nearest odd-integer QIM (NO-QIM) two adjacent intervals around zero are covered in the dead-zone. However, as can be seen in Fig. 4.4, that does not much affect the performance. Equation (4.11) gives the entropy of a UTQ in terms of the step size and offset values. When using the NE-QIM approach, the same equation can be used for obtaining the entropy since the two quantizers are exactly equivalent. When using the NO-QIM, since there are two dead-zones around zero, the entropy equation for such a quantizer slightly differs from (4.11) (see Appendix A).

Nevertheless, the same trend for the differences in entropy is observed. In fact, the weighted average entropy of the two quantizers (in NE-QIM and NO-QIM) also follow the same pattern of Fig. 4.2 if plotted.

4.4 Scalable Audio Coding using Watermarking

In a scalable audio coding based on REQ, after quantization in each layer entropy coding is performed. The entropy coding used in AAC for this scalable coder is Huffman coding which we use here as well. For each quantization resolution a Huffman codebook is built which is used by both the coder and the decoder. In these codebooks there are codewords corresponding to each quantizer reconstruction point which is found in the entropy coding process and sent to the receiver. These codewords are variable length and their average length determines the bitrate of each layer.

The main idea in our proposed method is that for each enhancement layer output we embed one bit of each codeword in the previous layer using QIM. By this approach the average bitrate of each layer is exactly decreased by unity (except the base layer which has no preceding layer):

$$\sum_i p_i(w_i - 1) = \sum_i p_i w_i - \sum_i p_i = B - 1, \tag{4.12}$$

where B is the average bitrate and w_i is the codeword lengths.

On the other hand, since we are using QIM for the previous layer, to keep the distortion unchanged we need to double the quantization resolution of that layer. This will be done for all layers except the last one for which QIM is not performed (it has no succeeding layer). However, as we showed in section 2, at low resolutions (which is typically the case for each layer of REQ scalable coding), when we double the quantization resolution its output entropy (hence the bitrate after entropy coding) is increased by less than one. Therefore, we save one bit in bitrate of a layer by watermarking in the previous layer, and at the same increase the bitrate of that layer by an amount less than one by doubling the quantization

resolution which in total leads to a savings in bitrate:

$$\begin{aligned} B' &= (B_1 + \Delta B_1) + (B_2 + \Delta B_2 - 1) + (B_3 + \Delta B_3 - 1) \\ &\quad + \cdots + (B_M - 1) \\ &= B - [(N_L - 1) - \Delta B_1 - \Delta B_2 - \cdots - \Delta B_{N_L - 1}], \end{aligned} \tag{4.13}$$

where B' is the new total bitrate using WSAC (the proposed method), B is the total bitrate using REQ, B_i is the bitrate of layer i before using QIM, N_L is number of layers and ΔB_i is the increment in its bitrate after using QIM which is less than unity. Assuming that the same resolution is used for all layers and the bitrate increment for them equals to ΔB we can write

$$B' = B - (N_L - 1)(1 - \Delta B). \tag{4.14}$$

This gives a clear equation for savings in bitrate which equals to $(N_L - 1)(1 - \Delta B)$. The more the number of layers, the more we save in bitrate.

As shown in Appendix A, the probability of the watermarked bit directly affects the performance of QIM. We will see later that we use the first bit of Huffman codewords for watermarking. In Fig. 4.5 we have plotted the probability of this bit being zero versus the quantization resolution. As it can be seen, for very low resolutions the probability is very close to 1, but as we go to high resolutions it tends to a value close to 0.4.

In fine-grain scalable coding, low-resolution quantization is performed in each layer. For instance consider scalable AAC based on REQ. In 16 kbps/layer and 32 kbps/layer coding, the resolution in each layer is between 1 to 4 bits/sample. In Fig. 4.6 we have compared the UTQ and QIM systems when the watermarked bit's probability to be zero was 0.7. From Fig. 4.5 this value is the first-bit zero probability when the quantization resolution is close to 3 bit/sample. For the QIM system we used a doubled-resolution UTQ, watermarked by a 0.7-probability random bit. The bitrate for the QIM is the average Huffman codeword length minus the one bit that we save from watermarking. As can be seen in Fig. 4.6, at low resolutions the overall RD performance in WSAC is better.

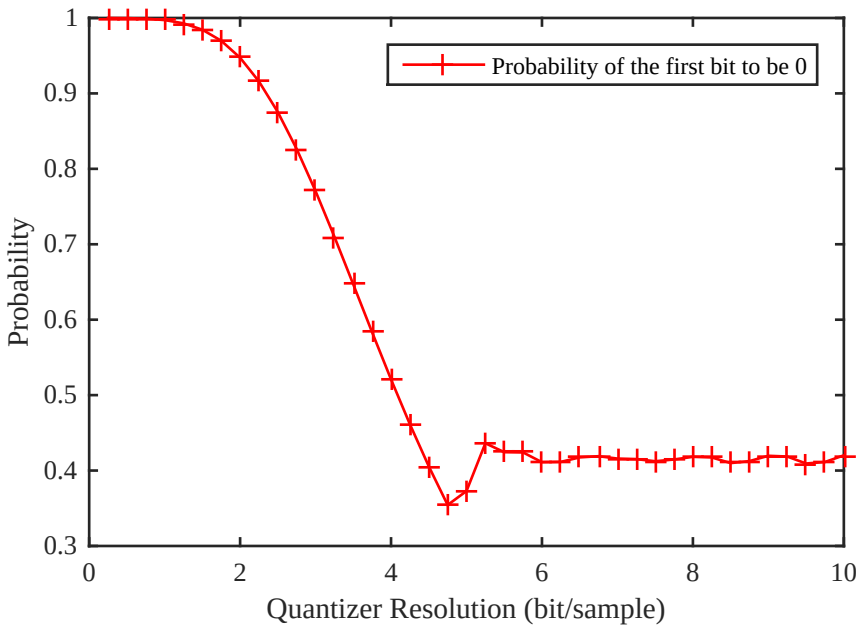


Fig. 4.5 Probability of the first bit of the Huffman codewords to be zero versus the quantization resolution. For very low resolutions the probability is very close to 1, as we go to high resolutions it tends to a value close to 0.4.

4.4.1 Coding

Consider the REQ scalable coding system (Fig. 4.1). First we perform coding based on REQ from first layer to the last. However, instead of using regular quantizers we use doubled-resolution quantizers and in each layer apply QIM to them based on nearest-even integer rule (NE-QIM). In the next step, from the last layer to the first we perform the following algorithm:

If the first bit of the output codeword for the current layer is '1', requantize the input of the previous layer using QIM based on nearest-odd integer rule (NO-QIM)and obtain the new codeword. Otherwise, keep the codeword obtained in the first step.

This algorithm is repeated for each layer. Figure 4.7 shows the block diagram of WSAC for a 3-layer WSAC. Note that the last layer remains unchanged since it does not have any succeeding layer.

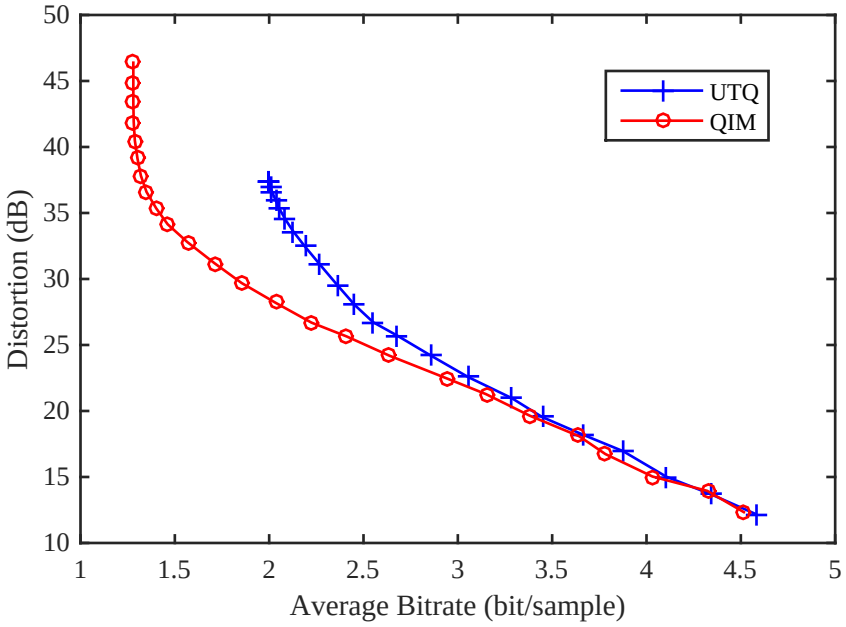


Fig. 4.6 Comparison of the UTQ and QIM systems when the watermarked bit's probability to be zero was 0.7. At low resolutions the overall RD performance in WSAC is better.

4.4.2 Decoding

The receiver starts decoding from the base (first) layer to the last layer it receives. For each layer, after decoding the variable length codeword, if the quantization index is even the first bit of the next layer output codeword is considered to be '0'. Otherwise it is '1'. This bit will be added to the beginning of the next layer's output codeword (if received).

In the REQ system, we simply add the outputs of the layers to reconstruct the original signal in a scalable manner. Here, however, a modification is required for the reconstruction pattern. Consider a 2-layer WSAC. In the first step we use NE-QIM for the first layer. In the next step, however the NO-QIM might be used for this layer depending on the second layer output. Reconstruction of the original signal at the receiver is dependent on which QIM is used for the first layer since the input of the second layer, which is the reconstruction error of the first layer, was formed in the first step of QIM in which NE-QIM was used for the layers, see Fig. 4.8. The reconstruction error resulting from NE-QIM is denoted by e_e in the figure. If the decoded quantizer index for the first layer is odd, which means NO-QIM was applied to the first layer, for reconstructing the original signal there are two

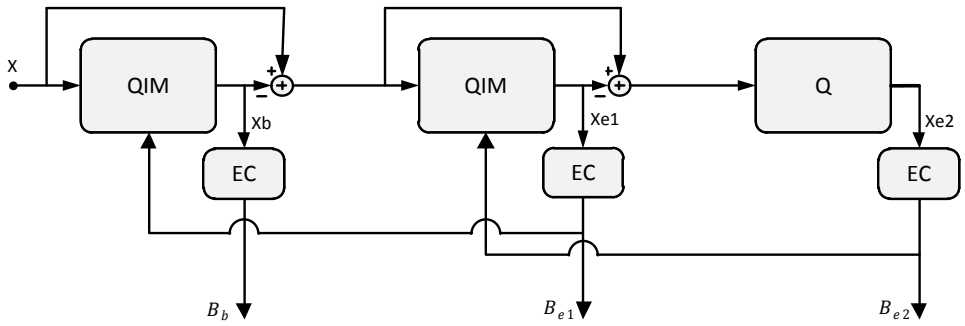


Fig. 4.7 Block diagram of a 3-layer WSAC EC is entropy coding.

possible cases: e_e is positive or negative. If e_e is positive the reconstruction formula is

$$x = \hat{x} + e_e - \Delta, \tag{4.15}$$

and for the negative case

$$x = \hat{x} + e_e + \Delta. \tag{4.16}$$

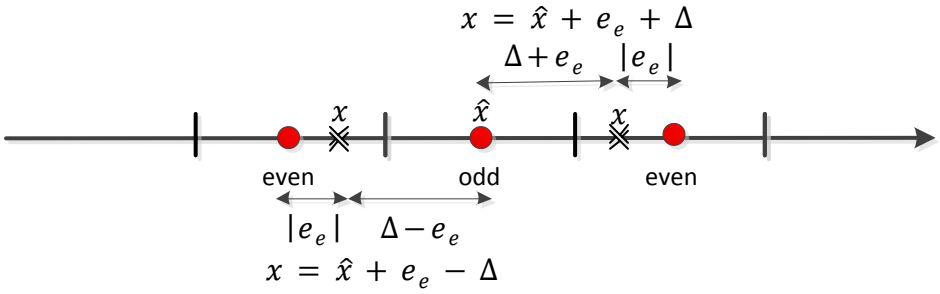


Fig. 4.8 Reconstruction scenarios in WSAC

If the decoded quantizer index for the first layer is even, reconstruction is simply performed by

$$x = \hat{x} + e_e. \tag{4.17}$$

Note that what receiver actually uses is the quantized version of e_e which is the output

of the next layer.

This is the reconstruction algorithm used for all layers in WSAC. Note that for each layer the appropriate step size Δ should be used which might be different from other layers.

4.5 Simulation and Results

Two common REQ cases were considered for our simulation. A 4-layer system with 4-bit resolution quantizers in each layer and a 5-layer system with 3-bit quantizers. For WSAC, since the resolutions are doubled, the corresponding quantizer resolutions used were 5-bit and 4-bit respectively. The quantizations were performed using AAC quantization formula. A set of Huffman coding tables were generated for different resolutions which were used for entropy coding in both REQ and WSAC. Laplacian random variables were generated as the input signal with the desired values of L_f parameter.

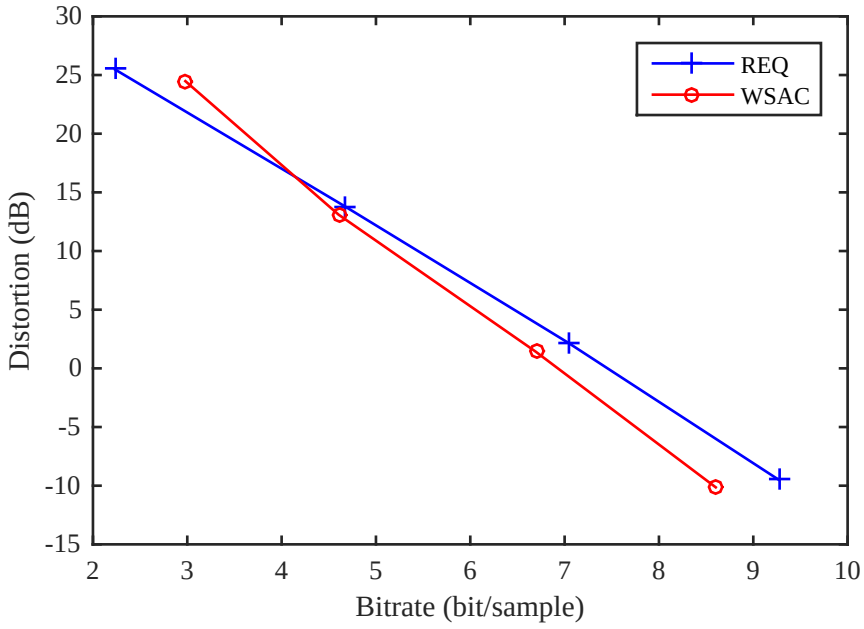


Fig. 4.9 Comparison of 4-layer WSAC and REQ ($L_f = 7$). 4-bit resolution quantizers used for REQ and 5-bit resolution quantizers for WSAC (except the 4-bit resolution quantizer used for the last layer).

Figures 4.9 and 4.10 show the comparisons between WSAC and the REQ scalable coding systems in terms of bitrate-distortion performance. In the first of the figures, the 4-layer

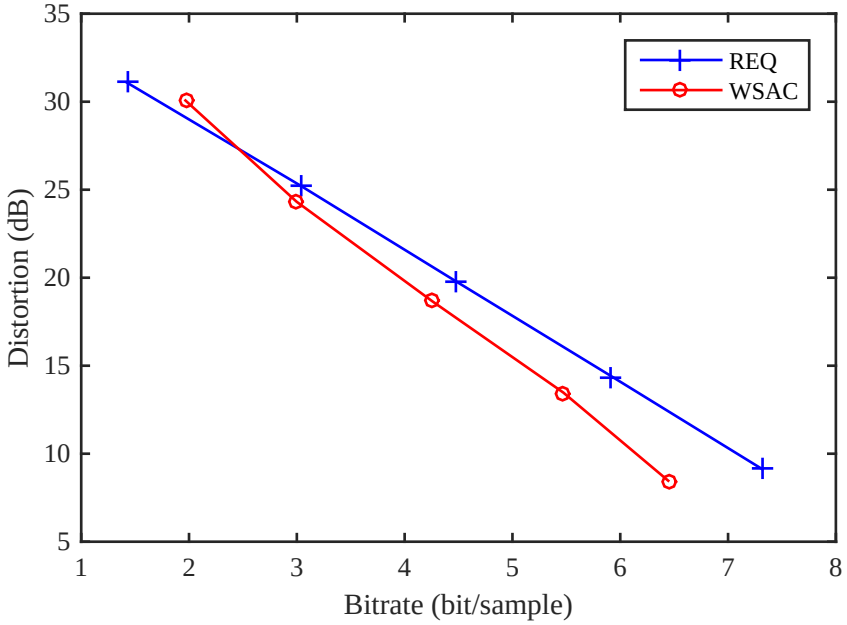


Fig. 4.10 Comparison of 5-layer WSAC and REQ ($L_f = 7$). 3-bit resolution quantizers used for REQ and 4-bit resolution quantizers for WSAC (except the 3-bit resolution quantizer used for the last layer).

systems were compared and in the second one the 5-layer systems. The points show the possible rate-distortion pairs, that is for a 1-layer system, a 2-layer system and so on. As can be seen in these two figures when the number of layers increases the performance of WSAC becomes considerably better than REQ. The more the number of layers used, the better WSAC performs compared to REQ. The only case where REQ works better is a one layer system which is obvious since in that case a double-resolution quantizer is used in the base layer while there are no more layers and hence no savings for the other layers using QIM. In fact the proposed method starts to outperform when more than one layer is sent. Also comparing the two figures reveals that in the 5-layer systems, the amount of WSAC's outperforming increases since ΔB in (4.14) is smaller for a 3-bit resolution quantizer compared to the 4-bit one.

In Fig. 4.11 the performance of the 4-layer systems were compared for $L_f = 10$. It can be seen that the performance difference between the two systems becomes larger for larger L_f or equivalently smaller input variance.

Figure 4.12 shows comparison of 4-layer S-AAC and WSAC. The average noise-to-

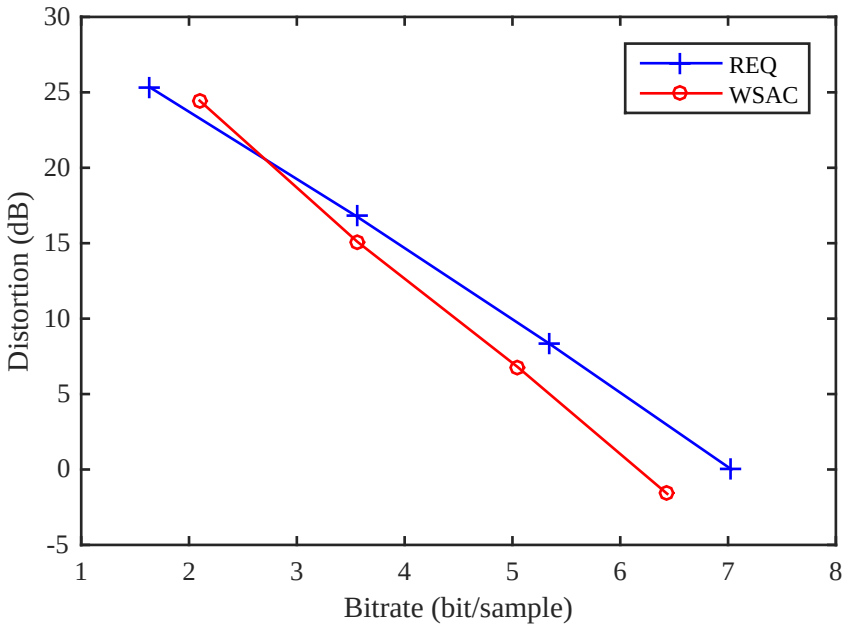
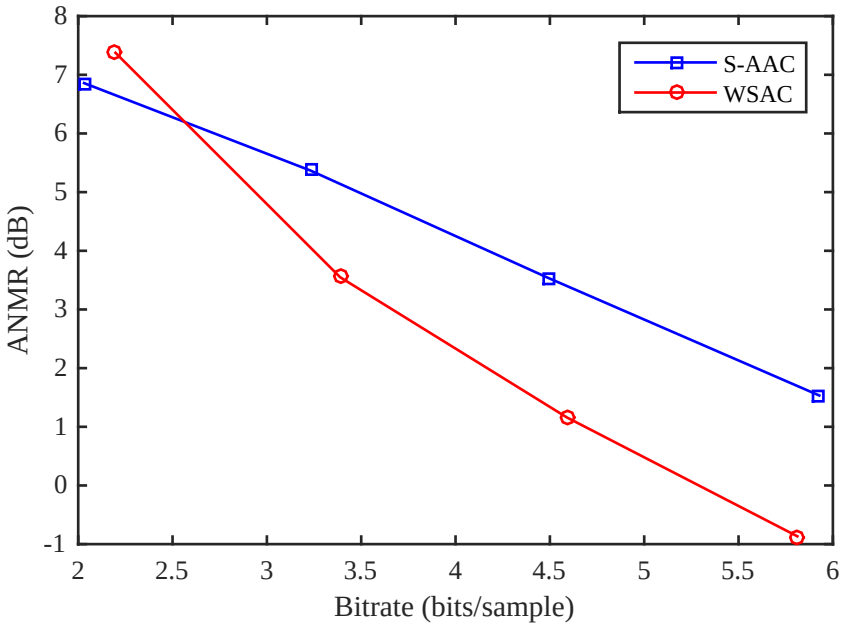


Fig. 4.11 Comparison of 4-layer WSAC and REQ ($L_f = 10$). 4-bit resolution quantizers used for REQ and 5-bit resolution quantizers for WSAC (except the 4-bit resolution quantizer used for the last layer).

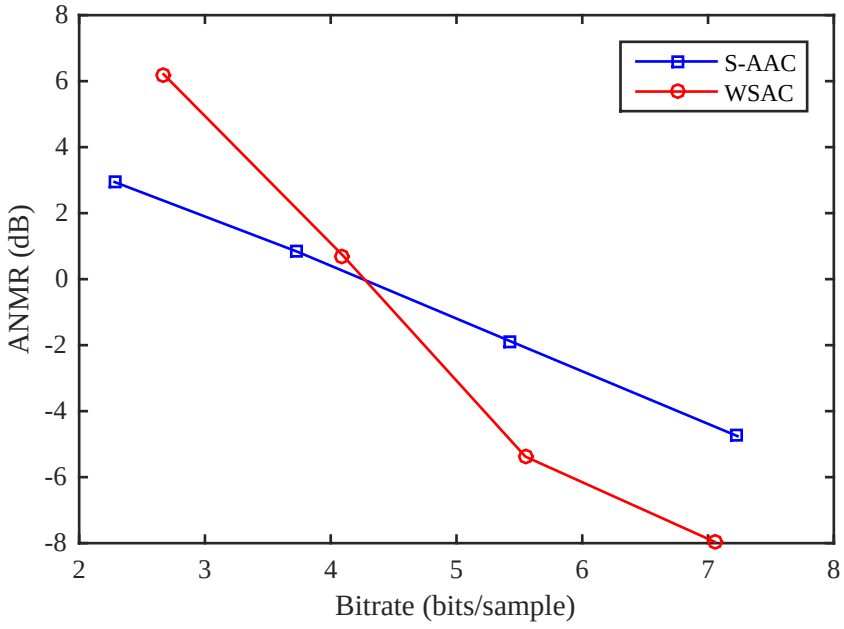
mask ratio (ANMR) values obtained for these systems were plotted vs. the bitrates in bit/sample. The bitrates were matched to the (a) 16kbps per layer and (b) 32kbps per layer in S-AAC. We will talk about this matching in more detail later in Chapter 6. Note that, for WSAC it is not straightforward to reach an exact target bitrate because of the feedback loop (Fig. 4.7) that is involved. However, the plots provide a fair comparison for RD performance. It can be seen that WSAC starts to outperform the REQ system used in S-AAC when more than one layer is used.

4.6 Summary

REQ scalable coding is a practical scalable coding scheme which is used in MPEG-4 audio coding. We proposed a modified version of such a system in which a watermarking technique known as QIM was used to embed some of the information of each layer in the previous one. The proposed method considerably outperforms REQ scalable coding in terms of rate-distortion and can be considered as a suitable replacement for practical scalable audio coders.



(a)



(b)

Fig. 4.12 Comparison of 4-layer S-AAC and WSAC. The ANMR values were plotted vs. the bitrates in bit/sample. The bitrates were matched to the (a) 16kbps/layer and (b) 32kbps/layer S-AAC. WSAC starts to outperform the REQ system when more than one layer is used.

Chapter 5

New Bit-Plane Probability Calculations for Scalable to Lossless Audio Coding

5.1 Introduction

The state of the art MPEG-4 Audio adopts two main scalable audio coding systems [98]. The first one, Scalable Advanced Audio Coding (S-AAC), is based on reconstruction error quantization of AAC coder. The second system is called MPEG-4 scalable-to-lossless (SLS) and was released as a standard audio coding tool in June 2006 [12,89]. This system has two modes: a perceptual-core mode and a non-core mode. In both modes the input signal is transformed into the frequency domain using the Integer MDCT (IntMDCT), and then the resulting coefficients are encoded. In the perceptual-core mode the coefficients are encoded using AAC perceptual coding which gives a base-quality signal. Then the residual of the coefficients are coded using a specific bit-plane arithmetic coding called bit-plane Golomb code (BPGC) [8,23]. In the non-core mode, the BPGC is applied directly to the IntMDCT coefficients.

In this chapter we will investigate the statistical properties of the residual signal for the perceptual-core mode of SLS coding and we will show that the bit-plane probabilities used by BPGC are not well matched to the residual signal. We will then propose an alternative approach which depends on the base signal and leads to better estimation of the bit-plane

properties of the residual signal. The proposed approach can also be easily extended to consider the clipping effect in bit-plane coding of an unbounded signal.

The chapter is organized as follows. In the next section, we will briefly discuss the Integer MDCT. AAC and SLS coding schemes are explained in Sections 5.3. Section 5.4 discusses bit-plane coding. In Section 5.5 the BPGC using arithmetic coding is covered. Section 5.6 derives the statistical properties of the residual signal in SLS. New bit-plane probability calculations are presented in Section 5.7, the L_p parameter is calculated in Section 5.8 and finally BPC using new probabilities is discussed in Section 5.9. The results are presented in Section 5.10 and Section 5.11 summarizes this chapter.

5.2 Integer MDCT

5.2.1 Calculating MDCT by Using DCT-IV

The MDCT for a window of signal can be calculated by applying type-IV discrete cosine transform (DCT-IV) to a Time Domain Aliased (TDA) version of the windowed signal [62, 63] which gives a new set of M coefficients where $M = N/2$ is the block size and N is the window size (Fig. 5.1). For TDA a matrix is formed from the window coefficients. This

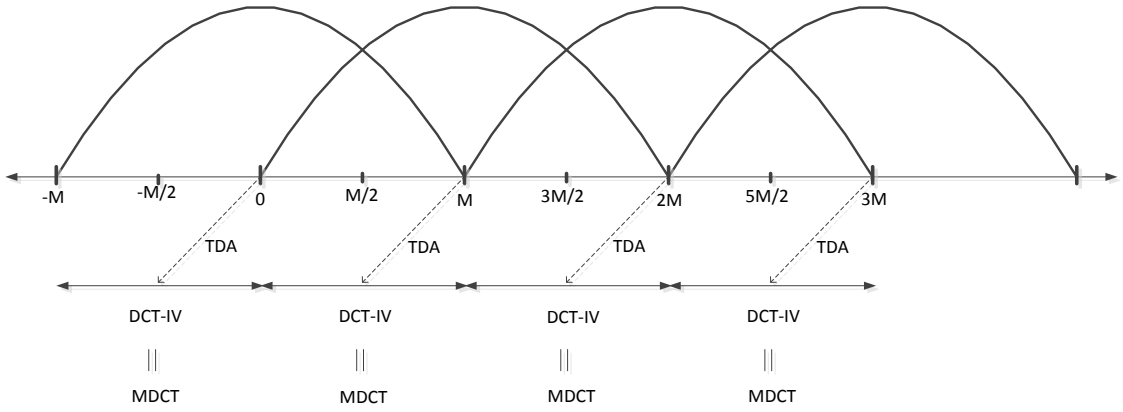


Fig. 5.1 MDCT Using DCT-IV

matrix is multiplied by each block of signal to give its TDA'ed version. Note that, the TDA'ed coefficients obtained from each block are related to the two overlapped parts of the window for that block: half ($M/2$) of these coefficients are for the current window (the window which have this block in its right part) the other half are for the next window (the

window which have this block in its left part). The DCT-IV is applied to a block of M TDA'ed coefficients in which the right-half coefficients come from the left-half of the second block of the current window and the left-half coefficients come from the right-half of the first block. The DCT-IV matrix is multiplied by a block of TDA'ed coefficients and gives the final MDCT coefficients for the corresponding window.

5.2.2 Integer Implementation

Both the TDA and DCT-IV matrices can be decomposed into lifting matrices which are used for the integer implementation. A 2×2 lifting matrix looks like

$$\mathbf{L} = \begin{bmatrix} 1 & 0 \\ k & 1 \end{bmatrix} \tag{5.1}$$

Depending on the position of the element k and the ones (which can be the diagonal or the counter diagonal elements) there can be 4 types of lifting matrices. This type of matrix enables us to use a rounding operation in the multiplication $\mathbf{y} = \mathbf{Lx}$, while making the inverse process lossless. The rounding operation is performed as

$$\begin{aligned} y_1 &= x_1 \\ y_2 &= \text{round}(k \times x_1) + x_2 \end{aligned} \tag{5.2}$$

and the inverse process as

$$\begin{aligned} x_1 &= y_1 \\ x_2 &= y_2 - \text{round}(k \times x_1) \end{aligned} \tag{5.3}$$

5.3 AAC and SLS Coding

In the perceptual-core mode of MPEG-4 SLS coding, the IntMDCT coefficients are quantized in each scalefactor band (SFB) of a data frame using AAC quantization operation. This operation is presented here again with more detail as [3]

$$i_n[l] = \text{sgn}(c_n[l])\text{nint}(|2^{-s[n]/4} c_n[l]|^{3/4} - 0.0946), \tag{5.4}$$

where $i_n[l]$ is the l^{th} quantized coefficient index in band n, $c_n[l]$ is the corresponding IntMDCT coefficient, and $s[n]$ is the scalefactor used for that band. The nint() and sgn() operations denote the nearest integer and signum functions and 0.0946 is an offset value which is also referred to as the magic number. The equation can also be expressed as

$$i_n[l] = \mathrm{Q}(g[n]\ c_n[l]^{3/4}), \tag{5.5}$$

where

$$Q(x) = \mathrm{sgn}(x) \times \mathrm{nint}(|x| - 0.0946), \tag{5.6}$$

and

$$g[n] = 2^{-s[n]/4}. \tag{5.7}$$

The integer scalefactors $s[n]$ are obtained using a psychoacoustic model and control the Noise-to-Mask ratios (NMR) in the scalefactor bands. In the next step, the residue of the IntMDCT coefficients is calculated by forming the quantization error. The residual in each SFB are then coded using BPGC. The subtraction process for obtaining the residual is however different from a typical error mapping: Instead of subtracting the quantized coefficients from the original ones, the difference between an input (IntMDCT) coefficient and its corresponding quantization interval lower threshold is considered to be the error. The thresholds are obtained using

$$\mathrm{thr}(i_n[l]) = \begin{cases} \mathrm{sgn}(i_n[l])(2^{s[n]/4}|i_n[l] - 0.4054|^{4/3}), & i_n[l]) \neq 0 \\ 0, & i_n[l]) = 0, \end{cases} \tag{5.8}$$

where $\mathrm{thr}(i_n[l])$ is the interval lower threshold for the quantized coefficient index $i_n[l]$ in SFB n. The residual signal $r[l]$ is then obtained as

$$r_n[l] = \begin{cases} c_n[l] - \lfloor \mathrm{thr}(i_n[l]) \rfloor, & i_n[l] \neq 0 \\ c_n[l], & i_n[l] = 0, \end{cases} \tag{5.9}$$

where $\lfloor . \rfloor$ is the floor function. BPGC is then applied to this residual signal in each SFB which forms a fine-grain SLS coding pattern.

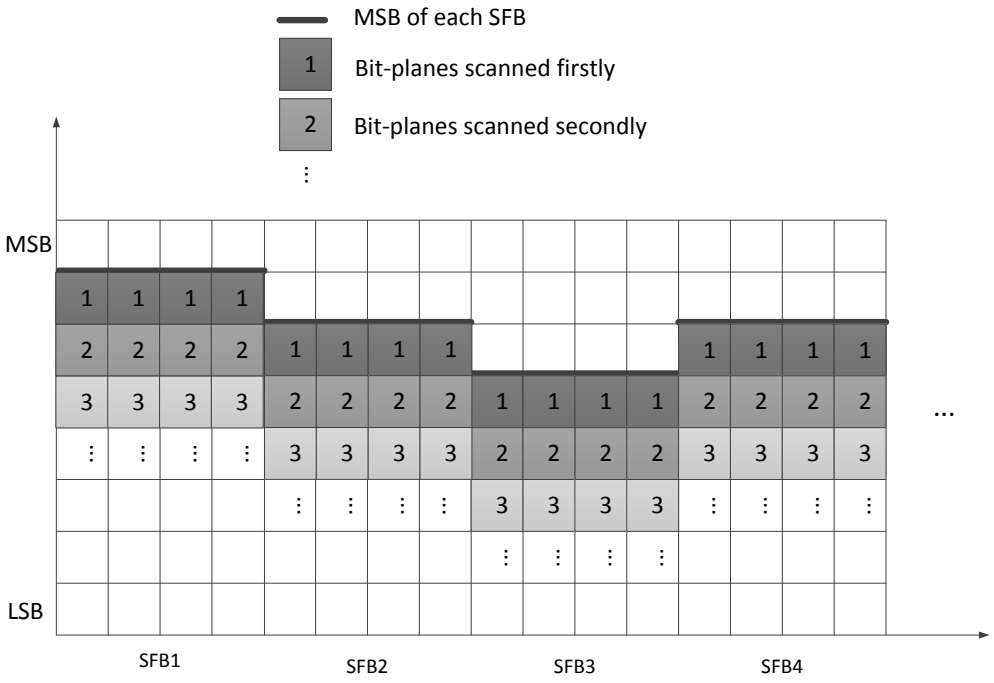


Fig. 5.2 Bit-plane coding of the residual signal. In each SFB, the bit-planes are scanned from the M_nth plane to the LSB and are coded using arithmetic coding plane by plane.

5.4 Bit-Plane Coding

Figure 5.2 shows the bit-plane coding for the residual signal. The figure shows the first 4 SFBs in each of which there are four residual coefficients to be encoded. The number of coefficients per SFB is not constant and is larger in the higher frequency bands. The magnitude of the residual coefficients in SFB n is expressed in a M_n-bit binary format as

$$|r_n[l]| = \sum_{j=0}^{M_n} b[l,j]2^j \tag{5.10}$$

where M_n is the most significant bit-plane to be coded in SFB n (indicated by the bold line in Fig. 5.2). The parameter M_n is calculated in the encoder based on the range of the error in each SFB satisfying

$$2^{M_n-1} \leq \max\{|r_n[l]|\} < 2^{M_n} \tag{5.11}$$

In each SFB, the bit-planes are scanned from the M_nth plane to the LSB and are entropy-coded using arithmetic coding plane by plane.

5.5 BPGC Using Arithmetic Coding

In BPGC, as explained in the previous section, each bit-plane of the coefficients is entropy-coded using arithmetic coding [12]. The arithmetic coding uses a probability assignment rule that is derived from the statistical properties of an exponentially distributed source [8]. Closed forms have been obtained in [8] for bit-plane probabilities of a Laplacian source which were adopted in BPGC [23]. In [12] it was shown that the IntMDCT coefficients can be well modelled from a Laplace distribution. Based on this assumption, it is assumed that the magnitude of the residual signal can be approximately considered an exponential source. Therefore, the probabilities obtained for Laplacian source are used for the residual signal as well.

For each SFB the standard deviation (or λ parameter) of the residual signal is estimated by the mean of the signal. The Lambda parameters are then used in the closed-forms obtained in [8] to give the probabilities. The bit-planes are scanned from MSB to LSB and coded using arithmetic coding. In the case of non-core SLS coding, where there is no perceptual core, the bit-plane coding is applied directly to the input IntMDCT coefficients instead of the residual signal. In both cases if all the bit-planes are received by the decoder, lossless quality is achieved. However, by truncating the bit-planes a fine-grain scalable lossy-to-lossless (SLS) coding is obtained.

In the next section we will discuss the properties of the residual signal and we will address some issues regarding the bit-plane probabilities used in BPGC.

5.6 Statistical Properties of The Residual Signal

Consider Fig. 5.3. A specific quantization interval was shown for $t_i \leq c < t_i + 1$, where t_i is the beginning threshold of that interval. The pdf of a Laplacian source is given by

$$f_X(x) = \frac{1}{2}\lambda e^{-\lambda|x|}, \tag{5.12}$$

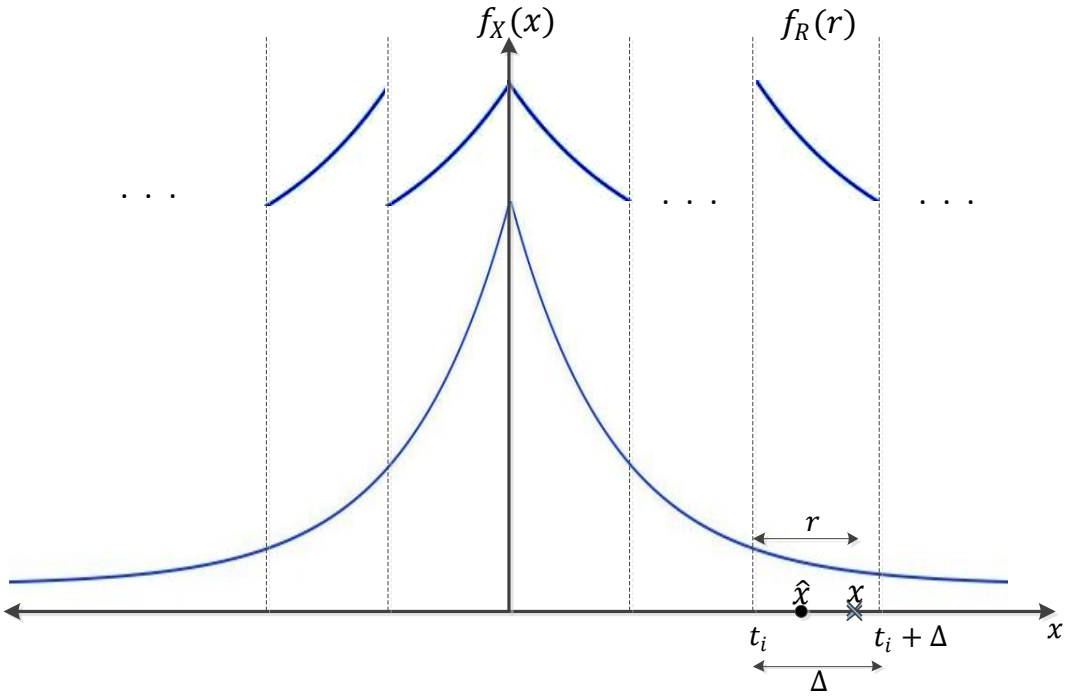


Fig. 5.3 pdf properties of the residual signal

where $\lambda = \sqrt{2}/\sigma$ and σ is the standard deviation of the signal. In BPGC the absolute value of the residual signal is coded and there is a sign bit which is sent separately. In fact, the one sided non-negative signal is considered which has an exponential pdf in the form of

$$f_X(x) = \begin{cases} \lambda e^{-\lambda x}, & x \geq 0 \\ 0, & x<0 \end{cases} \tag{5.13}$$

The probability of such an exponential signal to be in a specific interval $p_i = p(t_i \leq x < t_{i+1})$ can be obtained by

$$p_i = (e^{-\lambda t_i} - e^{-\lambda t_{i+1}}). \tag{5.14}$$

Considering $\Delta = t_{i+1} - t_i$, we have

$$p_i = e^{-\lambda t_i}(1 - e^{-\lambda \Delta}). \tag{5.15}$$

The residual for this interval is $r = x - t_i$ and we have

$$f_X(x|t_i{\leq}x{<}t_{i+1}) = \frac{f_X(x)}{p_i} = \frac{\lambda e^{-\lambda x}}{e^{-\lambda t_i}(1 - e^{-\lambda\Delta})}. \tag{5.16}$$

The conditional pdf of the residual in this specific interval is

$$f_R(r|t_i{\leq}x{<}t_{i+1}) = \left.\frac{f_X(x|t_i{\leq}x{<}t_{i+1})}{r'(x)}\right|_{x=t_i+r} = \frac{\lambda e^{-\lambda r}}{1 - e^{-\lambda\Delta}}. \tag{5.17}$$

If we use a uniform quantizer, the pdf of the residual for such a quantizer would be

$$\begin{aligned} f_R(r) &= \sum_i p_i f_R(r|t_i{\leq}x{<}t_{i+1}) = \frac{\lambda e^{-\lambda r}}{1 - e^{-\lambda\Delta}} \sum_i p_i \\ &= \frac{\lambda e^{-\lambda r}}{1 - e^{-\lambda\Delta}}, \end{aligned} \tag{5.18}$$

which is equal to the conditional pdf obtained for a specific interval due to the memoryless property of the Laplace distribution. It is very important to note that this pdf is not in the typical form of an exponential source (5.13). The reason is that the residual signal is a signal bounded within the interval ($0{\leq}r{<}\Delta$). Now, let us calculate the mean of the residual signal. The mean can be obtained by

$$\bar{r} = E[r] = \int_0^{\Delta} r f_R(r)\, dr = \frac{1}{\lambda} - \frac{\Delta}{e^{\lambda\Delta} - 1}. \tag{5.19}$$

which is also the same for the conditional expectation of an interval of width Δ. It can be seen that the residual mean is different from that of an exponential signal (for which mean= $1/\lambda$). In a special case where $\Delta \to \infty$ (unbounded exponential), $\bar{r} = 1/\lambda$. From Equations (5.18) and (5.19) it is clear that assuming an exponential source for the residual signal and obtaining the pdf parameters using the statistics (mean) of the observed residual signal leads to inaccurate bit-plane probabilities: In BPGC the λ (or equivalently L) parameter for the residual is approximated by $1/\bar{r}$ in each SFB. On the other hand, the pdf of the residual is directly dependent on the quantization step sizes. The other issue is that the step sizes are not equal in AAC quantization. Having the interval thresholds from (5.8)

the step sizes can be obtained by

$$\Delta_n(i) = t_n(i+1) - t_n(i), \tag{5.20}$$

where $\Delta_n(i)$ is the step size of a quantization index i for SFB n and $t_n(i)$ and $t_n(i+1)$ are the corresponding interval thresholds. It can be seen that the the step sizes are dependent not only on the scalefactor parameter used in each SFB, but also on the quantization interval index. Therefore the bit-plane probabilities should be a function of the input (IntMDCT coefficients) statistics, together with the scalefactors and quantization indices. In the following we will propose a straightforward approach for calculating the bit-plane probabilities of an exponential source bounded within an arbitrary interval.

5.7 Bit-Plane Probability Calculations

Consider a signal bounded within the interval $0 \leq x \leq \Delta$ (see Fig. 5.4). We want to obtain the N most significant non-zero bits of this signal. The MSB of this signal b_1 (we index the bit-planes from MSB to LSB) can be considered to be zero if the signal is $0 \leq x < \Delta/2$, and one if $\Delta/2 \leq x < \Delta$. The probability of $b_1 = 0$ then can be obtained by

$$P_1 = \frac{p(0 \leq x < \Delta/2)}{p(0 \leq x < \Delta)}, \tag{5.21}$$

and $b_1 = 1$ with the probability of $Q_1 = 1 - P_1$. Now consider the second bit b_2. The probability of $b_2 = 0$ can be written as (Fig. 5.4)

$$\begin{aligned} P_2 = &\frac{p(0 \leq x < \Delta/4)}{p(0 \leq x < \Delta/2)} p(0 \leq x < \Delta/2) \\ &+ \frac{p(\Delta/2 \leq x < 3 \times \Delta/4)}{p(\Delta/2 \leq x < \Delta)} p(\Delta/2 \leq x < \Delta). \end{aligned} \tag{5.22}$$

Due to the memoryless property of the Laplace distribution the two fractions in the above equation are equal. Also, since $p(0 \leq x < \Delta/2) + p(\Delta/2 \leq x < \Delta) = 1$ we get

$$P_2 = \frac{p(0 \leq x < \Delta/4)}{p(0 \leq x < \Delta/2)}. \tag{5.23}$$

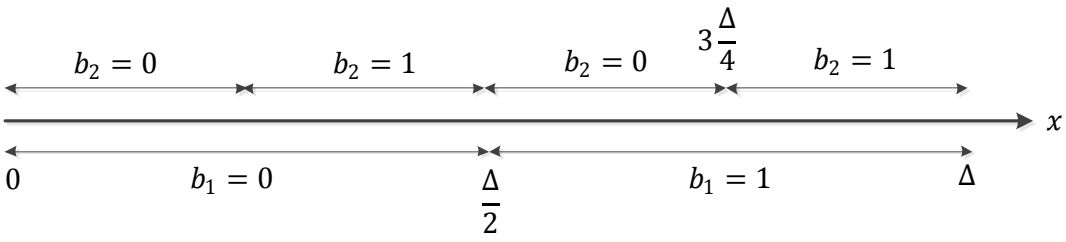


Fig. 5.4 Obtaining the N most significant non-zero bits of a bounded signal

Using the same strategy the kth significant bit can be obtained by

$$P_k = \frac{p(0 \leq x < \Delta/2^k)}{p(0 \leq x < \Delta/2^{k-1})}. \tag{5.24}$$

Using (5.15) in (5.24) gives

$$P_k = \frac{1 - e^{\frac{-\lambda\Delta}{2^k}}}{1 - e^{\frac{-\lambda\Delta}{2^{k-1}}}} = \frac{1}{1 + e^{\frac{-\lambda\Delta}{2^k}}}, \tag{5.25}$$

and

$$Q_k = 1 - P_k = \frac{1}{1 + e^{\frac{\lambda\Delta}{2^k}}}. \tag{5.26}$$

Equations (5.25) and (5.26) give the bit-plane probabilities for an exponential signal bounded in the interval Δ.

Now let us consider a case where we have an unbounded exponential signal. In general, for a binary representation of such a signal an infinite number of bits are required. However, in practice the signal is bounded before coding, which means clipping happens. Assume the signal is bounded to be $0 \leq x \leq X_m$, and let us define $P_e = p(x > X_m) = e^{-\lambda X_m}$. Using a similar strategy used in (5.22) we can obtain

$$\begin{aligned} P_k^c &= P_k'(1 - P_e), \\ Q_k^c &= Q_k'(1 - P_e) + P_e, \end{aligned} \tag{5.27}$$

where the superscript c stands for the clipping consideration and

$$\begin{aligned} P'_k &= \frac{1}{1+e^{-\frac{\lambda X_m}{2^k}}}, \\ Q'_k &= \frac{1}{1+e^{\frac{\lambda X_m}{2^k}}}. \end{aligned} \tag{5.28}$$

5.8 Calculating the L_p Parameter

In the practical coder, a parameter L_p is calculated which makes the probability implementation easier. The frequency (probability) assignment rule used in the MPEG-4 SLS standard is expressed in terms of L_p. Following the same strategy used in the standard, here we can obtain a new L_p based on the new probabilities. In (5.25) the bit-plane index k goes from MSB to LSB. Note that the binary representation for the residual in SLS is obtained as if the residual is an unbounded signal. The way the we defined the bit-plane indices from MSB to LSB here is based on the fact that the signal is bounded within an interval which is more precise as seen above. This binary representation can be equivalently obtained by a scaling followed by a typical representation which can be expressed as

$$\mathbf{r}_b = \text{bin}[r \times (\frac{2^M}{\Delta})] \tag{5.29}$$

where bin(.) is the binary representation function and

$$M = \lfloor \log_2 \Delta \rfloor + 1, \tag{5.30}$$

$\mathbf{r}$ is a vector of binary symbols and $1 \leq \frac{2^M}{\Delta} < 2$ depending on the Δ value. If we want a reversed order for the bit-plane indices to be consistent with the standard (going from LSB to MSB), the new probabilities can be expressed as

$$P_k = \frac{1}{1+e^{\frac{-\lambda\Delta}{2^{M-k}}}} = \frac{1}{1+e^{-(\frac{\lambda\Delta}{2^M})2^k}} \tag{5.31}$$

Now let us define the new L_p as

$$L_p = \lfloor \log_2 \frac{2^M}{\lambda\Delta} \rfloor = \lfloor \log_2 \frac{1}{\lambda\Delta} + M \rfloor. \tag{5.32}$$

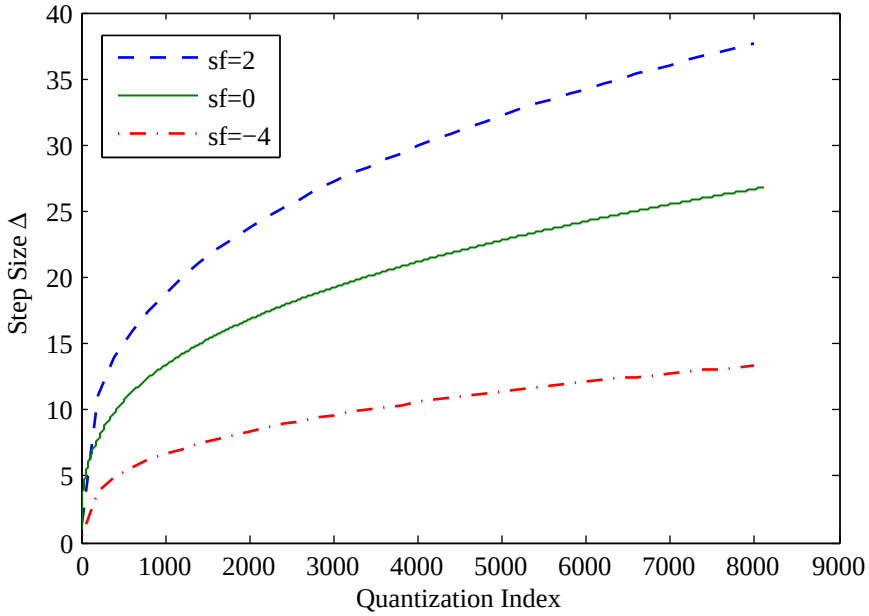


Fig. 5.5 AAC quantizer step size vs. quantization index for three different scalefactor values. Increasing the scalefactor increases the step sizes. Also, due to the companding used in the quantization, for each specific scalefactor the step sizes increase by going to the higher indices.

5.9 Bit-Plane Coding Using the New Probabilities

We saw in the previous section that, bit-plane coding of the residual signal should be performed considering the quantization parameters used in the core layer. In other words the probabilities are a function of the scalefactors and quantization indices. Figure 5.5 shows step sizes of the AAC quantizer versus quantization indices (which are in the range of $(0 \leq |i| \leq 8191)$ for three different scalefactor values. It can be seen that increasing the scalefactor increases the step sizes. Also, due to the compression used in the quantization, for each specific scalefactor the step sizes increase by going to the higher indices. Therefore, the bit-plane probabilities obtained in (5.25) can be expressed by

$$P_k(s,i) = \frac{1}{1 + e^{-\frac{\lambda \Delta(s,i)}{2^k}}}, \tag{5.33}$$

where s and i stand for the scalefactor and quantization index respectively. $\Delta(s,i)$ can be obtained from (5.20) and (5.8). Having this set of probabilities, the bit-plane coding can

be performed using arithmetic coding based on Algorithm 1. This is a modified version of the algorithm proposed in [8], where 1) instead of the residual we use the input (IntMDCT) coefficients to approximate the λ parameter, 2) the new set of core-based probabilities are used instead. A similar algorithm can be used for non-core bit-plane coding of the input (IntMDCT) coefficients, considering the clipping effect. In this case the line regarding the core parameters should be omitted and equations in (5.27) should be used for bit-plane probabilities.

Algorithm 1: Core-based Residual Bit-plane Coding

- Take input IntMDCT coefficients $c_n[l]$.
- Calculate the λ parameter for each scalefactor band n using $\frac{1}{\lambda} = |\bar{c}_n| = (\sum_l |c_n[l]|)/L_p(n)$.
- Take the scalefactors and quantization indices from the core layer.
- Calculate the bit-plane probabilities using (5.33).
- Scan from MSB plane to LSB plane and perform arithmetic coding for bit-plane symbols using the obtained probabilities).

5.10 Simulation and Results

We compare our proposed core-based method for obtaining the residual signal bit-plane probabilities (referred to as CoreBPP) with the method proposed in [8], which is used for BPGC in SLS coding. We took the IntMDCT coefficients of the same 15 audio files used in [12] as input signal. These coefficients were quantized using (5.7) for different step sizes. The bit-plane probabilities of the residual signals were computed using the two methods and were compared with what obtained experimentally (ExpBPP). Experiments showed that for $\lambda\Delta = \sqrt{2}\Delta/\sigma < 10$ the new method leads to clearly better results. In fact, for smaller $\lambda\Delta$ values the difference between the methods becomes greater. Such a comparison is shown in Fig. 5.6 for a sample case of $\lambda\Delta = 2.56$. Note that, when a Laplacian source is considered as input, the CoreBPP and ExpBPP probabilities become much closer.

Also, Fig. 5.7 shows a comparison of bit-plane probabilities of the input signal (IntMDCT coefficients) for $\lambda X_m = 3$ when 1) clipping is considered using (5.27) and referred to as ClipBPP, 2) without considering clipping [8]. The experiments showed that for

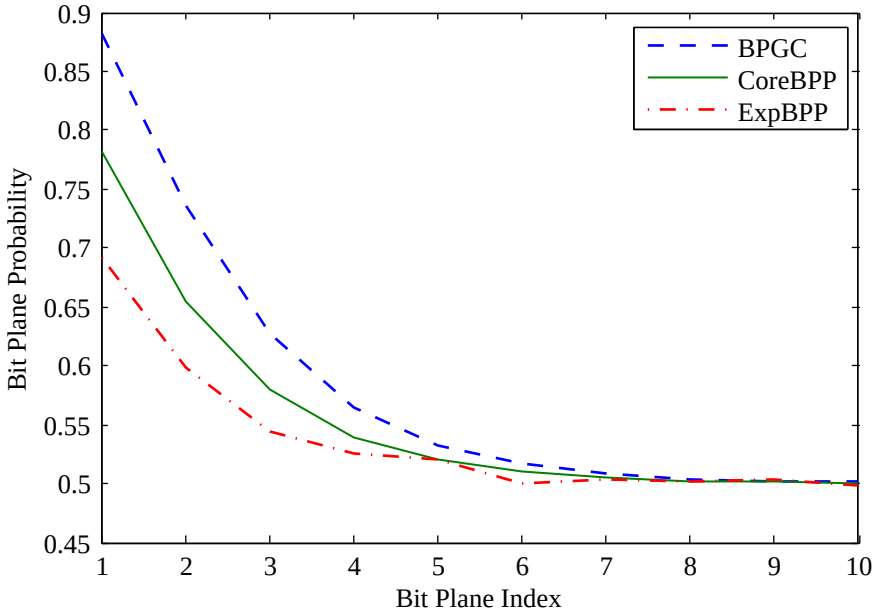


Fig. 5.6 Bit-Plane probability vs. bit-plane index for the residual signal ($\lambda\Delta = 2.56$). The bit-plane probabilities of the residual signals were computed using BPGC and CoreBPP and were compared with what obtained experimentally. For smaller $\lambda\Delta$ values the difference between the methods becomes greater.

$\lambda X_m \leq 4$ the difference between the two methods become considerable.

The new bit-plane probabilities were applied to the SLS reference software. The L_p calculation procedure was modified according to the new probabilities. Figure 5.8 compares SLS and the modified SLS (shown as MSLS in the figure) in terms of the Objective Difference Grade (ODG) given by PEAQ (Perceptual Evaluation of Audio Quality [99]) as implemented in [100]. PEAQ is a standardized algorithm for objectively measuring perceived audio quality. The ODG ranges from -4 (very annoying noise) to 0 (imperceptible distortion). The comparison here is for a commonly used case where the core is AAC working at 64 kbps and the SLS (enhancement) layers are coded at 32 kbps steps. As can be seen, MSLS leads to better ODG values compared to SLS. The performance difference is higher at lower bitrates and as we go to the higher bitrates the two systems tend to the transparent quality of $ODG = 0$.

Note that, the maximum performance difference is about the difference we get by adding an extra layer to SLS at high rates and about half of that at lower rates. Although the difference in ODG is small at some rates, the overall comparison shows that using MSLS

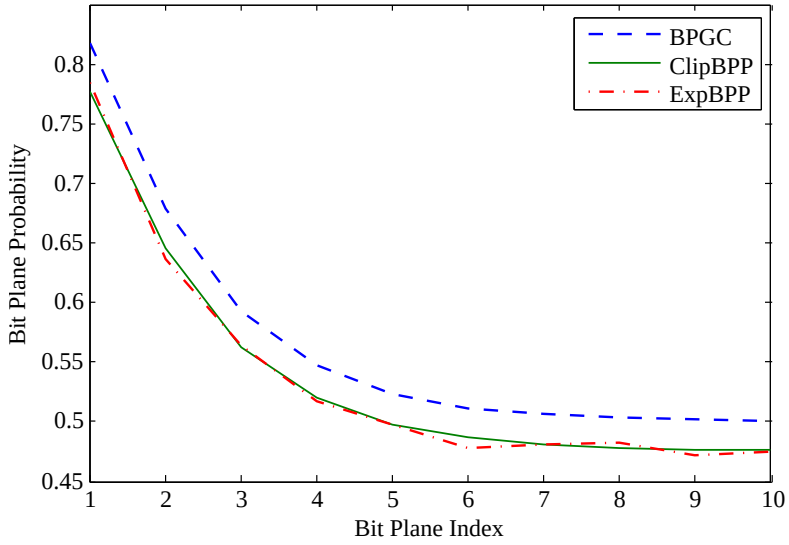


Fig. 5.7 Comparison of bit-plane probabilities of the input IntMDCT coefficients for $\lambda X_m = 3$ when 1) clipping is considered (ClipBPP) and 2) without considering clipping (BPGC). For $\lambda X_m \leq 4$ the difference between the two methods becomes considerable.

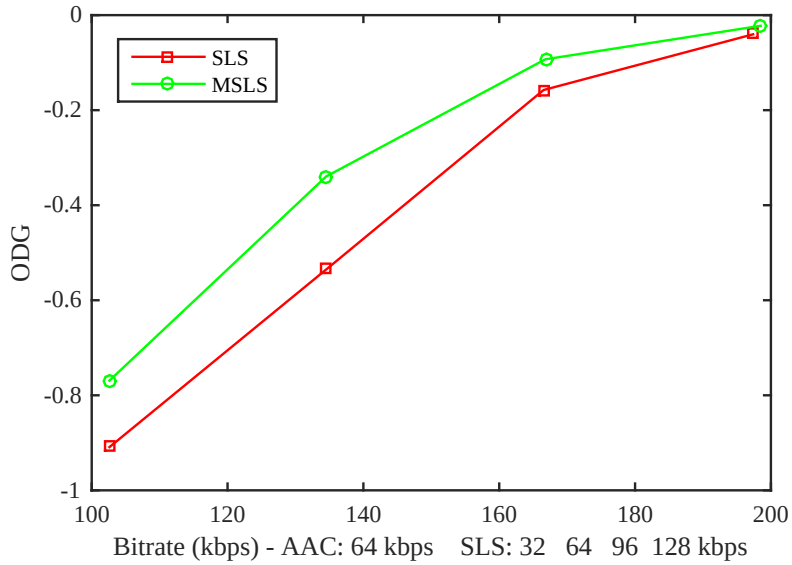


Fig. 5.8 Comparison of SLS and the modified SLS (MSLS). The comparison is for a commonly used case where the core is AAC working at 64 kbps and the SLS layers are coded at 32 kbps steps. MSLS leads to better ODG values compared to SLS.

is beneficial, specially when no additional complexity is introduced.

5.11 Summary

An alternative approach was proposed for calculating the bit-plane probabilities of the residual signal for perceptual-core mode of MPEG-4 SLS coding. The new approach matches the quantization performed in the core layer. Therefore, the probabilities are estimated considerably better for the residual signal compared to what used in BPGC. Also, the clipping effect can be easily considered in the proposed method, which is useful for coding an un-bounded exponential.

Chapter 6

Fine-Grain Scalable Audio Coding by Trellis-Based Optimized Scalable Entropy Coding

6.1 Introduction

It has been shown that there is a considerable performance gap between MPEG-4 scalable coding schemes and a non-scalable AAC coder operating at the same bitrate [24]. This sub-optimality results from the fact that in scalable AAC the layers are coded independently. This leads to a scalability penalty which becomes bigger at higher bitrates. One of the key facts behind the sub-optimality of S-AAC, which has not been dealt with previously is the entropy coding which is performed separately for each layer (Fig. 6.1). In practical coding systems, including S-AAC, the coefficients are quantized using an entropy-constrained quantizer. The quantized coefficients are then encoded in a bitstream using an entropy coding scheme. This entropy coding has an overhead for each layer. In S-AAC, where Huffman coding is performed as entropy coding, the upper bound of the bitrate for each layer is the entropy of the symbols plus one bit; thus the upper bound of the combined layers is the entropy of all symbols combined plus N bits, where N is the number of layers. For instance, if the sample rate is 48kHz and a 4-layer S-AAC is used, this overhead leads to up to 192 kbps difference in the total bitrate.

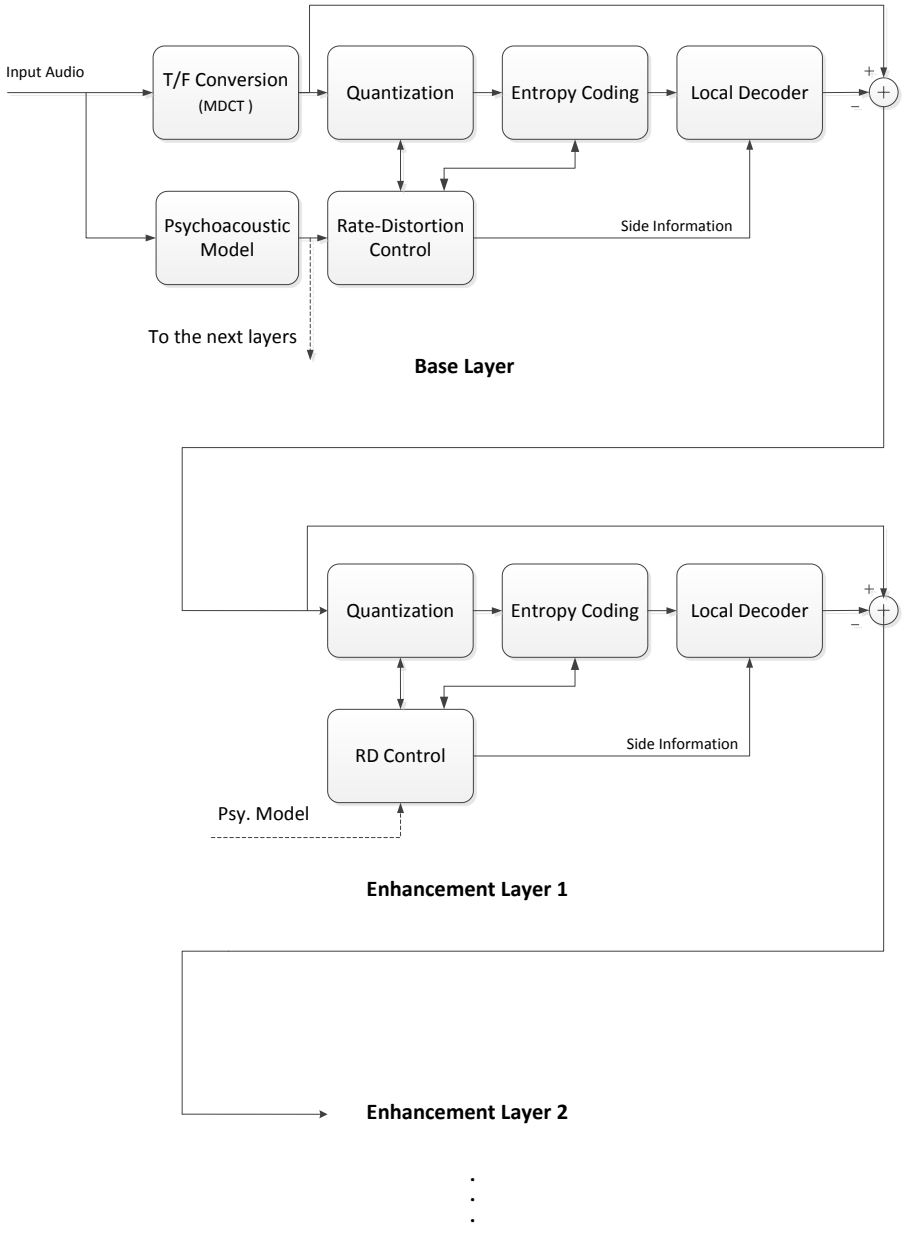


Fig. 6.1 Salable Audio Coding in S-AAC. The base layer performs AAC and gives a minimum bit rate with acceptable distortion. The coded audio is then subtracted from the original input using a local decoder. The residual signal is passed to the next layer where it is encoded using AAC again and forms the first enhancement layer. This enhancement procedure is repeated to as many as layers required.

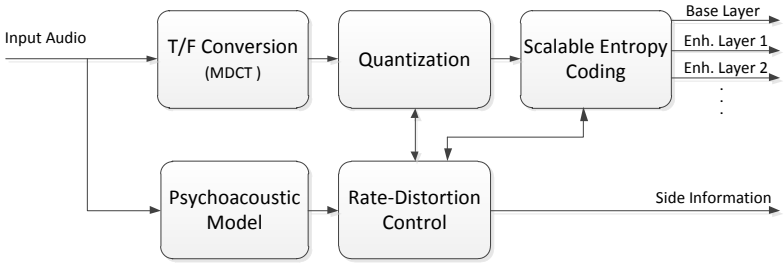


Fig. 6.2 Block diagram of the proposed scalable coding approach. Instead of performing separate and independent quantization and entropy coding at each layer, there is only one single full quantizer and the entropy coding is performed in a scalable manner.

In this chapter we propose a fine-grain scalable coding system where, the entropy coding is performed in a scalable manner for a single full quantizer with the highest resolution (Fig. 6.2). In the proposed method, a Huffman-like coding tree is created where internal nodes are mapped to the quantization reconstruction points. The tree can be pruned at any internal node to control the rate-distortion performance of the encoder. The proposed approach was partly presented in [25], where we introduced joint entropy-scalable coding of signals improving [101] and established an infrastructure for such a coding system for a general quantizer with arbitrary intervals and reconstruction points distribution. In that work, a simple and memoryless merging criterion was used for forming the coding tree. It is shown here that, although even that simple measure leads to performance improvement over S-AAC, there is still a significant performance gap compared to a non-scalable coding system. Here, we propose new set of measures and a trellis-based approach to keep track of merges in creating the coding tree. The proposed approach seeks to create the best possible set of pruned trees.

This chapter is organized as follows. In the next two sections, we will briefly discuss the Huffman coding and its redundancy issues. Bases of the proposed approach for creating the scalable coding tree are presented in Sections 6.4, 6.5 and 6.6. Section 6.7 discusses the proper measures for the proposed approach. In Section 6.8 the trellis-based approach is explained and Section 6.9 implements the approach by defining a function of the proposed measures as the cost function. The results are presented in Section 6.10 and Section 6.11 summarizes the chapter.

6.2 A Quick Review of Huffman Coding

Huffman coding is one of the most commonly used entropy coding techniques. The codebook is generated by assigning symbols to leaves in an unbalanced binary tree. To build the coding tree, nodes are created by merging either leaves or nodes until all leaves are part of a single tree. In the Huffman coding process (See Figure 6.3), we start merging with the two nodes which have the smallest probabilities in the set and keep on merging with the same pattern to the end. After creating the Huffman tree what we finally use is the end nodes of the tree for each of which a codeword has been assigned. There is no use of the internal nodes because of the "Unique Prefix Property" of the coding. The property requires that: *no code is a prefix to any other code.*

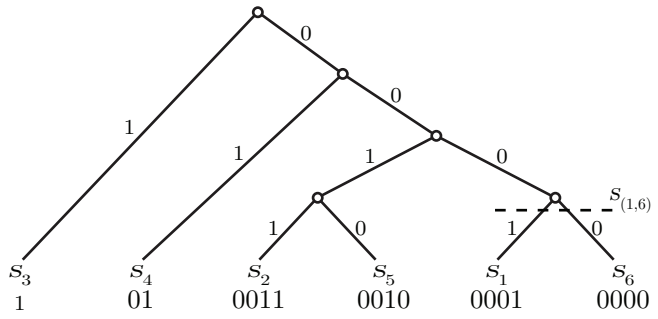


Fig. 6.3 Huffman coding tree

6.3 Comments on Huffman Coding Redundancy

In general it is well known that the Huffman coding redundancy is less than the unity. However, there have been publications on finding tight bounds for this redundancy including [102]. Clearly, the bounds obtained from any approaches always depend on the statistics of the source.

In Fig. 6.4 we have plotted these bounds for a Laplacian source based on the formulation presented in [102]. In this figure, the horizontal axis shows the quantization resolution which changes in the steps of 1.5 dB (0.25 bits/sample). The plots include a UTQ quantizer output entropy for a Laplacian source, the average codeword length of the quantizer output after Huffman coding, the upper and lower redundancy bounds from [102], and the actual

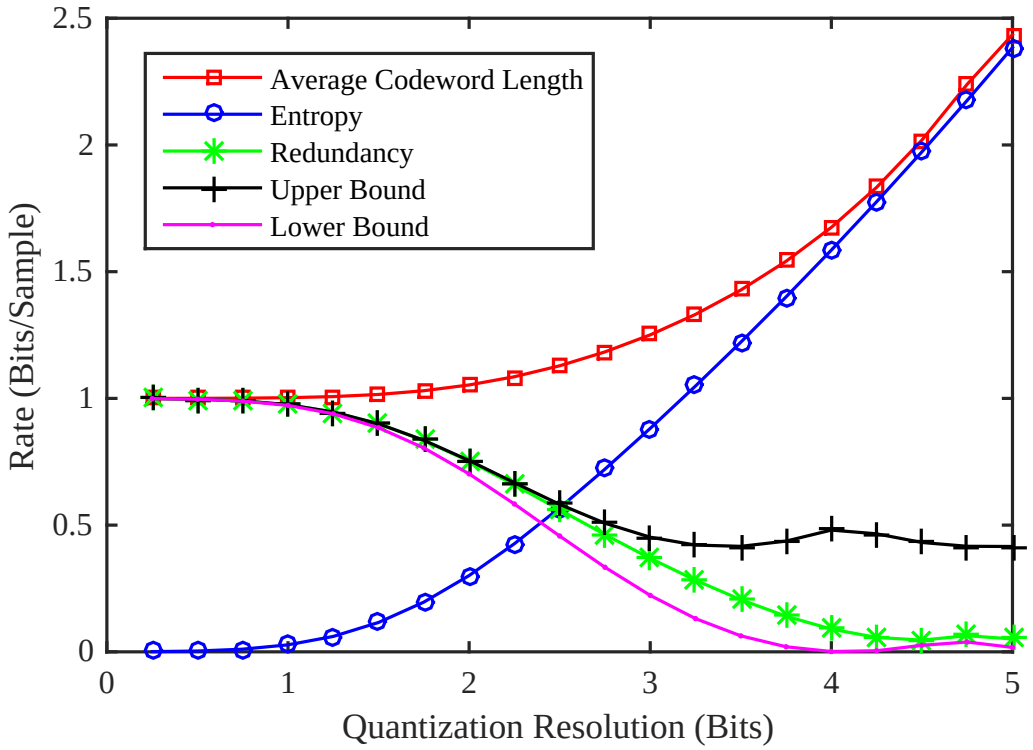


Fig. 6.4 Huffman coding and its redundancy bounds. The plots include a UTQ quantizer output entropy for a Laplacian source, the average codeword length of the quantizer output after Huffman coding, the upper and lower redundancy bounds from [102], and the actual redundancy. The redundancy becomes smaller as the resolution increases.

redundancy. Finally, the actual redundancy is shown in green. This was obtained by subtracting the blue line from the red. Note that, these plots are only for a Laplacian source and the specific UTQ quantizer which is used in AAC.

It can be seen that the redundancy is bigger at low resolutions and as we go to the higher resolutions it tends to zero. In fine-grain scalable coding we are interested in the low resolutions. Even these tight bounds implies that this could result in a difference of tens of kbps in bitrate of the overall scalable coding.

6.4 The Scalable Entropy Coding

As mentioned before, there is no use of the internal nodes in a Huffman tree. However, if we prune the tree at a specific internal node so that parts of the tree between that internal

node and the end nodes are discarded, we can create a new tree with a new set of end nodes which has again the unique prefix property. This way we can reduce the bitrate of the symbols. For example at node $s(5,6)$ in Fig. 6.5, with a new codeword 111 in the resulting bitstream. However, in the context of the original system, this codeword has no meaning as it cannot be mapped to a symbol.

6.5 Relating the Internal Nodes of a Huffman Tree to the Quantization Reconstruction Points

Consider the case where symbols represent the outputs of a scalar quantizer. In the construction of the coding tree if we constraint the merging so that the leaves being merged represent quantizer outputs that are neighboring Voronoi regions, the resulting node can be assigned a new reconstruction point and be treated as a leaf which means we can effectively get a new quantizer if the tree is pruned at that node. Such a set of quantizers is shown in Fig. 6.6, where the tree describes a set of quantizers $(Q_1, Q_2, ...)$ resulting from pruning a quantizer-encoding tree from the bottom up.

As the tree gets pruned, each new quantizer has a smaller entropy and larger distortion compared to the previous one. The reduction of the average bit rate of the quantizer is obtained by

$$\Delta B = p_1 b + p_2 b - (p_1 + p_2)(b-1) = p_1 + p_2 \tag{6.1}$$

where b is the number of bits assigned to two nodes before merging and p_1 and p_2 are the probabilities of the nodes or leaves. By pruning the tree at different possible nodes, we can create a large set of quantizers and hence obtain fine grain bit rate scalability. We note that the receiver needs to know which tree to use to decode a given bitstream; thus an index indicating the pruning level (that is, the quantizer labels, $Q_1, Q_2, \ldots$) needs to be sent as side information. In fact, this label information replaces side information used in the practical scalable coders (including S-AAC) where a scale factor is sent for each layer so that the receiver knows which quantization resolution is used for each of them.

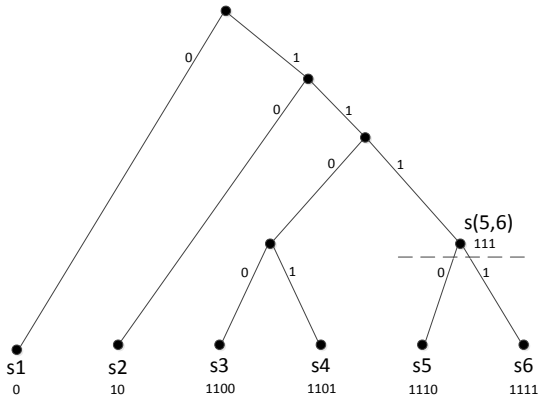


Fig. 6.5 Huffman coding tree and pruning. If the internal nodes are used, the tree can be pruned at any internal node to change the bitrate in a fine-grain manner.

6.6 Merging Quantizer Regions to Build the Coding Tree

The construction of the scalable entropy coding tree is determined not only by the probability density function (*pdf*) of the signal to be encoded, but also the distortion metric we wish to optimize. For a signal x with pdf given by $f(x)$, consider the scalar quantizer $Q(x)$ with distortion

$$\begin{aligned} D &= E[e^2] \\ &= \int_x (x - Q(x))^2 f(x)\, dx \\ &= \sum_i P_i D_i, \end{aligned} \tag{6.2}$$

where D_i is the conditional distortion in interval i of the quantizer. P_i is the probability of $x \in X_i$, so if we write $\hat{x}_i = Q(x)|_{x \in X_i}$,

$$D_i = E[e^2|X_i] = \int_{x \in X_i} (x - \hat{x}_i)^2 \frac{f(x)}{P_i} dx. \tag{6.3}$$

Consider merging the adjacent quantizer regions X_k and X_{k+1}. Now, we have a new

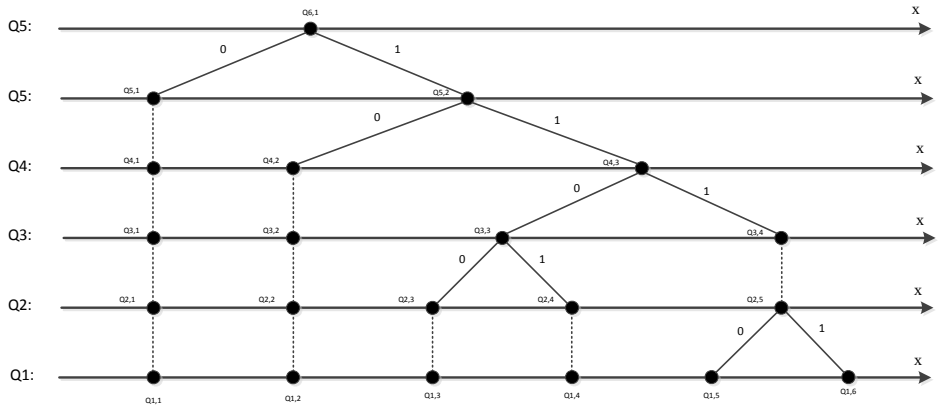


Fig. 6.6 Creating new quantizers by merging the nodes. The internal nodes are mapped to reconstruction points and a Huffman-Like tree is created. By pruning the tree at the internal nodes a set of quantizers with different RD performance is created.

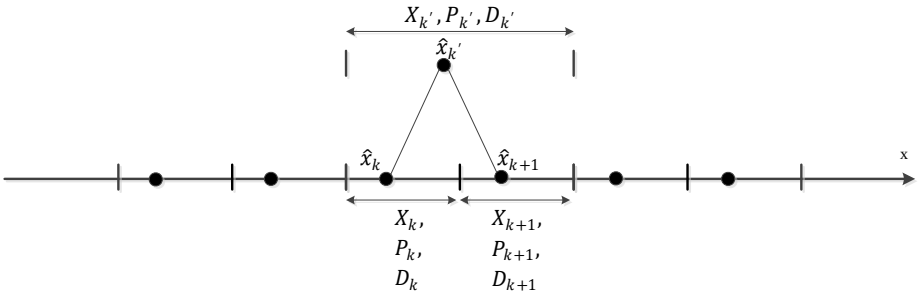


Fig. 6.7 Merging two reconstruction points in a quantizer. Intervals X_k and X_{k+1} are merged to a new interval X'_k with new reconstruction point, probability and distortion.

quantizer with slightly higher distortion, with distortion given by

$$\begin{aligned} D' &= \sum_{i \neq k, k+1} P_i D_i + P_{k'} D_{k'} \\ &= D - (P_k D_k + P_{k+1} D_{k+1}) + P_{k'} D_{k'}. \end{aligned} \tag{6.4}$$

The difference in distortion between the old and new quantizers can now be written as $\Delta D = D' - D$ or

$$\Delta D = P_{k'} D_{k'} - (P_k D_k + P_{k+1} D_{k+1}). \tag{6.5}$$

Now suppose we want to merge two quantization regions to form a new interval (See Fig.

6.7). Changing the reconstruction point of one interval does not change the distortions of other intervals, so the best reconstruction point for the new node is obtained by minimizing the conditional distortion in the new node's interval,

$$\begin{aligned} D_{k'} &= E[(x-\hat{x}_{k'})^2 \mid X'_k] \\ &= \int\limits_{x\in X'_k} (x-\hat{x}_{k'})^2 \frac{f(x)}{P_{k'}}\,dx, \end{aligned} \tag{6.6}$$

giving

$$\begin{aligned} \frac{dD_{k'}}{d\hat{x}_{k'}} &= -2\int\limits_{x\in X'_k} (x-\hat{x}_{k'})\frac{f(x)}{P_{k'}}\,dx = 0 \\ \Rightarrow \hat{x}_{k'} &= \int\limits_{x\in X'_k} x\frac{f(x)}{P_{k'}}\,dx \\ &= E[x \mid X'_k]. \end{aligned} \tag{6.7}$$

The new distortion $D_{k'}$ can be expressed in terms of the conditional expectations of the two merging nodes in their own intervals. Thus,

$$\begin{aligned} P_{k'}D_{k'} &= P_{k'}E[e^2 \mid X'_k] \\ &= P_kE[e^2 \mid X_k] + P_{k+1}E[e^2 \mid X_{k+1}], \end{aligned} \tag{6.8}$$

where

$$\begin{aligned} E[e^2 \mid X_k] &= \int\limits_{x\in X_k} (x-\hat{x}_{k'})^2\frac{f(x)}{P_k}\,dx \\ &= \int\limits_{x\in X_k} [(x-\hat{x}_k)^2 + (\hat{x}_k-\hat{x}_{k'})^2 \\ &\quad + 2(x-\hat{x}_k)(\hat{x}_k-\hat{x}_{k'})]\frac{f(x)}{P_k}\,dx \\ &= D_k + (\hat{x}_k-\hat{x}_{k'})^2 \\ &\quad + 2(\hat{x}_k-\hat{x}_{k'})(E[x \mid X_k]-\hat{x}_k), \end{aligned} \tag{6.9}$$

and

$$E[e^2 \mid X_{k+1}] = D_{k+1} + (\hat{x}_{k+1} - \hat{x}_{k'})^2 + 2(\hat{x}_{k+1} - \hat{x}_{k'})(E[x \mid X_{k+1}] - \hat{x}_{k+1}). \tag{6.10}$$

Consequently,

$$\begin{aligned} P_{k'} D_{k'} &= P_k E[e^2 \mid X_k] + P_{k+1} E[e^2 \mid X_{k+1}] \\ &= P_k D_k + P_{k+1} D_{k+1} \\ &\quad + P_k(\hat{x}_k - \hat{x}_{k'})^2 + P_{k+1}(\hat{x}_{k+1} - \hat{x}_{k'})^2 \\ &\quad + 2P_k(\hat{x}_k - \hat{x}_{k'})(E[x \mid X_k] - \hat{x}_k) \\ &\quad + 2P_{k+1}(\hat{x}_{k+1} - \hat{x}_{k'})(E[x \mid X_{k+1}] - \hat{x}_{k+1}), \end{aligned} \tag{6.11}$$

which gives us the distortion of the new interval in terms of the new and old reconstruction points and the weights and conditional expectations of the old nodes.

Finally, using (6.5) we get

$$\begin{aligned} \Delta D &= P_k(\hat{x}_k - \hat{x}_{k'})^2 + P_{k+1}(\hat{x}_{k+1} - \hat{x}_{k'})^2 \\ &\quad + 2P_k(\hat{x}_k - \hat{x}_{k'})(\bar{x}_k - \hat{x}_k) \\ &\quad + 2P_{k+1}(\hat{x}_{k+1} - \hat{x}_{k'})(\bar{x}_{k+1} - \hat{x}_{k+1}), \end{aligned} \tag{6.12}$$

where the conditional expectations have been replaced by $\bar{x}_k$ and $\bar{x}_{k+1}$.

Equation (6.5) gives the distortion increase for a general case where the reconstruction points can be at arbitrary positions within quantizer intervals. The new reconstruction point $\hat{x}$ in this equation can be obtained as

$$\hat{x}_{k'} = E[x \mid X'_k] = \frac{P_k}{P_{k'}} \bar{x}_k + \frac{P_{k+1}}{P_{k'}} \bar{x}_{k+1}. \tag{6.13}$$

6.7 Obtaining an Appropriate Metric

Now that we have equations giving the distortion increase and the bit rate decrease resulting from merging, we need a metric for finding the best choice of nodes for merging at each step. In Huffman coding we pick the two nodes which have the smallest probabilities. For our case we need a metric which considers both the bitrate decrease and the distortion

increase resulted from merging and we only merge the neighboring nodes. One might think of the Lagrange multiplier method that first comes into the mind. However, the problem here is a bit different from regular constrained distortion and bitrate minimization. First of all, during the intermediate merges, the resulting bitrates are not known. Similar to Huffman coding, the codewords are assigned to the leaf and intermediate nodes only after the full tree is formed. In fact, what we have here is the bitrate difference values from Eq. (6.1). The problem here is choosing an overall path among the specific paths that can be created on the RD plane by different possible merging combinations. While one of these paths may result in the minimum distortion at a specific intermediate point (bitrate), there is no guarantee that it does the same at other points. It will become clearer in the following paragraphs. Consider a candidate metric which is a linear combination of the squared value of the distortion and bitrate changes defined as

$$M = \alpha \Delta D^2 + \beta \Delta B^2. \tag{6.14}$$

This is a simple metric which was used in [25] and [101]. If $\alpha = \beta$, the metric is equivalent to the squared length of the difference vector on the rate-distortion plane. Figure 6.8 shows two successive merges using two different metrics. Consider the first merge. By choosing (6.14) as the metric, we get to the point $b1$, whereas using another metric defined as

$$M = \frac{\Delta D}{\Delta B}. \tag{6.15}$$

we get to $b2$. It can be seen that, although for the path with metric M_2 the difference vector has a larger length, the final point is closer to the bitrate axis. This means that the distortion increase is smaller for the same rate decrease. This implies that, choosing the vector's slope may be a better metric choice. Now consider the destination point after the two merges. Suppose that we choose the overall difference vector's slope as the metric. It can be seen that, although the path M_1 leads to the point $c1$ with smaller overall slope, the area confined between the path and the bitrate axis is larger compared to the path M_2 with the final point $c2$.

To have an appropriate metric for scalable coding, not only the overall changes in the distortion and bitrate are important, but also the intermediate points should be considered. Recall from Section 6.6 that, the proposed fine-grain scalable coding is achieved by pruning

the tree at intermediate nodes, resulted from the successive merges. Figure 6.8 shows that path M_1 is better than path M_2, as far as the final point is considered. However, since the first merge leads to a larger distortion increase, the overall path of M_1 is not necessarily better than M_2. These observations imply that, choosing the area between the overall path and the bitrate axis at each merge is a better metric than (6.14) and (6.15).

The above discussion suggests that the appropriate way to form the coding tree is to keep track of all the merges while the tree is formed, as opposed to the greedy method used in [101] and [25]. In other words, all the possible overall paths should be compared and the path minimizing a metric be chosen as the best path. Consider an N-bit quantizer with 2^N intervals. There are $2^N - 1$ possibilities for the fist merge, $2^N - 2$ for the second one and so on. Therefore , to form the whole tree $(2^N - 1)!$ possible paths have to be considered. Comparing all these paths leads to a huge complexity. For example, for an 7-bit quantizer with 128 intervals the number of possibilities is about 3×10^{213}, which makes the implementation impractical. Note that, in practice we may need to consider a quantizer with even higher resolutions e.g. a 16-bit quantizer. In this paper we propose a trellis-based approach for finding the best RD path. For the trellis-based optimization using the Viterbi algorithm[1], we need a metric that is additive.

In our method, we pick the area as the merging metric for two reasons. First, because of the above facts regarding the ability for taking care of the intermediate points (comparing with M_1 and M_2) and second, because of the additive property which is required in the trellis-based optimization. In the following, we will obtain a relation for this metric which gives the area for a path after n merges.

Consider Fig. 6.9. We want to obtain the area between the path and the horizontal line drawn from the beginning point. Note that, the area between this line and the bitrate axis is the same for all paths starting from the same point and it will appear in the metric equation for all these paths. Therefore, since we are looking for a path with minimum metric, that area can be ignored.

[1]The Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence of states in a trellis [103].

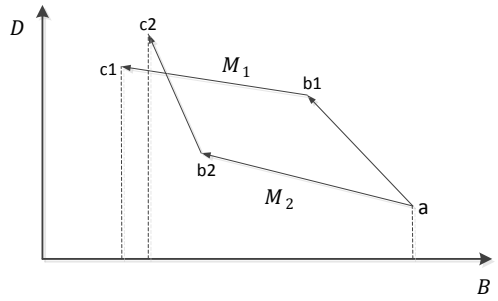


Fig. 6.8 Two successive merges using two different metrics. Choosing the difference vector's slope is a better metric compared to its length for the first merge, whereas, it is not a good metric for the overall path.

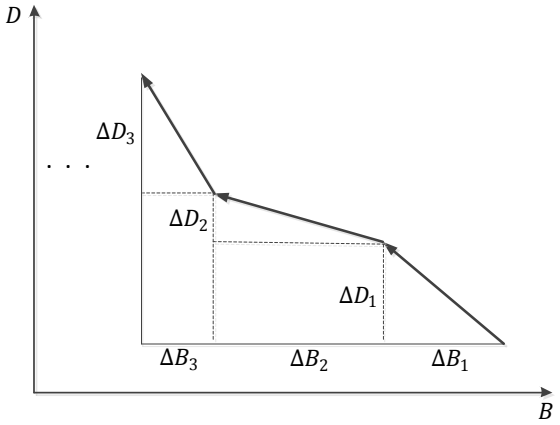


Fig. 6.9 The area between the path and the horizontal line drawn from the beginning point is considered as a metric for merging decision. The first 3 merges were shown in the figure.

After each merge i the area added to the path is

$$a_i = \frac{1}{2}\Delta B_i \Delta D_i + \Delta B_i \sum_{j=1}^{i-1} \Delta D_j, \tag{6.16}$$

where a_i, ΔD_i and ΔB_i are the area, the distortion increase and the bitrate decrease after merging i respectively. The area for the overall path after n merges, A_n, can be obtained

by

$$\begin{aligned} A_n &= \sum_{1}^{n} a_i \\ &= \sum_{1}^{n} \frac{\Delta B_i \Delta D_i}{2} + \sum_{i=2}^{n} \Delta B_i \sum_{j=1}^{i-1} \Delta D_j \\ &= \sum_{1}^{n} \frac{\Delta B_i \Delta D_i}{2} + \sum_{i=2}^{n} \sum_{j=1}^{i-1} \Delta B_i \Delta D_j, \end{aligned} \tag{6.17}$$

This equation can also be expressed as a matrix multiplication form

$$A_n = \mathbf{b}_n^T \mathbf{C}_n \mathbf{d}_n, \tag{6.18}$$

where

$$\mathbf{b}_n = \begin{bmatrix} \Delta B_1 \\ \Delta B_2 \\ \vdots \\ \Delta B_n \end{bmatrix}, \tag{6.19}$$

$$\mathbf{d}_n = \begin{bmatrix} \Delta D_1 \\ \Delta D_2 \\ \vdots \\ \Delta D_n \end{bmatrix}, \tag{6.20}$$

and

$$\mathbf{C}_n = \begin{bmatrix} \frac{1}{2} & 0 & 0 & \cdots & 0 \\ 1 & \frac{1}{2} & 0 & \cdots & 0 \\ 1 & 1 & \frac{1}{2} & & 0 \\ \vdots & & & \ddots & \vdots \\ 1 & 1 & 1 & \cdots & \frac{1}{2} \end{bmatrix}. \tag{6.21}$$

In the above relation, $\mathbf{C}_n$ is a $n \times n$ lower triangular matrix with all the non-zero elements equal to one, except the diagonal elements which are equal to $\frac{1}{2}$.

In our approach, the overall area from the beginning is considered as the metric. In other words, if we start from a quantizer with L intervals, the path which minimizes the

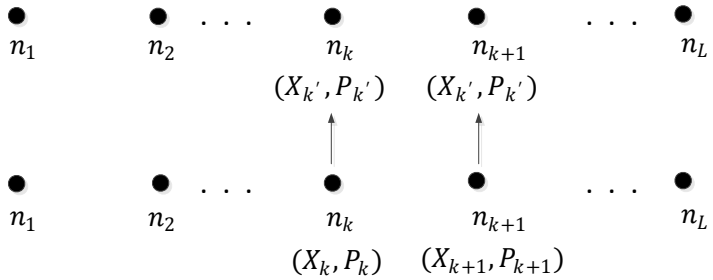


Fig. 6.10 Copying strategy to keep the number of nodes constant. Nodes n_k and n_{k+1} are merged to a new node with ${X_k}'$ and ${P_k}'$. This node is copied and two new nodes with the same label are created.

metric after the $L-1$ merges

$$M = A_{L-1} = \mathbf{b}_{L-1}^T \mathbf{C}_{L-1} \mathbf{d}_{L-1}. \tag{6.22}$$

is considered as the best path.

6.8 Trellis-Based Optimization for Creating the Coding Tree

In this section we present a trellis-based algorithm to find the best path for creating the coding/decoding tree discussed in Section 6.6. Consider again an N-bit resolution quantizer. There are 2^N intervals in such a quantizer. In practical coders typically the absolute value of the quantizer indices are coded and there is a sign bit which is sent separately. We use the same method here. This means, we consider a one-sided positive signal to be quantized by a $N-1$ bit quantizer. The goal is here then to create a scalable coding tree for such a quantizer.

Let us define $L = 2^{N-1}$. We have $L-1$ possibilities for the first merge. After each merge, the number of possibilities decreases by one. We create a trellis consisting of states and stages and we define a cost function. At each stage we want to find the best incoming path to each state which minimizes the cost function. We define the states to be the $L-1$ merging possibilities. Note that, the number of possibilities decreases when we go to subsequent stages. To maintain a trellis with the same number of states at stages, at each merging, instead of replacing the two merging nodes with the new node, we copy the new node so that the number of nodes remain constant at each stage. These two new nodes

will be neighboring nodes with the same new reconstruction point and probability labels. Note that, the physical position of the nodes are not important here, since we are dealing only with the labels. Figure 6.10 shows such a copying strategy. In this figure nodes n_k and n_{k+1} are merged into a new node with ${X_k}'$ and ${P_k}'$. This node is copied and two new nodes with the same label are created.

Now we define the states to be the possibilities of merging at each stage. For instance, for the first merging the possibilities we have are: merging n_1 and n_2, n_2 and n_3 and so on. Accordingly, the states are defined to be $(1,2),(2,3),\cdots,(L-1,L)$. At the second stage, we have L nodes but $L-1$ actual possibilities because nodes k and $k+1$ are the same. However, we keep the number of states to be constant $(L-1)$ at all the stages by keeping the number of nodes to be constant (L). To explain the reason, see Fig. 6.11. We have $L-1$ stages and at each stage we have $L-1$ states. Suppose we go from state $(1,2)$ to state $(3,4)$ from stage 1 to stage 2. This means that at the first stage we merge nodes n_1 and n_2 and at the second stage we merge nodes n_3 and n_4. However, consider now we go from state $(1,2)$ to state $(2,3)$. Since after the first merge the new node is copied to form a new pair of similar nodes n_1 and n_2, merging n_2 and n_3 at the next stage means that the resulting new node is going to be merged with node n_3. In brief, the new node obtained from these two successive merges is a node resulting from merging nodes n_1, n_2 and n_3. This new node is copied and replaces all these three nodes. In general, after each merge, the new node replaces all the children nodes that formed this node during the previous merges.

It is also important to note that, there is no path going from one state to a similar one at any of the following stages. For instance, no path goes from $(1,2)$ to $(1,2)$ since two nodes can be merged only once. In other words, the overall path will not pass the same state twice.

Now we define the cost to be a function of the distortion and the bitrate vectors:

$$\text{cost} = f(\mathbf{d},\mathbf{b}) \tag{6.23}$$

In order for the Viterbi algorithm to guarantee finding the best path, the cost function should be additive. As mentioned in the previous section, we consider the area as a metric

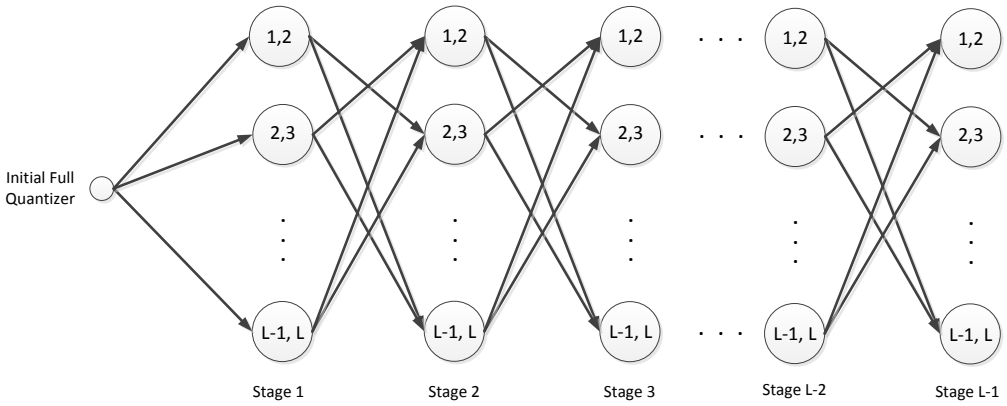


Fig. 6.11 Trellis-based approach for finding the best path on the RD plane. There are $L-1$ stages and $L-1$ states at each stage. Each state represents the the nodes being merged at the current stage. The initial quantizer is shown by a single sate at the beginning of the trellis where the merging starts.

for finding the goal path. Equation (6.22) can also be expressed in a recursive relation:

$$\begin{aligned} A_n &= A_{n-1} + a_n = A_{n-1} + \Delta B_n (\sum_{i=1}^{n-1} \Delta D_i + \frac{\Delta D_n}{2}) \\ &= A_{n-1} + \Delta B_n \mathbf{c}_n \mathbf{d}_n, \end{aligned} \tag{6.24}$$

where $\mathbf{c}_n$ is the n^{th} (and last) row in the matrix $\mathbf{C}_n$. The new area at stage n, a_n, has two components: a_n^r (rectangular part) and a_n^t (triangular part). With a good approximation a_n^t can be considered fixed for a specific transition from state i at stage $n-1$ to the state j at stage n, since ΔB_n and ΔD_n are the same (strictly so if the previous states in the path are not neighbors to the state j). However, a_n^r depends not only on ΔD_n, but also on the overall distortion for the previous merges, sum($\mathbf{d}_{n-1}$). To satisfy the additive property of the cost function, we consider the cost function to be the overall area for the current merge plus a prediction of a^r for the next merge. To show this, consider the first merge at state

1. From (6.1) the average ΔB resulting from this merge can be obtained by

$$\begin{aligned} E[\Delta B_1] &= \frac{\sum_{i=1}^{i=L-1}(P_i + P_{i+1})}{L-1} = \frac{\sum_{i=1}^{i=L-1} P_i + \sum_{i=2}^{i=L} P_i}{L-1} \\ &= \frac{(1-P_L)+(1-P_1)}{L-1} \\ &= \frac{2-(P_1+P_L)}{L-1} \end{aligned} \tag{6.25}$$

At the next stages, although the number of (different) nodes decreases, the summation in the numerator of above relation remains the same (except for the marginal nodes). However, the number of merging possibilities decreases at each stage. Therefore, $E[\Delta B]$ at stage n can be estimated by

$$\tilde{E}[\Delta B_n] = \frac{2-(P_1+P_L)}{L-n} \tag{6.26}$$

which is a function of stage index n. Therefore, a^r at stage $n+1$ can be predicted by

$$\tilde{a}^r_{n+1} = \tilde{E}[\Delta B_{n+1}] S(\mathbf{d}_n) \tag{6.27}$$

where $S(\mathbf{d}_n) = \sum_{i=1}^{n} \Delta D_i$. We define

$$\begin{aligned} r(n) &= k\tilde{E}[\Delta B_{n+1}] = k\frac{2-(P_1+P_L)}{L-n-1} \\ &= \frac{K}{L-n-1} \end{aligned} \tag{6.28}$$

where K is a constant that can be adjusted based on the statistics of the input signal and the experiments. The decision rule for finding the best path arriving at state j at stage n is expressed by

$$\begin{aligned} I_n^{(j)} = \underset{i}{\operatorname{argmin}} \Big\{ & A_{n-1}^{(i)} + a_n^{(i\to j)} \\ & + r(n)\ (S^{(i)}(\mathbf{d}_{n-1}) + \Delta D_n^{(i\to j)}) \Big\} \end{aligned} \tag{6.29}$$

where $I_n^{(j)}$ is the index of the state from stage $n-1$ giving the best path arriving at state j at stage n, $A_{n-1}^{(i)}$ is the minimum area stored for state i at stage $n-1$, $a_n^{(i\to j)}$ is the new

area to be added for going from state i to j, $S^{(i)}(\mathbf{d}_{n-1})$ is sum of the distortion stored for state i and $\Delta D_n^{(i\to j)}$ is the new distortion for going from state i to j at stage n. Since ΔBs and ΔDs are stored for the kept paths, the overall area in the above relation can also be calculated using (6.18).

Having formed the trellis and defined the cost function, we have the following algorithm from stage 1 to stage $L-1$:

Algorithm 2: Trellis-Based Algorithm for Finding the best RD Path

- For each state find all the possible incoming paths. Only paths that have not passed this state before are acceptable.
- Calculate the cost for all these paths. Keep the path resulting in the minimum cost and discard the rest.
- Go to the next stage.

At the last stage, a maximum of $L-1$ paths are obtained using the above algorithm, one for each state . It is said maximum because, it is possible that one state is discarded during the above process. It could happen when all the possible paths have already passed that state at intermediate stages. These remaining paths are compared and the one with the minimum cost is picked as the best path in the trellis. From this path, the successive merges are identified which are used for creating the coding tree.

6.9 Applying the Proposed Approach to a Practical Quantizer

To create a test environment, we apply the proposed scalable coding scheme to a practical quantizer which is used in AAC. In AAC and S-AAC, the MDCT coefficients are quantized in each scalefactor band (SFB) of a data frame using AAC-UTQ mentioned before and given again as [3]:

$$i_n[l] = \mathrm{sgn}(c_n[l])\mathrm{nint}(|2^{-s[n]/4}c_n[l]|^{3/4} - 0.0946), \tag{6.30}$$

At the decoder the coefficients are reconstructed as

$$\hat{c}_n[l] = \mathrm{sgn}(i_n[l])\big(2^{s[n]/4}\big|i_n[l]\big|^{4/3}\big), \tag{6.31}$$

where $\hat{c}_n[l]$ is the l^{th} reconstructed coefficient in band n.

It can be noticed from the above equations that the quantizer resolution is changed by changing the scalefactor. In other words, instead of changing the quantizer intervals, there is a full quantizer with fixed intervals and the input signal (consisting of the MDCT coefficients) is scaled. The inverse scalefactor is applied at the decoder and the whole process is equivalent to changing the quantizer resolution. It is useful to note that in AAC quantization the ratio of the offset to the step size of the quantizer remains the same (0.0946) as the scalefactor s (equivalently the step size) changes. For a Laplacian source, the relationship between this ratio r and the step size parameter Δ can be obtained by

$$r = k\Delta - \frac{1}{e^{\frac{1}{k\Delta}} - 1}, \tag{6.32}$$

where k is the ratio of the scale parameter of a Laplacian source with zero mean to the quantizer step size Δ. This relation was plotted in Fig. 6.12 for $k = 1$. It can be seen that the ratio r is neither constant nor has a linear relation with Δ. Equation (6.12) provides a proper general distortion increase relation for all types of quantizers where the reconstruction points can take any positions within the intervals and may not exactly match the statistics of the input signal.

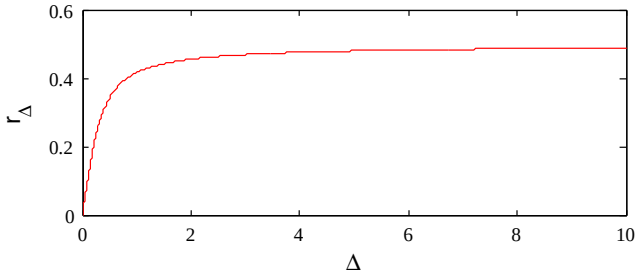


Fig. 6.12 Offset to step size ratio r versus step size parameter Δ ($k = 1$) for AAC quantizer.

We apply the scalable coding systems to an N-bit AAC quantizer with loading factor $L_f = 7$. The quantization loading factor L_f is defined as $L_f = x_m/\sigma$, where σ is the standard deviation of the input signal (MDCT coefficients) and x_m is the quantization limit which is for instance $2^{13} - 1$ in AAC quantization. Normally $L_f \geq 7$ is chosen for Laplacian signals to avoid quantization overload. For faster simulations, we set the limit to

be $2^{N-1}-1$ which means an N-bit quantizer was used, and then adjust the input statistics to match the loading factor. Note that, changing the quantizer resolution ($L_f = 2x_m/\Delta$) in S-AAC and NS-AAC, to go for resolutions less than N bits, is still performed by scaling the input signal ((MDCT coefficients)) by changing the scalefactor. In AAC quantizer, the step size for the quantization index i in SFB n can be obtained by

$$\Delta(i_n) = \text{thr}(i_n + 1) - \text{thr}(i_n), \tag{6.33}$$

where $\text{thr}(i_n)$ and $\text{thr}(i_n + 1)$ are the corresponding interval thresholds and are given by

$$\text{thr}(i_n) = \begin{cases} \text{sgn}(i_n)(2^{s[n]/4}|i_n - 0.4054|^{4/3}), & i \neq 0 \\ 0, & i = 0, \end{cases} \tag{6.34}$$

We start merging from the N-bit initial quantizer mentioned above with distinct interval thresholds and reconstruction points. The reconstruction points are mapped to the nodes discussed in Sections 6.6 and 6.7. Using the trellis-based approach we find the best path on the RD plane for the defined cost function. Using this path we can form all the intermediate quantizers which can be obtained by pruning the coding tree at arbitrary intermediate nodes (Fig. 6.6).

6.10 Simulation and Results

6.10.1 Methodology

In this section we present the theoretical and practical results from applying the competing systems to a Laplacian source and to the real audio. For practical results, objective and subjective measurements have been performed for evaluating the output audio quality. Ten audio files were taken as input signals for the simulations that includes speech, vocal music, solo instrumental music, and orchestral music. These audio files were selected from two common databases for audio quality assessment [104] and [105]. The files are all mono with sampling frequency of 48kHz. The MDCT coefficients are calculated for these files and grouped into scalefactor bands (SFBs). In AAC the input signal is coded frame by frame. The frame size is 2048 which gives 1024 MDCT coefficients. The MDCT coefficients are then grouped into 49 SFBs (for 48 kHz sample rate). In each SFB, the coefficients are quantized independently based on the masking thresholds obtained from the AAC

psychoacoustic model for that SFB. The quantization indices for the quantized coefficients are then entropy-coded using Huffman Tables. In S-AAC the MDCT coefficients in each SFB are quantized using REQ.

In our simulations, we obtain the MDCT coefficients and apply TrelSEC and S-AAC to them in each SFB. In S-AAC we apply a 4-layer REQ system. The bitrates in bit/sample for the four layers are matched to the practical bitrate steps in kbps used in S-AAC. The MPEG-4 reference software supports three bitrate steps: 16 kbps, 32 kbps and 48 kbps. Although different combination of theses rates are allowed, in our simulations we consider the bitrate per layer to be constant in each system. Note that, the final bitrate in kbps includes all the information required for decoding an audio frame including the encoded quantized coefficients together with the side information including the scalefactor and the Huffman table index used for each SFB. What we want to compare here is the quantization distortion versus the bitrate resulting from the entropy coding in the two coding systems. Therefore, we just consider the bitrate of the entropy-coded quantized coefficients in bits/sample. To match these bitrates to the total bitrates mentioned above, the quantization in each layer is performed so that, the Average Noise-to-Mask Ratio (ANMR) becomes the same as what obtained for the above total bitrates from the reference software. The ANMR for an audio file is defined as

$$\text{ANMR} = \frac{1}{N_f N_s} \sum_{f=1}^{N_f} \sum_{s=1}^{N_s} \frac{d(f,s)}{m(f,s)}, \tag{6.35}$$

where $d(f,s)$ is the squared error in band s of frame f, $m(f,s)$ is the masking threshold obtained for that band from the psychoacoustic model (here we use the masking thresholds from the AAC reference software), N_f is the number of frames in the audio file, and N_s is the number of bands which is 49 here (for 48 kHz sample rate used in our simulations). To obtain the corresponding bitrates in bits/sample, rate-distortion (RD) loops were created. The first RD loop is for S-AAC which calculates the quantization and entropy coding rate (in bits/sample) required to achieve specified distortions in ANMR. These ANMR values were calculated for the audio signals encoded using AAC/S-AAC reference software for each bitrate (16 kbps/layer, 32 kbps/layer,...). For TrelSEC, another RD loop adjusts the rate in each layer of TrelSEC coding to match the above bitrates. Then, the resulting distortion in ANMR is calculated for TrelSEC. Therefore, in our comparison, the bitrate for the above systems are assumed to be almost the same and the resulting distortions are

compared.

For subjective measurements, we performed the test Multiple Stimuli with Hidden Reference and Anchor (MUSHRA) [106] on the coded audio files to evaluate the subjective quality. To avoid listener fatigue only the 2-layer systems were compared. We had 10 audio files and for each file 3 coded audio files and one reference (the original uncompressed file). These files were presented to 10 relatively experienced listeners and scored on a scale of 0 to 100 where 0 means bad and 100 means excellent quality which is given to the reference.

6.10.2 Results

Figure 6.13 shows a comparison of three coding systems: S-AAC, non-scalable AAC (NS-AAC or simply AAC[2]) and the proposed coding system denoted by Trellis-Based Scalable Entropy Coding (TrelSEC) in terms of SNR(dB) versus bitrate (bits/sample). For the test, a Laplacian source was generated and was applied to the quantizers. A set of Huffman coding tables were generated corresponding to the various quantization resolutions and were used for entropy coding in both S-AAC and NS-AAC. In Scalable AAC a 4-layer REQ system was used with a 3-bit quantizer at each layer.

Note that, in practice the quantization resolution required at each layer rarely exceeds 3 bits for fine-grain scalable bitrate increment steps such as 16 kbps and 32 kbps. Also, there is a theoretical abound in the plot which is the SNR for an optimal entropy-constrained quantizer for a Laplacian source[3] from [107]. The SNR for the NS-AAC can also be considered as a bound for MPEG-4 scalable coding. In this system (NS-AAC) the bitrate of the AAC quantizer is changed gradually in the steps of 0.25 bit by changing the scalefactor. The corresponding SNRs (in approximately 1.5 dB steps) are obtained for the applied Laplacian input. Figure 6.13(a) shows the results for the case where a 7-bit resolution full quantizer was used. It can be seen that the proposed method performs clearly better than S-AAC. There is a gap between the TrelSEC and NS-AAC as expected. However, it can be seen that at low rates, TrelSEC is performing even better than NS-AAC. The reason is that UTQ is not the optimal entropy-constrained quantizer at low resolutions [69]. It is an optimal entropy-constrained quantizer for exponential sources only at high rates. In our approach, we start merging quantizer regions from a UTQ. However, as we go through

[2]We use NS-AAC in the rest of chapter to emphasize the non-scalability.

[3]Note that, there is no explicit closed-form RD function (with MSE distortion criterion) for non-Gaussian memoryless sources [28].

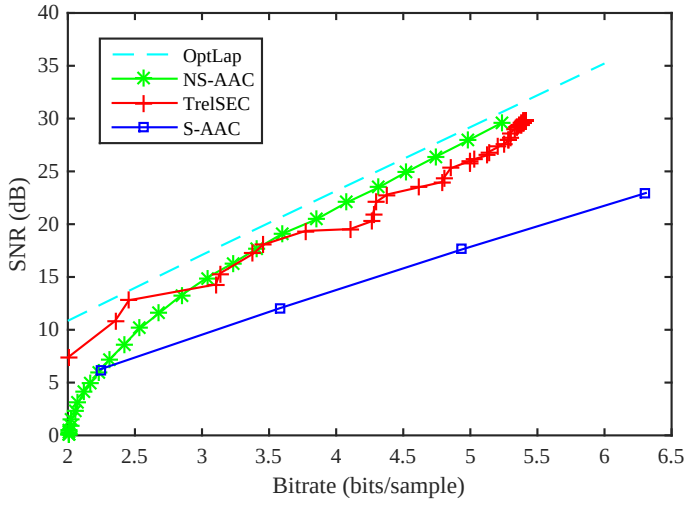
merges the new quantizers are created by minimizing the cost function and are not uniform any more.

Figure 6.13(b) shows a comparison of the above systems when a 10-bit resolution full quantizer was used and for two different cases: when no prediction for the a^r is involved in the decision rule (mode 1) and when the decision rule in (6.29) was used (mode 2). It can be seen that even in mode 1 the proposed scheme performs better than S-AAC. However, in mode 2 the gap between TrelSEC and NS-AAC decreases considerably.

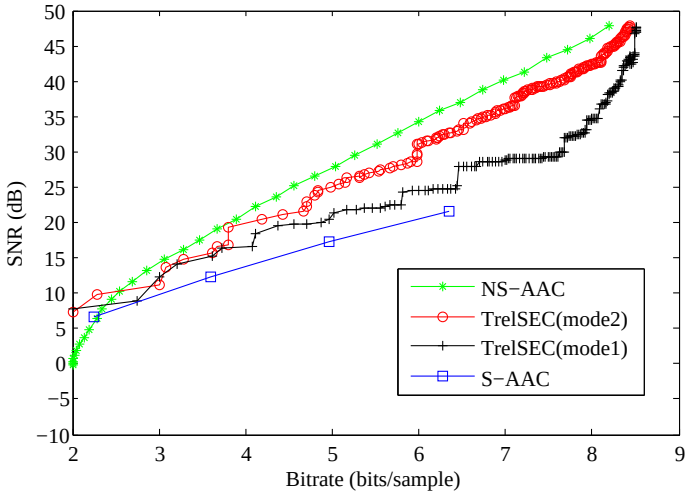
Now we present the objective and subjective evaluation results for the proposed scalable coding approach and compare it with what obtained from Scalable AAC. Figure 6.14 shows the comparison of 4-layer S-AAC, TrelSEC and NS-AAC. The distortion of these systems in ANMR were plotted for the bitrates matched to the 16 kbps, 32 kbps and 48 kbps per layer. For example in Fig. 6.14(a) a single layer bitrate is matched to 16 kbps, two layers matched to 16 kbps+16 kbps and so on. It can be seen that there is a considerable difference in performance of S-AAC and TrelSEC. In TrelSEC the ANMR reaches negative values much faster than in S-AAC. This means that in TrelSEC the distortion tends to inaudibility much faster than in S-AAC. In the case of 16 kbps/layer which is a fine-grain scalable coding, TrelSEC performs even better than NS-AAC for the first two layers because of the fact mentioned above regarding the non-optimality of UTQ at low resolutions. Figure 6.14(b) shows such a comparison when the bitrates are matched to 32 kbps/layer. It is seen that, as the bitrate in each layer increases, the performances of S-AAC and TrelSEC become closer. It is obvious since the higher bitrate of layers in S-AAC means being closer to NS-AAC. At 48 kbps/layer (Fig. 6.14(c)) it becomes even closer. In this case the performance of a 1-layer TrelSEC and S-AAC is almost the same.

Figure 6.15 shows the results of another objective test for using PEAQ. The ODG values obtained for these systems in were plotted for the bitrates matched to the 16 kbps, 32 kbps and 48 kbps per layer. Again, it can be seen that there is a considerable difference in performance of S-AAC and TrelSEC. In fact, the ODG results for TrelSEC are extremely close to that of the non-scalable AAC.

In addition to these objective tests, we also performed MUSHRA test on the coded audio files to evaluate the subjective quality. The subjective test results are presented together with the ANMR objective results in Tables 6.1 and 6.2 for the 2-layer S-AAC, NS-AAC and TrelSEC for bitrates matched to 16 kbps/layer and 32 kbps/layer respectively. For each



(a)



(b)

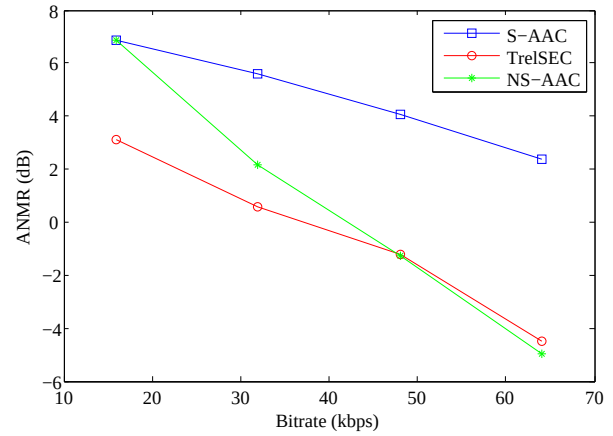
Fig. 6.13 Performance of TrelSEC vs. S-AAC and NS-AAC for a Laplacian source. The SNR for an optimal entropy-constrained quantizer for a Laplacian source from [107] was also plotted as a theoretical bound. A Laplacian source was applied to the systems and the quantization was performed using AAC quantizer ($L_f = 7$). In Scalable AAC a 4-layer REQ system was used with a 3-bit quantizer at each layer. For TrelSEC all the resulting RD pairs were plotted. In NS-AAC the bitrate was changed in 0.25 bit steps by changing the scalefactor. 7-bit and 10-bit full quantizers were used in (a) and (b) respectively.

audio file, the result is provided. In the last row, the average results from all audio files are indicated. It can be seen that TrelSEC has the highest average score, being very close to NS-AAC. While NS-AAC has the highest average score as expected, TrelSEC stands well above S-AAC.

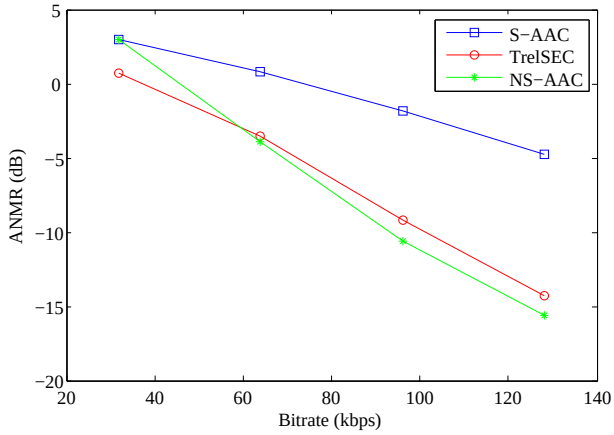
In the end we compare the performance of the WSAC (Scalable Audio Coding Using Watermarking) system with the above systems. Figure 6.16 shows comparison of 4-layer S-AAC, WSAC, TrelSEC, and NS-AAC scalable coding systems. In this case the ANMR values obtained for these systems were plotted vs. the bitrates in bit/sample but again matched to (a) 16 kbps per layer and (b) 32 kbps per layer. Note that, for WSAC it is not straightforward to reach an exact target bitrate as in the other systems, because of the feedback loop that is involved. It can be seen that although WSAC starts to outperform the REQ system used in S-AAC when more than one layer is used, it considerably underperforms the TrelSEC.

6.11 Summary

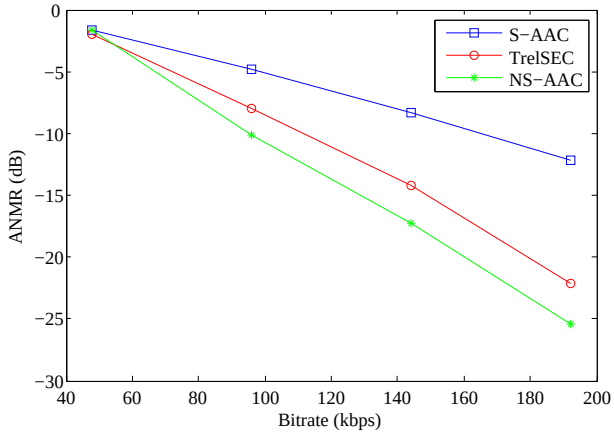
[!h] Scalable audio coding based on REQ is used in practical systems such as MPEG-4 S-AAC. In such a coder, quantization and coding is performed separately for each layer and the coder becomes more suboptimal as the number of layers increases. We have proposed a scalable coding method for a single quantizer in which entropy coding is performed in a fine-grain scalable manner. This approach using the proposed metrics and a trellis-based optimization algorithm seeks to make a coding tree corresponding to a path on the RD plane. By pruning the tree at the internal nodes of the coding tree, a variety of quantizers can be created with different RD performance, corresponding to the generated RD path. Objective and subjective tests were run to evaluate the proposed system's performance. The results show that it considerably outperforms S-AAC scalable coding system. In fact, it performs very close to an equivalent non-scalable system and can be considered as a suitable retrofit with improved performance for practical scalable audio coders.



(a)

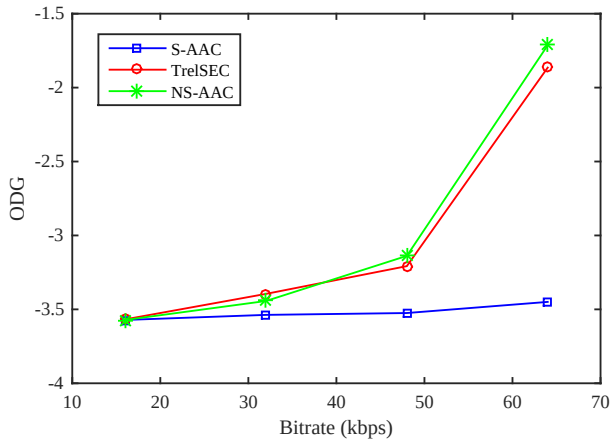


(b)

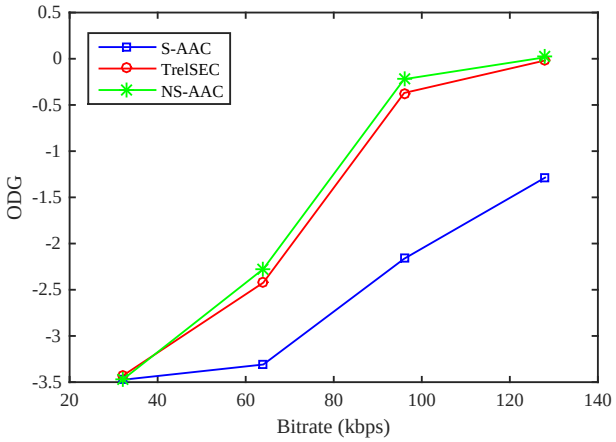


(c)

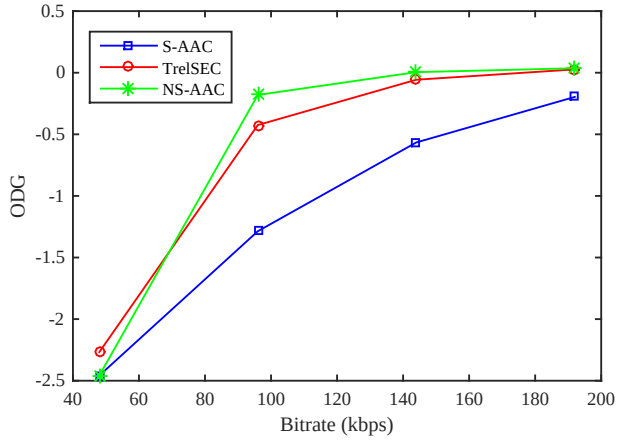
Fig. 6.14 Comparison of 4-layer S-AAC, TrelSEC and NS-AAC scalable coding systems in ANMR. The distortion of the three systems in Average Noise-to-Mask Ratio (ANMR) were plotted vs. the matched total bitrates in kbits/second (kbps) and for three cases: (a) 16 kbps per layer, (b) 32 kbps per layer, and (c) 48 kbps per layer. The masking thresholds came from the MPEG-4 audio reference software psychoacoustic model and the quantization was performed using AAC quantizer in all the systems ($L_f = 7$).



(a)



(b)



(c)

Fig. 6.15 Comparison of 4-layer S-AAC, TrelSEC and NS-AAC scalable coding systems. The ODG values obtained from the objective test using PEAQ were plotted vs. the matched total bitrates in kbits/second (kbps) and for three cases: (a) 16 kbps per layer, (b) 32 kbps per layer, and (c) 48 kbps per layer.

Table 6.1 Objective and subjective quality measurements for 2-layer S-AAC, TrelSEC and NS-AAC for bitrates matched to 16 kbps/layer. The objective measure is Average Noise-to-Mask Ratio (ANMR) in dB and the MUSHRA test was performed as the subjective test. The results for 10 different audio files are shown, together with the average.

	ANMR			Subjective Test		
	S-AAC	**TrelSEC**	**NS-AAC**	**S-AAC**	**TrelSEC**	**NS-AAC**
Male Speech	5.68	0.63	1.91	32.1	56.1	50.6
Female Speech	5.82	0.85	2.17	16.6	42.4	38.8
Vocal (Bass)	5.51	0.57	2.12	34.8	53.2	46.7
Vocal (Soprano)	5.63	0.66	2.15	43.6	52.8	51.0
Violin	5.64	0.65	2.62	28.5	45.9	43.5
Trumpet	5.51	-0.38	2.05	28.5	51.7	46.0
Quartet	5.58	0.68	2.08	37.9	48.3	50.1
Drums	5.81	0.90	2.11	21.9	31.4	23.3
Harpsichord	5.69	0.93	2.73	20.6	35.2	37.6
Orchestra	5.32	0.46	1.66	29.8	35.5	36.7
Average	5.62	0.59	2.16	29.4	45.8	42.4

Table 6.2 Objective and subjective quality measurements for 2-layer S-AAC, TrelSEC and NS-AAC for bitrates matched to 32 kbps/layer. The objective measure is Average Noise-to-Mask Ratio (ANMR) in dB and the MUSHRA test was performed as the subjective test. The results for 10 different audio files are shown, together with the average.

	ANMR			Subjective Test		
	S-AAC	**TrelSEC**	**NS-AAC**	**S-AAC**	**TrelSEC**	**NS-AAC**
Male Speech	1.04	-3.57	-3.74	50.8	55.4	67.8
Female Speech	1.26	-3.12	-3.24	33.9	55.2	56.1
Vocal (Bass)	0.77	-3.48	-3.86	52.1	65.9	65.0
Vocal (Soprano)	0.75	-3.70	-4.16	73.7	74.9	77.6
Violin	0.73	-3.22	-3.94	58.2	71.7	81.9
Trumpet	-0.02	-5.55	-6.21	51.6	72.4	68.6
Quartet	0.85	-3.27	-3.62	62.9	77.6	76.7
Drums	1.29	-3.14	-3.22	51.0	55.8	55.1
Harpsichord	0.91	-2.89	-3.41	33.2	49.0	50.8
Orchestra	0.73	-3.53	-3.77	69.0	65.1	65.8
Average	0.83	-3.55	-3.92	53.6	64.3	66.5

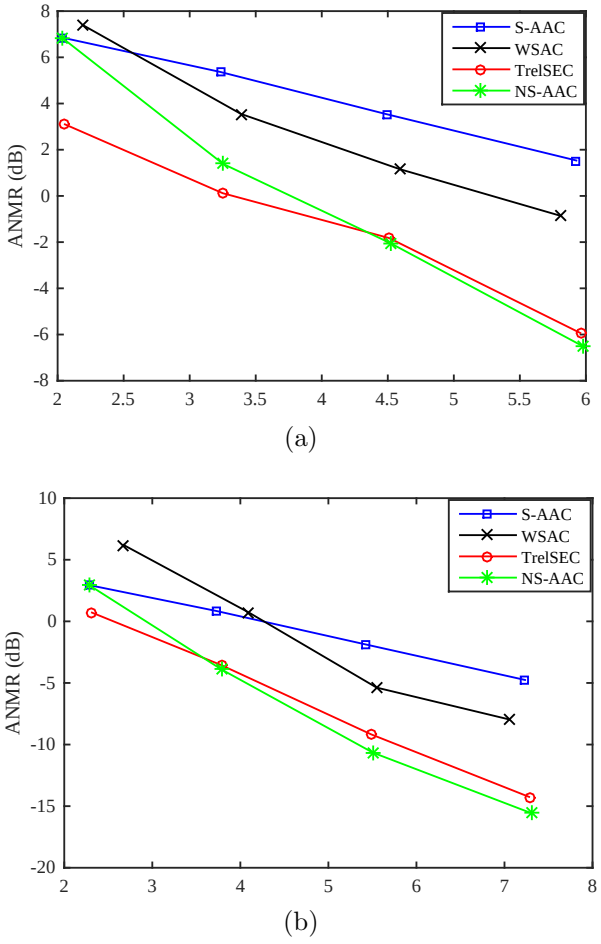


Fig. 6.16 Comparison of 4-layer S-AAC, WSAC, TrelSEC, and NS-AAC scalable coding systems. The ANMR values obtained for these systems were plotted vs. the matched total bitrates in bits/sample and for two cases matched to (a) 16 kbps per layer and (b) 32 kbps per layer.

Chapter 7

Conclusion

7.1 Summary of the Research

In this dissertation we have three different contributions to practical scalable audio coding systems with the latest one presented in Chapter 6. The proposed methods are specifically useful for fine-grain scalable coding where bitrate increment is provided in small steps which is a desired feature for many applications.

In Chapter 4 we presented the first contribution, scalable audio coding using watermarking. We proposed using a watermarking technique called Quantization Index Modulation in scalable coding. Using this technique some of the information of each layer output is watermarked in the previous layer. This approach leads to a saving in bitrate while keeping the distortion almost unchanged. This makes the proposed scalable coding system more efficient in terms of Rate-Distortion when more than one layer is received. The results showed that the proposed method outperforms scalable audio coding based on REQ.

In Chapter 5 the next contribution was provided which was a technique to augment the BPC-based scalable coding used in MPEG-4 audio. Considering the properties of the residual signal, core-based bit-plane probabilities were provided for MPEG-4 Audio Scalable to Lossless Coding (SLS), which better matches the quantization and coding performed in the core layer. Using the same strategy, new probabilities are obtained to consider the clipping effect in bit-plane coding of an unbounded signal. Simulations showed that considering the core layer parameters improves the bit-plane probabilities estimation compared to the existing method.

Perhaps the most important contribution has been presented in Chapter 6. A very fine-

grain scalable coding approach was proposed by designing a scalable entropy coding using a trellis-based optimization. In the proposed scheme, the entropy coding procedure of a single quantizer was made scalable. By constructing a Huffman-like coding tree where the internal nodes can be mapped to the reconstruction points, the tree can be pruned at any internal node to control the rate-distortion (RD) performance of the encoder in a fine-grain manner. A set of metrics and a trellis-based approach were proposed to create a coding tree so that an appropriate path is generated on the RD plane. The results showed the proposed method outperforms the scalable audio coding performed based on reconstruction error quantization as used in practical systems, e.g. in S-AAC.

Among the proposed methods, the latter showed a great improvement on the prior scalable audio coding systems. The results showed that using this technique we can get close to an equivalent non-scalable coder in terms of the RD performance, yet having scalability. The proposed scheme also provides very fine granularity of bitrate increments. In fact the proposed coding/decoding tree provides a bit-by-bit scalability.

7.2 Future Work Suggestions

In Chapter 4 we used the QIM technique to watermark a single bit of a codeword from each layer into the previous layer. This was performed using a modulo-2 QIM. This idea can be extended for using higher modulus. For instance, we can watermark two bits at the same using modulo-4 QIM. It would be interesting to evaluate the performance of such a system and compare it with the one presented here.

As seen in the thesis, there are different factors that play important roles in the performance of the proposed system such as the quantization resolutions and the statistics of the watermarked bits. Higher-modulo QIM tends to be applied to higher resolutions, while the bit sequence form codewords might have statistics which contribute to the performance of the system compared to a single bit watermarking. Another extension to this work could be applying the idea to a vector quantizer where a different moduli needs to be defined for the quantizer cells. The two suggested extensions could be also mixed. This requires using two different modulus at the same time, one for the watermarked bits and the other one for the quantizer.

We proposed a very fine-grain scalable audio coding by designing a scalable entropy coding scheme which was presented in Chapter 6. We applied the idea to a scalar quantizer.

Based on the proposed metrics, the neighboring intervals in the scalar quantizer are merged into a new interval and the process is repeated. A very interesting future work is to see if this idea could be applied to a vector quantizer. In a vector quantizer, instead of the intervals we deal with the cells. Can the neighboring intervals be easily replaced by the neighboring Voronoi regions? Can the same RD metrics proposed here be used to find the best candidates at each merge? A research on this topic can answer the questions. What is clear is that, the tree generation process in this case will be more time-consuming and complex compared to that of a scalar quantizer. However, this is just a one-time and off-line process. The tree and the corresponding codebook are generated and the coding/decoding process can be performed as fast as required by working with a set of tables.

Appendix A

Entropy Calculations for WSAC

In the Watermark-Based Scalable Audio Coding (WSAC) system, the even and odd quantizers follow a UTQ quantizer which is used in AAC. Using the QIM technique either of these quantizers are used representing the watermarked bit. We have two quantizers Q_e and Q_o with entropies E_e and E_o. Except the difference in the quantization indices which is even in E_e and odd in E_o, the quantizers are also different in that Q_e has a dead-zone around zero while Q_o has two dead-zones around 1 and -1. The overall quantizer Q which is a combination of these two quantizers has a single entropy which we will show is a function of these entropies. We have

$$p'_i = \begin{cases} P_0 p_i & i : \text{even} \\ P_1 p_i & i : \text{odd} \end{cases} \tag{A.1}$$

where p'_i is the probability of interval i in Q, p_i is the probability of that interval in either Q_e or Q_o if i is even or odd respectively. We define

$$\begin{aligned} E'_e &= -\sum_{i=\text{even}} p'_i \log_2(p'_i) \\ &= -\sum_{i=\text{even}} P_0 p_i \log_2(P_0 p_i) \\ &= -P_0 \sum_{i=\text{even}} p_i \log_2(p_i) - P_0 \log_2(P_0) \sum_{i=\text{even}} p_i \\ &= P_0 E_e - P_0 \log_2 P_0 \end{aligned} \tag{A.2}$$

where E_e is the entropy of Q_e and P_0 is the probability of the watermarked bit being 0. Similarly for E'_o we get

$$E'_o = -\sum_{i=\text{odd}} p'_i \log_2(p'_i) = P_1 E_o - P_1 \log_2(P_1) \tag{A.3}$$

where E_o is the entropy of Q_o and P_1 is the probability of the watermarked bit being 1. Note that the probabilities p_i correspond to the interval widths in Q_e and Q_o, each of which covers the whole source range. That is why in the above equations $\sum_{i=\text{even}} p_i = 1$ and $\sum_{i=\text{odd}} p_i = 1$.

The entropy of the overall quantizer Q can be obtained as

$$\begin{aligned} E &= -\sum_i p'_i \log_2(p'_i) \\ &= E'_e + E'_o \\ &= [P_0 E_e + P_1 E_o] - [P_0 \log_2(P_0) + P_1 \log_2(P_1)] \\ &= \bar{E} + E_b \end{aligned} \tag{A.4}$$

where $\bar{E}$ is the average entropy of Q_e and Q_o and E_b is the entropy of the watermarked bit. The entropy increase resulted from doubling the resolution is

$$E - E_e = \bar{E} + E_b - E_e \tag{A.5}$$

which depends on E_e, E_o and entropy of the watermarked bit.

There is one dead-zone in Q_e and two in Q_o. The corresponding dead-zone thresholds T_e and T_o are

$$\begin{aligned} T_e &= \frac{\Delta}{2} + sh \\ T_o &= \Delta + sh \end{aligned} \tag{A.6}$$

where sh is the shifting or offset value. For the dead-zone probabilities p_0 and $p_{-1} = p_1$ we

have

$$\begin{aligned} p_0 &= 1 - e^{-\lambda T_e} \\ p_1 &= p_{-1} = \frac{1}{2}(1 - e^{-\lambda T_o}) \end{aligned} \tag{A.7}$$

The entropies E_e and E_o are calculated following the same procedure presented in Chapter 4 which give

$$\begin{aligned} E_e = \lambda \log_2(e) e^{-\lambda T_e} (T_e + \frac{\Delta e^{-\lambda \Delta}}{1 - e^{-\lambda \Delta}}) + e^{-\lambda T_e} \\ - e^{-\lambda T_e} \log_2(1 - e^{-\lambda \Delta}) - (1 - e^{-\lambda T_e}) \log_2(1 - e^{-\lambda T_e}) \end{aligned} \tag{A.8}$$

and

$$\begin{aligned} E_o = \lambda \log_2(e) e^{-\lambda T_o} (T_o + \frac{\Delta e^{-\lambda \Delta}}{1 - e^{-\lambda \Delta}}) + e^{-\lambda T_o} \\ - e^{-\lambda T_o} \log_2(1 - e^{-\lambda \Delta}) - (1 - e^{-\lambda T_o})[\log_2(1 - e^{-\lambda T_o}) - 1] \end{aligned} \tag{A.9}$$

Appendix B

Arithmetic Coding

Consider a source represented by a discrete random variable X, associated with probabilities $p_X(i)$. In arithmetic coding of a sequence of such a source, first the interval $[0, 1)$ is partitioned into cells. Each cell is associated with a source symbol and the size of the cell is proportional to the probability of the symbol. Figure B.1 shows such a partitioning for a 4-symbol source with $s1$, $s2$, $s3$ and $s4$ being the symbols. This partitioning can be repeated for each cell according to the sequence of the symbols.

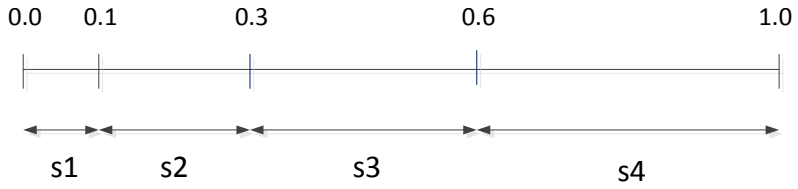


Fig. B.1 Partitioning the interval $[1, 0)$ for the source symbols $s1$,$s2$,$s3$ and $s4$.

Figure B.2 shows the repeated portioning for a sequence of 4 symbols: $s2$, $s4$, $s3$ and $s2$. In the end, the truncated binary representation of the last cell's midpoint is considered the codeword for the whole sequence. Consider the cumulative distribution function of the source associated with the cell thresholds x which can be defined as

$$F_X(x) = \sum_{i}^{n} p_X(i), \tag{B.1}$$

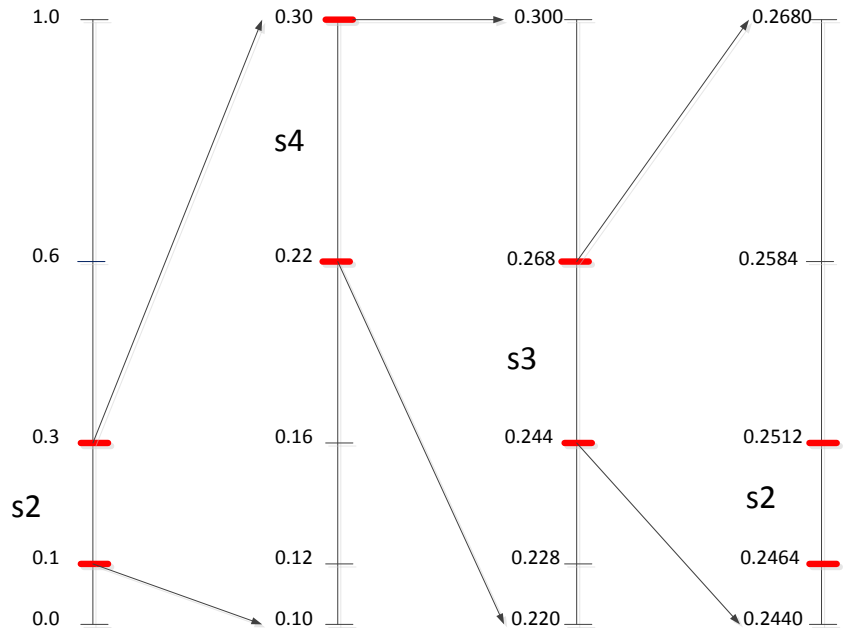


Fig. B.2 Arithmetic Coding for a 4-symbol source. The partitioning is repeated for each cell according to the sequence of the symbols, here $s2$, $s4$, $s3$ and $s2$.

where $p_X(i)$ is the probability associated with the i^{th} cell (and its corresponding symbol), n is the n^{th} cell that has the upper threshold x and $F_X(0) = 0$. We truncate the binary representation of the midpoint of a cell to get a l-bit representation. In order for this truncated value not to move to the lower cell, we should have

$$l = \min\{m : 2^{-m} < (F_x(x) - F_x(x-1))/2\}. \tag{B.2}$$

This gives

$$\begin{aligned} l &= \min\{m : m > -\log_2((F_x(x) - F_x(x-1))/2)\} \\ &= \lfloor -\log_2(F_x(x) - F_x(x-1))\rfloor + 1 \\ &= \lfloor -\log_2(p_X(n))\rfloor + 1. \end{aligned} \tag{B.3}$$

where $\lfloor . \rfloor$ is the floor function. As a sequence of symbols is coded, the new cells are generated with new thresholds (Fig. B.2). It can be easily shown again that to have a l-bit

binary representation of the last cell's midpoint,

$$l = \lfloor -\log_2(p_X(S)) \rfloor + 1, \tag{B.4}$$

where $p_X(S)$ is the probability of the sequence S associated with that cell. For the above example, we have

$$p_X(S) = p_X\{s2, s4, s3, s2\} = p_X(2).p_X(4).p_X(3).p_X(2). \tag{B.5}$$

Therefore

$$l = \lfloor -\log_2(p_X(2).p_X(4).p_X(3).p_X(2)) \rfloor + 1, \tag{B.6}$$

To decode the sequence, the decoder simply tracks the l-bit number it receives, from the original interval $[0, 1)$ to the last. For our example, the decoder receives a binary number which is within the interval $[0.2464, 0.2512)$. Comparing this number with the original interval $[0, 1)$, the decoder sees that this number is in the second cell associated with $s2$. Comparing with the second interval $[0.1, 0.3)$, $s4$ is decoded and so on.

References

[1] *Information Technology--Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to 1.5Mbits/sec, Part 3: Audio*, ISO/IEC JTC1/SC29 MPEG, 11172-3, 1993.

[2] *Information Technology–Generic Coding of Moving Pictures and Associated Audio, Part 7: Advanced Audio Coding*, ISO/IEC 13818-7, 1997.

[3] M. Bosi, K. Brandenburg, S. Quackenbush, L. Fielder, K. Akagiri, H. Fuchs, M. Dietz, J. Herre, G. Davidson, and Y. Oikawa, "ISO/IEC MPEG-2 Advanced Audio Coding," *J. Audio Eng. Soc.*, vol. 45, no. 10, pp. 789–814, Oct. 1997.

[4] M. Davis, "The AC-3 multichannel coder," in *Proc. 95th AES Conv.*, 1993.

[5] K. Akagiri, M. Katakura, H. Yamauchi, E. Saito, M. Kohut, M. Nishiguchi, and K. Tsutsui, "Sony systems," *The Digital Signal Processing Handbook*, pp. 43–1, 1998.

[6] K. Brandenburg and B. Gill, "First ideas on scalable audio coding," in *Proc. 97th AES Conv.*, Nov. 1994.

[7] D. Ning and M. Deriche, "A bitstream scalable audio coder using a hybrid WLPC-wavelet representation," in *Proc. IEEE ICASSP*, Apr. 2003, vol. 5, pp. V–417–V–420.

[8] H. Huang, H. Shu, and S. Rahardja, "Bit-plane arithmetic coding for Laplacian source," in *Proc. IEEE ICASSP*, Mar. 2010, pp. 3358–3361.

[9] T. Li, S. Rahardja, and S. N. Koh, "Frequency region-based prioritized bit-plane coding for scalable audio," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 16, no. 1, pp. 94–105, 2008.

[10] D. H. Kim, J. H. Kim, and S. W. Kim, "Scalable lossless audio coding based on MPEG-4 BSAC," in *Proc. 113th AES Conv.*, Oct. 2002, Paper Number: 5679.

[11] S. Kim, S. H. Park, and Y. B. Kim, "Fine grain scalability in MPEG-4 Audio," in *Proc. 111th AES Conv.*, Nov. 2001, Paper Number:5491.

[12] R. Yu, S. Rahardja, L. Xiao, and C. C. Ko, "A fine granular scalable to lossless audio coder," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 14, no. 4, pp. 1352–1363, Jul. 2006.

[13] A. Aggarwal, S. L. Regunathan, and K. Rose, "Optimal prediction in scalable coding of stereophonic audio," in *Proc. 109th AES Conv.*, Sep. 2000.

[14] A. Aggarwal, S.L. Regunathan, and K. Rose, "Asymptotically optimal scalable coding for minimum weighted mean square error," in *Proc. IEEE Data Compression. Conf.*, Mar. 2001, pp. 43–52.

[15] A. Aggarwal and K. Rose, "Approaches to improve quantization performance over the scalable Advanced Audio Coder," in *Proc. 112th AES Conv.*, Apr. 2002.

[16] E. Ravelli, V. Melkote, T. Nanjundaswamy, and K. Rose, "Cross-layer rate-distortion optimization for scalable Advanced Audio Coding," in *Proc. 128th AES Conv.*, May 2010, pp. 365–368.

[17] E. Ravelli, V. Melkote, T. Nanjundaswamy, and K. Rose, "Joint optimization of base and enhancement layers in scalable audio coding," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 21, no. 4, pp. 711–724, 2013.

[18] E. Ravelli, V. Melkote, T. Nanjundaswamy, and K. Rose, "Joint optimization of the perceptual core and lossless compression layers in scalable audio coding," in *Proc. IEEE ICASSP*, Mar. 2010, pp. 365–368.

[19] S. Regunathan, K. Rose, and A. Aggarwal, "Compander domain approach to scalable AAC," in *Proc. 110th AES Conv.*, May 2001.

[20] A. Aggarwal and K. Rose, "A conditional enhancement-layer quantizer for the scalable MPEG Advanced Audio Coder," in *Proc. IEEE ICASSP*, May 2002, vol. 2, pp. II–1833–II–1836.

[21] A. Aggarwal, S. U. Ryu, and K. Rose, "Efficient scalable coding of stereophonic audio by conditional quantization and estimation-theoretic prediction," in *Proc. IEEE ICASSP*, Apr. 2003, vol. 5, pp. V–465–V–468 vol.5.

[22] B. Edler, "Technical description of the MPEG-4 audio coding proposal from University of Hannover and Deutsche Bundespost telekom," *ISO/IEC, JTC1/SC29/WG11 MPEG95/MO414*, 1995.

[23] R. Yu, C. C. Ko, S. Rahardja, and X. Lin, "Bit-plane Golomb coding for sources with Laplacian distributions," in *Proc. IEEE ICASSP*, Apr. 2003, vol. 4, pp. 277–80.

[24] *MPEG-4 Audio Verification Test Results: Audio on Internet*, ISO/IEC JTC1/SC29/WG11 MPEG98/N2425, Oct. 1998.

[25] M. Movassagh, J. Thiemann, and P. Kabal, "Joint entropy-scalable coding of audio signals," in *Proc. IEEE ICASSP*, Mar. 2012, pp. 2961–2964.

[26] M. Movassagh and P. Kabal, "Scalable audio coding using watermarking," in *Proc. IEEE Int. Conf. Multimedia and Expo (ICME)*, 2013, pp. 1–6.

[27] M. Movassagh and P. Kabal, "New bit-plane probability calculations for scalable to lossless audio coding," in *Proc. IEEE ICASSP*, May 2014, pp. 3675–3679.

[28] N. Jayant and P. Noll, *Digital Coding of Waveforms: Principles and Applications to Speech and Video*, Englewood Cliffs, NJ, 1984.

[29] R. M. Gray and D. L. Neuhoff, "Quantization," *IEEE Trans. Inform. Theory*, vol. 44, no. 6, pp. 2325–2383, Oct. 1998.

[30] A. Aggarwal, S. L. Regunathan, and K. Rose, "Efficient bit-rate scalability for weighted squared error optimization in audio coding," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 14, no. 4, pp. 1313–1327, Jul. 2006.

[31] Y. Linde, A. Buzo, and R. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. 28, no. 1, pp. 84–95, 1980.

[32] J. Makhoul, S. Roucos, and H. Gish, "Vector quantization in speech coding," *Proc. IEEE*, vol. 73, no. 11, pp. 1551–1588, 1985.

[33] D. Paul, "A 500-800 bps adaptive vector quantization vocoder using a perceptually motivated distance measure," in *Proc. IEEE Conf. Global Telecoms. (GLOBCOM)*, 1982, vol. 1, pp. 1079–1082.

[34] V. Cuperman, "On adaptive vector transform quantization for speech coding," *IEEE Trans. Commun.*, vol. 37, no. 3, pp. 261–267, 1989.

[35] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.

[36] T. Cover and J. Thomas, *Elements of Information Theory*, John Wiley & Sons, 2012.

[37] D. Huffman, "A method for the construction of minimum redundancy codes," *Proc. IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[38] J. Rissanen, "Generalized Kraft inequality and arithmetic coding," *IBM Journal of research and development*, vol. 20, no. 3, pp. 198–203, 1976.

[39] I. Witten, R. Neal, and J. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, vol. 30, no. 6, pp. 520–540, 1987.

[40] T. Berger, *Rate Distortion Theory: A Mathematical Basis for Data Compression*, Prentice-Hall, 1971.

[41] J. Johnston, "Estimation of perceptual entropy using noise masking criteria," in *Proc. IEEE ICASSP*, Apr. 1988, vol. 5, pp. 2524–2527.

[42] R. McAulay and T. F. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 34, no. 4, pp. 744–754, 1986.

[43] X. Serra and J. Smith, "Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition," *Computer Music Journal*, pp. 12–24, 1990.

[44] A. Spanias, "A hybrid transform method for analysis/synthesis of speech," in *Signal Processing*, Dec. 1991, vol. 24, pp. 217–229.

[45] E. B. George and M. Smith, "A new speech coding model based on a least-squares sinusoidal representation," in *Proc. IEEE ICASSP*, Apr. 1987, vol. 12, pp. 1641–1644.

[46] E. B. George and M. Smith, "Perceptual considerations in a low bit rate sinusoidal vocoder," in *Proc. IEEE Int. Phoenix Conf. on Comp. and Commun.*, 1990, pp. 268–275.

[47] E. B. George and M. Smith, "Analysis-by-synthesis/overlap-add sinusoidal modeling applied to the analysis and synthesis of musical tones," *J. Audio Eng. Soc*, vol. 40, no. 6, pp. 497–516, 1992.

[48] *Information Technology--Generic Coding of Moving Pictures and Associated Audio: Audio (non-backwards compatible coding, NBC)*, ISO/IEC JTC1/SC29/WG11 MPEG, MPEG-2 NBC/AAC, 1996.

[49] *MPEG-4 Audio Committee Draft 14496-3*, ISO/IEC JTC1/SC29/WG11 N1903, Oct. 1997.

[50] G. Stroll, M. Link, and G. Theile, "Masking-pattern adapted subband coding: Use of the dynamic bit-rate margin," in *Proc. 84th AES Conv.*, 1988.

[51] R. Veldhuis, M. Breeuwer, and R. Van D. Waal, "Subband coding of digital audio signals without loss of quality," in *Proc. IEEE ICASSP*, May 1989, pp. 2009–2012.

[52] G. Theile, M. Link, and G. Stoll, "Low-bit rate coding of high quality audio signals," in *Proc. 82nd AES Conv.*, 1987.

[53] J. Johnston, D. Sinha, S. Dorward, and S. Quackenbush, "AT&T perceptual audio coding (PAC)," in *Audio Engineering Society Conference: Collected Papers on Digital Audio Bit-Rate Reduction*, 1996.

[54] M. G. Bellanger, G. Bonnerot, and M. Coudreuse, "Digital filtering by polyphase network: Application to sample-rate alteration and filter banks," *IEEE Trans. Acoustic., Speech, Signal Process.*, vol. 24, no. 2, pp. 109–114, 1976.

[55] J. Rothweiler, "Polyphase quadrature filters– a new subband coding technique," in *Proc. IEEE ICASSP*, Apr. 1983, vol. 8, pp. 1280–1283.

[56] K. Brandenburg and J. D. Johnston, "Second generation perceptual audio coding: the hybrid coder," in *Proc. 88th AES Conv.*, 1990.

[57] H. Malvar, "Lapped transforms for efficient transform/subband coding," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 38, no. 6, pp. 969–978, 1990.

[58] T. Ramstad and J. Tanem, "Cosine-modulated analysis-synthesis filterbank with critical sampling and perfect reconstruction," in *Proc. IEEE ICASSP*, Apr. 1991, pp. 1789–1792.

[59] H. Malvar, "Modulated QMF filter banks with perfect reconstruction," *IET Electronics Letters*, vol. 26, no. 13, pp. 906–907, 1990.

[60] R. Koilpillai and P. P. Vaidyanathan, "New results on cosine-modulated FIR filter banks satisfying perfect reconstruction," in *Proc. IEEE ICASSP*, Apr. 1991, pp. 1793–1796.

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

[62] R. Geiger, T. Sporer, J. Koller, and K. Brandenburg, "Audio coding based on integer transforms," in *Proc. 111th AES Conv.*, 2001.

[63] T. Li, S. Rahardja, R. Yu, and S.N. Koh, "On integer MDCT for perceptual audio coding," *IEEE Trans. Audio, Speech, and Lang. Process.*, vol. 15, no. 8, pp. 2236–2248, 2007.

[64] R. Geiger, R. Yu, J. Herre, S. Rahardja, S. W. Kim, X. Lin, and M. Schmidt, "ISO/IEC MPEG-4 high-definition scalable Advanced Audio Coding," *J. Audio Eng. Soc.*, vol. 55, no. 1/2, pp. 27–43, 2007.

[65] E. Allamanche, R. Geiger, J. Herre, and T. Sporer, "MPEG-4 low delay audio coding based on the AAC codec," in *Proc. 106th AES Conv.*, 1999.

[66] A. Gersho and R. M. Gray, *Vector quantization and signal compression*, Boston, MA: Kluwer, 1993.

[67] A. Gersho, "Asymptotically optimal block quantization," *IEEE Trans. Inform. Theory*, vol. 25, no. 4, pp. 373–380, Jul. 1979.

[68] P. A. Chou, T. Lookabaugh, and R. M. Gray, "Entropy-constrained vector quantization," *IEEE Trans. Acoust. Speech and Signal Process.*, vol. 37, no. 1, pp. 31–42, Jan. 1989.

[69] G. J. Sullivan, "Efficient scalar quantization of exponential and Laplacian random variables," *IEEE Trans. Inform. Theory*, vol. 42, no. 5, pp. 1365–1374, Sep. 1996.

[70] Jia Li, N. Chaddha, and R. M. Gray, "Asymptotic performance of vector quantizers with a perceptual distortion measure," *IEEE Trans. Inform. Theory*, vol. 45, no. 4, pp. 1082–1091, May 1999.

[71] S. H. Park, Y. B. Kim, and Y. S. Seo, "Multi-layer bit-sliced bit-rate scalable audio coding," in *Proc. 103th AES Conv.*, Sep. 1997.

[72] S. Strahl, H. Hansen, and A. Mertins, "A dynamic fine-grain scalable compression scheme with application to progressive audio coding," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 19, no. 1, pp. 14–23, 2011.

[73] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 243–250, Jun. 1996.

[74] C. Dunn, "Efficient audio coding with fine-grain scalability," in *Proc. 111th AES Conv.*, Nov. 2001.

[75] B. Leslie, C. Dunn, and M. Sandler, "Developments with a zerotree audio codec," in *Proc. 17th AES Int. Conf.: HQ Audio Coding*, Aug. 1999.

[76] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H. 264/AVC standard," *IEEE Trans. Circuit Syst. Video Technol.*, vol. 17, no. 9, pp. 1103–1120, 2007.

[77] J. Han, V. Melkote, and K. Rose, "An estimation-theoretic framework for spatially scalable video coding," *IEEE Trans. Img. Process.*, vol. 23, no. 8, pp. 3684–3697, 2014.

[78] C. A. Segall and G. J. Sullivan, "Spatial scalability within the H. 264/AVC scalable video coding extension," *IEEE Trans. Circuit Syst. Video Technol.*, vol. 17, no. 9, pp. 1121–1135, 2007.

[79] S.-H. Kim and Y.-S. Ho, "Fine granular scalable video coding using context-based binary arithmetic coding for bit-plane coding," *IEEE Trans. Circuit Syst. Video Technol.*, vol. 17, no. 10, pp. 1301–1310, 2007.

[80] S. Golomb, "Run-length encoding," *IEEE Trans. Inform. Theory*, vol. 12, no. 3, pp. 399–401, Jul. 1966.

[81] R. Gallager and D. Van Voorhis, "Optimal source codes for geometrically distributed integer alphabets (corresp.)," *IEEE Trans. Inform. Theory*, vol. 21, no. 2, pp. 228–230, 1975.

[82] *Information Technology--Generic Coding of Moving Pictures and Associated Audio, Part 3: Audio (MPEG-2 BC-LSF)*, ISO/IEC JTC1/SC29/WG11 MPEG, IS13818-3, 1994.

[83] A. Spanias, T. Painter, and V. Atti, *Audio Signal Processing and Coding*, Wiley, 2007.

[84] D. Schulz, "Improving audio codecs by noise substitution," *J. Audio Eng. Soc.*, vol. 44, no. 7/8, pp. 593–598, 1996.

[85] J. Ojanpera, M. Vaananen, and L. Yin, "Long term predictor for transform domain perceptual audio coding," in *Proc. 107th AES Conv.*, Sep. 1999.

[86] N. Iwakami and T. Moriya, "Transform-domain weighted interleave vector quantization (TwinVQ)," in *Proc. 101st AES Conv.*, Nov. 1996.

[87] N. Iwakami, T. Moriya, A. Jin, T. Mori, and K. Chikira, "Fast encoding algorithms for MPEG-4 TwinVQ audio tool," in *Proc. Workshop and Exhibition on MPEG-4*, 2001, pp. 1–4.

[88] Y. T. Hwang, N. J. Liu, and M. C. Tsai, "An MPEG-4 Twin-VQ based high quality audio codec design," in *Proc. IEEE Workshop on Signal Process. Syst.*, 2001, pp. 321–331.

[89] *Information Technology—Coding of Audio-Visualf Objects—Part 3: Audio—Amd. 3: Scalable Lossless Coding (SLS)*, ISO/IEC std. ISO/IEC JTC1/SC29, 14496-3:2005/Amd.3:2006, 2006.

[90] E. Ravelli, V. Melkote, and K. Rose, "A perceptually enhanced scalable-to-lossless audio coding scheme and a trellis-based approach for its optimization," in *Proc. IEEE WASPAA*, Oct. 2009, pp. 329–332.

[91] H. Fastl and E. Zwicker, *Psychoacoustics: Facts and Models*, vol. 22, Springer Science & Business Media, 2007.

[92] D. Rocchesso, *Introduction to Sound Processing*, Mondo estremo, 2004.

[93] B. C. J. Moore, *An Introduction to the Psychology of Hearing*, Brill, 2012.

[94] E. Terhardt, "Calculating virtual pitch," *Hearing research*, vol. 1, no. 2, pp. 155–182, 1979.

[95] M. R. Schroeder, B. S. Atal, and J. L. Hall, "Optimizing digital speech coders by exploiting masking properties of the human ear," *J. Acoust. Soc. Am.*, vol. 66, no. 6, pp. 1647–1652, 1979.

[96] B. Chen and G. W. Wornell, "Quantization index modulation: a class of provably good methods for digital watermarking and information embedding," *IEEE Trans. Inform. Theory*, vol. 47, no. 4, pp. 1423–1443, May 2001.

[97] R. Yu, X. Lin, S. Rahardja, and C. C. Ko, "A statistics study of the MDCT coefficient distribution for audio," in *Proc. IEEE Int. Conf. Multimedia and Expo (ICME)*, Jun. 2004, vol. 2, pp. 1483–1486.

[98] *Information Technology—Coding of Audio-Visual Objects—Part 3: Audio—Subpart 4: General Audio Coding (GA)*, ISO/IEC std. ISO/IEC JTC1/SC29, 14496-3:2005, 2005.

[99] *Method for Objective Measurements of Perceived Audio Quality (PEAQ)*, ITU-R Rec. BS.1387-1, 2001.

[100] P. Kabal, Audio File Programs and Routines, [online]. Available: http://www-mmsp.ece.mcgill.ca/Documents/Downloads/AFsp/.

[101] T. Verma and T. Meng, "A scalable entropy code," in *Proc. IEEE Data Compression Conf. (abstract)*, Mar. - Apr. 1998, p. 581.

[102] S. Mohajer, P. Pakzad, and A. Kakhbod, "Tight bounds on the redundancy of Huffman codes," *IEEE Trans. Inform. Theory*, vol. 58, no. 11, pp. 6737–6746, Nov. 2012.

[103] Jr. Forney, G.D., "The Viterbi algorithm," *Proc. IEEE*, vol. 61, no. 3, pp. 268–278, Mar. 1973.

[104] European Broadcasting Union, "Sound Quality Assessment Material Recordings for Subjective Tests," EBU SQAM CD [Online] Available: https://tech.ebu.ch/publications/sqamcd, Oct. 2008.

[105] "Call for Proposals on MPEG-4 Lossless Audio Coding,", ISO/IEC JTC1/SC29/WG11 (MPEG), N5040, Shanghai, China, Oct. 2002.

[106] *Method for the Subjective Assessment of Intermediate Quality Level of Coding Systems*, ITU-R Rec. BS.1534-1, 2003.

[107] P. Noll and R. Zelinski, "Bounds on quantizer performance in the low bit-rate region," *IEEE Trans. Commun.*, vol. 26, no. 2, pp. 300–304, 1978.