Flexible Audio Coder

MARCUS ASTEBORG

KTH Electrical Engineering

Master Thesis
Stockholm, Sweden May 2011
XR-EE-SIP 2011:007

Abstract

As modern communications networks are heterogeneous and, therefore, highly variable, the design of source coders should reflect this network variability. It is desired that source coders are able to adapt instantly to any bit-rate constraint. Source coders that possess this property offer coding flexibility that facilitates optimal utilization of the available communication channel within a heterogeneous network. Flexible coders are able to utilize feedback information and therefore perform an instant re-optimization. This property implies that flexible audio coders are better suited for networks with high variability due to the fact a single configuration of the flexible coder can operate at continuum of bit-rates.

The aim of the thesis is to implement a flexible audio coder in a real-time demonstrator (VoIP application) that facilitates instant re-optimization of the flexible coding scheme. The demonstrator provides real-time full-duplex communications over a packet network and the operating bit-rate may be adjusted on the fly. The coding performance of the flexible audio coding scheme should remain comparable to non-flexible schemes optimized at their operating bit-rates.

The report provides a background for the thesis work and describes the real-time implementation of the demonstrator. Finally, test results are provided. The coder is evaluated by means of a subjective MUSHRA test. The performance of the flexible audio coder is compared to relevant state-of-the-art codecs.

Acknowledgements

First of all I would like to thanks to my supervisor Janusz Klejsa. Janusz has been the foundation of this work and I am deeply grateful for all the knowledge he has shared. Janusz has give me support during late weekend nights and tirelessly answered my questions.

I would also like to thank my examiner Professor Bastiaan Kleijn and Minyue Li for the opportunity to do this thesis. Finally, all the love in the world to my dear Carolina whom have always supported me whatever I have chosen to do in my life.

List of Abbreviations

ACELP	Algebraic Code Excited Linear Prediction
ACR	Absolute Category Rating
AEC	Acoustic Echo Canceller
AMR-WB	Adaptive Multi-Rate Wideband
AR	Autoregressive
BWE	Bandwidth Extension
CDF	cumulative distribution function
CELP	Code Excited Linear Prediction
DDNSQ	Delayed Decision Noise Shaping Quantizer
DPQ	Distribution Preserving Quantization
erf	error function
EVD	Eigenvalue Decomposition
FAC	Flexible Audio Coder
FEC	Forward Error Correction
GMM	Gaussian Mixture Model
iid	Independent, Identically Distributed
IP	Internet Protocol
ISP	Immittance Spectral Pair
kbps	Kilobit Per Second
KLT	Karhunen-Loève Transform
LAN	Local Area Network

LP	Linear Prediction
LPC	Linear Prediction Coefficients
LTP	Long Term Prediction
MDCT	Modified Discrete Cosine Transform
MLT	Modulated Lapped Transform
MOS	Mean Opinion Score
MSE	Mean Square Error
MUSHRA	MUltiple Stimuli with Hidden Reference and Anchor
pdf	probability density function
PEAQ	Perceptual Evaluation of Audio Quality
PESQ	Perceptual Evaluation of Speech Quality
RTP	Real-Time Protocol
S-MSVQ	Split-Multistage Vector Quantization
SNR	signal-to-noise ratio
SVD	Singular Value Decomposition
SWB	Super Wideband
TCP	Transmission Control Protocol
TDAC	Time-Domain Aliasing Cancellation
TDBWE	Time-Domain Bandwidth Extension
TFC	Transform Coding
VoIP	Voice over IP
VQ	Vector Quantizer
WB	Wideband

Contents

Chapter 1

Introduction

As modern communications networks are heterogeneous and, therefore, highly variable, the design of source coders should reflect this network variability. It is desired that source coders can adapt instantly to any bit-rate constraint. Source coders that possess this property offer coding flexibility that facilitates optimal utilization of the available communication channel within a heterogeneous network. Common audio coding schemes are based on the source coding methods that are in generally non-flexible. As a result they perform sub-optimally when the properties of the communication channel are varying. This drawback of the traditional audio coding approaches is addressed by flexible audio coding.

Heterogeneous networks typically provide feedback channels that can be used to transmit auxiliary information to the encoder, for instance, about the available bit-rate or a packet-loss rate. Non-flexible coding schemes are not able to utilize this information fully as they only have a finite number of operating modes associated with a final set of bit-rates and possible coder configurations. This often results in sub-optimal performance, as it is common that the coder is faced with operating conditions that are different than the conditions that it was designed for in a heterogeneous network. Flexible coders are able to utilize feedback information and therefore perform an instant re-optimization. This property implies that flexible audio coders are better suited for networks with high variability due to the fact a single configuration of the flexible coder can operate at continuum of bit-rates.

In the remainder of this chapter, an overview of the state-of-the-art audio coding for communication purposes is presented. The overview provides background information and explains the thesis motivation in more detail. Finally the thesis objective and the outline is stated.

1.1 Audio Coding for Real-time Communications

This section focuses on providing an introduction to real-time audio communications. In particular, wideband (WB) monophonic schemes are discussed as they are essential components of the state-of-the-art audio communication systems. These schemes may be further extended also to super wideband (SWB) coding scenarios and to multi-channel cases (such extensions are beyond the scope of this thesis).

A real-time audio communication system consists of several functional blocks, with the audio coding algorithm as one of them. A high level view of the real-time audio communication system can be seen in Figure 1.1.

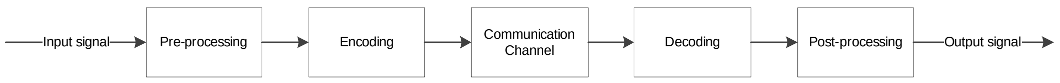


Figure 1.1: High-level diagram of communication system.

An input signal is first sampled from a microphone to obtain a discrete-time representation. This operation is usually followed by several pre-processing steps. The pre-processing can include an acoustic echo canceller (AEC), noise reduction, resampling and equalization. After the signal is preprocessed, it is encoded into a bit-stream that will be transmitted through a communications network. The encoding is implemented by a dedicated encoder of the audio coder that is used. Depending on the coding scenario, the bit-stream is divided into packets or frames that are transmitted through the communication channel to the receiver. At the receiver, the frames can be buffered and sorted by a jitter-buffer if the frames are not received in order. In internet protocol (IP) networks this corresponds to the UDP transport protocol, as with TCP-IP the packet ordering and the forward error correction (FEC) is handled by the transport layer. The received bit-stream is then decoded by the decoder block of the audio coding algorithm. Decoding may be followed by post-processing which can include equalization, signal level adjustment or enhancement (e.g., noise reduction). Finally, the post-processed signal is ready to be played out.

The performance of the audio coding schemes is to a large extent determined by properties of a communication channel (e.g., packet loss rate, available bit-rate). A state-of-the-art audio coder should be able to utilize the available communication channel in a best possible way. State-of-the-art audio coders, are, in general, not flexible. This implies that they facilitate only a rate-distortion trade-off within a limited set of operating bit-rates. Usually, it is difficult to obtain a coding algorithm that remains efficient for a wide range of operating bit-rates. Instead, existing codecs are highly tuned for their respective bit-rates. This leads to a situation where there are several state-of-the-art codecs that co-exist and address different regions of a practical range of operating bit-rates. In this thesis we focus on coding scenarios, where operating bit-rate falls between 0.5 and 2 bits per sample. The state-of-the-art audio coders that are relevant from the point of view of this thesis are gathered and summarized in Table 1.1 where the bit-rate is given in kilobit per second (kbps).

G.722.1 is a low-complexity codec which has two operating bit-rates, 24 kbps and 32 kbps [1]. This codec is transform based and uses the modulated lapped transform (MLT) [2]. It uses overlapped frames and a fixed transform. To achieve

Table 1.1: Wide-band real-time communication codecs

Codec	G.722.1	G.722.2/AMR-WB	G.729.1	Opus
Standard	ITU-T	ITU-T/3GPP	ITU-T	IETF
Date	1999	2001	2006	2011
Bit-rate (kbps) Type	24, 32 Configurable	6.6 - 24 Configurable	14 - 32 Configurable	8 - 96 Variable
Frame (ms)	20	20	20	10 - 60
Look Ahead (ms)	20	5	29	5
Bandwidth (kHz)	7	7	4 / 7	4 - 48
Key Technology	TFC LC	ACELP BWE	ACELP TFC TDBE TDAC	DDC TFC Stereo

efficiency, a considerable coding delay is required (40 ms). The main goal for this coder is to provide low-complexity coding and this is achieved at a price of coding efficiency and flexibility.

Another important codec is G.722.2 from ITU-T, which is also known as adaptive multi-rate wideband (AMR-WB) in the 3GPP standardization [3, 4]. AMR-WB has nine operating bit-rates and has an internal sampling frequency of 12.8 kHz, which produces an effective bandwidth of 6.4 kHz. To obtain good coding performance for wide-band signals, the missing part of the signal bandwidth must be reconstructed. Therefore, in the highest supported bit-rate of 23.85 kbps, the coder uses bandwidth extension (BWE) which extends the bandwidth from 6.4 to 7 kHz [5]. The key technology of the AMR-WB codec is the code excited linear prediction (CELP), which is implemented by means of an algebraic code yielding the algebraic code excited linear prediction (ACELP) scheme [6]. This coder provides the state-of-the-art coding quality of around 16 kbps. It does not, however, provide flexible rate-distortion coding. Extending the coding approach of AMR-WB to other operating bit-rates is non-trivial and requires a considerable effort. AMR-WB is therefore unable to fully adjust to the available bit-rate in a heterogeneous network, especially if the adaptation is desired to be performed in an on-the-fly manner.

AMR-WB also has an extension which is called AMR-WB+ [7]. The extended version of AMR-WB was standardized in the course of 3GPP activities. The extension supports bit-rates up to 36 kbps, has stereo capabilities and is interoperable with AMR-WB. The main advantage of AMR-WB+ is that it provides superior quality compared to AMR-WB. This is possible by the introduction of several coding modes, where the coder can use different coding strategies (CELP, or transform coding (TFC)). As the algorithmic delay of this coder is between 60 – 90 ms, it is not feasible for real-time communications.

Another codec is G.729.1 [8], which is an embedded speech and audio codec that supports interoperability with G.729. The embedded coding within G.729.1 generates a structured bit-stream that consists of layers corresponding to different quality levels. The bitstream is generated in such a way that the bit-rate can easily be reduced by stripping off bits from the bit-stream by reducing the number of transmitted layers. The first layer is the core layer that corresponds to the legacy G.729 format which is at 8 kbps in a narrow-band scenario. The second layer enhances the narrow-band quality and operates at 12 kbps. The third layer uses time-domain bandwidth extension (TDBWE) [8] to extend the bandwidth to wide-band and operates at 14 kbps. The following layers uses time-domain aliasing cancellation (TDAC) [8] and add 2 kbps every layer up to the maximum bit-rate of 32 kbps. G.729.1 offers a set of operating bit-rates on its rate-distortion performance, which implies a combination of different techniques making the extension to the continuous rate-distortion trade-off case non-trivial. However, as the use of TDAC that uses a modified discrete cosine transform (MDCT) [9] for analysis, 20 ms of overhead is added, which is not ideal for low-delay real-time communications.

Opus is a new state-of-the-art flexible embedded codec that is currently under evaluation for inclusion of a IETF standard [10]. The Opus codec is a combination of SILK [11] and CELT [12]. Opus has three main operation modes; speech, audio and automatic. This codec is flexible since it can operate over a wide range of bit-rates ranging from 6 kbps narrow-band up to 128 kbps full-band. At lower bit-rates, SILK is used to code the lower band (up to 8 kHz) of the frequency spectrum and when the bit-rates allows, the higher band of the spectrum is coded with CELT resulting in a so-called hybrid mode. SILK uses a flexible quantization method which is called delayed decision noise shaping quantizer (DDNSQ) that provides very good speech performance at lower rates. The SILK coder is very efficient for speech signals, however its performance depends on the type of signal. For bit-rates lower than 1 bits/sample it is not efficient for audio. The audio mode is therefore addressed by the CELT codec. The audio mode, which only uses the CELT codec, is comprised of a transform based codec which is based on the usage of MDCT. Moreover, the audio mode also support stereo. When CELT is used, the operating bit-rate is restricted to be not less than 32 kbps in order to maintain the efficiency of the energy envelope preservation.

The combination of different coding strategies for different types of signal is certainly efficient in terms of coding quality, however it leads to complicated coding schemes that are very hard to optimize for any specific bit-rate. Recently, flexible audio coding was proposed with a goal to provide the state-of-the-art over a continuous range of operating bit-rates using a tractable and consistent coding method [13, 14].

1.2 Thesis Objective and Outline

The main focus of this thesis is to implement a framework for the operation of an audio coder that uses a flexible audio coding scheme. The framework should demonstrate the rate-distortion trade-off that is performed in a flexible manner. Specifically, the implementation task includes generation of a bit-stream, packetization, transmission through a network, decoding and play-out. A simple network model is assumed, where there are no packet losses and the transmission delay is constant. This model is not realistic, but still provides means for evaluation of coder flexibility. The usage of a more accurate network model is beyond the scope of this thesis.

The goal for this thesis is to implement a flexible audio coder scheme that is based on the work of [14, 13]. This flexible audio coding is achieved by means of statistical modelling of the audio signal and by utilizing distribution preserving quantization (DPQ) [15].

The implementation should demonstrate the following properties of the flexible audio coder:

- real-time operation,
- instant re-optimization of the scheme,
- coding performance comparable to non-flexible schemes optimized at their operating bit-rates.

In addition to these properties, the benefit of the DPQ approach may be evaluated by comparing it to a classical entropy-constrained quantization. This is implemented by providing the user with the possibility to choose between DPQ and the entropy constrained quantization operating at a variable bit-rate. This implementation work consists of:

- designing a voice over IP (VoIP) application to utilize the flexible audio codec,
- implementing instances of audio encoder and decoder,
- implementing the bit-stream generation (entropy coder and entropy decoder),
- designing and implementing a demonstration specific real-time protocol (RTP).

This thesis is organized as follows.

Chapter 2 introduces fundamental concepts in audio coding using source coding tools. As the thesis focuses on the flexible audio coding, some important source coding concepts that facilitate the flexible-rate distortion trade-off are described. Finally, a basic introduction to flexible audio coding is presented.

Chapter 3 provides a method for flexible audio coding which is then followed by a high level architecture view of such a codec. The chapter is then summarized with a detailed description of how flexible audio coding methods are implemented in an audio codec.

Chapter 4 gives an overview of the implementation of a real-time demonstrator that was created during the work on this thesis.

In Chapter 5, the real-time implementation is evaluated by means of objective and subjective testing and the obtained results are discussed in detail.

Chapter 6 provides some final conclusions and discusses possible improvements.

Chapter 2

Fundamental Concepts in Audio Coding

This chapter contains a short introduction to audio coding. The presentation focuses on common techniques used in real-time audio coding for communication and, in particular, on these aspects that are relevant to flexible audio coding.

In the first section, a brief discussion on the fundamental concepts in source coding is made. These concepts are then related to their applications in audio coding. The discussion provides a technical background to the implementation task of this thesis. The chapter is summarized by providing audio coding attributes that are important from the point of view of designing a flexible audio coder.

2.1 Source Coding Background

In this section it is assumed that the signals are discrete-time with discrete amplitudes (unless otherwise stated). This means that the analog signal is first sampled and a discrete-time representation is obtained. The discrete-time signal with samples taking continuous values is then quantized to yield discrete-valued signal. Since the discrete-time is used, all the quantized samples may be indexed by a time index i. A value of the considered signal may be then seen as a time-indexed random variable X_i. In this section it is assumed that the signal is stationary.

The first useful information-theoretic quantity that characterizes X_i is the entropy. Assume that X_i can take any value x from the discrete set of possible values $\mathcal{A}$ with a probability $p_{X_i}(x) = P(X_i = x)$. The entropy of X_i is defined as

$$H(X_i) = -\sum_{x\in\mathcal{A}} p_{X_i}(x)\log_2(p_{X_i}(x)) \tag{2.1}$$

and specified in bits. The entropy specifies how many bits is required to reconstruct the process X_i without any loss of information. One may also consider blocking the consecutive realizations of X_i (with respect to the time index i) into a k-dimensional random vector $X_i^k = [X_{i-k}, \cdots, X_i]^T$ (assuming that the past samples are available). In this case the entropy can be defined as joint entropy

$$H(X_i^k) = -\mathbb{E}\left\{\log_2 p_{X_{i-k},\ldots,X_i}(X_{i-k}, \ldots, X_i)\right\}, \tag{2.2}$$

where time indices lower than k indicate previously observed scalar values, $\mathbb{E}\{\cdot\}$ denotes an expectation and $p_{X_{i-k},\ldots,X_i}$ denotes a joint pdf.

This first-order entropy in equation 2.1 does not account for dependency between samples. The joint entropy in equation 2.2 may be used to define a k-th order entropy $H_k(X_i^k) = \frac{1}{k}H(X_i^k)$. As the block processing of samples is a typical approach in coding, it is convenient to define the entropy rate

$$H_\infty \equiv \lim_{k\to\infty} \frac{1}{k} H(X_{i-k}, \cdots, X_i), \tag{2.3}$$

where the $k \to \infty$ is the dimension of the vector or block.

If the samples are independent, identically distributed (iid) the entropy rate the block is $H_\infty(X_i) = H_1(X_i)$. This means that if the samples are iid within the block, there is no redundancy between samples. This property may be evaluated by means of the redundancy rate

$$\rho(X_i) = H_1(X_i) - H_\infty(X_i). \tag{2.4}$$

The definition of the redundancy in equation 2.4 is considered on sample-by-sample basis. As mentioned earlier, block processing is more common in practice and therefore a useful parameter is the total redundancy rate that is defined by

$$\rho_T(X_i) = L - H_\infty(X_i), \tag{2.5}$$

where L is the average bitrate per sample. In equation 2.4 the block size is $m = 1$, however if instead the length is considered to be m the redundancy can be redefined as the inter-block redundancy

$$\rho_m(X_i) = H_m(X_i) - H_\infty(X_i). \tag{2.6}$$

Another redundancy rate is the dependencies by samples in the block and have the notation intra-block redundancy rate

$$\rho_c(X_i) = L - H_m(X_i), \tag{2.7}$$

These two combined gives the total redundancy rate

$$\rho_T(X_i) = \rho_m(X_i) + \rho_c(X_i). \tag{2.8}$$

The equation 2.8 describes the average redundancy within a coding block that should be exploited by a coding algorithm.

Another important property from the point of view of source coding is related to the fact that some distortion might be tolerated in the received signal. This property may be exploited and leads to the lossy coding paradigm that is commonly used in audio and video coding. The lossy coding is based on irrelevancy removal, lossy coding is necessary when the rate budget is restricted and perfect reconstruction is not possible. It exploits the fact that some information that is not needed for the receiver to interpret the signal may be removed leading to an acceptable level of distortion. This is known as the irrelevancy rate

$$\zeta(X_i) = H_\infty(X_i) - R(D), \tag{2.9}$$

where $R(D)$ is the rate-distortion bound.

The rate-distortion bound $R(D)$ is a function that associates the number of bits R needed to represent the coded signal and the distortion D, where the distortion represents the difference between the original signal and the reconstructed signal. Since in the context of audio coding the objective is to minimize the perceptually perceived distortion, an appropriate distortion measure that evaluates the difference should be used. However these perceptually-relevant distortion measures are often complex, so instead a simple sample based distortion criterion such as the mean square error (MSE) is used which is known to be relevant to the perception only at very low distortions. To make the MSE efficient as a distortion measure, the signal is companded into the perceptually weighted domain.

$R(D)$ provides a theoretical lower bound of the compression, based on the distortion and probability density function (pdf) of the source. The $R(D)$ serve as a guideline while designing a practical source coder but does not provide any practical tools to achieve this bound. As a description of the general rate-distortion function is unnecessary for this thesis, only the specific case with a Gaussian distributed variable X is reiterated

$$R(D) = \frac{1}{2}\log_2(\frac{\sigma_x^2}{D}), D \leq \sigma_x^2, \tag{2.10}$$

where σ_x^2 is the variance of the random variable X.

This concludes the short discussion about basic source coding concepts. These concepts are useful in the contexts of lossy and lossless coding.

2.1.1 Lossy Coding

The main purpose of irrelevancy removal (lossy coding) is to minimize the transmission of irrelevant information. This is usually achieved by appropriate design of quantizers. In the remainder of this section the formulation of two optimality criteria for signal quantizers that are particularly useful in source coding is provided.

Scalar quantization is a simple form of lossy coding. It is used to encode scalar continuous random variables. This is represented by a mapping $\mathcal{Q}$ of a real line $\mathbb{R}$ to a codebook $\mathcal{C} = \{c_i\}_{i\in\mathcal{I}}$, where $\mathcal{I}$ is a set of indices. The mapping is often performed in two steps. First, an encoding step assigns an index i to a point representing the value to be quantized by means of $\mathcal{Q}_e$. The index can be converted into a bit-stream or a binary codeword. It can be further stored or transmitted. The index can be decoded into a reconstruction point c_i by means of $\mathcal{Q}_d$. Therefore, the quantization can be seen as a pair of encoding and decoding mappings:

$$\begin{aligned} \mathcal{Q}_e : &\quad \mathbb{R} \to \mathcal{I}, \\ \mathcal{Q}_d : &\quad \mathcal{I} \to \mathcal{C}, \end{aligned} \tag{2.11}$$

where the subscripts e and d represent encoding and decoding operations respectively.

The scalar quantizer is not the optimal way to quantize a signal (in particular audio signals). The drawback of a scalar quantizer is that it can not remove all irrelevancy. This can be illustrated in the entropy constrained case by by comparison to the rate-distortion function for a entropy constrained (fixed average rate) scalar quantizer operating on a Gaussian random variable with no inter-symbol redundancy [16]

$$H(X_i) = \frac{1}{2}\log_2(\frac{\sigma_{x_i}^2}{D}) + \frac{1}{2}\log_2(\frac{2\pi e}{12}). \tag{2.12}$$

If equation 2.12 is used in equation 2.9 it can be seen that $\zeta(X_i) \approx 0.25$ bits, which shows that a scalar quantizer cannot remove all irrelevancy. The scalar quantizer cannot achieve $R(D)$, however it has an advantage of low-complexity and facilitates analytical optimization with an aid of the high-rate theory [16].

The vector quantizer (VQ) is a generalization of the scalar quantizer, and as the name suggests, it operates on multi-dimensional space. The difference to the scalar quantizer is due to multi-dimensional quantization cells, which, in general, can exist in a k-dimensional space $\mathbb{R}^k$. The operation of the vector quantizer can be illustrated by the same two-stage mapping as the scalar quantization, except that the dimensionality of the signals space is now k. This implies that the reconstruction points are also in $\mathbb{R}^k$.

$$\begin{aligned} \mathcal{Q}_e : &\quad \mathbb{R}^k \to \mathcal{I}, \\ \mathcal{Q}_d : &\quad \mathcal{I} \to \mathcal{C}^k. \end{aligned} \tag{2.13}$$

The vector quantizer can, with increasing dimensionality k and after fulfilling some technical conditions, converge asymptotically to the rate-distortion bound [16]. This indicates that a vector quantizer can remove both irrelevancy and redundancy as it converges closer to the rate-distortion bound than the scala quantizer. Irrelevancy is removed because of the limitation of precision and the means to remove the redundancy are obtained due to the dependency between samples are reflected by the codebook vectors. This may suggest that the vector quantizer is the ultimate solution for all practical audio coding problems. However this is not the case. As the dimensionality of the vector quantizer increases, the computational complexity increases exponentially, which complicates the design and usage. The complexity problem can be solved partially by usage of structured vector quantization (e.g., lattice quantizers). However, other problems appear, such as a difficulty in obtaining an efficient coding of quantization indices (e.g., due to truncation problems).

Since an index $i \in \mathcal{I}$ must be converted into its binary representation for the transmission over network, there are two approaches that are commonly used. One approach assumes that every index i is assigned with a codeword of a fixed length of B bits. This approach results in so-called fixed-rate coding and the corresponding quantizer is resolution constrained. The other practical approach aims at coding the indices at a variable bit-rate and is presented in the next section. The bit-rate constraint is formulated in terms of a fixed average bit-rate and the obtained quantizer is thereof entropy constrained. This thesis focuses on the later case.

2.1.2 Lossless Coding

After irrelevancy removal, the signal is described with a finite number of discrete random variables. A method called lossless coding or entropy coding can now be used to code a signal without introducing any type of distortion. A direct method to convert the quantized indices would be to describe them each individually with a binary codeword, where a set of codewords is called a code. As different codewords have different lengths, the mean codeword length may be used to characterize such a code. As seen in the previous section, coding the quantization indices with respect to the first-order entropy equation does not account for dependency between samples. Instead of code on an individual sample-basis, one can create a codeword for a sequence of iid samples, and then it is then often possible to reach an optimal (the lowest possible) average codeword length, which is lower than for the single sample scenario. When coding a concatenated sequences of variables with codewords, the codewords should be assigned uniquely. This guarantees the that decoding is possible. A constraint on possible codeword lengths $\{l(x)\}_{x\in\mathcal{A}}$ is used to design a practical code. This constraint is described by the Kraft inequality $\sum_{x\in\mathcal{A}} 2^{-l(x)} \leq 1$. The Kraft inequality can then be used to obtain the source coding theorem [16],

$$H(X) \leq L < H(X) + 1, \tag{2.14}$$

where $L = \sum_{x\in\mathcal{A}} p_X(x)l(x)$. The source coding theorem states that the minimum average length of a uniquely decodable code is between the entropy $H(X)$ and $H(X) + 1$ bit. Therefore an estimate of the entropy of a source could be used to evaluate the performance of a practical implementation of a lossless coder.

There are different approaches to apply practical lossless coding to a signal but it can in general be divided up into two categories. First there are codes that require a-priori probability distributions. There are also universal codes that do not require this. The first category exploits the knowledge of the probability mass function of a source, this means that the coding scheme is able to assign short codewords to the sequences of indices with high probability and longer codewords to the sequences with lower probabilities (e.g., the indices on the tail of the distribution). Two common implementations of these lossless coders are Huffman-coding and arithmetic coding [16]. The second category do not use any a-priori statistical properties of the signal and therefore no prior pmf estimation is needed (in practice, the statistics of indices are learnt directly form the coded sequence instead), a methodology is the the Ziv-Lempel code, this method is however rather theoretical and impractical.

2.2 Audio Coding Techniques

In this section common audio coding techniques for the forward adaptive coding are briefly discussed. In the forward adaptive coding paradigm, the signal samples are blocked into blocks that are subject to coding. An efficient coding scheme should exploit the statistical dependencies between the blocks (inter-dependencies) and the statistical dependencies within a block (intra-block dependencies). In this section several common tools to remove redundancy in audio signals are presented: the inter-block coding methods, which exploit inter-block redundancy, will be described, and the intra-block coding methods.

The forward adaptive audio coders that perform coding on a block basis can further be sub-divided into two groups with respect to the signal representation that is used. Hence audio coders can roughly be divided up in two general classes:

- Waveform approximation coders
- Parametric coders

Waveform approximation coders strive to reconstruct the original signal with the smallest quantization error as possible while parametric coders reconstruct the signal from parameters and do not converge towards the original signal.

Parametric coders, as the name suggest, use a parametric representation to describe important properties of the signal that are needed for an efficient signal reconstruction. In general, a parametric coding scheme is able to operate a low-bit rate achieving a toll quality. However as the quality can only reach a certain level and saturates, therefore pure parametric coders are only used at the lowest bitrates. In waveform coding the signal is quantized and the goal is to minimize the error between the reconstructed and the original audio signal.

In modern audio coding these techniques are often combined to provide the best of both worlds in hybrid coding. Some parameters might be extracted and sent as side information, the residual signal (signal after parameter extraction) is then quantized in terms of waveform quantization and this combination gives better performance than each technique would do by itself.

This thesis focuses on the waveform coding approach and therefore the following discussion is limited to this case.

2.2.1 Inter-block Coding

The inter-block redundancy described in equation 2.6 is commonly exploited in audio and speech coding for more efficient coding. When encoding the n-th block, the information from previous blocks can be used to calculate the predicted block

$$r = \mathbb{E}(X_n | X_{n-1}, \cdots, X_0), \tag{2.15}$$

where $X_{n-1}, \cdots, X_0$ is the previous blocks. The predicted block r describes describes the prediction about a currently encoded block based on the previously coded blocks. The vector r may be interpreted as so-called ringing. The ringing can be decomposed into two contributions: one accounting for the short term dependency between the samples on a boundary between the coded blocks and one accounting for the longer dependencies. The decomposition of the ringing vector is usually motivated by a convenience of implementation. We limit our discussion to the case where the estimator described by equation 2.15 is linear.

The short term ringing subtraction, which often is noted as ringing subtraction, is based on exploiting of sample dependencies on the boundary between two blocks. The ringing subtraction is commonly used in the state-of-the-art audio coding schemes. For instance in CELP schemes, the ringing of the synthesis filter [17] used in the analysis-by-synthesis scheme needs to be accounted for. The ringing subtraction operation exploits the signal envelope-related dependencies. These dependencies may be described by an autoregressive (AR) model. The model is used to design whitening filters that yield signal representation that may be efficiently quantized. In practice, the AR is estimated by means of linear prediction (LP).

Another important tool to remove inter-block dependencies is a long term prediction. (LTP) The LTP exploits a periodic nature of audio signals, which is normally occurring during a large portion of speech and audio segments. This periodic nature of audio signals is commonly modelled by the term pitch [18]. The notion of LTP is implemented by means of pitch subtraction. The pitch subtraction operation is driven by a pitch model

$$E(t) = X(t) - \beta X(t - T), \tag{2.16}$$

where $E(t)$ is the residual (or so-called innovation) after the pitch contribution $\beta X(t - T)$ has been subtracted from the current signal in a open loop. $X(t)$ and β is the pitch decay factor and T is the pitch delay (pitch lag). The equation 2.16 may be rewritten as a filtering operation. If the $\mathcal{Z}$ transform is used, the pitch can be modelled by a one tap filter

$$X(z) = \frac{E(z)}{1 - \beta z^{-T}}, \tag{2.17}$$

where $X(z) = \mathcal{Z}\{X(t)\}$ and $E(z) = \mathcal{Z}\{E(t)\}$. Both of these methods must be used to reduce redundancy in practice. The LTP has a huge impact on the coding performance and, therefore, it is essential for any state-of-the-art codec operating at low or moderate rates.

The ringing subtraction operation performed at the encoder has its counter-operation performed at the decoder. In order to reconstruct the signal at the decoder, the ringing contribution must be estimated and added to the reconstructed signal. As the decoder has only access to previously reconstructed signal $\widehat{X}$, the reconstruction can only be based on that signal. The encoder has access to the original signal. If the prediction at the encoder would be based on the original signal, there would be a mismatch between the encoder and the decoder which means they are not in sync, which would affect the performance of the coding scheme. This corresponds to a situation, where so called open-loop prediction is used. Alternatively, a closed-loop prediction may be used, where the encoder perform a local decoding and uses the locally decoded signal to facilitate its prediction. This corresponds of using the decoded signal $\widehat{X}_{n-1}, \cdots, \widehat{X}_0$ in equation 2.15. Closed-loop prediction complicates the encoder but has a number of advantages over the open-loop prediction.

2.2.2 Intra-block Coding

As the first two methods focus on remove redundancy based on the signal from previously decoded blocks, the following two methods focus on exploiting the redundancy between the samples within the currently coded block according to equation 2.7.

The first method to exploit these dependencies is prediction, which is also the most common tool used in codecs for audio communications. It offers relatively low delay, low complexity and good performance in practice. The term prediction refers to the method predicting the next sample of a process based on the previous samples. Formally, the prediction operation employed to intra-block coding requires that the signal is stationary within a block. This assumption is usually realistic for blocks with duration of approx. 20ms.

The original signal is used as condiments in a analysis filter to compute a prediction residual. The original signal is simply filtered by the analysis filter to obtain the residual signal. The rate reduction bound for stationary Gaussian process with variance σ_x^2 and with prediction error variance σ_p^2 can be defined as

$$\rho(X_i) \geq \frac{1}{2} \log \frac{\sigma_x^2}{\sigma_p^2}, \tag{2.18}$$

where p is the predictor order and $\sigma_p^2 \leq \sigma_{p-1}^2$ for $p \in \{1, 2, \cdots\}$ [16]. When $p \to \infty$ the bound is tightest, this is however not practical due to memory and complexity constrains of practical implementations. Therefore for wideband audio signals the order 16 of the predictor p is normally used. Since the variance of the prediction error is lower it seems natural to quantize this for transmission. However one of the main drawbacks of linear prediction is that the filtering operation is not invariant with the error criterion, this means that the MSE criterion which was optimal in the weighted signal domain is not optimal in the so-called excitation domain. The prediction can be seen as a non-unitary transform, which has certain disadvantages for coding. Nevertheless, the prediction is commonly used in the state-of-the-art audio coders.

Another approach that addresses intra-block dependencies assumes signal quantization in the transform domain (so-called transform coding). Transform coding is also a method to exploit redundancy and it is commonly used in audio coding. While prediction removed redundancy in terms of minimize the variance, the transform tries to concentrate energy of the signal coefficients within a few number of them. This approach is able to exploit the redundancy.

The unitary transforms are commonly used in coding. If the transform is unitary, it conserves the MSE criterion meaning that the squared error in the signal domain is the same as the squared error in the transform domain. This is in contrast to prediction, which does not render such a property.

Two types of transform which is commonly used in audio coding is non-adaptive transforms and adaptive optimized transforms. As the names suggest non-adaptive transforms are fixed and the adaptive transforms are calculated for individual blocks based on some statistical model.

Both transforms types have their benefits and drawbacks. The transforms with adaptation may be optimal since it is calculated based on the statistics of the actual signal. It can therefore achieve a better energy concentration than a non-adaptive transform. This however comes with the price of complexity. The model needs to be estimated and then the actual transform need to be derived from that model. Also since the signal statistics is unknown at the decoder these have to be quantized and transmitted which increases the bitrate.

The fixed transforms can be asymptotically optimal as the dimension is $k \to \infty$ which is not achievable in practice due to a delay constraint, but for practical block lengths, the non-adaptive transform are still able to concentrate the signal energy. A design trade-off for the non-adaptive transform is therefore between the coding delay and the coding gain. Also since, in this case, the transforms are fixed they do not require any additional transmission.

The main benefit of unitary transforms is that they render the efficiency of the MSE criterion that is used to design the quantizers operating in the transform domain.

2.2.3 Audio Coding Structure

In the previous sections the theory and the methods to apply the theory was introduced, in this section it will be combined to present a audio coding scheme. If equation 2.8 and equation 2.9 is used in equation 2.5 this gives the total average rate

$$L = \rho(X_i) + \rho_c(X_i) + \zeta(X_i) + R(D). \tag{2.19}$$

This can then be realized in a coder architecture which can be seen in figure 2.1. First redundancy is exploited by means of signal processing which has been described in the previous section which described inter and intra-block coding. In the second step the irrelevancy is removed by means of quantization using a perceptually meaningful distortion measure. The quantization outputs quantization indices that still include some redundancy (they require entropy coding). The redundancy removal (from the quantization indices) is finally done by lossless coding.

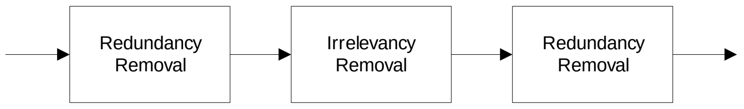


Figure 2.1: Audio coding scheme block diagram.

This concludes the introduction of some common concepts in audio coding. In the next sections audio quality assessment is discussed as the subjective coding quality versus operating bit rate is of high importance for evaluating the performance of audio coders. The chapter is concluded by some discussion about important audio coding attributes.

2.3 Audio Quality - Subjective Quality Assessment

Lossy coding obviously introduces a distortion. The audio quality of the audio codec becomes the parameter of a high practical interest. For newly developed codecs the audio quality needs to be evaluated to make sure that such a codec performs satisfactory (e.g., compared to the current state-of-the-art codecs). This evaluation is often categorized into two methodologies: subjective quality evaluation (performed by means of listening tests) and objective quality measurements (that can be automated).

The subjective evaluation is done by means of listening test that needs to be performed to assure the subjective performance of the audio codec. Typically, such a test involves a large number of listeners that evaluate the codec over a large number of test conditions. There are different types of subjective test with different types of rating. Commonly used test methodologies is:

- absolute category rating (ACR) which is commonly used for transmission quality assessment with a scale from 0 to 5,
- multiple stimuli with hidden reference and anchor (MUSHRA) [19],
- A/B test that facilitates a simple comparison of two test conditions.

The test results obtained form a large number of listeners are statistically evaluated. For example, the average test scores may be calculated, which gives a mean opinion scores (MOS).

The MUSHRA test is a subjective assessment for intermediate quality levels of coding systems. The MUSHRA test has a scale which ranges from 0 to 100, which makes it possible to rate smaller differences, also a hidden reference together with an low pass filtered reference (Anchor) is used. If an anchor signal is used the scale is closer to an absolute scale. All codecs are compared at the same time which makes the requirement of participants smaller than the ACR test. A/B test is used to compare single variable test, this can for example be tuning of some parameter.

The subjective tests are expensive in terms of man hours and therefore the main motivation to use objective test is to reduce cost. As the subjective tests are both costly and time consuming therefore in many situation objective evaluation tools are used instead. Objective test is cost effective and also repeatable on larger scale compared to subjective tests. A main disadvantage of the objective test is that it does not necessarily produce a reliable results (especially in the case where a newly developed coding scheme is analyzed).

Simple measures as signal-to-noise ratio (SNR) with MSE criterion are not perceptually relevant at low bit rates therefore a more advanced objective measure is needed. Perceptual evaluation of speech quality (PESQ) is a standard for objective testing of speech quality. It uses the same scale as the ACR test and the result from PESQ should correspond to similar test results from a subjective test. The current PESQ standard supports wideband speech signals between (50-7000Hz) [20]. The successor of PESQ is P.OLQA which supports super wideband (SWB) speech signals (50 - 14000Hz). This is needed for evaluation of the next generation state-of-the-art codecs which will support SWB. However, P.OLQA and PESQ only supports speech and can therefore not be used on general audio signals. This is the purpose perceptual evaluation of audio quality (PEAQ) standard [21]. However PEAQ does not support the low bitrates with high distortions which is needed for evaluation of audio codecs operating at low bit rates and it is therefore only useful and recommended at higher rates.

2.4 Audio Coding Attributes

Except the coding quality there are several other attributes of audio coding scheme that are important form the point of view of the design. These attributes describe the performance of a codec in terms of practically-relevant parameters. Some of them are useful in terms of constraining the design. Some of the attributes are practically related by means of design trade-off, i.e. low transmission rate results in lower audio quality due to higher distortion, etc. The discussion starts with a presentation about delay.

2.4.1 Delay

Delay is an important audio coding attribute. In order to keep a conversation in communication system in decent quality, a delay smaller than 200 ms is preferred [22]. But the conversational quality starts to degrade quickly already at 100 ms. There are many components in a audio communication system that influence the end-to-end delay for the end user:

- Sound capture
- Preprocessing
- Encoding

- Network
- Jitter buffers
- Decoding
- Postprocessing
- Sound playback

As seen, the algorithmic delay of the audio codec is just one contributor to the total delay of the system. The shorter delay in the codec the longer delay is acceptable in the network and the other components. This has created a market for low-delay codecs. However, these typically require a much higher bit-rate to achieve a similar quality than codecs with moderate delay such as 25 ms. As lengthy coding delay is not desired, it can certainly help to achieve a more efficient coding scheme. Depending on the application, the acceptable coding delay of codecs for real-time communications varies between 16 and 40 ms.

2.4.2 Operating Bit-rate

One of the most important attributes of audio coding is the number of bits that is required to transmit the audio bit-stream to the decoder. When bits are used to specify the transmission rate, it is usually specified in terms of bps. Since the bit rate used in practical applications usually is much higher the term kilobit per second (kbps) is more commonly used. If the sampling rate for the system is known then the term bits per sample can also be used. In this thesis for wideband coding systems that use a sampling frequency of 16 kHz the bit-rate is described as

- Low-rate < 5 kbps
- Medium-rate < 8 kbps
- Intermediate-rate < 16 kbps
- High-rate ≥ 16 kbps

In practice, the bit-rate may be specified in a number of ways. By far, the two most common scenarios include the case of a fixed bit-rate and the case of a average bit-rate. Therefore, the bit-rate constraint is often formulated in terms of fixed rate constraint and average bit-rate (entropy) constraint. These cases are not dual and in principle the systems can be optimized according to any of the two criteria. The bit-rate constraint is closely related to the design of signal quantizers. For example, the constrained resolution approach tries to minimize the distortion by finding the optimal centroid density function given a fixed codeword length (a fixed rate) yielding a resolution-constrained quantizer. The constrained entropy (at high bit-rates) leads to a uniform density of the reconstruction points, which leads to uniform entropy constrained quantizers.

The most common quantization method used in the state-of-the-art audio coding is the resolution constrained quantization. Therefore many state-of-the-art coders operate at a fixed rate. This is mainly because the traditional communications systems used to operate at a fixed rate (e.g., radio channels in GSM). The constrained resolution systems have an advantage since generation of the bit-stream becomes very simple in this case. In principle, vector quantizers optimized for the resolution-constrained scenario can be used to achieve an efficient coding. The design typically involves iterative procedures (e.g., Lloyd-Max algorithm) that prevents flexible coding.

2.4.3 Computational Complexity and Memory Requirements

The computational complexity and the memory required for an audio transmission system is associated with some operational and implementation costs. These costs affect applicability and usefulness of a particular coding approach. It is therefore of a high practical interest to reduce the associated costs during the design of an audio coding communication system.

The memory requirement in a codec is mostly dependent by the number of codebooks that needs to be stored. The number of codebooks is therefore often limited to some small number of operating modes. This however limits the flexibility of the audio codec. A small memory footprint is important for embedded implementations as well as the download size of the codec when it used in VoIP applications.

Computational complexity is a typical concern when developing codecs. This is especially true since codecs for real-time communications should be able to perform real-time operation on small embedded devices such as mobile phones. Performance can be evaluated either through theoretical analysis [23] in the design phase or by profiling in the implementation phase. Profiling is done by running an instrumented codec on a specific platform to evaluate some performance related counters. This means that the codec needs to be implemented and properly instrumented before any benchmarking can be performed. Therefore, profiling is commonly used to find bottlenecks in implementations. These bottlenecks can then be optimized for a specific platform with machine specific code. Theoretical analysis is often a more useful tool to evaluate an algorithm while it is developed. It gives a rough estimate on the performance.

The algorithmic complexity may be conveniently expressed using the notation is called the *Big-O*. If $T(n)$ is the run time of an encoder and n is the dimension of the input vector, then the run time can be expressed as $T(n) = O(f(n))$ where $f(n)$ is an encoder function. The two following rules [23] makes the usage of the *Big-O* straightforward to use:

1. *Constant factors do not matter.* $O(df(n)) = O(f(n))$ where d is a constant.

2. *Low-order terms do not matter.* Only the largest polynomial has to be kept since it will dominate any other polynomials at large n.

2.4.4 Transmission-Error Sensitivity

Any audio codec applied to coding through a network with losses will face the problem of transmission errors. The transmission errors can involve bit-errors where some bits of the transmitted bitstream are inverted. In addition, there can be packet losses, where some part of the bitstream is lost due to dropping of a packet. Often telecommunication system over networks use the transport protocol user datagram (UDP) since it has smaller overhead and lower delay due to no retransmission of packets if an error occurs. Since no error handling is used on the transport layer this needs to be handled at a higher layer in the OSI stack. Usually a channel coder is used ensure that no packet loss or bit-errors is accepted. A jitter buffer is used to do ensure that the packets are provided to the decoder in the correct order. Some codecs are designed to be robust against frame erasures such as G.718 and is thereby a good choice for VoIP applications in IP networks [24].

Chapter 3

Flexible Audio Coder

In this chapter a high-level view on the architecture of the flexible audio coder (FAC) is provided. The description is based on the work of [13, 14]. The architecture includes a number of state-of-the-art components that are based on AMR-WB codec [4] and on the quantization method of [15]. The high-level view focuses on the functional description of the obtained coding structure and provides implementation details for encoder and the decoder.

3.1 Coding Driven by a Statistical Signal Model

The philosophy behind this flexible audio coding scheme is motivated by using a statistical model of the audio signal. The audio signal is modeled as a stochastic sequence of samples {X(t)}, these are blocked into non-overlapping signal frames x_m. An m-th frame is then a realization of a k-dimensional random vector X^k. It is assumed that the signal is stationary within a duration of a frame. The random vector X^k have a distribution specified by $f_X(x)$. If the subscript m is dropped $f_X(x)$ can be approximated by a generic mixture model

$$f_X(x) \approx \int p_\Theta(\theta) g_{X|\Theta}(x|\theta) d\theta \tag{3.1}$$

where $p_\Theta(\theta)$ is a pdf of the parameter Θ and $g_{x|\Theta}(x|\theta)$ is the pdf of s conditioned on the model parameters Θ.

Efficient coding of such a source described by 3.1 is done in two steps. The first step is to estimate a vector of model parameters that governs the statistical signal model for the signal block. The estimation is performed according to the maximum likelihood criterion,

$$\hat{\theta}(x) = \underset{\theta}{\text{argmax}} g_{x|\Theta}(x|\theta) \tag{3.2}$$

where $\hat{\theta}(x)$ is the vector of estimated model parameters. Since the parameters need to be known at the decoder side they need to be quantized for transmission. Let the quantized model be denoted by $\overline{\theta}$, the quantized model can then be used to calculate the adaptive codebook $\mathcal{C}$ which is used to quantize the signal.

The approach yields a two-stage description of the signal. Each block is encoded by a description including the quantized model parameters $\overline{\theta}$ and a representation of the quantized signal, where the quantizer is based on $\mathcal{C}$. Such an approach falls into the class of forward adaptive coding methods. The approach leads to a practical coding architecture that is described in the following section.

3.2 Coder Architecture

A high-level overview of the audio coder can be seen in figure 3.1. The coder

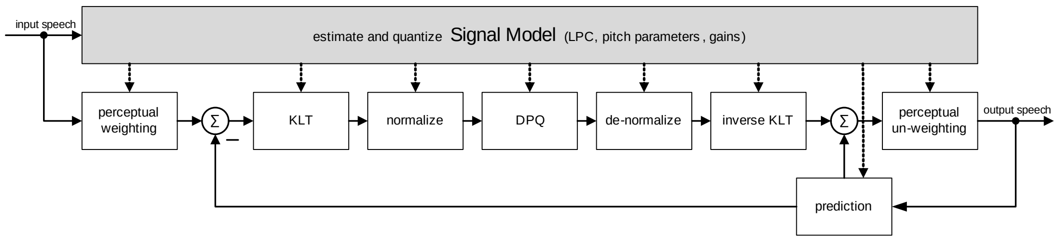


Figure 3.1: High-level view on the architecture of the flexible audio coder.

is implemented as a model driven forward-adaptive coding scheme. The coding is performed on a block basis in the perceptually weighted domain. As the block diagram indicates the perceptual weighting is based on the signal model (specifically on a quantized LPC model, see Section 3.2.2). To inter-block dependencies are removed by the means of ringing subtractions. The ringing subtraction is implemented in a closed-loop fashion (the estimation of ringing is based on the previously reconstructed signal). The intra-block decencies are handled by a signal adaptive transform. The perceptually weighted signal is transformed to a transform domain yielding a vector of transform coefficients. The transform coefficients are quantized by means of a scalar quantizer. Prior to quantization they are normalized. Reconstruction comprises de-normalization, applying of an inverse transformation, ringing addition and finally inverse weighting. These operations may be now divided between the encoder and the decoder.

The encoding can be roughly divided into following operations:

- Estimation and quantization of the (probabilistic) signal model;
- Generation of the bit-stream for the description of the signal model;
- Weighting the input signal into a perceptually optimized domain;
- Computation of ringing and ringing subtraction;
- Calculation of the optimal adaptive transformation given the signal model;
- Signal normalization;

- Signal transformation;
- (Scalar) quantization in the transformed domain;
- Generation of the bit-stream for the signal description (arithmetic coding).

The decoding operations include:

- Decoding of the signal model bit-stream;
- Computation of the perceptual model, the (inverse) adaptive transform and the ringing signal based on the decoded signal model;
- Decoding of the bit-stream with the signal description;
- De-quantization and signal de-normalization;
- Inverse transformation of the de-quantized signal;
- Addition of the ringing signal;
- Inverse (perceptual) weighting yielding the final reconstruction of the signal.

The statistical model introduced in Section 3.2.1 needs to be specified in more detail. A block of the input signal is modelled by a multivariate Gaussian model. The model is estimated at the encoder. Since it must be known to the decoder it is parameterized and quantized. The multivariate Gaussian model has two parameters: mean and covariance matrix. The parametrization of the multivariate Gaussian model is obtained by representing its covariance matrix by means of an AR model. The AR model is concatenated and consists of two parts. One part model the short-term dependencies. This part may be conveniently described by the linear prediction coefficients (LPC). The other part describes the long-term dependencies. This part is based on the autoregressive pitch model that includes pitch-lag and pitch-gain. The concept of usage of the statistical model is described in more detail in Section 3.2.1.

3.2.1 Signal Model

As already stated in 3.1, the signal distribution can be approximated by a mixture model. In the previous section the mixture component has been chosen to be a multivariate Gaussian model. If the signal is zero-mean, the covariance matrix becomes the only model parameter. The mixture component in equation 3.1 is normally distributed according to

$$g_{x|\Theta} = \mathcal{N}(\mu_x, \Sigma_x), \tag{3.3}$$

where each block is specified by the mean μ_x and the covariance matrix Σ_x. The generic mixture model is now assumed to be a Gaussian mixture model (GMM).

In order to parameterize a single model component, the mean and the covariance matrix is needed. Since it is not efficient to encode and transmit the full covariance matrix, the covariance matrix is represented with an aid of an AR model.

The AR model used for the parameterization is a concatenated model that consists of two parts. The composite model used in this thesis is based on the short-term AR model combined with a with the pitch model that accounts for long-term signal dependencies 2.17:

$$\frac{\sigma}{A(z)} = \frac{\sigma}{1 + a_1 z^{-1} + ... + a_p z^{-p}} \times \frac{1}{1 - \beta z^{-T}} \tag{3.4}$$

where σ is the signal gain, a_n is the n-th LPC of a p-th order model, β is the LTP coefficient (i.e. pitch gain) and T is the pitch lag. In practice the pitch lag can be non integer, which can be efficiently handled by interpolation and the polynomial in the denominator of the LTP filter would then use more than one tap.

The model is used to compute an impulse response $h = h_1, h_2, ..., h_k$ that in general may be k-dimensional. This impulse response is then used to build a structured matrix,

$$\mathbf{H} = \begin{bmatrix} 1 & & & & & \\ h_1 & 1 & & & & 0 \\ & h_1 & 1 & & & \\ & & \ddots & \ddots & & \\ & & & & & \\ h_k & & & & h_1 & 1 \end{bmatrix} \tag{3.5}$$

which is a lower triangular Toeplitz matrix. The covariance matrix is then estimated as:

$$\mathbf{\Sigma_x} = \mathbf{H}^{\mathrm{T}}\mathbf{H} \tag{3.6}$$

The mean vector μ_x is not explicitly transmitted. Instead the mean vector is estimated at the decoder as $\mu_x = \mathbb{E}\{X_m | \hat{x}_{m-1}, \hat{x}_{m-2}, ..\}$. As such an estimate is based on the previously decoded blocks, it is available at the decoder and in the local decoder in the encoder. The estimation of μ_x can be interpreted as computation of ringing. The ringing accounts for both: short term and long term dependencies between the blocks.

As stated before, the signal model needs to be transmitted as side information in the form of a two-stage description of the signal. One question that may arise how the available bit rate R should be distributed between the signal model parameters θ and the signal itself. The problem of finding the optimal bit distribution between the model and the signal is particularly important for the flexible coder. Since there are infinitely many operating points the rate allocation cannot be optimized experimentally for all these points. The high-rate theory provides a solution to this problem. In the paper [25] it has been shown that under high-rate assumptions the optimal model rate remains constant and independent from the mean signal distortion. This strategy is used in the FAC implementation.

3.2.2 Perceptual Weighting

The flexible coding architecture assumes quantization in the perceptually optimized domain. The mapping to that domain is implemented by means of filtering with an adaptive filter. In the considered architecture a state-of-the-art weighting method is used. The weighting filter is derived from the AR model as follows

$$W(z) = \hat{A}(z/\gamma_1)H_{de-emph}(z), \tag{3.7}$$

where $\hat{A}$ is the quantized AR model, γ_1 is the perceptual weighting factor and $H_{de-emph(z)}$ is the de-emphasis filter [4]. Worth to note is that the weighting filter is derived from the quantized model while in AMR-WB the weighting is based on the unquantized model. This is due to the fact that the quantized model is used to derive the adaptive transform.

3.2.3 Karhunen-Loève Transform

If the covariance matrix $\mathbf{\Sigma_x}$ of the input signal is estimated then the input signal dependent optimal transform can be calculated (assuming Gaussianity). The optimal transform in this case is the Karhunen-Loève Transform (KLT), which diagonalizes the covariance matrix of the input signal. It renders an uncorrelated signal in the transform domain (if Gaussian assumption is valid, the transform are realizations of independent random variables). It can be shown that by diagonalizing the covariance matrix the energy of transform coefficients is concentrated, which helps to improve the MSE performance of scalar quantizers used to quantize the transform coefficients. The calculation of the KLT transform involves eigenvalue decomposition (EVD) of the covariance matrix,

$$\mathbf{\Sigma_x} = U\Lambda U^T \tag{3.8}$$

where U^T is the KLT and Λ is a diagonal matrix with the eigenvalues. However the EVD is computationally expensive as it has the computational cost of $O(n^3)$, where n is the rank of the matrix. Therefore even that the KLT is the optimal transform, it is not common in practical implementations that aim at real-time operation. Due to that reason, some non-adaptive transformations are used that attempt to approximate the effect of the KLT. In particular the MLT is often used. To enable the usage of KLT in a operating audio coder the complexity needs to be reduced. This is possible by exploiting the structure of the statistical signal model. The covariance matrix is a normal matrix and positive semi-definite, therefore, the singular value decomposition (SVD) may be used. In the implementation of the KLT-computing block of the coder, the SVD algorithm is used, which has the complexity $O(n^2\log(n))$. The SVD is performed in two stages; first the matrix is reduced to a tridiagonal matrix by using Lanczos method [26] which normally has the complexity of $O(n^2\log(n))$. Since the covariance matrix has a Toeplitz structure, the tridiagonalizaion may be performed by using FFT, which reduces the computational complexity. The second step solves the eigenvalues by the means of a QL-algorithm which has the complexity of $\mathcal{O}(n^2)$. The total cost is therefore $\mathcal{O}(n^2 \log(n))$ compared to the ordinary cost $\mathcal{O}(n^3)$ of a EVD which facilitates a codec implementation of the algorithm which can operate in real-time with a limited block length.

3.2.4 Distribution Preserving Quantization

When the signal is decorrelated by the transform, the coefficients needs to be quantized. The quantization should preserve the signal values and minimize the distortion based on the distortion measure. A classical approach in this case would be to use scalar quantizer with an MSE criterion. The distortion for the regular entropy-constrained scalar quantizer (assuming high-rate) is given by

$$D = \frac{\Delta^2}{12}, \tag{3.9}$$

where Δ is the step size of the quantizer. As the quantization step size Δ increases, the MSE also increases, thus the design trade-off between the distortion and the rate can be made by adjusting the quantization step size.

A classical approach to implement a quantizer is by the usage of a rounding operation, i.e.,

$$Q(x) = \left[\frac{x}{\Delta}\right], \tag{3.10}$$

where x is the coefficient, $Q(x)$ denotes a index which is a quantized coefficient x and $[\cdot]$ denotes the rounding operation. To de-quantize the index (i.e., reconstruct the signal), the result is obtained by simply multiplying the index by the step size.

Due to energy concentration of the transform only a few transform coefficients have variances that are large compared to the quantization step-size. Hence, in normal operation, many transform coefficients are reconstructed by their mean value. This phenomenon is related to the reverse water-filling process that appears if quantization is performed in the MSE-optimal way. The reconstruction with a mean value may lead to several coding artifacts, which are especially severe in low-rate audio coding. In order to combat these artifacts, the DPQ scheme has been proposed [15].

Every coefficient is assumed to be independent due to KLT and normally distributed, the optimal configuration of the scalar quantizer in equation 3.10 results in the reverse water-filling process. The process determine the rate distribution between the coefficients [16] assigning the coefficients that have a variance above some threshold (defined by the step size) with a non-zero rate and all the remaining coefficients are allocated with a zero rate. This means that if a transform coefficient has a lower variance than the "water level" [16], it is reconstructed by its mean. As this is optimal in the MSE sense this leads to distortions and low perceptual quality at lover rates (when the step size is high). These distortions is commonly described as "birdies" and "mumbling" [27]. The "birdies" is a effect of the coefficients reconstructed as the mean and the variation of the energy between block, this variation introduces holes in the spectrum that appear as audible transform artifacts. The "mumbling" is generated by bandwidth reduction.

To prevent this distortions at a lower rate a DPQ scheme [15] has been implemented. The DPQ scheme preserves the distribution of the source signal while keeping the MSE at the possible minimum. This yields a scheme which provides good perceptual performance at a low rate while providing a MSE-optimal quantizer at high rates. The scheme is based on two important steps, first a subtractive dither followed by a distribution preserving transformation, see figure 3.2 where

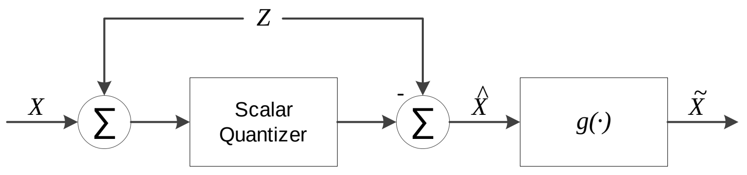


Figure 3.2: Diagram of DPQ.

X is the transform coefficients, Z is the additive dither, $\hat{X}$ is the quantized input with subtracted dither. The dither is added before the quantization described in equation 3.10. This gives a quantization index computed from

$$Q(x+z) = \left\lfloor \frac{x+z}{\Delta} \right\rceil, \tag{3.11}$$

where z denotes the dither value. The dither value is a realization of pseudo random uniformly distributed random variable, i.e. $Z \sim \mathcal{U}(0, \Delta)$. In the decoder this dither is subtracted after dequantization. Since subtractive dithering is used, the reconstruction is computed upon subtracting a realization of dither, i.e.,

$$\hat{x} = Q(x+z) \cdot \Delta - z.$$

Finally the transformation $g(\cdot)$ is performed on $\hat{x}$, this yields an the final reconstruction $\tilde{x}$. The transform $g(\cdot)$ is given by,

$$g(\hat{x}) = F_X^{-1}\left(\frac{1}{\Delta}\int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} F_X(\hat{x}-\tau)d\tau\right), \tag{3.12}$$

where $F_X(\cdot)$ is the cumulative distribution function (CDF) of X and $F_X(\cdot)^{-1}$ is its inverse. See [15] for more details.

The usage of DPQ implies that a suitable mechanism for maintaining a synchronicity between the encoder and the decoder must be developed to be able to use the dither. In addition, since the quantization indices are encoded by means of an entropy coder, a method for accounting for the dithering during entropy coding must be established. These practical problems were solved within this thesis.

A pseudo random nature of the dither means that the dither possesses statistical properties of a random variable Z, however, it is known by both encoder and decoder. It has been found that in a practical implementation the generator of the dither signal may be restarted every 20 ms, therefore the dither values may be stored in the memory of encoder and decoder. Every frame a pointer to a current dither value may be restarted, so the encoder and decoder can stay in sync.

The DPQ has a property that at high bit-rates it is equivalent to the classical entropy constrained quantization. Since DPQ involves a transformation that depends on the signal, it is computationally more expensive that the classical quantization. In addition, DPQ requires a very accurate signal model. If such a model is not available DPQ will introduce some additional distortion. In the proposed implementation of FAC, the usage of DPQ is heuristically tuned aiming at achieving the best possible coding performance. At low bit-rates DPQ is used for all the transform coefficients. As the bit-rate increases the transform coefficients with the largest variances are quantized by the classical entropy quantization and all the remaining coefficients are quantized by DPQ. This scheme is implemented by a function that determines the number of highest variance coefficients that are quantized by a regular quantization based on a value of the step size. By this measure no additional transmission is needed. The usage of a combination of DPQ with a classical entropy constrained quantization has been experimentally tuned to achieve the best possible coding performance.

The implementation made during the thesis work includes both the classic entropy-constrained approach together with the new DPQ scheme.

3.2.5 Arithmetic Coding

Once the coefficients are quantized the indices needs to be losslessly coded, in this implementation an arithmetic coder [28] is used. Since the signal is assumed to be Gaussian distributed, the error function (erf) can be used to calculate the high and the low probabilities for the symbol. The implementation also needed to be modified to take the dithering in the DPQ into account. The CDF of a Gaussian distribution that is zero mean is given by,

$$\Phi(\frac{x}{\sigma}) = \frac{1}{2}\left(1 + \operatorname{erf}(\frac{x}{\sigma\sqrt{2}})\right) \tag{3.13}$$

where x is the quantized coefficient and σ is the standard deviation for that coefficient. Since the input to the arithmetic coder is the indices $Q(x)$, where $Q(\cdot)$ denotes quantization. They need to be dequantized and the dither should be subtracted for probabilities of the low and high.

$$x_{high} = (Q(x(n)) + 0.5 - z(n)) \cdot \Delta \tag{3.14}$$

$$x_{low} = (Q(x(n)) - 0.5 - z(n)) \cdot \Delta \tag{3.15}$$

where $z(n)$ is dither.

The dither $z(n)$ needs to be in sync with the DPQ, the pseudo random dither used in the arithmetic encoder and decoder is the same used in the DPQ, i.e., the arithmetic coder is synchronized with the distribution preserving quantizer.

As the numerical precision is limited, which should be taken into account during implementation. The indices that appear on the tail of the signal distribution need to be carefully handled, i.e., one should prevent the situation when index probability is essentially 0. One way to deal with too low probability values is by scaling the coefficients. If it turns out that the probability is too low to, a scaling is applied to the coefficient and it is quantized again.

In the decoder the CDF is calculated in the same way as in the encoder and the bitstream is decoded into indices. The described implementation was originally written in C++ [28] so it was rewritten to ANSI C as the reference implementation of AMR-WB is provided in ANSI C.

3.3 Implementation based on AMR-WB

For the implementation of the key-technologies of the flexible audio coder the state-of-the-art codec AMR-WB was chosen as a core. AMR-WB is well known and the main advantage of implementing based on the functional block of this codec is that it has carefully tuned model estimators and quantizers that provide state-of-the-art performance. These blocks can be combined with the flexible coding core yielding a complete audio coder.

This section will provide some introduction to the AMR-WB codec and also the specifics around the new implemented components. For more comprehensive information about AMR-WB see [4, 3].

3.3.1 Encoder

The encoder diagram in figure 3.3, is a modified version of the diagram of the AMR-WB encoder that can be seen in [3]. The source code for AMR-WB provides a floating point implementation of the encoder, but the decoder and the local decoder in the encoder is implemented in a fixed point convention.

The input audio is processed in frames of length of 320 samples, which corresponds to 20 ms at a sampling-rate of 16 kHz. The signal is pre-proceeded by down-sampling to 12.8 kHz, which is the internal sampling rate used by the core codec. This results in a reduction of a frame size to 256 samples.

The input signal is filtered by a high-pass filter with a cut-off frequency at 50 Hz. This operation removes DC component and attenuates unwanted low frequencies. The high-pass filter is followed by a pre-emphasis filter. The pre-emphasis is a first order high-pass filter. This filter emphasize the higher frequencies by introducing a so-called spectral tilt. This spectral tilt allows for better modeling of the higher frequencies by means of LP.

The LP analysis is done every frame using the autocorrelation method. First, the signal is windowed by an asymmetric window. The window uses 5ms look-ahead and 5ms memory resulting in a 30 ms frame that is used for computation of LPC. The 5 ms lookahead is increases the total algorithmic delay of the codec to a gross delay of 25 ms. The autocorrelation sequence from the windowed frame is used to

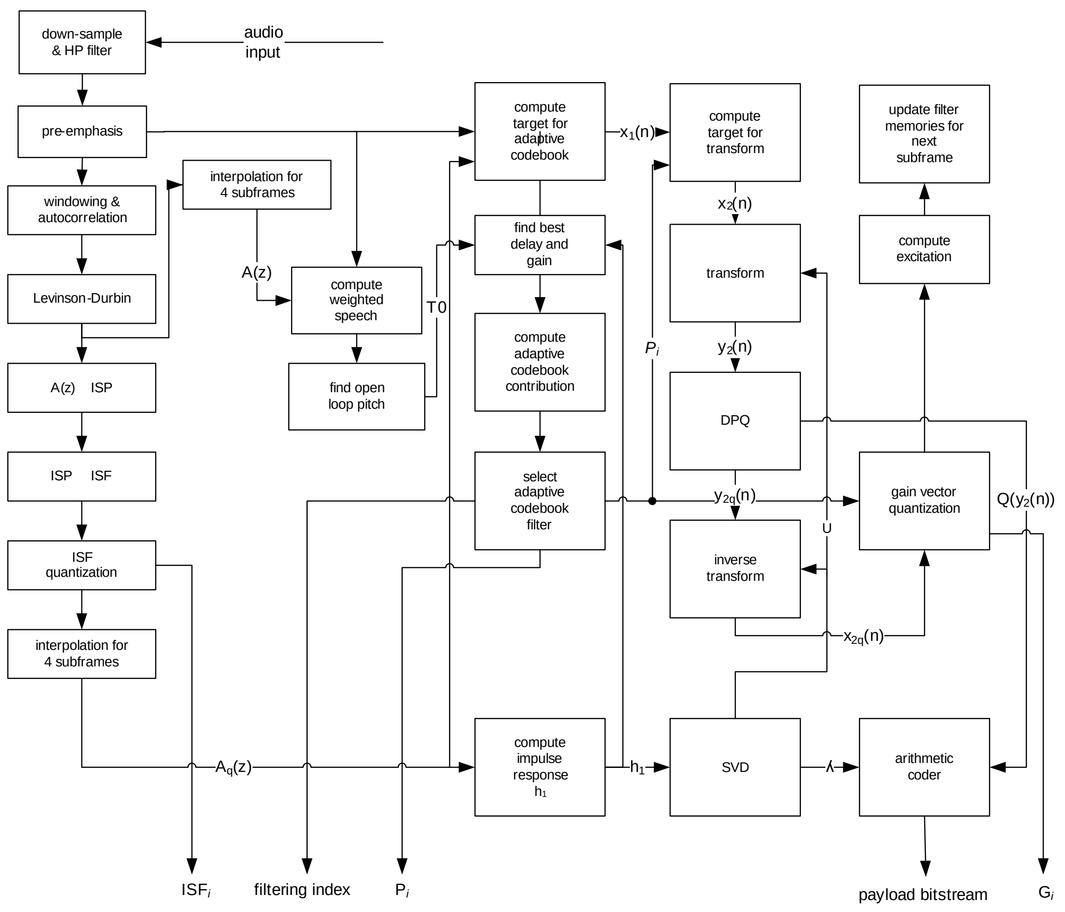


Figure 3.3: Detailed block diagram of the flexible audio encoder.

compute the LPC by using the Levinson-Durbin algorithm. As the LP coefficients need to be quantized, they are converted to Immittance Spectral Pair (ISP) and quantization is performed in the ISP domain to ensure stability. The LPC-based model is quantized by split-multistage vector quantization (S-MSVQ). Since the encoder is operating on sub-frame basis the quantized LP coefficients is interpolated for the three first subframes by linear interpolation [4]. The asymmetrical shape of the window provides a good model of the fourth subframe and therefore the model is not interpolated for that frame.

The interpolated LPC-based model is obtained from the pre-emphasized signal. The model is used to compute weights of a perceptual weighting filter. The weighting operation accounts for the spectral tilt introduced in model estimation.

The weighted signal is used for estimation of parameters of the pitch model. The parameters of the pitch model are estimated in two steps The first step comprises open-loop estimation of a pitch lag. The signal is analyzed every other subframe (10 ms intervals) to find the open-loop pitch lag. After this initial estimation the pitch lag estimation is further refined by a closed-loop estimation algorithm. While performing the refinement the pitch estimator operates on subframes (5 ms).

The first target signal $x_1(n)$ (see figure 3.3) is calculated by weighting the pre-emphasized signal by the analysis filter based on the quantized LPC-based model. This operation is followed by ringing subtraction (zero input response of the weighted syntesis filter is used as ringing). In the standard configuration of AMR-WB the unquantized signal model is used since it turns out that the codec has better perceptual performance by introducing a mismatch between the corresponding operations at the encoder and decoder. In the case of the implementation of the flexible audio coder, an adaptive transformation is used, which requires that the mismatch is eliminated. Therefore the adaptive transform is derived from the quantized model. In addition the quantized model is also used for the weighting of the signal as the usage of the unquantized model turned out to result in a worse performance.

The closed-loop pitch analysis is performed to find both the pitch lag and pitch gain. The lag is searched around the open-loop pitch lag and the resolution is fractional by 1/4 of the sample. The adaptive codebook index (previous decoded blocks) is encoded and the adaptive codebook contribution is removed from the target signal $x_1(n)$ which creates the new target signal $x_2(n)$. This signal is in the weighted signal domain with the mean removed which is the adaptive transform is applied on.

Computation of the signal-adaptive transform that is used in the flexible coding core requires a composite signal model that will be used to obtain an estimate of a covariance matrix. The composite signal model is calculated with the methodology described in 3.2.1 and the adaptive transform is calculated every subframe by the method described in 3.2.3. The standard deviation σ from the model is given by the SVD, which is used by both the distribution preserving quantization and the entropy coder. The signal is then quantized by the means of DPQ which was described in 3.2.4. The dequantized indices (i.e., the reconstructed transform coefficents) are transformed back to the weighted signal domain. The frame is then filtered by the analysis filter to obtain a representation in the excitation domain. This step is necessary to be able to re-use pitch estimation routines that are available in AMR-WB.

The signal gain is calculated using an algorithm of AMR-WB and quantized together with the pitch gain with a gain vector quantizer. The filter memories are finally updated and the next subframe is read for processing.

When processing of all subframes is completed, a full frame of the indices from the quantizer is losslessly coded into a bitstream. The flexible coding approaches uses a constant model bit-rate. As AMR-WB is using a constant bit-rate for the model parameters for its modes associated with higher bitrates [3, 4], quantization from these modes can be used directly in flexible coding.Finally the bitstream is combined with a bitstream of the signal model, as the signal model is coded by a fixed rate. Only the length of the total number of bits needs to be transmitted. The fixed-rate bit-stream needs to be combined with a signal representation that uses a variable bit-rate into a single variable bit-stream. This was one of the implementation tasks in this thesis. The encoder output parameters that are included in the generated bit-stream are shown in table 3.1 where X is the total number of bits in the signal bitstream including the 109 bits from the model. The bits are structured by a fixed bit-mask into octets. As the original bitmask from AMR-WB could not be used, a new bitmask was implemented. The new mask is shown in table 3.1. If the total number of bits is not a multiple of the number octets, the last octet of bits is padded with zeros.

Table 3.1: Encoder output parameters.

Bits	Description
1	VAD-flag
2-9	index of 1st ISP subvector
10-17	index of 2nd ISP subvector
18-23	index of 3rd ISP subvector
24-30	index of 4th ISP subvector
31-37	index of 5th ISP subvector
38-42	index of 6th ISP subvector
43-47	index of 7th ISP subvector
48 – 56	adaptive codebook index
57	LTP-filtering-flag
58-64	codebook gains
65-70	adaptive codebook index (relative)
71	LTP-filtering-flag
72-78	codebook gains
79-87	adaptive codebook index
88	LTP-filtering-flag
89-95	codebook gains
96-101	adaptive codebook index (relative)
102	LTP-filtering-flag
103-109	codebook gains
110-X	signal bitstream

3.3.2 Decoder

The decoder diagram in figure 3.4, is a modified version of the diagram of the AMR-WB decoder that can be seen in [3].

As mentioned in the previous section the decoder of AMR-WB is implemented in the fixed point arithmetics, which introduces a problem as the encoder codebook for the LPC coefficients uses a floating point representation. To solve this problem, the decoding counterpart of the original floating-point encoding blocks was

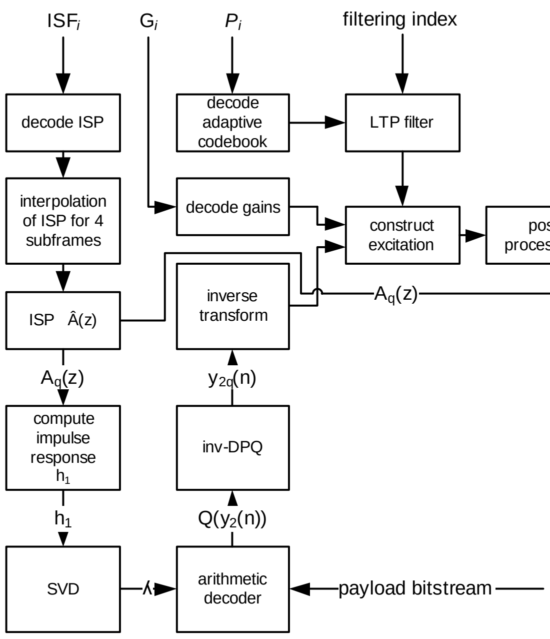


Figure 3.4: Detailed block diagram of the flexible audio decoder.

implemented. The new floating-point functional blocks were used at the decoder to ensure bit-exactness between the encoder and the decoder. This is necessary as the transform that is calculated from the model needs to be the same at the encoder and the decoder to ensure that the encoder and the decoder are in sync.

Once the LPC coefficients are decoded, the covariance matrix is computed. Then the transform and a vector of standard deviations are calculated for the first subframe using the KLT. The entropy encoding was done on a frame basis but the decoding is done on subframe basis. This is since the LTP coefficient (pitch gain) is calculated with dependency on the previous subframe. The LTP coefficient affects the impulse response h which of course affects the SVD, and which finally gives the standard deviation for the arithmetic decoder.

After the bitstream for the first subframe has been decoded into indices they are dequantized. Then the coefficients are transformed back to weighted signal domain. The reconstructed signal is then transformed (filtered) to the excitation domain, where the ringing is added. Then the signal synthesis is performed. Finally, the signal is de-emphasized and up-sampled, which concludes the decoding of a subframe. The same approach is used for the remaining three subframes within a 20ms frame.

Chapter 4

VoIP application

In this chapter, a description of the implementation of the FAC codec in a real-time hardware demonstrator will be provided. The demonstrator is a basic VoIP application that provides full-duplex communications between two computers over a network. The demonstrator operates in real-time and facilitates adaptation to any target bitrate between 8 and 32 kbps. The main goal for the demonstrator is to enable assessment of the flexible operation of the flexible audio coder. A secondary goal is to demonstrate and evaluate the impact of DPQ on the perceptual quality obtained with the flexible audio coder. The demonstrator consists of two PCs operating with Linux. In addition, the demonstrator also facilitates operation with a single PC in a local decoding mode (obtained by a loopback network configuration). If two PC are used, they should be connected with fixed local area network (LAN) on a local network setup that is assumed to provide no packet losses and low delay. See figure 4.1 for a diagram of the setup.

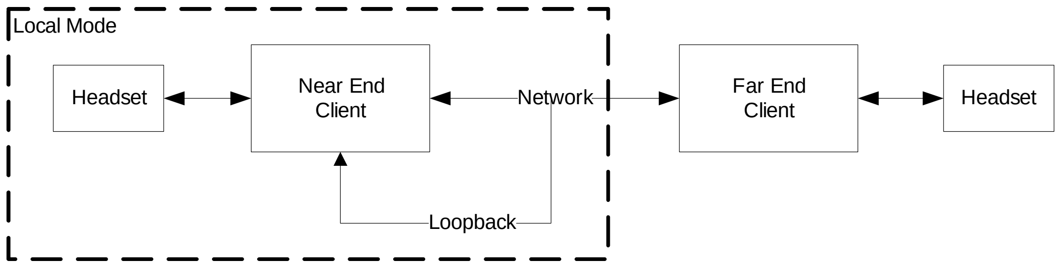


Figure 4.1: Demonstrator setup.

The audio from a laptop computer is commonly routed out through a headset or the internal loudspeaker, while the input audio is provided through a microphone that can either be built in or in an external headset.

Since there is no acoustic echo cancellation (AEC) implemented in the application, headphones must be used to avoid acoustic feedback into the microphone. The demonstrator provides the possibility to control the bitrate and the DPQ from the transmitter to the sender, however if the application is used over a network with full-duplex each transmitter controls the bitrate and the DPQ separately.

The media handling in the demonstrator is provided by the media framework GStreamer, which allows for real-time processing of audio. The media handling in this setup includes audio capture from an audio source at a desired sampling rate, audio playback on selected device and also transmission and reception of data over the network.

The graphical user interface (GUI) is written in GTK+ [29] and therefore it is preferred to use the demonstrator under Gnome or Xfce environment. The GUI provides controls for setting of the target bitrate and a switch to enable and disable DPQ. This enables a comparison of a quality of two available configurations of the platform, DPQ and entropy constrained quantization.

4.1 GStreamer

GStreamer is an open-source media framework which can be used for creating streaming media applications [30]. The framework is flexible and can process any kind of dataflow, however it targets at audio and video applications. GStreamer provides means for constructing a media pipeline that may consist of a number of plug-ins that facilitate data processing within the pipeline.

The plug-in is a library providing a collection of subroutines that implement services within the pipeline. These subroutines are called elements. The elements provide their functionality after inserting them into pipeline and linking appropriately. The most important types of elements and their respective functionalities are as follows:

- Source element: Produce data for the pipeline.
- Sink element: Consumes data from the pipeline.
- Filter element: Process a stream of data in the pipeline.
- Bin element: Contains other elements.

See figure 4.2 for a typical system diagram employing these types of elements.

Sources are usually located at the start of the pipeline where input data is generated. The input data can be read from files, sound sources or network interface for example. There are also sources that generate data by themselves such as elements creating test signals. An element is called a filter if it process incoming data in some way and then sends it further downstream in a pipeline. The filter element can be used to implement signal filters, transforms, codecs or muxers for example. Finally the sink is the output of the pipeline, this element consumes the data stream and may write the data to a file or send the output to a soundcard or network interface. The input and the outputs of the elements are called pads. The pads have properties that are called "caps". The "caps" describe the data that is

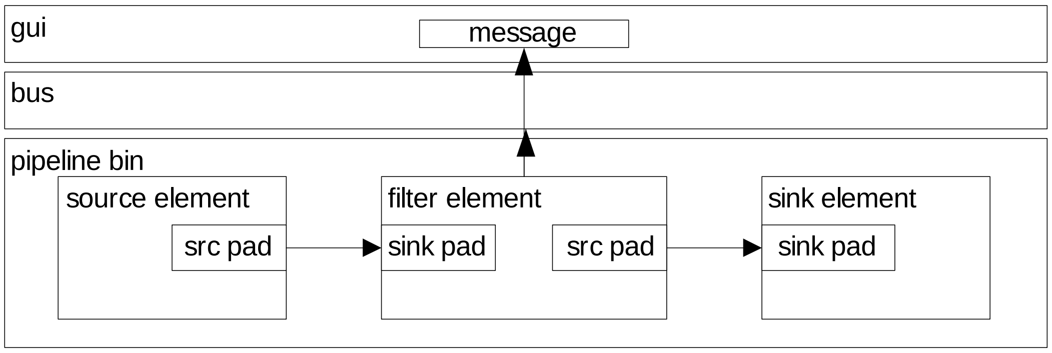


Figure 4.2: Three linked GStreamer elements in a pipeline bin.

flowing out of source pads into sink pads. This implies that the "caps" on the output and input pads between the elements need to be compatible. To transfer the data between elements the data is allocated in a buffer defined as a structure that also contains auxiliary information such as a size of data, time-stamp, duration, caps and offset. To ensure the playback is done in sync, the source element assigns the buffer with a time-stamp that provides the sink element when the samples should be played. A bin is a container for linked elements and a pipeline is sub-type of a bin that controls the state of all elements in the bin, i.e. one can change the state to all elements by changing the state of the pipeline. The different states for an element are:

- NULL: the default state, deallocate all memory hold by the element.
- READY: in the ready state, all memory is allocated but all properties of the element are reset.
- PAUSED: in the pause state, all properties are initiated but the clock is not running, which means that the elements are not processing the data stream.
- PLAYING: same state as pause, but the clock is now running and the data stream in processed.

To allow the pipeline to communicate with the application, the elements can send messages out to a bus. The bus is a system that forwards messages from the pipeline to an application. Every pipeline creates a bus automatically but a message handler needs to be initiated on the bus. This allows the bus to periodically check for messages that have been sent to the bus from the pipeline and forward them to the application. When the application gets a message from the bus a callback function is called which then can check the type of message and then perform the proper operation. This can, for example, be to update the current bitrate to the GUI.

4.1.1 FAC Encoder

To be able to use the FAC coder in full duplex, the encoder and decoder was implemented in two separate plug-ins. A diagram of the encoder plug-in that is named "amrflexenc" can be seen in figure 4.3. The input to the element is a mono

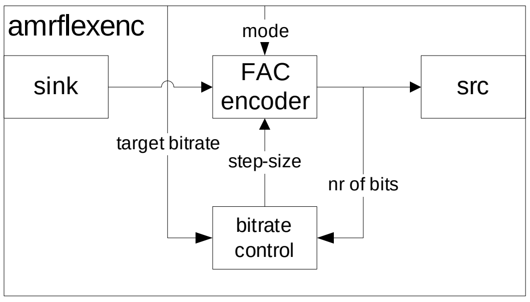


Figure 4.3: Shows the plugin implementation of the FAC encoder.

audio signal with sampling rate 16 kHz. However this does not guarantee that the input buffer to the element has the same size as the input frame to the FAC encoder. To ensure the input frame was always fixed at 320 samples each frame, a class from GStreamer called GstAdapter was used. This object extracts 320 samples from the input buffer to use as an input to the FAC encoder. This is performed in a loop which iterates until the number of bytes available in the buffer is less than the input size N. The bitrate and the modes of the encoder are controlled by two element properties, mode and target bit-rate. The available modes are entropy constrained scalar quantizer and DPQ, the target-bitrate is the maximum encoding bitrate between 8 to 32 kbps. The bitrate control uses feedback to increase and decrease the step size in steps of 0.1 depending whether the bitrate is below or over the target bitrate. The average bitrate is calculated over 50 frames. See algorithm 1 for pseudocode.

Algorithm 1 Bitrate Control

```
Require: frame = 0, bps = 0
  repeat
    encode frame and update frame counter
    update bps := bps + nr of bits
  until frame = 50
  error = target bps - bps
  if error <= 0 then
    increase step size to decrease bitrate.
  else
    decrease step size to increase bitrate
  end if
```

The output data is added to the output buffer along with the length of the package. At the decoder, all information about the incoming package can be read from the package header and the buffer (there are no parameters needing to be specified here). See figure 4.4 for a diagram of the decoder plug-in called "amrflexdec".

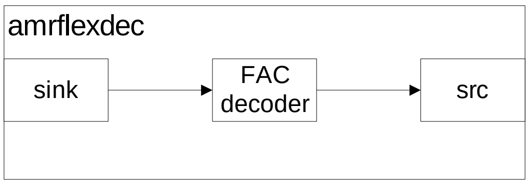


Figure 4.4: Shows the plugin implementation of the FAC decoder.

4.1.2 Real-time Protocol

For the decoder to successfully decode, three additional parameters needs to be transmitted to the decoder. Firstly the step size needs to be transmitted for the dequantization, secondly the mode which indicates if DPQ is used or not and finally the length of the packets.

The length of the packets is simply specified by setting the length of the buffer within GStreamer. The mode is decoded with one bit and finally the step size is simply transmitted as it is. However, the step size is limited between 1 and 26.5 which corresponds to 256 different step size and therefore uses one octet in the packet header.

4.1.3 Client Pipeline

The client pipeline that handles the media can be seen in figure 4.5.

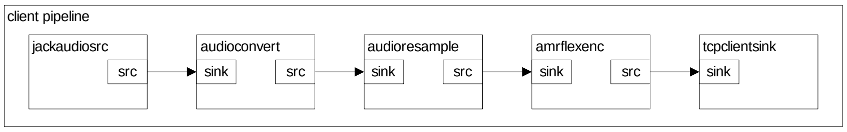


Figure 4.5: Client-pipeline.

The first element in the client pipeline is the audio source element, this generate the data from an external audio source such as a microphone. The next two elements ensure that the input to the encoder element has 16 bit input and 16 kHz sampling rate. The audio is then encoded with the encoder element and finally sent out to the destination with the transmission control protocol (TCP) sink. One may ask why TCP was chosen and not UDP which should be an obvious choice for a VoIP application. The reasons were the following, no forward-error-correction scheme is implemented, if TCP is used this allows for error free packages which can be decoded in order and no jitter buffering needs to be considered. The additional delay that

TCP introduces is not an issue since the demonstrator is supposed to be used on LAN with low delay.

4.1.4 Server Pipeline

The server pipeline that handles the media can be seen in figure 4.6.

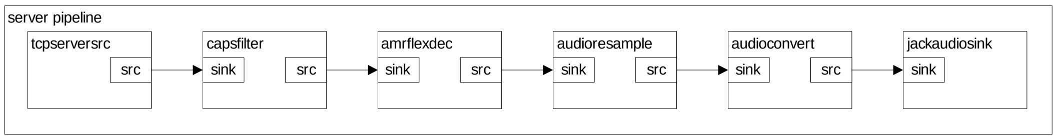
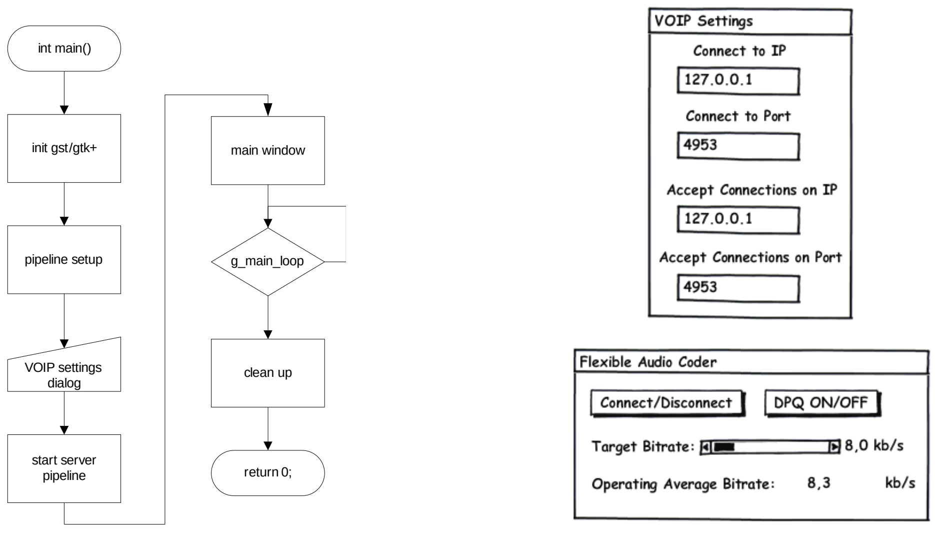


Figure 4.6: Server-pipeline.

The TCP source generates the data from the network and is then sent through a so-called caps filter. The reason why this caps filter is needed is because the caps properties of the data is not preserved over network transfer and had to be reassigned to the buffer to be compatible with the decoder element. The data stream is decoded and finally converted and sent out to playback by the audio element. The chosen sink and source elements is JACK plugins. JACK is a real-time, low latency audio system [31]. It was used since the Linux standard solution for audio ALSA had upsampling problems which caused distortions in the playback.

4.2 GUI

For the GUI implementation, GTK+ was chosen since it is built on GLib 2.0, as is GStreamer. Therefore the API was similar and the coding convention through the application was consistent. See figure 4.2 for a flowchart of the application and a mockup of the GUI. In the left figure in figure 4.2, the flowchart of the main function is shown. Initially GStreamer and GTK+ are initialized. Then the client and server pipeline is set up, to configure the IP settings on the pipelines an IP settings dialog comes up. To use the local mode, one presses OK and usees the current settings. If a full duplex setup with two clients is desired then fill in the IP addresses and ports. When the settings for the connections are provided the server is automatically started, followed by creating the main windows and showing it. The program then enters the main loop. Here the application just idling and listening for signals from the GUI. Signals are activated when some task is performed in the GUI, like pressing a button. If a button is pressed the corresponding callback function is called, for example the client pipeline is started when the connect toggle button is pressed. The DPQ toggle buttons changes the mode in the encoder in the pipeline, the slider adjusts the target bitrate and the operating average bitrate label reports the current bitrate every second. To increase the convergence speed of the bitrate control, the step size is adjusted for every new frame when the slider has been moved instead of once per second.



Connect to IP: 127.0.0.1
Connect to Port: 4953
Accept Connections on IP: 127.0.0.1
Accept Connections on Port: 4953
Target Bitrate: 8,0 kb/s
Operating Average Bitrate: 8,3 kb/s

Figure 4.7: Left: a flowchart of the application. Right: mockups of the GUI.

In addition to the VoIP application a local file stream application was implemented to allow encoding and decoding of an audio file.

Chapter 5

Results

In this chapter, evaluation of the FAC and the real-time implementation is presented. The subjective test of the codec was performed with an informal MUSHRA test and the demonstrator performance was evaluated using practical testing. At the end of this chapter, some discussion is included about the different attributes such as complexity, delay, bit-rate and transmission-error sensitivity.

5.1 Test Plan

The test goal for the evaluation of FAC is to demonstrate its on-par performance to the existing state-of-the-art audio codec. The codec is compared to comparable standardized codecs by means of a subjective listening test to evaluate the relative performance. The performance is understood in terms of subjective quality in combination with attributes such as complexity, delay operating bit-rates and frame concealment robustness. Different workgroups have different criterions and test plans to evaluate their standardized codecs. In the following sections the description of the testplan is provided. Before any subjective test is utilized objective measurements such as PESQ can be used. As time and resources are limited an MUSHRA test was conducted instead of a ACR test. This allowed for a lower number of listeners. Also the number of test excerpts are limited due to limited resources for conducting the listening test.

5.1.1 Subjective Codec Test

For the subjective test a MUSHRA test was chosen as a subjective test. The test is effective because it includes a hidden reference and a bandwidth limited anchor signal. The anchor signal is the reference signal downsampled to have a cut-off frequency of 3.5 kHz. This allows the subjects to experience a low quality reference as well as high quality reference. At higher bit-rates, if the codec have a good performance it can be hard to distinguish the reference from an excerpt with good quality.

See figure 5.1 for an example mockup for a MUSHRA GUI. As seen in the

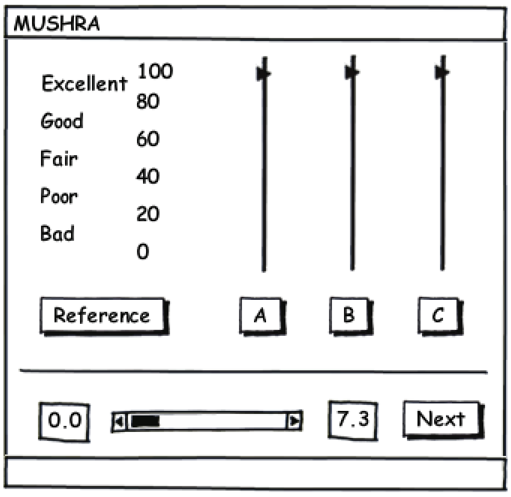


Figure 5.1: Example mockup of MUSHRA GUI.

mockup there are three vertical sliders which indicates that it is one excerpt that is going to be rated against the reference and the anchor. The buttons below the sliders are used to play the excerpts. The horizontal slider at the bottom can be used to narrow down signal to specific parts. When all signals are rated, the subject can go on to the next test signal in the test database.

5.1.2 Codecs Under Test and Selection of the Test Items

The FAC is compared to AMR-WB and G.729.1 which gives a direct indication how the flexible quantization performs against the vector quantizer of AMR-WB and the hybrid coding scheme of G.729.1 (a combination of transform coding and CELP).

To keep the test time low the number of excerpts was limited to five and the number of condition was limited to two 16 kbps and 24 kbps. The excerpts were chosen from the MPEG database and can be seen in table 5.1

Table 5.1: Test items.

Type	File Name	Description
Speech	es02.wav	Male German speech
Speech	es03.wav	Female English speech
Complex Sound Mixture	sc01.wav	trumpet solo and orchestra
Single Instrument	si01.wav	harpsichord
Simple Sound Mixture	sm01.wav	bagpipes

5.1.3 Processing of the test samples

All processing of the samples was performed with tools from the ITU-T toolset G.191 [32]. Firstly the samples were downsampled to 16 kHz and prefiltered by a P341 filter from the ITU-T toolset. Then the samples were processed by the codecs at two different operating bit-rates 16 kbps and 23 kbps. For AMR-WB these bit-rates correspond to modes 4 and 7 [4]. The target bit-rates were calculated for FAC as the average target bit-rate with a fixed step size for the whole excerpt. The anchor signal was created by downsampling the reference signal to 7 kHz. Finally all the coded excerpts were upsampled to 48 kbps as a requirement for the test setup. The codecs are mono so the files are played out in a diotic presentation.

5.1.4 Listening Panel and Test setup

Since the test subject could not be compensated for their time spent, it was acceptable with any number of participants with any experience of listening test. The listening panel that participated in the MUSHRA test consisted of 10 listeners with previous experience in listening tests based on MUSHRA. The listeners had no training before the session and were asked to evaluate the perceived quality of the excerpts compared to the reference.

The playback device was a laptop PC. The PC had an external soundcard Phase 26 from Terratec with headphones DT 770 PRO from Beyerdynamic. The listeners could adjust the loudness to a comfortable level.

5.2 Results of Subjective Test

The subjective tests were done in two sessions, first a test with excerpts coded at 16 kbps followed by a test at 24 kbps. All of the listeners identified the references in both tests without errors. The test results for 16 kbps can be seen in Figure 5.2. The average value is used with 95% confidence intervals for the 10 test subjects. Here it can be seen that AMR-WB get a slightly higher score than the FAC, but both of these get a higher score than G.729.1. In the 24 kbps test FAC performed better than both AMR-WB and G.729.1 and the result can be seen in Figure 5.3

A relatively large extent of 95% confidence intervals is due to a small number of test excerpts that were used.

5.3 Results of Objective Test

Due to limited resources the perceptual evaluation of FAC was used only for two operating bit-rates. In order to demonstrate that trade-off between the rate and the quality achieved by FAC an objective quality assessment tool was used. The codec has been evaluated for the speech excerpts by means of PESQ, which is an objective quality evaluation tool for speech.

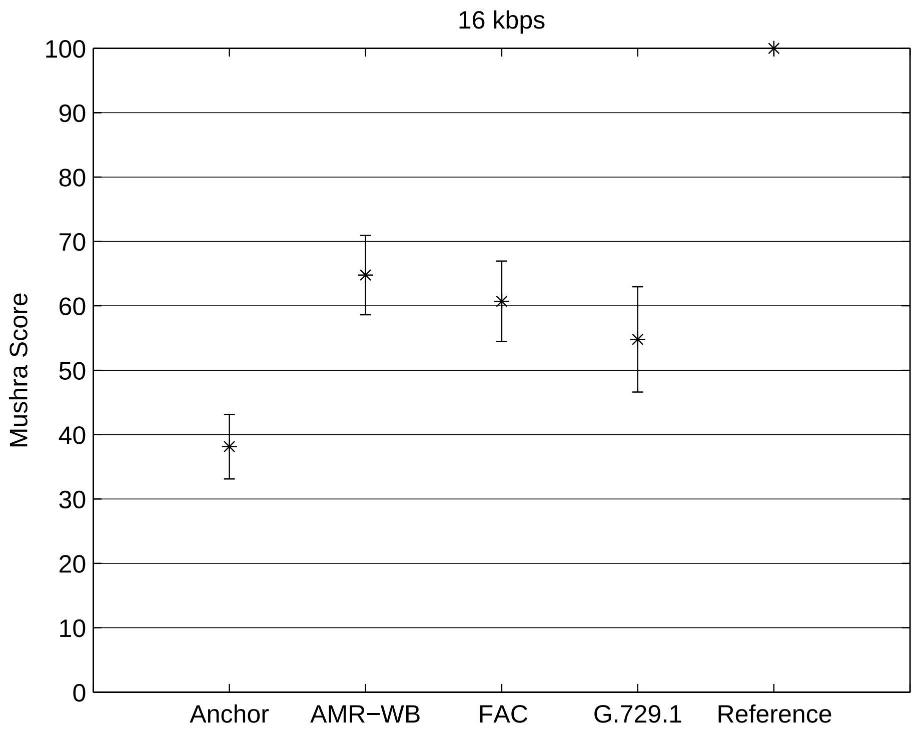


Figure 5.2: Subjective scores from MUSHRA test.

PESQ measurements were done to ensure that the quality improved when the bit-rate increased. The MOS scores is plotted in Figure 5.4. The objective results indicates that the objective quality indeed increases with increasing bit-rate.

5.4 Real-time Demonstrator

The real-time demonstrator operates in real-time. The coding delay of the demonstrator consists mostly of the algorithmic delay of 25 ms of the codec and that the audio interface JACK. The operating bit-rates were between 7 and 32 kbps. However, as hinted previously, the encoder and decoder has high complexity due to the SVD and the DPQ scheme. On a laptop with a dual core CPU where each core is operating at 1800 MHz, the implemented FAC application utilized approximately 50% of each core. It should be noted that no hand optimizations were made and it is probably possible to reduce the complexity. Unfortunately, the transmission error sensitivity is limited, as each composite model is dependent on previous blocks pitch-gain. If this is not correct the encoder and decoder loses sync and never recovers due to the transform. This can be prevented by restructuring the core of AMR-WB, and quantize the pitch gain before the signal quantization. This

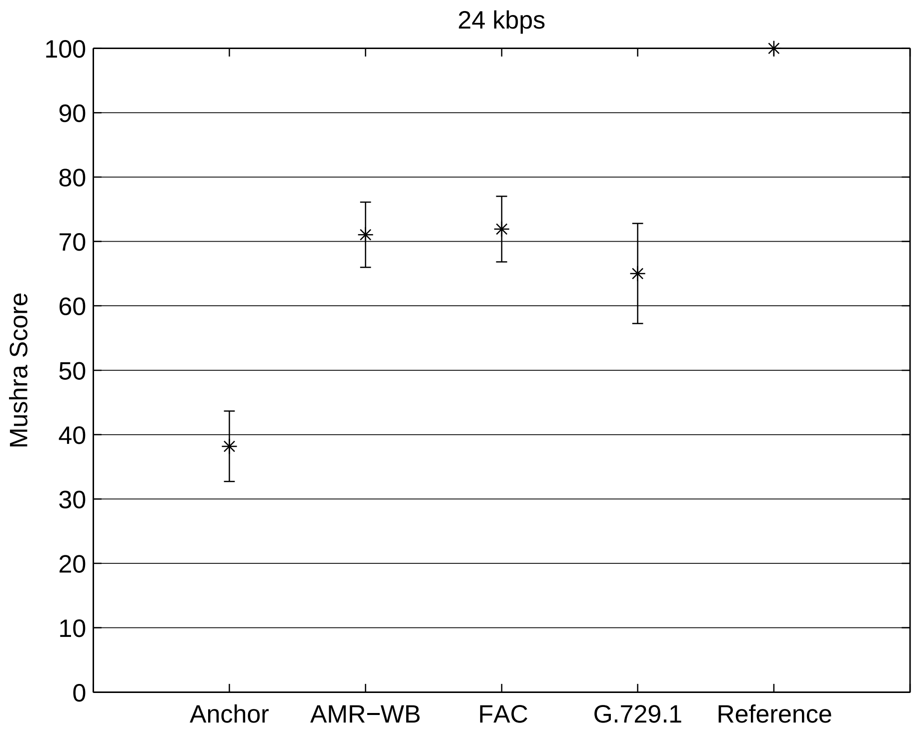


Figure 5.3: Subjective scores from MUSHRA test.

is however complex and was not attempted in this implementation.

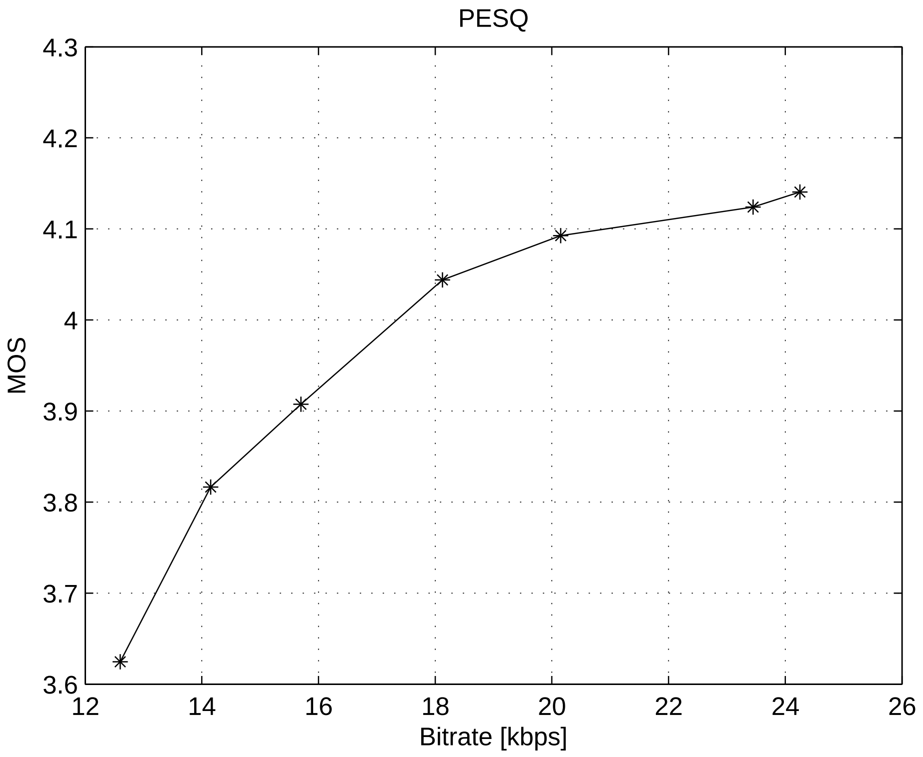


Figure 5.4: Objective MOS score from PESQ.

Chapter 6

Discussion and Conclusions

6.1 Real-time Demonstrator

The real-time demonstrator shows that the proposed coding scheme can be used in real-time operation. The demonstrator was implemented as a VoIP application, which could operate in real-time in a local mode or in a network communications mode. In addition another application, that could playback audio files was implemented for easier demonstration of the FAC.

6.2 Codec Performance

The conclusion from the MUSHRA test is that this specific implementation of the FAC performs comparable to the state-of-the-art codecs such as AMR-WB and G.729.1 in terms of subjective quality. It is worth noting that the state-of-the-art codecs were highly optimized for their respective operating points, while the FAC facilitates operation at any bit-rate.

6.3 Conclusions

The work of this thesis demonstrates that it is possible to apply the flexible audio coding scheme for an audio communication over a network. In addition, it is possible to obtain an audio coding system that can be redesigned on the fly in a response to a constraint on the operating bit-rate achieving at least the state-of-the-art performance.

A demonstrator was implemented in Linux with aid of the GStreamer framework and a GUI was implemented. The application demonstrated real-time operation and also instant re-optimization of the scheme. The coding performance was on-pair with non-flexible schemes at two bit-rates.

6.4 Future Work

Due to lack of time some parts could not be improved. Some proposals are described below

6.4.1 Real-time Demonstrator

The real-time demonstrator could be improved in a number of ways. Some suggestions is to.

- Better step size quantizer
- Create a feedback channel and let the decoder decide the operating bit-rate
- Implement using UDP with jitter buffer
- Improve stability and robustness of the application

6.4.2 Codec

Instead of using AMR-WB, another codec might be used as a core such as SILK and replace the DDNSQ with the one used in this thesis. Care can be taken to ensure that there are no block dependencies for calculation of the adaptive transform.

Bibliography

[1] ITU-T Recommendation G.722.1, “Low-complexity coding at 24 and 32 kbit/s for hands-free operation in systems with low frame loss,” May 2005.

[2] H. Malvar, “Lapped transforms for efficient transform/subband coding,” *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 38, no. 6, pp. 969–978, 1990.

[3] ITU-T G.722.2, “Wideband coding of speech at around 16 kbit/s using adaptive multi-rate wideband,” June 2003.

[4] 3GPP TS 26.193, “Speech codec speech processing functions: Adaptive multi-rate - wideband,” Januari 2005.

[5] J. Paulus and J. Schnitzler, “16 kbit/s wideband speech coding based on unequal subbands,” in *ICASSP*. IEEE, 1996, pp. 255–258.

[6] C. Laflamme, J. Adoul, H. Su, and S. Morissette, “On reducing computational complexity of codebook search in CELP coder through the use of algebraic codes,” in *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*. IEEE, 1990, pp. 177–180.

[7] 3GPP TS 26.290, “Audio codec processing functions; extended adaptive multi-rate - wideband,” December 2008.

[8] ITU-T G.729.1, “G.729-based embedded variable bit-rate coder: An 8-32 kbit/s scalable wideband coder bitstream interoperable with G.729,” May 2006.

[9] A. Poularikas, *Transforms and applications handbook*. CRC, 2009.

[10] J. M. Vallin and K. Vos, “Definition of the Opus Audio Codec,” *IETF Standards Track Internet-Draft*, 2011.

[11] K. Vos, S. Jensen, and K. Soerensen, “Silk speech codec,” *IETF Standards Track Internet-Draft*, 2009.

[12] J. Valin, T. Terriberry, and G. Maxwell, “A full-bandwidth audio codec with low complexity and very low delay,” 2009.

[13] A. Ozerov and W. B. Kleijn, "Flexible quantization of audio and speech based on the autoregressive model," in *Signals, Systems and Computers, 2007. ACSSC 2007. Conference Record of the Forty-First Asilomar Conference on.* IEEE, 2008, pp. 535–539.

[14] J. Klejsa, M. Li, and W. B. Kleijn, "Flexcode-flexible audio coding," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on.* IEEE, 2010, pp. 361–364.

[15] M. Li, J. Klejsa, and W. B. Kleijn, "Distribution Preserving Quantization With Dithering and Transformation," *Signal Processing Letters, IEEE*, vol. 17, no. 12, pp. 1014–1017, 2010.

[16] W. B. Kleijn, "A basis for source coding," lecture notes KTH, Stockholm, 2010.

[17] P. Vary and R. Martin, *Digital speech transmission.* Wiley Chichester (West Sussex), 2006.

[18] B. Moore, *An introduction to the psychology of hearing.* Emerald Group Pub Ltd, 2003.

[19] ITU-R BS.1534, "Method for the Subjective Assessment of Intermediate Sound Quality (MUSHRA)," pp. 1543–1, 2001.

[20] ITU P. 862.2, "Wideband Extension to Recommendation P. 862 for the Assessment of Wideband Telephone Networks and Speech Codecs," 2005.

[21] ITU-R BS.1387-1, "Method for objective measurements of perceived audio quality," 1998.

[22] J. Benesty, M. Sondhi, and Y. Huang, *Springer handbook of speech processing.* Springer Verlag, 2008.

[23] A. Aho and J. Ullman, *Foundations of computer science.* WH Freeman & Co., 1994.

[24] ITU G. 718, "Frame error robust narrowband and wideband embedded variable bit-rate coding of speech and audio from," 2006.

[25] W. B. Kleijn and A. Ozerov, "Rate distribution between model and signal," in *Applications of Signal Processing to Audio and Acoustics, 2007 IEEE Workshop on.* IEEE, 2007, pp. 243–246.

[26] F. Luk and S. Qiao, "A fast singular value algorithm for Hankel matrices," in *Fast algorithms for structured matrices: theory and applications: AMS-IMS-SIAM Joint Summer Research Conference on Fast Algorithms in Mathematics, Computer Science, and Engineering, August 5-9, 2001, Mount Holyoke College, South Hadley, Massachusetts*, vol. 323. Amer. Mathematical Society, 2003, p. 169.

[27] C. M. Liu, H. W. Hsu, and W. C. Lee, “Compression artifacts in perceptual audio coding,” *IEEE Transactions on Audio, Speech and Language Processing*, vol. 16, no. 4, pp. 681–695, 2008.

[28] E. Bodden, M. Clasen, and J. Kneis, “Arithmetic coding revealed,” *Proseminar Datenkompression 2001*, 2001.

[29] Gtk+. [Online]. Available: http://www.gtk.org/

[30] W. Taymans, S. Baker, A. Wingo, R. S. Bultje, and S. Kost, *GStreamer Application Development Manual.*

[31] P. Davis and T. Hohn, “Jack audio connection kit,” in *Linux Audio Developers Conference*, vol. 2003, 2003.

[32] ITU-T Rec. G.191, “Software tools for speech and audio coding standardization,” 2005.