Project no: FP6-2002-IST-C 020023-2

Project title: FlexCode

Instrument: STREP

Thematic Priority: Information Society Technologies

D1.1: Baseline Source Coder

Due date of deliverable: 2007-02-01

Actual submission date: 2007-05-29 (latest version)

Start date of project: 2006-07-01 Duration: 36 Months

Organisation name of lead contractor for this deliverable: KTH

Revision: 1.12

Project co-funded by the European Commission within the Sixth Framework Programme 2002-2006		
Dissemination Level		
PU	Public	**AFTER JULY 1, 2008**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	**UNTIL JULY 1, 2008**

Table of Contents

Abstract

This report describes the technologies and implementation of the baseline source coder of the *FlexCode* project. The *FlexCode* source coding approach is motivated by the increasing need for source coders that can adapt to the network conditions and user requirements at any time. High-rate quantization theory, the so-called sensitivity-matrix, and multiple-description coding form vital components of the approach. A set of new technologies that were developed during the first 18 months of the project is discussed. These include a method to determine the rate distribution between signal and data, a scalable quantization method based on bit-plane coding, new methods for enhancing quantization efficiency based on signal replication, improved methods for lattice quantization, and sensitivity-matrix based approaches towards perceptual modeling. Finally, the practical implementation of the baseline *FlexCode* source coder is described. This includes a discussion of the coder architecture and coder specifications, a brief description of each of the routines and a brief analysis of the computational complexity of the coder.

Chapter 1

Introduction

1.1 Introduction

The general objective of the *FlexCode* project is to to develop a new generation of generic coding algorithms that are significantly more flexible than existing algorithms, which generally are application-specific, while retaining or improving on the efficiency and quality of current state-of-the-art algorithms. The objective addresses the coding needs for a rapidly growing infrastructure providing an increasing diversity of services with an increasingly heterogeneous network.

The algorithm development work of *FlexCode* is separated into source coding and channel coding. Work package 1 (WP1) deals with source coding and WP2 with channel coding. This report is part of deliverable D1.1, which is the first deliverable of WP1.

WP1 has as objective to develop a source coding methodology and its implementation. The specific objectives of WP1 are

- to develop generic, low-complexity models of perception;
- to provide a methodology for a flexible, efficient, low-complexity, generic source coder;
- to create a practical implementation aimed at audio signals.

The flexibility in the *FlexCode* source coder allows for instantaneous adaptation of the coding to satisfy rate, quality, and robustness requirements by using computable quantizers and probabilistic signal models. Flexibility is also achieved by using innovative embedded coding techniques that allow the decoding of the bit stream even when layer(s) of bits have been removed after encoding (for example if a particular transmission leg or receiver has low capacity). The work of WP1 includes multiple-description coding (MDC) techniques that address packet loss (as MDC is an integral part of source coding, it is natural to include it in WP1). Again the emphasis is on creating flexible coding: the new MDC coding schemes are *scalable*, thus facilitating the instantaneous adaptation of the level of robustness to the packet loss rate.

The *FlexCode* source coder is based on high-rate theory. High-rate theory allows the analytic optimization of the source coder. The quantizers and MDC configuration are specified by means of equations that are based on probabilistic signal models. The equations have analytic solutions that can be adapted to the quality and network requirements at hand. The source coder can migrate seamlessly from single-description coding to multiple-description coding. Bit-plane coding techniques lead to a layered bit stream that facilitates a reduction in rate after encoding.

The high-rate theory based source coding approach is complemented with the sensitivity matrix approach used for describing sophisticated perceptual models. The sensitivity matrix approach replaces distortion measures that would make quantization computationally intractable by a local quadratic approximation. This allows the introduction of sophisticated distortion perception-based measures that have hitherto not been used for source coding because of practical problems.

1.2 Description of the Source Coder

As stated in Section 1.1, the *FlexCode* approach is based on high-rate quantization theory and the sensitivity matrix. This approach naturally leads to a flexible source coder that can adapt to rate, quality, and packet-loss conditions in real time. To make the principles work well in complete system, a number of new technologies were developed in WP1. In addition, to illustrate the approach a practical audio coder is under development in WP1. The audio coder is to operate at rates between 10 and 30 kb/s. The baseline version of this practical coder, which operates on a 16 kHz sampled signal, is described in this report.

Existing methods towards audio and speech coding that operate at rates between 10 and 30 kb/s can be separated roughly into linear-prediction (autoregressive model) based and transform based coding methods. As was envisioned, flexible source coding schemes based on both linear prediction based coding and transform coding were initially developed under *FlexCode*. It was found that the two approaches naturally converge and only one *FlexCoder* source coder, albeit with different options, is currently under development. The coder has contributions to its components from the partners KTH, Ericsson, France Telecom, and Nokia.

In this Section, we first discuss the natural convergence of the flexible prediction and transform approaches when scalable high performance is required. We then discuss how this combines with the theory based approach towards quantization and multiple description coding.

1.2.1 Convergence of Predictive and Transform Based Approaches

To increase the efficiency of quantization of low-dimensional data vectors, coding architectures are designed to minimize the inter-dependencies between vectors or scalars that are to be quantized. The most common architectures are predictive and transform coding. This section shows that even when starting from these different vantage points, the requirement for high efficiency leads to similar coding structures. For the purpose of simplicity, and without loss of generality of the discussed reasoning, we assume that the distortion measure is a simple squared error criterion.

If a predicted value for the next signal sample is subtracted from this sample at the encoder before being quantized, and added to the quantizer output at the decoder, then the dynamic range of the signal that is to be quantized is reduced. For a scalar quantizer, the reduction of the dynamic range of the signal leads to a reduced quantization error at a given fixed or mean coding rate. As the predicted sample must be identical at both encoder and decoder, the decoded signal must be used for prediction in both cases and the procedure is referred to as *closed-loop* prediction. It is well-known (e.g., [1]) that linear-prediction based analysis-by-synthesis (LPAS) methods such as the ubiquitous CELP algorithm [2] are generalizations of closed-loop prediction.

As discussed in [3, 4], closed-loop prediction (and LPAS) methods have a significant disadvantage. *The quantization cell shape of predictive coding is not preserved during the closed-loop prediction process.* Better performance can be obtained if the prediction is not seen as simply a method to reduce correlations but as the estimation of a probabilistic model of the signal [3]. That is, the signal is viewed as having been generated by an autoregressive process and the parameters specifying this process are estimated with the linear prediction analysis. The probabilistic model approach suggests direct quantization in the signal domain using the signal model. Unfortunately, coding in the signal domain based on this signal model is difficult since it requires that quantizers must be available for each realization of the probabilistic signal model. This leads to an intractable quantizer design and storage problem.

A practical method towards quantization in the signal domain using a probabilistic autoregressive model was proposed in [3, 4, 5] (and earlier associated conference publications). To make coding practical, an adaptive unitary transform of the zero-state response signal is used. The transform is selected to describe correlations associated with the autoregressive signal model. For each successive signal block the transform is the Karhunen-Loève transform for the zero-state response of the autoregressive model for that signal block. Scalar quantization in the transform domain is now effective since the correlations between the samples are removed, and since reverse waterfilling, e.g., [6], can be invoked. Knowledge of the signal statistics facilitates the analytic design of optimal quantizers. Since the transform is unitary, it preserves the geometry of the signal, thus preserving the quantization cell shape, eliminating the loss in coding gain associated with predictive coding. As is shown in [3, 5] the method performs well even for short transform blocks, thus preserving the low-delay coding delay advantage of prediction-based coding.

To place our work in context, we note that the TCX coding work of [7] is to some extent similar in spirit to [3, 5], as it uses autoregressive model and the Fourier transform and codes in a *colored* spectral domain. The main source coding *FlexCode* innovations over this earlier work is the probabilistic interpretation of the autoregressive models, which facilitates quantizer design without further consideration of the data, and the usage of the Karhunen-Loève transform that is optimal for the model of the block rather than a fixed Fourier transform. Other work that resembles the *FlexCode* approach is [8] and [9], where the signal is first subjected to a prediction error filter, flattening the signal spectrum, prior to a transform with a discrete cosine transform and quantization with a vector quantizer. In this case the quantizers are trained and not optimal for the block at hand. These earlier approaches do not use a probabilistic interpretation of the signal model.

Fixed transforms are commonly used in source coding. Sinusoids are eigenfunctions of time-invariant linear operators and, asymptotically with increasing block length, form the eigenvectors of the covariance matrix of a stationary signal. This means that discrete Fourier transform (DFT) and discrete cosine transform

(DCT) decorrelate stationary signals. As stated above, decorrelation leads to efficient coding with scalar quantizers. As a result block-based Fourier and cosine transforms are commonly used in audio coding. In practice, the DCT converges faster to ideal decorrelation behavior with increasing block length than the DFT [1]. More-over, the DCT removes the issues stemming from the usage of complex representations. As a result, lapped transforms based on the DCT are commonly used for audio coding. Perfect-reconstruction lapped transforms were invented in the context of audio coding and can be traced back to the original work of Princen and Bradley [11].

Transform coding based on the DCT requires the specification of a quantizer step size for the transform coefficients. In earlier audio coders (e.g., [12, 13]), information relating to the signal power and the masking level for each of a set of frequency bands is transmitted to the decoder as side information. It has not been common to use an explicit probabilistic signal model for this purpose, although, as mentioned above, spectral models have been used, e.g. [7, 14]. At low rates the benefit of probabilistic signal models is well motivated as such models facilitate the encoding of the power spectrum with high efficiency. Since the encoding of the autoregressive model parameters is well understood, usage of the autoregressive model to describe the spectral density of the signal is natural.

The reasoning described above has led to a single *FlexCode* coder implementation that has two main configurations that are both based on a (Gaussian) autoregressive model assumption:

- Fixed-transform configuration: the coder is based on a critically-sampled perfect-reconstruction DCT filterbank with smooth overlapped windows for each block. The rate allocation for the coefficients is based on the estimated probabilistic signal model. The transform coefficients are quantized using to embedded scalar quantizers that can be adapted to rate and quality requirements. These quantizers will be replaced by MDC quantizers in the future. The method is relatively low computational complexity.

- Adaptive-transform configuration: for each signal block the zero-input response of the autoregressive signal model is subtracted first. The zero-response autoregressive-model based Karhunen-Loève transform is then applied to the remainder signal. The resulting coefficients are quantized as in the fixed-transform setup. The computational complexity for the computation of the Karhunen-Loève transform is relatively high.

1.2.2 The *FlexCode* Source Coding Approach

The *FlexCode* source coding approach includes a number of significant innovations. We describe the main innovations. The firs main innovation is that the signal is quantized with scalable quantizers that are directly adapted to the probabilistic source model and to the distortion at hand. This implies that the average or instantaneous rate can be reset at any time, as a function of network or user quality constraints. The second main innovation is that the quantization indices can be encoded using an effective embedded arrangement such that when bits can be

[1]The DCT has better symmetry properties and only the DCT facilitates critical sampling of transforms with overlapping smooth windows between the blocks (e.g., [10]).

stripped from the encoded signal representation the bit stream remains decodable, at a minimal loss of overall efficiency. The third innovation is that the source coding approach uses a *scalable* multiple-description coding (MDC), facilitating a continuously variable trade-off between robustness against packet loss and bit rate for a given level of reconstruction fidelity. We anticipate to use that the coder will be configured to use either encoding embedded coding or MDC, but not both, as existing results [15, 16] suggest relatively little benefit from joint optimization. The fourth innovation is that the distortion of the signal is described with a sensitivity matrix approach, which facilitates the usage of sophisticated distortion measures without a significant computational penalty. It does this by locally approximating the distortion with a quadratic model. Thus, the complex static model is replaced by an adaptive quadratic model.

The innovations are combined in a source coding architecture that exploits the source model both to compute the adaptive quadratic distortion model and to specify the setting of the quantizers. In the following we outline the encoding architecture, including some comments on their functionality:

1. The probabilistic model for the signal is estimated. This estimation process takes the form of a linear-predictive analysis specifying the autoregressive model coefficients. The model can take various levels of sophistication in terms of model order and in terms of modeling the pitch structure of the audio signal by means of dedicated autoregressive models. It is noted that very high precision is required for autoregressive models dedicated to pitch.

2. The probabilistic signal model is quantized using conventional approaches for quantizing autoregressive models. Conventional methods can be used since it was proven in the project (cf. Section 2.2) that the rate for the model is independent of the overall rate.

3. The signal model is used to specify a pre-processing step that renders an unweighted squared-error criterion a good approximation of perceived error (as described in [17]). That is, the signal is now represented in a "perceptual-domain". The *FlexCode* approach makes this pre-processing possible for sophisticated distortion measures because it is represented with an *adaptive* squared-error criterion. To this purpose an analysis of the sophisticated distortion measure must be applied on the reconstructed signal, which is available at both encoder and decoder.

4. The perceptual-domain signal is transformed to a representation that has relatively independent coefficients using either the Karhunen-Loève transform (after subtraction of the zero-input response of the autoregressive filter) or the lapped DCT.

5. The transformed coefficients are quantized using adaptive scalar quantizers or adaptive lattice quantizers. Depending on the setting of the coder, the quantizers are optimized for constrained-entropy or contrained-resolution quantization. This step will be combined with adaptive MDC in later versions of the coder.

6. The quantization indices are prepared for transmission. In the case of

constrained-entropy coding this includes the step of lossless encoding using arithmetic coding.

The decoder performs the corresponding decoding steps in inverse order.

1.3 Report Overview

This report consists of three chapters: this introduction, a chapter introducing new technologies, and a chapter describing the practical implementation of the baseline coder.

Chapter 2 focuses on a particular subset of new technologies developed for *FlexCode* . All these technologies have been implmented individually and will be part of the *FlexCode* source coder. However, some of the methods have not yet been implemented in the current system because of time constraints. The technologies will be supplemented by additional technologies that are currently being developed. The new technologies that Chapter 2 describes are rate distribution between model and signal data (Section 2.2), bit plane coding to allow rate reduction after coding (Section 2.3), techniques that use signal replication to enhance the efficiency for coding the data (Section 2.4), efficient approaches towards lattice quantization (Section 2.5) and sensitivity-matrix based approaches towards perceptual modeling (Section 2.6).

Chapter 3 describes the practical implementation of the baseline *FlexCode* source coder. It starts with a description oft the coder architecture and coder specifications. It describes in detail how the *FlexCode* principles were incorporated in the implementation. A brief description of each of the routines used is provided. The computational complexity of the coder is analyzed. Finally a brief results section provides some indication of the performance of the baseline coder as described in this chapter.

Bibliography

[1] W. B. Kleijn and K. K. Paliwal, "An introduction to speech coding," in *Speech Coding and Synthesis*, W. B. Kleijn and K. K. Paliwal, Eds. Elsevier, 1995, pp. 1–47.

[2] B. S. Atal and M. R. Schroeder, "Stochastic coding of speech at very low bit rates," in *Proc. Int. Conf. Comm.*, Amsterdam, 1984, pp. 1610–1613.

[3] M. Y. Kim and W. B. Kleijn, "KLT-based adaptive classified VQ of the speech signal," *IEEE Trans. Speech Audio Processing*, vol. 12, no. 3, pp. 277–289, 2004.

[4] A. Ozerov and W. B. Kleijn, "Flexible quantization of audio and speech based on the autoregressive model," in *Proceedings Asilomar Conference on Signals, Systems & Computers*, Nov. 2007.

[5] M. Li and W. B. Kleijn, "A low-delay audio coder with constrained-entropy quantization," in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, Nov. 2007, pp. 191–194.

[6] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.

[7] R. Lefebvre, R. Salami, C. Laflamme, and J.-P. Adoul, “High quality coding of wideband audio signals using transform coded excitation (TCX),” in *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, vol. 1, pp. 193 – 196.

[8] N. Iwakami, T. Moriya, and S. Miki, “High-quality audio-coding at less than 64 kbit/s by using transform-domain weighted interleave vector quantization (TwinVQ),” in *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, vol. 5, 1995, pp. 3095 – 3098.

[9] T. Mirya, N. Iwakami, A. Jin, K. Ikeda, and S. Miki, “A design of transform coder for both speech and audio signals at 1 bit/sample.”

[10] I. Daubechies, *Ten lectures on wavelets*. Philadelphia: SIAM, Academic Press, 1992.

[11] J. Princen and A. Bradley, “Analysis/synthesis filter bank design based on time-domain aliasing cancellation,” *IEEE Trans. Acoustics, Speech, Signal Processing*, vol. 34, pp. 1153–1161, 1986.

[12] M. Bosi, K. Brandenburg, S. Quackenbush, L. Fielder, K. Akagiri, H. Fuchs, M. Diets, J. Herre, G. Davidson, and Y. Oikawa., “ISO/IEC MPEG-2 Advanced Audio Coding,” *J. Audio Eng. Soc.*, vol. 45, no. 10, pp. 789–812, 1997.

[13] J. Herre and B. Grill, “Overview of MPEG-4 audio and its applications in mobile communications,” in *Proc. Int. Conf. Signal Process.*, Beijing, 2000, pp. 11–20.

[14] A. Härmä, U. K. Laine, and M. Karjalainen, “An experimental audio codec based on warped linear prediction of complex valued signals,” in *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, vol. 1, 1997, pp. 323–327.

[15] C. Hegard and T. Berger, “Rate distortion when side information may be absent,” *IEEE Trans. Inf. T.*, vol. 31, pp. 727–734, 1985.

[16] C. Tian, J. Chen, and S. N. Diggavi, “Multiuser successive refinement and multiple description coding,” *IEEE Trans. Inf. T.*, vol. 54, pp. 921–931, 2008.

[17] B. Edler and G. Schuller, “Audio coding using a psychoacoustic pre- and post-filtering,” in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Istanbul, 2000, pp. 881–884.

Chapter 2

New Source-Coding Technology

2.1 Introduction

This chapter highlights a set of new source coding technologies that were developed for *FlexCode*. The technologies are aimed towards a scalable, robust coding system. In this section we provide a brief overview of the technologies, which are then described in detail in the remaining sections of this chapter.

The first technology that we describe relates to rate allocation. In almost any practical source coder models are used. Except for coders operating at very low delay, the model is transmitted to he decoder. This means that one has to decide on the distribution of rate between the data and the model. Section 2.2 describes a method that shows that, if the signal is divided into segments of a particular duration, and the model structure is fixed, then the optimal bit allocation for the model parameters does not vary with the overall rate. The distribution is worked out in details for the autoregressive (AR) model case. It is shown that the square error criterion in the signal domain is consistent with the commonly used root mean square log spectral error for the model parameters. Interesting, even without the usage of perceptual knowledge, we obtain a rate allocation for the model that is consistent with what is commonly used. Section 2.2 follows [1] closely.

The rate-distribution technology of section 2.2 is directly relevant to FlexCode. The source coder of FlexCode uses an AR model, and instead of a conventional search for optimal rate distribution, we select a fixed rate for the AR model independent of overall rate. This significantly speeds up the development process.

The second technology that we present relates to model-based bit plane coding, Most transform audio coders are built upon modified discrete cosine transform (MDCT), scalar quantization and Huffman coding. Huffman coding is inflexible in the sense that it depends on a particular training sequence and requires storage of coding tables. Huffman coding may be replaced by arithmetic coding, which in general has a better performance. The cornerstone is then to define properly the symbols to be arithmetically coded and to estimate efficiently the related probabilites. We develop here a new model-based method that applies arithmetic coding in bit planes to bring the additional flexibility of bitstream scalability. Section 2.2 is adapted after [2].

The bit plane coding technology described in section 2.3 has been developed for transform speech and audio coding. It finds a direct application in the MDCT coding part of the Flexcode source coder.

A new technology, entitled Multi-mode excitation reconstruction, was developed for FlexCode. This technology provides FlexCode with a mechanism to dynamically select parts of the spectra that should not be quantized, but reconstructed from other spectra regions. The key idea is that the codec runs in multiple modes, in which continuous parts of spectra are quantized. Then minimization of pre-defined criteria selects the optimal mode.

The presented technology improves the robustness of FlexCode at low bitrates, without introducing artifacts when sufficient numbers of bits are available, and the entire spectra can be quantized. This makes the codec competitive at larger range of bitrates and signal bandwidths. Detailed description of the technology, as well as simulations that prove its efficiency, can be found in Section 2.4.

Section 2.5 presents various new techniques of lattice quantization, both in the context of rate constrained, as well as entropy constrained coding. The interest in lattices as quantizers comes from their reduced memory requirements for storage, low complexity encoding and decoding algorithms, their optimality for uniform sources and approximate optimality if used in conjunction with entropy constrained coding. Within the FlexCode, lattice quantization are employed both for the signal model and for transform coefficients, possibly in conjunction with Gaussian mixture models.

The techniques described in section 2.5 provide practical non-complex ways of vector quantizers and they are successfully applied to the quantization of the line spectrum frequencies of the audio signal within the rate constrained approach. In addition, the proposed lattice rotation techniques allowing efficient use of the structured quantizers for moderate and low bit-rates, enabling for the FlexCode the coverage of a larger bit-rate domain. The proposed entropy coding methods enable the use, within practical limits, of lattice quantizers for higher dimensions, in the context of entropy constrained coding.

Section 2.6 describes two new technologies for the integration of perceptual modeling in *FlexCode*. The first is multidimensional companding, which facilitates separation of perceptual aspects from the quantization and coding stages. This offers flexibility in the choice and adaptation of perceptual models while retaining optimality at all individual stages of the coder. The second technology is the derivation of spectral perceptual model parameters from the signal model to maximally exploit the redundancy between the two models.

As described in detail in section 2.6, *FlexCode* performs practical multidimensional companding by means of a pre- (and post-) weighting filter on the time domain input signal. The optimal filter parameters are obtained from the sensitivity matrix for the perceptual model. The spectral perceptual model parameters are derived in a straightforward fashion from the signal model spectrum known to both, encoder and decoder. This process takes into account the spectral envelope and the spectral fine structure (e.g., pitch).

Bibliography

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

[2] T. M. N. Hoang, M. Oger, S. Ragot, and M. Antonini, “Embedded transform coding of audio signals by model-based bit plane coding,” in *Proc. ICASSP*, 2008.

2.2 Rate Distribution Between Model and Signal

In the flexible coding system envisioned by *FlexCode*, the rate is an adjustable parameter. This means that one must adjust the rates for both the quantization of the signal model parameters and for the quantizers that operate on the signal. This section shows that the optimal rate distribution follows a simple rule (at least asymptotically) that is easily implemented in a practical coding system.

2.2.1 Introduction

It is common practice to use a model in the encoding of audio signals. The model provides a characterization of the statistical dependencies that exist between signal samples. Usage of the model allows more efficient encoding of the signal. In audio and speech coding, it is common to use adaptive models that describe the short-term statistics of the signal (statistics that are meaningful within signal segments of 5 to 30 ms). When a model is used, two sets of data must be transmitted: on the one hand the *model parameters* and on the other hand the *signal coefficients* that specify the signal given the model (we do not consider the case of backward adaptation in this section).

Source coding is often formulated as a minimization of the bit rate required to transmit the signal at a given fidelity. If modeling is used, then it must be decided how to allocate the rate between model parameters and signal coefficients. The standard approach to rate allocation between model parameters and signal coefficients is based on experimental evidence. Optimization by experimentation is a laborious approach that is feasible only if it can be performed off-line. Thus, this approach to rate-allocation is natural only for coders that operate at a predetermined rate.

Communication networks and applications of audio coding in general are becoming increasing heterogeneous. To facilitate usage in various environments, audio coders must be able to operate at a range of rates. This implies that off-line experiment-based optimization of the rate-distribution between model parameters and signal coefficients is not desirable. This motivates the work in this section, which shows how a rate distribution between model parameters and signal coefficients can be derived theoretically based only on knowledge of certain signal properties. Our general approach towards rate distribution is based on that used in the context of the minimum description length (MDL) principle [1, 2]. The analytic relation for the bit rate allocation given in this section provides a step towards source coding algorithms that can adapt in real-time to the allowed rate set by an external control mechanism.

We provide practical results for the rate distribution for the particular case of autoregressive (AR) modeling, also often referred to as linear-predictive modeling. AR modeling has long been used in speech coding and is becoming more common for audio coding, particularly in the context of a low delay constraint, e.g., [3]. Our results show that the rates commonly used for the AR model in speech coding, e.g., [4, 5], can be explained based only on coding efficiency and a squared error criterion operating directly on the speech. Importantly, this means that perceptual aspects play only a minor role in the bit allocation for the model.

To determine the rate allocation between model parameters and signal coefficients, we must define relations between rate and distortion for these variables. To this purpose we use a model of quantization that is accurate only in the asymptotic limit of high rate. Thus, our results are guaranteed only for high rates. However, experimental evidence indicates that the resulting principles hold over a wide range of rates, e.g., [6].

The remainder of this section starts with a derivation of generic rate-allocation results in section 2.2.2. More detailed results are worked out for the AR case in section 2.2.3. To show that our results remain valid for practical rate allocations, we confirm the theory through experimental evidence in section 2.2.4.

2.2.2 Rate Allocation

We consider a stochastic process (signal) X_i. To encode the signal we divide the signal into coding blocks of k samples. For each block, the k samples are encoded independently from the other blocks, using a signal model. Thus, we try to optimize the encoding of random vectors X^k using models that are specified by a set of random model parameters Θ.

We now compute the number of bits required to encode a particular data sequence x^k, using a model θ, when the coder operates at a mean distortion D. A particular data model θ corresponds to an assumed probability density of the data $p_{X^k|\Theta}(\cdot|\theta)$. We assume that the cells are identical in shape and write the relation between mean distortion and cell volume V as

$$D = CV^{\frac{2}{k}}, \tag{2.1}$$

where C is the coefficient of quantization, a constant that depends only on the geometry of the quantization cell.

The data sequence (vector) x^k falls into a particular quantization cell, with index $i = i(x^k)$, $i \in \mathbb{N}$. The probability of this quantization index is, under the high-rate assumption,

$$p_I(i(x^k)) = Vp_{X^k|\Theta}(x^k|\theta), \tag{2.2}$$

where, in general, $V = V(i)$ is a function of i. The source-coding theorem effectively states that the codeword length required for a particular index i is $-\log(p_I(i))$. (To facilitate notational brevity, we use nats as unit of codeword length.) If we constrain the average rate of the indices, $H(I) = -\sum_i p_I(i)\log(p_I(i))$, then we obtain a so-called *constrained-entropy* quantizer. In such quantizers, V does not depend on x^k under the high-rate assumption. Thus, the codeword length required to encode a particular x^k with the coder that operates at mean distortion D is

$$L_{X^k|\Theta}(x^k|\theta) = -\log\bigl(p_{X^k|\Theta}(x^k|\theta)\bigl(\frac{D}{C}\bigr)^{-\frac{2}{k}}\bigr). \tag{2.3}$$

The codeword length of (2.3) can be minimized by selecting the best model parameters for the particular sequence. This simply the parameter vector that maximizes the probability density $p_{X^k|\Theta}(x^k|\theta)$. Assuming a uniform prior distribution for Θ^k, the optimal parameter set has maximum likelihood parameter for the sequence x^k. We write the resulting maximum likelihood model as

$$\hat{\theta}(x^k) = \operatorname*{argmax}_{\theta} p_{X^k|\Theta}(x^k|\theta). \tag{2.4}$$

While the maximum likelihood model minimizes the codeword length required for the signal vector x^k, the parameters $\hat{\theta}(x^k)$ of such a model can, in general, not be encoded at a finite rate.

To make the rate required for the parameters finite, we restrict the set of allowed parameter vectors to a countable set. The set $\{\bar{\theta}\}$ of admissable parameter vectors corresponds to the reconstruction vectors of a quantizer for the random parameter vector Θ. Let $p_{\bar{\Theta}}(\cdot)$ be the probability mass function for the countable set of allowed parameter vectors $\{\bar{\theta}\}$. Then the codeword length required to describe the model for $\bar{\theta}$ is

$$L_{\bar{\Theta}}(\bar{\theta}) = -\log(p_{\bar{\Theta}}(\bar{\theta})). \tag{2.5}$$

The rate required to encode x^k consists of the rate for the model and the rate for encoding the sequence x^k given the model:

$$\begin{aligned} L(x^k) =& L_{\bar{\Theta}}(\bar{\theta}(x^k)) + L_{X^k|\bar{\Theta}}(x^k|\bar{\theta}(x^k)) \\ =& -\log(p_{\bar{\Theta}}(\bar{\theta}(x^k))) - \log\big(p_{X^k|\bar{\Theta}}(x^k|\bar{\theta}(x^k))(\frac{D}{C})^{-\frac{2}{k}}\big) \\ =& \psi(\bar{\theta}(x^k), \hat{\theta}(x^k), x^k) - \log\big(p_{X^k|\hat{\Theta}}(x^k|\hat{\theta}(x^k))(\frac{D}{C})^{-\frac{2}{k}}\big) \end{aligned} \tag{2.6}$$

where

$$\psi(\bar{\theta}, \hat{\theta}, x^k) = -\log(p_{\bar{\Theta}}(\bar{\theta})) - \log\left(\frac{p_{X^k|\bar{\Theta}}(x^k|\bar{\theta})}{p_{X^k|\hat{\Theta}}(x^k|\hat{\theta})}\right) \tag{2.7}$$

is the *index of resolvability* [2]. The index of resolvability collects the terms of the overall rate that involve the quantized model parameters $\bar{\theta}$. The last term of (2.6) is the rate required to encode the signal vector x^k given the ideal (maximum-likelihood) model $\hat{\theta}$.

The index of resolvability (2.7) consists of two terms that have a clear interpretation. The first term represents the rate for the model parameters. The second term is the increase in the rate for the signal x^k resulting from using the non-optimal $\bar{\theta}$ instead of the optimal $\hat{\theta}$.

We are interested in the average performance for coding audio signal vectors X^k. Thus, we average (2.6) over the random vector X^k. Let $\mathrm{E}[\cdot]$ denote expectation over the ensemble of all audio signal vectors. The expected codeword length for X^k is then

$$\mathrm{E}[L(X^k)] = -\mathrm{E}[\psi(\bar{\theta}(X^k), \hat{\theta}(X^k), X^k)] - \mathrm{E}[\log(p_{X^k|\hat{\Theta}}(X^k|\hat{\theta}(X^k)))] + \frac{2}{k}\log(\frac{D}{C}). \tag{2.8}$$

The bit allocation for the model is determined by the mapping $\bar{\theta}(x^k)$. The optimal bit allocation for the model $\bar{\theta}$ is the result of a trade-off between the rate required for the model and the mean rate penalty resulting from using the quantized model if the same distortion must be attained. This trade-off involves only the mean index of resolvability (the first term of (2.8)). An important corrollary is that, under the assumptions of our derivation, *the optimal rate for the model parameters is unaffected by the mean signal distortion D*. The rate required for the model parameters depends on the structure of the model and the statistical properties of the data.

Table 2.1: Bit rates of the AMR-WB coder [7].

Rate	6.6	8.85	12.65	14.25	15.85	18.25	19.85	23.05
AR model parameters	36	46	46	46	46	46	46	46
pitch-model parameter	23	26	30	30	30	30	30	30
excitation	48	80	144	176	208	256	288	352

2.2.3 Application to Autoregressive Model

Autoregressive (AR) models are commonly used in speech coding and are becoming more common for audio applications. This model class forms a natural first application for the theory. Our goal is to describe the index of resolvability for the constrained-entropy case in terms that are easily computed and interpreted. We assume that the random signal vector X^k has a Gaussian multivariate distribution

$$p_{X^k|\Theta}(x^k|\theta) = \frac{1}{\sqrt{(2\pi)^k \det(R_\theta)}} \exp\left(-\frac{1}{2} x^{kT} R_\theta^{-1} x^k\right), \tag{2.9}$$

where R_θ is the model covariance matrix for X^k corresponding to the AR model with parameters θ. To find the matrix R_θ, we model the random vector X^k as a segment of a stationary AR process. This implies that the matrix is Toeplitz, symmetric, and has as first column the autocovariance function of a signal generated with the AR model. The autocovariance function is the inverse discrete-time Fourier transform of the transfer function of AR filter, which means the first column of R_θ is

$$R_\theta(n,0) = \frac{1}{2\pi}\int_0^{2\pi} \frac{\sigma^2}{|A(\mathrm{e}^{j\omega})|^2} \mathrm{e}^{jn\omega} d\omega, \tag{2.10}$$

where σ^2 is the excitation signal variance (gain) and $A(z) = 1 + a_1 z^{-1} \cdots + a_m z^{-m}$ for an m'th order AR model. The model parameters are then $\theta = [\sigma^2, a_1, a_2, \cdots a_m]$.

As a first step towards obtaining the mean index of resolvability we determine an expression for $\log(p_{X^k|\Theta}(x^k|\theta))$. We note that the factor multiplying the exponential in (2.9) is, asymptotically with increasing k,

$$\begin{aligned} -\frac{1}{2}\log((2\pi)^k \det(R_\theta)) &\approx -\frac{k}{2}\log(2\pi) - \frac{k}{2}\frac{1}{2\pi}\int_0^{2\pi} \log(R_\theta(\mathrm{e}^{j\omega}))d\omega \\ &= -\frac{k}{2}\log(2\pi) - \frac{k}{2}\log(\sigma^2) \end{aligned} \tag{2.11}$$

where we used Szegö's theorem and that $R_\theta(z) = \sigma^2/|A(z)|^2$ and that since $A(z)$ is a monic minimum-phase polynomial $\int_0^{2\pi} \log(|A(\mathrm{e}^{j\omega})|^2)d\omega = 0$. We then consider the argument of the exponential in (2.9). Let $R_{x^k}(\mathrm{e}^{j\omega})$ be the Fourier transform of the auto-covariance function of the segment x^k. Then, again asymptotically with increasing k, we have

$$\frac{1}{2k} x^{kT} R_\theta^{-1} x^k \approx \frac{1}{4\pi}\int_0^{2\pi} \frac{R_{x^k}(\mathrm{e}^{j\omega})}{R_\theta(\mathrm{e}^{j\omega})} d\omega. \tag{2.12}$$

Thus, based on (2.11) and (2.12) we can approximate (2.9) as

$$\log(p_{X^k|\Theta}(x^k|\theta)) \approx -\frac{k}{2}\log(2\pi\sigma^2) - \frac{k}{4\pi}\int_0^{2\pi} \frac{R_{x^k}(\mathrm{e}^{j\omega})}{R_\theta(\mathrm{e}^{j\omega})} d\omega. \tag{2.13}$$

Ignoring scaling effects, the index of resolvability (2.7) is

$$\begin{aligned}\psi(\bar{\theta},\hat{\theta},x^k) \approx & -\log(p_{\bar{\Theta}}(\bar{\theta}))+ \\ & \frac{k}{4\pi}\int_0^{2\pi}\left(\frac{R_{x^k}(\mathrm{e}^{j\omega})}{R_{\bar{\theta}}(\mathrm{e}^{j\omega})}-\frac{R_{x^k}(\mathrm{e}^{j\omega})}{R_{\hat{\theta}}(\mathrm{e}^{j\omega})}\right)d\omega. \\ = & -\log(p_{\bar{\Theta}}(\bar{\theta}))+ \\ & \frac{k}{4\pi}\int_0^{2\pi}\frac{R_{x^k}(\mathrm{e}^{j\omega})}{R_{\hat{\theta}}(\mathrm{e}^{j\omega})}\left(\frac{R_{\hat{\theta}}(\mathrm{e}^{j\omega})}{R_{\bar{\theta}}(\mathrm{e}^{j\omega})}-1\right)d\omega. \end{aligned} \tag{2.14}$$

Assuming that the effect of model quantization on the power spectrum is small, we use the expansion $u = 1 + \log(u) + \frac{1}{2}\log(u)^2 \cdots$:

$$\psi(\bar{\theta},\hat{\theta},x^k) \approx -\log(p_{\bar{\Theta}}(\bar{\theta}))+\frac{k}{4\pi}\int_0^{2\pi}\frac{R_{x^k}(\mathrm{e}^{j\omega})}{R_{\hat{\theta}}(\mathrm{e}^{j\omega})}\left(\log(\frac{R_{\hat{\theta}}(\mathrm{e}^{j\omega})}{R_{\bar{\theta}}(\mathrm{e}^{j\omega})})+\frac{1}{2}\log(\frac{R_{\hat{\theta}}(\mathrm{e}^{j\omega})}{R_{\bar{\theta}}(\mathrm{e}^{j\omega})})^2\right)d\omega. \tag{2.15}$$

The ratios $\frac{R_{x^k}(\mathrm{e}^{j\omega})}{R_{\bar{\theta}}(\mathrm{e}^{j\omega})}$ and $\frac{R_{\hat{\theta}}(\mathrm{e}^{j\omega})}{R_{\bar{\theta}}(\mathrm{e}^{j\omega})}$ represent the effect of modeling and the effect of quantization respectively. The modeling ratio averages to unity for maximum likelyhood gain. It is reasonable to assume that the effects of modeling and quantization are independent and we can approximate

$$\psi(\bar{\theta},\hat{\theta},x^k) \approx -\log(p_{\bar{\Theta}}(\bar{\theta}))+\frac{k}{4\pi}\int_0^{2\pi}\left(\log(\frac{R_{\hat{\theta}}(\mathrm{e}^{j\omega})}{R_{\bar{\theta}}(\mathrm{e}^{j\omega})})+\frac{1}{2}\log(\frac{R_{\hat{\theta}}(\mathrm{e}^{j\omega})}{R_{\bar{\theta}}(\mathrm{e}^{j\omega})})^2\right)d\omega. \tag{2.16}$$

Neglecting the effect of gain quantization, (2.16) can be written as

$$\psi(\bar{\theta},\hat{\theta},x^k) \approx -\log(p_{\bar{\Theta}}(\bar{\theta}))+\frac{k}{4\pi}\int_0^{2\pi}\frac{1}{2}\log(\frac{R_{\hat{\theta}}(\mathrm{e}^{j\omega})}{R_{\bar{\theta}}(\mathrm{e}^{j\omega})})^2 d\omega, \tag{2.17}$$

where the second term is the well-known *mean square log-spectral distortion* measure, which is commonly used to evaluate the performance of prediction coefficient quantizers, e.g., [8, 4].

By averaging the index of resolvability (2.17) over the ensemble of signal vectors, we obtain the equation that governs the rate allocation for the model parameters for the AR model

$$\mathrm{E}[\psi(\bar{\theta},\hat{\theta},x^k)] = R(\bar{\Theta}) + \frac{k}{4}D(\bar{\Theta},\hat{\Theta}) \tag{2.18}$$

with

$$R(\bar{\Theta}) = -\mathrm{E}[\log(p_{\bar{\Theta}}(\bar{\Theta}))] \tag{2.19}$$

and

$$D(\bar{\Theta},\hat{\Theta}) = \mathrm{E}\left[\frac{1}{2\pi}\int_0^{2\pi}\log(\frac{R_{\hat{\Theta}}(\mathrm{e}^{j\omega})}{R_{\bar{\Theta}}(\mathrm{e}^{j\omega})})^2 d\omega\right]. \tag{2.20}$$

We recognize in (2.18) a Lagrangian that minimizes the average rate for the parameter vector quantizer under a constraint on the mean log spectral squared error. The Lagrange multiplier is $\frac{k}{4}$. We can replace the mapping $x^k \to \bar{\theta}$ by the simpler two-stage mapping $x^k \to \hat{\theta} \to \bar{\theta}$.

The model parameters describe a manifold in the log power spectral domain with dimensionality $d \leq |\Theta|$. With increasing rate, the optimal constrained-entropy quantizer is asymptotically uniform on this manifold in the log-power-spectral domain. The performance of a high-rate constrained-entropy quantizer using the square error and lying on the manifold scales with rate as

$$D(\bar{\Theta}, \hat{\Theta}) = dC\mathrm{e}^{-\frac{2}{d}(R(\bar{\Theta}) - h(\hat{\Theta}))}, \tag{2.21}$$

where C is the coefficient of quantization, $h(\hat{\Theta})$ is the differential entropy of $\hat{\Theta}$ *as measured in the log power spectral domain*. The objective is to find the model rate $R(\bar{\Theta})$ that minimizes

$$\mathrm{E}[\psi(\bar{\theta}, \hat{\theta}, x^k)] = R(\bar{\Theta}) + \frac{k}{4}dC\mathrm{e}^{-\frac{2}{d}(R(\bar{\Theta}) - h(\hat{\Theta}))}, \tag{2.22}$$

which is solved by the optimal model rate allocation

$$R(\bar{\Theta}) = h(\hat{\Theta}) + \frac{d}{2}\log\left(\frac{k}{2}C\right). \tag{2.23}$$

2.2.4 Results and Verification

The principles introduced in this section are of a general nature. We verify the principles with applications to the coding of speech. The motivation for the selection of the speech signal is that the AR model is commonly used in this context, which means reasonable model choices are well understood. More-over, existing results for standardized coders can provide a first indication of the principles derived herein.

Corroborative Earlier Results

The present section discusses the distribution of the bit rate for entropy-constrained coding, which is common in audio coding. The results are essentially identical for the case constrained-resolution coding. Practical results for constrained-resolution display the correct behavior. Table 2.1 shows bit allocations used in the adaptive-multirate wide-band (AMR-WB) speech coder [7]. It is seen that the bit allocation for the model parameters is independent of the rate of the codec, except at low rates. In contrast, the bit allocation for the excitation (the signal) increases rapidly with the overall rate. These bit allocations confirm our theoretical findings.

2.2.5 Experimental Verification

The verification was performed for tenth-order AR modeling on 8 kHz sampled speech. We used the TIMIT database [9].

We first estimated the differential entropy and the manifold dimensionality of the random parameter vector θ, using the methods described in [10]. The dimensionality of the manifold was found as $d = 7.9$ and the differential entropy was 8.5 bits. From (2.23) it then follows that the optimal rate for the model is 19.0 bits for scalar quantization and 17.2 bits for vector quantization. The rates correspond to a root mean square log spectral distortion of 1.29 dB, close to the 1 dB commonly used on heuristic grounds [4].

The second step of our verification work is to confirm that the mean of the summation of model rate and signal rate is minimized using the measured rates for

$$\mathrm{E}[L(x^k)] = \mathrm{E}[L_{\bar{\Theta}}(\bar{\theta}(x^k))] + \mathrm{E}[L_{X^k|\bar{\Theta}}(x^k|\bar{\theta}(x^k))]. \tag{2.24}$$

To confirm this, we measured the average rate that a coder operating on speech requires for the speech signal, for a given speech-signal distortion and with varying quantization accuracy for the model parameters. To this purpose, we extracted 10000 randomly located speech blocks of 160 samples (20 ms) from the 1680 utterance evaluation part of TIMIT. For each block we performed linear-predictive analysis (using a Hann window) to obtain a set of AR model parameters. To approximate optimal parameter quantization on the data manifold in the log spectral distortion domain, we converted the parameters to the line-spectral frequencies (LSFs) and computed the (diagonal) sensitivy matrix of the LSFs [11]. We then performed scalar quantization of the scaled LSFs. We used these parameters to estimate the bit allocation required for scalar quantization of the 40-dimensional speech vector x^k located in the center of the 160 sample block. The vector was first decorrelated using a model-based Karhunen-Loève transform, then scalar quantized. The rate was estimated using numerical integration of the probability density function over the cell. We multiplied by four to get the rate for a stationary block of 160 samples, approximating a common speech coder scenario.

In Fig. 2.1 we show the outcome of the experiments. It provides the overall rate as a function of the rate allocated to the model for a range of distortions for the signal. It is seen that at overall coding rates of about 2 to 5 bits per sample, the overall coding rate is minimized when the model rate is about 20 bits. It is seen that this rate is independent of the overall coding rate. As expected from heuristic reasoning, the actual optimal model rate decreases when the signal distortion is high and the overall rate falls below the range of rates where the theory is valid.

2.2.6 Conclusions

In this section we considered the coding of signal segments with a model. We concluded that with increasing rate the rate allocation for the model becomes a constant and is independent of the overall rate allocation. An existing speech coding standard and our own experimental confirm the theoretical results. We conclude that our method can be used to predict the optimal model coding rate. For the AR model, our approach leads to the commonly used squared log spectral distortion measure for the prediction parameters. More-over, we can conclude that the required accuracy of the AR parameters is not a direct function of perceptual effects. Our results mean that audio (as well as other source) coders that adapt in real-time to changing network conditions can use a fixed quantizer for the signal model parameters.

Bibliography

[1] J. Rissanen, "Modeling by the shortest data description," *Automatica*, vol. 14, pp. 465–471, 1978.

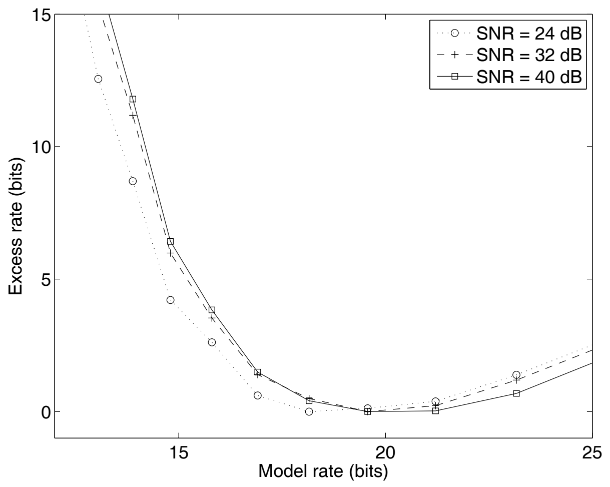


Figure 2.1: *The excess in overall rate (over its minimum value) for a block of 160 samples as a function of the model rate for different signal distortion levels. The data are for scalar quantization. The corresponding minimum rates are 340, 530, and 736 bits per block. The theory predicts a model rate of 19 bits.*

[2] A. Barron and T. M. Cover, "Minimum complexity density estimation," *IEEE Trans. Inform. Theory*, vol. 37, no. 4, pp. 1034–1054, 1991.

[3] U. Krämer, G. Schuller, S. Wabnik, J. Klier, and J. Hirschfeld, "Ultra low delay audio coding with constant bit rate," *Proc. 117th Audio Engineering Society Convention*, pp. 22 – 33, 2004.

[4] K. K. Paliwal and B. S. Atal, "Efficient vector quantization of LPC parameters at 24 bits/frame," *IEEE Trans. Speech Audio Process.*, vol. 1, no. 1, pp. 3–14, 1993.

[5] P. Hedelin, "Single stage spectral quantization at 20 bits," in *Proc. Int. Conf. Acoust. Speech Signal Process.*, San Francisco, 1992, pp. 57–60.

[6] R. Vafin and W. B. Kleijn, "Entropy-constrained polar quantization and its application to audio coding," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 2, pp. 220–232, Mar. 2005.

[7] B. Bessette, R. Salami, R. Lefebvre, M. Jelinek, J. Rotola-Pukkila, J. Vainio, and H. Mikkola, "The adaptive multirate wideband speech codec (AMR-WB)," *IEEE Trans. Speech Audio*, vol. 6, no. 8, pp. 620–636, 2002.

[8] A. Gray and J. Markel, "Quantization and bit allocation in speech processing," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-24, pp. 459–473, 1976.

[9] DARPA-TIMIT, "Acoustic-phonetic continuous speech corpus," *NIST Speech Disc 1-1.1*, 1990.

[10] M. Nilsson and W. B. Kleijn, “On the estimation of differential entropy from data located on embedded manifolds,” *IEEE Transactions on Information Theory*, vol. 53, no. 7, pp. 2330–2341, Jul. 2007.

[11] W. R. Gardner and B. D. Rao, “Theoretical analysis of the high-rate vector quantization of LPC parameters,” *IEEE Trans. Speech Audio Process.*, vol. 3, no. 5, pp. 367–381, 1995.

2.3 Model-based bit plane coding

2.3.1 Introduction

Model-based coding has already shown promising results for the quantization of linear predictive coding (LPC) coefficients [1], autoregressive waveform coding of speech [2], audio transform coding [3] and entropy-constrained vector quantization [4]. In this section we show that a similar statistical modeling approach can be used in bit plane coding.

Bit plane coding is an entropy coding method wherein an integer sequence is decomposed in bit planes from most significant bits (MSB) to least significant bits (LSB). It is used for instance in audio and image coding, see MPEG-4 BSAC [5], proprietary audio coders [6, 7], and JPEG2000 [8]. The bitstream associated with bit plane coding is scalable (or embedded) in the sense that it allows simple rate adaptation by truncation and decoding of partially received coded data. This flexibility is particularly attractive for communications over heterogenous networks [9].

Model-based bit plane coding consists in estimating the symbol probability in bit planes based on a pdf model for the underlying source. Specifically we consider here the problem of coding of a source $\mathbf{X} = [x_1 \; x_2 \; \ldots \; x_N]$. After uniform scalar quantization with stepsize q, we obtain an integer sequence $\mathbf{Y} = [y_1 \; y_2 \; \ldots \; y_N]$, with $y_i = [x_i/q]$, where $[.]$ is the rounding to the nearest integer. Bit plane coding is applied to $\mathbf{Y}$.

In the following the probability density function (pdf) of $\mathbf{X}$ is approximated by a generalized Gaussian model (also called super-Gaussian). Other pdf models may be applied.

2.3.2 Bit plane decomposition of an integer sequence

The sign-magnitude representation of an integer sequence $\mathbf{Y} = [y_1 \; y_2 \; \ldots \; y_N]$ of length N is defined as:

$$y_i = (-1)^{sign(y_i)} |y_i| \tag{2.25}$$

where $sign(y_i)$ is the sign bit associated with y_i, with the convention

$$sign(y_i) = \begin{cases} 1 & \text{if } y_i < 0 \\ 0 & \text{if } y_i \geq 0 \end{cases}, \tag{2.26}$$

and each magnitude (absolute value) $|y_i|$ is written in natural binary format

$$|y_i| = \sum_{k=0}^{K-1} bit_k(y_i).2^k \tag{2.27}$$

with $bit_k(y_i)$ being the k^{th} bit of the natural binary decomposition of $|y_i|$ and K the number of bit planes needed to decompose the complete sequence $\mathbf{Y}$. The number K is given by:

$$K = \max\left(\lceil \log_2 \max_{i=1,\ldots,N} |y_i| \rceil, 1 \right) \tag{2.28}$$

where $\lceil \cdot \rceil$ is the rounding to the nearest integer towards $+\infty$ and $\log_2(0) = -\infty$.

Note that for the sake of simplicity $bit_k(y_i)$ corresponds here to the natural binary representation of integers. Other representations such as Gray coding or reflected Gray coding, one's complement, two's complement, may be used.

Using the above sign-magnitude representation, the k-th bit plane of $\mathbf{Y}$, $k = 0, \ldots, K-1$, is defined as:

$$\mathbf{P}_k(\mathbf{Y}) = [bit_k(y_1)\ bit_k(y_2)\ \ldots\ bit_k(y_N)], \tag{2.29}$$

while an additional bit plane can be defined to gather all sign bits:

$$\mathbf{S}(\mathbf{Y}) = [sign(y_1)\ sign(y_2)\ \ldots\ sign(y_N)]. \tag{2.30}$$

The bit planes $\mathbf{P}_{K-1}(\mathbf{Y})$ and $\mathbf{P}_0(\mathbf{Y})$ correspond to the most significant bits (MSB) and least significant bits (LSB), respectively.

2.3.3 Bit plane coding principle

Bit plane coding usually proceeds from MSB to LSB, from more important to less important data. If the underlying error criteria is the mean square error criterion bit planes $\mathbf{P}_k(\mathbf{Y})$ are coded sequentially from $k = K-1$ to $k = 0$ in accordance with the *reverse waterfilling principle* [10]. However, if perceptual criteria are incoporated, the parsing of bit planes may be non-sequential.

Note that magnitude and sign coding are usually intertwined:

- The sign bit $sign(y_i)$, $i = 1, \ldots, N$, is transmitted only if $|y_i| \neq 0$.
- To allow decoding partially received coded data, $sign(y_i)$ is transmitted as soon as one of the bits $\{bit_k(y_i)\}_{k=0,\ldots,K-1}$ is equal to one and has just been coded.

A simplified algorithm for bit plane coding is given below:

```
for $k = K-1$ to $0$ do {Loop from MSB to LSB}
  for $n = 1$ to $N$ do
    <code> $bit_k(y_i)$
    if $bit_k(y_i) = 1$ AND $sign(y_i)$ is not yet coded then
      <code> $sign(y_i)$
    end if
  end for
end for
```

Any coding method `<code>` may be used to code bits $bit_k(y_i)$ and $sign(y_i)$. Adaptive and/or context-based arithmetic coding [11, 12] is often used for this purpose.

2.3.4 Bit plane coding based on a pdf model of the source

Preliminary: generalized Gaussian pdf model

The pdf of a zero-mean generalized Gaussian random variable x of standard deviation σ is given by:

$$g_{\sigma,\alpha}(x) = \frac{A(\alpha)}{\sigma} e^{-|B(\alpha)x/\sigma|^{\alpha}}, \tag{2.31}$$

where α is a shape parameter describing the decay rate of the density function,

$$A(\alpha) = \frac{\alpha B(\alpha)}{2\Gamma(1/\alpha)} \qquad \text{and} \qquad B(\alpha) = \sqrt{\frac{\Gamma(3/\alpha)}{\Gamma(1/\alpha)}}, \tag{2.32}$$

with

$$\Gamma(\alpha) = \int_0^\infty e^{-t} t^{\alpha+1} \, dt. \tag{2.33}$$

Note that the special cases $\alpha = 1$ and 2 correspond to the Laplacian and Gaussian distributions, respectively.

In order to estimate the shape parameter α many procedures can be used, see for instance [13].

Model-based estimation of probabilities for entropy coding of bit planes

The integer sequence $\mathbf{Y} = [y_1 \; y_2 \; \ldots \; y_N]$ is obtained from uniform scalar quantization of $\mathbf{X} = [x_1 \; x_2 \; \ldots \; x_N]$ with stepsize q:

$$y_i = \left[\frac{x_i}{q}\right], \tag{2.34}$$

where [.] is the rounding to the nearest integer.

Assuming X is a realization of a zero-mean generalized Gaussian random variable of variance σ, the probability of y_i is given by [14]:

$$p(y_i) = \int_{qy_i - q/2}^{qy_i + q/2} g_{\sigma,\alpha}(x) dx \tag{2.35}$$

where q is the stepsize and $g_{\sigma,\alpha}(x)$ is the p.d.f. defined in Eq. 2.31. Without loss of generality, the source $\mathbf{X}$ can be normalized to a unit variance so that the stepsize can be normalized to q/σ and σ is normalized to 1.

To be more specific, given the above model for the pdf of $\mathbf{X}$ and the resulting probability model for $p(|y_i|)$, the probability of having the $bit_k(y_i)$ equal to zero can be estimated as follows:

$$p(bit_k(y_i) = 0) = p(|y_i|) \times \delta_{bit_k(y_i),0} \tag{2.36}$$

with

$$\delta_{x,y} = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases}$$

Note that Eq. 2.36 is valid only for the natural binary decomposition.

It is also assumed that number of bit planes K is sent as side information to the decoder, so that the decoder can estimate properly $p(y_i)$. Based on this assumption, the a priori information $|y_i| \leq M$ can be used, so the probability of having zero in bit plane $\mathbf{P}_k(\mathbf{Y})$ is given by [14]:

$$p(b_k = 0 | \; |y_i| \leq M) = \frac{p(bit_k(y_i) = 0, |y_i| \leq M)}{p(|y_i| \leq M)} \tag{2.37}$$

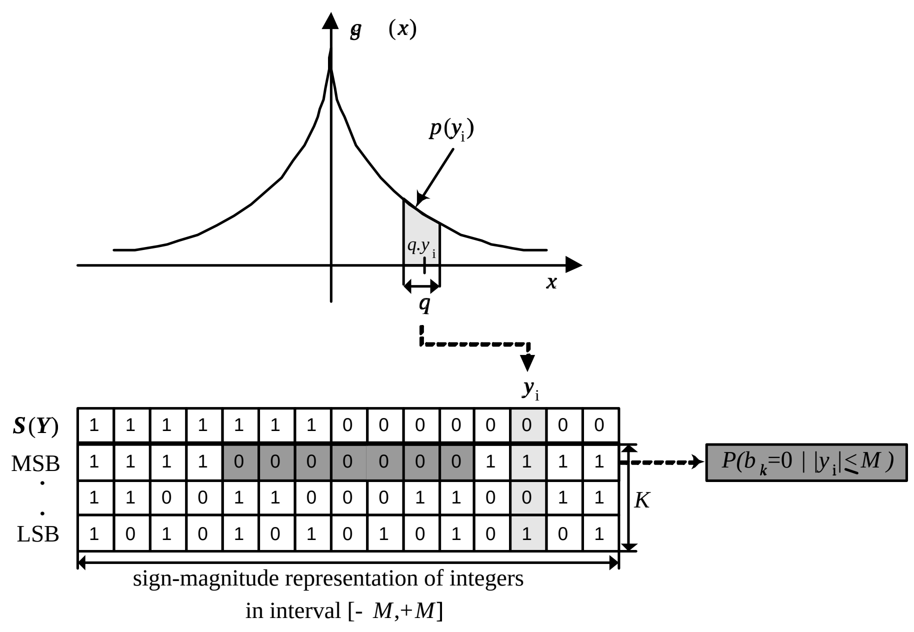

g (x)
p(y_i)
q.y_i
x
q
y_i
S(Y) 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
MSB 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1
1 1 0 0 1 1 0 0 0 1 1 0 0 1 1
LSB 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
K
P(b_k=0 | |y_i|≤M)
sign-magnitude representation of integers in interval [- M,+M]

Figure 2.2: *Principle of model-based probability estimation for bit plane coding.*

where $b_k = bit_k(y_i)$ for any $1 \leq i \leq N$ and

$$p(|y_i| \leq M) = \sum_{y_i=-M}^{M} p(y_i|\ |y_i| \leq M) = \int_{-q(M+1/2)}^{q(M+1/2)} g_{\sigma,\alpha}(x)dx \qquad (2.38)$$

Therefore:

$$p(b_k = 0 \mid |y_i| \leq M) = \frac{\displaystyle\sum_{y_i=-M}^{M} p(y_i) \times \delta_{bit_k(y_i),0}}{\displaystyle\sum_{y_i=-M}^{M} p(y_i)} \qquad (2.39)$$

The probability of having one in bit plane $\mathbf{P}_k(\mathbf{Y})$ can be obtained from the relationship:

$$p(b_k = 0 \mid |y_i| \leq M) + p(b_k = 1 \mid |y_i| \leq M) = 1 \qquad (2.40)$$

The estimation of $p(b_k = 0 \mid |y_i| \leq M)$ is illustrated in Fig. 2.2.

The above model-based probability estimation relies on several assumptions:

- The source $\mathbf{X}$ is i.i.d and has a generalized Gaussian p.d.f.
- The number K of bit planes is transmitted as side information to the decoder
- Bit planes $\mathbf{P}_k(\mathbf{Y})$ are coded independently from each other

This yields the following algorithm:

```
Require: Transmit the number of bit planes $K$ as side information and compute
  $M = 2^K - 1$
  Estimate shape parameter $\alpha$ given $[x_1\ x_2\ \ldots\ x_N]$
  Calculate probabilities $p(y_i)$, $i = 0, \cdots, M$ using Eq. 2.35
  for $k = K - 1$ to 0 do {Loop from MSB to LSB}
    Calculate $p(b_k = 0 \mid |y_i| \leq M)$ according to Eq. 2.39
    Calculate $p(b_k = 1 \mid |y_i| \leq M)$ using Eq. 2.40
    for $n = 1$ to $N$ do
      <entropy code> $bit_k(y_i)$ based on $p(b_k \mid |y_i| \leq M)$
      if $bit_k(y_i) = 1$ AND $sign(y_i)$ is not yet coded then
        <code> $sign(y_i)$
      end if
    end for
  end for
```

Generalization to context-based bit plane coding The coding of bit planes $\mathbf{P}_k(\mathbf{Y})$ with $k < K - 1$ following the MSB may use the available knowledge of bit planes, $\mathbf{P}_{K-1}(\mathbf{Y}) \ldots \mathbf{P}_{k+1}(\mathbf{Y})$, that have already been (de)coded.

In this generalization, the MSB is coded with model-based probability estimation as discussed previously. Exploiting the assumption of an i.i.d. generalized Gaussian model, the usable conditional information (context) for $bit_k(y_i)$ is limited to the knowledge of $bit_{k+1}(y_i), \cdots, bit_{K-1}(y_i)$. For more details, see [14].

Results

WB-PESQ [15] was used to evaluate objectively the quality of an example transform audio coder using bit plane coding. Only clean speech sentences sampled at 16 kHz are used to compute the average WB-PESQ (MOS-LQO) scores at various bit rates (from 16 to 40 kbit/s). Note that for bit plane coding the encoder operated at maximal bit rate (40 kbit/s), while the decoder bitrate was equal to or lower than the encoder bitrate.

Figure 2.3 shows the WB-PESQ scores for transform audio coding using bit plane coding (basic or model-based) compared to two reference coders (stack-run coding [3] and ITU-T G.722.1). It appears that the use of model-based probability estimation improves the performance of bit plane coding.

Note that bit plane coding requires virtually no storage.

2.3.5 Conclusions

A complete model-based method for transform coding of audio signals can be found in [14], where the input signal is mapped in perceptual domain by linear-predictive weighting filtering followed by modified discrete cosine transform (MDCT). In this framework model-based bit plane coding brings an improvement compared with basic bit plane coding based on adaptive arithmetic coding, and allows to close the gap between bit plane coding and non-embedded (monolithic) entropy coding [14].

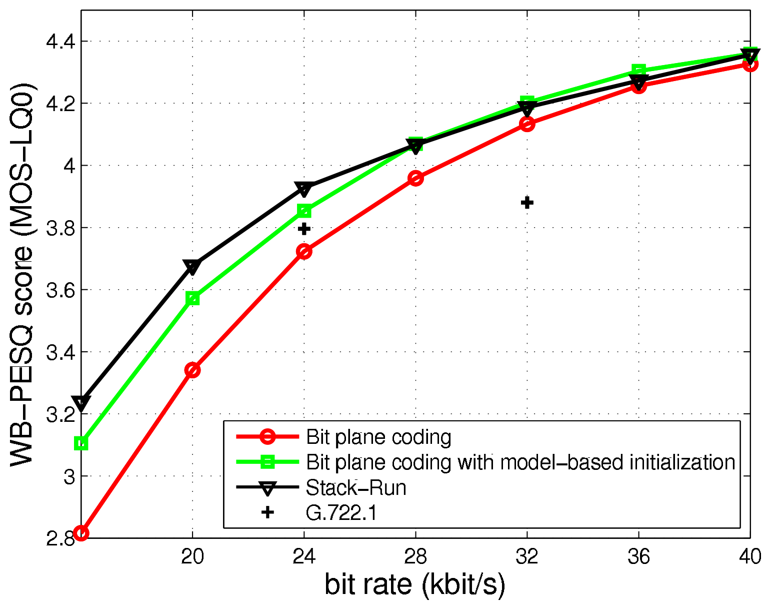


Figure 2.3: Average WB-PESQ score of transform audio coding with (basic or model-based) bit plane coding – Comparison with stack-run coding and ITU-T G.722.1.

Bibliography

[1] A. D. Subramaniam and B. D. Rao, "PDF optimized parametric vector quantization of speech line spectral frequencies," *IEEE Trans. Speech and Audio Proc.*, vol. 11, no. 2, pp. 130–142, Mar 2003.

[2] J. Samuelsson, "Waveform quantization of speech using Gaussian mixture models," in *Proc.ICASSP*, vol. 1, 2004, pp. 165–168.

[3] M. Oger, S. Ragot, and M. Antonini, "Transform audio coding with arithmetic-coded scalar quantization and model-based bit allocation," in *Proc. ICASSP*, vol. 4, 2007, pp. 545–548.

[4] D. Zhao, J. Samuelsson, and M. Nilsson, "GMM-based entropy-constrained vector quantization," in *Proc. ICASSP*, vol. 4, 2007, pp. 1097–1100.

[5] S. H. Park and al., "Multi-layer bit-sliced bitrate scalable audio coding," in *AES 103rd Convention*, Aug 1997.

[6] C. Dunn, "Efficient audio coding with fine-grain scalability," Sep 2001.

[7] J. Li, "Embedded audio coding (eac) with implicit auditory masking," *ACM Multimedia 2002*, Dec 2002.

[8] D. S. Taubman and M. W. Marcellin, *JPEG2000: Image Compression Fundamentals, Standards and Practice*. Springer, 2001.

[9] B. Geiser, S. Ragot, and H. Taddei, "Embedded Speech Coding: From G.711 to G.729.1," in *Advances in Digital Speech Transmission*, R. Martin, U. Heute, and C. Antweiler, Eds. Wiley, 2008.

[10] T. M. Cover and J. A. Thomas, *Information Theory*. Wiley, 1991.

[11] G. G. Langdon, "An introduction to arithmetic coding," *IBM I. Res. Dev. 28*, pp. 135–149, Mar 1984.

[12] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *ACM Communications*, Jun 1987.

[13] S. G. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 11, pp. 674–693, Jul. 1989.

[14] T. M. N. Hoang, M. Oger, S. Ragot, and M. Antonini, "Embedded transform coding of audio signals by model-based bit plane coding," in *Proc. ICASSP*, 2008.

[15] ITU-T Rec. P.862.2, "Wideband extension to Recommendation P.862 for the assessment of wideband telephone networks and speech codecs," Nov 2005.

2.4 Multi-Mode Excitation Reconstruction Scheme for Improved Coding of Audio

2.4.1 Introduction

In the *FlexCode* concept the source coder should adapt to continuously varying bit-budget and source statistics. With the decrease of a bitrate (and/or increase of signal bandwidth) it is more efficient to encode only important parts of the signal and generate the remaining parts, e.g., [1, 2, 3, 4]. The conventional solution is to quantize low-frequency regions and reconstruct high-frequency regions of the spectra. All bits are allocated to the frequency components below pre-defined frequency index, and at the decoder the remaining (unquantized) components are reconstructed from the quantized ones. This approach is static, and the quantization and reconstruction regions are pre-determined, but might not be optimal under changes of bitrate or input source characteristics.

A more advanced solution, suitable for variable bit rates, is to dynamically detect the regions to be quantized and regions to be reconstructed. In [5] the frequency bands are ordered accordingly to the perceptual importance of the corresponding part of spectrum envelope. The regions with higher energy are quantized, and used to reconstruct the remaining part of the signal.This approach guarantees that the most important regions are quantized, but has no control over the level of artifacts introduced with signal reconstruction. As the fine structure continuity is not preserved, this approach can potentially create artifacts related to pitch discontinuities.

In general, none of the approaches presented above searches for the optimal point between quantization errors in the low-frequency and reconstruction artifacts in the high-frequency spectral regions. In this section we present an improved scheme for audio coding. This scheme in an optimal way selects the range of frequencies to be quantized and the range of frequencies to be reconstructed from the quantized ones.

2.4.2 The Framework

First we present a coding framework, in which the proposed multi-mode residual quantization is used. Typically in audio coding, short segments of signal are first transformed to a new domain, then encoded and transmitted, and finally the signal is inversely transformed and reconstructed at the decoder. For the simplicity of presentation we constrained our transform coding scheme to be modified discrete cosine transform (MDCT) [6, 7], and will refer to transform coefficients as MDCT coefficients.

To increase the compression efficiency, the transform coefficients are first flattened by the spectrum envelope, and then quantized. These flattened-quantized coefficients, together with a quantized version of the spectral envelope are transmitted to the decoder, and the signal is reconstructed. In the following we shall refer to the flattened coefficients as residual MDCT vector. There are few different ways to define the spectral envelope. In [8] the spectrum envelope is calculated through frequency response of autoregressive (AR) coefficients. In [9] and [10] the spectrum envelope is calculated through grouping MDCT coefficients and calculating the mean energy in each group. These groups can be of uniform length [10], or the length can increase towards high-frequency [9]. Our study on the multi-mode excitation reconstruction is in the framework, similar to [8]. Block-scheme of the encoder is presented in Fig. 2.4. The windowed signal block s is analyzed to extract the AR coefficients, and then MDCT transformed. The MDCT coefficients, denoted d are flattened by the AR spectrum envelope P_{AR} to form residual signal d_n. The overall gain G scales the variance of the residual signal to unity. This residual signal is encoded by means of companding, followed by uniform scalar quantizers (SQ). The autoregressive coefficients are independently encoded by means of Gaussian mixture model vector quantizer (GMM-VQ) [11]. The technology, presented in this section, is about efficient quantization/reconstruction of residual signal d_n.

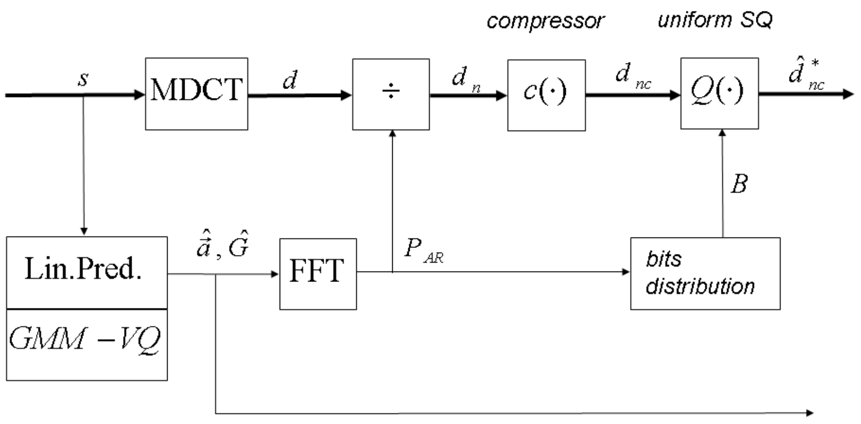


Figure 2.4: MDCT coding scheme with AR spectral envelope, and SQ of the residual.

2.4.3 Multi-Mode Scheme Excitation Reconstruction

In the proposed scheme the MDCT residual is encoded in different codec modes. The winning mode is selected as the one that minimizes a pre-defined criterion. In all modes continuous parts of the spectrum (all starting at lowest-frequency component, but of different length) are quantized, regardless of the energy of the spectrum envelope. The modes that quantize a short part (and therefore reconstruct a large part) of the vector can achieve high-quality at low-frequencies at a cost of increased number of artifacts in high-frequency regions. The modes that quantize a large part (and therefore reconstruct a short part) of the vector avoid artifacts in high-frequencies, but cannot achieve sufficient quality at low-frequencies. Depending on the bitrate and signal self-similarity, different modes can give the best trade-off between low- and high-frequency distortions. Note that in practice the quantized regions are not used directly to reconstruct the remaining regions, but reconstruction is done through a virtual codebook technique, described in the next section.

One particular four-mode implementation of this scheme is presented in Fig. 2.5. When the codec is in mode A) the entire residual vector is quantized, thus the available bits are spread over the entire dimension. In mode D) all bits are spent for quantization of the lower-quarter of the vector, and the remaining frequencies are reconstructed. In general, with decreasing the bit-budget the preference of the modes goes from A to D, as human perception is sensitive to fine-structure errors in low-frequency regions. If enough bits are available, and the low-frequency regions are quantized with sufficient resolution, the preferred modes will be A and B. With increasing the self-similarity of the signal, the preference goes from A to D, as the process of reconstruction introduces less artifacts. By searching through all modes the systems balances between high resolution quantization of low-frequency regions, and introducing artifacts in high-frequency regions.

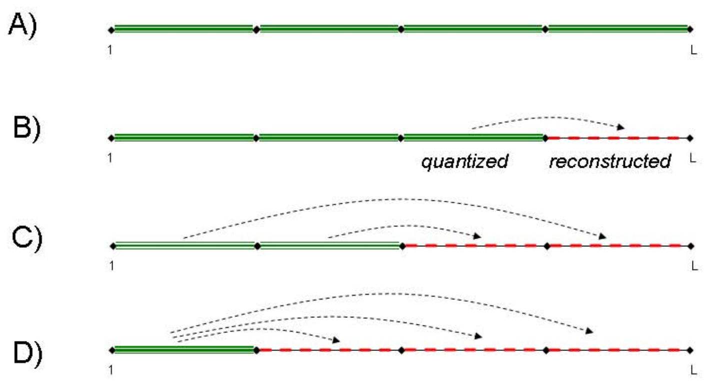


Figure 2.5: Example of 4 competing modes for residual quantization. The quantized regions are in green-solid, while the reconstructed regions are in red-dashed. The length of the residual vector is denoted by L.

2.4.4 Implementation

Critical point in the algorithm is the decision criterion that selects the optimal mode for residual quantization. The mode selection is defined as a minimization problem:

$$\mathbf{y}^* = \arg\min_k \; D(\mathbf{x}, \mathbf{y}^k), \tag{2.41}$$

where k is the index over all modes, **x** is residual reference MDCT vector, and **y** - quantized residual MDCT vector from the reconstruction codebook. The distortion $D(\cdot)$ is calculated as follows: As a first step the sign is removed from both vectors and they are smoothed

$$X_n = (1 - \alpha_n)|x_n| + \alpha_n \; X_{n-1}, \tag{2.42}$$

$$Y_n = (1 - \alpha_n)|y_n| + \alpha_n \; Y_{n-1}. \tag{2.43}$$

Here the index $n \in \{1 \ldots L\}$ is over the entire vector dimension, and $|\cdot|$ denotes the absolute value. The weight is not constant, but increases towards high-frequencies

$$\alpha_n = \left(\frac{n}{L}\right)^6. \tag{2.44}$$

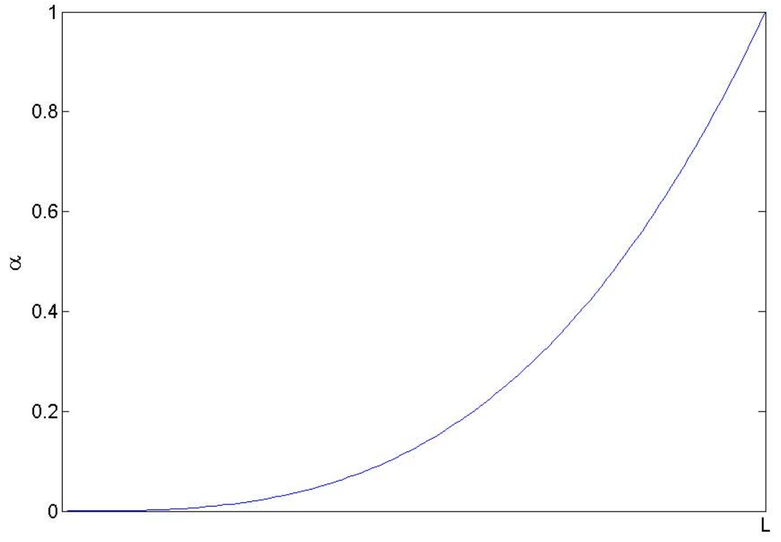


Figure 2.6: Frequency dependent weighting factor.

This weighting scheme makes the contribution of the fine structure information from the high-frequencies less important, while preserving sufficient resolution in the low-frequencies. After the pre-processing step a sign-difference is calculated over the residual vectors:

$$D = \frac{1}{L} \sum_{n=1}^{L} (Y_n - X_n)^{\beta}, \tag{2.45}$$

$$\beta = \begin{cases} 4 & \text{if} \quad (Y_n - X_n) \geq 0 \\ 2 & \text{if} \quad (Y_n - X_n) < 0, \end{cases}$$

Such a sign-difference adds heavier penalty for "new" spectral components, and less for "missing" spectral components. This is motivated by the fact that the human brain can partly interpolate the missing spectral components, thus making certain distortion less objectionable.

As mentioned earlier the quantized residual coefficients are not used directly to reconstruct the remaining high-frequency regions, but they are first processed to create a reconstruction codebook. Such a processing consists of two steps: 1) compression of coefficients with 10% largest absolute values (see Fig. 2.7), and 2) overall energy attenuation of 30%. Such a procedure removes outliers that can

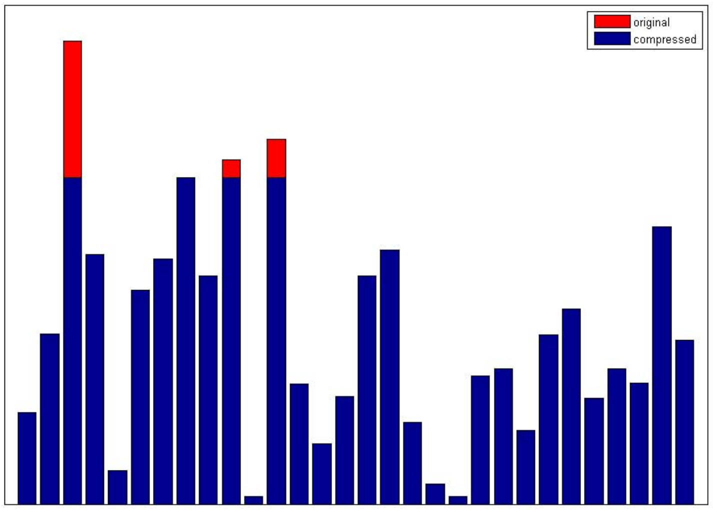


Figure 2.7: The compression procedure attenuates the coefficient with 10% largest absolute values, to the highest level, among the remaining 90% lowest energy coefficients. Only the absolute values of residual coefficients are presented in the figure.

cause annoying perceptual artifacts, but leads to energy loss in the high-frequency regions of the reconstructed signal. Tilt correction postfilters can be used to compensate for such effect. One possibility is to use postfilter of the form:

$$H_t(z) = 1 - \mu\, z^{-1}, \tag{2.46}$$

that is controlled by the parameter μ with a typical value $\mu = 0.4$. Different choice of tilt correction postfilter is:

$$H_t(z) = \alpha\, z^{-1} - \beta + \alpha\, z^{+1}, \tag{2.47}$$

with suitable values for the parameter $\alpha = 0.0825$ and $\beta = 0.5825$. Naturally the choice of tilt correction, and value of control parameters, depend on the decoder design, and particularly on other postfiltering schemes, used at the decoder.

2.4.5 Advantages of the Multi-Mode Scheme

The multi-mode residual quantization offers an improved quality in transform audio coding schemes. The improvement comes through selection of the optimal mode, for the current bitrate and input source characteristics. In the following all simulations are performed with the coding scheme from Fig. 2.4, and wideband sources. Tables 2.2 and 2.3 provide statistics of the model occurrence (% of frames for which a particular mode is selected) with bitrate and source type. One can easily notice that the optimal size of quantized (and respectively reconstructed) region is not only a function of the bitrate, but also varies with the source.

Table 2.2: Mode occurrence with bitrate and source type - Speech, German male

	A	B	C	D
12 kb/s	4.8 %	14.6 %	11.3 %	69.4 %
22 kb/s	16.7 %	7.9 %	26.3 %	49.2 %
32 kb/s	15.2 %	16.7 %	51.8 %	16.4 %

Table 2.3: Mode occurrence with bitrate and source type - Music, Castanets

	A	B	C	D
12 kb/s	3.4 %	4.2 %	6.3 %	86.1 %
22 kb/s	3.6 %	24.5 %	35.7 %	36.2 %
32 kb/s	3.2 %	55.7 %	36.9 %	4.2 %

Results from objective evaluation with the state-of-the-art and ITU-T standard for quality assessment of wideband speech [12] are presented in Table 2.4. Results from the proposed dynamic scheme are under *Multi-mode scheme*, while the last two columns of the table represent two implementations of the same codec, but with fixed size of quantization/recostruction regions. It is easy to see that the proposed scheme gives a superior performance, which is also confirmed in informal listening tests.

Table 2.4: Overall quality improvement of the multi-mode scheme in comparison with the conventional solutions - WB-PESQ

	Multi-mode scheme	Quantize entire spectrum	Quantize lower-half
12 kb/s	3.528	3.387	3.399
22 kb/s	3.819	3.592	3.739
32 kb/s	3.876	3.775	3.864

Bibliography

[1] M. Dietz, L. Liljeryd, K. Kjorling, and O. Kunz, "Spectral band replication, a novel approach in audio coding," in *Preprint, Audio Eng. Soc 112th Convention*, 2002.

[2] 3GPP TS 26.404, "Enhanced aacPlus encoder SBR part," 2007.

[3] J. Makinen, B. Bessette, S. Bruhn, P. Ojala, R. Salami, and A. Taleb, "AMR-WB+: a new audio coding standard for 3rd generation mobile audio services,"

in *Proc. IEEE Int. Conf. Acous., Speech, Signal Processing*, 2005, pp. 109–112.

[4] 3GPP TS 26.290, "Extended Adaptive Multi-Rate - Wideband (AMR-WB+) Codec; Transcoding Functions," 2007.

[5] B. Kövesi, D. Massaloux, and A. Sollaud, "A scalable speech and audio coding scheme with continuous bitrate flexibility," in *Proc. IEEE Int. Conf. Acous., Speech, Signal Processing*, 2004, pp. 273–276.

[6] J. Princen, A. Johnson, and A. Bradley, "Subband/transform coding using filter bank designs based on time domain aliasing cancellation," in *Proc. IEEE Int. Conf. Acous., Speech, Signal Processing*, 1987, pp. 2161–2164.

[7] J. Princen and A. Bradley, "Analysis/synthesis filter bank design based on time-domain aliasing cancellation," *IEEE Trans. Acoustics, Speech, Signal Processing*, vol. 34, pp. 1153–1161, 1986.

[8] N. Iwakami, T. Moriya, and S. Miki, "High-quality audio-coding at less than 64 kbit/s by using transform-domain weighted interleave vector quantization (TwinVQ)," in *Proc. IEEE Int. Conf. Acous., Speech, Signal Processing*, 1995, pp. 3095–3098.

[9] 3GPP TS 26.403, "Encoder specification AAC part," 2004.

[10] ITU-T Rec. G.722.1, "Low-complexity coding at 24 and 32 kbit/s for hands-free operation in systems with low frame loss," 2005.

[11] A. Subramaniam and B. Rao, "PDF optimized parametric vector quantization of speech line spectral frequencies," *IEEE Trans. Speech, Audio Processing*, vol. 11, pp. 130–142, 2003.

[12] ITU-T Rec. P.862.2, "Wideband extension to Recommendation P.862 for the assessment of wideband telephone networks and speech codecs," 2005.

2.5 Lattice Quantization

Lattice quantization has attracted interest of the lossless compression community during the last years due to its reduced memory requirements for storage and low complexity encoding and decoding algorithms.

Within the *FlexCode*, lattice quantization techniques will be employed both for the signal model and for the transform coefficients in rate constrained as well as entropy constrained framework.

2.5.1 Introduction

Most of the lattice based coding methods rely on fixed rate coding or on a semi-variable rate coding where the vector to be quantized is split in several sub-blocks for which the rate is variable, but the overall bit rate for the global vector is fixed [1]. There exist also variable rate encoding of lattice codevectors. Most of these methods rely on the grouping of codevectors on classes such as leader classes or shells [2] or apply directly entropy coding methods to the lattice codevector components [3].

In order to preserve the flexibility brought by the Gaussian mixture models [3], [4], the use of the lattice quantization within the FlexCode had to be restricted to generalized rectangular lattice truncations, which make straightforward the combination of the mixture models and the companded lattice quantization. An additional advantage of this type of truncation consists of the very low complexity algorithms related to it.

In the subsequent parts of this section we define first the generalized rectangular truncation. The use of a lattice as a quantizer is intrinsically linked to the existence of an indexing algorithm of the lattice points. The presentation of adequate new indexing algorithms make the subject of the second part. The primary tools being established, the report will focus further on their use in rate and entropy constrained coding algorithms by presenting newly introduced coding techniques based on lattice rotation and new ways of lattice entropy encoding.

2.5.2 Rectangular Lattice Truncation

A rectangular truncation of norm K of the lattice Λ is defined as [5]:

$$\Lambda_K = \Big\{ \mathbf{x} = (x_1, x_2, \ldots, x_n) \in \Lambda \ \, | N(\mathbf{x}) \leq K \Big\} \tag{2.48}$$

where the n-dimensional real vector $(x_1, x_2, \ldots, x_n)$ belongs to the lattice Λ.

A generalization of the above formula is so-called generalized rectangular truncation that has different maximum absolute norms, $\{K_i\}_{i=1:n}$ along different dimensions.

$$\Lambda_{K_i} = \Big\{ \mathbf{x} = (x_1, x_2, \ldots, x_n) \in \Lambda \ \, ||x_i| \leq K_i \Big\}. \tag{2.49}$$

We have tested the use of the generalized rectangular D_n lattice truncation on data consisting of 16-dimensional line spectral frequencies (LSF) vectors of audio signals. Gaussian mixture models are used and each input vector is presumed to belong to a given mixture component, companded accordingly and quantized in the rectangular lattice truncation. To each mixture component a bit allocation per

Method	SD[dB]
Uniform scalar quantization	0.88
Lattice quantization with D_4	0.66
Lattice quantization with D_8	0.63
Lattice quantization with D_{16}	0.70

Table 2.5: *Spectral distortion values for uniform scalar and lattice quantization.*

component is assigned through a reversed water filling algorithm. The number of levels derived from the bit allocation for each dimension is proportional to the size of the truncation of the lattice on the corresponding dimension. There are 16 mixture components and the one having the smallest distortion is selected and its index encoded.

The results in terms of spectral distortion (SD) are presented in Table 2.5. Different values for the lattice dimension are allowed by grouping the input vector components into 4, 8, or 16 dimensional sub-vectors. Also the result obtained with uniform scalar quantization is presented.

The use of the lattice quantizer improves the spectral distortion by 20% to 28%, depending on the lattice dimension. The 16 dimensional lattice does not give the best results, as could have been expected by increasing the dimension of the vector quantizer, because in 16 dimensions the lattice D_n is not one of the most efficient in terms of the second order normalized moment [6].

2.5.3 Indexing of Rectangular Lattice Truncations

We will consider in the following the Z_n lattice. The same maximum norm along all the dimensions will be assumed in the following algorithms, but generalization to the different norm case is straightforward. The exterior shell of the truncation is formed by the points:

$$\overline{\Lambda_K} = \Big\{\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \Lambda \ |N(\mathbf{x}) = K\Big\}. \tag{2.50}$$

This section presents two indexing algorithms for rectangular lattice truncations, for lattices that can be represented as union of co-sets of Z_n. These algorithms are discussed in more detail in [7].

The lattice points within the truncation can be indexed considering all the truncation at once, or considering it shell by shell. The former case corresponds to the first indexing algorithm, while the latter corresponds to the second one.

Base Representation Indexing (IA1)

This is one of the most natural indexing algorithms for points situated in a rectangular truncation of an integer lattice Z_n and it relies on the observation that on each of the n dimensions there are $(2K + 1)$ possible values, where K is the maximum absolute norm of the truncation.

The indexing algorithm using base representation is presented next.

Coding algorithm

Input: n-dimensional integer vector $(x_1, x_2, \ldots, x_n)$ having maximum absolute norm equal to K

Output: integer index, $I^{(1)}$

$$I^{(1)} = \sum_{i=1}^{n} (2K+1)^{(i-1)} (x_i + K) \tag{2.51}$$

Decoding algorithm:

Input: integer index, $I^{(1)}$

Output: n-dimensional integer vector having maximum absolute norm equal to K

Represent the index in the base $(2K + 1)$.

Product Code Indexing (IA2)

In [8] the use of a product code type index in which at least the sign bits are separated has shown to have good error resilience performance. Using a similar approach, we divide the information contained in the codevector into several entities appropriate to the rectangular truncation:

- The number of the significant (non zero) components (A);
- The number of maximum valued components (in absolute value) (B);
- The position of the maximum valued components (C);
- The values of the significant non-maximum components (D);
- The position of the significant non maximum values (E);
- The signs of the significant components (F).

The borders between the bits corresponding to different entities that form the index are not strict, except for the bits corresponding to the signs. The strict border of the sign bits is due to the fact that they are situated at an extreme of the index and the cardinality of the set describing all the sign combinations is a power of two. The indexing corresponding to the bits ordering $\boxed{\text{A / B / C / D / E} \mid \text{F}}$ is presented in the next subsection. The delimiter "|" represents a strict border.

Coding algorithm: **Input**: n-dimensional integer vector having maximum absolute norm equal to K

Output: integer index, $I^{(2)}$

1. Count the number of significant (non-zero) components in the vector, S;
2. Calculate the offset
$$O_0^{(2)}(S) = \sum_{i=n}^{S+1} T_i = \sum_{i=n}^{S+1} 2^i \binom{n}{i} \left(K^i - (K-1)^i\right);$$
3. Count the number of maximum components, in absolute value, i.e. having the absolute value equal to K, M;

4. Calculate

$$O_1^{(2)}(S,M) = 2^S \binom{n}{S} \sum_{i=S}^{M+1} \binom{S}{i} (K-1)^{(S-i)};$$

5. Calculate the position index, relative to the whole vector length, of the maximum components, I_k;

6. Calculate the order index of non-maximum significant components, I_{nk};

7. Calculate the position index relative to the length of the vector without maximum components, of the non-maximum significant components, I_{pos_nk};

8. Calculate the index of sign bits, I_B;

9. Calculate the index

$$\begin{aligned} I^{(2)} &= O_0^{(2)}(S) + O_1^{(2)}(S,M) + \\ &+ I_k 2^S \binom{n-M}{S-M} (K-1)^{S-M} + \\ &+ I_{nk} 2^S \binom{n-M}{S-M} + I_{pos\_nk} 2^S + I_B. \end{aligned} \tag{2.52}$$

The index I_B is obtained then as:

$$I_B = \sum_{i=1}^{S} b_i 2^{i-1},$$

where b_i is 0 if the i-th significant component is negative and 1 otherwise.

Decoding algorithm

Input: integer index, $I^{(2)}$

Output: n-dimensional integer vector having maximum absolute norm equal to K

1. Calculate the number of significant components, S, such that $O_0^{(2)}(S-1) > I^{(2)} \geq O_1^{(2)}(S,M)$;

2. Update the index $I^{(2)} = I^{(2)} - O_0^{(2)}(S)$;

3. Calculate the number of maximum components, in absolute value, M, such that $O_1^{(2)}(S, M-1) > I^{(2)} > O_1^{(2)}(S,M)$;

4. Update the index $I^{(2)} = I^{(2)} - O_1^{(2)}(S,M)$;

5. Calculate

$$I_k = \left\lfloor I^{(2)} / \left(2^S \binom{n-M}{S-M} (K-1)^{S-M} \right) \right\rfloor$$

and

$$I^{(2)} = I^{(2)} \bmod \left(2^S \binom{n-M}{S-M} (K-1)^{S-M} \right);$$

6. Decode I_k to obtain the position of the maximum valued components within the whole vector;

7. Calculate
$$I_{nk} = \left\lfloor I^{(2)} / \left(2^S \binom{n-M}{S-M} \right) \right\rfloor$$
and
$$I^{(2)} = I^{(2)} \bmod \left(2^S \binom{n-M}{S-M} \right);$$

8. Decode I_{nk} to find the value of the non maximum values (less than K in absolute value);

9. Calculate $I_{pos\_nk} = \lfloor I^{(2)}/2^S \rfloor$ and
$$I_B = I^{(2)} \bmod 2^S;$$

10. Decode I_{pos_nk} to find the position of the non maximum values within the whole vector without the maximum values;

11. Decode I_B to the sign bits.

The encoding and decoding of the positions is done using the enumeration algorithm for the binomial coefficients presented in [2]. In this approach, the input of the enumeration algorithm may be for example the vector $z = (z_1, z_2, \ldots, z_n) \in \{0,1\}^n$ such that there are exactly M unitary components in the vector, at positions corresponding to the positions of maximum valued components in the codevector. Additionally, a position vector $p = (p_0, \ldots, p_{M-1}) \in \{0, \ldots, n-1\}^M$ is created, which specifies the exact location of each of the maximum valued component. Since there are $\binom{n}{M}$ such z vectors, they can be enumerated like binomial coefficients following the algorithm given by the next equations:

$$\begin{aligned} I_{pos}(n, M, p) &= \sum_{i=1}^{p_0} \binom{n-i}{M-1} + \\ &\quad I_{pos}(n - p_0 - 1, M - 1, \\ &\quad (p_1, \ldots, p_{M-1}) - p_0 - 1) \end{aligned} \tag{2.53}$$

and $I_{pos}(n', 1, [i]) = i, 0 \leq n' \leq n$. I_{pos} can then be used as the desired position index I_k or I_{pos_nk}.

To recover the position vector p from an index I_{pos}, the following algorithm may be used:

1. $i = 0$

2. while $(M > 0)\{$

   - find j such that $\sum_{i=1}^{j} \binom{n-i}{M-1} \leq I_{pos} < \sum_{i=1}^{j+1} \binom{n-i}{M-1}$
   - $p_i = j$
   - $I_{pos} = I_{pos} - \sum_{i=1}^{j} \binom{n-i}{M-1}$
   - $n = n - j - 1$

```
    • $M = M - 1$
    • $i = i + 1$
}
```

The vector z may then be recovered by inserting the value 1 at the positions indicated in the vector p and the value 0 at all the other positions.

The encoding of the non maximum valued components is done using the base representation algorithm IA1.

2.5.4 Entropy Constrained Lattice Quantization

The advantages of the entropy constrained coding over the resolution constrained coding are due to a better adaptation to the local statistics of the source. These, together with the low complexity of the quantization even for high dimensional spaces, when using lattice based tools, make the lattice entropy constrained tool a very attractive approach.

Entropy constrained methods using lattice quantizers have been previously presented in the literature. Most of these methods rely on the grouping of codevectors in classes such as leader classes or shells [2] or apply directly entropy coding methods to the lattice codevector components [3]. However the former method becomes less practical when the number of classes increases (with the increase of the bit rate and for some of the truncation shapes), while the latter is from the start less efficient than a direct entropy coding of the lattice vectors indexes, but obviously less complex.

We present in this section a new indexing method for lattice vectors that makes use of the product code indexing method presented in the previous section. The proposed method is exemplified on rectangular truncation of lattices, where the number of leader classes is relatively high and the shape of the truncation is successfully used in conjunction with companding.

The different informational entities extracted from the vector, can be also interpreted as means of classifying the vectors into different sets. The existence of several entities implies the division of all the vectors into sets, sub-sets and so forth. If the index corresponding to all or part of the set (subset) types are entropy encoded, an entropy code can be obtained for the initial lattice vector.

For instance, given the 4 dimensional vector (2 -3 0 -1), having maximum norm equal to 3, it has three significant components (A), one maximum valued component (B), index 1 for the position of the maximum valued component (C) and index 1 for the position of the non maximum valued components (E) (see [7]). There is at least one significant value and four at the most, therefore there are four possible symbols for the number of significant components, which can be entropy encoded. Furthermore, the number of maximum valued components can be entropy encoded, as well as the position indexes of the maximum valued components and so on.

The practical interest of the proposed method becomes evident if we consider, for instance, that there are 16 dimensional lattice points to be entropy encoded, at a bit-rate of 2 bits per sample. The probabilities for $2^{2 \cdot 16}$ symbols should be stored, whereas if one lattice point index is split into several different types of data the variability of symbols considered in the entropy coding is reduced, assuming that there are different entropy encoders for the different types of information.

Bit Rate Calculation

Consider the n-dimensional vectors from the Z_n rectangular truncation of norm K. Any vector from this set can be represented using N_0 bits, where

$$N_0 = \lceil \log_2((2K+1)^n) \rceil. \tag{2.54}$$

If the entity corresponding to the number of significant values is entropy encoded on n_1 bits, the current vector from the set of vectors can be represented on N_1 bits instead of N_0, where

$$\begin{aligned} N_1 &= n_1 + \Bigg\lceil \log_2 \Bigg(2^S \Bigg(\binom{n}{1} \binom{n-1}{S-1} (K-1)^{S-1} + \\ &\qquad \binom{n}{2} \binom{n-2}{S-2} (K-1)^{S-2} + ... + \binom{n}{S} \Bigg) \Bigg) \Bigg\rceil, \end{aligned} \tag{2.55}$$

S is the number of significant components.

If the number of significant components is entropy encoded on n_1 bits, the number of maximum valued components is encoded on n_2 and the index of positions for the maximum valued components is encoded on n_3 bits then the current vector from the set of vectors can be represented on N_3 bits, where

$$\begin{aligned} N_3 &= n_1 + n_2 + n_3 + \\ &\qquad \Bigg\lceil \log_2 \Bigg(2^S \binom{n-M}{S-M} (K-1)^{S-M} \Bigg) \Bigg\rceil \end{aligned} \tag{2.56}$$

and M is the number of maximum valued components whose position is already coded on n_3 bits.

The presented method can provide bit-rate savings up to 30% from the bit-rate allocated to the spectral coefficients within an audio coding scenario, with respect to the fixed rate lattice quantization. In addition to the improved compression efficiency, the proposed method enables the use of lattice entropy encoding in higher dimensions.

2.5.5 Lattice Rotation for Low Bit-rate Quantization

At moderate bit-rates (below 2 bits per sample), for non-symmetric sources, or data having vanishing directions, lattice border effects are significant and the rate distortion curves depend on the orientation of the lattice truncation.

Therefore, if the lattice is rotated such that the denser direction corresponds to the denser direction in the data implying that the rate distortion performance can be improved.

Figure 2.8 presents rate-distortion curves for Z_2 and A_2 lattices for independent Gaussian sources with same standard deviation on both directions, for different lattice rotation angles. As expected, there is just one curve for each of the lattices, and the performance of the lattice A_2 is better. Entropy constrained coding is considered and H is the experimental entropy of the codewords.

When the sources have no longer a symmetric probability density function, the rate-distortion curves are sensitive to the angle of rotation of the lattice like proved in the figures 2.9 and 2.10. In addition, by comparing the two figures, it can be

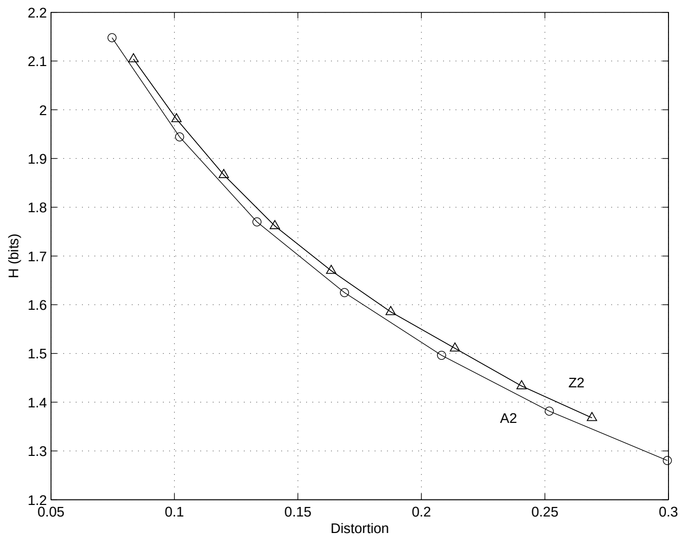


Figure 2.8: *Rate distortion functions for Z_2 and A_2 lattices for non-correlated Gaussian sources with same standard deviation on both directions.*

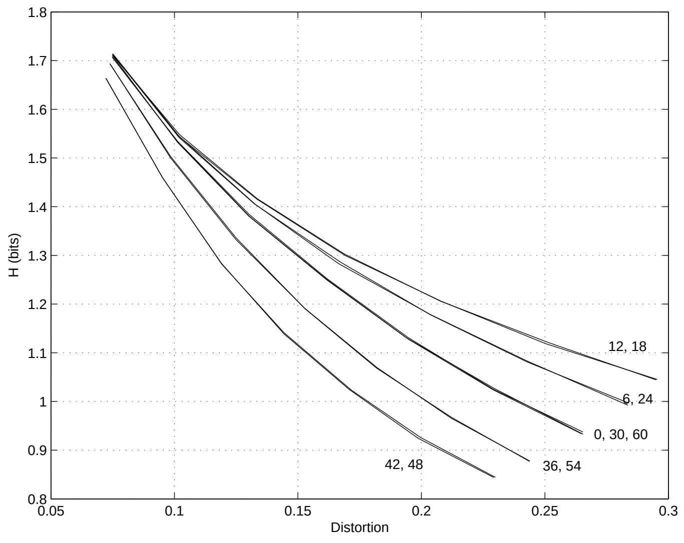


Figure 2.9: *Rate distortion functions for rotated A_2 lattices for Gaussian correlated sources with correlation coefficient 0.9.*

observed that the optimally rotated Z_2 lattice gives better performance than the optimally rotated A_2 lattice.

The lattice rotation in conjunction with the proposed entropy constrained method will be used for the coding of the transform coefficients within the *FlexCode*.

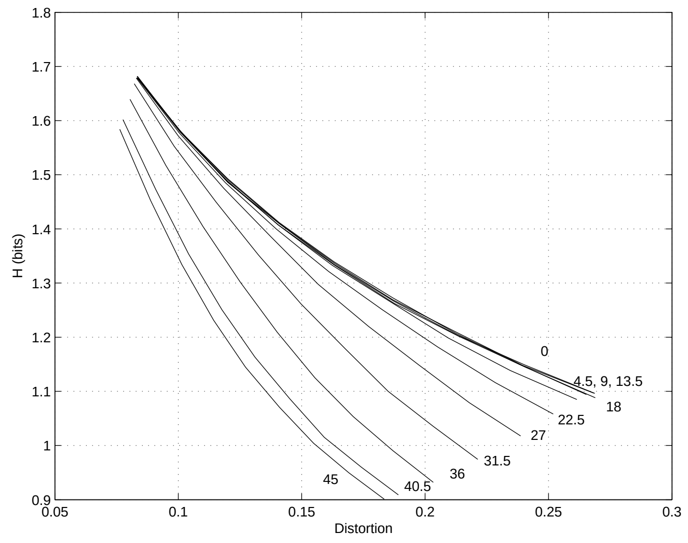


Figure 2.10: *Rate distortion functions for rotated Z_2 lattices for Gaussian correlated sources with correlation coefficient 0.9.*

2.5.6 Conclusions

Lattice quantizers provide practical non-complex ways of vector quantization that are known to have better performance than the scalar quantizers. In addition, the proposed lattice rotation techniques enable efficient use of the structured quantizers for moderate and low bit-rates, providing for the *FlexCode* codec the coverage of a larger bit-rate domain. The proposed entropy coding methods enable the use of lattice quantizers for higher dimensions, in the context of entropy constrained coding.

Bibliography

[1] S. Ragot, B. Bessette, and R. Lefebvre, “Low-complexity multi-rate lattice vector quantization with application to wideband TCX speech coding at 32 kbit/s,” in *Proc. of ICASSP 2004*, vol. 1, Montreal, Canada, May, 17-21 2004.

[2] A. Vasilache and I. Tabus, “Indexing and entropy coding of lattice codevectors,” in *Proceedings of ICASSP 2001*, Salt Lake City, Utah, USA, May, 7-11 2001.

[3] D. Zhao, J. Samuelsson, and M. Nilsson, “GMM-based entropy-constrained vector quantization,” *Proc.ICASSP*, vol. 4, pp. 1097–1100, 2007.

[4] A. Ozerov and W. B. Kleijn, “Flexible quantization of audio and speech based on the autoregressive model,” in *IEEE Asilomar Conference on Signals, Systems, and Computers (CSSC), Pacific Grove, CA*, 2008.

[5] A. Vasilache, “On vector quantization with regular geometric structure,” Ph.D. dissertation, Tampere University of Technology, 2003.

[6] J. H. Conway and N. J. A. Sloane, *Sphere packings, lattices and groups*, 3rd ed. Springer, 1999.

[7] A. Vasilache, "Indexing of lattice codevectors applied to error resilient audio coding," in *Proceedings of the AES* 30th *International Conference*, Saariselkä, Finland, March, 15-17 2007.

[8] A. C. Hung, E. K. Tsern, and T. H. Meng, "Error-resilient pyramid vector quantization for image compression," *IEEE Trans. on Image Processing*, vol. 7, no. 10, October 1998.

2.6 Perceptual Modeling

Multidimensional perceptual companding offers the flexibility required in *Flex-Code* to integrate perceptual distortion. The reason lies in the effective decoupling of perceptual aspects from quantization and coding. This section shows how *Flex-Code* achieves perceptually optimal coding by using perceptual companding followed by squared error optimal quantization. Furthermore, we provide a solution that exploits the redundancy between the signal model and the perceptual model.

2.6.1 Introduction

Present perceptual coding schemes do not offer flexibility in the way perceptual distortion measures are integrated into the coder. For instance, AAC [1] uses an iterative, heuristic Analysis-by-Synthesis loop guided by the perceptual model to obtain weighting coefficients (scale factors) for the MDCT-transformed input signal. This input signal is then quantized and Huffman coded. The inflexibility lies in the latter: these Huffman tables are carefully trained and tuned and depend not only on the data used for tuning, but also on the particular perceptual model. AAC has no means to exploit knowledge about a signal model other than through the trained Huffman tables again (or to some extend the window-switching process). Such a highly tuned system is sensitive to changes in each part it consists of and should be carefully retrained after such changes. Furthermore, despite having proven successful, the procedures used in AAC are not known to be optimal in an information theoretic sense.

In *FlexCode* we use multidimensional companding [2] to separate perceptual aspects from quantization and coding. That is, minimizing the squared error on the compressed signal in quantization will lead to minimal perceptual error (see 2.6.3). At each stage, components are (ex-)changeable without affecting the other stage. This allows for tractable equations yielding optimal solutions in each processing stage in *FlexCode*.

The current state of the perceptual model needs to be known at the decoder. A typical approach is to send information about the perceptual model to the decoder (i.e., many transform audio coders send a masking curve as side information). Since the perceptual model is derived from the signal, one can expect redundancy between the signal model and the perceptual model. In *FlexCode* we avoid sending this information twice by deriving the perceptual model from the signal model (see 2.6.2), an approach typically known only from coding schemes with very basic perceptual models (such as [3] or [4]). For reasons of computational complexity, we currently only use the spectral perceptual model proposed in [5].

The remainder of this section is organized as follows. We introduce the spectral perceptual model and explain how it is derived from the signal model in 2.6.2. In 2.6.3 we explain how to obtain the optimal perceptual compander for the spectral perceptual model based on its Sensitivity Matrix description. Section 2.6.3 introduces the practical implementation of such a compander by means of a perceptual weighting filter on the time-domain signal.

2.6.2 The Spectral Perceptual Model Obtained from the Signal Model

A distortion measure based on a perceptual model aims at describing the human auditory sensation evoked when comparing a corrupted signal to the original. Here we describe the spectral perceptual model proposed by van de Par et al. in [5] which was chosen for the *FlexCode* coder due to its low computational complexity and demonstrate how it can be obtained from the signal model.

We denote the original signal by $\mathbf{x}$ (masker), and the corrupted signal by $\hat{\mathbf{x}}$, that is

$$\hat{\mathbf{x}} = \mathbf{x} + \mathbf{s}, \tag{2.57}$$

with the distortion signal $\mathbf{s}$ (maskee). We denote the Fourier transform of $\mathbf{x}$, $\hat{\mathbf{x}}$ and $\mathbf{s}$ by $\mathbf{X}$, $\hat{\mathbf{X}}$ and $\mathbf{S}$, with entries indexed by frequency f, i.e., X_f, $\hat{X}_f$ and S_f.

The van de Par model [5] defines the detectability as

$$d_{Par}(\mathbf{X}, \mathbf{S}) = C_s \hat{L} \sum_i \frac{\frac{1}{N}\sum_f |h_f|^2 |g_f^i|^2 |S_f|^2}{\frac{1}{N}\sum_f |h_f|^2 |g_f^i|^2 |X_f|^2 + C_a}, \tag{2.58}$$

where h_f is the transfer function of the outer-middle ear filter, g_f^i is that of the i^{th} gamma-tone filter and $\hat{L}$ is the effective duration. Two constants C_a and C_s are chosen such that the threshold of detectability corresponds to $d_{Par}(\mathbf{X}, \mathbf{S}) = 1$.

The perceptual model can be extracted from the signal at the encoder and transmitted to the decoder. Since there is redundancy between the signal model and the perceptual model, transmitting both is not efficient. In *FlexCode*, we calculate the van de Par model based on the signal model, so that no extra information for the perceptual model needs to be transmitted. Simply, we use the transfer function of the AR signal model as an approximation of the Fourier transform of the input signal, i.e.,

$$X_f = \frac{\sigma}{A(e^{jf})} \times \frac{\sigma_p}{1 - \beta e^{-jfd}}, \tag{2.59}$$

where $\sigma/A(z)$ is the AR model and $\sigma_p/(1 - \beta z^{-d})$ is the pitch model.

2.6.3 The Optimal Perceptual Compander for the Spectral Perceptual Model

In this section we outline the derivation of the optimal perceptual compander for the In this section we outline the derivation of the Sensitivity Matrix for the distortion measure in (2.58). Let $\mathbf{x} = [x_1, \ldots, x_N]^T$ be a vector of random source samples and $\hat{\mathbf{x}} = Q(\mathbf{x})$ a distorted source vector. Let D be the expected value of the distortion measure of interest $d(\mathbf{x}, \hat{\mathbf{x}})$,

$$D = E[d(\mathbf{x}, \hat{\mathbf{x}})]. \tag{2.60}$$

For small distortions, that is, closely around the unique minimum $\hat{\mathbf{x}} = \mathbf{x}$ of $d(\mathbf{x}, \hat{\mathbf{x}})$, a large group of relevant distortion measures can be expressed as

$$d(\mathbf{x}, \hat{\mathbf{x}}) \approx \frac{1}{2}(\mathbf{x} - \hat{\mathbf{x}})^T \mathbf{D}_x(\mathbf{x})(\mathbf{x} - \hat{\mathbf{x}}). \tag{2.61}$$

$\mathbf{D}_x(\mathbf{x})$ is a positive semi-definite N by N dimensional matrix, the so-called sensitivity matrix. Equation (2.61) implies a *locally quadratic* behavior of $d(\mathbf{x}, \hat{\mathbf{x}})$. It can be shown that a high-rate vector quantizer minimizing the right-hand side of (2.61) will have the same centroid density, Voronoi region shapes and performance as the quantizer minimizing the minimizing the original distortion measure [6].

Let us investigate entropy-coded companding vector quantization, i.e.,

$$\mathbf{x} \rightarrow \mathbf{F}(\cdot) \rightarrow \mathbf{Q}(\cdot) \rightarrow \mathbf{F^{-1}}(\cdot) \rightarrow \hat{\mathbf{x}}$$

where $\mathbf{F}(\cdot)$ is an invertible one-to-one compressor function with inverse $\mathbf{F^{-1}}(\cdot)$ and $\mathbf{Q}(\cdot)$ a lattice quantizer. Let $\mathbf{F}'(\cdot)$ be the Jacobian for the compressor $\mathbf{F}(\cdot)$. It is shown in [2] that the optimal compressor function minimizing (2.61) satisfies

$$\mathbf{F}'(\mathbf{x})^T \mathbf{F}'(\mathbf{x}) = c\,\mathbf{D}_x(\mathbf{x}). \tag{2.62}$$

Furthermore, for the optimal compander and for small distortions, the mean squared-error on the compressed signal corresponds to the perceptual error on the output [2]. This is what offers the flexibility desired in *FlexCode*, that is, the decoupling of perceptual aspects from the actual quantization, which can then be done in a mean squared-error optimal sense.

We propose using a linear compressor function

$$\mathbf{F}(\mathbf{x}) = \mathbf{F}'(\mathbf{x})\,\mathbf{x}, \tag{2.63}$$

which fulfills (2.62) if we regard $\mathbf{F}'(\mathbf{x})$ constant once $\mathbf{x}$ is known. That is, from (2.62) the optimal compressor is obtained as the root of the Sensitivity Matrix.

It remains to derive the optimal compander for the distortion measure defined in (2.58). With

$$|\mathbf{S}| = |\mathbf{X} - \hat{\mathbf{X}}| \tag{2.64}$$

the perceptual distortion (2.58) becomes

$$d_{Par}(\mathbf{X}, \hat{\mathbf{X}}) = C_s \hat{L} \sum_i \frac{\frac{1}{N}\sum_f |h_f|^2 \left|g_f^i\right|^2 \left|X_f - \hat{X}_f\right|^2}{\frac{1}{N}\sum_f |h_f|^2 \left|g_f^i\right|^2 |X_f|^2 + C_a}. \tag{2.65}$$

Deriving the Sensitivity Matrix for the distortion measure in (2.65) based on the guidelines in ([7]) is unnecessarily complicated. Since the distortion measure is already in a quadratic form, an expression for the Sensitivity Matrix can easily be obtained just by rewriting (2.65) in matrix-vector notation (as outlined in [8]). Let $\mathbf{H}$ be a diagonal N-dimensional matrix whose diagonal is formed by the frequency response of the OM filter, i.e. $\mathbf{h}$. In the same fashion $\mathbf{G}^i$ is defined, so that the frequency response $\mathbf{g}^i$ of the channel-i auditory filter forms the diagonal of $\mathbf{G}^i$. We then have

$$d_{Par}(\mathbf{X}, \hat{\mathbf{X}}) = \frac{C_s \hat{L}}{N} \sum_i \frac{\left(\mathbf{G}^i \mathbf{H} |\mathbf{X} - \hat{\mathbf{X}}|\right)^T \left(\mathbf{G}^i \mathbf{H} |\mathbf{X} - \hat{\mathbf{X}}|\right)}{\frac{1}{N}\left(\mathbf{G}^i \mathbf{H} \mathbf{X}\right)^T \left(\mathbf{G}^i \mathbf{H} \mathbf{X}\right) + C_a}. \tag{2.66}$$

Since $|\mathbf{X} - \hat{\mathbf{X}}|$ is independent of the channel number i it is possible to transform the expression to

$$d_{Par}(\mathbf{X}, \hat{\mathbf{X}}) = |\mathbf{X} - \hat{\mathbf{X}}|^T \left[\frac{C_s \hat{L}}{N} \sum_i \frac{(\mathbf{G}^i \mathbf{H})^T (\mathbf{G}^i \mathbf{H})}{\frac{1}{N}\left(\mathbf{G}^i \mathbf{H} \mathbf{X}\right)^T \left(\mathbf{G}^i \mathbf{H} \mathbf{X}\right) + C_a}\right] |\mathbf{X} - \hat{\mathbf{X}}|. \tag{2.67}$$

Comparing (2.67) with (2.61) one can see that $\mathbf{D}_X(\mathbf{X})$ is given by

$$\mathbf{D}_X(\mathbf{X}) = 2\frac{C_s\hat{L}}{N}\sum_i \frac{(\mathbf{G}^i\mathbf{H})^T(\mathbf{G}^i\mathbf{H})}{\frac{1}{N}\left(\mathbf{G}^i\mathbf{H}\mathbf{X}\right)^T\left(\mathbf{G}^i\mathbf{H}\mathbf{X}\right) + C_a} \tag{2.68}$$

and also that the Sensitivity Matrix-based representation of the distortion measure (2.61) and the original distortion measure (2.58) are equivalent in this case.

It is easy to see that the matrix defined in (2.68) is diagonal. That is, the optimal perceptual compressor for the spectral perceptual model consists of multiplying the spectral magnitude $\mathbf{X}$ with a diagonal weighting matrix $\mathbf{W}(\mathbf{X})$ before quantization

$$\mathbf{Y} \quad = \quad \mathbf{W}(\mathbf{X})\,\mathbf{X}, \tag{2.69}$$

or element-wise

$$Y_f \quad = \quad w_f\,X_f, \qquad f = \{1, \ldots, N\}, \tag{2.70}$$

where $\mathbf{Y} = [Y_1, Y_2, \ldots, Y_N]^T$ is the compressed signal and

$$w_f = [\mathbf{W}(\mathbf{X})]_{f,f} = [\mathbf{D}_X(\mathbf{X})]^{\frac{1}{2}}_{f,f}\,. \tag{2.71}$$

The quantized and coded signal is then expanded accordingly

$$\hat{\mathbf{X}} = \mathbf{W}^{-1}(\mathbf{X})\,\mathbf{Q}(\mathbf{Y}). \tag{2.72}$$

In section 2.6.4 we explain how (2.69) and (2.72) are currently implemented in *FlexCode*, namely by means of filters operating on the time signals.

2.6.4 The Perceptual Weighting Filter

To implement the optimal perceptual compander derived in section 2.6.3 we chose to use pre- and post-filters as originally proposed in [9]. A typical quantizer minimizes the squared error. So does the main structure of *FlexCode*. If we apply a pre-filter such that the squared error in the filtered signal is equivalent to (2.58), we can keep the main structure. We notice that with (2.71) and (2.68)

$$d_{Par}(\mathbf{X}, \mathbf{S}) = \sum_f |S_f w_f|^2 = \sum_f |X_f w_f - \hat{X}_f w_f|^2. \tag{2.73}$$

If the original spectrum is weighted by w_f and the coded one by w_f^{-1}, we change from perceptual distortion into squared error. Moreover, the weighting over spectrum can be done by linear filtering in time domain, as illustrated in Figure 2.11.

Given an amplitude response, there are many ways to obtain a filter that approximates that amplitude response [10]. An all-pole filter can describe the inverse filter quite accurately, because the amplitude response of the inverse filter, which shapes the noise, is similar to the spectrum of the signal, which is described by an all-pole model in *FlexCode*. Levinson algorithm can be used to derive the filter.

There are several practical considerations.

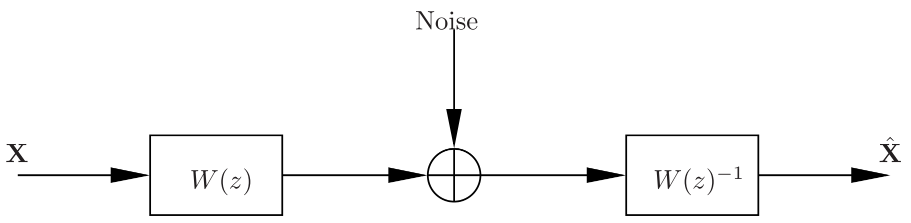


Figure 2.11: *Pre- and post-filters.*

1. Frequency resolution.
   We must decide the frequency grid for $|w_f|^2$(2.71). The equivalent rectangular bandwidth (ERB) of a gamma-tone filter decreases with decreasing center frequency. The frequency resolution must be chosen such that the filter with the smallest ERB can have enough accuracy.

2. Zero drift across segments.
   The auditory filter attenuates frequency component near $0Hz$ severely, which means distortion in that frequency range does never matter. Then the perceptual weighting may cause a floating DC in the coded waveform, which is not proper for coding applications. A possible solution is to limit the attenuation of auditory filter at low frequency.

3. Filter order.
   Since the number of peaks shown in the aimed amplitude response is less than both the number of peaks in the signal spectrum and the number of gamma-tome filters, a good choice of the filter order is the larger number between the two.

4. The choice for C_a.
   It is tricky to utilize C_a in coding applications, since the loudness at which the sound is played is not known. However, we do have some clues for choosing a proper value. One possible assumption is that sound with full range sinusoid corresponds to the maximum loudness people accept when they listen to music through regular electrical devices. Another assumption is that sound with relatively constant power is played at a comforting loudness. According to the first aspect, we can fix this parameter by a constant. *FlexCode* applies this idea. According to the second one, we may further make it adaptive.

5. The choice for C_s.
   Only to minimize the detectability in a particular block does not need C_s. This parameter can help to keep a constant detectability over blocks. A constant detectability can contribute to a small bit stream over the segments that have little power.

2.6.5 Conclusions

The above described integration of perceptual modeling in *FlexCode* has three main advantages. The sensitivity matrix can be derived for a wide class of distortion measures, offering a unified approach to integrate advanced perceptual models and thereby extendability in case better models for human hearing become available in the future. Multidimensional companding facilitates separability (and thereby optimality) of *FlexCode* components. Finally, the redundancy between the signal model and the perceptual model is exploited by deriving the perceptual model from the signal model parameters instead of sending perceptual model parameters to the decoder.

Bibliography

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

[2] T. Linder, R. Zamir, and K. Zeger, "High-Resolution Source Coding for Non-Difference Distortion Measures: Multidimensional Companding," vol. 45, no. 2, pp. 548–561, Mar. 1999.

[3] M. R. Schroeder and B. S. Atal, "Code-Excited Linear Prediction (celp): High-Quality Speech at Very Low Bit Rates," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, vol. 10, Tampa, Florida, USA, 1985, pp. 937–940.

[4] J. H. Chen, N. Jayant, and R. V. Cox, "Improving the Performance of the 16 kb/s LD-CELP Speech Coder," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, vol. 1. San Francisco, California, USA: IEEE, Mar. 1992, pp. 69–72.

[5] S. van de Par, A. Kohlrausch, G. Charestan, and R. Heusdens, "A New Psychoacoustical Masking Model for Audio Coding Applications," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, vol. 2, Orlando, FL, USA, 2002, pp. 1805–1808.

[6] W. R. Gardner and B. D. Rao, "Theoretical Analysis of the High-Rate Vector Quantization of LPC Parameters," vol. 3, no. 5, pp. 367–381, Sep. 1995.

[7] J. H. Plasberg and W. B. Kleijn, "The Sensitivity Matrix: Using Advanced Auditory Models in Speech and Audio Processing," *IEEE Trans. Audio, Speech and Language Process.*, vol. 15, no. 1, pp. 310–319, Jan. 2007.

[8] P. Petkov, "The Sensitivity Matrix for a Spectral Auditory Model," Master's thesis, KTH (Royal Institute of Technology), May 2005.

[9] B. Edler and G. Schuller, "Audio Coding Using a Psychoacoustic Pre- and Post-Filter," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing ICASSP '00*, G. Schuller, Ed., vol. 2, 2000, pp. II881–II884 vol.2.

[10] L. B. Jackson, *Digital filters and signal processing : with MATLAB exercises*. Kluwer Academic, 2006.

Chapter 3

Implementation of the Baseline Source Coder

3.1 Introduction

This section outlines goals of the implementation by referring to the *FlexCode* principles and presents solutions that have been implemented within the baseline platform. The first goal of the implementation is to facilitate research and design processes on the proposed coder architecture. The baseline platform should also provide a flexibility that would allow for incorporating further extensions like Multiple-Description Coding and new Lattice quantization methods (Sec. 2.5). Additionally, the baseline coder is required to become a starting point for proceeding towards a real-time demonstration. These goals have been achieved by the implemented baseline platform.

Section 3.2 describes the proposed architecture and provides the specifications of the baseline coder. Firstly it contains a high-level overview that explains how the *FlexCode* principles are incorporated within the baseline platform. Secondly it describes in more detail solutions that are used. In section 3.3 the guidelines for the implementation are presented as well as the motivation for the selected approach and practical solutions within the source code. Section 3.4 contains comments on complexity of the coder and on the possibilities of reducing it. Section 3.5 presents preliminary results.

3.2 Architecture and Specifications

The architecture of the baseline coder has been designed to match the principles of *FlexCode*. As a result, a fully scalable coder has been obtained. The coding scheme can adapt to any particular rate on a frame by frame basis, and computational complexity does not depend on the rate. Such properties can be obtained by exploiting high-rate theory and probabilistic source modelling that allow to design quantizers using analytical expressions for given optimality criteria. Independently of the design requirements the architecture preserves this flexibility for cases of constrained entropy and constrained resolution.

Another level of flexibility has been achieved by using the same coder architecture independent of the transform that is used. The architecture can employ

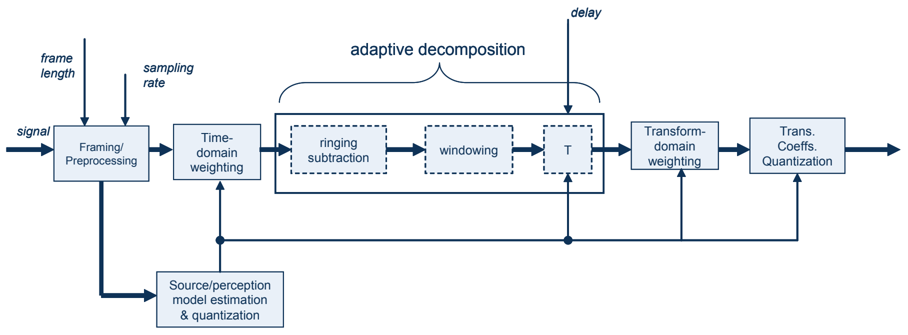


Figure 3.1: High-level overview of the baseline architecture of *FlexCode*.

the Karhunen-Loeve transform (KLT) or the modulated-lapped transform (MLT) in coding. The KLT is intended to be used in a generic configuration of the coder, while the MLT is its low-complexity equivalent.

The baseline architecture includes generic modelling of the perception with a perceptual model that is derived from the signal model. This results in low complexity of the perceptual modelling and in addition it is not required to transmit or store the perceptual model. The current architecture of the baseline source coder uses the van de Par perceptual model [1].

In the below sub- sections the architecture of the baseline coder is described. Firstly a high-level overview is presented and later the architecture is explained in more detail. Finally a detailed specification of the baseline source coder is provided.

3.2.1 High-level Overview of the Baseline Architecture

A high-level diagram of the baseline architecture is depicted in Fig. 3.1. In the proposed coding scheme an audio signal is segmented into frames of arbitrary length. Each frame is additionally segmented into an arbitrary number of sub-frames. The segmentation depends on the configuration of the coder. Frames and sub-frames can be overlapped if needed. The framing includes signal preprocessing by means of a high-pass filter.

The signal model is derived of a set of p-order linear prediction coding (LPC) coefficients and a prediction gain obtained every frame and an open-loop pitch period estimate obtained every 10 ms. The signal model consisting of the LPC model and the pitch model based on the open-loop pitch period estimate is used as basis for the derivation of the van de Par perceptual model.

The next stage of processing is the time domain weighting. A perceptual filter obtained from the perceptual model is used to render the signal in a domain where the squared error can be assumed perceptually relevant, i.e., the noise shaping that occurs by the perceptual post-filter gives the quantization noise a perceptually optimal shape.

The time domain weighting is followed by an adaptive decomposition. This includes ringing subtraction that removes intra-block dependencies. Since the ringing subtraction operates in a closed-loop configuration, this stage introduces a delay in the coding. In case when MLT is used, a windowing is applied, which also includes a window-switching mechanism. Currently the ringing subtraction only works for non-overlapping windows. The implementation of ringing subtraction for overlapping windows is yet unsolved.

The adaptive decomposition is finalized by applying a transform to the signal to be encoded. In this step the inner-block dependencies are removed from the encoded signal. This part of architecture is generic in terms of transform that is used. Changing the transform affects the data-flow within the coder, but it does not affect the architecture as the components of the architecture are designed to be generic. This stage of processing includes transform computation, which in case of KLT uses the composite signal model that consists of the AR signal model derived from the LPC model and the pitch model. In case of MLT transform the composite model is not used in transform derivation, instead the transform is fixed for a given sub-frame size.

In the next stage the transform is applied directly to the signal on sub-frame basis. This can be followed by normalization if such is required by the channel coder. Finally the transform coefficients are quantized.

The system contains a local decoder to allow some blocks of the coder to run in a closed-loop fashion. The local decoder generates the decoded signal from the signal model and the quantized transform coefficients. The signal is run through the inverse perceptual filter to arrive in the original signal domain. Optionally a post-filter can be applied to the decoded signal.

3.2.2 Detailed Description of the Baseline Architecture

A detailed block diagram of the baseline architecture of *FlexCode* source coder is shown in Fig. 3.2.

The baseline source coder can operate on frames and sub-frames of arbitrary lengths. There are however constraints that a multiple of sub-frames results in a frame. The architecture is fully scalable in terms of the rate. Moreover the components of the architecture are generic in terms of the transform that is used. Selection of the transform affects the data-flow between the functional blocks of the coder, therefore for clarity it is convenient to consider KLT coder and MLT coder separately. The architecture is discussed here by describing the functional blocks that can be distinguished within the platform in the sub-sequent paragraphs.

The coding is performed on frames of segmented signal. The segmentation is preceded by setting up the framing algorithm. The length determination block makes the decisions about the length of the frames and sub-frames. This functionality is used in particular when the window switching mechanism is enabled and the decisions are based on the feedback from the transient detection block. Additionally the lengths of frames are constrained by a maximum acceptable delay. In the simplest setup the coder operates with a fixed frame length and with a fixed number of sub-frames within the frame. Usage of MLT introduces overlap between the sub-frames, while in case of KLT coder such overlap is not necessary.

The framing block performs the actual segmentation of the signal to be encoded. The framing block is designed in a way that supports variable frame and sub-frame lengths and overlapping as defined in the length determination block. Additionally, a pre-processing is introduced within the framing block by means of a high-pass filter with a cut-off frequency of 67 Hz. The preprocessing aims at removing DC

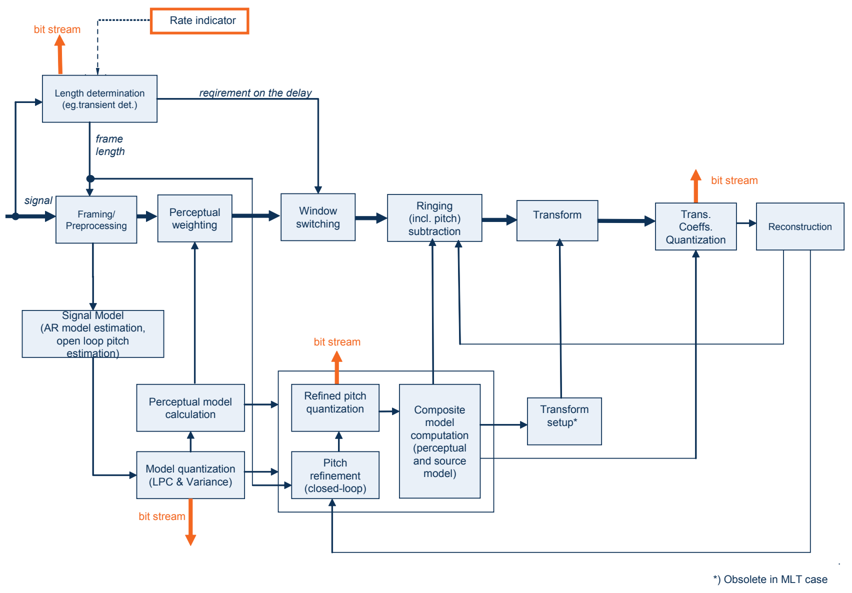

Rate indicator
bit stream
Length determination (eg.transient det.)
reqirement on the delay
frame length
signal
Framing/ Preprocessing
Perceptual weighting
Window switching
Ringing (incl. pitch) subtraction
Transform
bit stream
Trans. Coeffs. Quantization
Reconstruction
Signal Model (AR model estimation, open loop pitch estimation)
bit stream
Perceptual model calculation
Refined pitch quantization
Composite model computation (perceptual and source model)
Transform setup*
Model quantization (LPC & Variance)
Pitch refinement (closed-loop)
bit stream
*) Obsolete in MLT case

Figure 3.2: Detailed block diagram of the *FlexCode* baseline platform.

component of the signal to be encoded.

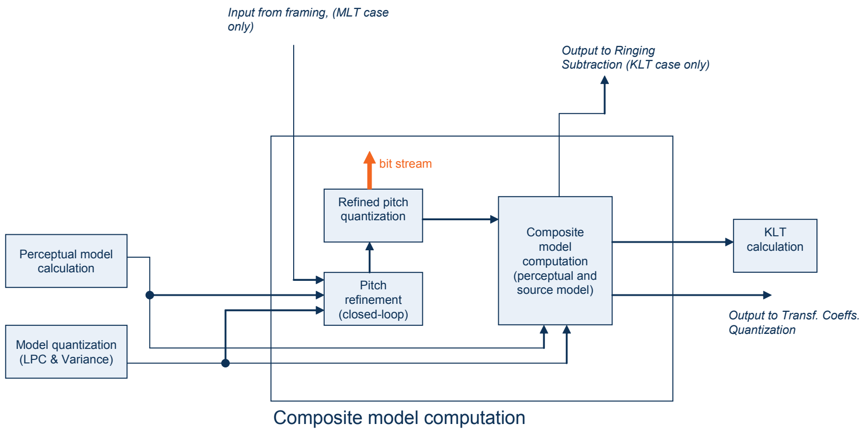
The signal modelling is an important functionality of the flexible baseline platform as usage of the statistical signal models is crucial to obtain scalability of the quantizers within the coder. In case of the KLT coder the signal model is also used to derive the transform. In addition, the signal model composed of AR-model and pitch model is used to derive the perceptual model. A set of LPC coefficients is obtained every frame and interpolated for the sub-frames. The LPC coefficients are quantized in the LSF domain using a GMM. The LPC coefficients and prediction variance σ are obtained by using Levinson-Durbin algorithm. Let $\{a_i\}_{i=1}^{p}$ be the interpolated LPC coefficients for a current sub-frame. The corresponding LP synthesis filter is of form

$$\frac{\sigma}{A(z)} = \frac{\sigma}{1 + a_1 z^{-1} + a_2 z^{-2} + ... + a_p z^{-p}}. \tag{3.1}$$

The signal modeling block performs also an open-loop pitch prediction. The corresponding LTP synthesis filter is of form

$$\frac{\sigma_p}{1 - \beta z^{-d}}, \tag{3.2}$$

where β is a pitch model gain, σ_p is a variance of the pitch and d is the pitch period. The pitch model is incorporated together with the AR-model creating a complete signal model that is accurate enough to be used to derive a corresponding perceptual model. The parameters of the pitch model are quantized. The quantization can be performed with constrained entropy or resolution. The statistical properties of the open-loop pitch period are modelled by a Gaussian mixture model. In the current version of the baseline platform the open-loop pitch estimator is implemented according to the G.729.1 standard [2].



Input from framing, (MLT case only)
Output to Ringing Subtraction (KLT case only)
bit stream
Refined pitch quantization
Composite model computation (perceptual and source model)
KLT calculation
Perceptual model calculation
Pitch refinement (closed-loop)
Output to Transf. Coeffs. Quantization
Model quantization (LPC & Variance)
Composite model computation

Figure 3.3: Computation of the composite model.

The signal model is used in a perceptual modelling block to derive the van de Par perceptual model according to the method presented in Sec. 2.6. The perceptual weighting is performed on sub-frame basis.

The pitch refinement block performs closed-loop pitch estimation. The closed-loop pitch analysis includes finding a closed-loop pitch period and the corresponding gain. The refined pitch model parameters are quantized and the quantization can be performed in constrained entropy or constrained resolution. The analysis is performed on a signal that is deemphasized by means of perceptual weighting. The closed-loop pitch analysis requires fine time-resolution, as even small deviations of the pitch delay seriously affect the performance of model. The closed-loop pitch estimation takes into account two possible situations. One where the pitch period is larger than the length of a sub-frame, the other where the pitch period is smaller than the sub-frame length. In both cases a maximum likelihood estimator is used to get the estimates for refined pitch period and the gain of the adaptive codebook.

The composite model is a refined signal model that is necessary to derive the KLT. It consists of the signal model that is interpolated on sub-frame basis and the refined-pitch model obtained during closed-loop pitch analysis. Additionally, the composite model includes the perceptual model necessary for computation of the KLT. A detailed block diagram describing the relations between the blocks required for the composite model computation is shown in Fig. 3.3.

The baseline coder removes redundancy in two steps. The first one deals with an intra-block redundancy, which is removed by means of ringing subtraction. This aims at removing the zero-input model response from the sub-frame that is going to be encoded. An inner-block redundancy is removed later by means of a transform.

In the KLT case the transform computation requires usage of the composite model. First an impulse response h of the composite model is computed, taking into account the perceptual filter that is used. The impulse response is truncated to a size of a sub-frame. Let $h_0, h_1, ...h_{K-1}$ denote the truncated impulse response for a sub-frame containing K samples. Let

$$H = \begin{bmatrix} h_0 & 0 & \dots & 0 \\ h_1 & h_0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ h_{K-1} & h_{K-2} & \dots & h_0 \end{bmatrix}, \tag{3.3}$$

then the estimate of the covariance matrix Σ of the sub-frame can be found as

$$\Sigma = HH^t. \tag{3.4}$$

Finally, an Eigenvalue decomposition $\Sigma = U\Lambda U^t$ is performed. U is an orthogonal matrix for which $U^tU = I$ and $\Lambda = diag\{\lambda_0, ..., \lambda_{K-1}\}$ is a diagonal matrix of Eigenvalues.
In case of the MLT the computation of the transform is reduced to finding a set of sinusoidal basis functions including appropriate windowing. For a given sub-frame size these basis functions are fixed.

The transform is applied directly to the perceptually weighted signal segmented into sub-frames. In case of KLT a K-dimensional sub-frame is transformed into K-dimensional vector of coefficients. In case of MLT a $2K$-dimensional sub-frame is transformed into a K-dimensional vector of coefficients.

The quantization of transform coefficients looks different in cases of constrained entropy and constrained resolution. In the case of constrained entropy a uniform quantization is optimal (as long as the MSE error criterion can be assumed) with a fixed quantization step Δ. Let y be a K-dimensional vector of transform coefficients, which may be written as $[y_0, ..., y_k, ..., y_{K-1}]^t$. A scalar quantizer is applied independently to each dimension. The quantization index can by obtained by $\hat{d}_k = [y_k/\Delta]$, where $[\cdot]$ denotes rounding. The quantization indices are available for the channel coder.

In case of constrained resolution a normalization using the standard deviation $\sqrt{\lambda_k}$ of the transform coefficients in vector y is applied first. Next a scalar compressor (optimal for a Gaussian random variable with variance 1) is used independently for each dimension. Later a uniform scalar quantizer is applied. The Fox algorithm is used to allocate the bits for the scaled and compressed transform coefficients that are going to be encoded and transmitted. [3]

The quantization of the transform coefficients can be extended to incorporate an adaptive MDC quantization whenever a robustness against packet losses is needed. This is not included in the present baseline system. The rate spent on transmitting the quantized coefficients is the largest contribution to the total rate. Using MDC technique to protect this part of the bitstream against packet losses appears natural and consistent with the notion of MDC that it is applicable whenever a degraded quality is acceptable. In a system designed to combat packet losses there is also a need to protect the signal model. As the average rate that is spent on the model is constant and also significantly smaller than the rate used to transmit the transform coefficients, the model can be protected by means of forward error correction (FEC).

The existence of a closed-loop is required for performing ringing subtraction and the refined pitch estimation. This motivates implementing a local reconstruction block. The block reconstructs the signal using the quantized transform coefficients and the quantized model parameters.

3.2.3 Specifications of the Baseline Coder

The baseline architecture has an algorithmic delay that depends on its configuration. The lowest algorithmic delay is obtained in a configuration when only the computation of the signal model and the ringing subtraction contribute to it. The baseline platform is fully flexible in terms of its configuration, therefore a detailed specification can be given for a particular setup of the coder. As an example we give a specification of a coder that is set up to work with audio signals with a sampling frequency of 16 kHz using the KLT. A reasonable setup of the coder for such a sampling frequency assumes frames of 20 ms length and usage of 4 sub-frames, 5 ms each without overlapping. The delay introduced by LPC analysis is then 25 ms as 5 ms look-ahead is taken into account while deriving the model parameters. The current setup includes open-loop pitch estimation that is performed according to the G.729.1 standard. Thus, the pitch period is predicted every 10 ms. The open-loop pitch estimation does not contribute to the algorithmic delay as it is performed in parallel fashion to the signal model estimation. Closed-loop pitch estimation and ringing subtraction require algorithmic delay of one sub-frame. In the current setup it is 10 ms. The algorithmic delay increases as window switching feature of the coder is enabled, depending on the length of the longest window that is used.

The baseline platform for the *FlexCode* source coder is fully flexible with respect to the sampling frequency, target bitrate, and optimality conditions for the quantizers that are used.

3.3 Implementation of the Baseline Coder

3.3.1 Guidelines for Implementation

There is a number of reasons to select `MATLAB` as a suitable environment for constructing the baseline platform. The most important reason is a requirement to have a possibility for fast implementation of the solutions provided by all the partners. The implementation in `MATLAB` provides a good insight into the coder at algorithmic level, therefore it stimulates collaboration between the partners. Additionally, it is required to have a good transparency in terms of progress, which is easily obtained using the high-level programming language of `MATLAB`.

The baseline platform implemented in `MATLAB` is a starting point for a real time implementation that will be created using `C/C++`. This is part of work falling under **WP.4**, which aims at integrating the programs delivered by **WP.1** and **WP.2** into a single system. The computational efficiency of all the routines will be investigated. Based on this analysis the computational complexity will be decreased if needed.

3.3.2 Flexible Baseline Platform

The main routine of *FlexCode* implements the algorithm that is segmented in hierarchies of design blocks as shown in Figures 3.2 and 3.3. Such an approach simplifies the management of the platform and improves the transparency of the coding algorithm.

The blocks correspond to the functionalities of the coder that can be considered as interacting entities. The interface of the blocks is generic. An example of the interface is shown below.

```
[output, state] = functional_block(input, setup, state)
```

Each block is implemented as a separate function and has a standard interface to communicate with other blocks. Each block has a separate memory (state) and a separate setup. The outputs of the blocks are organized within structures that are accessible by other blocks depending on the necessity. This allows to achieve a clear data flow between the blocks. Another advantage of this approach is that it results in a flexible configuration of a whole coder as it can be easily changed by rearranging the data flow between the blocks or introducing new blocks.

The main routine of the baseline platform consists of two parts. The first one is an initialization, during which all the blocks are configured depending on the design constrains. The second part includes a loop containing the blocks that are used during data flow.

3.3.3 Routines

The routines of the main function are explained below in the order they are called in the main loop of the encoder. The first few routines are the initialization routines. Their task is to define the setup of the different functional blocks and to initialize their memory (state). To ensure that the different blocks are setup consistently some of the initialization routines use the output of preceding initialization routines.

```
general_setup = init_main(rate, constraints)
```

The routine prepares a setup structure for the main function. It requires an input with a target total rate (averaged rate in the entropy-constrained case) and additionally a set of constraints including an optimality criterium for the quantizers (constrained entropy or resolution), duration of a frame and number of sub-frames within a frame. Also the transform that is used for coding is specified.

```
[setup_length_det, state_length_det] =
init_length_determination(general_setup)
```

The routine prepares a setup structure for the length determination block. This structure can handle a setup for variable lengths of frames and subframes and additionally configure transient detection mechanism that is a part of the window switching feature of the coder.

```
[setup_framing, state_framing] = init_framing(general_setup,
setup_length_det)
```

The routine prepares a setup structure for the framing block. It can handle variable lengths of a frames and subframes and overlapping of frames and subframes.

```
[setup_signal_model, state_signal_model] =
init_signal_model(general_setup, setup_length_det)
```

The function initializes the signal modeling block including a setup of order of the prediction, type of performed interpolation of AR-model coefficients, open-loop pitch estimation, and quantization of model parameters.

```
[setup_perc_model, state_perc_model] =
init_perc_model(general_setup, setup_length_det)
```

The function generates a setup structure for the block performing perceptual model derivation.

```
[setup_normalize, state_normalize] = init_normalize( ...
general_setup, step_size)
```

This function creates a setup structure for the normalization block. It includes a setup for a quantizer of a sub-block gain.

```
[setup_window_switch, state_window_switch] =
init_window_switch(general_setup)
```

This function creates a setup structure for the block performing window switching.

```
[setup_pitch_refine, state_pitch_refine] =
init_pitch_refine(general_setup, setup_length_det,
setup_signal_model)
```

This function prepares a setup structure for the pitch refinement block. This setup includes initialization of the filters used in the closed-loop pitch estimation and settings for the interpolation.

```
[setup_pitch_ref_quant, state_pitch_ref_quant] =
init_pitch_ref_quant(general_setup,setup_pitch_refine)
```

This function generates a setup structure for quantization of the parameters of the refined pitch model.

```
[setup_comp_model, state_comp_model] =
init_comp_model(general_setup, setup_length_det,
setup_signal_model, setup_perc_model, setup_pitch_refine)
```

This function generates a setup structure for the block performing computation of the composite model.

```
[setup_ring_sub, state_ring_sub] = init_ring_sub(general_setup,
setup_comp_model)
```

This function generates a setup structure for a ringing subtraction block. It also initializes the filters used in the ringing subtraction block.

```
[setup_transf_comp, state_transf_comp] =
init_transf_comp(general_setup, setup_comp_model)
```

This function generates a setup structure for the block computing the transform.

```
setup_transform = init_transform(general_setup, ...
setup_length_det)
```

This function generates a setup structure for the block applying the transform to the perceptually weighted subframe.

```
[setup_coeff_quant, state_coeff_quant] =
init_coeff_quant(general_setup, setup_length_det, step_size)
```

This function generates a setup structure for the block performing quantization of the transform coefficients.

```
[output_local_reco, setup_local_reco, state_local_reco] =
init_local_reco(general_setup, setup_comp_model, ...
setup_transform)
```

This function generates a setup structure for the block performing local reconstruction.

Functions described below are executed within main loop of the encoder function. They process the signal either on a frame basis or a subframe basis.

```
[output_length_det,stream_length_det, state_length_det] = ...
length_determination(signal, setup_length_det, ...
state_length_det)
```

This function computes the lengths of frames and subframes. It operates on the signal to be encoded performing segmentation. Additionally, this block incorporates the transient detection module that is a part of the window switching functionality of the coder.

```
[output_framing, state_framing] = framing(signal,
output_length_det, setup_framing, state_framing)
```

This function performs signal segmentation according to the decisions made in the length determination block. Additionally, pre-processing (high-pass filtering) is applied here.

```
[output_signal_model, stream_signal_model, state_signal_model] =
        signal_model(output_framing, ...
        setup_signal_model, state_signal_model)
```

This function performs signal model extraction. The signal model is obtained every frame. The model parameters include LP coefficients and signal variance. The model can be extended to include LTP pitch modelling. In such case an open-loop estimate of the pitch period is found for 10 ms blocks. This functional block includes quantization of the model parameters with support of constrained entropy and constrained resolution cases. The model parameters are interpolated for the subframes. Interpolation is performed on the quantized parameters.

```
[output_perc_model, state_perc_model] =
perc_model(output_signal_model, output_framing, setup_perc_model,
state_perc_model)
```

This function computes the perceptual model based on the signal model. A filter for perceptual weighting is computed within this routine. The filtering is applied to the signal delivered from the framing block. It is possible to access the filtered signal organized in frames and subframes depending on coder configuration.

```
[output_window_switch, state_window_switch] = ...
      window_switch(output_perc_weight, output_length_det, ...
      setup_window_switch, state_window_switch)
```

This function applies the different windows necessary to obtain the switching mechanism in conjunction with the length determination and framing blocks. It generates and applies proper windows.

```
[output_pitch_refine, state_pitch_refine] = ...
            pitch_refine(output_perc_model, output_signal_model,
            output_local_reco, setup_pitch_refine, ...
            state_pitch_refine, n_subframe)
```

This function performs closed-loop pitch refinement. The function operates on the subframe basis. The closed-loop parameters of the pitch model include refined pitch period (supporting fractional delay) and adaptive-codebook gain. Since the estimation runs in closed loop this routine requires an input from the local reconstruction block.

```
[output_pitch_ref_quant, stream_pitch_ref, state_pitch_ref_quant]
= pitch_ref_quant(output_pitch_refine, ...
setup_pitch_ref_quant, state_pitch_ref_quant)
```

This function performs quantization of the refined pitch parameters. The refined pitch parameters are quantized on subframe basis. The output structure is the same as the output of the pitch refinement block, so the quantization of pitch parameters can be easily bypassed for debugging purposes.

```
[output_comp_model, state_comp_model] = ...
comp_model(output_pitch_ref_quant, output_signal_model,...
output_perc_model, setup_comp_model, state_comp_model)
```

This function performs calculation of the composite model. The composite model is computed on subframe basis. The model is used in derivation of the KLT. The composite model incorporates the LP model that is interpolated for a subframe basis and the refined LTP model obtained in the pitch refinement block. The composite model exists as a separate block only to make the coding algorithm more transparent.

```
[output_ring_sub, state_ring_sub] = ...
ring_sub(coded_signal,output_comp_model, ...
output_perc_model, output_window_switch, ...
setup_ring_sub, state_ring_sub, n_subframe)
```

This function performs ringing subtraction. The ringing is subtracted on the sub-frame basis and the procedure runs in a closed-loop fashion. Therefore this block requires an access to the coded signal, which is available at the local decoder and the composite model. The ringing subtraction block computes and subtracts the zero-input model response.

```
[output_transf_comp, state_transf_comp] = ...
transf_comp(output_comp_model, setup_transf_comp, ...
state_transf_comp, n_subframe)
```

This function performs computation of the transform. In case of KLT a composite model must be accessible by this block. The output structure contains the transform itself and variances of the components obtained during Eigenvalue decomposition. If the MLT is used instead, the computation of the transform is reduced to finding a proper set of sinusoidal basis functions and the composite model is not used within this block. As noted earlier, the transform is fixed for a given sub-frame size in case of the MLT.

```
[output_transform] = ...
transform(output_ring_sub, output_transf_comp, setup_transform)
```

This function applies the transform to the signal. It returns a structure containing a vector of transform coefficients.

```
[output_normalize,state_normalize] = ...
normalize(output_transform, output_comp_model, ...
output_ring_sub, n_subframe, setup_normalize, ...
state_normalize)
```

An estimate of a gain for each subframe is found and quantized within this block. In addition, the function applies normalization of the transform coefficients vector.

```
output_coeff_quant, stream_coeff] = coeff_quant(output_normalize,
output_comp_model, output_transf_comp, output_length_det,
output_rate_dist, setup_coeff_quant, n_subframe)
```

This function performs quantization of the normalized transform coefficients. This quantization is performed on subframe basis. Both, entropy constrained quantization and resolution constrained quantization are available.

```
[output_local_reco, state_local_reco] = ...
local_reco(output_transf_comp, output_comp_model, ...
output_ring_sub, output_coeff_quant, output_normalize, ...
output_length_det, setup_local_reco, state_local_reco, ...
n_subframe)
```

This function implements a local decoder. The decoder performs signal reconstruction on subframe basis using the quantized model parameters and the quantized transform coefficients. The block is necessary to implement closed-loop elements within the coder architecture.

3.4 Comments on the Complexity

At the current stage of development complexity is not our major concern. Still, the architecture is designed to allow a clear insight into complexity at all the stages of the coding algorithm. The stages that have the highest complexity are identified and throughout the next period of the development an effort will be made to reduce the complexity associated with them. An estimate of the relative complexity associated with the different steps is shown in Figure 3.4. The reduction of complexity is also a natural step towards a real-time demonstration platform. However, the clear separation of functional entities is likely to cause a number of duplicate computations in the codec. Once the codec has matured, optimization towards avoiding such duplicate computations has to be made.

The current baseline platform was designed to facilitate solutions provided by the partners. It is done in a way that assures a full flexibility of coder configuration and yet transparency in terms of the algorithms that are used. This transparency led to a certain level of inefficiency. Therefore the complexity can be significantly decreased at more mature stages of the coder, for instance, by reducing the number of filtering operations or by taking the advantage of the flexible architecture and selecting a proper configuration of the coder for a specific application.

The baseline platform is fully flexible in terms of the used transform. The mentioned kind of flexibility is also important for reduction of the complexity. The KLT allows for achieving *FlexCode* goals regarding the flexibility and scalability, however computation of the transform requires a relatively high computational effort. The MLT configuration is intended to be used as an alternative approach whenever a low complexity is needed. Reduction of complexity in the computation of the KLT transform is now a research topic at KTH.

In general the philosophy of *FlexCode* leads towards a low-complexity coder with low memory requirements.

3.5 Preliminary Results

The progress towards objectives of *FlexCode* is evaluated by means of subjective listening tests on a regular basis. This allows to develop the coder and compare the introduced technologies with respect to the state-of-the-art methods. The developments of **WP.1** are evaluated with comparison to the technologies selected by **WP.5** and the subjective listening test are performed according to the MUSHRA method [4].

An example of MUSHRA test results is shown in Figure 3.5. The test results are averaged for 12 listeners. The average scores are also computed for the four classes of audio clips, which are used during the tests (music, speech + music, speech, noisy speech). All the coders were configured to operate at 24 kbps and ITU-T G.729.1 has been selected as representative state-of-the-art coder at that rate.

Bibliography

[1] S. van de Par, A. Kohlarusch, G. Charestan, and R. Heusdens, “A new psychoacoustical masking model for audio coding applications,” in *Proc. IEEE Inter-*

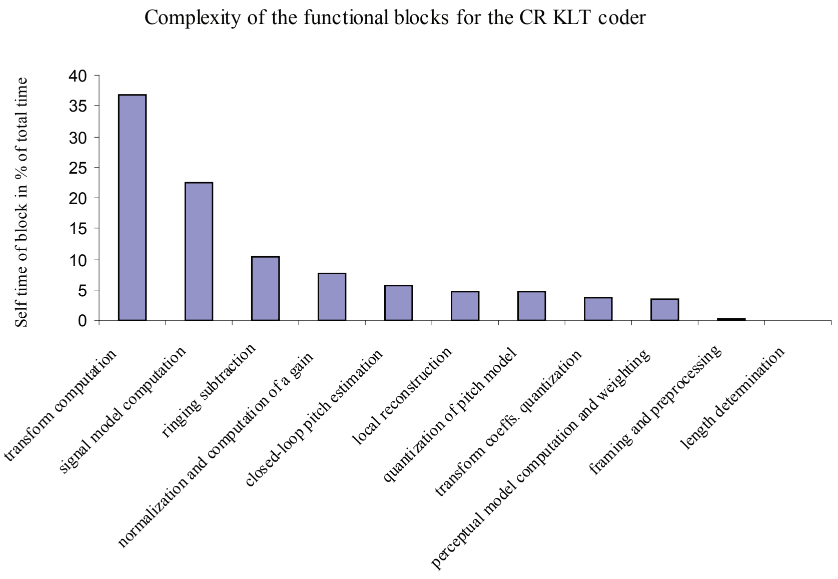


Figure 3.4: Percentage of total processing time per block for the KLT resolution constrained coder.

national Conference on Acoustics, Speech, and Signal Processing (ICASSP '02), A. Kohlrausch, Ed., vol. 2, 2002, pp. 1805–1808.

[2] "ITU-T Rec. G.729.1 "An scalable 8-32kbp/s scalable wideband coder bitstream interoperable with G.729"," Tech. Rep., May 2006.

[3] Y. Shoham and A. Gersho, "Efficient bit allocation for an arbitrary set of quantizers," *IEEE Transactions on Acoustical and Speech Signal Processing*, vol. 36, pp. 1445–1453, 1988.

[4] "Radiocommunication Sector BS.1534-1 ITU Method for subjective assesment of intermediate quality level coding systems (MUSHRA)," Tech. Rep., January 2003.

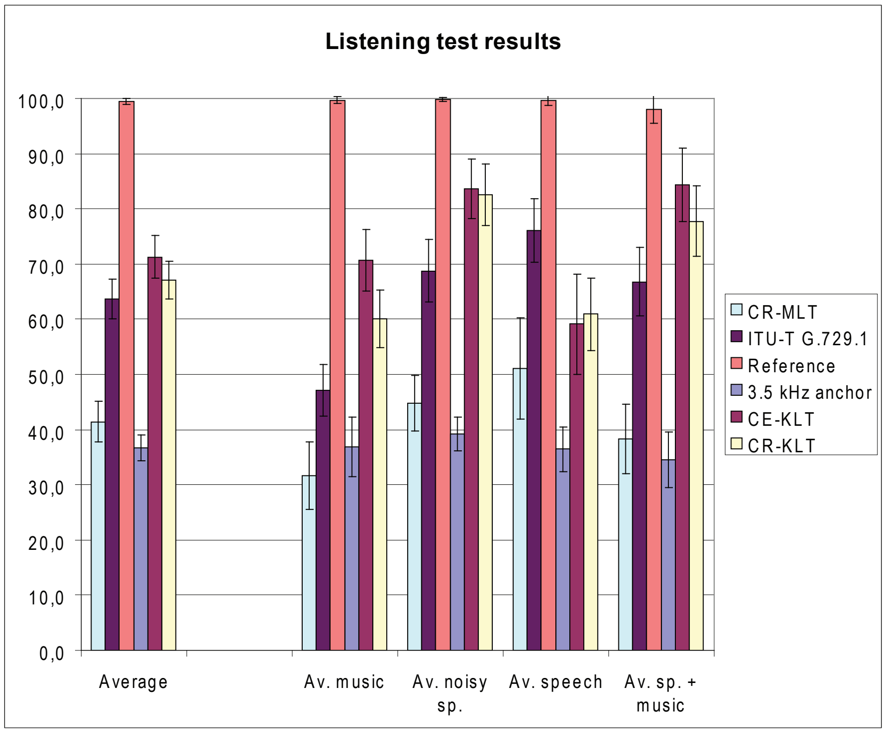


Figure 3.5: Averaged MUSHRA test results (respectively: all clips, 4 music clips, 2 noisy speech clips, 4 speech clips, 2 speech + music clips). The coders appear in the following order: *FlexCode* CR-MLT, ITU-T G.729.1, Reference, 3.5kHz Anchor, *FlexCode* CE-KLT, *FlexCode* CR-KLT. All the results are averaged for 12 listeners. All the coders operate at 24 kbps.