ANALYSIS AND APPLICATION OF LATTICE VECTOR QUANTIZATION USING MIXTURE MODELS AND BIT-PLANE CODING

By

WISARN PATCHOO

A dissertation submitted in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

WASHINGTON STATE UNIVERSITY

School of Electrical Engineering and Computer Science

AUGUST 2011

To the Faculty of Washington State University:

The members of the Committee appointed to examine the dissertation of WISARN PATCHOO find it satisfactory and recommend that it be accepted.

Thomas R. Fischer, Ph.D., Chair

Benjamin Belzer, Ph.D.

Jose Delgado-Frias, Ph.D.

ACKNOWLEDGEMENT

I would like to express my deepest appreciation to my advisor, Professor Thomas R. Fischer, for his unending patience, devoted guidance, continuous encouragement and support, and exceptional constructive suggestions throughout this research work. He taught me many things about doing research and how to be a good researcher, even though sometimes he might not have known that he did it. This work cannot be finished without his superb guidance and support. I am truly appreciative.

I would also like to thank my commitee members, Professors Jose Delgado-Frias and Benjamin Belzer, for their valuable comments and suggestions which absolutely helped improve this dissertation.

I am financially supported by the Bangkok University faculty development program and I am particularly grateful to Bangkok University, Thailand, for this support for a long year of Ph.D study and for giving me an opportunity to achieve my goal in education.

During five years of my study at WSU, I am supported both spiritually and physically from many people. I would like to thank my friends in WSU/UI Thai student association for their cheerful conversations and suggestions. Sometimes, it really lifts my mind up, clears my brain, and make me keep stepping further. I would also like to thank IRL members for useful discussions which sometimes helped me in my work. Moreover, I want to

thank the faculty and staff of the School of Electrical Engineering and Computer Science for their help and support.

I especially wish to thank my parents and my family for their endless enrougement, support, love and understanding. They have been waiting for me for quite a while to be a part of family's story again, after I left them for Ph.D studies five years ago. Last but not least, I really want to thank "Lhing" for being someone meaningful to me.

ANALYSIS AND APPLICATION OF LATTICE VECTOR QUANTIZATION USING MIXTURE MODELS AND BIT-PLANE CODING

Abstract

by Wisarn Patchoo, Ph.D.
Washington State University
AUGUST 2011

Chair: Thomas R. Fischer

This thesis studies lattice vector quantization (LVQ) with application to audio and image sources. The performance of nonzero pulse amplitude quantization implicit in algebraic codebook code-excited linear prediction (ACELP) is examined and it is demonstrated that the quantization used in ACELP is effective in a rate-distortion sense at the encoding rates commonly used. A block-based Gaussian mixture model (GMM) is used to model the marginal distribution and the block energy distribution of transform audio data. The expectation-maximization algorithm is used to estimate the GMM parameters. A GMM-based rate-distortion function is derived and shown to closely match the observed spherical LVQ performance. Then, we move forward to the lattice VQ on transformed image. The GMM is used to motivate a subband image coding algorithm based on lattice-based spherical VQ and lattice-based pyramid VQ. The algorithm partitions a subband image into blocks of various sizes, depending on their energy and complexity constraints on the enumeration encoding of lattice codevectors. Using the cubic lattice, the algorithm provides performance competitive with the set partitioning in hierarchical trees (SPIHT) algorithm.

A bit-plane coding method is developed for the encoding of binary lattice codevectors as binary codewords, yielding an embedded bitstream. In sign-magnitude representation, only a few least significant bit-planes are constrained due to the structure of the lattice, while there is no restriction on other more significant bit-planes. Simple encoding methods for the lattice-defining bit-planes of the D_4, RE_8, and Barnes-Wall 16-dimensional lattices are described. Simulation results for these lattices show that partial decoding of the resulting embedded bitstream provides about the same performance as for the integer lattice. When the entire bitstream is fully decoded, the granular gain of the lattice is realized.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Source coding, a terminology due originally to Shannon's classic works in the development of information theory, is a basic process performed in digital communication systems. The purpose of source coding is to represent source samples in compact form. This can be achieved by identifying and using structure that exists in the data to eliminate redundant or irrelevant data. Source coding can be categorized into two classes: Lossless source coding, in which there is no loss of information and the original source data can be reconstructed perfectly from its compact form; and lossy source coding, which allows some difference between original and reconstructed data. In general, lossy coding gives higher compression than lossless coding. One form of lossy coding, known as vector quantization (VQ), is widely used in many modern applications. In this thesis, we focus on a variant of VQ called *lattice vector quantization (LVQ)*.

The idea of vector quantization was put forward by Shannon's classic work [1] and

further developed in [2], where he showed that for a given coding rate R and a distortion criterion D, the least distortion between original and reconstructed source data achievable by any quantizer is equal to a function $D(R)$, called the *distortion-rate function*[1]. The distortion-rate function is determined by the source characteristics and the distortion criterion, and this optimum performance bound can be approached by coding sequences of source data of arbitrarily large length. He called such a code a *source code subject to a fidelity criterion*, but a code of this type can also be called a *vector quantizer*, and the encoding process is called *vector quantization* [3]. Since then, vector quantization has been widely studied and developed for a variety of data compression applications.

A basic block diagram of vector quantization is shown in Figure 1.1. At the encoder, source data are grouped together as vectors and each vector is then compared with codevectors in a codebook to find the closest codevector with respect to a given distortion measure. Once the best codevector is found, its index is transmitted to a decoder and the codevector is reconstructed from a lookup table.

Early vector quantizer implementations used brute-force nearest neighbor encoding, which compares the source vector to be quantized to every codevector in the codebook. This type of implementation requires a large number of computations, with complexity that grows exponentially with the vector dimension. Structured codebooks were studied and developed in order to reduce the complexity required in VQ [4]. VQ can be designed

[1]The equivalent problem is the problem of minimizing rate subject to a distortion constraint. Shannon called the solution to this equivalent problem the *rate-distortion function*, denoted by $R(D)$.

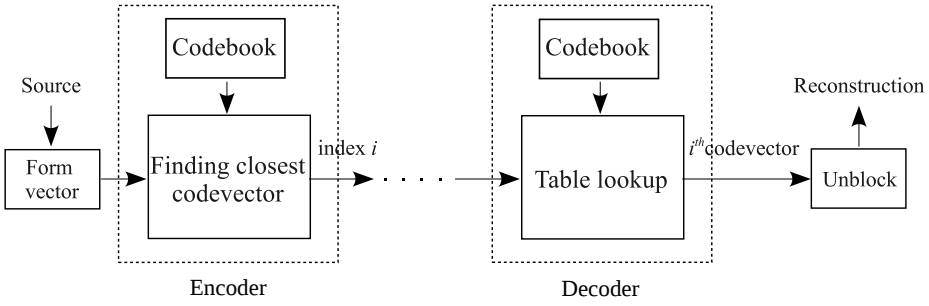


Figure 1.1: Basic block diagram of vector quantization

based on a clustering algorithm. A well-known work by Linde, Buzo, and Gray [5] is a example of the clustering approach based VQ, which extended Lloyd's algorithm [6] to vector quantization. Other vector quantization methods are based on various structures that trade off complexity for performance. Such VQ methods include lattice VQ [7], pyramid VQ [8], tree-structured VQ [3], trellis-coded VQ [9], etc. Entropy-constrained VQ was first studied in [10] and further developed in [11]. Recent developments in VQ combined the quantization with prediction or transformation to gain better performance. Such methods include code-excited linear prediction (CELP) [12], subband coding [13] , etc.

A lattice vector quantizer is a structured vector quantizer for which the codevectors in the codebook are formed as the scaled or translated subsets of regular lattices. The lattice codevectors are constrained to have form $\sum_{i=1}^{N} c_i \mathbf{u}_i$, where c_i is any integer and $\{\mathbf{u}\}_i$ is a set of linearly independent vectors that forms a basis of the lattice. A lattice Voronoi region is the set of all points closer to the origin than to any other lattice vector. Figure 1.2 shows an example of the Voronoi region of the 2-dimensional hexagonal lattice, A_2, and its basis, compared with that of the 2-dimensional integer lattice, $\mathbb{Z}_2$.

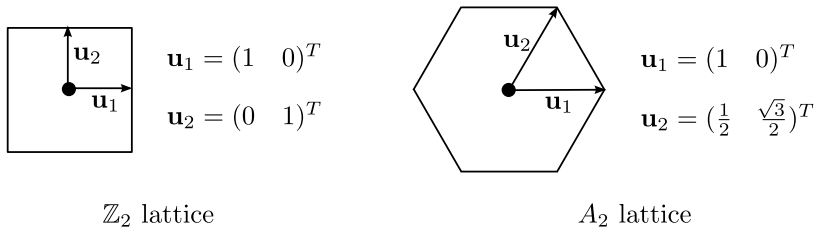


Figure 1.2: Voronoi regions of $\mathbb{Z}_2$ and A_2 lattices and their basis vectors.

Lattice VQ has several advantages such as low-complexity, less memory requirement, and simple implementation. There are also several algorithms for fast quantizing and encoding for LVQ using certain lattices [14], [15]. Lattice VQ was proposed by Gersho [16], using a high-rate quantization approximation [17]. Since then it has been intensively studied by many researchers. Many variants of lattice VQ have been proposed. Our interest is primarily in lattice spherical VQ (SVQ) [18] and lattice pyramid VQ (PVQ) [19]. Several modern wide-band audio coding standards such as [20] and [21] have adopted lattice SVQ as the core quantizer.

1.2 Dissertation Outline

The main purpose of this thesis is to study lattice quantization of audio and images. For audio sources, we focus on analyzing and modeling the performance of lattice quantization. First, the performance of nonzero pulse amplitude quantization implicit in algebraic codebook code-excited linear prediction (ACELP) is analyzed and shown that it is effective in a rate-distortion sense at a particular, commonly used encoding rate. Algebraic codevectors

have amplitude constrained to ±1 which can be considered as the points in the integer lattice. Next, a mathematical model is developed to describe the performance of lattice-based spherical VQ (LSVQ) used in discrete Fourier transform (DFT) based transform audio data compression. A Gaussian mixture model (GMM) is used to model both the marginal density and the block energy density of the transform audio coefficients, and a GMM-based rate-distortion function for LSVQ is derived. The GMM-based rate-distortion function can be used to model or estimate the expected performance of LSVQ. The GMM-based rate distortion function is shown to accurately model the empirical performance of a RE_8 lattice spherical VQ used in the AMR-WB+ wideband audio coding standard. Next, we consider use of lattice VQ in subband image or wavelet transformed image compression. A subband image coding algorithm based on LSVQ and LPVQ is proposed. The algorithm is motivated by the nonlinear energy dependence apparent in the block energy density of subband image coefficients, and by the observation that the block energy density can be modeled as a mixture of memoryless Gaussian or Laplacian distributions. Finally, using Forney's characterization of binary lattices [22], we introduce the notion of "lattice-defining" bit-planes, and develop methods for the efficient bit-plane coding of lattice codevectors, resulting in an embedded bitstream. We show that progressive decoding of bit-plane encoded codevectors from the D_4, RE_8, or Λ_{16} (Barnes-Wall) lattices provides similar performance to the standard integer (cubic) lattice, with the lattice granular gain achieved when the entire bit-stream is decoded.

A brief outline of the thesis is as follows. Chapter 2 discusses the non-zero pulse am-

plitude quantization implicit in ACELP and studies the problem of optimum pulse amplitude quantization. A basic block diagram of ACELP is introduced. A model for non-zero ACELP pulse amplitudes is described, and used in rate-distortion analysis. Simulations based on mean square error (MSE) and mean opinion score (MOS) compare the optimized pulse quantization to standard methods. In Chapter 3, the Gaussian mixture model and expectation-maximization (EM) algorithm are described. A rate-distortion function based on a GM model is developed. The effectiveness of the model is evaluated by comparing the GM-based rate-distortion function to the cubic lattice SVQ performance, and to the RE_8 lattice SVQ performance used in the AMR-WB+ audio coding standard [23]. In Chapter 4, a block-based image coding algorithm using quadtree partitioning is proposed. The empirical density of the block energy of subband coefficients is shown to display the nonlinear dependence between block subband coefficients. An alternative version of the proposed algorithm is also discussed. Then, the performance of the proposed algorithm using the cubic lattice is compared with that of the set partitioning in hierarchical tree (SPIHT) image coding algorithm [24]. In Chapter 5, the notion of lattice-defining bit-planes is introduced. Methods to encode the lattice-defining bit-planes for the D_4, RE_8, and Λ_{16} lattices are described. Then, simulation results for progressive decoding of bit-plane encoded D_4, RE_8, and Λ_{16} lattice codevectors are provided and compared with that of integer lattice codevectors. Using the D_4 lattice, the proposed method of encoding lattice-defining bit-planes is then used in the SPIHT algorithm, and the performance of the modified SPIHT image coding is compared with the original SPIHT image compression.

1.3 Summary of Research Contribution

This thesis makes the following contributions.

1. Analysis of ACELP method of non-zero pulse position selection of algebriac codebook search.
    - The selection implies a conditionally bimodal distribution on the selected non-zero pulse amplitudes.
    - For the MSE distortion measure the simple 1-bit uniform quantization of non-zero pulse amplitudes used in ACELP is optimum in a rate-distortion sense.
    - Small gains in SNR (no more than 1 dB) are possible for the ACELP method of nonzero pulse position selection by refined amplitude quantization, but such increase in SNR requires a significant increase in pulse amplitude quantization rate.
2. GMM-based rate-distortion function of LSVQ in transform audio coding.
    - A rate-distortion function based on GMM is developed and shown to accurately model the performance of LSVQ.
    - The GMM-based rate-distortion function with number of classes equal to 4 is sufficient to adequately model the LSVQ performance.
3. Block-based image coding algorithm for subband image compression.

- A block-adaptive lattice VQ for subband image coding is developed using LSVQ and LPVQ. The performance of the proposed coding algorithm using the cubic lattice is competitive with that of the SPIHT algorithm, with a little better performance at low and moderate encoding rates.

4. Bit-plane coding of lattice codevectors.

   - If lattice codevectors are represented in sign/magnitude form, only a few least significant bit-planes, called lattice-defining bit-planes, are constrained to some particular forms, which are unique for each lattice. Hence, any convenient bit-plane coding can be used to encode the more significant bit-planes, with the modification only required for coding lattice-defining bit-planes.

   - Simple methods are described to encode lattice-defining bit-planes for D_4, RE_8, and Λ_{16} lattices.

   - Simulations show that for lattice VQ of a uniform source, standard bit-plane coding together with the proposed methods to handle encoding of lattice-defining bit-planes gives performance about the same as that of a cubic lattice, when the encoded bit-stream is truncated, and the lattice granular gain is realized when the encoded bit-stream is fully decoded.

   - The SPIHT algorithm is modified to encode using D_4 lattice codevectors using the idea of lattice-defining bit-plane.

Chapter 2

Analysis of Amplitude Quantization in ACELP Excitation Coding[2]

Linear-prediction-based analysis-by-synthesis (LPAS) coding has been widely used in speech coding. The most important form of LPAS coding is code-excited linear prediction (CELP) introduced in [26], [12] which uses a predefined set of sequences as an excitation codebook. Several standards have adapted CELP as the core algorithm for speech coding [21]-[27].

In CELP speech coding, the encoder determines the linear prediction coefficients (LPC), quantizes the LPC parameters, and encodes them for transmission. The encoder also selects the excitation sequence as the sum of two quantized excitations: an adaptive codebook excitation and a fixed codebook excitation. The adaptive (or pitch) codebook excitation is determined conditioned on the LPC synthesis filter. The fixed codebook excitation is selected conditioned on both the synthesis filter and the adaptive codebook excitation.

Algebraic CELP (ACELP) [28] imposes a particular structure on the fixed codebook excitation. The codevectors are sparse with only m non-zero pulse positions allowed in a

[2]Portions of this chapter were presented at the 2010 Data Compression Conference [25].

sub-frame of length L. The L samples in a sub-frame are partitioned into K interleaved tracks of pulse positions, and the m total non-zero pulse positions are partitioned into m_k non-zero pulse positions in track k, for $k = 0, 1, \ldots, K-1$, with $m = \sum_{k=0}^{K-1} m_k$. The algebraic codebook is a product code of the form $g\boldsymbol{c}$ where g is a (quantized) sub-frame gain, and $\boldsymbol{c}$ is the sparse, L-dimensional codevector with m non-zero amplitudes restricted to the values ± 1. A suboptimum, but efficient, search algorithm is used to select the algebraic codevector (e.g., [21]).

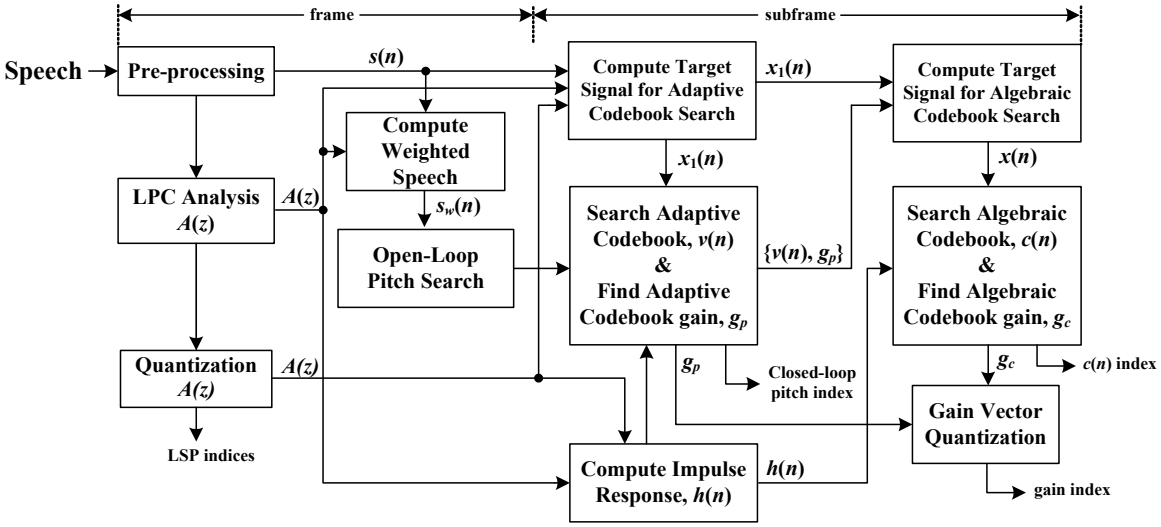
In [29] an enhancement layer coding method, for an 8 kbps base layer G.729 speech coder, is developed by refining the amplitude quantization of the base-layer ACELP pulses. The 1.6 kb/s refinement in [29] uses 8 bits per 5 ms subframe to refine the 4 non-zero pulse amplitudes. Vector quantization (VQ) is used to jointly quantize and encode the pulse amplitudes, with a total rate of 12 bits (4 bits for each ACELP sign bit and 8 bits for the VQ) or an average of 3 bits/pulse for amplitude quantization. The additional 2 bits/pulse refinement amplitude quantization leads to 0.05 to 0.08 increase in perceptual evaluation of speech quality (PESQ) mean opinion score (MOS). This relatively modest increase in MOS suggests that 1 bit/pulse (i.e., sign bit) amplitude quantization is already quite effective in quantization performance.

In this chapter we study the coding performance advantages possible by using the optimum ACELP codebook amplitudes compared to the ACELP codebook. We consider codevector amplitude quantization and the synthesized speech quality. Our formulation models the non-zero ACELP pulse positions as random variables with a bimodal distribu-

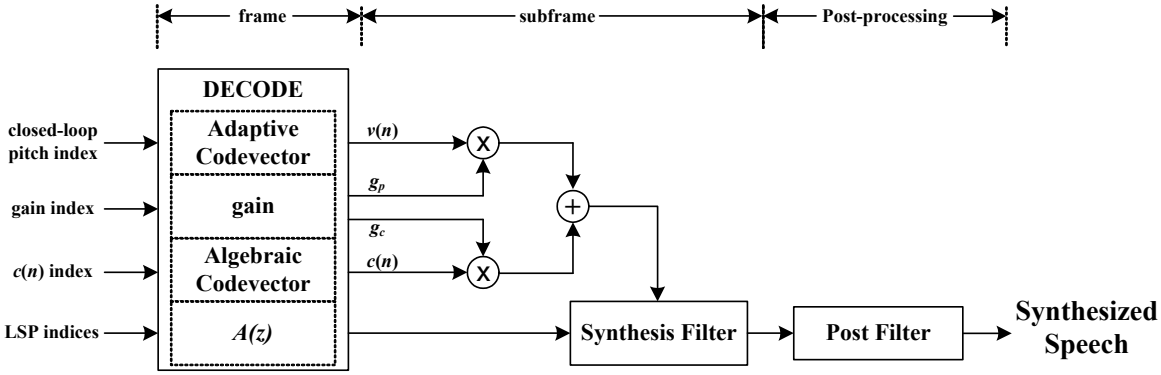
tion. Rate-distortion analysis of the quantization of such random variables indicates that at the (typical) ACELP encoding rate of 1 bits per amplitude and for the square error distortion measure, simple uniform scalar quantization is optimum. At rate larger than 1 bit per pulse amplitude, some increase in SNR is possible, but the increase in SNR yields only modest improvement in perceptual quality, as measured by mean opinion score.

2.1 Problem Statement

A block diagram of algebraic CELP is shown in Figure 2.1 based on the Adaptive Multi-rate (AMR) speech coding standard in [21], and it is used as the ACELP reference structure in this chapter. Each N-sample frame of speech is windowed and analyzed to determine the linear prediction filter, $A(z)$. The coefficients of $A(z)$ are represented as line spectral frequencies [30], quantized, and encoded subject to the operating bit rate, using split-multistage vector quantization [3]. The quantized version of $A(z)$, denoted by $\hat{A}(z)$, is also reconstructed for later processes. Then, each N-sample frame is further divided into four sub-frames of L samples, (e.g., $N = 160$ and $L = 40$ in [21]). Each L-sample sub-frame, denoted by $\boldsymbol{s}$, is filtered using a weighting filter formed from $A(z)$ to produce $\boldsymbol{s}_w$, which is used in open-loop pitch analysis to estimate the open-loop pitch lag. Then, excitation sequence selection is performed on a sub-frame basis. The excitation sequence consists of two parts: An adaptive (pitch) codebook vector and an algebraic codebook vector (also called an innovation vector). The synthesized speech signal is constructed from these two



(a) Analysis



(b) Synthesis

Figure 2.1: Block diagram of Algebraic CELP.

excitations and quantized gains as shown in Figure 2.1(b).

To determine the excitation codevectors, first, the impulse response of the weighted synthesis filter, $h(n)$, is computed from $H(z)W(z) = A(z/\gamma_1)H_{de-emph}(z)/\hat{A}(z)$, where a weighting filter $W(z) = A(z/\gamma_1)H_{de-emph}(z)$, $H_{de-emph} = 1/(1 - 0.68z^{-1})$, and γ_1 is a perceptual weighting factor equal to 0.92. Then, a target signal for adaptive codebook search, denoted by $\boldsymbol{x}_1$, is computed by subtracting the zero input response of the weighted synthe-

sis filter, $h(n)$, from the filtered signal $\boldsymbol{s}_w$. Next, an adaptive codebook search is performed. It consists of closed loop pitch search, adaptive codebook selection, and adaptive codebook gain computation. A closed-loop pitch search is performed around the open-loop pitch lag to estimate the fractional pitch lag. The adaptive codebook vector and adaptive codebook gain, denoted by $\boldsymbol{v}$ and g_p, respectively, are computed based on the estimated fractional pitch lag, and they are then used to compute the target signal for the algebraic codebook search. More specifically, a target signal for algebraic codebook search is obtained by subtracting the contribution from the adaptive codebook vector weighted by the adaptive codebook gain from the target signal used in the closed-loop pitch search, $\boldsymbol{x}_1$, yielding the L-dimensional (sub-frame) target signal, $\boldsymbol{x}$. That is, $\boldsymbol{x} = \boldsymbol{x}_1 - g_p\boldsymbol{y}$, where $\boldsymbol{y} = \boldsymbol{v} * \boldsymbol{h}$.

The algebraic codebook is searched to minimize the mean-square error between the weighted input speech target signal, $\boldsymbol{x}$, and the filtered algebraic codevector. More specifically, the algebraic codebook is searched to minimize

$$\|\boldsymbol{x} - g_c H\boldsymbol{c}\|^2 \tag{2.1}$$

where H is a lower triangular Toeplitz matrix with diagonal $h(0)$ and lower diagonal $h(1)$, $h(2), \ldots, h(L-1)$, with $h(n)$ the impulse response of the weighted synthesis filter, g_c is the algebraic codevector gain, and $\boldsymbol{c}$ is the algebraic codevector with m non-zero pulse positions. Minimizing over the choice of gain, yields the optimum gain as $g_{c,opt} = \frac{\boldsymbol{x}^T H\boldsymbol{c}}{\boldsymbol{c}^T H^T H\boldsymbol{c}}$. Substituting this gain into (1) and manipulating indicates that the optimum algebraic code-

vector must maximize the ratio

$$B_k = \frac{(\boldsymbol{d}^T \boldsymbol{c}_k)^2}{\boldsymbol{c}_k^T H^T H \boldsymbol{c}_k} \tag{2.2}$$

where $\boldsymbol{d} = H^T \boldsymbol{x}$, and $\boldsymbol{c}_k$ is the k^{th} algebraic codevector [21].

In the AMR encoding, the m non-zero pulse positions and their signs are found using a non-exhaustive analysis-by-synthesis search technique. The positions of the non-zero pulse amplitudes and their signs are encoded and transmitted to the decoder together with the quantized codevector gain.

2.2 Excitation Pulse Modeling

Since the non-zero pulse amplitudes in the algebraic codevector are constrained to have integer value of ±1, the normalized amplitudes can be viewed as being quantized using a 1-bit scalar quantizer. The implicit quantization of these amplitudes is investigated, as follows. Let $\boldsymbol{c}$ be an algebraic codevector of dimension L obtained from the ACELP algebraic codebook, with $\boldsymbol{x}$ the target signal for the algebraic codebook search. Assume that m non-zero pulses are allowed per sub-frame and they are selected according to the track structure using the search algorithm in [21]. The sub-frame mean-square error (MSE) is then $\|\boldsymbol{x} - g_c H \boldsymbol{c}\|^2$, where $\boldsymbol{c}$ is sparse (at most m non-zero positions in the L-dimensional codevector). If the sub-frame gain, g_c, is absorbed into the codevector amplitudes, then consider the problem of selecting the optimum non-zero pulse amplitude $\|\boldsymbol{x} - H\boldsymbol{c}\|^2$. Denote the m non-zero positions in the codevector as $0 \leq i_0 < i_1 < \cdots < i_{m-1} < L$. Since

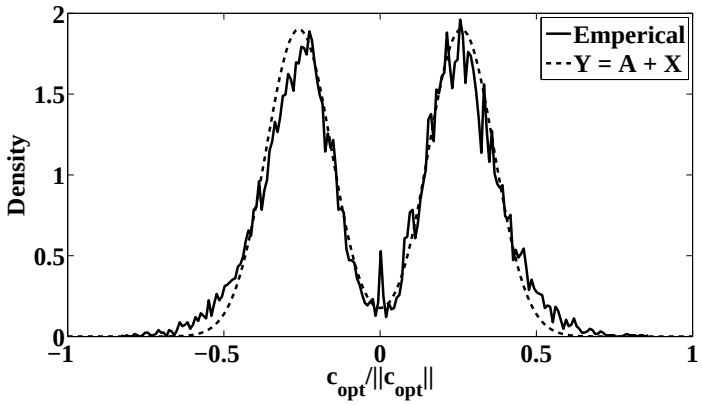


Figure 2.2: Probability density function of Y based on (2.3)

the remaining $L - m$ dimensions of $\boldsymbol{c}$ are zero, it follows that $H\boldsymbol{c} = \tilde{H}\tilde{\boldsymbol{c}}$, where $\tilde{c}_j = c_{i_j}$, $j = 0, \ldots, m-1$ and $\tilde{H} = [\boldsymbol{h}_{i_0}\boldsymbol{h}_{i_1} \cdots \boldsymbol{h}_{i_{m-1}}]$, with $\tilde{H}$ an L row by m column matrix with $\boldsymbol{h}_{i_j}$ the i_j^{th} column of H. Minimizing $\|\boldsymbol{x} - H\boldsymbol{c}\|^2 = \left\|\boldsymbol{x} - \tilde{H}\tilde{\boldsymbol{c}}\right\|^2$ yields the optimum (unquantized) non-zero pulse amplitude as $\tilde{\boldsymbol{c}}_{opt} = (\tilde{H}^T\tilde{H})^{-1}\tilde{H}^T\boldsymbol{x}$. The empirical density of the amplitudes of $\tilde{\boldsymbol{c}}_{opt} / \left\|\tilde{\boldsymbol{c}}_{opt}\right\|$ is shown in Figure 2.2 (for the AMR 12.2 kbps mode with $m = 10$ non-zero pulses per sub-frame of 40 samples). Clearly, the density is zero mean, symmetric, and bimodal.

A simple model for the bimodal optimum non-zero pulse amplitudes $\tilde{\boldsymbol{c}}_{opt}$ is

$$Y = A + X \tag{2.3}$$

where A is a discrete, binary random variable with equiprobable amplitude $\pm a$ and X is a zero-mean, continuous Gaussian random variable with variance σ_X^2, independent of A. An

alternative model might be

$$Y = A + sign(A)X \tag{2.4}$$

where $sign(A)$ is the sign of the random variable A. Figure 2.2 also shows the probability density function of Y based on (2.3) when X is a zero-mean Gaussian random variable, with the parameters a and σ_X^2 selected to model the bimodal density of the non-zero pulse amplitudes. Use of (2.4) allows more flexible modeling of the non-zero pulse amplitudes, but for the present purposes the model in (3) is adequate.

2.3 Rate-Distortion Analysis

Since (2.3) provides a reasonable model for the empirical density of the normalized optimum pulse amplitudes in $\tilde{\mathbf{c}}_{opt}$, it is used as a model for rate-distortion analysis of the direct quantization of $\tilde{\mathbf{c}}_{opt}$. Consider the problem of minimum MSE quantization of Y given by (2.3). The Shannon lower-bound to the rate-distortion function is given by [31]

$$R_{SLB,Y}(D) = h(Y) - 0.5\log_2(2\pi eD) \tag{2.5}$$

where $h(Y)$ is the differential entropy of Y. From (2.3) and the convexity of the logarithm function, it follows that

$$h(Y) \leq H(A) + h(X) = 1 + h(X) \tag{2.6}$$

where $H(A)$ is the entropy of A, which is equal to 1 since A is a binary random variable with equi-probable outcomes. Substituting (2.6) into (2.5) yields

$$R_{SLB,Y}(D) \leq 1 + h(X) - 0.5\log_2(2\pi eD) \leq 1 + R_{SLB,X}(D) \tag{2.7}$$

where $R_{SLB,X}(D)$ is the Shannon lower-bound to the rate-distortion function of X. As a becomes large, equality in (2.7) is approached. For that case, $R_{SLB,Y}(D)$ is well-approximated as $R_{SLB,Y}(D) \approx 1 + R_{SLB,X}(D)$. Assuming that a is large enough, solving for the distortion then yields

$$D_{SLB,Y}(R) = D_{SLB,X}(R-1). \tag{2.8}$$

In ACELP coding, the non-zero pulse amplitudes are quantized at a rate of 1 bit/sample as the amplitudes ±1, with the codevector scaled by a quantized gain. This can be thought of as 1 bit/dimension scalar quantization of the m normalized pulse amplitudes, with the normalization factor represented as the quantized codevector gain. From (2.8), it can be seen that at rate $R = 1$, the smallest distortion to be expected for direct quantization of Y is the variance of X. Moreover, this distortion is achievable by using simple 1-bit scalar quantization (with reproduction levels $\hat{Y} = \pm E[Y|Y > 0]$). Rate-distortion theory thus indicates that it is not possible to get better coding performance than this at $R = 1$ bit/sample. Certainly at encoding rates larger than 1 bit/sample vector quantization of Y may provide smaller MSE than scalar quantization. But for parameter a large enough and at the encoding rate of 1 bit/sample, direct scalar quantization of Y is optimum in a rate-distortion

sense. Hence, no improvement is to be expected even if one uses more elaborate quantization methods such as vector quantization.

In extrapolating from the analysis above to ACELP coding, two qualifications should be noted. First, in ACELP speech coding the objective is not to directly quantize the non-zero pulse amplitudes to minimize MSE, but rather to minimize the weighted MSE in (2.1). Second, if there is dependence between the dimensions of $\tilde{\boldsymbol{c}}_{opt}$, then vector quantization or joint entropy coding may offer improvement. However, we will demonstrate in the next section that for 1 bit/pulse amplitude quantization, correlation between sign bits is small and joint coding of the sign bits offers negligible improvement in coding efficiency.

2.4 Two-State Markov Chain Modeling of Sign-bit Correlation

Suppose that the algebraic codevector pulse positions are determined as in [21], and the amplitudes represented as ± 1. In this section, we model the sign-bit correlation using a two-state Markov model [31]. Then, the entropy rate for the Markov model provides an estimate of the smallest encoding rate necessary to encode the algebraic codevector sign-bit information.

The correlation matrix of the optimal algebraic codevector sign-bit information is estimated as

$$\Lambda = \frac{1}{N} \sum_{n=1}^{N} \tilde{\boldsymbol{s}}(n) \tilde{\boldsymbol{s}}^T(n), \tag{2.9}$$

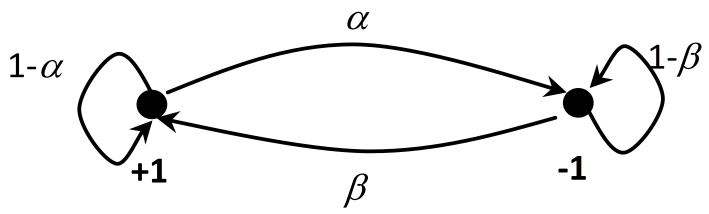


Figure 2.3: Two-state Markov chain model

where $\tilde{\mathbf{s}}(n) = (\tilde{s}_0(n), \tilde{s}_1(n), \ldots, \tilde{s}_{m-1}(n))^T = sign(\tilde{\mathbf{c}}_{opt}(n))$, $\tilde{\mathbf{c}}_{opt}(n)$ is the vector of optimal non-zero pulse amplitudes for subframe n, and N is the total number of subframes.

Denote the first-order correlation of non-zero pulse position i as $\Lambda_1(i)$, estimated as

$$\Lambda_1(i) = \frac{1}{N}\sum_{n=1}^{N} \tilde{s}_i(n)\tilde{s}_{i+1}(n), \tag{2.10}$$

with backward and forward first-order correlation the same. Since there are two possible sign values, +1 or −1, we can model the event of sign value change between adjacent positions of $\tilde{\mathbf{c}}_{opt}$ using a two-state Markov chain. The model is shown in Figure 2.3, where $\alpha = \Pr(s_{i+1} = -1 | s_i = +1)$ is the probability that given sign value +1 at pulse position i, the sign value of the next pulse position $i + 1$ is −1, and $\beta = \Pr(s_{i+1} = +1 | s_i = -1)$. Averaging over all subframes, the first-order correlation is empirically estimated to be −0.05 (for $m = 10$ non-zero pulse positions per subframe and the AMR 12.2 kbps encoding mode).

Consider the first-order correlation

$$\Lambda_1(k) = \frac{1}{N}\sum_{n=1}^{N} \tilde{s}_i(k)\tilde{s}_{i+1}(k) = \frac{1}{N}\left[\sum_{n=1}^{N_{+1}}(1) + \sum_{n=1}^{N_{-1}}(-1)\right]$$
$$= \frac{1}{N}\left[N_{+1} - N_{-1}\right] = P_{+1} - P_{-1} = -0.05, \tag{2.11}$$

where P_{+1} and N_{+1} are the estimated probability and number of occurrences of the event that the sign values of pulse position k and $k+1$ are identical, and P_{-1} and N_{-1} are similarly defined. Using the fact that $P_{+1} + P_{-1} = 1$ and (2.11) yields $P_{+1} = 0.475$ and $P_{-1} = 0.525$. From Figure 2.2, it is reasonable to assume that $\Pr(s_k = -1, s_{k+1} = -1) = \Pr(s_k = +1, s_{k+1} = +1)$, and $\Pr(s_k = +1, s_{k+1} = -1) = \Pr(s_k = -1, s_{k+1} = +1)$. Then, solving for α and β, we obtain $\alpha = 0.525 = \beta$. From [31], the entropy rate of the two-state Markov model is

$$H^* = \frac{\beta}{\alpha+\beta}H_b(\alpha) + \frac{\alpha}{\alpha+\beta}H_b(\beta), \tag{2.12}$$

where $H_b(.)$ is the binary entropy function. Hence, for α and $\beta = 0.525$, the entropy rate is $H^* = 0.99$ bit/sample. Since this is almost the same as the 1 bit/sign-bit encoding rate used in ACELP, it implies that no significant improvement in coding efficiency is feasible.

2.5 Simulation Results

In this section, we will compare ACELP using optimum (unquantized) pulse amplitude codevectors in section 2.2 with that using original algebraic codevectors. The simula-

tion is performed using ACELP with the 12.2 kbps mode in [21]. In this mode, each subframe of 40 samples is partitioned into 5 interleaved tracks, with two non-zero pulse positions allowed in each track. The optimum pulse amplitudes are computed, subject to the restriction that non-zero pulse positions must be the same as those obtained from the ACELP algorithm. Then, the original algebraic codevectors are replaced with the optimum (unquantized) pulse amplitude codevectors. Two performance comparisons are presented: weighted mean-square error as defined in (2.1); and mean opinion score (MOS) [32]. The results are shown in Tables 2.1 and 2.2, where MSE_{opt} is the average normalized MSE when optimum unquantized codevectors amplitudes are used and MSE_{ACELP} is the average normalized MSE using quantized ACELP fixed codevectors. The average normalized MSE is defined as

$$MSE_{Normalize} = \frac{1}{N}\sum_{n=1}^{N}\frac{\|\boldsymbol{x}_n - g_n H_n \boldsymbol{c}_n\|^2}{\|\boldsymbol{x}_n\|^2} \tag{2.13}$$

where N is the total number of subframe.

It can be seen that using optimum (unquantized) codevector amplitudes results in smaller MSE, but this does not have significant effect in the MOS sense. In other words, the MOS results imply that the segment or part of speech improved by using the optimum codevector might not be influential in human hearing, and thus the speech quality perceived by the listener might be almost the same as that using the ACELP fixed codevector.

In [29] enhancement layer coding is studied using refinement quantization of the four non-zero ACELP base-layer pulses, at an effective rate of 2 bits/pulse (in addition to the 1

Table 2.1: Average normalized MSE

Speech	$\boldsymbol{MSE_{ACELP}}$	$\boldsymbol{MSE_{opt}}$
Male1	0.72	0.61
Male2	0.55	0.50
Female1	0.44	0.36
Female2	0.41	0.21

Table 2.2: Mean opinion score

Speech	$\boldsymbol{MOS_{ACELP}}$	$\boldsymbol{MOS_{opt}}$
Male1	3.96	3.96
Male2	3.89	3.92
Female1	3.77	3.81
Female2	4.18	4.21

bit/pulse base-layer encoding rate). The small increase in MOS reported is consistent with the results in table 2 for the unquantized optimum non-zero amplitudes.

Chapter 3

Gaussian Mixture Modeling of Transform Audio Coding[3]

3.1 Introduction

Transform coding is an effective approach to audio coding. Such transformations include the discrete Fourier transform (DFT) [23], [36], the discrete cosine and modified discrete cosine transforms, and subband decomposition [37]. One recent example is DFT-based transform coded excitation (TCX) used in the adaptive multi-rate wideband audio coding algorithm [23]. The transform coding consists of three steps. First, the data sequence is divided into frames of size N and then a given transformation is performed on each frame. The second step is quantizing the transformed sequence subject to a fixed rate per frame constraint. The final step is encoding the quantized transformed sequence into a binary bitstream [38].

Spherical vector quantization (SVQ) has been shown to be an efficient way to quantize audio transform data [18], [39] and has been used in an audio coding standard [23], [36].

[3]Portions of this chapter were published in [33], [34], and [35].

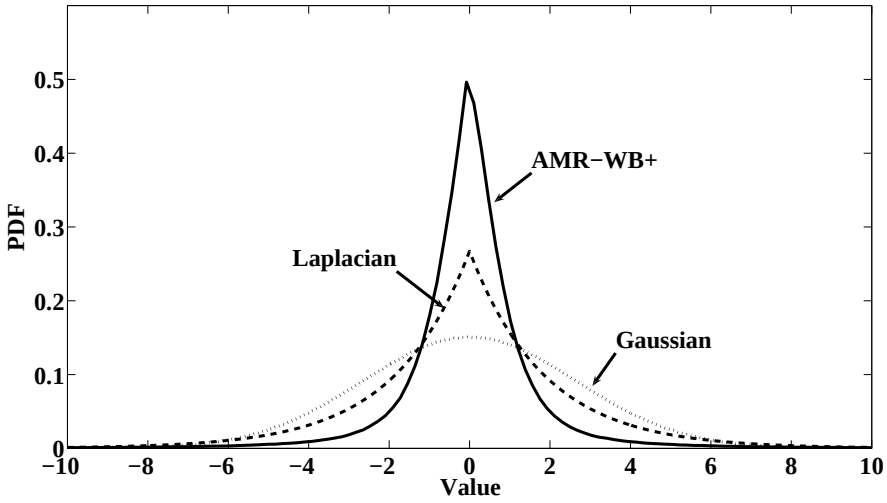


Figure 3.1: Empirical density of transform coefficients compared to that of memoryless Gaussian and Laplacian random variables with the same mean and variance.

SVQ can be structured as a type of multi-rate, classified vector quantizer [40] that uses a product code to encode lattice codevectors as binary codewords. The product code consists of 1) a code for representing the codevector energy (squared radius), and 2) a code for representing a lattice codevector, conditioned on the codevector energy. The product code partitions the lattice codevectors into concentric "shells" of codevectors. SVQ construction is motivated by the spherical geometry of the high probability volume of a memoryless Gaussian source probability density function [41], [42]. The lattice SVQ in [18], [39] is relatively simple to implement and remarkably effective, with an observed operational rate-distortion performance (for encoding audio transform data) significantly better than the memoryless Gaussian rate-distortion function.

As shown in Figure 3.1, the audio transform coefficients are reasonably well mod-

eled as having marginal Gaussian or Laplacian densities [39], [18]. However, memoryless Gaussian and Laplacian rate-distortion functions [3] are poor estimators of actual SVQ performance in transform audio coding, as will be shown later in this chapter. This is due to the strong energy dependence in transform audio coefficient data. This can be seen in Figure 3.2, which compares the empirical probability density function (pdf) of audio transform coefficient block squared radius (the block energy) to the block squared radius of memoryless Gaussian data, and Laplacian data, for block sizes L = 4, 8, 16, and 32. Clearly, for every block size the empirical density of the transform audio block squared radius differs significantly from that of memoryless Gaussian or Laplacian data of the same mean and variance. Since the correlation between coefficients is small, the empirical density is indicative of non-linear (energy) dependence in the transform coefficient data.

A Gaussian mixture model (GMM) [43], [44] is used in this chapter to model vectors of audio transform coefficients. It is shown in [45] that any continuous pdf can be approximated by a Gaussian mixture density. A GM model has been successfully employed in several areas, e.g., speech recognization [46], speech coding [47]-[48], image coding [49], etc. In [50], a generalized Gaussian (GG) mixture model is used to model image subband coefficients. A lattice VQ encodes lattice codevectors partitioned in "shells" matched to the GG shape parameters [51]. The experimental results in [50] focus on a mixture of Laplacian densities and report GG mixture modeling average relative mean squared error distortion within 6 to 20 percent of empirical rate-distortion LVQ performance, with maximum relative error as large as 32 percent.

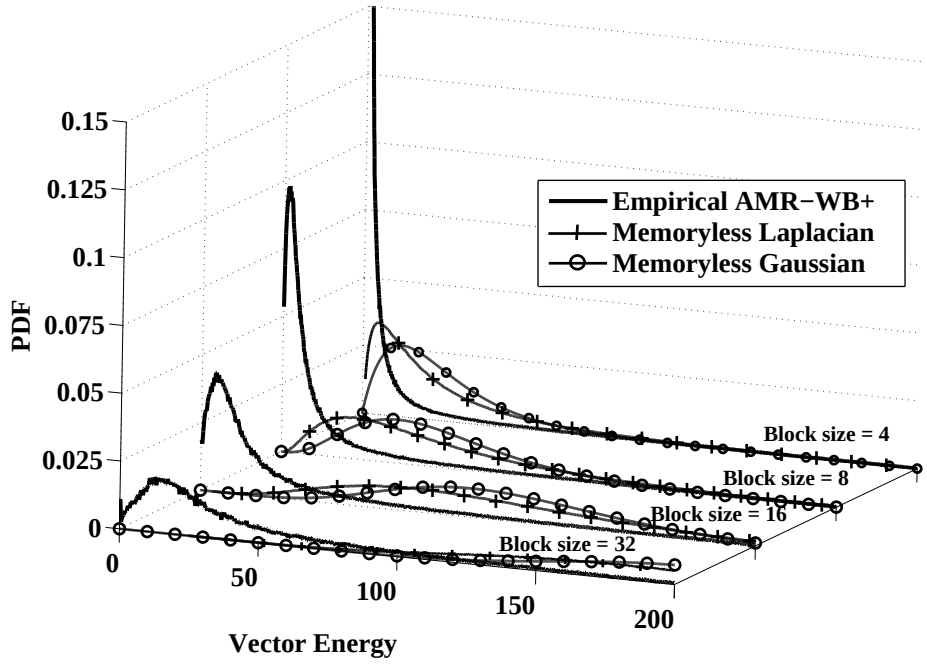


Figure 3.2: Empirical density of transform coefficient vector energy compared to that of memoryless Gaussian and Laplacian vectors.

In this chapter, we focus on modeling the performance of lattice spherical vector quantization (SVQ) in transform audio coding. This is done in two steps. First, we model the transform audio coefficient data using a Gaussian mixture model. Then, a rate-distortion function based on the GM model is developed and used to estimate the performance of SVQ. GM model parameters are estimated using the Expectation-Maximization (EM) algorithm [43], [44], [52] and two alternative methods to estimate model parameters are proposed. As an application example of the proposed method, the model developed is shown to accurately describe the RE_8 lattice SVQ performance used in [23].

3.2 Gaussian Mixture Model

A K-class Gaussian mixture pdf for L-dimensional random vector $\boldsymbol{U}$ is a parameterized function of the form

$$f_{mix}(\boldsymbol{u}|\boldsymbol{\Theta}) = \sum_{k=1}^{K} P(k) f_{\mathrm{U}|\boldsymbol{\Theta}}(\boldsymbol{u}|\theta_k) \tag{3.1}$$

where $P(k)$ denotes the prior probability or the probability that $\boldsymbol{U}$ is generated by the k^{th} class, and the component distribution, $f_{\mathrm{U}|\boldsymbol{\Theta}}(\boldsymbol{u}|\theta_k)$, is a multivariate Gaussian distribution defined as

$$f_{\mathrm{U}|\boldsymbol{\Theta}}(\boldsymbol{u}|\theta_k) = \frac{1}{(2\pi)^{L/2}|\boldsymbol{C_k}|^{1/2}} \mathrm{e}^{\{-\frac{1}{2}(\boldsymbol{x}-\mu_k)^T \boldsymbol{C}_k^{-1}(\boldsymbol{x}-\mu_k)\}} \tag{3.2}$$

where μ_k and $\boldsymbol{C}_k$ are the mean vector and covariance matrix of the k^{th} class, respectively. The mixture model's parameters are defined as the set $\boldsymbol{\Theta} = \{P(1), \ldots, P(K), \theta_1, \ldots, \theta_K\}$, where $\theta_k = \{\mu_k, \boldsymbol{C}_k\}$, for $k = 1, \ldots, K$.

3.2.1 Gaussian Mixture Model for Audio Transform Coefficients

Let X be a real-valued sequence of length N to be quantized and encoded, formed from consecutive transform coefficients. Assuming L divides N, partition X into N/L real-valued vectors (blocks) of size L, denoted as $\boldsymbol{Y}$. The transformation is assumed to remove the linear dependence in X, and hence also in $\boldsymbol{Y}$. Also, it is clear from Fig. 3.1 that X has zero mean. We further assume a stationary property in each component class. So, now $\theta_k = \{0, \sigma_k^2\}$. Therefore, a K-class, L-dimensional Gaussian mixture model in (3.1) for vector Y reduces

to

$$f_{mix}(\boldsymbol{y}|\boldsymbol{\Theta}) = \sum_{k=1}^{K} P(k) f_{\mathbf{Y}|\sigma_k^2}(\boldsymbol{y}|\sigma_k^2) \tag{3.3}$$

where $f_{\mathbf{Y}|\sigma_k^2}(\boldsymbol{y}|\sigma_k^2) = \frac{1}{(2\pi\sigma_k^2)^{L/2}} \exp(-\frac{1}{2\sigma_k^2} \sum_{l=1}^{L} y_l^2)$.

An alternative way to define the mixture model for audio transform coefficients is based on block or vector energy of $\boldsymbol{Y}$. Let the normalized energy be $Z = \frac{\epsilon}{\sigma_X^2}$, where $\epsilon = \sum_{l=1}^{L} y_l^2$ is the block energy (square radius) of the vector $\boldsymbol{Y}$, and σ_X^2 is the variance of X. Suppose $\boldsymbol{Y}$ is generated from the k^{th} class. Write the block energy as $Z = \frac{\sigma_k^2}{\sigma_X^2} \frac{\epsilon}{\sigma_k^2} = w_k Z_k$, where $w_k = \frac{\sigma_k^2}{\sigma_X^2}$ and $Z_k = \frac{\epsilon}{\sigma_k^2}$. The components of $\boldsymbol{Y}$ are independent and identically distributed and thus Z_k is Chi-square distributed [53]. Then, the mixture model of block energy Z can be expressed as

$$f_{mix}(z|\boldsymbol{\Theta}) = \sum_{k=1}^{K} P(k) f_{Z|\sigma_k^2}(z|\sigma_k^2) \tag{3.4}$$

where $f_{Z|\sigma_k^2}(z|\sigma_k^2) = \frac{1}{w_k} \hat{f}\left(\frac{z}{w_k}\right)$ and $\hat{f}(z)$ is Chi-square distributed with degree of freedom L, defined by $\hat{f}(z) = \frac{1}{2^{L/2}\Gamma(L/2)} z^{L/2-1} e^{-z/2}, z \geq 0$.

The model parameters $P(k)$ and σ_k^2, for $k = 1, \ldots, K$, can be estimated by several methods such as Expectation-Maximization (EM) algorithm [43]-[44], Markov-chain Monte Carlo algorithm [52], and Lloy clustering procedure [54]. In this paper, the EM algorithm is used since it is an algorithm widely used for finite mixture modeling.

Let $\boldsymbol{y}_m$ denote the m^{th} block of M total blocks and let $y_{m,l}$ denote the l^{th} component of $\boldsymbol{y}_m$, for $l = 1, \ldots, L$. From [44], the EM algorithm requires introduction of auxiliary variables, $w_{m,k}$, that represent how likely block $\boldsymbol{y}_m$ is generated by the k^{th} class, for blocks

$m = 1, \ldots, M$ and classes $k = 1, \ldots, K$. From [44] and (3.3), the expectation and maximization steps of the EM algorithm are as follows.

E-Steps

$$E[w_{m,k}] = \frac{f_{\mathbf{Y}|\sigma_k^2}(\boldsymbol{y}_m|\sigma_k^2)P(k)}{\sum_{j=1}^{K} f_{\mathbf{Y}|\sigma_j^2}(\boldsymbol{y}_m|\sigma_j^2)P(j)} \tag{3.5}$$

for $m = 1, \ldots, M$ and $k = 1, \ldots, K$.

M-Steps

$$\sigma_k^2 = \frac{\sum_{m=1}^{M} E[w_{m,k}](\frac{1}{L}\sum_{l=1}^{L} y_{m,l}^2)}{\sum_{m=1}^{M} E[w_{m,k}]} \tag{3.6}$$

for $k = 1, \ldots, K$, where $E[\cdot\cdot]$ denotes expectation. Alternatively, let z_m be the normalized block energy of $\boldsymbol{y}_m$, for $m = 1, \ldots, M$. Similar to above, by observing normalized block energy, the E-M steps for estimating the model parameters of the mixture model in (3.4) are as follows.

E-Steps

$$E[w_{m,k}] = \frac{f_{z|\sigma_k^2}(z_m|\sigma_k^2)P(k)}{\sum_{j=1}^{K} f_{z|\sigma_j^2}(z_m|\sigma_j^2)P(j)} \tag{3.7}$$

for $m = 1, \ldots, M$ and $k = 1, \ldots, K$.

M-Steps

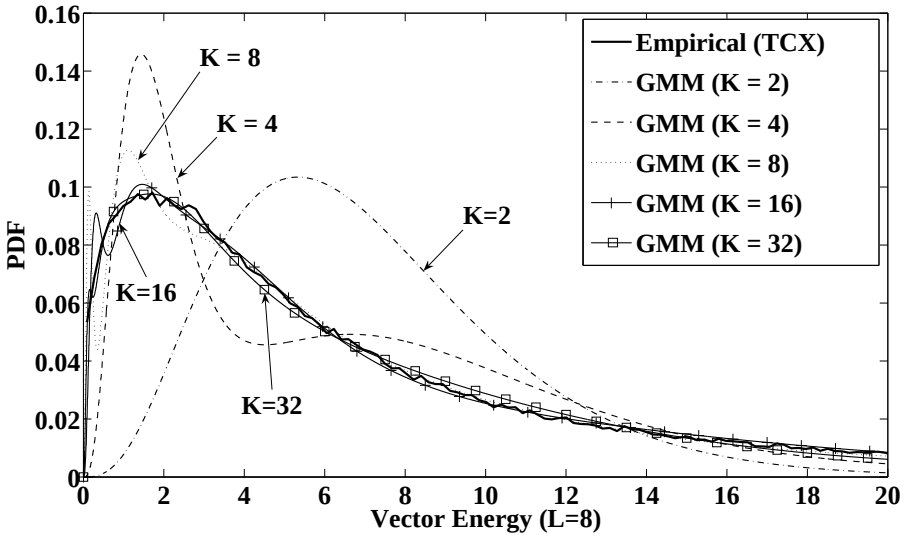
$$\sigma_k^2 = \frac{\sum_{m=1}^{M} E[w_{m,k}] z_m \sigma_X^2}{k \sum_{m=1}^{M} E[w_{m,k}]}. \tag{3.8}$$

The EM algorithm is used to estimate the mixture model parameters using transform coefficient vectors computed from a database of two minutes of wideband audio, 20% speech (two male and two female talkers) and 80% music (from nine different recordings). The resulting GMM energy density is compared to the empirical density of the transform coefficient vector energy in Figure 3.3 for K = 2, 4, 8, 16, and 32 classes. The transform coefficient marginal density of the GMM is also compared to the empirical density in Figure 3.4. It is clear that as the number of classes increases, the mixture models from both methods provide good approximations to both the vector energy density and the marginal density. However, one empirical observation is that GMM parameters based on (3.7) and (3.8) converge faster than those based on (3.5) and (3.6).

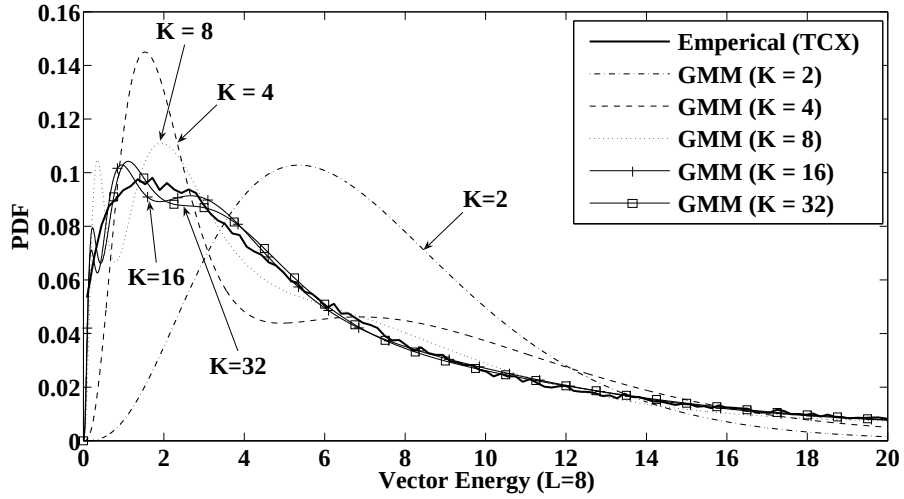
3.3 Rate Distortion Function based on Gaussian Mixture Model

For small mean-squared error (MSE) distortion, D, the rate-distortion performance of entropy-coded quantization of a memoryless Gaussian source is

$$R(D) = \frac{1}{2} \log_2 \left(\frac{2\pi e}{12\beta} \cdot \frac{\sigma^2}{D} \right) \tag{3.9}$$

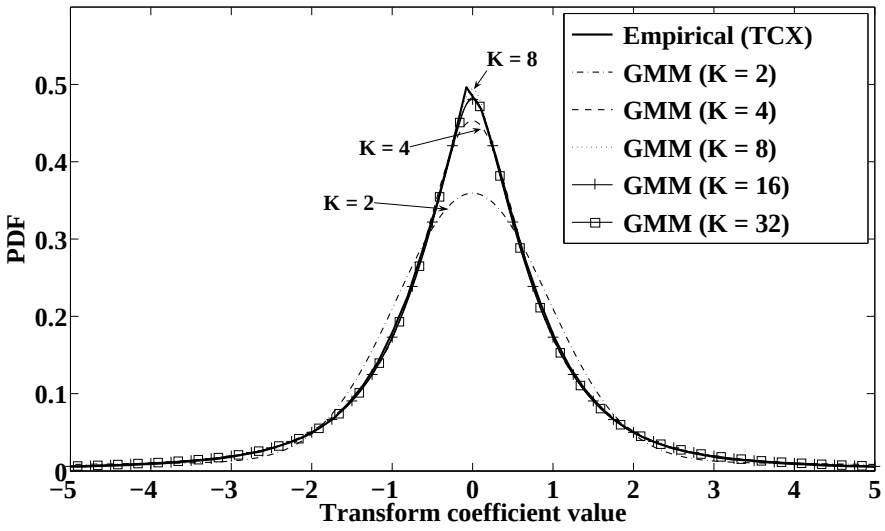


(a)

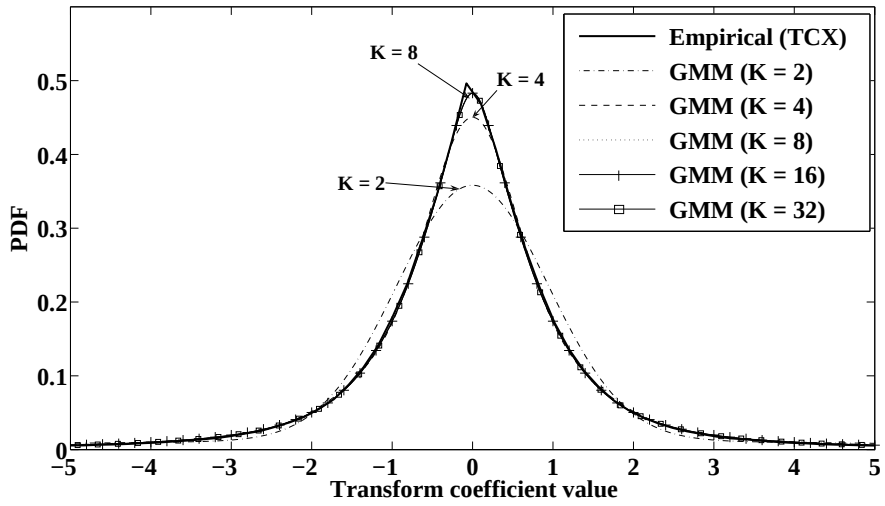


(b)

Figure 3.3: The Gaussian mixture model of transform coefficient block square radius compared to the empirical density ($L = 8$): (a) Model based on (3.5) and (3.6); (b) Model based on (3.7) and (3.8)



(a)



(b)

Figure 3.4: The Gaussian mixture model marginal density of transform coefficients compared to the empirical density ($L = 8$): (a) Model based on (3.5) and (3.6); (b) Model based on (3.7) and (3.8);

where σ^2 is the source variance and $1 \leq \beta \leq (2\pi e/12)$ reflects the *granular gain*, also called the *space filling advantage* [55], of the quantization method. For uniform quantization, $\beta = 1$ and (3.9) is the Gish-Pierce asymptote [56]. For vector quantization using the E_8 lattice [7], [57], $\beta \approx 1.16$ (or 0.65 dB). For $\beta = 2\pi e/12$, (3.9) is the rate-distortion function for the memoryless Gaussian source [31].

Now, consider a K-class Gaussian mixture source model of vector (block) length L. Each class is modeled to have block components that are independent and identically distributed (i.i.d), as mention in Section 3.2. Using (3.9), the rate-distortion function for quantization and encoding the k^{th} class can be expressed as

$$R_k(D) = \frac{1}{2}\log_2\left(\frac{2\pi e}{12\beta}\cdot\frac{\sigma_k^2}{D}\right) \tag{3.10}$$

where σ_k^2 is the k^{th} class variance and D is assumed small compared to σ_k^2. One coding strategy is to first classify a source vector and then quantize and encode that vector conditioned on the class. Additional rate is necessary to specify the block class. As we assume an i.i.d sequence of source blocks, the minimum rate for encoding the class is the entropy, $H(k) = -\sum_{k=1}^{K} P(k)\log_2 P(k)$ bits/block, where $P(k)$ is the probability of class k. The aver-

age encoding rate for classification-based quantization and encoding is thus modeled as

$$R_{GMM}(D) = \frac{1}{L}H(K) + \sum_{k=1}^{K} P(k)\frac{1}{2}\log_2\left(\frac{2\pi e}{12\beta}\cdot\frac{\sigma_k^2}{D}\right) \tag{3.11}$$

$$= \frac{1}{L}H(K) + \frac{1}{2}\log_2\left(\frac{2\pi e}{12\beta D}\prod_{k=1}^{K}(\sigma_k^2)^{P(k)}\right). \tag{3.12}$$

Define the first term in (3.12) as the classification rate, $R_K = \frac{1}{L}H(K)$, and the second term as the rate conditioned on the classification, $R_{class}(D)$.

3.4 Experimental Results

To evaluate the effectiveness of the mixture model rate-distortion function in (3.12), first we perform spherical VQ similar to [23]. However, for simplicity, the Z_8 lattice is used instead of the RE_8 lattice, and we compare the estimated encoding rate from (3.12) to the spherical VQ performance. Then, later in this section, we use (3.12) to estimate the encoding rate of RE_8 lattice spherical VQ in the AMR-WB+ standard [23].

The rate required for lossless coding of the Z_8 SVQ codevectors is determined as follows. Let $\boldsymbol{v}$ be a Z_8 codevector with squared radius $r = \sum_{i=1}^{L} v_i^2 = \|\boldsymbol{v}\|^2$. A product code is used to encode $\boldsymbol{v}$, consisting of two parts: 1) a code is used to specify the sphere of squared radius r, and 2) a code is used to specify the codevector on a given sphere. The

Table 3.1: $R_{class}(D)$ corresponding to Table 3.2 (bits per sample)

SVQZ_8	**$R_{class}(D)$**					
	***K*=1**	***K*=2**	***K*=4**	***K*=8**	***K*=16**	***K*=32**
Δ = 0.5	4.46	3.44	3.19	3.14	3.14	3.15
Δ = 1.0	3.47	2.46	2.20	2.16	2.16	2.17
Δ = 2.0	2.54	1.53	1.28	1.28	1.27	1.29

ideal required rate can be expressed as

$$R_i = \frac{1}{L}[\log_2(1/P(r)) + \log_2 N(r)] \quad \text{bits/sample,} \tag{3.13}$$

where $P(r)$ is the probability that $\boldsymbol{v}$ has squared radius r and $N(r)$ is the number of Z_8 lattice points that lie on the sphere. $N(r)$ can be computed off-line from the theta function for the Z_8 lattice [57]. In practice the allowed range of lattice codevector radius can be truncated, and *overload* lattice codevectors losslessly encoded using the method of Voronoi extension (as in [23], [39]), or by simply partitioning the overload codevector into subblocks, and using a separate lossless code to encode the subblocks. In the experimental results to follow, the latter method is used, together with (3.13), to compute Z_8 SVQ encoding rates.

The source data are the spectrally pre-shaped and scaled transform coefficients from the AMR-WB+ encoding method. The transform coefficients are quantized using the scaled Z_8 lattice, and the average encoding rate computed using (3.13). The squared error distortion

Table 3.2: Estimated rate, $R_{GMM}(D)$ in (3.12), and empirical average encoding rate of SVQZ_8 at various step sizes

K	Average rate (bits/sample)					
	$\Delta = 0.5$		$\Delta = 1.0$		$\Delta = 2.0$	
	$R_{GMM}(D)$	SVQZ_8	$R_{GMM}(D)$	SVQZ_8	$R_{GMM}(D)$	SVQZ_8
1	4.46	3.39	3.47	2.42	2.54	1.53
2	3.53		2.54		1.61	
4	3.40		2.41		1.49	
8	3.47		2.50		1.61	
16	3.56		2.58		1.69	
32	3.69		2.72		1.84	

is controlled in the simulations by adjusting the Z_8 lattice step size, and is computed as

$$D = \frac{1}{L \cdot M} \sum_{m=1}^{M} \sum_{l=1}^{L} (y_{m,l} - \hat{y}_{m,l})^2, \tag{3.14}$$

where M is the number of data vectors, $L = 8$ is the vector dimension coresponding to Z_8, and $\hat{\boldsymbol{y}}$ is the lattice SVQ codevector for $\boldsymbol{y}$. The distortion is thus the average squared error from lattice SVQ, and the rate, from (3.13) is the (idealized) spherical lattice VQ encoding rate. For a given lattice step size, the resulting simulation distortion, D, is used in (3.12) to determine the GMM estimate of encoding rate, $R_{GMM}(D)$.

The simulation results are summarized in Tables 3.1-3.2, comparing the average rate required for spherical VQ using the scaled Z_8 lattice ($SVQZ_8$) to the GMM rate-distortion function $R_{GMM}(D)$ in (3.12), with $\beta = 1$ (corresponding to Z_8 latice) and for several step sizes (equivalent to several signal-to-noise ratios, SNR). From Tables 3.1 and 3.2, it can be seen

that the bit rate necessary to specify the block class, R_K, costs roughly 0.1 bits/dimension in classification rate for each doubling of the number of classes in the GMM. Examining Table 3.2 shows that for a single class (a memoryless Gaussian source model), the rate-distortion model over-estimates the rate by a significant margin. For effective rate-distortion modeling, $K = 4$ is a sufficient numbers of classes to capture the available *classification gain*, and increasing K beyond 4 needlessly wastes rate in the modeling. This can be seen from Table 3.1 in which the conditional class encoding rate in (3.12), $R_{class}(D)$, saturates for $K \geq 4$. Note from Fig. 3.3 that the K-class mixture modeling estimate of the empirical block energy density continues to improve as K ranges from 1 to 32. For $K = 4$ the mixture model energy density is a rather coarse estimate of the empirical density. However, for modeling of rate-distortion performance, $K = 4$ classes is adequate. The 4-class GMM rate estimate is close to the (ideal) observed Z_8 VQ encoding rate, underestimating it by no more than 0.04 bits/sample.

The GMM rate-distortion function in (3.12) is used to estimate the average encoding rate of the RE_8 lattice VQ in the AMR-WB+ algorithm. The value $\beta = 1.16$ (corresponding with RE_8 lattice) is used in (3.12) [7], [57] for various distortions corresponding to different encoding modes of AMR-WB+ [23]. The results are shown in Tables 3.3-3.4. We note that the encoding in [23] uses a fixed rate per frame, whereas the GMM rate-distortion function in (3.12) does not impose this constraint. Hence, one expects the GMM rate-distortion mode to lower bound the observed RE_8 lattice VQ performance.

From Table 3.4, the rate-distortion modeling with $K = 4$ classes reasonably well

Table 3.3: $R_{class}(D)$ corresponding to Table 3.4 (bits per sample)

AMR-WB+	$R_{class}(D)$					
rate (kbps)	K**=1**	K**=2**	K**=4**	K**=8**	K**=16**	K**=32**
10.4	1.57	0.64	0.61	0.60	0.59	0.59
16.8	2.06	1.08	0.95	0.90	0.90	0.90
24.0	2.52	1.50	1.26	1.26	1.25	1.27

Table 3.4: Comparison between $R_{GMM}(D)$ in (3.12) and average rate using RE_8 in AMR-WB+

K	**10.4 kbps**		**16.8 kbps**		**24.0kbps**	
	$R_{GMM}(D)$	**AMR WB+**	$R_{GMM}(D)$	**AMR WB+**	$R_{GMM}(D)$	**AMR WB+**
1	1.57	0.6	2.06	1.05	2.52	1.54
2	0.72		1.16		1.58	
4	0.81		1.15		1.46	
8	0.93		1.23		1.59	
16	1.00		1.32		1.67	
32	1.13		1.43		1.81	

models the AMR-WB+ rate at high rate (24.0 kbps), similar to the $SVQZ_8$ case. The results in Table 3.3 also demonstrate again that for rate-distortion modeling in (3.12), the number of classes, K, equal to 4 is enough to capture classfication gain of the mixture model.

At low and medium rates, however, the rate-distortion modeling in (3.12) does not adequately predict the AMR-WB+ encoding rate. The reason is that as the rate decreases, the frame gain (normalization factor) increases and the number of source vectors encoded as the zero codevectors increases. Thus, the overall distortion gets larger and the small

Table 3.5: Results of the comparison based on the modified modeling

K	10.4 kbps		16.8 kbps		24.0kbps	
	$R_{GMM}(D)$	AMR WB+	$R_{GMM}(D)$	AMR WB+	$R_{GMM}(D)$	AMR WB+
1	0.75	0.60	1.41	1.05	2.12	1.54
2	0.59		1.02		1.51	
4	0.58		1.00		1.45	
8	0.62		1.08		1.54	
16	0.62		1.10		1.60	
32	0.66		1.15		1.67	

distortion assumption in (3.12) is not valid. Some modifications have to be made in order to use rate-distortion modeling in (3.12) at low and medium rates.

A modification to the modeling approach is to use the GMM to model only *significant* source vectors, where significant means a source vector encoded using the AMR-WB+ RE_8 lattice VQ as a non-zero codevector. Using only significant source vectors, GMM parameters are again estimated using the EM algorithm, and the average encoding rate $R_{GMM}(D)$ is computed from (3.12). The total estimated rate, $R_{total}(D)$, is then computed by

$$R_{total}(D) = \frac{(\hat{R}_{GMM}(D) \times N_{nonzero}) + (R_{zero} \times N_{zero})}{N_{nonzero} + N_{zero}} \tag{3.15}$$

where $\hat{R}_{GMM}(D) = R_{GMM}(D) + 0.125$ is the estimated encoding rate for significant source vectors based on (3.12), plus an additional 0.125 bits/sample (or 1 bit/source vector) to distinguish the codeword as not having the same prefix as the zero vector codeword. R_{zero}

is the encoding rate for zero codevectors, which for AMR-WB+ [23] is a fixed rate of 1 bit/vector (or 0.125 bits/sample). N_{zero} and $N_{nonzero}$ are the number of zero codevectors and nonzero codevectors, respectively.

The estimated encoding rate based on the modified GMM approach is presented in Table 3.5. This provides a better prediction of the observed AMR-WB+ encoding rates. Again, $K = 4$ is a sufficient number of classes.

Chapter 4

Block-Adaptive Lattice Vector Quantization in Image Coding

4.1 Introduction

Subband image decomposition [58], typically using a wavelet transform [59], is an effective method of image representation, supporting several desirable image compression properties, such as scalability, region-of-interest coding, and embedded coding for progressive transmission [60]-[61]. An octave subband structure is commonly used (e.g., [61]-[62]). In all except the lowest frequency subband, the subband (or, referred to interchangeably in this chapter, wavelet) coefficients have marginal density that is well-modeled as Laplacian, or generalized Gaussian with shape parameter $\alpha = 0.7$ [59], [63]. The subband coefficients also have small correlation [63], but they are generally not memoryless, instead displaying an evident nonlinear energy dependence. This is shown in Figure 4.1, comparing the empirical block energy density for contiguous blocks of subband coefficients, to the energy density of blocks of memoryless Laplacian or generalized Gaussian data of the same mean and variance. The empirical block energy density is highly peaked at small energy,

and heavy-tailed, reflective of the large number of small amplitude subband coefficients, with the larger magnitude coefficients typically clustered, rather than distributed uniformly throughout the subbands. Figure 4.1 uses block energy defined as $\|\mathbf{x}\|_2^2 = \sum_{x_i \in \text{block}} |x_i|^2$, but similar results are obtained defining the block energy as the ℓ_1 norm. Efficient subband image coding algorithms take advantage of such energy dependence, such as by using a quad-tree data structure across [24],[60] or within subbands [64], or by using local contexts in bit-plane coding [62].

The image subband block energy density can be modeled as a mixture of memoryless Gaussian or Laplacian random variables [49]-[50]. A suitable block-based lattice vector quantizer (LVQ) can be formed using concentric spheres (motivated by the Gaussian mixture model), or pyramids (motivated by the Laplacian mixture model). A spherical LVQ using the RE_8 lattice [18] is used in the AMR-WB+ audio coding standard [23]. A pyramid-based LVQ has been used for image coding [19], [65]. Lattice-based vector quantization offers the potential of relatively simple quantization and granular coding gain [57]. However, the mapping from lattice codevectors to binary codewords can be cumbersome, with complexity that typically grows with the lattice dimension.

In this chapter, we propose an image coding algorithm based on lattice-based spherical vector quantization (LSVQ) or lattice-based pyramid VQ (LPVQ), motivated by Figure 4.1. Quadtree partitioning is used to divide regions of coefficients into blocks to be lattice vector quantized and encoded. The block sizes are selected based on block energy constraints. If a block has too large energy, it is partitioned into smaller blocks. The maximum block energy

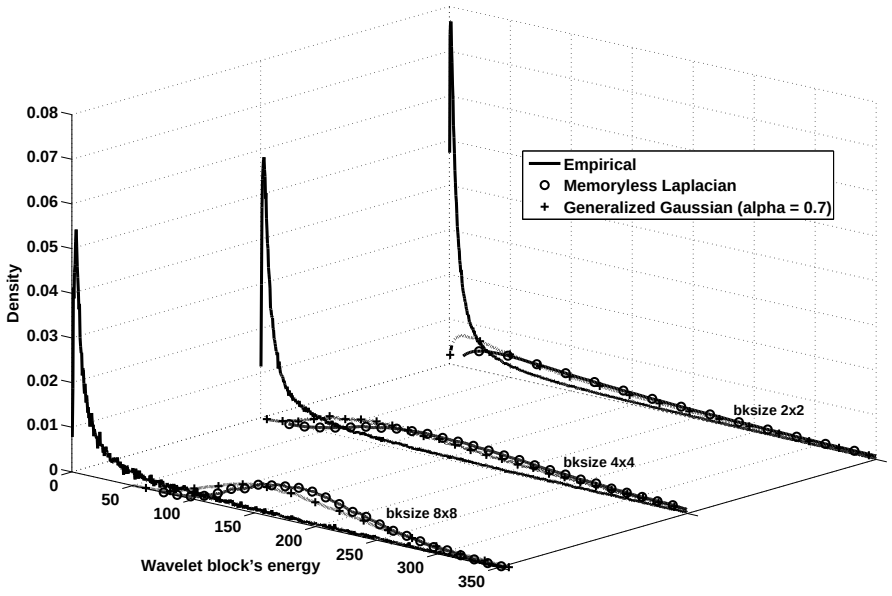


Figure 4.1: Empirical density of wavelet block's energy compared to that of memoryless Laplacian and generalized Gaussian vectors with the same mean and variance.

is selected subject to complexity constraints on the enumeration of lattice codevectors as binary codewords. Blocks of coefficients are vector quantized and encoded using a product code based on LSVQ or LPVQ. The block partitioning is motivated by SPIHT-type algorithms, [64], [66], however, instead of partitioning based on significant and insignificant bit-plane coefficients, the proposed method partitions based on block energy. The resulting bitstream is not embedded, but offers the potential granular gain of the lattice.

4.2 Block-Adaptive Lattice VQ

This section briefly describes the basic concepts of the proposed block-based image coding using set partitioning, and introduces the notation and terminology used in this chapter. An image is decomposed into subbands, typically using a wavelet transform. This transformed image, or portions of it, are structured as sets of coefficients. A set is called *e-limited* if its energy is less than a threshold; otherwise it is called *non-e-limited*. The threshold is generally dependent on the number of coefficients in the set (the vector dimension). The block partitioning algorithm begins with large sets of coefficients. Sets are tested, and if non-e-limited, are split into subsets for further testing. Sets that are e-limited are quantized and encoded without further partitioning of the set. This can be done directly, or in a progressive manner. Partitioning can be done based on spatial orientation trees [64], [67] or quadtrees [66]. The goal of partitioning is to keep splitting off clusters of large energy coefficients while maintaining a large set of relatively small energy coefficients which are jointly vector quantized and encoded. By doing this, the energy clustering evident in Figure 4.1 can be exploited and the quantization and encoding can be performed efficiently. In this work, partitioning based on quadtrees, similar to [66], is used and the testing condition uses either the ℓ_2 or ℓ_1 norm for set energy. More specifically, an e-limited set in this work implies a set that can be quantized and encoded using a product code based on LSVQ or LPVQ and hence, the vector of set coefficients lies within a bounding sphere or pyramid of respective ℓ_2 or ℓ_1 radius.

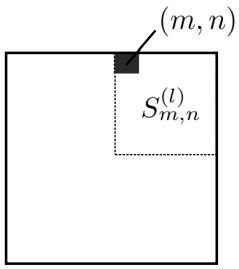


Figure 4.2: Set $S^{(l)}_{m,n}$

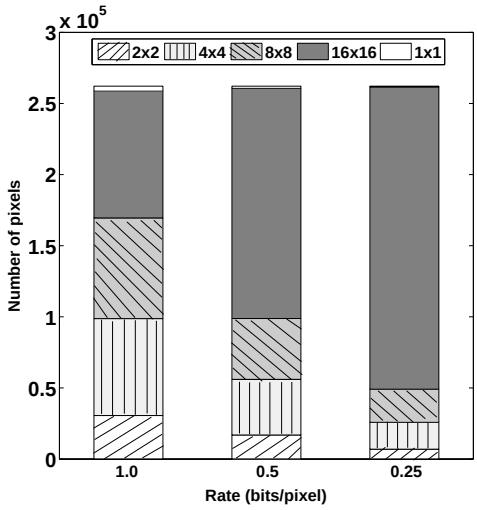


Figure 4.3: Number of pixels for each block size

Let $c_{i,j}$ and $\hat{c}_{i,j}$ be the unquantized and corresponding quantized version of the wavelet coefficient at coordinate (i, j) of a transformed image of size $N \times N$. Denote a subset (or block) at level l due to quadtree parititioning, and its corresponding quantized version, as $S^{(l)}_{m,n}$ and $\hat{S}^{(l)}_{m,n}$, respectively, where (m, n) is the coordinate of a given coefficient in $S^{(l)}_{m,n}$ used to refer to that subset, e.g., the coordinate of the coefficient at the block top-left corner, as shown in Figure 4.2. Denote subsets (or subblocks) resulting from quadtree partitioning

$S^{(l)}_{m,n}$ as $O(S^{(l)}_{m,n})$. Define $E(\hat{S}^{(l)}_{m,n}, \nu) = \sum_{\hat{c}\in\hat{S}^{(l)}_{m,n}} |\hat{c}_{i,j}|^{\nu}$, where $\nu \in \{1, 2\}$, as the energy in a block of quantized coefficients. A set $\hat{S}^{(l)}_{m,n}$ is said to be e-limited if

$$E(\hat{S}^{(l)}_{m,n}, \nu) \leq T^{(l)} \tag{4.1}$$

where $T^{(l)}$ is a predefined threshold for partition level l, and the threshold can vary with partition level.

The LSVQ and LPVQ use as codevectors all lattice vectors within a bounding sphere or pyramid. Let n denote the vector dimension and Λ_n the lattice. We assume a binary lattice is of the form described in [22], with $\Lambda_n \subseteq Z^n$, where Z^n is the cubic lattice of dimension n. Let $N_{\Lambda,1}(n, k)$ denote the number of lattice points satisfying $||\mathbf{x}||_1 = k$, and let $N_{\Lambda,2}(n, k)$ denote the number of lattice points satisfying $||\mathbf{x}||_2^2 = k$, $k = 0, 1, \ldots$. These values are summarized as the lattice nu-function [19] and theta function [57], respectively. A variable-length code, C_ν, (e.g., a Huffman code) is used to encode the energy, with codeword denoted by $\mathbf{c}_\nu(||\mathbf{x}||_\nu^\nu)$. An enumeration code, C_c, uses $\lceil \log_2(N_{\Lambda,\nu}(n, ||\mathbf{x}||_\nu^\nu)) \rceil$ bits/vector to encode the codevector, conditioned on the energy. Thus, the encoding bits for the quantized block $\hat{S}^{(l)}_{m,n}$ are composed of two parts: 1) a codeword used to specify the sphere or the pyramid, $\mathbf{c}_\nu(||\mathbf{x}||_\nu^\nu)$, and 2) a codeword used to specify the codevector on a given sphere or pyramid, $\mathbf{c}_c$. Encoding and decoding using codes C_ν and C_c can be implemented primarily using look-up tables.

Figure 4.3 shows the relative distribution of blocks of various sizes resulting from the

quadtree partitioning of wavelet transformed image based on the proposed algorithm and the ℓ_2 energy measure. Similar distributions are obtained for the ℓ_1 case. At low overall encoding rate there are many large blocks and very few singleton blocks, most of the latter from the low frequency subband. Even at large encoding rate, there are relatively few singleton blocks, and the abundance of 2×2 and 4×4 blocks suggest the granular gains of the D_4, E_8, Λ_{16}, or other lattices can improve the encoding signal-to-noise ratio (SNR).

4.3 Coding Algorithm

The main idea of the partitioning algorithm is simple. Large subsets are tested according to an energy threshold to determine whether partitioning is required. As soon as subsets have energy within the testing threshold, they are encoded using a product code based on LSVQ ($\nu = 2$) or LPVQ ($\nu = 1$) and are removed from further consideration. Since the order in which subsets are tested is important and needs to be maintained, a list is used to store the testing order. Denote the list of sets as LS. Also, define the binary testing function as

$$EF(\hat{S}_{m,n}^{(l)}, \nu) = \begin{cases} 0, & \text{if } E(\hat{S}_{m,n}^{(l)}, \nu) \leq T^{(l)} \\ 1, & \text{if } E(\hat{S}_{m,n}^{(l)}, \nu) > T^{(l)}. \end{cases} \tag{4.2}$$

Assume N is power of 2. The proposed encoding algorithm is presented in Algorithm 1.

The algorithm uses the functions $LVQ()$ to perform lattice VQ, *encode*() to generate the

Algorithm 1 Block-Adaptive LVQ

Given: subband image $\{c_{i,j}\}$ and chosen ν-norm
Initialize: $LS = \{(0,0)\}$, $l = 0$
Start:

```
while LS ≠ φ do
  for all (m, n) ∈ LS except those added in current l do
    if N > 1 then
      Ŝ^(l)_{m,n} = LVQ(S^(l)_{m,n}, N);
      output EF(Ŝ^(l)_{m,n}, ν);
      if (EF(Ŝ^(l)_{m,n}, ν) = 1) then
        add O(S^(l)_{m,n}) to the end of LS;
        remove (m, n) from LS;
      else
        output encode(Ŝ^(l)_{m,n}, N, ν);
        remove (m, n) from LS;
      end if
    else
      Ŝ^(l)_{m,n} = round(S^(l)_{m,n});
      output encode_scalar(Ŝ^(l)_{m,n});
      remove (m, n) from LS;
    end if
  end for
  N ← N ≫ 2;
  l ← l + 1;
end while
```

binary codeword representing a lattice codevector, and $encode_{scalar}()$ to generate the binary codeword for a singleton block. To quantize the block $S^{(l)}_{m,n}$ using LVQ, the vector is formed based on raster scanning. The decoding algorithm is similar to the encoding algorithm, replacing *output* to *input*, *encode*() to *decode*(), and quantization is no longer required. In the algorithm main loop (while loop), the block size is decreased if the block is non-e-limited. The decoder tracks the block size from the sequence of energy-testing bits. The algorithm makes no assumption on the lattice (or lattices) used. However, if the integer lattice is used, lattice quantization can be done in advance, before the algorithm execution, and the partitioning can be done directly to quantized blocks, $\hat{S}^{(l)}_{m,n}$.

A simple modification of Algorithm 1 makes the partitioning dependent on the spatial orientation of each subband. Such a modified algorithm is presented in Algorithm 2. There are four lists for four different types of subbands (list $LS[0-3]$ is for subblocks in LL, LH, HL, and HH subband, respectively). Each element in the list contains two block information parameters, the coordinate (m, n) used to refer to $S^{(l)}_{m,n}$, and an array L = [row,col], which specifies the the size (row and column) of $S^{(l)}_{m,n}$. The main idea is unchanged. The block of wavelet coefficients, $S^{(l)}_{m,n}$, is tested using (4.2) but now the size of block $S^{(l)}_{m,n}$ is different for different subbands. As soon as the block $S^{(l)}_{m,n}$ of size L has energy within the threshold, it is encoded using either LSVQ or LPVQ, as described above. The quadtree partitioning for non-e-limited set $S^{(l)}_{m,n}$ is done with respect to the spatial orientation of the subband, as shown in Figure 4.4. Observe that there are always only four entries (blocks) in $LS[0]$ and the partitioning for each entry in this list is done differently based on which

Algorithm 2 Block-Adaptive LVQ with spatial orientation

Given: subband image $\{c_{i,j}\}$ and chosen ν-norm
Initialize: $L = [N, N]$, $LS[0] = \{\{(0,0), L\}\}$, $LS[1] = \{\}$, $LS[2] = \{\}$, $LS[3] = \{\}$, $l = 0$, $M = N/2$.
Start:

while $LS \neq \phi$ **do**
  for $i = 0$ to 3 **do**
    if $LS[i] \neq \phi$ **then**
      for each element $\in LS[i]$ except those added in current l **do**
        if $L \neq [1,1]$ **then**
          if $i = 0$ or $i = 3$ **then**
            $L = [M, M]$;
            $\hat{S}^{(l)}_{m,n} = LVQ(S^{(l)}_{m,n}, L)$;
          else
            if $i = 1$ **then**
              $L = [N, M]$;
              $\hat{S}^{(l)}_{m,n} = LVQ(S^{(l)}_{m,n}, L)$;
            else
              $L = [M, N]$;
              $\hat{S}^{(l)}_{m,n} = LVQ(S^{(l)}_{m,n}, L)$;
            end if
          end if
          output $EF(\hat{S}^{(l)}_{m,n}, L, \nu)$;
          if $(EF(\hat{S}^{(l)}_{m,n}, L, \nu) = 1)$ **then**
            if $i = 0$ **then**
              for $j = 0$ to 3 **do**
                add $O(S^{(l)}_{m,n})$ to the end of $LS[j]$ together with its block size $\hat{L}$;
                remove element $\{(m,n), L\}$ from $LS[0]$;
              end for
            else
              add $O(S^{(l)}_{m,n})$ to the end of $LS[i]$ together with its block size $\hat{L}$;
              remove element $\{(m,n), L\}$ from $LS[i]$;
            end if
          else
            output $encode(\hat{S}^{(l)}_{m,n}, L, \nu)$;
            remove $\{(m,n), L\}$ from $LS[i]$;
          end if
        else
          $\hat{S}^{(l)}_{m,n} = round(S^{(l)}_{m,n})$;
          output $encode_{scalar}(\hat{S}^{(l)}_{m,n})$;
          remove $\{(m,n), L\}$ from $LS[i]$;
        end if
      end for
    end if
    $N \leftarrow N \gg 2$; $M \leftarrow M \gg 2$; $l \leftarrow l + 1$;
  end for
end while

list ($LS[0], \ldots, LS[3]$) its subblocks are added to. For all lists, the partitioning can be done until $S^{(l)}_{m,n}$ becomes a single pixel, which then is encoded as a scalar value. The encoder functions *encode*() and $encode_{scalar}$() are defined the same as in Algorithm 1 except one more parameter is required, which is the size of $S^{(l)}_{m,n}$. Similar to Algorithm 1, if the integer lattice is used, the quantization can be performed in advance before the algorithm is executed, and the partitioning can be done directly to quantized blocks, $\hat{S}^{(l)}_{m,n}$.

4.4 Simulation Results

To evaluate the performance of the proposed algorithm, a five-level subband decomposition is generated using a 9/7 biorthogonal filter bank [24]. For simplicity, the simulation uses integer (cubic) lattice VQ. Both LVQ based on ℓ_1 (LPVQ) and ℓ_2 norms (LSVQ) are used. A Huffman code is generated for C_ν using empirical energy probabilities for each partition level. For Algorithm 1, the largest allowed block size is 16×16 ($N = 16$) so that the allowed possible block sizes due to partitioning are 16×16, 8×8, 4×4, 2×2, and 1×1. For Algorithm 2, the block size is considered as the number of pixels in the block since the block is not square anymore. The largest allowed block size is of 256 pixels. The allowed possible block sizes due to partitioning are 256, 64, 16, 4, 2, and 1 pixels.

The simulation results based on Algorithm 1 are shown in Table 4.1, where $\hat{\mathbf{R}}_T$ and $\mathbf{R}_T$ are total encoding rate with and witout entropy coding raw energy-testing bits, respectively.

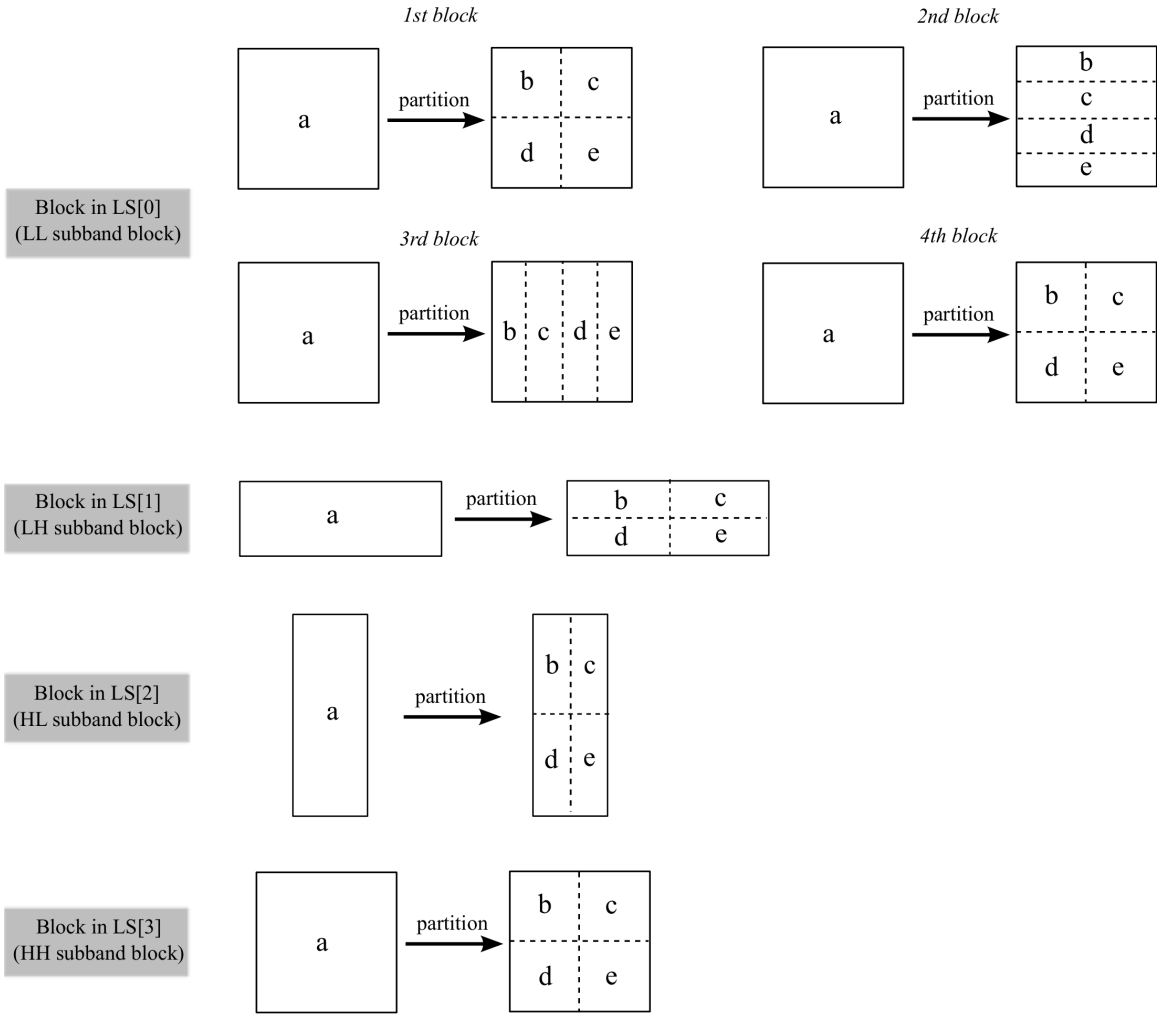


Figure 4.4: Partition scheme for the algorithm 2, where block a is partitioned into four subblocks b, c, d, and e

LENA

	PSNR	39.683	38.439	37.537	36.810	35.652	34.525	33.156	32.457	31.865	30.080
ℓ_2	$\mathbf{R}_T$	0.949	0.724	0.593	0.504	0.388	0.303	0.222	0.190	0.166	0.109
	$\hat{\mathbf{R}}_T$	0.930	0.709	0.581	0.492	0.379	0.295	0.215	0.185	0.161	0.105
ℓ_1	$\mathbf{R}_T$	0.906	0.692	0.566	0.481	0.371	0.289	0.212	0.181	0.157	0.102
	$\hat{\mathbf{R}}_T$	0.876	0.669	0.548	0.466	0.359	0.280	0.205	0.174	0.151	0.097

BARBARA

	PSNR	35.753	34.863	34.078	32.461	30.570	28.923	27.630	26.612	25.750	24.483
ℓ_2	$\mathbf{R}_T$	0.981	0.868	0.776	0.611	0.443	0.321	0.245	0.192	0.154	0.105
	$\hat{\mathbf{R}}_T$	0.949	0.840	0.751	0.592	0.430	0.312	0.238	0.187	0.149	0.101
ℓ_1	$\mathbf{R}_T$	0.967	0.855	0.765	0.601	0.436	0.317	0.242	0.190	0.151	0.101
	$\hat{\mathbf{R}}_T$	0.929	0.822	0.736	0.580	0.421	0.306	0.233	0.183	0.145	0.097

GOLDHILL

	PSNR	35.900	34.935	33.486	32.171	30.725	30.052	29.516	28.652	27.969	26.638
ℓ_2	$\mathbf{R}_T$	0.974	0.807	0.591	0.430	0.288	0.233	0.195	0.144	0.112	0.066
	$\hat{\mathbf{R}}_T$	0.946	0.784	0.575	0.419	0.281	0.227	0.190	0.140	0.108	0.063
ℓ_1	$\mathbf{R}_T$	0.939	0.777	0.568	0.413	0.276	0.223	0.186	0.137	0.106	0.062
	$\hat{\mathbf{R}}_T$	0.906	0.752	0.551	0.401	0.268	0.215	0.180	0.132	0.102	0.058

AERIAL

	PSNR	34.001	31.439	29.478	27.827	26.622	25.691	24.929	24.327	22.586	21.773
ℓ_2	$\mathbf{R}_T$	1.255	0.875	0.626	0.447	0.338	0.266	0.214	0.179	0.097	0.070
	$\hat{\mathbf{R}}_T$	1.214	0.847	0.607	0.435	0.329	0.258	0.207	0.173	0.094	0.067
ℓ_1	$\mathbf{R}_T$	1.191	0.830	0.594	0.424	0.319	0.251	0.202	0.169	0.092	0.065
	$\hat{\mathbf{R}}_T$	1.145	0.801	0.576	0.412	0.309	0.243	0.195	0.163	0.087	0.061

Table 4.1: Encoding rate (in bpp) and PSNR (in dB) when LSVQ (ℓ_2-norm) and LPVQ (ℓ_1-norm) are used with and without entropy coding of the raw energy-testing bits.

From the table, the LPVQ performs slightly better than the LSVQ. By entropy coding the energy-testing bits (i.e.,arithmatic coding [68]), the total encoding rate reduces about 3-5%. So, there is a possible small gain by entropy coding those bits. Moreover, the reduction by entropy coding the energy testing-bits in LPVQ is larger than in LSVQ which implies that there are more statistical dependence among those bits in LPVQ than in LSVQ. Using better lattices, based on the partitioned block sizes, allows improvements in LVQ SNR, with up to 0.36 dB increase for the D_4 lattice (and 2×2 blocks), up to 0.65 dB increase for the E_8 lattice (and 8-dimensional vectors), and up to 0.86 dB for the 16-dimensional Barnes-Wall lattice [57]. The performance of Algorithm 1 with entropy coding the energy-testing bits is compared with SPIHT in Figure 4.5-4.8. The performance is competitive with the SPIHT algorithm (using adaptive arithmetic coding), with LPVQ performing a little better at low and moderate encoding rate [69]. The simulation results based on Algorithm 2 demonstrate there is a very small improvement (about less than 1%) in rate using this algorithm on some images, such as Goldhill and Lena, at moderate or high rate when LSVQ is used, as shown in Table 4.2.

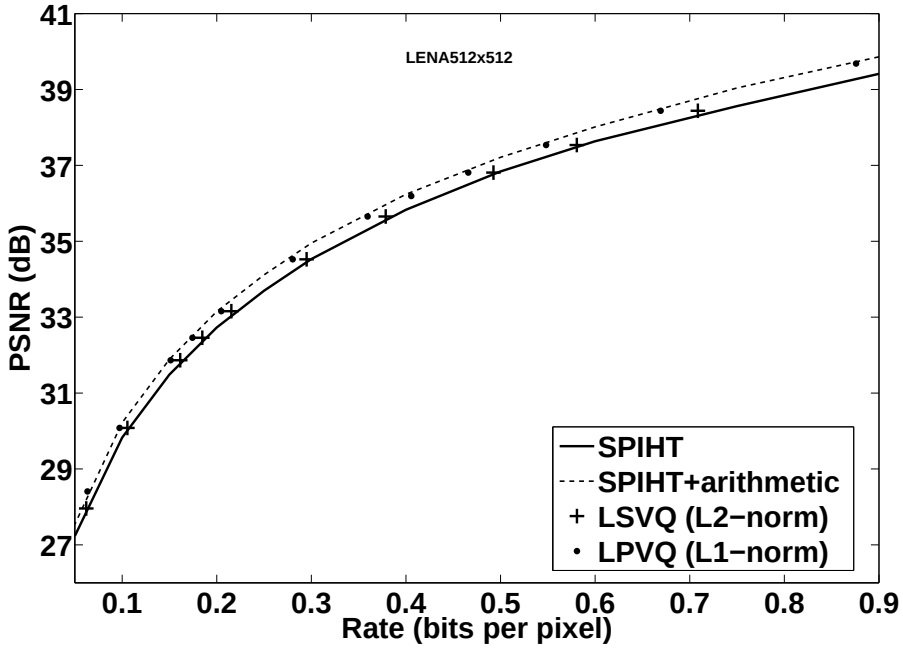


Figure 4.5: Performance comparison for *Lena* image

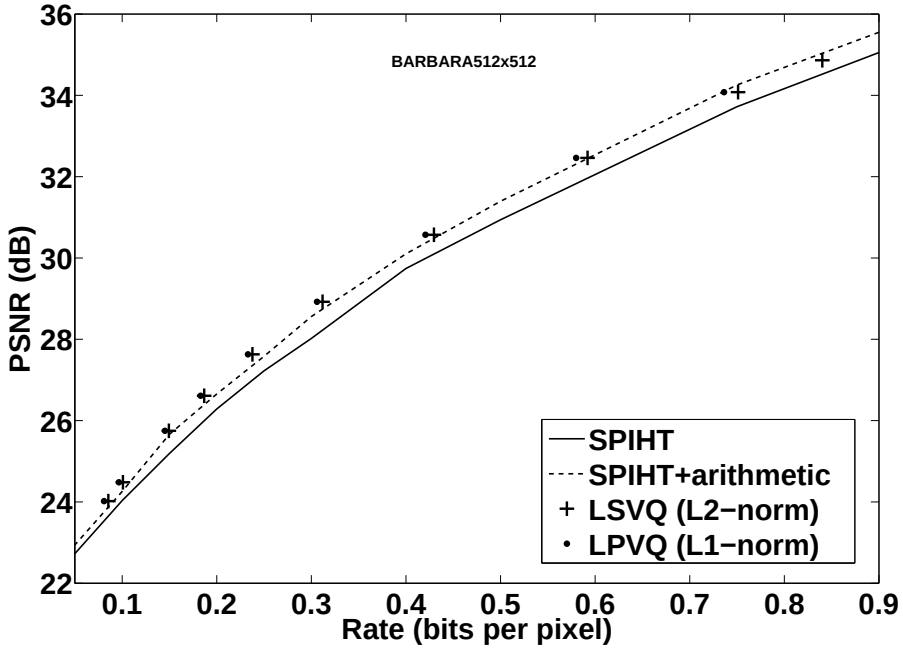


Figure 4.6: Performance comparison for *Barbara* image

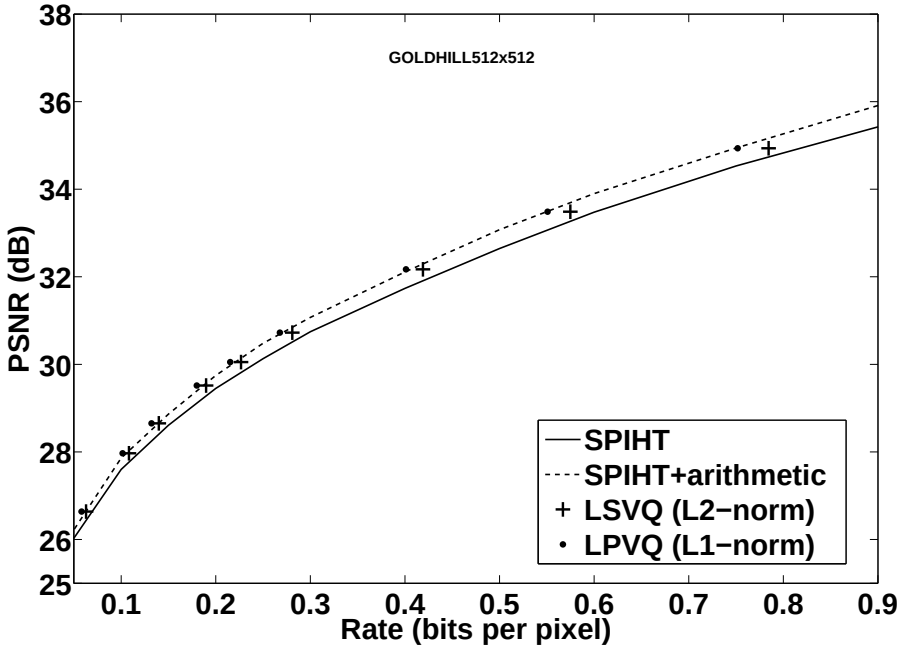


Figure 4.7: Performance comparison for *Goldhill* image

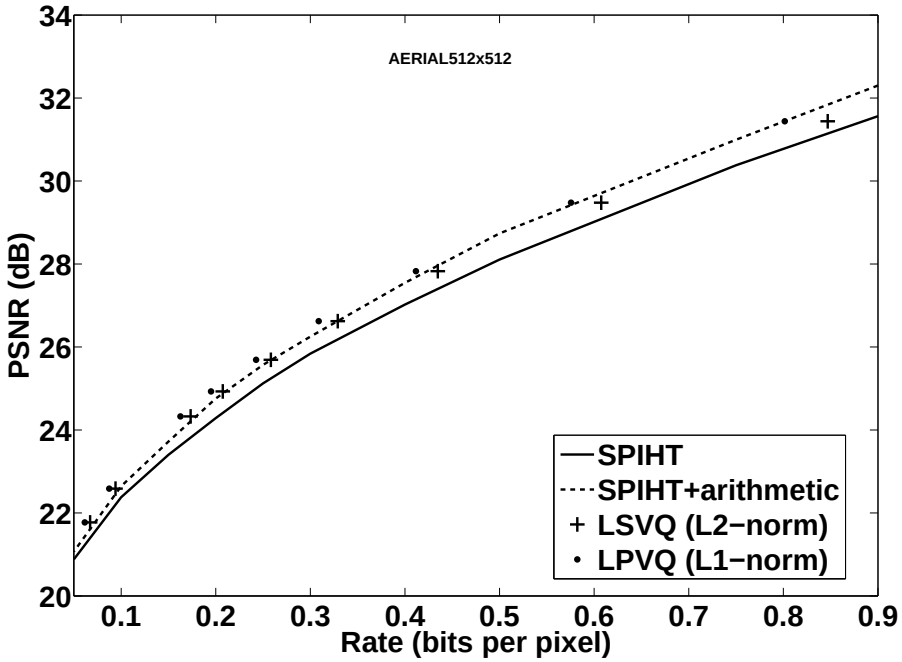


Figure 4.8: Performance comparison for *Aerial* image

Image	PSNR (dB)	Δ_R (%)
Lena	33.16	0.32
	35.65	0.80
	39.68	0.40
Barbara	32.46	-0.39
	35.75	-0.19
	39.77	0.65
Goldhill	33.49	1.46
	35.90	1.13
	38.86	0.88

Table 4.2: Percent improvement in rate using Algorithm 2 with LSVQ, where $\Delta_R = (R_T^{old} - R_T^{new})/R_T^{old} \times 100$.

Chapter 5

Bit-Plane Coding of Lattice Codevectors

Bit-plane coding is an effective method of variable-length coding in image compression [60]-[61], and provides the functionality of an embedded bit-stream. This chapter describes the bit-plane coding of lattice VQ codevectors, focusing on the D_4, RE_8, and 16-dimensional Barnes-Wall (denoted by Λ_{16}) lattices [7], [57]. Using Forney's characterization of these (and other) lattices [22], it becomes apparent that if the (binary) lattice codevectors are represented in sign-magnitude form, only a few of the least significant magnitude bit-planes are "lattice-defining," and the lattice imposes no restriction on the more significant bit planes. Hence, any convenient bit-plane coding method can be used on the more significant magnitude bit planes, with modification only required for coding the lattice-defining least significant bit planes. In this chapter, we describe the lattice-defining magnitude bit-planes, and suggest a method for losslessly encoding them. As an example, we consider image compression using the SPIHT algorithm, but modified to use the D_4 lattice.

5.1 Binary Lattices

A real N-dimensional *lattice* [22], Λ, is an infinite set of real vectors (points, N-tuples) in real N-dimensional space $\mathbb{R}^N$ that forms a commutative group under vector addition. If the vectors of the lattice are integer-valued, it is called an *integer lattice*. A *sublattice* Λ' of Λ is a subset of points in Λ that itself is also an N-dimensional lattice. The sublattice Λ' induces a partition Λ/Λ' of Λ into $|\Lambda/\Lambda'|$ disjoint *cosets* of Λ', where $|\Lambda/\Lambda'|$ is the *order* of the partition. Any cosets of Λ' can be written as $\Lambda' + \mathbf{c}$, for some $\mathbf{c} \in \Lambda$, and $\mathbf{c}$ is called a *coset representative*. A coset containing $\mathbf{0}$ is called the *zero coset* since $\Lambda' + \mathbf{0}$ is itself, Λ', and $\mathbf{0}$ is always used as the coset representative for the zero coset. Let $[\Lambda/\Lambda']$ be any system of coset representatives $\mathbf{c}$, one for each coset. Then, Λ is the union of the $|\Lambda/\Lambda'|$ cosets $\Lambda' + \mathbf{c}, \mathbf{c} \in [\Lambda/\Lambda']$.

A *binary lattice* [22],[70], Λ, is defined as an N-dimensional integer lattice that has $2^m\mathbb{Z}^N$ as a sublattice for some integer m. Clearly, it is also a sublattice of the N-dimensional cubic lattice $\mathbb{Z}^N$. After the integer lattice, the simplest lattice is the D_N lattice, consisting of all the integer N-tuples with the sum of coordinates even. A fast encoding algorithm exists for lattice VQ using the D_N lattice [57]. The *2-depth* of a binary lattice is the least m for which $2^m\mathbb{Z}^N$ is a sublattice. The integer (cubic) lattice, $\mathbb{Z}^N$, trivially has 2-depth 1. Other examples of binary lattices are the Schläfli lattice, D_4, and the Gosset lattice E_8, both with 2-depth 1, and the 16-dimensional Barnes-Wall lattice, Λ_{16}, with 2-depth 2.

Forney [22] shows that a family of Barnes-Wall lattices and its principal sublattices can

be generated by iteratively applying a simple construction called the *squaring construction* to a one-dimensional partition chain of integer lattices $\cdots/\mathbb{Z}/\mathbb{Z}/2\mathbb{Z}/2\mathbb{Z}/4\mathbb{Z}/4\mathbb{Z}/\cdots$. He also points out the close relationship between this family of lattices and the family of Reed-Muller (RM) codes [71] and develops explicit formulas expressing the lattice in terms of simpler lattices and the appropriate RM code. From [22], denote the lattice constructed by the iterated squaring construction as $\Lambda(r, n)$, where $r \in \mathbb{Z}$ and the lattice dimension is $N = 2^{n+1}$. $\Lambda(r, n) = \mathbb{Z}^N$ for $r \geq n$, and $\Lambda(r, n) = R^{-r}\Lambda(0, n)$ for $r \leq 0$, where R is the *rotation operator*, which operates separately on each pair of lattice dimensions by rotating by 45° and scaling by $2^{1/2}$. For example, $R = \left[\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right]$, for $N = 2$, and $R = \left[\begin{smallmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{smallmatrix}\right]$, for $N = 4$. Note that $R^2 = 2I$, where I is the identity operator. By general properties of the square construction, Forney shows that $\Lambda(r, n)$ can be generated using the formula below

$$\Lambda(r,n) = \begin{cases} 2^{(n-r)/2}\mathbb{Z}^{2N} + \displaystyle\sum_{r+1 \leq r' \leq n,\ n-r' \text{ odd}} \mathrm{RM}(r', n+1)2^{(r'-r-1)/2}, & n-r \text{ even} \\ 2^{(n-r+1)/2}\mathbb{Z}^{2N} + \displaystyle\sum_{r+1 \leq r' \leq n,\ n-r' \text{ even}} \mathrm{RM}(r', n+1)2^{(r'-r-1)/2}, & n-r \text{ odd,} \end{cases} \tag{5.1}$$

where RM$(r', n+1)$ is a Reed-Muller code of order r' and length $N = 2^{n+1}$. When $r = 0$, $\Lambda(0, n)$ is the N-dimensional Barnes-Wall lattice. For example,

$$D_4 = \Lambda(0,1) = 2Z^4 + (4,3,2), \tag{5.2}$$

$$E_8 = \Lambda(0,2) = 2Z^4 + (8,4,4), \tag{5.3}$$

$$\Lambda_{16} = \Lambda(0,3) = 2D_{16} + (16,5,8), \tag{5.4}$$

where $(4,3,2)$, $(8,4,4)$, and $(16,5,8)$ are (n,k,d) RM codes with block length n, k information bits, and minimum Hamming distance d. Also, for rotated lattices the above formulas become

$$R\Lambda(r,n) = \begin{cases} 2^{(n-r+1)/2}\mathbb{Z}^{2N} + \displaystyle\sum_{r \le r' \le n,\ n-r' \text{ even}} RM(r', n+1)2^{(r'-r-1)/2}, & n-r \text{ odd} \\ 2^{(n-r)/2}\mathbb{Z}^{2N} + \displaystyle\sum_{r \le r' \le n,\ n-r' \text{ odd}} RM(r', n+1)2^{(r'-r-1)/2}, & n-r \text{ even.} \end{cases} \tag{5.5}$$

For example,

$$RD_4 = \Lambda(-1,1) = 2\mathbb{Z}^4 + (4,1,4), \tag{5.6}$$

$$RE_8 = \Lambda(-1,2) = 4\mathbb{Z}^8 + 2(8,7,2) + (8,1,8). \tag{5.7}$$

The expressions in (5.1)-(5.7) suggest a simple algorithm for lattice VQ encoding. For example, from the expression for Λ_{16}, assume input vector $\mathbf{x}$, lattice codevector $\mathbf{y}$, and let

$\mathbf{c}_i, i = 1, \cdots, 32$, be the binary codewords in the (16, 5, 8) RM code. Then Λ_{16} VQ encoding can be accomplished using the following algorithm.

0) Let $\mathbf{y} = VQ_{2D_{16}}(\mathbf{x})$ denote lattice VQ using the $2D_{16}$ lattice.

1) For $i = 1$ to 32, form $\mathbf{y}_i = \mathbf{c}_i + VQ_{2D_{16}}(\mathbf{x} - \mathbf{c}_i)$.

2) Find $i^* = \text{argmin}_i \|\mathbf{x} - \mathbf{y}_i\|^2$.

3) The Λ_{16} codevector closest to $\mathbf{x}$ is $\mathbf{y}_{i^*}$.

Hence, Λ_{16} lattice VQ (LVQ) can be accomplished using 32 $2D_{16}$ lattice VQ operations. The RE_8 lattice used in wideband speech coding standards [23], [36] is also implemented based on this concept by, from (5.7), using just two $2D_8$ LVQ operations.

5.2 Sign/Magnitude Bit-plane Coding of Binary Lattices

Denote $\mathbf{x} = (x_1, \ldots, x_N)$ as a lattice codevector and $\mathbf{b} = (b_s, b_K, \ldots, b_1)^T$ as a binary (column) vector used to represent each x_i in sign/magnitude form, where b_s is the sign bit and $(b_K, \ldots, b_1)^T$ are magnitude bits, with b_1 denoting the least significant bit and b_K the most significant bit. Let $B(\mathbf{x})$ be the binary array of bits used to represent a lattice codevector $\mathbf{x}$.

More specifically,

$$B(\mathbf{x}) = \begin{bmatrix} b_{s,1} & b_{s,2} & \cdots & b_{2,N} \\ b_{K,1} & b_{K,2} & \cdots & b_{K,N} \\ \vdots & \vdots & \ddots & \vdots \\ b_{1,1} & b_{1,2} & \cdots & b_{1,N} \end{bmatrix}.$$

Also, denote $\mathbf{b}_j = (b_{j,1}, \ldots, b_{j,N})$ as a binary vector corresponding to the j^{th} bit-plane (row $(K+2-j)$ of $B(\mathbf{x})$), with Hamming weight of $\mathbf{b}_j$ denoted by $wt(\mathbf{b}_j)$.

Forney's representation describes a binary lattice as a union of a scaled integer lattice offset by a sum of scaled binary codevectors from certain RM codes. For example, the RE_8 lattice in (5.7) is made up of the union of 256 cosets of $4Z^8$, with coset representatives of the form $2\mathbf{c}_{1,i} + \mathbf{c}_{2,j}$, $i = 1, \cdots, 128$; $j = 1, 2$, where $\mathbf{c}_{1,i}$ are binary codewords in the $(8, 7, 2)$ RM code and $\mathbf{c}_{2,j}$ are codewords in the $(8, 1, 8)$ RM code. In sign-magnitude representation, the number of bit-planes that are lattice-defining thus corresponds to the 2-depth of the lattice. The lattices D_4 and E_8 have 2-depth 1, and so have only the least significant magnitude bit-plane, $\mathbf{b}_1$, as lattice defining. The lattices RE_8 and Λ_{16} have 2-depth 2, and so have both bit planes $\mathbf{b}_1$ and $\mathbf{b}_2$ as lattice-defining. The lattice imposes no restriction on the higher-level magnitude bit-planes.

Now, suppose that a large set of source samples is lattice vector encoded, and that the lattice codevectors are then represented in sign-magnitude form. If the lattice has 2-depth L, then all magnitude bit-planes above level L are unrestricted by the lattice structure, and hence any bit-plane coding algorithm can be used for compression (for example, using the

method in SPIHT [24] or JPEG2000 [61]). The bit-plane coding encodes one bit-plane at a time, from the most significant magnitude bit-plane down to the least significant bit-plane. All coefficients are initially classified as *insignificant*. As soon as the first "1" in the magnitude bits of a coefficient is encoded, the sign-bit is then immediately encoded and the coefficient is classified as *significant*. Significant coefficients already have their sign-bit encoded in the bit-stream, and so if an additional 1 occurs in the lower level bit-planes, there is no need to encode the sign-bit again. When the bit-plane coding reaches the L^{th} bit-plane (the most significant lattice-defining bit-plane), then possible bit patterns are restricted due to the lattice structure, and the bit-plane coding method should be suitably modified. Such a bit-plane coding method will yield an embedded bit-stream. If the bit-stream is truncated, a truncated representation of the lattice codevector can then be generated. Since a lattice affects the granular fidelity of the quantization, intuitively one expects that if the bit-stream is truncated at a bit-plane above the 2-depth of the lattice, then the truncated codevector representation should be little different from that of a truncated integer lattice representation, provided that the integer lattice has the same point density.

Assume that sign-bit, b_s, is encoded as 1 for negative components and as 0 for positive components, Methods to losslessly encode the lattice-defining bit-planes for D_4, RE_8, and Λ_{16} are as follows.

$\mathbf{D_4}$: The D_4 lattice is a subset of the $\mathbb{Z}^4$ lattice, with the restriction that for any codevector $\mathbf{x} \in D_4$, the sum of all components is even, $\sum_{i=1}^{4} x_i$ = even. This implies that in

sign-magnitude bit representation, $(\sum_{i=1}^{4} b_{1,i}) \text{mod}_2 = 0$ and the least significant magnitude bit-plane, $\mathbf{b}_1$, is thus lattice-defining (consistent with 2-depth of one for the D_4 lattice). There is no restriction on higher level magnitude bit-planes. There are exactly 2^3 least significant bit-plane 4-dimensional vectors, $\mathbf{b}_1$, and hence $\mathbf{b}_1$ can be encoded using only 3 bits.

RE_8 : From (5.7), it is clear that RE_8 codevectors have just two possible bit patterns for $\mathbf{b}_1$, which are $\mathbf{0}$ and $\mathbf{1}$ corresponding to coset $2D_8$ and $2D_8 + \mathbf{1}$, respectively. Also, using (5.7) it can be shown that for any codevectors $\mathbf{x} \in RE_8$, 1) when $\mathbf{b}_1 = \mathbf{0}$, $wt(\mathbf{b}_2)$ is even; 2) when $\mathbf{b}_1 = \mathbf{1}$ and $wt(\mathbf{b}_s)$ is odd, $wt(\mathbf{b}_2)$ is odd; and 3) when $\mathbf{b}_1 = \mathbf{1}$ and $wt(\mathbf{b}_s)$ is even, $wt(\mathbf{b}_2)$ is even. There are no restrictions on other bit-planes. The $\mathbf{b}_1$ and $\mathbf{b}_2$ bit-planes can be encoded as follows

a. Send 1 bit for $\mathbf{b}_1$. This determines if $\mathbf{x} \in 2D_8$ or $\mathbf{x} \in \{2D_8 + \mathbf{1}\}$.

b. If $\mathbf{x} \in 2D_8$, then encode $\mathbf{b}_2$ using 7 bits (followed by any necessary sign-bits for any components of $\mathbf{x}$ that become significant in bit-plane $\mathbf{b}_2$).

c. If $\mathbf{x} \in \{2D_8 + \mathbf{1}\}$, then first encode sign-bits not already encoded in $\mathbf{b}_s$ (since $\mathbf{b}_1 = \mathbf{1}$, a sign-bit is necessary for every component of $\mathbf{x}$). From the fully encoded sign bit-plane, determine $wt(\mathbf{b}_s)$. If $wt(\mathbf{b}_s)$ is even, then $\mathbf{b}_2$ must have even weight, and there are 128 possible even-weight bit patterns. If $wt(\mathbf{b}_s)$ is odd, then $\mathbf{b}_2$ must also have odd weight, and there are 128 possible odd-weight bit patterns. Conditioned on the weight of $\mathbf{b}_s$, $\mathbf{b}_2$ can be encoded using 7 bits.

A total of 8 bits can be used to encode the $\mathbf{b}_1$ and $\mathbf{b}_2$ bit-planes.

Λ_{16} : From (5.4), there are 2^5 possible $\mathbf{b}_1$ bit patterns, each corresponding to a codeword of the $(16, 5, 8)$ RM code. To consider possible bit patterns of $\mathbf{b}_2$, let n_1 be the number of positions of negative components of $\mathbf{x}$ that are coincident with the position of the 1's of $\mathbf{b}_1$. It can be shown that 1) when n_1 is even, $wt(\mathbf{b}_2)$ is even; and 2) when n_1 is odd, $wt(\mathbf{b}_2)$ is also odd. Using these facts, the lattice-defining bit-planes $\mathbf{b}_1$ and $\mathbf{b}_2$ can be encoded as follows.

a. Encode $\mathbf{b}_1$ using 5 bits. This determines if $\mathbf{x} \in 2D_{16}$ or a coset $\{2D_{16} + \mathbf{c}\}$, where $\mathbf{c} \neq \mathbf{0}$.

b. If $\mathbf{x} \in 2D_{16}$, then $\mathbf{c} = \mathbf{0}$, and encode $\mathbf{b}_2$ using 15 bits for the 2^{15} possible even-weight bit patterns. Follow this with any necessary sign-bits (for any components of $\mathbf{x}$ that become significant in bit-plane $\mathbf{b}_2$).

c. If $\mathbf{x} \in \{2D_{16} + \mathbf{c}\}$, where $\mathbf{c} \neq \mathbf{0}$, then first encode any sign-bits not already encoded corresponding to 1's in $\mathbf{c}$. This allows the decoder to determine n_1, and hence whether the weight of $\mathbf{b}_2$ is even or odd. There are exactly 2^{15} possible bit-patterns for each case and thus the $\mathbf{b}_2$ bit-plane can be encoded using 15 bits. Follow the encoding of $\mathbf{b}_2$ with any necessary sign-bits for the remaining components of $\mathbf{x}$ that become significant in bit-plane $\mathbf{b}_2$.

The $\mathbf{b}_1$ and $\mathbf{b}_2$ bit-planes of a Λ_{16} codevector can be encoded using 20 bits.

Comment: The D_4 bit-plane encoding is easily generalized to the D_n lattice for arbitrary

dimension n. The D_n lattice has 2-depth 1, and the $\mathbf{b}_1$ bit-plane is lattice-defining. There are 2^{n-1} even-weight allowed $\mathbf{b}_1$ bit-patterns in the D_n lattice and these can be encoded using $n-1$ bits.

5.3 Performance Comparison for Uniform Source

An embedded bit-stream can be decoded at any truncation point, and the decoded coefficients formed from a truncated sign-magnitude bit representation. In the following, we assume a large number of source samples have been quantized, and that bit-plane coding is used to construct the embedded bit-stream. The numerical results presented correspond to bit-stream truncation at the end of bit-planes. The performance of progressive decoding of bit-plane encoded $2D_4$, RE_8, and Λ_{16} lattice codevectors is shown in Figure 5.1-5.3 compared with that of $2\mathbb{Z}^n$ lattice VQ, which is used as the reference. The $2D_4$ lattice is used instead of the D_4 for a convenient comparison since the point density of $2D_4$ and the reference lattice, $2\mathbb{Z}^4$, are not too much different. The uniform source vectors of variance σ^2 are generated and used as the input to the lattice VQ, followed by bit-plane coding. The encoding rates are calculated from the number of encoded bits in the bit-stream at the end of each bit-plane. Ordinary bit-plane coding without entropy coding is performed for all bit-planes, except the lattice-defining bit-planes. The lattice-defining bit-planes are encoded using the the methods described in section 5.2. The truncation points for progressive bit-plane decoding are at the end of the bit-planes, and the truncated lattice codevectors are generated

to minimize the average squared error with respect to the undecoded lower bit-planes. For example, for the RE_8 lattice if the encoded bit-stream is truncated at the end of 4^{th} bit-plane ($\mathbf{b}_4$), then the reproduction of each component of RE_8 codevectors is $\hat{x}_i + \Delta_4/2 - 1$, where $\Delta_4 = 2^3$ and $\hat{x}_i$ is the decoded value based on the truncated bit-plane representation. This comes from the fact that with decoded output value n at the end of b_4 bit-plane, when other lower bit-planes are used, possible decoded value might be $n, n+1, \ldots, n+\Delta-1$.

From the figures, it can be seen that at low encoding rates (when truncation points are at the end of higher bit-planes) the progressive decoding performance of the $2D_4$, RE_8 and Λ_{16} lattices provides about the same SNR as the progressive decoding of the reference $2\mathbb{Z}^n$ lattice. As bit-plane decoding is performed further down to lower bit-planes, the performance drops a little compared with the $2\mathbb{Z}^n$ lattice, with the maximum SNR drop occurring at the end of the bit-plane before decoding the lattice-defining bit-planes. SNR drops of about 0.05 dB, 0.6 dB, and 0.3 dB, for the $2D_4$, RE_8, and Λ_{16} lattices, respectively, are observed. After the lattice-defining bit-planes are decoded, the granular gain of the lattice is realized. Note that in Figure 5.1, since the point density of $2D_4$ is half that of $2\mathbb{Z}^4$, the final rate of $2D_4$ lattice VQ encoding is less than that of $2\mathbb{Z}^4$ by 0.25 bits/sample. Similarly, in Figure 5.3 Λ_{16} has point density 16 times denser than that of $2\mathbb{Z}^{16}$ so that the final rate of Λ_{16} lattice VQ encoding is 0.25 bits/sample more compared with that of $2\mathbb{Z}^{16}$. When all bit-planes are decoded, the granular gain of the lattice is realized.

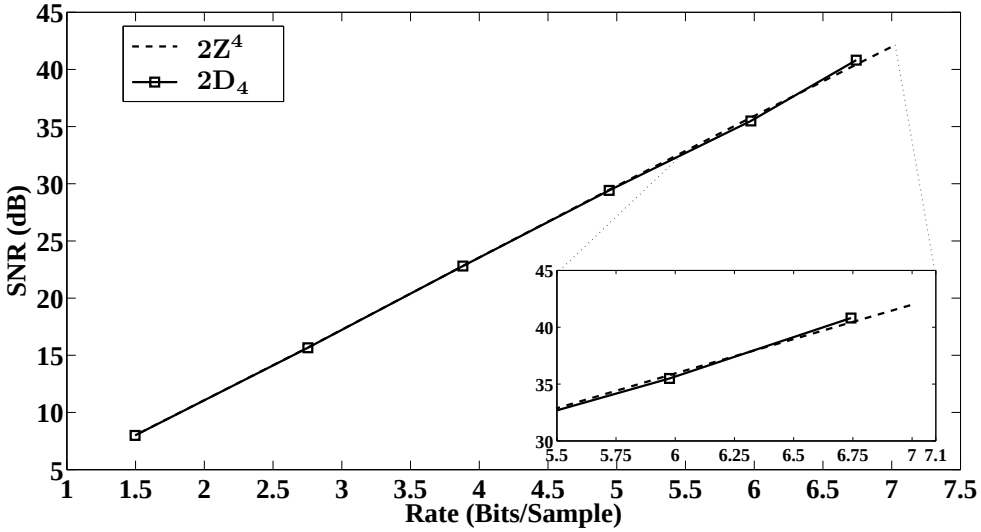


Figure 5.1: SNR comparison between $2D4$ and $2\mathbb{Z}^4$

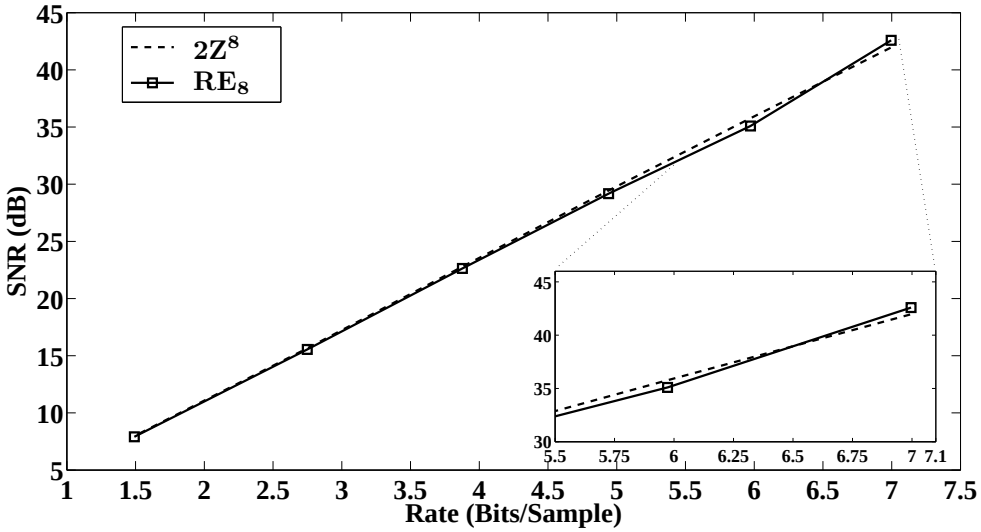


Figure 5.2: SNR comparison between RE_8 and $2\mathbb{Z}^8$

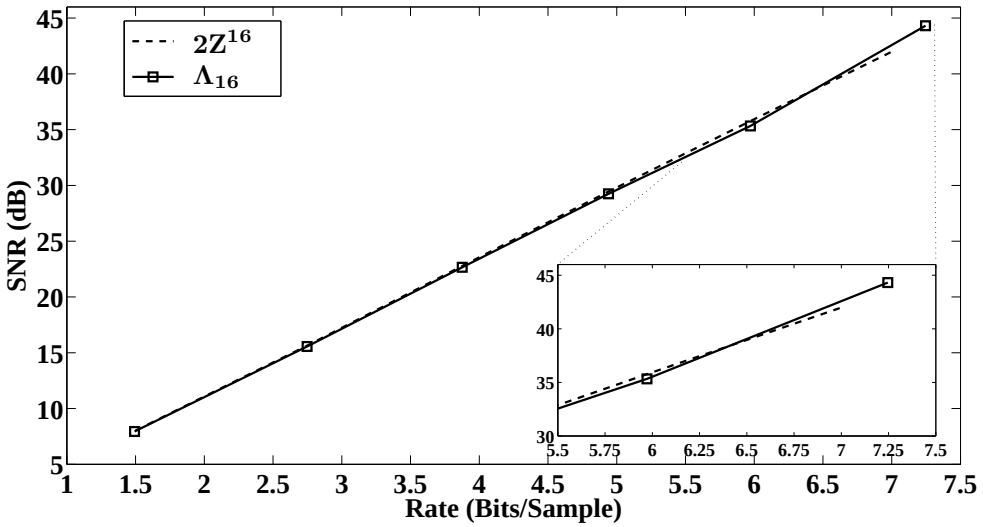


Figure 5.3: SNR comparison between Λ_{16} and $2\mathbb{Z}^{16}$

5.4 SPIHT Image Coding with Lattice VQ

As discussed in section 5.2, in sign-magnitude representation of lattice codevectors, there is no restriction on all magnitude bit-planes above the lattice-defining bitplanes, and hence any bit-plane coding algorithm can be used to encode those bit-planes. In this section, we will investigate the coding performance of the SPIHT algorithm [24] using D_4 lattice codevectors. The coding scheme is shown in Figure 5.4.

Vectors, $\mathbf{x}$, are formed consistent with the quadtree data structure used in the SPIHT encoding algorithm. That is, four neighboring pixels in the same subband are grouped together as a vector (see Figure 5.5). Each vector $\mathbf{x}$ is quantized using a D_4 lattice quantizer. The quantized version of $\mathbf{x}$, denoted by $\hat{\mathbf{x}}$, is then encoded using the SPIHT bit-plane coding algorithm, with a small modification as follows. For all bit-planes except the b_1 bit-plane

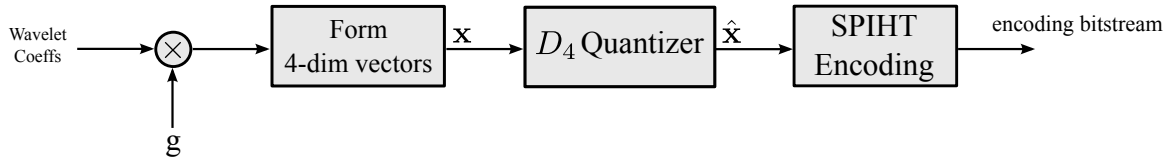


Figure 5.4: SPIHT with D_4 lattice codevectors

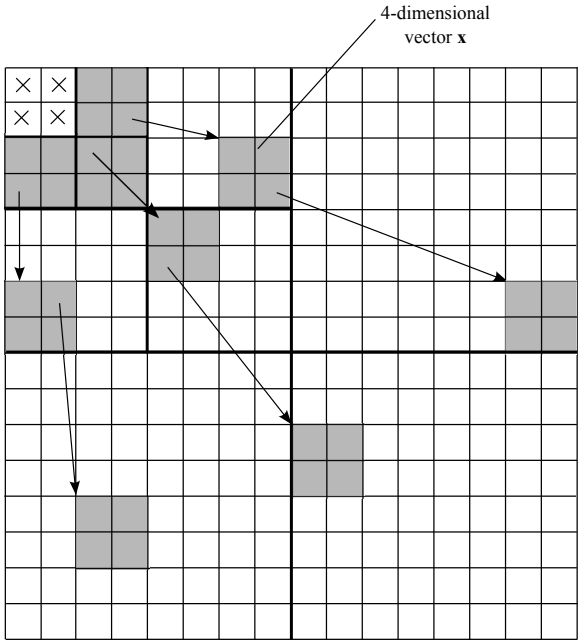


Figure 5.5: Forming 4-dimensional vector

(the lattice-defining bit-plane of D_4), the encoding algorithm is exactly the same as [24]. Each pixel is tested for significance (sorting pass) with respect to the threshold 2^n, for $n = n_{\max}, n_{\max} - 1, \ldots, 1$, where $n_{\max}$ is the largest magnitude bit-plane. For significant pixels, the refinement-bits are output when the encoding moves further down to each lower bit-plane (refinement pass). When the algorithm reaches the b_1 bit-plane, the significance test needs a little modification. Let $D(i, j)$ and $O(i, j)$ be the set of all descendants of node (i, j), and the set of immediate descendants of node (i, j), respectively. Also, denote $L(i, j) = D(i, j) - O(i, j)$. SPIHT [24] uses three ordered lists: A list of significant pixels (LSP), a list of insignificant pixels (LIP), and a list of insignificant sets (LIS). For entries in the LIS, there are two types: type A if it represents $D(i, j)$, and type B if it represents $L(i, j)$. The modified sorting pass of SPIHT for b_1 bit-plane is described in Algorithm 3, where $\hat{\mathbf{x}}(i, j)$ is the quantized D_4 codevector that has coordinate (i, j) as its component. Also, define

$$S_n(\kappa) = \begin{cases} 1, & \text{if } \max_{(i,j)\in\kappa} \|c_{i,j}\| \geq 2^n \\ 0, & \text{otherwise}, \end{cases} \tag{5.8}$$

where $c_{i,j}$ is the quantized wavelet coefficient at coordinate (i, j).

From the Algorithm 3, it can be observed that

- The pixels in $O(i, j)$ in the Algorithm 3 are matched with the way that 4-dimensional vectors are formed.
- Since the b_1 bit-plane is the last bit-plane, there is no need to update the LSP and LIP.

Algorithm 3 Modified Sorting Pass of SPIHT for b_1 bit-plane

```
for each entry (i, j) in LIP do
  if b_1 of x̂(i, j) is not encoded yet then
    encode b_1 of x̂(i, j) using 3 bits and output those bits followed by the sign of the
    components of x̂(i, j) that are significant at b_1 bit-plane;
  end if
end for
for each entry (i, j) in LIS do
  if entry is type A then
    output S_n(D(i, j));
    if S_n(D(i, j)) = 1 then
      encode b_1 of O(i, j) using 3 bits and output those bits followed by the sign of
      each components of O(i, j) that are significant at b_1 bit-plane;
      if L(i, j) ≠ φ then
        move (i, j) to the end of the LIS as type B;
      else
        remove (i, j) from the LIS.
      end if
    end if
  end if
  if entry is type B then
    output S_n(L(i, j));
    if S_n(L(i, j)) = 1 then
      add each (k, l) ∈ O(i, j) to the end of the LIS as type A;
      remove (i, j) from the LIS.
    end if
  end if
end for
```

- Encode $O(i, j)$ using 3 bits, corresponding to the encoding method for the lattice-defining bit-plane of the D_4 lattice , as described in Section 5.2.

- If any pixel in $O(i, j)$ is significant at the b_1 bit-plane, then the b_1 bit of all other pixels in $O(i, j)$ is also encoded.

- In sorting pass of entry (i, j) in LIP, $\mathbf{b}_1$ bit-plane encoded in this step might include b_1 bits of some pixels of $\hat{\mathbf{x}}(i, j)$ that are already significant. If that happens, b_1 bit of those significant pixels are not encoded again in refinement pass.

Although b_1 bits of some significant pixels might already be encoded together with the pixels in LIP, the refinement pass is still required for the case that all pixels in the same $\hat{\mathbf{x}}$ are already significant at bit-planes above the b_1 bit-plane. For refinement pass, instead of outputting the b_1 bit for each pixel, the $\mathbf{b}_1$ bit-plane is encoded using 3 bits, as described in Section 5.2. The other steps in the SPIHT algorithm are unchanged.

The simulation results using SPIHT with D_4 codevectors are shown in Figures 5.6 - 5.9 and compared with ordinary SPIHT. It can be seen that the PSNR of SPIHT with D_4 codevectors is about the same as that of ordinary SPIHT when bit-planes above the lattice-defining bit-plane are decoded, with a little drop in SNR (roughly < 0.3 dB) occurring at the end of b_2 bit-plane. This is consistent with the results in Section 5.3 for the bit-plane coding of a uniform source. When the b_1 bit-plane is decoded, the PSNR curve of SPIHT using D_4 codevectors moves above the curve for ordinary SPIHT. When the b_1 bit-plane is completely decoded, the PSNR of SPIHT using D_4 codevectors is larger than ordinary

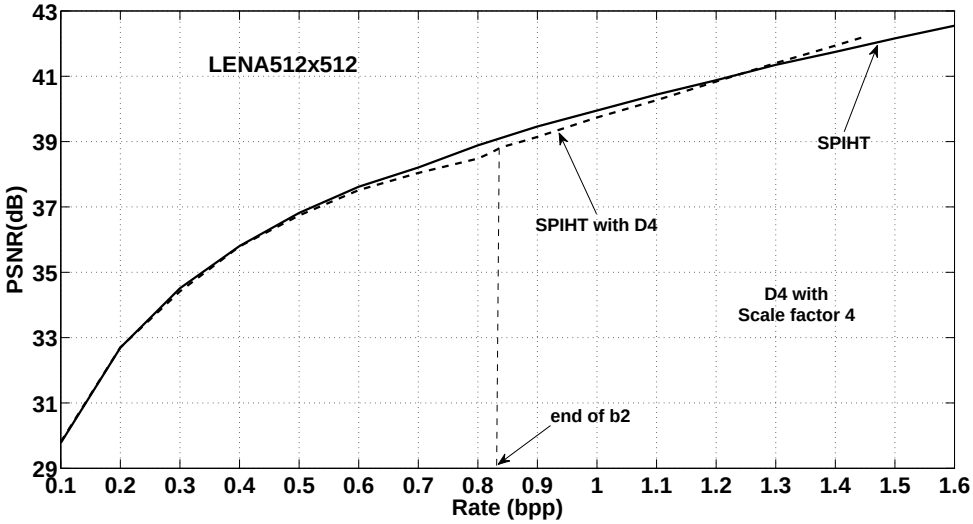


Figure 5.6: SNR of SPIHT with D_4 codevectors and scale factor $g = 1/4$ for *Lena* image

SPIHT by 0.3 dB, which is consistent with the granular gain of the D_4 lattice.

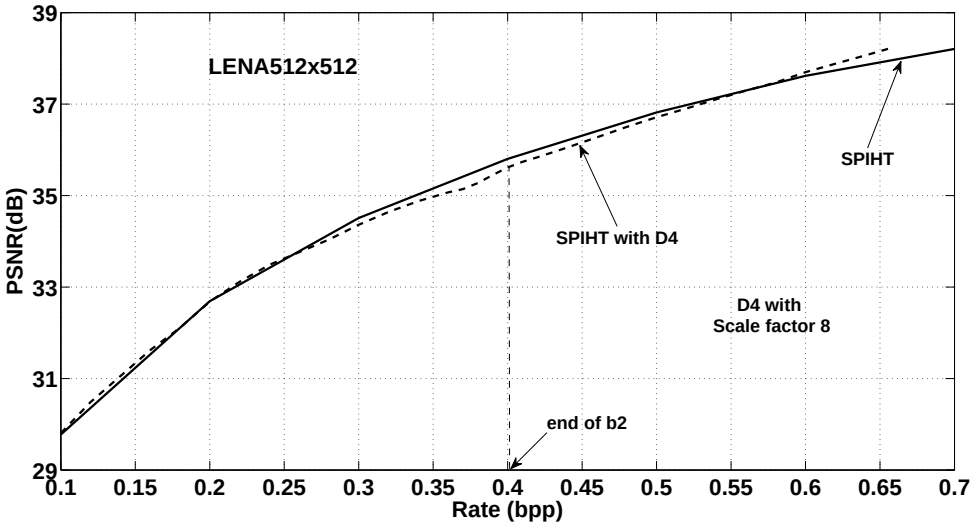


Figure 5.7: SNR of SPIHT with D_4 codevectors and scale factor $g = 1/8$ for *Lena* image

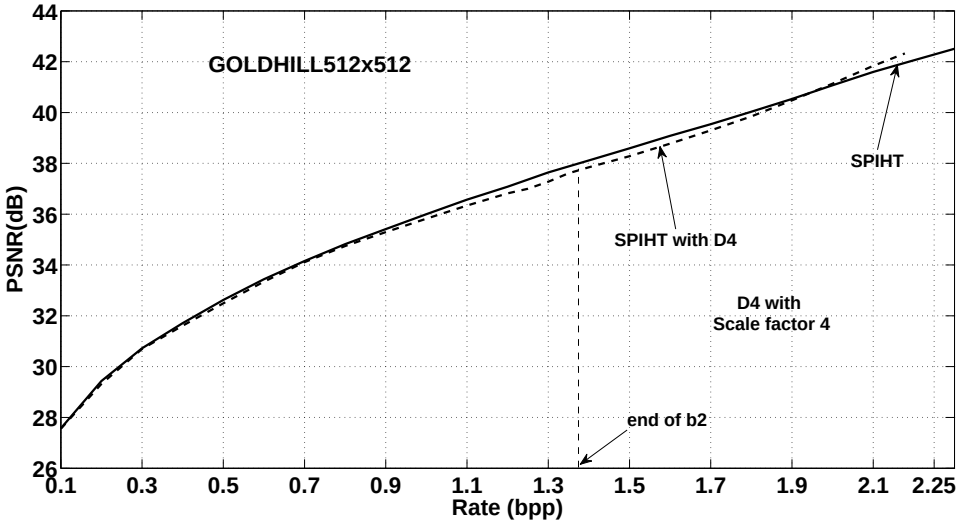


Figure 5.8: SNR of SPIHT with D_4 codevectors with scale factor $g = 1/4$ for *Goldhill* image

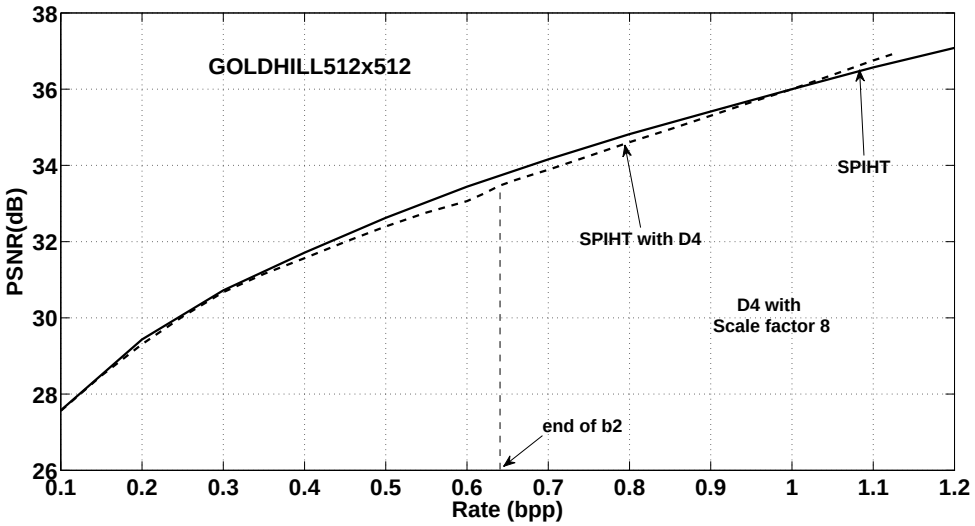


Figure 5.9: SNR of SPIHT with D_4 codevectors with scale factor $g = 1/8$ for *Goldhill* image

Chapter 6

Conclusion

In Chapter 2, the amplitude quantization implicit in algebraic or fixed excitation codebook search of ACELP is studied. We consider the potential improvement in coding performance possible by better quantization in the fixed codebook portion of the encoding algorithm. For the ACELP method of non-zero pulse position selection, it is concluded that small gains in SNR (no more than about 1 dB) are possible by refined amplitude quantization, but this increase in SNR requires significant increase in pulse amplitude quantization rate (as in [29]) and, even using ideal, unquantized, pulse amplitudes, results in very modest increase in perceptual speech quality, as measured by the PESQ [32] method of evaluating mean opinion score. Further, the ACELP method of pulse position selection implies a conditionally bimodal distribution on the selected non-zero pulse amplitudes, and for the MSE distortion measure the simple 1-bit uniform quantization of these amplitudes used in the ACELP algorithm is essentially optimum.

In Chapter 3, an accurate model of the SVQ performance in transform audio coding us-

ing Gaussian mixture models is developed. The Gaussian mixture model is used to model vectors of transform audio data and the EM algorithm is used to estimate GMM parameters. Two alternative methods are used to determine the mixture model parameters. A rate-distortion function based on GMM is developed and used to estimate the actual average encoding rate of SVQ. The effectiveness of the model is evaluated by comparing the estimated rate from the model with the average encoding rate of Z_8 lattice SVQ and with the RE_8 lattice VQ used in the AMR-WB+ standard. The simulation results show that the estimated rate from the model with four classes reasonably well models SVQ performance, especially at high rate (small distortion). At low and medium rates, a modified model partitions source vectors into insignificant (encoded as zero codevector) and significant (encoded as non-zero codevector) classes. The GMM rate-distortion function is used only to estimate the encoding rate for nonzero source vectors since the encoding rate for zero source vector is fixed and pre-defined. With the modification, the model accurately estimates SVQ performance for low, medium, and high encoding rates. The results also indicate that GMM rate-distortion modeling with $K = 4$ classes is sufficient to capture the available classification gain of the mixture model for transform audio coding.

In Chapter 4, we propose an algorithm based on LPVQ and LSVQ for image coding. The proposed algorithm partitions the subband (or wavelet) image into blocks of various sizes by comparing ℓ_1 or ℓ_2 block energy, respectively, with the thresholds defined according to complexity constraints on enumeration encoding of lattice points. We show that based on the proposed algorithm the energy clustering can be exploited effectively. Coding

blocks or sets of coefficients using LVQ also allows the possible improvement in SNR by using better covering lattices, such as D_4 or E_8. From the simulation, the proposed algorithm provides a better performance compared to the SPIHT algorithm without arithmetic coding, and is competitive with SPIHT using arithmetic coding, with slightly better performance at low or moderate encoding rates for LPVQ.

In Chapter 5, we describe the bit-plane coding of lattice VQ codevectors, focusing on the D_4, RE_8, and Λ_{16} lattices. By using Forney's representation of the lattices, it can be seen that in sign-magnitude form some least-significant bit-planes are constrainted to have some specified bit patterns due to the structure of the lattices, while there are no restrictions on other more significant bit-planes. Those least significant bit-planes are called "lattice-defining" bit-planes. Hence, lattice codevectors can be bit-plane encoded using any convenient bit-plane coding method for the more significant bit-planes, with modification required only for the lattice-defining bit-planes. Three simple methods for bit-plane coding on the lattice-defining bit-planes of D_4, RE_8, and Λ_{16} are presented. Using standard bit-plane coding together with the proposed method to handle the lattice-defining bit-planes, simulations for a uniform source demonstrate that the bit-plane coding of lattice codevectors can provide about the same performance as that of the integer lattice if the bit-stream is truncated. When the entire bit-stream is decoded, the granular gain of the lattice is realized.

The SPIHT algorithm image coding using D_4 lattice codevectors is investigated. The modifications necessary for the sorting and refinement passes of the SPIHT algorithm for encoding the b_1 bit-plane of the D_4 codevectors are discussed. The simulation results show

that when a truncated bit-stream corresponding to bit-planes above the lattice-defining bit-planes is decoded, the performance of the SPIHT algorithm using D_4 codevectors is about the same as that of ordinary SPIHT, with a small drop in PSNR as the bit-plane decoding moves further down to the end of b_2 bit-plane. When the b_1 bit-plane is decoded, the performance of the SPIHT algorithm using D_4 codevectors gradually moves close to ordinary SPIHT, and when the b_1 bit-plane is completely decoded, the granular gain of the D_4 lattice is realized.

Bibliography

[1] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, 1948.

[2] C. E. Shannon, "Coding theorems for a discrete source with a fidelity criterion," *IRE Nat Conv. Rec*, vol. 4, pp. 142–163, 1959.

[3] A. Gersho and R. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Pub, 1992.

[4] T. Berger, "Optimum quantizers and permutation codes," *Information Theory, IEEE Transactions on*, vol. 18, pp. 759 – 765, 1972.

[5] Y. Linde, A. Buzo, and R. Gray, "An algorithm for vector quantizer design," *Communications, IEEE Transactions on*, vol. 28, pp. 84 – 95, 1980.

[6] S. Lloyd, "Least squares quantization in PCM," *Information Theory, IEEE Transactions on*, vol. 28, pp. 129 – 137, 1982.

[7] J. H. Conway and N. J. A. Sloane, "Voronoi regions of lattices, second moments of polytopes, and quantization," *Information Theory, IEEE Transactions on*, vol. 28, no. 2, pp. 211 – 226, Mar 1982.

[8] T. R. Fischer, "A pyramid vector quantizer," *Information Theory, IEEE Transactions on*, vol. 32, pp. 568 – 583, 1986.

[9] T. R. Fischer, M. W. Marcellin, and M. Wang, "Trellis-coded vector quantization," *Information Theory, IEEE Transactions on*, vol. 37, pp. 1551 –1566, 1991.

[10] N. Farvardin and J. Modestino, "Optimum quantizer performance for a class of non-Gaussian memoryless sources," *Information Theory, IEEE Transactions on*, vol. 30, pp. 485 – 497, 1984.

[11] P. A. Chou, T. Lookabaugh, and R. M. Gray, "Entropy-constrained vector quantization," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 37, pp. 31 –42, 1989.

[12] M. Schroeder and B. S. Atal, "Code-excited linear prediction (CELP) high quality speech at very low bit rates," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '85.*, 1985, vol. 10, pp. 937 – 940.

[13] M. Vetterli and J. Kovačević, *Wavelets and subband coding*, Prentice-Hall, Englewood Cliffs, NJ, 1995.

[14] J. H. Conway and N. J. A. Sloane, "Fast quantizing and decoding and algorithms for lattice quantizers and codes," *Information Theory, IEEE Transactions on*, vol. 28, no. 2, pp. 227 – 232, Mar 1982.

[15] J. H. Conway and N. J. A. Sloane, "A fast encoding method for lattice codes and quantizers," *Information Theory, IEEE Transactions on*, vol. 29, no. 6, pp. 820 – 824, Nov 1983.

[16] A. Gersho, "Asymptotically optimal block quantization," *Information Theory, IEEE Transactions on*, vol. 25, no. 4, pp. 373 – 380, Jul 1979.

[17] R. M. Gray and D. L. Neuhoff, "Quantization," *Information Theory, IEEE Transactions on*, vol. 44, no. 6, pp. 2325 –2383, Oct 1998.

[18] M. Xie and J. P. Adoul, "Embedded algebraic vector quantization (EAVQ) with application to wideband audio coding," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Atlanta, GA, USA*, 1996, vol. 1, pp. 240–243.

[19] M. Barlaud, P. Sole, T. Gaidon, M. Antonini, and P. Mathieu, "Pyramidal lattice vector quantization for multiscale image coding," *IEEE Trans. on Image Processing*, vol. 3, no. 4, pp. 367 –381, 1994.

[20] *Adaptive Multi-Rate(AMR) Speech Codec;Transcoding functions*, 3GPP TS 26.09, 2003.

[21] *Digital telecommunications system (Phase 2+);Adaptive Multi-Rate (AMR) speech*, ETSI EN 301 704 V.7.2.1.

[22] G. D. Forney, "Coset codes–Part II: Binary lattices and related codes," *IEEE Trans. on Inform. Theory*, vol. 34, pp. 1152 – 1153, 1988.

[23] *Extend AMR Wideband Codec; Transcoding functions*, 3GPP TS 26.29, 2005.

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

[25] W. Patchoo, T. R. Fischer, Changho Ahn, and Sangwon Kang, "Analysis of amplitude quantization in ACELP excitation coding," in *Data Compression Conference (DCC), 2010*, Mar 24-26 2010, pp. 550 –550.

[26] B. S. Atal and M. R. Schroeder, "Stochastic coding of speech signal at very low bit rates," in *Acoustics, Speech, and Signal Processing, 1994. ICASSP-94., 1994 IEEE International Conference on*, May 1984, pp. 1610 – 1613.

[27] *Coding of speech at 16 kbit/s using low-delay code excited linear prediction*, ITU G.728.

[28] J. P. Adoul, P. Mabilleau, M. Delprat, and S. Morissette, "Fast CELP coding based on algebraic codes," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'87.*, 1987, pp. 1957–1960.

[29] M. D. Meuleneire, M. Gartner, S. Schandl, and H. Taddei, "An enhancement layer for ACELP coder," in *Multimedia Signal Processing, 2006 IEEE 8th Workshop on*, oct 2006, pp. 124 –127.

[30] F. Itakura, "Line spectral representation of linear predictive coefficients of speech signals," *J. Acoust. Soc. Amer*, pp. Supplement no. 1,S35, 1975.

[31] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, John Wiley and sons, 2006.

[32] *Perceptual evaluation of speech quality PESQ: An objective method for end-to-end speech quanlity assessment of narrow-band telephone networks and speech codecs*, ITU-T Rec. P.862 (02/2001).

[33] W. Patchoo and T. R. Fischer, "Gaussian-mixture modeling of lattice-based spherical vector quantization performance in transform audio coding," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, Mar 14-19 2010, pp. 373 –376.

[34] W. Patchoo and T. R. Fischer, "Rate-distortion modeling of spherical vector quantization performance in transform excitation audio coding," in *Electrical Engineering/Electronics Computer Telecommunications and Information Technology (ECTICON), 2010 International Conference on*, May 19-21 2010, pp. 1075 –1079.

[35] W. Patchoo and T. R. Fischer, "Rate-distortion modeling of spherical vector quantization performance in transform excitation audio coding," *ECTI Transaction EEC (ECTI Transaction EEC) Special Section on Papers Selected from ECTI-CON 2010*, pp. 49 – 55, 2011.

[36] *G.729 based Embedded Variable bit-rate coder: An 8-32 kbits/s scalable wideband coder bitstream interoperable with G.729*, ITU-T G.729.1, 2006.

[37] *Information Technology Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s Part 3: Audio*, ISO/IEC 11172.3, 1993.

[38] K. Sayood, *Introduction to Data Compression*, Morgan-Kaufmann, 2006.

[39] S. Ragot, B. Bessette, and R. Lefebore, "Low-complexity multi-rate lattice vector quantization with application to wideband TCX speech coding at 32 kbit/s," in *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, 2004, vol. 1, pp. I – 501–4 vol.1.

[40] R. M. Gray, "Gauss mixture vector quantization," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, 2001, pp. 1769 – 1772.

[41] D. J. Sakrison, "A geometrice treatment of the source coding of Gaussian random variable," *Information Theory, IEEE Transactions on*, pp. 481 – 486, 1968.

[42] T. R. Fischer and R. M. Dicharry, "Vector quantizer design for memoryless Gaussian, Gamma and Laplacian sources," *Information Theory, IEEE Transactions on*, pp. 1065 – 1069, 1984.

[43] R. Redner and H. Walker, "Mixture densities, maximum likelihood and the EM algorithm," *SIAM review*, vol. 26, no. 2, pp. 195–239, 1984.

[44] J. H. Piater, "Mixture models and expectation-maximization," *Lecture at ENSIMAG, May*, 2002.

[45] H. W. Sorenson and D. L. Aspach, "Recursive Bayesian estimation using Gaussian sums," *Automatica*, pp. 465 – 479, 1971.

[46] B. H. Juang, L. R. Rabiner, S. E. Levinson, and Sondhi, "Recent development in the application of hidden Markov models to speaker-independent isolated word recognition," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, 1985, pp. 9 – 12.

[47] P. Hedelin and J. Skoglund, "Vector quantization based on Gaussian mixture models," *Speech and Audio Processing, IEEE Transactions on*, pp. 385 – 401, 2000.

[48] D. Zhao, J. Samuelsson, and M. Nilsson, "On entropy-constrained vector quantization using Gaussian mixture models," *Communications, IEEE Transactions on*, pp. 2094 –2104, 2008.

[49] J. K. Su and R. M. Mersereau, "Coding using Gaussian mixture and generalized Gaussian models," in *Image Processing, 1996. Proceedings., International Conference on*, 1996, pp. 217 –220.

[50] L. Gullimot, Y. Gaudeau, S. Moussaoui, and J. M. Moureaux, "An analytical Gamma mixture based rate-distortion model for lattice vector quantization," in *EUSIPCO*, 2006, pp. 217 –220.

[51] P. Loyer, J. M. Moureaux, and M. Antonini, "Lattice codebook enumeration for generalized Gaussian source," *Information Theory, IEEE Transactions on*, pp. 521 – 528, 2003.

[52] D. Peel and G. Maclahlan, *Finite Mixture Models*, McGraw-Hill, 2002.

[53] A. Papoulis, S. U. Pillai, and S. Unnikrishna, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill New York, 2002.

[54] Y. Z. Huang, D. B. O'Brian, and R. M. Gray, "Classification of features and images using Gauss mixtures with VQ clustering," in *Data Compression Conference (DCC), 2004*, March 2004, pp. 13 – 21.

[55] T. D. Lookabaugh and R. M. Gray, "High-resolution quantization theory and the vector quantizer advantage," *Information Theory, IEEE Transactions on*, vol. 35, no. 5, pp. 1020 –1033, sep 1989.

[56] N. Jayant and P. Noll, *Digital coding of waveforms: principles and applications to speech and video*, Prentice Hall Professional Technical Reference, 1990.

[57] J. H. Conway and N. J. A. Sloane, *Sphere packings, Lattices, and Groups*, Springer Verlag, 1999.

[58] J. W. Woods and S. D. O'Neil, "Subband coding of images," *IEEE Trans. on Acoust. Speech and Signal Processing*, vol. 34, pp. 1278 – 1288, Oct 1986.

[59] P. Mathieu M. Antonini, M. Barlaud and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. on Image Processing*, vol. 1, no. 2, pp. 205 – 220, 1992.

[60] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. on Signal Processing*, vol. 41, no. 12, pp. 3445 – 3462, 1993.

[61] D. Taubman and M. W. Marcellin, *JPEG2000: Image compression fundamentals, standards and practice*, Springer, 2001.

[62] *Information technology – JPEG 2000 image coding system: Core coding system*, ISO/IEC 15444-1, 2004.

[63] N. Tanabe and N. Farvardin, "Subband image coding using entropy-coded quantization over noisy channels," *IEEE J. Sel. Areas Commun.*, vol. 10, pp. 926 – 943, June 1992.

[64] W. A. Pearlman, A. Islam, N. Nagaraj, and A. Said, "Efficient, low-complexity image coding with a set-partitioning embedded block coder," *IEEE Trans. on Circuits Syst. Video Technol.*, vol. 14, no. 11, pp. 1219 – 1235, 1996.

[65] Z. Mohd-Yusof and T. R. Fischer, "An entropy-coded lattice vector quantizer for transform and subband image coding," *IEEE Trans. on Image Processing*, vol. 5, no. 2, pp. 289 –298, 1996.

[66] J. Andrew, "A simple and efficient hierarchical image coder," in *Acoustics, Speech, and Signal Processing, 1997. ICASSP '97. Proceedings., 1997 IEEE International Conference on*, 1997, pp. 658 –661.

[67] C. Chrysafis, A. Said, A. Drukarev, A. Islam, and W. A. Pearlman, "SBHP - a low complexity wavelet coder," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings., 2000 IEEE International Conference on*, 2000, pp. 2035 –2038.

[68] I. H. Witten, R.M. Neal, and J.G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, pp. 520 – 540, 1987.

[69] Zheng Gao, Ben Belzer, and John Villasenor, "A comparison of the Z, E8, and Leech lattices for quantization of low-shape-parameter generalized Gaussian sources," *Signal Processing Letters, IEEE*, vol. 2, no. 10, pp. 197 –199, Oct 1998.

[70] G. D. Forney., "Coset codes–Part I: Introduction and geometrical classification," *IEEE Trans. on Inform. Theory*, vol. 34, pp. 1123–1151, 1988.

[71] S. Lin and J. D. Costello, *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, Englewood Cliffs, NJ, 2004.