INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Lossy Compression of Scientific Data via Wavelets and Vector Quantization

by

Jill R. Goldschneider

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

Doctor of Philosophy

University of Washington

1997

Approved by Eve A. Riskin

(Chairperson of Supervisory Committee)

Richard E. Ladner

Program Authorized
to Offer Degree Electrical Engineering

Date May 29, 1997

In presenting this dissertation in partial fulfillment of the requirements for the Doctoral Degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of the dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U. S. Copyright Law. Requests for copying or reproduction of the dissertation may be referred to University Microfilms, 1490 Eisenhower Place, P. O. Box 975, Ann Arbor, MI 48106, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature _____

Date _____

University of Washington

Abstract

Lossy Compression of Scientific Data via Wavelets and Vector Quantization

by Jill R. Goldschneider

Chairperson of Supervisory Committee: *Professor Eve A. Riskin*
Department of Electrical Engineering

The high volume of scientific data to be collected from Earth Observing System instruments will require large transmission bandwidth and storage capabilities for archiving. Lossy compression techniques can be used to compress the data, although by definition the original data cannot be recovered. The compression techniques used must ensure that the compressed data have both high visual fidelity and that calculations using the compressed data yield accurate results. In this dissertation, lossy compression algorithms based on the wavelet transform and vector quantization are developed. Optimal bit-allocation algorithms based on pruned tree-structured vector quantization techniques are developed for the discrete wavelet transform and for the more general discrete wavelet packet transform. These algorithms systematically find all quantizers on the lower convex hull of the rate-distortion curve, while for the wavelet packet transform, simultaneously selecting the best basis. The effects of such compression on USC database images is examined for benchmarking purposes, and the benefits of compressing ocean science acoustic sonar data used for the scientific analyses of target detection and temperature analysis are studied.

TABLE OF CONTENTS

List of Figures iv

List of Tables viii

Chapter 1: Introduction 1

Chapter 2: Vector Quantization 5

2.1 Compression Measures 5

2.2 Full Search Vector Quantization 6

2.3 Tree Structured Vector Quantization 7

2.4 Vector Quantizer Design: The Generalized Lloyd Algorithm 9

2.5 Pruned Tree Structured Vector Quantization 15

2.5.1 The GBFOS Pruning Algorithm 16

2.5.2 The Recursive Optimal Pruning Algorithm 23

2.6 Implementation 25

Chapter 3: Wavelets 35

3.1 Discrete Orthonormal Wavelets - One Dimension 36

3.2 Discrete Orthonormal Wavelets in One Dimension after Quantization: Rate, Distortion, and Entropy 43

3.3 Discrete Orthonormal Wavelets - Two Dimensions 46

3.4 Discrete Orthonormal Wavelets in Two Dimensions after Quantization: Rate, Distortion, and Entropy 48

3.5 Wavelet Packets 49
3.6 Implementation 50

Chapter 4: Wavelets and Quantization **51**
4.1 Wavelet Compression 51
4.1.1 Wavelet Scalar Quantization 52
4.1.2 Wavelets and Vector Quantization 53
4.1.3 Wavelets and Quantization for Video Compression 55
4.1.4 Wavelet Packet Compression 56
4.2 Compression using Wavelets and PTSVQ 57
4.2.1 Bit Allocation Using GBFOS for Wavelets and PTSVQ 59
4.2.2 Bit Allocation Using ROPA for Wavelets and PTSVQ 62
4.2.3 Optimal Bit Allocation for Wavelet Packets and PTSVQ . . . 65
4.2.4 Implementation 79

Chapter 5: Compression of USC Images **87**
5.1 Wavelet Bit Allocation Results for the USC Images 87
5.2 Wavelet Packet Bit Allocation Results for the USC Images 91
5.3 Conclusions 94

Chapter 6: Compression of Acoustic Data **108**
6.1 The Benthic Acoustic Measurement System and Data 110
6.2 Properties of the BAMS Data 111
6.3 Scientific Objectives of the BAMS Data 113
6.4 Evaluation of the Effects of Acoustic Data Compression 117
6.4.1 Experiment 1: Compression of Complex Acoustic Data 119
6.4.2 Experiment 2: Compression of Polar Acoustic Data 120
6.5 Results and Discussion 122

6.5.1 Effects of Compression on the Raw Data 122
6.5.2 Effects of Compression on an Individual Ping 123
6.5.3 Effects of Compression on the Backscatter Plots 126
6.5.4 Effects of Compression on the Correlation and Sound Speed Measurements 128
6.6 Conclusions and Future Work 133

Chapter 7: Conclusions and Suggestions for Future Work **154**

Bibliography **158**

Appendix A: Embedded Multilevel Error Diffusion **167**
A.1 Embedded Multilevel Error Diffusion 170
A.2 Color Palette Organization 173
A.2.1 Color Palette Organization for LSB Embedding 175
A.2.2 Color Palette Organization for MSB Embedding 176
A.3 Results 178
A.3.1 Results of embedding in the LSB's 179
A.3.2 Results of embedding in the MSB's 180
A.3.3 Limitations to Color Palette Use 183
A.3.4 Limitations to Embedded Error Diffusion 184
A.4 Conclusions 185

LIST OF FIGURES

1.1 Direct compression and transform compression. 2

2.1 Full search vector quantization. 7
2.2 Tree structured vector quantization. 8
2.3 TSVQ construction. 28
2.4 TSVQ design by greedy growing. 29
2.5 Voronoi regions for VQ, TSVQ, and PTSVQ. 30
2.6 GBFOS codebooks. 31
2.7 TSVQ design by GBFOS pruning. 32
2.8 GBFOS pruning. 33
2.9 ROPA pruning. 34

3.1 The Haar wavelet and scaling functions. 38
3.2 Multiresolution spaces. 39
3.3 Multiresolution decomposition of a Doppler signal. 40
3.4 The discrete wavelet transform of a Doppler signal. 42
3.5 The discrete wavelet transform in two dimensions. 47
3.6 Discrete wavelet and wavelet packet decomposition. 50

4.1 An image compression example. 58
4.2 Discrete wavelet transform bit allocation. 59
4.3 GBFOS bit allocation. 80
4.4 ROPA bit allocation. 81

4.5 Wavelet packet bit allocation. 82
4.6 WPT bit allocation simplifications. 83
4.7 The rate-distortion curves of a node before and after merging. 84
4.8 The previous and next slopes examined for WPT bit allocation. . . . 84
4.9 Node merging for WPT bit allocation during initialization. 85
4.10 Node merging for the main WPT bit allocation algorithm. 86

5.1 The distribution of wavelet coefficients among image subbands. 96
5.1 *(Continued)* 97
5.1 *(Continued)*. 98
5.2 Growing and pruning results for TSVQ's from different subband sources. 99
5.3 DWT/PTSVQ bit allocation results as a function of DWT depth. . . 100
5.4 DWT/PTSVQ bit allocation results as a function of vector dimension. 101
5.5 DWT/PTSVQ compression on the test image "Lenna." 101
5.6 WPT/PTSVQ bit allocation results as a function of WPT depth. . . 102
5.7 WPT/PTSVQ bit allocation results as a function of vector dimension. 103
5.8 WPT/PTSVQ test image results as a function of WPT depth. 104
5.9 WPT/PTSVQ test image results as a function of vector dimension. . 105
5.10 A comparison of DWT/PTSVQ and WPT/PTSVQ bit allocation. . . 106
5.11 Numerical bit allocations for a depth 3 WPT using dimension 4 PTSVQ's. 106
5.12 WPT/PTSVQ compression on the test image "Lenna." 107

6.1 Acoustic data targets. 135
6.2 Scan 5, ping 42. 136
6.2 *(Continued)*. 137
6.3 Scan 5, ping 84. 138
6.3 *(Continued)*. 139

6.4 Sound speed as a function of temperature. 140
6.5 SNR versus entropy for acoustic scan 5. 141
6.6 The decrease in standard deviation versus entropy for ping 42. 142
6.7 The decrease in standard deviation versus entropy for ping 84. 142
6.8 Simple averaging, ping 84 at 2.62 bits per complex value. 143
6.9 Experiment 1, PTSVQ, ping 84 at 6.48 bits. 144
6.10 Experiment 1, PTSVQ, ping 84 at 3.56 bits. 144
6.11 Experiment 1, wavelets/PTSVQ, ping 84 at 4.39 bits. 145
6.12 Experiment 1, wavelets/PTSVQ, ping 84 at 2.25 bits. 145
6.13 Experiment 2, PTSVQ, ping 84 at 7.82 bits. 146
6.14 Experiment 2, PTSVQ, ping 84 at 6.82 bits. 146
6.15 Experiment 2, wavelets/PTSVQ, ping 84 at 2.67 bits. 147
6.16 Experiment 2, wavelets/PTSVQ, ping 84 at 0.69 bits. 147
6.17 Original acoustic data scatter plots of scan 5. 148
6.18 Experiment 2, scattering plot results for wavelets/PTSVQ. 148
6.19 Experiment 2, scattering plot results for PTSVQ. 149
6.20 Experiment 1, scattering plot results for wavelets/PTSVQ. 149
6.21 Experiment 1, scattering plot results for simple averaging. 150
6.22 Experiment 1, scattering plot results for PTSVQ. 150
6.23 Original acoustic correlation and sound speed plots. 151
6.24 The mean absolute difference in correlation. 152
6.25 The mean absolute difference in sound speed measurements. 153

A.1 Binary error diffusion algorithm. 167
A.2 Embedded multilevel error diffusion. 171
A.3 Scalar quantizers used for grayscale embedded binary error diffusion. 173
A.4 Voronoi regions of a 256-word color palette. 176

A.5 Voronoi regions of the subset palettes used for LSB embedding. . . . 177
A.6 Voronoi regions of the subset palettes used for MSB embedding. . . . 178
A.7 The color palettes ordered for LSB and MSB embedding. 179
A.8 The 8-bit color error diffused Lenna image. 180
A.9 The 8-bit color embedded error diffused images. 181
A.10 Artifacts seen in enlarged portions of the embedded images. 182

LIST OF TABLES

2.1 GBFOS pruning initialization. 21
2.2 GBFOS pruning. 22
2.3 ROPA pruning. 26
2.4 ROPA pruning subroutine **min_ropa_node**. 26

4.1 GBFOS bit allocation. 62
4.2 ROPA bit allocation. 64
4.3 Wavelet packet bit allocation initialization. 76
4.4 Wavelet packet bit allocation. 77
4.5 Wavelet packet bit allocation subroutine **min_slope_node**. 77
4.6 Wavelet packet bit allocation subroutine **update**. 78

5.1 Codebook densities for DWT/PTSVQ bit allocation via GBFOS. . . 90
5.2 Additional codebooks found by ROPA for DWT/PTSVQ bit allocation. 90
5.3 Codebook densities for WPT/PTSVQ bit allocation. 93

6.1 Acoustic data range. 112
6.2 Acoustic data characteristics. 113
6.3 Error tolerances for acoustic data processing 119
6.4 Compression results of acoustic scatter plots. 127
6.5 Acoustic correlation measurement results due to compression. 131
6.6 Acoustic sound speed measurement results due to compression. 132

ACKNOWLEDGMENTS

I am deeply grateful to my advisor, Professor Eve Riskin, for introducing the subject of compression to me with such clarity and insight, and for her continual and patient guidance and support. Professor Riskin has helped to create a stimulating environment here for the study of broader topics of data and image compression. I have been inspired by her rigor and attention to detail and look forward to future collaborations with her.

For their support, comments, suggestions, and careful reading of my work, I am grateful to the members of my supervisory committee: Professors Richard Ladner, Murat Azizoğlu, and Linda Shapiro, and Dr. Andrew Bruce.

I would like to express my appreciation to Professor Darrell Jackson for his assistance and advice with research involving the acoustic backscattering experiments and for providing the data and analytical tools. I would also like to thank Dr. Don Percival for advice with the acoustic compression experiments, and Dr. Martin Siderius for help with initial data analysis.

Although not the main part of my dissertation, I had the opportunity to conduct research in the field of multitoning. Dr. Ping Wah Wong of Hewlett-Packard Laboratories assisted and advised me (from afar) regarding research involving binary and color halftoning. He carefully read my work and made many useful suggestions. Dr. Nicole Diaz greatly facilitated this work by providing the MCPM code used for the color halftoning experiments.

My husband, Ted Stern, provided many useful LaTeX and emacs tricks, and was gracious, patient, and loving throughout my studies. To my family I am deeply grateful for their patient tolerance and loving support these many years.

Finally, I would like to thank my labmates Srini, Holly, Todd, and Alex as well as the Electrical Engineering faculty, students, and staff for supporting my education, providing excellent research facilities, and for being a great group of knowledgeable and fun people to work with.

This work was supported in part by a NASA Graduate Student Fellowship in Global Change Research 4112-GC93-0191, by a research grant from Hewlett-Packard Laboratories, and by StatSci, a division of MathSoft Inc.

DEDICATION

To Ted.

Chapter 1

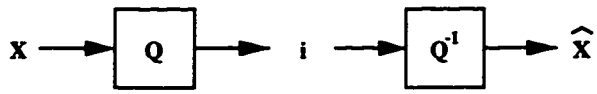
INTRODUCTION

The first Earth Observing System (EOS) platform is scheduled to be deployed in the late 1990's. It is expected to generate hundreds of gigabytes of data each day. To be useful, these enormous amounts of data must be well managed. Transmission and storage of these data would be greatly eased if the data could be effectively compressed.

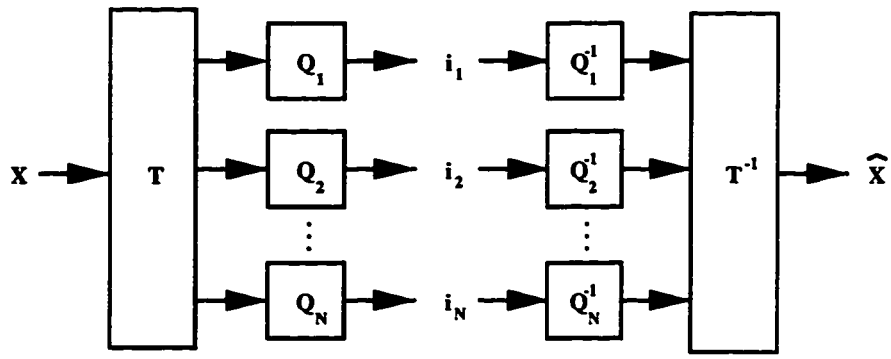
Two forms of data compression are possible. The first is *lossless*, *noiseless*, or *reversible* compression. Its advantage is that the original information can be completely recovered from the compressed data. Lossless compression is usually limited to compression ratios of 4:1 or less, and even these ratios are achievable only on sources that have considerable structure. For images and scientific data, lossless compression ratios are usually much lower, frequently 2:1 or less. Codes designed for noiseless compression may even cause data expansion. The common Ziv-Lempel file compression tool used in the Unix `compress` utility occasionally fails in this manner.

The second type of data compression is *lossy*, *noisy*, or *irreversible* compression. Lossy compression has the advantage that it typically provides much higher compression ratios than lossless compression. It has the disadvantage that it introduces error into the data so that the original data cannot be perfectly recovered. Because of their high compression ratios, only lossy compression techniques are considered here.

There are two general approaches to lossy data compression. The first approach is quantization of the raw data, which is shown in Figure 1.1(a). First, the raw



(a) Direct quantization of data.



(b) Transform quantization of data.

Figure 1.1: (a) Direct compression of raw data where X is the input data, Q is the quantizer (or encoder), i is the index stream, Q^{-1} is the decoder, and $\hat{X}$ is the lossy reproduction of the input data. (b) Transform compression where X is the input data, T is the transform, Q_j are the quantizers (or encoders), i_j are the index streams, Q_j^{-1} are the decoders, T^{-1} is the inverse transform, and $\hat{X}$ is the lossy reproduction of the input data.

data are encoded, or quantized, using a codebook; then the codebook indices are transmitted or stored; finally, the data are reconstructed by decoding the indices using a look-up table. The second approach is transform compression which is shown in Figure 1.1(b). First the data are transformed to some other space; then the coefficients of the transformed data are encoded, or quantized; next, the codebook indices are transmitted and stored; the coefficients are reconstructed by decoding the indices by table look-up; finally, the original data are reconstructed by transforming the reconstructed coefficients back to the original data space.

In this dissertation, I develop and implement lossy compression algorithms for transform compression based on vector quantization compression techniques and the

wavelet transform. These algorithms are optimized so that the compression error can be reduced to the lowest possible level for any rate. I then test these algorithms on earth science data and evaluate the effects of compression by comparing the analyses of raw and compressed data. Since EOS data are not presently available, other data must be used as models of EOS data. For the first set of experiments, I compress scanned photographic still images (from the USC database) as may be found in any image database. For the second set of experiments, I compress acoustic sonar data from the University of Washington's Benthic Acoustic Measurement System which consist of sea-bed scattering measurements taken in shallow water. These data are provided by Professor Darrell Jackson and others, and are used, among other applications, to detect mine-like objects and to conduct ocean temperature studies [35, 57, 58, 92].

It is difficult to say if any level of lossy compression can be determined to be acceptable since the compressed data may be used for various applications. With this in mind, my experiments involving lossy compression of earth science data have two goals. The first is to allow quick and easy previewing of the data (e. g. so that acoustic sonar data can transmitted over a low-bandwidth modem). The second is to study the preservation of scientific data analyses (i. e. ocean temperature measurements and mine detection from acoustic sonar data) as a function of compression.

The remainder of this dissertation is structured as follows. In Chapter 2, I review vector quantization compression techniques. In Chapter 3, I review the discrete orthonormal wavelet and wavelet packet transform. In Chapter 4, I review wavelet-based data compression and present three hybrid wavelet/VQ transform compression algorithms that use the generalized Breiman, Friedman, Olshen, and Stone (GBFOS) algorithm [17] and recursive optimal pruning algorithm (ROPA) [61] to do bit allocation for wavelets and wavelet packets. In Chapter 5, I discuss the results of these wavelet/VQ algorithms applied to the standard USC image data set. In Chapter 6, I examine the compression of acoustic sonar data using mine detection and temper-

ature studies. Conclusions and avenues of future work are presented in Chapter 7. In addition, an embedded form of multilevel halftoning, with color palette ordering algorithms, is presented in Appendix A as an alternative coding method for color images.

Chapter 2

VECTOR QUANTIZATION

Vector quantization (VQ) is a lossy compression technique that has been used extensively in speech and image compression. An extension of scalar quantization, VQ exploits the memory or correlation that exists between neighboring samples of a signal by quantizing them together rather than individually. Vector quantization is lossy because the encoder performs a many-to-one mapping. Discussions of vector quantization can be found in [1, 42, 45].

In this chapter, I review various compression measures in Section 2.1. I next discuss two basic forms of vector quantization, full search VQ and tree-structured VQ, in Sections 2.2 and 2.3, respectively. The Generalized Lloyd Algorithm which is commonly used to create these VQ's is discussed in Section 2.4. Finally, pruned tree-structured VQ and two pruning algorithms: the generalized Breiman, Friedman, Olshen, and Stone (GBFOS) algorithm [17] for classification and regression trees, and the recursive optimal pruning algorithm (ROPA) [61], are discussed in Section 2.5.

2.1 Compression Measures

Given N possible data symbols S_n, $n = 0, \ldots, N-1$, the rate R needed to uniquely represent the data with a fixed number of bits is defined as $R = \lceil \log_2 N \rceil$. The first order entropy, which is the lower bound on the number of bits required to uniquely represent the data when each symbol is coded independently, is defined as $-\sum_{n=0}^{N-1} P(S_n) \log_2 P(S_n)$. If blocks of data are coded, lower entropies can be achieved.

The compression ratio (CR) is defined as the number of bits required to represent the original data divided by the number of bits required to represent the compressed data. Assuming that all coding of the data will be followed by a lossless compression algorithm such as Huffman coding or LZW, the common Unix `compress` utility, first order entropy, rather than rate, is used to compute the compression ratios in this work.

2.2 *Full Search Vector Quantization*

All forms of vector quantization use codebooks for encoding and decoding. A codebook is a collection of codewords or possible reproduction vectors. In full search VQ, as shown in Figure 2.1, the encoder computes the distortion between an input vector X (group of data samples) and all codewords $Y_i, i = 0, \ldots, N-1$ in an unstructured codebook Y. The binary index of the codeword i that has the least distortion with respect to the input vector is transmitted (or stored). The decoder performs a simple table lookup with the transmitted (or stored) index i and outputs the reproduction vector Y_i. Note that $X \neq Y_i$ because this is a many-to-one mapping.

The rate R of a full search VQ codebook with vector dimension d is defined to be $R = \frac{\log_2 N}{d}$ bits per vector element. Using this definition, the size of the codebook can be rewritten as $N = 2^{Rd}$. The size of the codebook, and hence the size of the search, grows exponentially with rate and vector dimension. The storage requirements of the codebook are low as only the N codewords need to be stored. The full search, while computationally complex, guarantees that the best possible representation of the input vector will be selected. Several fast full-search methods which significantly reduce the search time by ordering the codebook (requiring a larger storage structure) to restrict searches to a small portion of the search space, have been developed for full search VQ, [48, 59, 77].

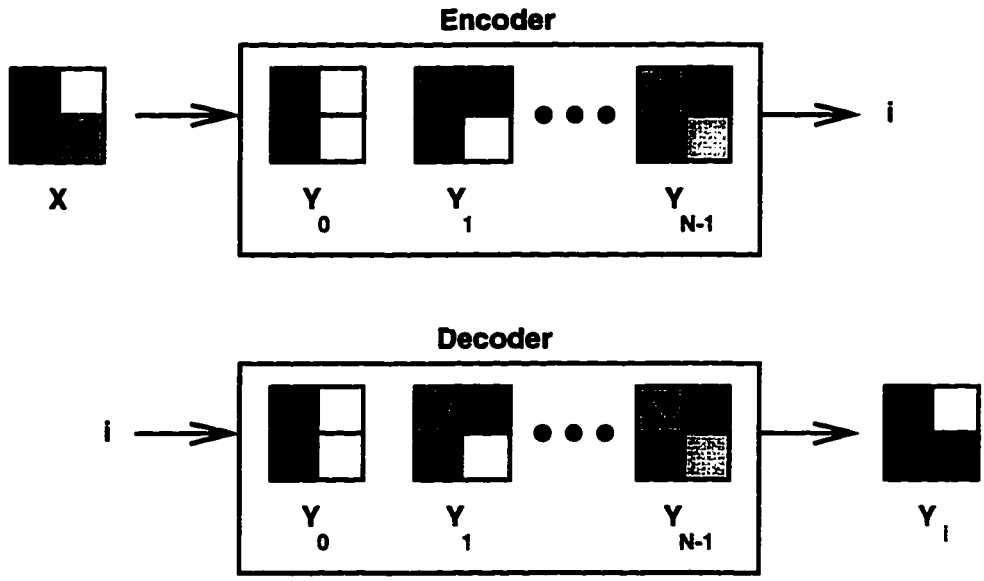


Figure 2.1: Full search vector quantization.

2.3 *Tree Structured Vector Quantization*

Tree-structured VQ (TSVQ) [21] is a low-complexity alternative to standard full search VQ. As shown in Figure 2.2, the codebook is structured as a binary (or M-ary) tree where the leaves of the tree are codewords Y_i, and the intermediate nodes of the tree are averaged versions of the codewords of their children. Beginning from the root node of the codebook, the encoder computes the distortion between an input vector X and a node's children and selects the child node that produced the lowest distortion with respect to the input vector. This process is repeated until a leaf node (codeword) is reached. The binary index i of the leaf node codeword (i. e. the path map from the root to the leaf node) is then transmitted (or stored). The decoder performs a simple table lookup with the transmitted (or stored) index i and outputs the reproduction vector Y_i. Note that $X \neq Y_i$, and that Y_i is not necessarily the best possible representation of the input vector that would be found among all of the codewords if a full search were done.

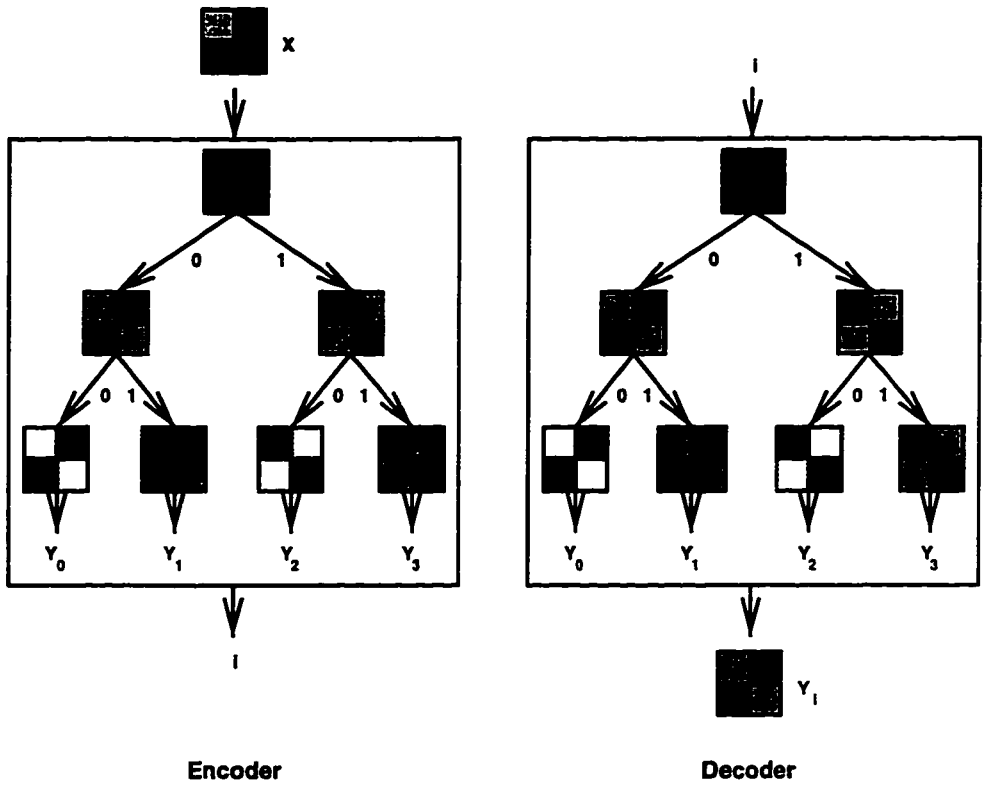


Figure 2.2: Tree structured vector quantization.

The rate R of a balanced binary tree structured VQ codebook with vector dimension d and N leaf codewords is defined to be $R = \frac{\log_2 N}{d}$ bits per vector element. Since the encoder performs a sequence of binary (or larger) searches instead of the one large search done in full search VQ, encoding complexity increases linearly with rate and vector dimension rather than increasing exponentially. TSVQ codebooks require twice the storage space of full search VQ with the same number of codewords since the $N - 1$ intermediary codewords must also be stored. In general the output of a TSVQ will suffer more degradation than the output of a full search VQ with the same number of codewords. This is due to the constraint on the search. However, an unbalanced TSVQ with the same *rate* as a full search VQ may have many more codewords, and may outperform a full search VQ in terms of distortion as well as

speed. Hence, the trade-off of greatly reduced search and design complexity for some possible increase in distortion usually makes TSVQ attractive.

2.4 *Vector Quantizer Design: The Generalized Lloyd Algorithm*

As shown previously in Figures 2.1 and 2.2, a quantizer Q maps a dimension d Euclidean source vector $X \in \mathbb{R}^d$ onto a finite set of reproduction vectors $Y = \{Y_i\}$, $i = 0, \ldots, N-1$, where Y is called a codebook and the $Y_i \in \mathbb{R}^d$ are codewords of the codebook. Associated with each codeword Y_i is a cell, or region, R_i which is a partition of $\mathbb{R}^d$. The cells are mutually exclusive, and the union of the cells must cover the entire space $\mathbb{R}^d$, that is

$$R_i \cap R_j = \emptyset, \quad i \neq j, \quad \text{and} \quad \bigcup R_i = \mathbb{R}^d.$$

Quantization works by assigning to any random vector X, the associated cell's codeword Y_i given that X is an element of cell R_i

$$Q(X) = \{Y_i : X \in R_i\}.$$

The mapping $Q : \mathbb{R}^d \rightarrow Y$ is many-to-one and is thus irreversible.

Quantizer performance is typically evaluated by a distance metric or distortion measure $d(x, y)$ where x and y are vectors in $\mathbb{R}^d$. The average distortion D due to quantization is evaluated as the expected distortion per source vector where the source X is treated as a random variable,

$$D = E[d(X, Q(X))] = E[d(X, Y)] = \sum_{i=0}^{N-1} P_i E[d(X, Y_i)|X \in R_i].$$

where P_i is the probability that the source vector X is in R_i.

Typical metrics used for $d(x, y)$ are the L_P norms

$$d(x, y) = [|x - y|^p]^{1/p},$$

and the squared error which is the square of the L_2 norm (i. e. the square of the Euclidean distance),

$$d(x, y) = |x - y|^2,$$

leading to the mean square error (MSE) distortion measure.

$$\text{MSE} = E[d(X, Q(X))] = E\left[|X - Q(X)|^2\right].$$

The MSE is often used due to its numerical tractability. Although it is not always perceptually meaningful, low MSE's (on compressed images) usually correspond to high quality, and high MSE's usually correspond to poor quality. Thus, in this work, I evaluate compression quality with the MSE.

Other commonly used measures are the signal-to-noise ratio (SNR) and the peak signal-to-noise ratio (PSNR)

$$\text{SNR} = 10 \log_{10} \frac{E[X^2]}{\text{MSE}}, \quad \text{PSNR} = 10 \log_{10} \frac{A^2}{\text{MSE}}$$

where A is the maximum value of X. PSNR is typically used to evaluate image quality. Since each pixel in an 8-bit image relates to color or grayscale intensity, the energy of different images can vary greatly even though each image is just as visually important. For example, a light gray image can be just as visually important as a dark gray image, but the power of these two images varies greatly due to the arbitrary numerical assignments given to pixels. For this work I use the PSNR to evaluate compression of photographic images, and the SNR to evaluate compression of scientific data.

The definitions of the previously mentioned distortion measures depend on well defined source characteristics. Unfortunately, good models for multidimensional data and image data are not common, so training data or empirical data are commonly used to approximate the probability distribution function of the source data. Training data are often a sequence of vectors or samples $X = \{X_i\}$ gathered from the data

source. The average distortion is still defined as

$$D = E[d(X,Y)] = \sum_{i=0}^{N-1} P_i E[d(X,Y_i)|X \in R_i].$$

but X now refers to the training data, R_i refers to partitions of the training data that are subsets of the previously defined cell subspaces, rather than to the entire subspace itself, and P_i is redefined as the probability that a training vector X is in partition R_i representing the probability that the codeword Y_i is used for encoding. In the remainder of this chapter I use X and R_i to refer to the training data and the partitions of the training data into each cell, respectively, although the descriptions are still valid if X is a random source variable and the R_i are subspaces of $\mathbb{R}^d$.

In general, one is concerned with optimal quantizers whose performance minimizes the average distortion $D = E[d(X,Y)]$. There are two conditions that must be met for a codebook to be (locally) optimal. These conditions are defined on the cells, and if they are met, the cells are called *Voronoi regions.* The first condition is the nearest neighbor (NN) condition which states that every value in a given Voronoi region R_i is "closer" or nearer to the codeword Y_i of the Voronoi region than to any other codeword with respect to a given distortion measure, that is

$$R_i = \{X : d(X,Y_i) \leq d(X,Y_j) \quad \text{for} \quad j = 0,\ldots,N-1\}.$$

In the case of a tie, the vector X can be arbitrarily assigned to the Voronoi region with the lesser index to maintain mutual exclusivity of the partitions. The second condition for optimality is the centroid condition which states that the codeword must minimize the average distortion of the entire Voronoi region with respect to a given distortion measure, that is

$$Y_i = \text{Centroid}(R_i) = \arg\min_{y \in R_i} E[d(X,y)|X \in R_i].$$

When the MSE distortion measure is used, the centroid corresponds to the "average" value or the geometric center of the Voronoi region. These two conditions imply that

the encoder is optimized for the given decoder by the nearest neighbor condition, and that the decoder is optimized for the given encoder by the centroid condition.

A generalization of a least squares quantization method for scalar data [70] to multidimensional data in 1980 was the first vector quantization design technique introduced in the literature, and is known as the generalized Lloyd algorithm (GLA) [69]. The GLA is a descent algorithm that finds codebooks with progressively lower average distortions by using iterations of the nearest neighbor and centroid conditions. The algorithm is initialized with an arbitrary set of initial codewords $Y[0] = \{Y_i\}, i = 0, \ldots, N-1$, training data X, and initial distortion $D[0] = E[d(X, Y[0])]$.

One iteration of the GLA, where n refers to the current iteration, works as follows. Given a codebook $Y[n] = \{Y_i\}$ the first step is to find the optimal partition of the training data using the nearest neighbor condition:

$$R_i = \{X : d(X, Y_i) \leq d(X, Y_j) \quad \text{for} \quad j = 0, \ldots, N-1\}.$$

Then, using the centroid condition, the optimal reproduction vectors Y_i for the given partitions R_i are found. These reproduction vectors constitute the $(n+1)^{\text{st}}$ codebook $Y[n+1]$

$$Y_i = \text{Centroid}(R_i) = \arg\min_{y \in R_i} E[d(X, y) | X \in R_i].$$

The distortion due to codebook $Y[n+1]$ is evaluated as

$$D[n+1] = E[d(X, Y[n+1])].$$

The GLA is a descent algorithm that will converge to a local minimum. However, while convergence is guaranteed, the number of iterations it takes to converge can be quite large. Typically the algorithm is halted when the relative change in distortion incurred between two iterations becomes arbitrarily small as defined by ϵ,

$$\frac{D[n] - D[n+1]}{D[n]} < \epsilon,$$

and the final codebook is determined to be $Y = Y[n+1]$.

To design a full search VQ with an arbitrary number of codewords, one may start with any initial codebook $Y[0]$ with the desired rate and apply the GLA to it until it converges to a locally optimal codebook Y. Obviously, different initial codebooks will converge to different locally optimal codebooks. The choice of initial codebooks is important and is discussed in [42].

One common method of creating a full-search codebook is the Linde-Buzo-Gray (LBG) splitting algorithm [69]. In this method an initial codebook of rate 0 containing 1 codeword, $Y^0 = \{Y_0\}$, is used where the optimal codeword is the centroid of the training data. Note that the superscript on the codebook Y^i now refers to the rate i of the codebook. An initial codebook of size 2 (rate 1) is created by "splitting" the codeword which may be done by adding a vector ε of arbitrarily small, random values, e. g. $Y^1[0] = \{Y_0, Y_0 + \varepsilon\} = \{Y_0, Y_1\}$. The GLA is applied to $Y^1[0]$ to create codebook Y^1. For each LBG iteration, the codebook size is increased by splitting the codewords of the previous codebook Y^n to create a larger initial codebook $Y^{n+1}[0]$, and the GLA is applied to the larger codebook to achieve the optimized codebook Y^{n+1}. The iterations continue until the desired rate is achieved. An artificial set of two-dimensional grayscale image training data with 262144 samples was created using the Gaussian distribution. This training sequence is used in this chapter to demonstrate the differences between different forms of VQ where each VQ is designed to have an average rate of 8.0 bits per vector. The Voronoi regions for a two-dimension VQ created by the LBG algorithm with 256 codewords are shown in Figure 2.5(a). Note that the Voronoi regions are convex and tend to have a honeycomb-like pattern. The average distortion of this full search VQ is 11.75.

The GLA is a flexible algorithm that can be used to create many different VQ schemes. The creation of an M-ary TSVQ T is illustrated in Figure 2.3. Each node $t_i \in T$ has an associated codeword Y_i and training data partition R_i. The nodes of the M-ary tree T are labeled such that each interior node t_i has children nodes $t_{Mi+1}, \ldots, t_{Mi+M}$, and the root node is t_0. The tree is initialized by creating a root

node t_0 which is assigned the entire training sequence R_0, Figure 2.3(a). The root node's codeword Y_0 is the centroid of the training sequence, and it represents the rate 0 TSVQ. The tree is grown, or opened, by creating M child nodes $t_1, \ldots, t_M$ descending from the root node. Using the GLA, M codewords, $Y_1, \ldots, Y_M$, from the root node's training data R_0 are found and assigned to each of the root's M children as the child node's codeword, Figure 2.3(b). The training data R_0 stored at the root node is then partitioned among the M children using the nearest neighbor rule and the node's assigned codeword so that each child node contains a non-overlapping subset, $R_i, i = 1 \ldots, M$, of the training data's space, Figure 2.3(c). Thus, assuming that both the nearest neighbor and centroid condition are satisfied, the codeword Y_i at each node t_i is the centroid of the portion of training data R_i stored at that node. In addition, each codeword has as associated encoding rate $R(Y_i) = P_i \text{depth}(t_i)$ and distortion $D(Y_i) = P_i E[d(X, Y_i)|X \in R_i]$, where $R(Y_i)$ is a tree functional value that increases monotonically with node depth, $D(Y_i)$ is a tree functional value that decreases monotonically with node depth, and the $P_i = \sum_{j=1}^{M} P_{Mi+j}$ are the probabilities that any node is visited during the tree search. This growth procedure is recursively applied to each leaf node of the TSVQ until the desired rate is achieved.

A fixed-rate, or balanced, TSVQ is created by growing the leaf nodes with the shortest path length from the root node. The Voronoi regions for a two-dimension balanced TSVQ with 256 codewords for the Gaussian distributed data is shown in Figure 2.5(b). The tree-structure constrains the search to a series of hyper-plane tests, so the balanced TSVQ, when compared to a full search VQ with the same rate, suffers more degradation. The average distortion of this balanced TSVQ is 12.97.

Fixed-rate trees are an inflexible approach to VQ since they do not permit easy adaptation to source statistics. Instead, an unbalanced tree produced by a greedy growing approach is preferred [81]. In this method, illustrated in Figure 2.4, each leaf node is opened, but its descendents are not immediately considered to be a part of the tree. Instead, the changes in rate and distortion that would arise if the descendents

of the current leaf node were used, rather than the current leaf node, are calculated, and the current leaf node which will maximize the decrease in distortion for its given increase in rate is grown.

Given that each leaf node t_i has an associated codeword Y_i, each codeword has encoding rate $R(Y_i) = P_i \text{depth}(t_i)$ and distortion $D(Y_i) = P_i E[d(X, Y_i)|X \in R_i]$. The average rate and distortion of the TSVQ T are calculated as

$$R = \sum_{t_i \in \tilde{T}} R(Y_i) \quad \text{and} \quad D = \sum_{t_i \in \tilde{T}} D(Y_i),$$

where $\tilde{T}$ refers to the leaf nodes of tree T. The change in distortion if the codeword Y_i of leaf node t_i is not used is calculated as $\Delta D(Y_i) = D(Y_i) - \sum_{j=1}^{M} D(Y_{Mi+j})$, and the change in rate is $\Delta R(Y_i) = R(Y_i) - \sum_{j=1}^{M} R(Y_{Mi+j}) = -P_i$. The leaf node that has the maximum value $-\frac{\Delta D(Y_i)}{\Delta R(Y_i)} = \frac{\Delta D(Y_i)}{P_i}$, maximizing the decrease in distortion for the given increase in rate, for all leaf nodes is grown. The greedy growing of leaf nodes continues recursively until the desired rate is achieved.

2.5 *Pruned Tree Structured Vector Quantization*

The full search and tree-structured vector quantizers described in Sections 2.2 and 2.3 implement a fixed rate code. That is, an equal number of bits is used to reproduce each input vector. A variable rate code usually results in much better performance by devoting more bits to regions of the data that are active or difficult to code, such as the edges, and fewer bits to less active regions, such as a solid background.

A variable rate code can be implemented in a TSVQ simply by using an unbalanced tree, for example the greedy growing technique [81] described above. An even better approach to designing an unbalanced trees is described in [23, 61]. In this approach a (balanced or unbalanced) TSVQ with a rate higher than the desired rate is designed. This high rate tree is then pruned (usually based on rate-distortion trade-offs) to yield an unbalanced TSVQ with the desired rate. The resulting coder

is called a pruned TSVQ (PTSVQ).

PTSVQ typically outperforms both fixed rate TSVQ and full search VQ over most rates of interest [23, 82]. The Voronoi regions for a two-dimension PTSVQ with an average rate of 8 bits per vector pruned from an unbalanced TSVQ using GBFOS, described next, with a rate of 14.4 bits per vector, designed for the Gaussian distributed data, is shown in Figure 2.5(c). The PTSVQ Voronoi regions are adapted to the more active regions of the data while the Voronoi regions of the balanced TSVQ are not so sensitive to the distribution of the data. PTSVQ may also have many more codewords than a fixed-rate codebook. This PTSVQ has 622 codewords while the fixed rate codebooks have only 256. The average distortion of this PTSVQ is 11.28 which is less than the average distortion of the full search VQ even though the full search VQ satisfies both the nearest neighbor and centroid conditions and is thus considered to be locally optimal.

2.5.1 The GBFOS Pruning Algorithm

Perhaps the most commonly used tree pruning algorithm is the generalized Breiman, Friedman, Olshen, and Stone (GBFOS) algorithm [17] for classification and regression trees which was first used for TSVQ's in [23]. The GBFOS algorithm finds in a given codebook T all the nested subtrees S sharing the same root node as T that lie on the lower convex hull of the rate-distortion curve. These subtrees give the optimal rate-distortion performance (i. e. the lowest distortion for a given rate) for any subtree of the given codebook. The PTSVQ codebooks are not universally optimal since there may be other codebooks that give a better rate-distortion performance. Hence the choice of the initial codebook T constrains the performance of the PTSVQ. Should optimal codebooks be scarce, *time-sharing* between a high rate and a low rate codebook can be done to yield the desired rate. By using a high-rate codebook and a low-rate codebook to encode data over different portions of time, time-sharing yields a codebook on the convex hull of the rate-distortion curve. An example of the

codebooks that can be found using GBFOS is shown in Figure 2.6 where $*$'s represent all possible pruned subtrees; $S_0, \ldots, S_3$ represent the GBFOS optimal codebooks; and $\widehat{S}$ is a codebook created by time-sharing S_2 and S_3.

One iteration of TSVQ design by pruning is illustrated in Figure 2.7. It is done by calculating, for all internal nodes, the changes in rate and distortion that would arise if the descendents of the internal nodes were removed, and the internal node used instead. The internal node that will minimize the increase in distortion for its given decrease in rate, if used instead of its descendents, is pruned. Pruning continues by removing one branch of the tree at a time until the desired rate is reached.

The original TSVQ T is an M-ary tree with nodes t_i labeled such that each interior node t_i has children nodes $t_{Mi+1}, \ldots, t_{Mi+M}$, and the root node is t_0. Each node has an associated codeword Y_i and training data partition R_i, where the encoding rate and distortion of the codewords Y_i are calculated as $R(Y_i) = P_i \text{depth}(t_i)$ and $D(Y_i) = P_i E[d(X, Y_i)|X \in R_i]$. Hence, the average rate and distortion of the TSVQ T can be written as

$$R = \sum_{t_i \in \tilde{T}} R(Y_i) \quad \text{and} \quad D = \sum_{t_i \in \tilde{T}} D(Y_i),$$

where $\tilde{T}$ refers to the leaf nodes of tree T.

When a branch headed by node t_i is pruned, all nodes of the branch are discarded, and the node t_i is turned into a leaf node. Its codeword Y_i is used to encode data rather than the leaf node codewords of the branch just pruned. To find the optimal subtrees, the branch that yields the minimum decrease in distortion for the increase in rate is selected. Each node $t_i \in T$ can be considered to be the root of a branch whose leaf nodes contain a partition, or subset, of the codewords of the entire TSVQ T. This subset is called quantizer Q_i, as it represents the TSVQ that would be created if only training data R_i were used to create a TSVQ. The average rate and distortion

of the subset TSVQ's Q_i are calculated as

$$R(Q_i) = \sum_{t_i \in \tilde{Q}_i} R(Y_i) \quad \text{and} \quad D(Q_i) = \sum_{t_i \in \tilde{Q}_i} D(Y_i),$$

where $\tilde{Q}_i$ refers to the leaf nodes of branch Q_i. If for all leaf nodes t_i, Q_i contains only the leaf node codeword $\{Y_i\}$, this relationship can be recursively written for internal nodes t_i as

$$R(Q_i) = \sum_{j=1}^{M} R(Q_{Mi+j}) \quad \text{and} \quad D(Q_i) = \sum_{j=1}^{M} D(Q_{Mi+j}).$$

Using this idea of subset quantizers Q_i, the change in distortion that would result if the branch Q_i headed by node t_i is pruned is easily calculated as $\Delta D(Y_i) = D(Y_i) - D(Q_i)$, and the change in rate is $\Delta R(Y_i) = R(Y_i) - R(Q_i)$. The internal node that has the minimum value $\lambda = -\frac{\Delta D(Y_i)}{\Delta R(Y_i)}$, for all internal nodes, is pruned. The resulting tree is the optimal subtree S located on the lower convex hull of the rate distortion curve. The λ value represents the negative value of the slope on the rate-distortion curve from the current tree to the subtree that results if node t_i is pruned. Since $\Delta D(Y_i)$ is always positive and $\Delta R(Y_i)$ is always negative, $\lambda = -\frac{\Delta D(Y_i)}{\Delta R(Y_i)}$ is always positive. This is a result of the monotonicity of the tree functional values $D(Y_i)$ and $R(Y_i)$ as a function of node depth. Pruning of internal nodes continues recursively until the desired rate is achieved by considering the newly found subtree S to be the current tree T, $T \leftarrow S$.

A pictorial example of GBFOS pruning is shown in Figure 2.8. The highest rate, or initial, tree T is represented as the $*$ in the lower right hand corner, and the rate 0 subtree comprised of $\{Y_0\}$ is the $*$ in the upper left hand corner of the rate-distortion plane. For each iteration of GBFOS, the space to be examined is limited to the *rectangular* region between the rate 0 codebook and the current subtree, Figure 2.8(a), since rate is monotonically increasing and distortion is monotonically decreasing with tree depth. One iteration of GBFOS involves calculating the slope from the current

subtree to all subtrees S that could be created by pruning only one internal node, and then selecting the subtree with the minimum slope as it lies on the lower convex hull, Figure 2.8(b). For the next iteration of GBFOS, a reduced area of the space, bounded by the rate 0 codebook and the current pruned subtree, is examined, Figure 2.8(c). The end results are the subtrees that lie on the lower convex hull of the rate-distortion curve, Figure 2.8(d).

Again, pruning is done by removing the branch which contributes the least increase in distortion for its given decrease in rate until the desired rate is achieved. Each time a TSVQ T is pruned to subtree S, and pruning needs to be done again, T is assigned the subtree S, $T \leftarrow S$, and the slope $\lambda = \min_{t_i \in T} \left(-\frac{\Delta D(Y_i)}{\Delta R(Y_i)} \right)$ must be reevaluated for every node. It is computationally costly and inefficient to evaluate the slope from the current tree T to every possible subtree each time the TSVQ is pruned. Instead, a fast and efficient computational solution that exploits the structure and properties of the TSVQ is used. Ideally, the rate-distortion characteristics of the optimal quantizer could be obtained from one merged quantizer which characterizes the *entire* tree. Towards this end, the following TSVQ data structure with the simplification of considering branch quantizers Q_i is used. This will make the characteristics of the optimal PTSVQ subtrees of the entire TSVQ accessible from the root node of the tree, while maintaining each pruned subtree as the entire tree structure itself.

The complete data structure for GBFOS is as follows. Given that each node t_i of the TSVQ has an associated codeword Y_i and training data partition R_i, the encoding rate $R(Y_i) = P_i \text{depth}(t_i)$ and distortion $D(Y_i) = P_i E[d(X, Y_i)|X \in R_i]$ of each codeword Y_i is stored at node t_i. Furthermore, as each node should represent a quantizer Q_i composed by the branch of the TSVQ that it heads, the following information is also stored at each node. The rate and distortion due to encoding training data R_i with the branch quantizer Q_i headed by node t_i are stored as $R(Q_i)$ and $D(Q_i)$. For leaf nodes $R(Q_i) = R(Y_i)$ and $D(Q_i) = D(Y_i)$, and for internal nodes

$R(Q_i) = \sum_{j=1}^{M} R(Q_{Mi+j})$ and $D(Q_i) = \sum_{j=1}^{M} D(Q_{Mi+j})$. This permits each internal node to have direct access to the rate-distortion characteristics of its entire branch from its children without requiring the specific details of the structure of that branch. Note that $R(Q_0)$ and $D(Q_0)$ are the rate and distortion of the current optimal GBFOS subtree stored at the root node. Finally, the slope $\lambda(Y_i)$ that would arise if the TSVQ were pruned at node t_i is saved as $\lambda(Y_i) = -\frac{D(Y_i)-D(Q_i)}{R(Y_i)-R(Q_i)}$, and the minimum slope of the branch quantizer Q_i is stored as $\lambda(Q_i) = \min(\lambda(Y_i), \lambda(Q_{Mi+1}), \ldots, \lambda(Q_{Mi+M}))$. For leaf nodes, both slope values are infinity. Note that $\lambda(Q_0)$ represents the slope from the current GBFOS optimal subtree to the next GBFOS optimal subtree, and that it also provides a search path from the root node to the minimum slope node.

GBFOS is initialized by finding the slopes that would result if each node were pruned. Then, until the desired rate is achieved, the lowest slope node is pruned, and the rate, distortion, and slope values of the subset quantizers of the ancestor nodes are recomputed. Only the ancestor nodes of the lowest slope node need to be updated since pruning affects only those nodes whose subset quantizers contain the leaf node codewords just removed from the tree. In this manner, each slope is examined only once, and for each iteration, only the portion of the TSVQ containing the pruned node is examined. Furthermore, the overall reported rate and distortion of the PTSVQ's are monotonically decreasing and increasing, respectively, with increasing λ, and all of the quantizers on the vertices of the lower convex hull of the rate-distortion curve are identified starting from the highest rate to the lowest rate quantizer. By pruning the entire tree to its root node, the optimal rate-distortion characteristics for all bit rates are found. This permits the use of a precise description of the quantizer by its rate-distortion characteristics for use in the design of more complex encoding schemes.

Initialization is outlined in Table 2.1. Its purpose is to compute the slopes that would result from pruning the tree at any one node, including the root node. By computing $\lambda(Q_i)$ as $\min(\lambda(Y_i), \lambda(Q_{Mi+1}), \ldots, \lambda(Q_{Mi+M}))$ for all nodes, a path through the TSVQ from the root node to the minimum slope node is created. When initial-

Table 2.1: GBFOS pruning initialization.

Step 1. For each leaf node t_i:
$\lambda(Y_i) \leftarrow \infty$,
$\lambda(Q_i) \leftarrow \infty$,
$R(Q_i) \leftarrow R(Y_i)$,
$D(Q_i) \leftarrow D(Y_i)$.

Step 2. For each interior node t_i,
(a) Find the rate and distortion of the branch:
$R(Q_i) \leftarrow \sum_{j=1}^{M} R(Q_{Mi+j})$,
$D(Q_i) \leftarrow \sum_{j=1}^{M} D(Q_{Mi+j})$,
(b) Find the minimum slope:
$\lambda(Y_i) \leftarrow -\frac{D(Y_i)-D(Q_i)}{R(Y_i)-R(Q_i)}$,
$\lambda(Q_i) \leftarrow \min(\lambda(Y_i), \lambda(Q_{Mi+1}), \ldots, \lambda(Q_{Mi+M}))$.

izing leaf nodes t_i where Q_i is the leaf codeword $\{Y_i\}$, it is assumed that the slopes, $\lambda(Q_i)$ and $\lambda(Y_i)$, are infinite to avoid an attempt to prune a leaf node. The rate and distortion of the branch descending from node t_i are stored as $R(Q_i)$ and $D(Q_i)$.

The main GBFOS algorithm is described in Table 2.2. Pruning continues until the entire tree has been pruned, $\lambda(Q_0) = \infty$, or the desired rate, R_{target}, has been reached. First, the node t_i that yields the minimum slope quantizer when pruned is found by following the path indicated by the minimum value stored at the root node, $\lambda(Q_0)$. Once this node t_i is found, it is pruned, and its minimum slope value is set to infinity so that the node will not be revisited. The ancestor nodes are updated in a manner similar to the initialization procedure, except that the initialization procedure is performed on *all* nodes whereas updating needs to be performed only on the pruned node and its ancestors. The average rate and distortion of each current subtree are stored as $R(Q_0)$ and $D(Q_0)$. The subtree structure is found by examining the slope values $\lambda(Y_i)$ where $\lambda(Y_i) = \infty$ indicates that the node is a leaf node.

The encoding and decoding complexities for PTSVQ are the same as for TSVQ. However, the design complexity of PTSVQ is greater since a large TSVQ must first be created and then pruned. Let N be the total number of nodes in the original

Table 2.2: GBFOS pruning.

Step 1. If $\lambda(Q_0) = \infty$, quit. The TSVQ has been pruned to its root node.

Step 2. Find the least slope node.
Assign $t_i \leftarrow t_0$ and $\lambda_{min} \leftarrow \lambda(Q_0)$.
While $\lambda(Y_i) \neq \lambda_{min}$:
Find $t_{Mi+j}, j = 1 \ldots, M$, such that $\lambda(Q_{Mi+j}) = \lambda_{min}$.
Assign $t_i \leftarrow t_{Mi+j}$.

Step 3. Prune node t_i and update its ancestors:
(a) Prune the quantizer at node t_i.
$\lambda(Y_i) \leftarrow \infty$,
$\lambda(Q_i) \leftarrow \infty$,
$R(Q_i) \leftarrow R(Y_i)$,
$D(Q_i) \leftarrow D(Y_i)$.
(b) Update the ancestors of node t_i.
While $t_i \neq t_0$, assign $t_i \leftarrow \text{parent}(t_i)$:
(i) Find the rate and distortion of the branch:
$R(Q_i) \leftarrow \sum_{j=1}^{M} R(Q_{Mi+j})$,
$D(Q_i) \leftarrow \sum_{j=1}^{M} D(Q_{Mi+j})$.
(ii) Find the minimum slope:
$\lambda(Y_i) \leftarrow -\frac{D(Y_i)-D(Q_i)}{R(Y_i)-R(Q_i)}$,
$\lambda(Q_i) \leftarrow \min(\lambda(Y_i), \lambda(Q_{Mi+1}), \ldots, \lambda(Q_{Mi+M}))$.

Step 4. The rate and distortion of the current subtree are $R(Q_0)$ and $D(Q_0)$.
If $R(Q_0) > R_{\text{target}}$ go to **Step 1.**
Otherwise, quit.

TSVQ T. The nodes of the tree must first be initialized with complexity N. Then the algorithm will find in the worst case N optimal codebooks. For each optimal codebook found, the tree will be searched for the node with the smallest λ and the node's ancestors have to be updated with complexity $\log N$. So the complexity of the GBFOS algorithm is $N \log N$. However, the time it takes to prune a TSVQ is insignificant compared to the time it takes to design a TSVQ.

2.5.2 The Recursive Optimal Pruning Algorithm

Although GBFOS pruning produces PTSVQ's that are optimal in the sense that the codebooks lie on the lower convex hull of the rate-distortion curve, GBFOS pruning cannot always achieve the desired rate since optimal points may be scarce. Time-sharing between a high rate and a low rate codebook can be done to yield the desired rate. Although this yields a codebook that is on the lower convex hull of the rate-distortion curve, it has the drawback of introducing non-stationary distortion throughout the data. The assignment of the different codebooks to different parts of the data may not be an easy task in this case, and may result in poor performance in sensitive regions of the data. It also interferes with the distribution of codeword indices which can affect lossless coding of VQ indices. The recursive optimal pruning algorithm (ROPA) [61] was introduced to overcome these drawbacks. For ROPA pruning, the tree is pruned by removing only unit branches, where a unit branch is a node with only M children. All of the optimal GBFOS subtrees are identified, and all of the suboptimal subtrees which are found minimize the area between the rate-distortion curve and the convex hull of the rate-distortion curve. This algorithm guarantees many more codebooks over the range of bit rates.

The ROPA algorithm works by first finding the minimum-slope GBFOS node in the tree T. The branch to be pruned at this node may be very large, and there may be a number of suboptimal codebooks within it. Instead of pruning the tree T at this GBFOS point, ROPA takes the branch that is to be pruned and processes it

while ignoring the rest of the tree. ROPA searches the branch to find the minimum slope node in the branch: this minimum slope node is not a GBFOS optimal node, that is, GBFOS optimal nodes are identified by ROPA only when it examines the *entire* tree. ROPA will next process the branch headed by the suboptimal minimum slope node. The process continues branch by branch until a minimum-slope node is reached that is a unit branch. The unit branch is pruned, and the distortion, rate, and minimum slope of the subset quantizers Q_i of the ancestors of the pruned node are recalculated. When a branch is completely pruned, the algorithm continues on the larger branch (or tree) from which the completely-pruned branch was first identified. This algorithm preserves all of the GBFOS optimal points by always identifying the minimum GBFOS point first and then pruning the branch M leaf nodes at a time until the GBFOS point is reached. In this fashion the entire tree is recursively pruned M leaf nodes at a time yielding a very dense set of pruned subtrees lying close to the convex hull, providing an alternative to time-sharing between two GBFOS trees.

A pictorial example of ROPA pruning is shown in Figure 2.9. After each pruning iteration of the GBFOS algorithm, Figure 2.8(c), a number of pruned codebooks are discarded as possible solutions since they are suboptimal. Some of these suboptimal codebooks lie in the triangular shaded regions bounded by the optimal PTSVQ's, Figure 2.9(a), while others lie outside of this region. ROPA bit allocation works by recursively pruning the tree in these shaded regions, identifying the suboptimal PTSVQ's which lie closest to the optimal rate-distortion curve, Figure 2.9(b).

ROPA uses the same data structures and initialization procedure of GBFOS pruning, Table 2.1. The main ROPA algorithm is described in Table 2.3. Pruning continues until the entire tree has been pruned, $\lambda(Q_0) = \infty$, or the desired rate, R_{target}, has been reached. First, the node t_i with the minimum slope quantizer is found by following the path indicated by the minimum value $\lambda(Q_0)$. Once this node t_i is found, ROPA is used to prune the branch it heads M leaf nodes at a time, starting from the highest rate subtree to the subtree formed by pruning node t_i. The subroutine

min_ropa_node(t_i), described in Table 2.4, is used to find the highest rate ROPA node t_j by recursively searching for the least slope node in each branch. For each node t_j that is pruned, its minimum slope value is set to infinity so that the node will not be revisited. The ancestor nodes are updated in a manner similar to the initialization procedure, except that the initialization procedure is performed on *all* interior nodes whereas updating needs to be performed only on the pruned node and its ancestors. The average rate and distortion of the current subtree are stored as $R(Q_0)$ and $D(Q_0)$.

A subtlety of the algorithm arises when there are two or more nodes with the same minimum slope. When this happens in the GBFOS algorithm, the solution is to prune all of the nodes at once since the rates of the "intermediate" codebooks can be achieved by time-sharing. Since ROPA was created to avoid time-sharing, each pruned branch must be examined. However, the order of examination does not matter since pruning one branch of the codebook does not affect the other branches, and the area under the ROPA rate-distortion curve will still be minimal with respect to the GBFOS. Therefore, the solutions found by ROPA are not unique. For every n identical minimum slopes, there will be $n!$ possible solutions. The complexity of ROPA is the same as GBFOS, $N \log N$, for N nodes in the original TSVQ T.

2.6 Implementation

Due to the inherent advantages of PTSVQ, its adaptability to source characteristics, its ability to outperform other basic forms of VQ, as well as the natural ease of allocating more bits to active regions of the source data, and fewer bits to less active regions of the data, PTSVQ is used throughout the remainder of this work. I have implemented in C the generalized Lloyd algorithm (GLA) [69] for designing full search VQ and balanced and unbalanced TSVQ codebooks. I have also implemented GBFOS pruning and ROPA pruning for the TSVQ codebooks. All of this code is available

Table 2.3: ROPA pruning.

Step 1. If $\lambda(Q_0) = \infty$, quit. The TSVQ has been pruned to its root node.

Step 2. Find the least slope GBFOS node. Assign $t_i \leftarrow t_0$ and $\lambda_{min} \leftarrow \lambda(Q_0)$.
While $\lambda(Y_i) \neq \lambda_{min}$:
Find $t_{Mi+j}, j = 1, \ldots, M$, such that $\lambda(Q_{Mi+j}) = \lambda_{min}$.
Assign $t_i \leftarrow t_{Mi+j}$.

Step 3. Perform ROPA pruning on the branch headed by the GBFOS node t_i:
If $\lambda(Q_i) < \infty$:
(a) Find the node with the highest rate ROPA subtree:
$t_j \leftarrow$ **min_ropa_node**(t_i) (see Table 2.4).
(b) Prune node t_j:
$\lambda(Y_j) = \infty$,
$\lambda(Q_j) = \infty$,
$R(Q_j) \leftarrow R(Y_j)$,
$D(Q_j) \leftarrow D(Y_j)$.
(c) Update the ancestors of node t_j.
While $t_j \neq t_0$, assign $t_j \leftarrow \text{parent}(t_j)$:
(i) Find the rate and distortion of the branch:
$R(Q_j) \leftarrow \sum_{m=1}^{M} R(Q_{Mj+m})$,
$D(Q_j) \leftarrow \sum_{m=1}^{M} D(Q_{Mj+m})$.
(ii) Find the minimum slope:
$\lambda(Y_j) \leftarrow -\frac{D(Y_j)-D(Q_j)}{R(Y_j)-R(Q_j)}$,
$\lambda(Q_j) \leftarrow \min(\lambda(Y_j), \lambda(Q_{Mj+1}), \ldots, \lambda(Q_{Mj+M}))$.
(d) The rate and distortion of the current subtree are $R(Q_0)$ and $D(Q_0)$.
If $R(Q_0) > R_{\text{target}}$ go to **Step 3.**

Step 4. The rate and distortion of the current subtree are $R(Q_0)$ and $D(Q_0)$.
If $R(Q_0) > R_{\text{target}}$ go to **Step 1.**
Otherwise, quit.

Table 2.4: ROPA pruning subroutine **min_ropa_node**(t_i).

Step 1. Find the minimum slope of the children nodes.
$\lambda_{min} \leftarrow \min(\lambda(Q_{Mi+1}), \ldots, \lambda(Q_{Mi+M}))$.
If $\lambda_{min} = \infty$, all of node t_i's descendents are leaf nodes, return node t_i.

Step 2. Find the minimum slope node.
While $\lambda(Y_i) \neq \lambda_{min}$:
Find $t_{Mi+m}, m = 1, \ldots, M$ such that $\lambda(Q_{Mi+m}) = \lambda_{min}$.
Assign $t_i \leftarrow t_{Mi+m}$.
Return **min_ropa_node**(t_i).

through the Data Compression Lab WWW homepage

`http://isdl.ee.washington.edu/COMPRESSION/homepage.html`,

or by anonymous ftp to `isdl.ee.washington.edu` in `/pub/VQ/code`. The code has been well tested and is being used by many others and in Dr. Riskin's class EE587: Vector Quantization and Data Compression.

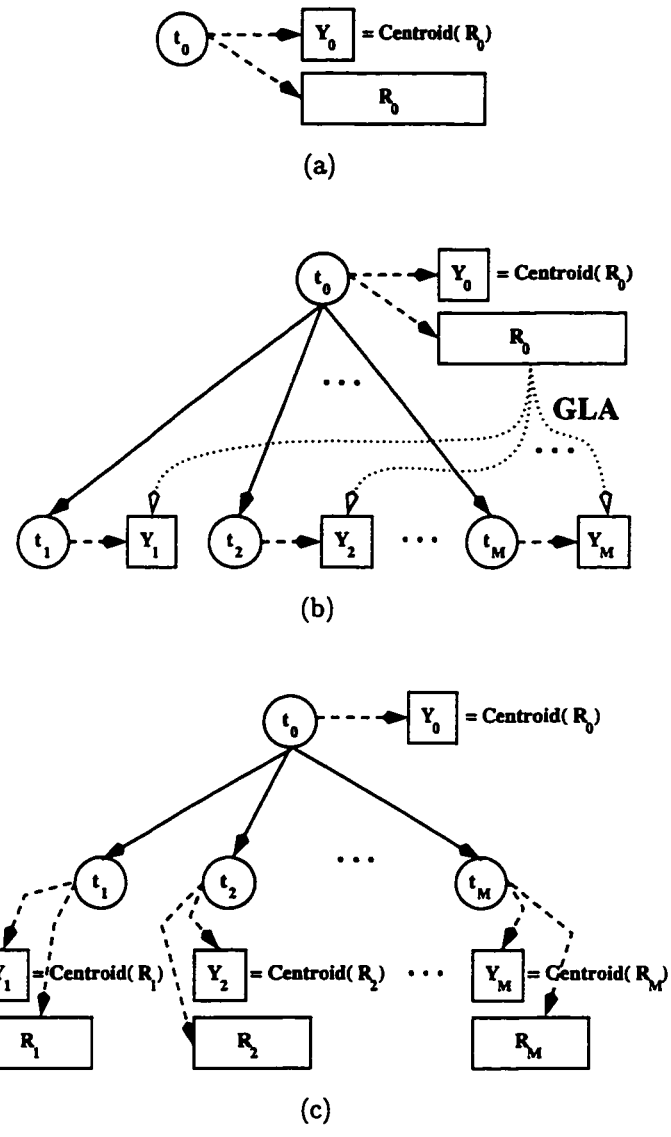


Figure 2.3: TSVQ construction. (a) A root node t_0 is created and assigned the complete training set with its centroid as a codeword. (b) M child nodes $t_1, \ldots, t_M$ are assigned to the root node, and the GLA is used to created M codewords from the training data which are assigned to the child nodes. (c) The root node's training data is partitioned among the children nodes using the nearest neighbor condition. The end result is that each node's codeword is the centroid of its portion of training data. This process is repeated recursively on leaf nodes until a TSVQ with the desired rate is created.

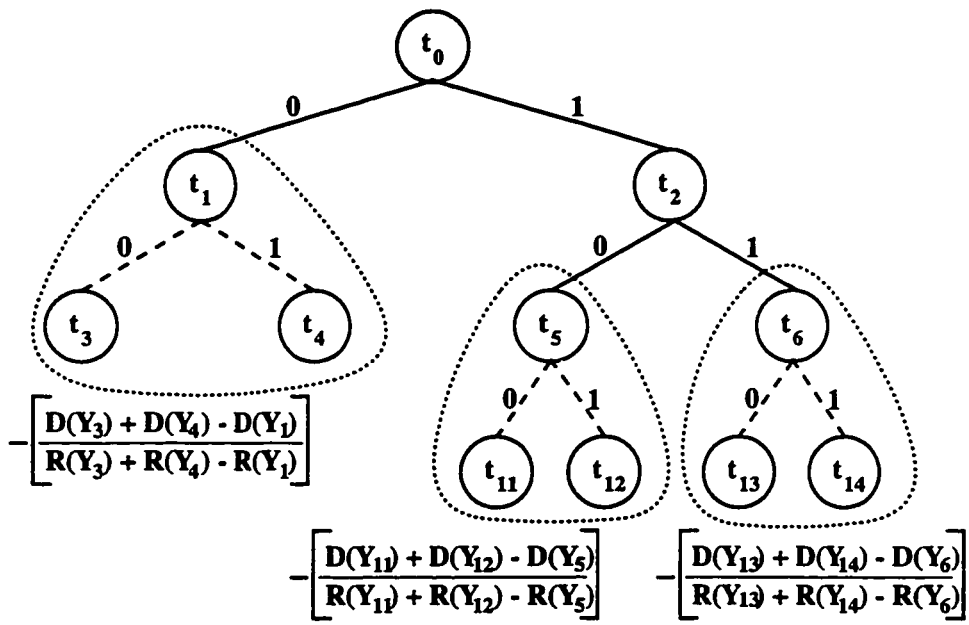
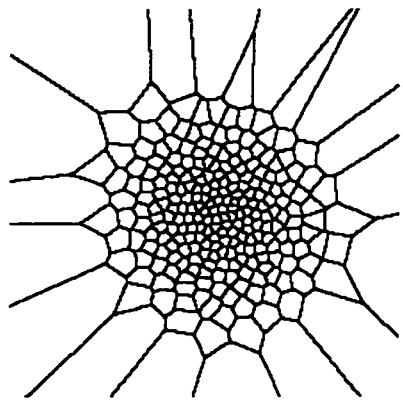
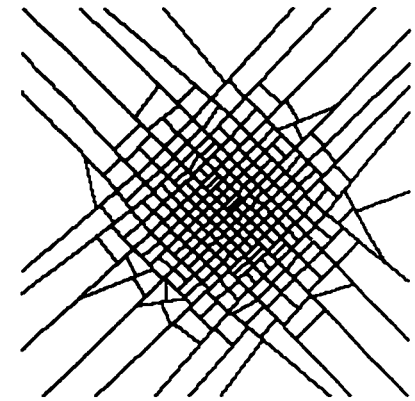


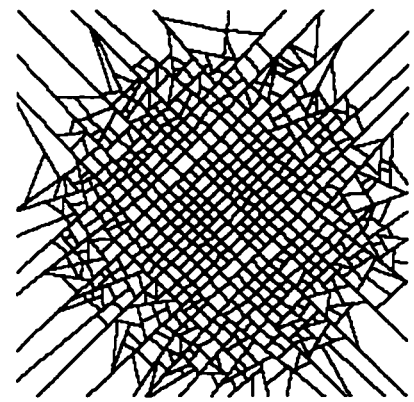
Figure 2.4: TSVQ design by greedy growing. The change in distortion versus the change in rate is calculated for each leaf node (t_1, t_5, and t_6) as if that leaf were actually grown. The leaf node that has the maximum decrease in distortion for its given increase in rate if opened is grown. The greedy growing of leaf nodes continues until the desired rate is achieved.



(a) Full search VQ. $D = 11.75$.



(b) Fixed-rate tree-structured VQ. $D = 12.97$.



(c) Pruned TSVQ. $D = 11.28$.

Figure 2.5: Voronoi regions for (a) full search VQ, (b) fixed-rate tree-structured VQ, and (c) pruned TSVQ trained on Gaussian distributed data. Each codebook has an average rate of 8.0 bits per vector.

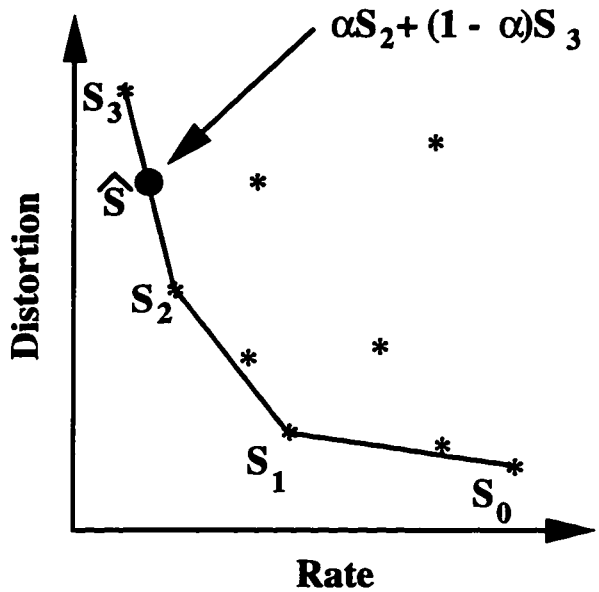


Figure 2.6: GBFOS codebooks. Each $*$ is a possible pruned subtree. S_0, S_1, S_2, and S_3 are GBFOS optimal PTSVQ's lying on the lower convex hull of the rate-distortion curve. $\widehat{S}$ is a codebook created by time-sharing two GBFOS codebooks.

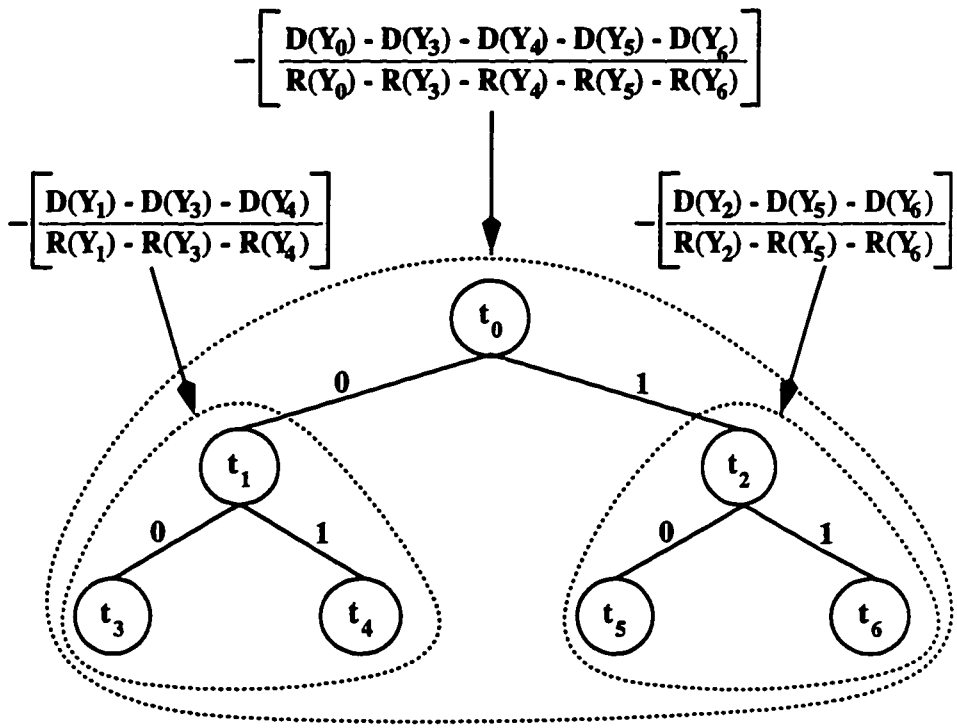


Figure 2.7: TSVQ design by GBFOS pruning where a high-rate TSVQ is pruned one branch at a time until the desired rate is achieved. In each pruning iteration, the change in distortion versus the change in rate is calculated for each internal node (t_0, t_1, and t_2) as if that node were actually pruned. The internal node that has the minimum increase in distortion for its given decrease in rate if used is pruned.

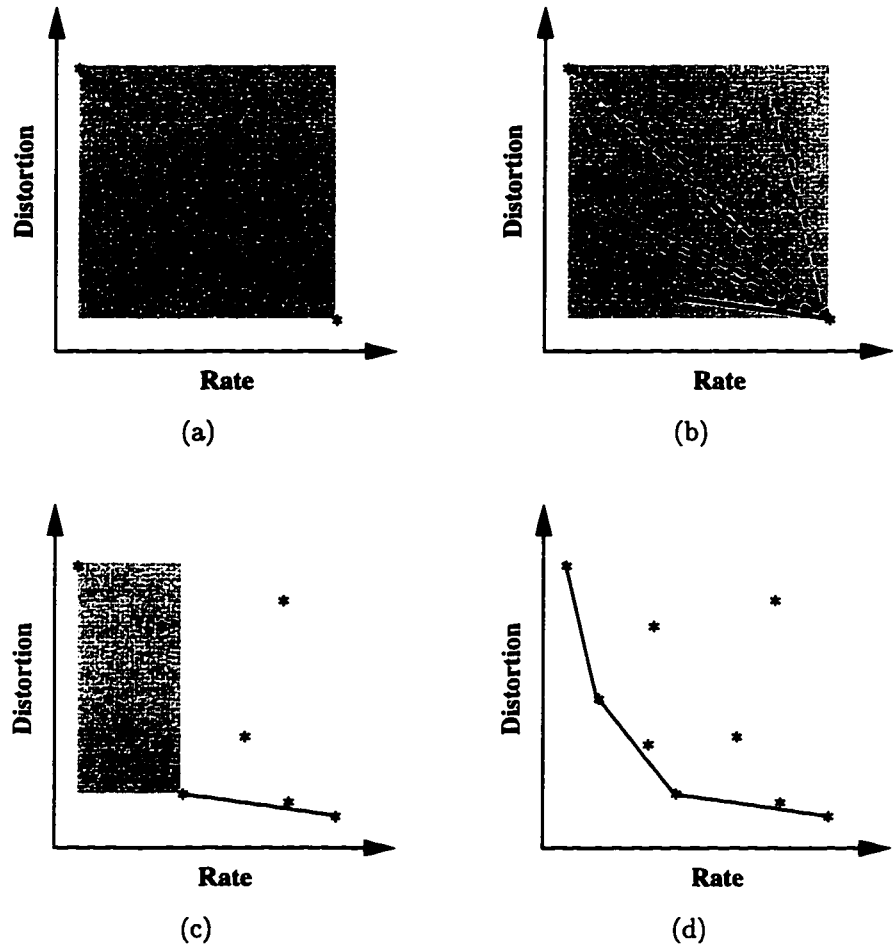


Figure 2.8: GBFOS pruning. Each * represents a possible pruned subtree where the * in the upper left hand corner is the codebook comprised of only the root node and the * in the lower right hand corner is the original TSVQ. (a) At each iteration of the GBFOS algorithm, the space to be examined is limited to the shaded rectangular region between the rate 0 codebook and the current TSVQ since the rate increases monotonically and the distortion decreases monotonically with increasing tree depth. (b) For one iteration of GBFOS, the * in the lower right hand corner of the shaded region is the current subtree, and the other *'s are possible pruned subtrees. The next optimal subtree has the least slope to the current subtree as indicated by the solid line. (c) For the next iteration of GBFOS, a reduced area of the space is examined which is limited to the shaded region between the rate 0 codebook and the current subtree found by the previous GBFOS iteration. (d) After pruning the tree to the root node, all optimal subtrees are identified, and are shown above as linked by the solid line.

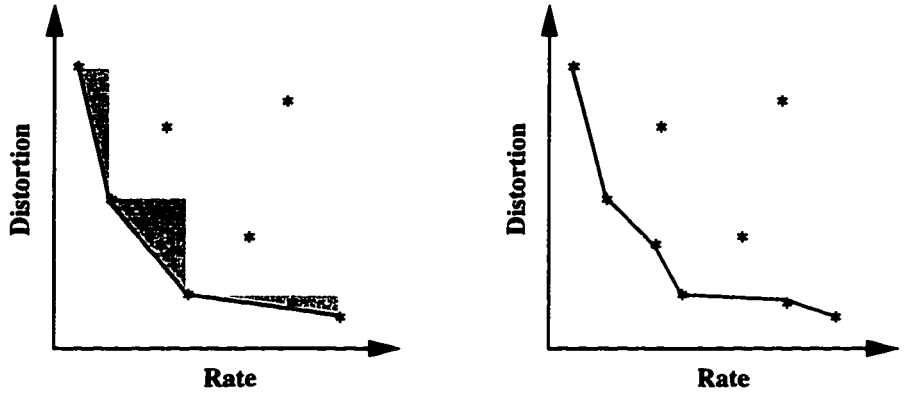


Figure 2.9: ROPA pruning. (a) After each iteration of the GBFOS algorithm, Figure 2.8, a number of pruned codebooks are discarded as possible solutions since they are suboptimal. Some of these suboptimal codebooks lie in the triangular shaded regions bounded by the optimal PTSVQ's, while others lie outside of this region. (b) ROPA works by recursively pruning the tree in these shaded regions, identifying the suboptimal PTSVQ's which lie closest to the optimal rate-distortion curve.

Chapter 3

WAVELETS

Traditional Fourier analysis decomposes a signal into a sum of orthogonal trigonometric functions, giving a frequency representation of the signal. However, Fourier analysis eliminates all temporal or spatial information. The frequency characteristics of a signal change over time or space. It is often desirable in image compression to capture the local frequency behavior of images, especially since a great portion of signal energy in the frequency domain is consumed by discontinuities. A *time-frequency* or space-frequency representation of a signal can be used to capture local frequency behavior of signals and images.

The *wavelet* transform is an important new tool for time-frequency analysis. Using wavelets, one can build simple orthonormal bases of $L^2(\mathbb{R}^d)$ which give good localization in both space and frequency. Wavelets provide a powerful framework for multiresolution analysis and image coding. The wavelet transform decomposes an image into localized orthogonal components which contain different low and high frequency information or scale and translation information. This decomposition provides a useful framework for image compression. The localization property is especially important for handling image features such as abrupt changes due to boundaries or edges.

Wavelet transform theory encompasses the continuous wavelet transform and the discrete wavelet transform. The continuous wavelet transform deals with continuous-time signals which are represented by a wavelet transform that is continuous in both scale and translation. The discrete wavelet transform deals with continuous-time and discrete-time signals which are represented by a wavelet transform that is discrete in both scale and translation. The discrete wavelet transform can be further

divided into wavelet frames and orthonormal wavelets. Wavelet frame theory gives an over-complete or redundant transform. Orthonormal wavelet theory (also called multiresolution analysis theory) gives an orthonormal, non-redundant transform. There are now a number of texts and tutorials available on wavelets [20,31,72,90,91]. For the rest of this chapter, I will concentrate on the discrete orthonormal wavelet transform.

3.1 Discrete Orthonormal Wavelets - One Dimension

The discrete orthonormal wavelet transform (DWT) is defined as a series of coefficients DWT$[m,n]$ resulting from the projection of a signal $f(x) \in L^2(\mathbb{R}^1)$ onto a family of orthonormal basis functions $\psi_{m,n}(x)$ called *wavelets*

$$\text{DWT}[m,n] = \langle f(x), \psi_{m,n}(x) \rangle, \quad m,n \in Z.$$

The inverse discrete wavelet transform simply reconstructs f as

$$f(x) = \sum_{m,n \in Z} \langle f(x), \psi_{m,n}(x) \rangle \psi_{m,n}(x) = \sum_{m,n \in Z} \text{DWT}[m,n] \psi_{m,n}(x).$$

The wavelet functions $\psi_{m,n}(x)$ have a number of interesting properties. The first is that they are all dilated and translated versions of a single function $\psi(x)$ called the *mother wavelet*

$$\psi_{m,n}(x) = 2^{-m/2} \psi(2^{-m}x - n).$$

The variable m controls the *dilation* or *scale* of the wavelet. The variable n controls the *translation* or time shift of the wavelet. As the scale m increases, the $\psi_{m,n}(x)$ become wider and shorter, and the translation step increases. As the scale m decreases, the $\psi_{m,n}(x)$ become thinner and taller, and the translation step decreases. The mother wavelet is defined to have an area of zero. It tends to look squiggly and bumpy like a high-pass filter. As an example, the Haar mother wavelet has been plotted in Figure 3.1(a).

The $\psi_{m,n}(x)$ themselves are orthonormal

$$\langle \psi_{m,n}(x), \psi_{j,k}(x) \rangle = \begin{cases} 0 & m \neq j \text{ or } n \neq k \\ 1 & m = j \text{ and } n = k, \end{cases}$$

and form an orthonormal basis for $L^2(\mathbb{R}^1)$, $\text{span}(\psi_{m,n}(x)) = L^2(\mathbb{R}^1)$. It is convenient to group the wavelets by scale m. These $\{\psi_{m,n}(x); n \in Z\}$ form an orthonormal basis for a space $W_m = \text{span}(\{\psi_{m,n}(x); n \in Z\})$ often called a *detail space*, where $\bigcup_{m \in Z} W_m = L^2(\mathbb{R}^1)$, and $W_m \cap W_n = \emptyset, \quad m \neq n$. Thus the wavelet coefficients are also grouped by scale. These groups are often called *subbands*. The wavelet coefficients for each scale m can be computed as the projection of $f(x)$ onto W_m. Direct as this projection may seem, there is an easier way to compute the wavelet coefficients by using the idea of *multiresolution analysis*.

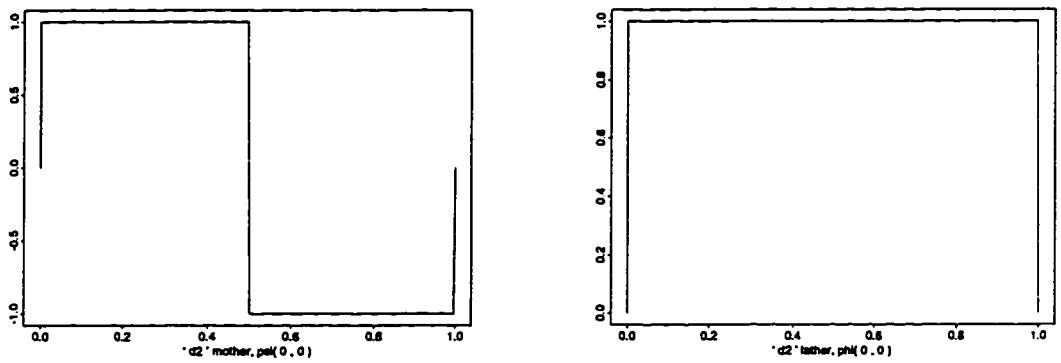
The idea behind multiresolution analysis theory is that a signal can be represented by a series of successive approximations and added detail. A signal $f(x)$ can be broken down into a number of coarse approximations at various resolutions. The difference between two successive approximations is called the detail. Given the lowest resolution (or most coarse) approximation of $f(x)$ and the differences between the successive approximations, multiresolution theory ensures that $f(x)$ can be reconstructed.

Having briefly discussed multiresolution analysis, I next review the *scaling* functions $\phi_{m,n}(x)$ which, like wavelets, have a number of interesting properties. The first is that they are all dilated and translated versions of a single function $\phi(x)$ called the *father* wavelet

$$\phi_{m,n}(x) = 2^{-m/2} \phi(2^{-m}x - n),$$

where the variables m and n affect $\phi_{m,n}(x)$ similarly as the $\psi_{m,n}(x)$ are affected. The father wavelet is defined to have an area of one. It tends to look smooth like a low-pass filter, as in Figure 3.1(b).

The $\phi_{m,n}(x)$ are also grouped by scale m. These $\{\phi_{m,n}(x); n \in Z\}$ form an orthonormal basis for a space $V_m = \text{span}(\{\phi_{m,n}(x); n \in Z\})$, often called the approxi-



(a) (b)

Figure 3.1: The Haar wavelet and scaling functions (a) the mother wavelet $\psi_{0,0}$ (b) the father wavelet $\phi_{0,0}$.

mation space, where $V_m \subset L^2(\mathbb{R}^1)$. The $\phi_{m,n}(x)$ do not form an orthogonal basis for $L^2(\mathbb{R}^1)$ as do wavelets, but the $\bigcup_{m\in Z} V_m = L^2(\mathbb{R}^1)$ and $\bigcap_{m\in Z} V_m = \{0\}$.

An interesting property of the V_m is that they form a *multiresolution ladder* of subspaces

$$\cdots \subset V_{m+1} \subset V_m \subset V_{m-1} \subset \cdots$$

where $f(x) \in V_m$ if and only if $f(2^m x) \in V_0$. The multiresolution ladder property says that each approximation space V_m contains the coarser approximation spaces $(V_j, j > m)$ within it as shown in Figure 3.2(a). By projecting the signal $f(x)$ onto V_m, a coarse approximation $f^m(x) = P_m f(x)$ is formed where $P_m : f(x) \to f^m(x)$. The *resolution* of the approximation is controlled by the scale parameter m. The resolution is said to decrease or become *coarser* as m increases. The resolution is said to increase or become *finer* or more detailed as m decreases. An example of the projection of a Doppler signal onto the Haar scaling functions is shown in Figure 3.3(a).

The fine and coarse spaces V_m and V_{m+1} are related by

$$V_m = V_{m+1} \oplus W_{m+1}$$

where $\oplus$ is the direct sum operator and W_{m+1} is the difference space formed by the wavelets at scale $m+1$ as shown in Figure 3.2(b). The previous equation is recursive and can be rewritten as

$$\begin{aligned} V_m &= V_{m+2} \oplus W_{m+2} \oplus W_{m+1} \\ V_m &= V_{m+3} \oplus W_{m+3} \oplus W_{m+2} \oplus W_{m+1} \\ &\vdots \\ V_m &= V_M \oplus \bigoplus_{k=m+1}^{M} W_k, \quad M > m. \end{aligned}$$

A fine resolution approximation space V_m is the direct sum of a coarse approximation space V_M and all of the difference spaces between W_M and W_{m+1}, (see Figure 3.2(c)).

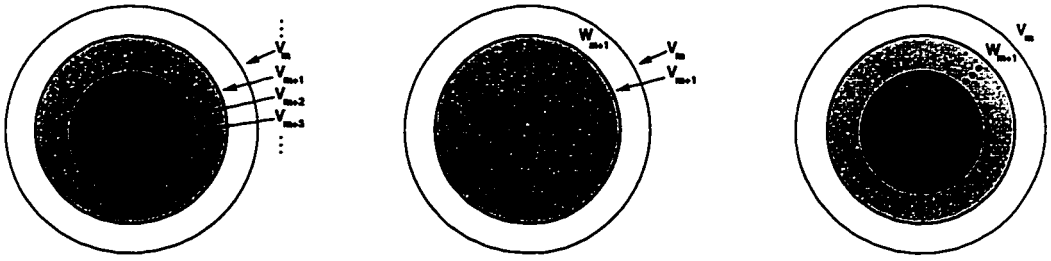
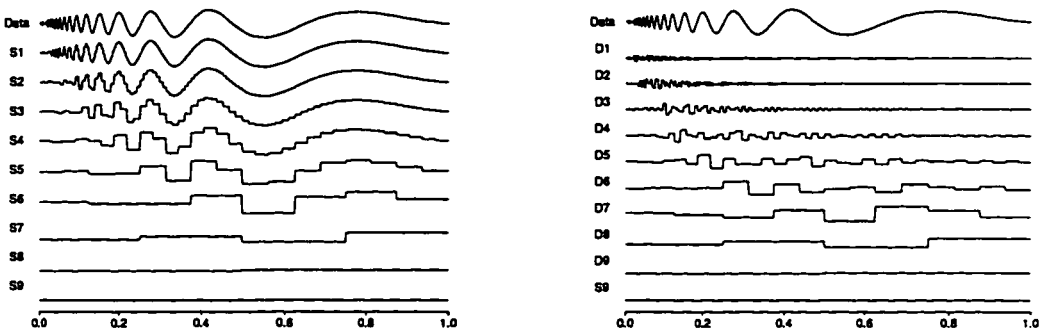


Figure 3.2: Multiresolution spaces (a) multiresolution ladder (b) relationship between approximation space and detail space (c) recursive relationship between approximation space and detail spaces.

Each set of wavelet coefficients at scale m, DWT$[m,n]$, is the projection of the signal f onto a space W_m. As seen by multiresolution analysis, the space W_m represents the difference of two approximations of f, DWT$[m,n] = P_{m-1}f(x) - P_m f(x)$, for $m \in Z$. Hence for a given scale m, DWT$[m,n]$ represents the *detail* between two successive levels of approximation of the signal. In general, $f(x)$ is decomposed as

$$f(x) = P_M f(x) + \sum_{k=M}^{-\infty} (P_{k-1}f(x) - P_k f(x)).$$

The signal $f(x)$ is a low resolution approximation $P_M f(x)$ plus a sum of differences. In Figure 3.3(b), 512 samples of a Doppler signal have been decomposed into a low resolution approximation ($S9$) and differences ($D1, \ldots, D9$) using the Haar wavelet and scaling functions.



(a) (b)

Figure 3.3: Multiresolution decomposition of a Doppler signal (512 samples) using the Haar wavelet and scaling functions (a) coarse approximations where $Sm \in V_m$ (b) differences where $Dm \in W_m$.

The previous discussion is valid for any $f(x) \in L^2(\mathbb{R}^1)$, and a countably infinite set of coefficients is generated. To use wavelet theory for digital signals such as sampled speech or images, it is assumed that the signal samples $f[n]$, $n = 0, \ldots, N-1$ (where $N = 2^L$ for convenience) represent the coefficients of finest approximation $f^0(x)$ of the continuous-time signal $f(x)$ available, that is that

$$f^0(x) = P_0 f(x) = \sum_{n=0}^{N-1} \langle f(x), \phi_{0,n} \rangle \phi_{0,n} = \sum_{n=0}^{N-1} f[n] \phi_{0,n}$$

where $V_0 = \text{span}(\phi_{0,n}(x))$ represents the finest approximation space. In this case $f(x)$ can be decomposed as

$$f^0(x) = P_0 f(x) \;\; = \;\; P_M f(x) + \sum_{m=1}^{M} (P_{m-1} f(x) - P_m f(x)), \quad 0 < M \leq L,$$

where M indicates the number of levels of decomposition. More specifically,

$$f^0(x) = \sum_{n=0}^{2^{L-M}-1} \langle f(x), \phi_{M,n}(x)\rangle \phi_{M,n}(x) + \sum_{m=1}^{M} \left(\sum_{n=0}^{2^{L-m}-1} \langle f(x), \phi_{m-1,n}(x)\rangle \phi_{m-1,n}(x) - \sum_{n=0}^{2^{L-m-1}-1} \langle f(x), \phi_{m,n}(x)\rangle \phi_{m,n}(x) \right), \quad 0 < M \leq L,$$

or

$$f(x) = \sum_{n=0}^{2^{L-M}-1} f^M[n]\phi_{M,n}(x) + \sum_{m=1}^{M} \sum_{n=0}^{2^{L-m}-1} \mathrm{DWT}[m,n]\psi_{m,n}(x), \quad 0 < M \leq L.$$

Since $\phi_{m,n}(x) = 2^{-m/2}\phi(2^{-m}(x - 2^m n))$ and $\phi_{m-1,n}(x) = 2^{-(m-1)/2}\phi(2^{-(m-1)}(x - 2^{m-1}n))$, each $\phi_{m,n}$ has a translation step that is twice as large as the translation step of the next finest basis function $\phi_{m-1,n}$. This means that for a signal $f[n]$ with N samples, where N is a power of two, each set of coefficients produced by $\langle f(x), \phi_{m,n}\rangle$ has half as many coefficients as that produced by $\langle f(x), \phi_{m-1,n}\rangle$. The total number of coefficients is $N/2 + N/4 + \ldots + 1 + 1 = N$. Each projection has to be calculated once and then differences can be taken. This implies that if the wavelet has finite support, the transform can be done with order N computations, [72]. Additionally, by fixing M, the decomposition can be fixed at any number of levels of decomposition from 1 to $\log_2(N)$. The discrete wavelet transform of the Doppler function used in the multiresolution examples is shown in Figure 3.4. Note the resemblance of this to the multiresolution decomposition in Figure 3.3(b); each DWT coefficient is just the magnitude of each wavelet in the multiresolution decomposition. Fast and efficient implementations of the DWT in order N, [72], are available, making the DWT a useful transform for data compression. I use commercially available packages (Section 3.6) whose implementation details can be found in the user documentation.

Now that the detail and approximation spaces have been defined, the $\phi_{m,n}$ and

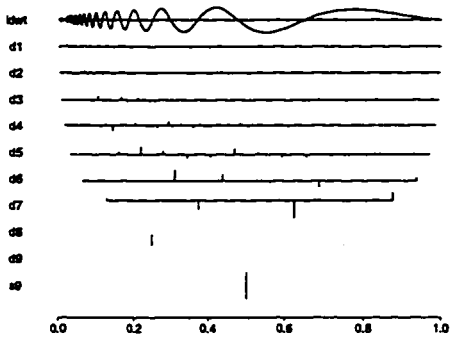


Figure 3.4: The discrete wavelet transform of a Doppler signal (512 samples) using the Haar wavelet and scaling functions.

the $\psi_{m,n}$ can be examined in more detail. It is a fact that

$$\begin{aligned} \phi_{m,n} &\in V_m \subset V_{m-1} \\ \psi_{m,n} &\in W_m \subset V_{m-1}. \end{aligned}$$

Every function that can exist in V_{m-1} must be a linear combination of the $\phi_{m-1,n}$ since $V_{m-1} = \text{span}(\phi_{m-1,n})$. Thus

$$\begin{aligned} \phi(x) &= \sum_n h_n \sqrt{2}\phi(2x-n) \\ \psi(x) &= \sum_n g_n \sqrt{2}\phi(2x-n). \end{aligned}$$

These are known as the *two-scale equations.* The coefficients h_n and g_n are used to identify each wavelet family. They are related to the filter coefficients used in subband filtering which consists of a process of high and low pass filtering followed by downsampling. This subband filtering process is applied recursively to the most recently low-pass filtered portion of the signal. Thus the coarse or approximation space of the DWT corresponds to a low-pass filtered version of the signal, while the detail space of the DWT corresponds to the high-pass filtered version (that is the high frequency components) of the signal.

3.2 *Discrete Orthonormal Wavelets in One Dimension after Quantization: Rate, Distortion, and Entropy*

When doing lossy transform compression, it can be shown that the average rate, average distortion, and entropy of a compressed signal is the same as the average rate, average distortion, and average entropy of the quantized coefficients of the transformed signal, respectively. For example, if the nine detail subbands and one coarse subband of Figure 3.4 are encoded independently, then the rate, entropy, and distortion of the signal after the inverse discrete wavelet transform are the average rate, average entropy, and average distortion of the subbands themselves. This will be convenient in performing optimal bit allocation to all subbands simultaneously.

Let $f(x)$ be the original one-dimensional signal and $\hat{f}(x)$ be the quantized signal. If the mean-square error is used as the distortion measure, then Parseval's relation applies, and the distortion D between the original signal and the quantized signal can be expressed as the mean-square error distortion between the transformed original signal $\mathrm{DWT}[m,n]$ and the transformed quantized signal $\widehat{\mathrm{DWT}}[m,n]$.

$$
\begin{aligned}
D &= E[(f(x)-\hat{f}(x))^2] \\
&= E\left[\left(\sum_{n=0}^{2^{L-M}-1} f^M[n]\phi_{M,n}(x) + \sum_{m=1}^{M}\sum_{n=0}^{2^{L-m}-1} \mathrm{DWT}[m,n]\psi_{m,n}(x) \right.\right. \\
&\qquad \left.\left. - \sum_{k=0}^{2^{L-M}-1} \hat{f}^M[k]\phi_{M,k}(x) - \sum_{j=1}^{M}\sum_{k=0}^{2^{L-j}-1} \widehat{\mathrm{DWT}}[j,k]\psi_{j,k}(x)\right)^2\right] \\
&= \frac{1}{2^L}\sum_{n=0}^{2^{L-M}-1}(f^M[n]-\hat{f}^M[n])^2 + \frac{1}{2^L}\sum_{m=1}^{M}\sum_{n=0}^{2^{L-m}-1}(\mathrm{DWT}[m,n]-\widehat{\mathrm{DWT}}[m,n])^2,
\end{aligned}
$$

since $\langle\psi_{m,n}(x),\psi_{j,k}(x)\rangle = \delta_{m-j,n-k}$ and $\langle\phi_{M,n}(x),\psi_{m,n}(x)\rangle = 0$ if $M \geq m$.

If D_m is defined to be the MSE distortion of the detail subbands $m = 1,\ldots,M$, and D_M^{coarse} is defined to be the MSE distortion of the coarse approximation subband, then the overall distortion is equal to the average mean-square error over all of the

subbands:

$$\begin{aligned} D_m &= \frac{1}{2^{L-m}} \sum_{n=0}^{2^{L-m}-1} (\mathrm{DWT}[m,n] - \widehat{\mathrm{DWT}}[m,n])^2, \\ D_M^{coarse} &= \frac{1}{2^{L-M}} \sum_{n=0}^{2^{L-M}-1} (f^M[n] - \hat{f}^M[n])^2, \\ D &= \frac{1}{2^M} D_M^{coarse} + \sum_{m=1}^{M} \frac{1}{2^m} D_m. \end{aligned}$$

The entropy of the compressed signal can also be found as a function of the entropy of the subbands. The entropy H of the entire compressed signal is defined as

$$H = -\sum_{i \in I} P_i \log P_i$$

where the signal is encoded with symbols $i \in I$, and each symbol is used with probability P_i and $\sum_i P_i = 1$. By defining $p_{i,m}$ as the probability of symbol i in subband m, the entropy of each subband, H_m, can be expressed as

$$H_m = -\textstyle\sum_i p_{i,m} \log p_{i,m}, \quad \text{where} \quad \sum_i p_{i,m} = 1.$$

I now rewrite the overall entropy H by grouping together the symbols used by the coarse approximation subband H_M^{coarse} and the detail subbands H_m. I assume that each symbol i is found in only one subband m and so $P_i = P_{i,m}$:

$$H = -\sum_{i \in I} P_i \log P_i = -\sum_i P_{i,M}^{coarse} \log P_{i,M}^{coarse} - \sum_{m=1}^{M} \sum_i P_{i,m} \log P_{i,m},$$

where

$$\sum_i P_{i,M}^{coarse} + \sum_{m=1}^{M} \sum_i P_{i,m} = 1.$$

The probability $P_i = P_{i,m}$ is related to the probability $p_{i,m}$ of a symbol i in subband m by

$$P_{i,m} = p_m p_{i,m},$$

where p_m is the probability of the subband and $p_M^{coarse}+\sum_{m=1}^{M} p_m = 1$. By substituting this relationship into the overall entropy equation it is found that the entropy of the signal is equal to the "average" encoding entropy over all of the subbands plus the entropy of the subbands themselves.

$$\begin{aligned}
H &= -\sum_i p_M^{coarse} p_{i,M}^{coarse} \log(p_M^{coarse} p_{i,M}^{coarse}) - \sum_{m=1}^{M} \sum_i p_m p_{i,m} \log(p_m p_{i,m}) \\
&= -p_M^{coarse} \sum_i p_{i,M}^{coarse} [\log p_{i,M}^{coarse} + \log P_M^{coarse}] - \sum_{m=1}^{M} p_m \sum_i p_{i,m} [\log p_{i,m} + \log p_m] \\
&= -p_M^{coarse} \sum_i p_{i,M}^{coarse} \log p_{i,M}^{coarse} - \sum_{m=1}^{M} p_m \sum_i p_{i,m} \log p_{i,m} \\
&\quad - p_M^{coarse} \log p_M^{coarse} \sum_i p_{i,M}^{coarse} - \sum_{m=1}^{M} p_m \log p_m \sum_i p_{i,m} \\
&= p_M^{coarse} H_M^{coarse} + \sum_{m=1}^{M} p_m H_m - p_M^{coarse} \log p_M^{coarse} - \sum_{m=1}^{M} p_m \log p_m.
\end{aligned}$$

The last two terms represent the entropy of the subbands, i. e. the cost of encoding the information of which subband each coefficient belongs to. However, the assignment of subbands is not really random, but is known. Therefore, it is not necessary to include these terms in the entropy calculations. Thus

$$H = p_M^{coarse} H_M^{coarse} + \sum_{m=1}^{M} p_m H_m.$$

For one-dimensional wavelets, $p_m = \frac{1}{2^m}$, and the entropy is

$$H = \frac{1}{2^M} H_M^{coarse} + \sum_{m=1}^{M} \frac{1}{2^m} H_m.$$

Similarly, the average rate R of the quantized output is equal to the average rate over all of the subbands

$$R = \frac{1}{2^M} R_M^{coarse} + \sum_{m=1}^{M} \frac{1}{2^m} R_m$$

where R_m is the rate of each detail subband and R_M^{coarse} is the rate of the coarse subband.

3.3 Discrete Orthonormal Wavelets - Two Dimensions

The DWT for multidimensional signals $f(x_1, x_2, \cdots, x_d) \in L^2(\mathbb{R}^d)$ can be extended to two or more dimensions by using tensor products of one-dimensional multiresolution analyses. For the two-dimension case $f(x, y) \in L^2(\mathbb{R}^2)$, the coarse approximation space $\mathbf{V}_m$ is defined as the tensor product of two 1-dimensional approximation spaces, $\mathbf{V}_m = V_m \otimes V_m = \text{span}(\{\Phi_{m;i,j}(x, y); i, j \in Z\})$, where

$$\Phi_{m;i,j}(x, y) = \phi_{m,i}(x)\phi_{m,j}(y) \in V_m \otimes V_m$$

is the father wavelet, m is the scale, and $i \in Z$ and $j \in Z$ are the translations in x and y. The same properties for one-dimension approximation spaces V_m also hold for two-dimension approximation spaces $\mathbf{V}_m$: $\bigcup_{m \in Z} \mathbf{V}_m = L^2(\mathbb{R}^2)$; $\bigcap_{m \in Z} \mathbf{V}_m = \{0\}$; and the $\mathbf{V}_m$ form a multiresolution ladder of subspaces

$$\cdots \subset \mathbf{V}_{m+1} \subset \mathbf{V}_m \subset \mathbf{V}_{m-1} \subset \cdots$$

where $f(x, y) \in \mathbf{V}_m$ if and only if $f(2^m x, 2^m y) \in \mathbf{V}_0$.

The wavelet functions can be found by expanding the tensor product

$$\begin{aligned} \mathbf{V}_{m-1} &= V_{m-1} \otimes V_{m-1} = (V_m \oplus W_m) \otimes (V_m \oplus W_m) \\ &= (V_m \otimes V_m) \oplus [(W_m \otimes V_m) \oplus (V_m \otimes W_m) \oplus (W_m \otimes W_m)] \\ &= \mathbf{V}_m \oplus \mathbf{W}_m. \end{aligned}$$

From this, it can be seen that the detail space $\mathbf{W}_m$ has three wavelets:

$$\begin{aligned} \Psi^1_{m;i,j}(x, y) = \phi_{m,i}(x)\psi_{m,j}(y) &\in V_m \otimes W_m, \\ \Psi^2_{m;i,j}(x, y) = \psi_{m,i}(x)\phi_{m,j}(y) &\in W_m \otimes V_m, \\ \Psi^3_{m;i,j}(x, y) = \psi_{m,i}(x)\psi_{m,j}(y) &\in W_m \otimes W_m, \end{aligned}$$

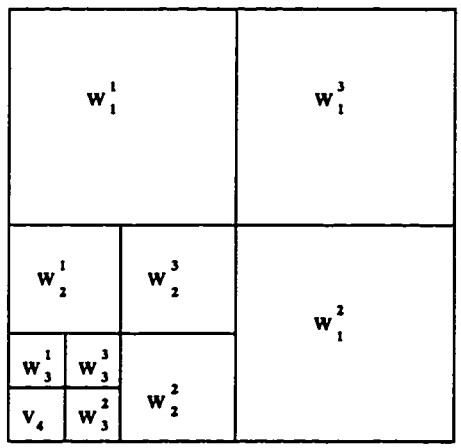
where Ψ^1, Ψ^2, and Ψ^3 respectively represent the horizontal, vertical, and diagonal components of the two-dimensional signal at scale m. Thus the detail space

$$\mathbf{W}_m = \{W^n_m; n = 1, 2, 3\} = \text{span}(\{\Psi^n_{m;i,j}; i, j \in Z, n = 1, 2, 3\})$$

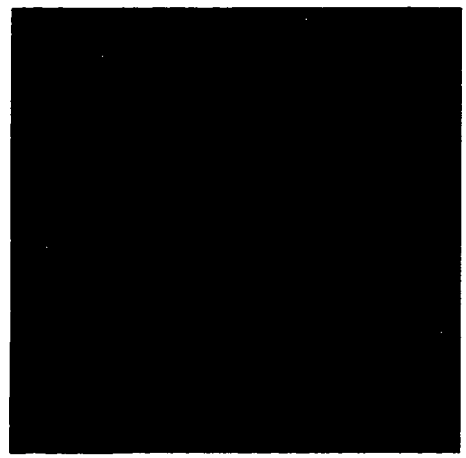
where $W_m^n, n = 1, 2, 3$ refer to the horizontal, vertical, and diagonal detail spaces respectively. Again, the properties for one-dimension detail spaces W_m also hold for two-dimension detail spaces $\mathbf{W}_m$. The $\mathbf{W}_m$ are orthonormal and form an orthonormal basis for $L^2(\mathbb{R}^2)$:

$$W_m^k \bigcap W_m^n = \emptyset, \qquad k \neq n,$$
$$\mathbf{W}_k \bigcap \mathbf{W}_m = \emptyset, \qquad k \neq m,$$
$$\bigoplus_{m \in Z} \mathbf{W}_m = L^2(\mathbb{R}^2).$$

The arrangement of coefficients is shown in Figure 3.5(a). An example of the two-dimension discrete wavelet transform is shown in Figure 3.5(b) where the MR image of "Eve's brain" is decomposed to three levels using the S8 wavelet.



(a) The arrangement of the coefficients of the two-dimension DWT.



(b) The two-dimension DWT of the MR image "Eve's brain" using the S8 wavelet with three levels of decomposition.

Figure 3.5: The discrete wavelet transform in two dimensions: (a) the arrangement of the coefficients of a three level decomposition, and (b) the MR image "Eve's brain" image decomposed to three levels.

3.4 *Discrete Orthonormal Wavelets in Two Dimensions after Quantization: Rate, Distortion, and Entropy*

As shown before, the average rate and average distortion of a transform compressed signal can be derived as the average rate and average distortion of the quantized coefficients of the transformed signal. This will be convenient in performing optimal bit allocation to all subbands simultaneously.

Assume that the two-dimensional signal to be decomposed, $f[i,j]$ is an $N \times N$ signal where $N = 2^L$. Then L is the maximum number of decomposition levels that can be computed. Let M be the highest level of decomposition performed on the signal where $1 \leq M \leq L$. Define the mean square error distortion of the detail subbands to be $D_{m,n}$, where $1 \leq m < M$ refers to scale and $n = 1, 2, 3$ refers to the three wavelet functions. The mean square error distortion of the coarse approximation subband is defined to be $D_{M,0}^{coarse}$. The overall distortion is equal to the average mean-square error over all of the subbands

$$D = \frac{1}{2^{2M}} D_{M,0}^{coarse} + \sum_{m=1}^{M} \sum_{n=1}^{3} \frac{1}{2^{2m}} D_{m,n}.$$

Entropy is similarly derived as in Section 3.2 where the entropy is the average encoding entropy over all subbands:

$$H = p_{M,0}^{coarse} H_{M,0}^{coarse} + \sum_{m=1}^{M} \sum_{n=1}^{3} p_{m,n} H_{m,n}$$

where $p_{m,n}$ is the probability of a subband at scale m and orientation n. For the two-dimensional case, $p_m = \frac{1}{2^{2m}}$, and the entropy is

$$H = \frac{1}{2^{2M}} H_{M,0}^{coarse} + \sum_{m=1}^{M} \sum_{n=1}^{3} \frac{1}{2^{2m}} H_m.$$

Similarly, the average rate R of the quantized output is equal to the average rate over all of the subbands

$$R = \frac{1}{2^{2M}} R_{M,0}^{coarse} + \sum_{m=1}^{M} \sum_{n=1}^{3} \frac{1}{2^{2m}} R_{m,n}$$

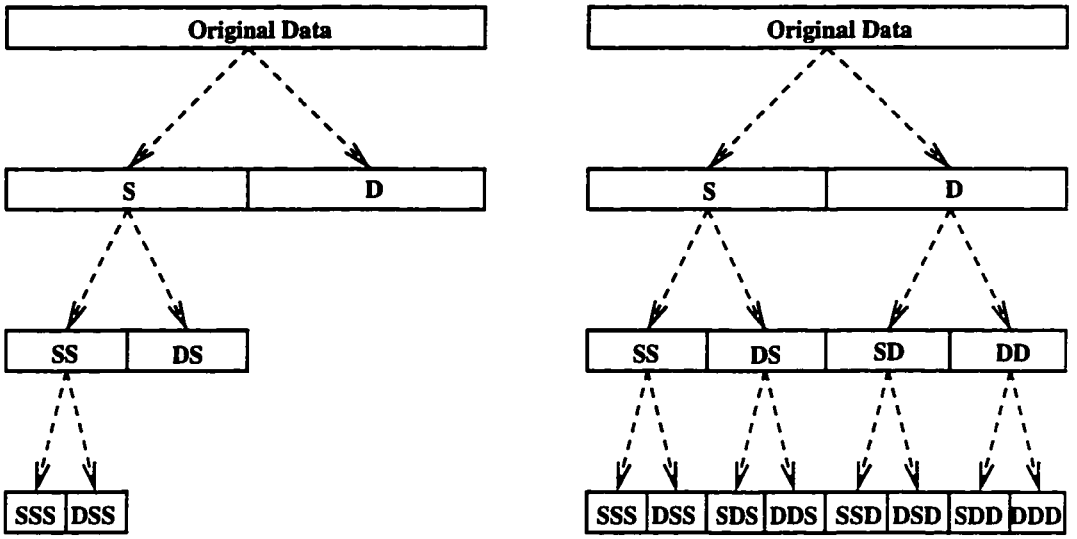
where $R_{M,0}^{coarse}$ and $R_{m,n}$ are the rates of the coarse and detail subbands respectively.

3.5 Wavelet Packets

Orthogonal wavelet packets are a direct extension of the filtering ideas of the DWT. For wavelet packets, each wavelet subband is further decomposed into smooth and detail spaces which correspond to different orthogonal frequency bands of the signal. In Figure 3.6(a), the subbands of the DWT with increasing levels of decomposition are shown in a tree structure which indicates the decomposition process. Wavelet packet decomposition with orthogonal subbands is shown in Figure 3.6(b), also in a tree structure to indicate the decomposition process. This structure is known as an isotropic wavelet packet tree (WPT) [97]. For wavelet packets, all subbands are further decomposed to smooth and detail subbands. Since each space is the direct sum of its subspaces, the original space can be represented by any subtree of the wavelet packet tree. For example, the DWT is one possible subtree of the WPT. Fast and efficient implementations of the wavelet packet transform can be done in order $N \log N$, which is equivalent to the time required by the fast Fourier transform. The use a predetermined subtree of the WPT, such as the DWT, will reduce computation time. The main advantage of wavelet packets is better signal representation by using wavelet packet frequency subbands which are adapted to the frequency characteristics of the signal. The search for the best representation of the signal by any subtree of the WPT is called *best basis selection* [24].

Unfortunately, the clear physical meaning of multiresolution analysis is lost when wavelet packets are used. However, wavelet packets can be exploited to better represent signals which do not have lowpass frequency characteristics. Quantization of wavelet packets is done in the same manner as the DWT since the wavelet packet decomposition is a linear, orthonormal transform. The rate and distortion of the original signal can be expressed as weighted sums of the rate and distortion of the

subspaces used to decompose the original signal. The entropy can be expressed as weighted sums of the entropy of the encoded subbands where each subband has its own quantizer.



(a) DWT.

(b) Wavelet packets.

Figure 3.6: (a) The discrete wavelet decomposition shown for a one-dimensional signal decomposed to three levels. (b) The wavelet packet decomposition shown for a one-dimensional signal decomposed to three levels. **S** and **D** refer to smooth and detail subbands. Note that the DWT is one possible subtree of the WPT.

3.6 Implementation

I have used both the S+WAVELETS toolkit developed by the Statistical Sciences Division, of MathSoft, Inc. and the WaveLab .600 package by Buckheit *et al.* at Stanford University. I have been an alpha and beta test site for the S+WAVELETS package.

Chapter 4

WAVELETS AND QUANTIZATION

In this chapter, I first describe current wavelet compression techniques in Sections 4.1. Next, in Section 4.2, I review several current bit allocation techniques for wavelet compression. I then introduce PTSVQ for wavelet-transform compression with bit allocation via GBFOS and ROPA in Sections 4.2.1 and 4.2.2, and PTSVQ for wavelet packet compression in Section 4.2.3.

4.1 Wavelet Compression

The discrete wavelet transform combined with scalar quantization (SQ) and VQ has led to numerous schemes for compression [2,4,8,9,12–15,24,26,32,33,36,40,73,79,80, 86,88,96]. Using a multiresolution framework, the wavelet transform organizes the coefficients to enable effective SQ and VQ encoding. The temporal/scale tessellation of the wavelet transform avoids block artifacts common with schemes using only temporal tessellation (e. g., JPEG and pure VQ coding schemes).

Wavelet encoding involves taking the discrete-time wavelet transform of the data and quantizing the wavelet subbands. The design of the quantizers is based on statistical analyses of the wavelet coefficients for sample data and requires careful study. In particular, there are two issues in particular that must be addressed. The first is bit allocation, i. e. the assignment of the rate for each of the wavelet subbands, and the second is the size of the vectors for each wavelet scale.

Bit allocation is the process of assigning a given number of bits to a set of different sources (e. g. wavelet subbands) to minimize the overall distortion of a coder. For

uniform scalar quantization, the bit allocation scheme chooses the size of the quantizer step, or bin, for each source. For VQ, the bit allocation scheme controls the vector dimension and the number of codewords for each source. A good bit allocation method usually results in much better performance by devoting more bits to regions of the data that are active or difficult to code and fewer bits to less active regions. After quantization the quantizer indices are usually entropy encoded.

The vector dimensions of each VQ may vary so that the codebook for each subband may have different numbers of vectors in it. The choice of vector dimension is usually made on an *ad hoc* basis, and is affected by the amount of training data available and the complexity of the vector quantizer. Theoretically, compression performance improves as the block size, or dimension, increases, but the trade-off of using a higher dimension is increased coding complexity. The goal is to design the coder that offers the highest quality while still providing real-time decoding and reasonable encoding complexity in software. In [29], vector dimension and bit allocation is determined by using the approximate bound on minimum distortion for high rate VQ for a given dimension. This bound is used to develop an algorithm for joint optimization of vector dimension and bit allocation using marginal analysis. In [11, 14], vector dimension is related to design and encoding complexity. Bit allocation is done by using a non-linear bit allocation algorithm that tries to minimize distortion with constraints on both rate and complexity. As I will be attempting low rate compression, I will use *ad hoc* dimension assignments that will be influenced by the compression application, and when possible, I will examine the effects of different vector dimensions.

4.1.1 Wavelet Scalar Quantization

In one of the biggest success stories of wavelets to date, the FBI has adopted a wavelet/scalar quantization (WSQ) standard for fingerprint image compression [13, 28] which uses an adaptive bit allocation scheme. Each subband has a different quantization step which is determined by the energy of the subband. The bin size is

a function of the image and must be stored with the quantized coefficients. In [15], the authors intend to improve the FBI's WSQ compression algorithm by using an optimal bit allocation technique to choose subband bin sizes.

The embedded zero-tree wavelet algorithm (EZW) of Shapiro is an iterative WSQ algorithm that uses a fixed threshold at each iteration to determine which coefficients across subbands in spatially corresponding position are to be quantized and encoded. The threshold is set to be the bin size of the uniform scalar quantizer which is used to quantize the coefficients. The threshold and quantization step are refined by one-half at each iteration and the algorithm results in an embedded code. The iterations repeat until all of the available bits are allocated. The quantization step used at each iteration applies to all of the subbands, unlike the previous methods which use different bin sizes for each subband. So only the original quantization step needs to be stored as overhead. This algorithm does not require any training. The EZW algorithm has recently received a great deal of attention, and the development of EZW-like algorithms is an active research area, see for example, [68, 85].

In [33], bit allocation based on wavelet coefficient magnitude is proposed, that is, only the N largest coefficients are preserved. Side information includes the positions and bit size of each preserved coefficient. An explanation of how bin width choice in wavelet scalar quantization is related to the error incurred in wavelet scalar quantization compression is also presented.

4.1.2 Wavelets and Vector Quantization

A comprehensive description of current wavelet VQ (WVQ) techniques can be found in [26]. I now briefly summarize several important WVQ papers.

In [2], full search vector quantization is performed on each wavelet subband creating a multiresolution codebook by using a mean-square error distortion measure. Bit allocation uses a total weighted mean-square error measure to improve the subjective image quality and is done by a noise shaping procedure which is a function of the

image and requires overhead to store. In [14] a multiresolution WVQ codebook is used which does bit allocation subject to constraints on rate and encoding complexity while minimizing distortion. In [11], full search VQ is also used in a WVQ scheme. A non-linear optimization scheme is used which involves modeling of rate, distortion, and complexity of the encoder to design the VQ's for each subband. This was meant to help select vector dimension size for each VQ.

In [87], fixed-rate full search VQ, lattice VQ and entropy-constrained VQ (ECVQ) are used to encode wavelet coefficients. The best results are found with lattice VQ and entropy-constrained VQ. The authors recommend using variable rate coding to improve performance. In addition it is also shown that to achieve optimal bit allocation across subbands, the slopes of the rate-distortion curves of each subband must be the same. Entropy-constrained VQ has also been used in [86] to produce a multiresolution codebook. Optimal bit allocation cannot be performed due to the nature of the distortion of the ECVQ encoder, but DPCM is used on the lowest resolution subbands, and Riskin's algorithm [82] is used to allocate bits to the rest of the subbands which use the multiresolution codebook.

The embedded zero-tree wavelet algorithm has been extended to work with VQ in [27, 30, 60]. For example the EZW is used with variable-rate TSVQ's in [27, 60]. In [27], at each iteration of the algorithm, spatially corresponding vectors across subbands are examined based on a threshold and encoded. Since uniform scalar quantization is not being used, there is no direct link between the threshold and the quantizer bin size as in the EZW algorithm. Instead a variable threshold is used that is determined by rate-distortion trade-offs. The iterations repeat until all of the available bits are allocated. This algorithm requires training and storage for the TSVQ's unlike the EZW algorithm which requires storing only the initial quantization step size. In [60] the authors present a similar method, except that they use mean-removed unit-variance normalized data. Unfortunately they do not explain how thresholding is done to determine significance of coefficients. In [30], the EZW is

extended to use regular-lattice VQ codebooks that consider vector orientation during the encoding phase.

Although the primary concern in [50] is progressive image transmission, the work described can be considered a WVQ image compression method. In this method, an increasing number of bits is arbitrarily allocated to an image at increasing resolutions. The coarse approximation is allocated a certain number of bits, then the coarse approximation with the next finer detail bands is allocated a larger number of bits, and so on. A PTSVQ is used in each detail subband. Bit allocation is done incrementally until the assigned rate is achieved. Bit allocation to each subband can only increase, so bit allocation for the entire image is not optimal since the *ad hoc* method of assigning rate to the image as resolution increases is not optimal.

4.1.3 Wavelets and Quantization for Video Compression

There have been numerous attempts to code image sequences with a wavelet approach, including [34, 43, 66, 67, 100], and research in this area has been rapidly increasing the last several years. In [66] a three-dimensional wavelet transform is done on eight consecutive frames (each frame is decomposed by a two-dimensional wavelet transform and then a one-dimensional wavelet transform is done over the time dimension). The goal is to preserve spatial and temporal edges during compression. The compression algorithm tries to predict which wavelet coefficients contribute to edges. The algorithm preserves those coefficients that likely contribute to edges and discards those that do not. Four bits are used for the preserved coefficient and one bit is used to indicate a discarded coefficient. Clearly, bit allocation is not optimal. The prediction is first performed in the spatial domain, and then performed in the time domain. In [43], this idea has been extended to predict which coefficients might contribute to an edge by performing prediction simultaneously in space and time.

In [100], several video compression schemes using the two-dimensional wavelet transform and motion compensation are compared. The first approach takes the

wavelet transform of each frame and then uses multiresolution motion compensation. For multidimensional motion compensation, motion vectors are found in the lowest resolution subband (or alternatively the lowest pyramid level) and either all other subbands use scaled versions of these motion vectors, or the motion vectors are used to predict all of the other subbands' motion vectors. The residual images are quantized and entropy encoded. The second approach is to perform interframe motion compensation and perform the wavelet transform on the residual image. The wavelet coefficients are quantized and entropy encoded. The system with the best results first did a wavelet transform, then did multiresolution motion compensation, and lastly encoded the coefficients with classified VQ. However motion compensation with wavelet decomposition of residual images encoded by VQ was not examined.

4.1.4 Wavelet Packet Compression

Several recent papers have addressed adaptive wavelet coding where the cost of encoding a signal that has undergone an orthogonal wavelet packet decomposition is minimized by choosing the best subtree of the wavelet packet tree (WPT). Perhaps the most important are [24] and [47, 79] which also address bit allocation. In [24], wavelet packet best basis assignment is examined using various cost measures including entropy-based constraints. It is proved that the best basis can be found using an iterative algorithm from the bottom of the WPT to the top where the costs of encoding the smaller subspaces and the cost of encoding the larger subspace comprised of the smaller subspaces are compared. The larger subspace is split into the smaller subspaces if the total cost of encoding the smaller subspaces is less than the cost of encoding the larger subspace. Otherwise, the larger, merged, subspace is used to encode the signal.

The methods in [47,79] address joint bit allocation and best basis selection. They are known as the single-tree and double-tree methods. In the single-tree method [79], quantizers are designed for every wavelet packet subband, and the rate-distortion

characteristics of each quantizer are computed. This method uses Lagrangian solution techniques, and it is shown that a bit assignment is optimal when all quantizers have the same slope on their local rate-distortion curves. The single-tree method works by first finding the best bit assignments at each node of the WPT for a given slope. Next, it finds the best wavelet packet assignment given this information. This process is iteratively repeated using different slope values until the bit budget is reached. It is shown that the cost function is convex so that it is possible to get convergence of the algorithm.

The double-tree method, [47], involves joint bit allocation and best basis assignment for signal segmentation in time. It works by finding the best single-tree assignment for the entire signal. The signal is then broken into different segments, and the best single-tree bit allocations for each of these time segments are found. Then these segments are segmented, and the best basis assignments are found, and so on. Once all of the possible segmentations and best quantization assignments for each segment are computed, the best overall bit assignment among all possible time-segmentations is found. This method was also developed for nonstationary signals. A good overview of the evolution of adaptive wavelet techniques and their applications can be found in [80].

4.2 Compression using Wavelets and PTSVQ

Studies have found that wavelet coefficients of images tend to have distributions which are highly peaked around zero [2]. Following [2,9,75] among others, I use VQ to code these coefficients. The main difference is that I use PTSVQ [23,81] as described earlier because it offers higher quality and has lower encoding complexity. It has also been found that PTSVQ works particularly well when the data have a distribution that is highly peaked [81]. Simulations of PTSVQ applied to prediction residuals, which are highly peaked around zero, in [81] gave excellent compression results, in part

due to suitability of PTSVQ to data with highly nonuniform distributions. Hence PTSVQ should offer excellent compression results when applied to wavelet data. As an example, Figure 4.1(a) shows an MR image of "Eve's brain" compressed with full search VQ and Figure 4.1(b) shows the same image compressed with PTSVQ. Both images have the same bit rate, but the PTSVQ image is of much higher image quality (even in the halftoned versions shown here) and measures a gain in SNR of 3.78 dB. In addition, PTSVQ gives smart bit allocation automatically because it devotes more bits to high distortion events, such as edges, and saves bits in lower distortion regions of the image, such as the background. Therefore my multiresolution codebook will consist of a PTSVQ codebook for each subband. I will use the following bit allocation algorithms to simultaneously and conveniently allocate bits among the wavelet subbands.

(a) The MR image compressed to 1.5 bpp using full search VQ.

(b) The MR image compressed to 1.5 bpp using PTSVQ.

Figure 4.1: The MR image "Eve's brain" compressed to 1.5 bits per pixel using (a) full search vector quantization and (b) pruned tree-structured vector quantization with an improvement is SNR of 3.78 dB.

4.2.1 Bit Allocation Using GBFOS for Wavelets and PTSVQ

For each subband, I create a very high-rate tree-structured VQ. I then treat each of these subband TSVQ codebooks as a source needing bit allocation. I simply use GBFOS pruning, described in Section 2.5.1, on all of the wavelet subbands together to do the bit allocation, as suggested in [23]. This will ensure that all subbands are given equal consideration while doing bit allocation for the entire set of data, Figure 4.2. This approach was also used by Oehler and Gray to simultaneously assign bits to a mean-shape-gain VQ with good results [76]. Using GBFOS as a form of bit allocation works well with TSVQ's since it is an extension of pruning a single tree. As done in pruning, bit allocation will be done by assigning high rate TSVQ's to the sources. The TSVQ's will then be pruned among the subbands until the desired rate is achieved. Alternatively, the TSVQ's can be pruned before bit allocation and a list of the optimal pruned subtrees and their performance characteristics can be made. Bit allocation is done by removing the quantizer with the least slope on the rate-distortion curve until the desired rate is achieved.

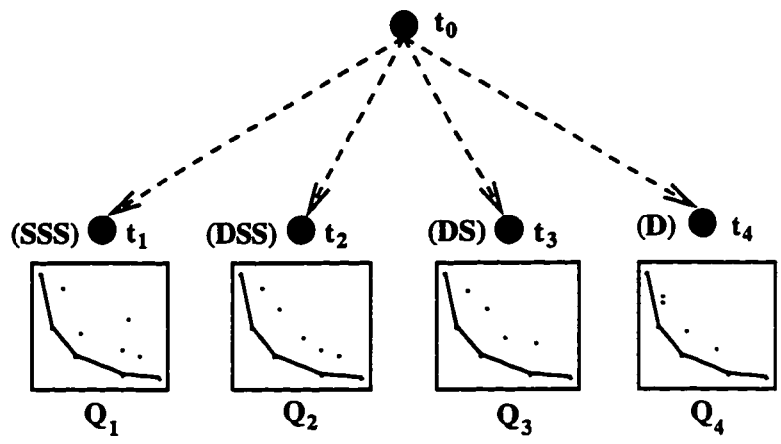


Figure 4.2: Discrete wavelet transform bit allocation for a one-dimensional signal decomposed to three levels. Each frequency subband of the wavelet packet tree has an associated TSVQ. Bit allocation proceeds by finding each GBFOS optimal quantizer as the slope of the rate-distortion function increases. **S** and **D** refer to smooth and detail subbands.

The problem is to allocate bits among the subbands such that the distortion D is minimized subject to the number of bits, or rate R, available where D and R are defined in Chapter 3.4. This can be done by finding the λ which minimizes the Lagrangian functional (shown here for the 2-dimension DWT)

$$J(D,R) = \frac{1}{2^{2M}} D_{M,0}^{coarse} + \sum_{m=1}^{M} \sum_{n=1}^{3} \frac{1}{2^{2m}} D_{m,n} + \lambda \left(\frac{1}{2^{2M}} R_{M,0}^{coarse} + \sum_{m=1}^{M} \sum_{n=1}^{3} \frac{1}{2^{2m}} R_{m,n} \right).$$

By differentiating $J(D,R)$ with respect to the rate allocated to each subband $R_{i,j}$ where $(i = 1, 2, \cdots, M; j = 1, 2, 3)$ and $(i = M; j = 0)$, and by setting the results to zero,

$$\begin{aligned} \frac{\partial J(D,R)}{\partial R_{i,j}} &= \frac{\partial}{\partial R_{i,j}} \left[\frac{1}{2^{2M}} D_{M,0}^{coarse} \right] + \frac{\partial}{\partial R_{i,j}} \left[\sum_{m=1}^{M} \sum_{n=1}^{3} \frac{1}{2^{2m}} D_{m,n} \right] + \\ &\quad \lambda \frac{\partial}{\partial R_{i,j}} \left[\frac{1}{2^{2M}} R_{M,0}^{coarse} \right] + \lambda \frac{\partial}{\partial R_{i,j}} \left[\sum_{m=1}^{M} \sum_{n=1}^{3} \frac{1}{2^{2m}} R_{m,n} \right] \\ &= \frac{1}{2^{2M}} \left[\frac{\partial D_{M,0}^{coarse}}{\partial R_{i,j}} + \lambda \frac{\partial R_{M,0}^{coarse}}{\partial R_{i,j}} \right] + \sum_{m=1}^{M} \sum_{n=1}^{3} \frac{1}{2^{2m}} \left[\frac{\partial D_{m,n}}{\partial R_{i,j}} + \lambda \frac{\partial R_{m,n}}{\partial R_{i,j}} \right] = 0, \end{aligned}$$

the value of λ which minimizes the Lagrangian functional is found to be $\lambda = -\frac{\partial D_{m,n}}{\partial R_{m,n}}$. Note that λ is the same for all subbands at scale m with orientation n, including the coarse approximation subband, and that the subband quantizers are assumed to be normalized by vector dimension as well as fraction of the signal space. Optimal bit allocation is achieved when the optimal quantizers of each subband have the same slope on the lower convex hull of the rate-distortion curve. This is known as the Pareto-optimality condition in economics [87]. Thus it is possible to use the rate-distortion information of each subband directly without further processing or scaling of the rate and distortion values of the quantizers of each subband by 2^{-2m}. The TSVQ's of the subbands can be pruned by just removing the node with the least value of λ among all of the subbands until the desired rate is reached.

The GBFOS bit allocation algorithm is given in Table 4.1 where it is assumed that there are M wavelet subbands, and that each subband i has an associated TSVQ

Q_i. Each Q_i has an associated rate $R(Q_i)$, distortion $D(Q_i)$, and minimum slope $\lambda(Q_i)$. The multiresolution codebook Q has an overall rate of $R(Q) = \sum_{i=1}^{M} R(Q_i)$, and distortion of $D(Q) = \sum_{i=1}^{M} D(Q_i)$. Bit allocation continues until there are no quantizers to be pruned, $\lambda_{min} = \infty$, or the desired rate, R_{target}, has been reached. It works by identifying the minimum slope quantizer, pruning it, and calculating the new rate and distortion.

The pruning of TSVQ's yields a set of optimally pruned TSVQ's for each of the subbands whose joint performance lies on the lower convex hull of the rate-distortion curve of the multiresolution codebook, i. e. the multiresolution codebooks have the lowest distortion for their average bit rate. For example, if one subband quantizer Q_j is pruned to $\widehat{Q}_j$, and its slope is $\lambda(Q_j) = -\frac{D(\widehat{Q}_j)-D(Q_j)}{R(\widehat{Q}_j)-R(Q_j)}$, then the multiresolution codebook Q is pruned to $\widehat{Q}$ with the same slope $\lambda(Q) = \lambda(Q_j)$:

$$
\begin{aligned}
R(\widehat{Q}) &= \left[\sum_{i=1}^{M} R(Q_i)\right] - R(Q_j) + R(\widehat{Q}_j), \\
D(\widehat{Q}) &= \left[\sum_{i=1}^{M} D(Q_i)\right] - D(Q_j) + D(\widehat{Q}_j), \\
\lambda(Q) &= -\frac{D(\widehat{Q}) - D(Q)}{R(\widehat{Q}) - R(Q)} = -\frac{D(\widehat{Q}_j) - D(Q_j)}{R(\widehat{Q}_j) - R(Q_j)} = \lambda(Q_j).
\end{aligned}
$$

Hence the previous slope of the multiresolution codebook is given by the slope of the codebook just pruned, and the next slope of a multiresolution codebook is given by the slope of the next codebook to be pruned. Therefore, if the subband quantizers are on the lower convex hull of their rate-distortion curves, and pruning is always done by choosing the smallest slope quantizer, the multiresolution codebooks will also lie on the lower convex hull of their rate-distortion curve.

A pictorial example of GBFOS bit allocation is shown in Figure 4.3. At each iteration of the GBFOS algorithm, the space to be examined is limited to the rectangular region between the rate 0 codebook and the entire multiresolution codebook, Figure 4.3(a). One iteration of GBFOS bit allocation involves finding the least slope

Table 4.1: GBFOS bit allocation.

Step 0.	Assign $R \leftarrow \sum_{i=1}^{M} R(Q_i)$ and $D \leftarrow \sum_{i=1}^{M} D(Q_i)$.
Step 1.	Find among all subband codebooks the quantizer with the least slope, $Q_i = \arg\min_{Q_j, j=1,\ldots,M} \lambda(Q_j)$. If $\lambda(Q_i) = \infty$, quit. All of the TSVQ's have been pruned to their respective root nodes.
Step 2.	Prune the minimum slope codebook Q_i to the next GBFOS optimal codebook $\widehat{Q}_i$ and update the TSVQ. Assign $R \leftarrow R - R(Q_i) + R(\widehat{Q}_i)$, $D \leftarrow D - D(Q_i) + D(\widehat{Q}_i)$, $Q_i \leftarrow \widehat{Q}_i$.
Step 3.	If $R > R_{\text{target}}$, go to **Step 1.** Otherwise, quit.

from the current multiresolution codebook to all other possible pruned codebooks, Figure 4.3(b). This represents the GBFOS pruning of one subband quantizer. For the next iteration of the GBFOS algorithm a reduced area of the space will be examined, Figure 4.3(c). The end results are the multiresolution codebooks that lie on the lower convex hull of the rate-distortion curve, Figure 4.3(d).

4.2.2 Bit Allocation Using ROPA for Wavelets and PTSVQ

GBFOS bit allocation cannot always achieve the desired rate since optimal points may be scarce. As a result, there may not be many quantizers for the bit allocation algorithm to choose from. *Time-sharing* between a high rate and a low rate multiresolution codebook can be done to yield the desired rate. Although time-sharing yields a codebook that is on the convex hull of the rate-distortion curve, again, it has the drawback of introducing nonstationary distortion throughout the compressed data and interferes with the distribution of codeword indices. ROPA [61], described in Section 2.5.2, was introduced to overcome these drawbacks. In ROPA pruning, the tree is pruned by removing only M leaf nodes at a time. All of the optimal GB-

FOS subtrees are identified, and all of the other subtrees which are found minimize the area between the rate-distortion curve and the convex hull of the rate-distortion curve. This algorithm guarantees many more codebooks over the range of bit rates.

The problem of low-rate bit allocation among wavelet subbands is addressed by extending ROPA to simultaneously and conveniently allocate bits among the wavelet subbands. The algorithm is applied to the TSVQ's used to code wavelet subbands. In doing bit allocation with ROPA, not only are all of the GBFOS quantizers found, but every non-GBFOS quantizer which is found minimizes the area between the rate-distortion curve and the convex hull of the rate-distortion curve [61]. This algorithm guarantees many more good multiresolution codebooks over the range of desired bit rates. Comparing this method to GBFOS bit allocation, ROPA produces many additional bit allocations that lie close to the rate-distortion curve.

ROPA bit allocation for finding the highest rate to the lowest rate quantizers (until the desired rate is achieved) works as follows. High-rate TSVQ's are designed for each subband. The subband TSVQ Q_i with the least slope $\lambda_{min} = \lambda(Q_i)$ is found, and its best GBFOS node identified. The branch headed by this GBFOS node is recursively examined until the ROPA node with the least λ value in it is found, see Section 2.5.2 for ROPA details. This ROPA node is pruned and the TSVQ is updated. If the newly pruned TSVQ combined with the other subband quantizers yields the target rate, R_{target}, pruning stops. Otherwise the GBFOS pruned branch is reexamined. If the GBFOS branch is completely pruned (in which case the minimum slope value of the tree exceeds the minimum slope value first found, $\lambda_{min}(Q_i) > \lambda_{min}$, the next GBFOS node among all TSVQ's is found, and the process repeats. The complete ROPA bit allocation algorithm is given in Table 4.2.

For the multiresolution codebook, all but one of the source quantizers chosen is on its local convex hull of the rate-distortion curve. When ROPA prunes the multiresolution codebook, it traverses the rate-distortion curve from one GBFOS point to another GBFOS point via non-optimal ROPA points, for only one source at a

Table 4.2: ROPA bit allocation.

Step 0.	Assign $R \leftarrow \sum_{i=1}^{M} R(Q_i)$ and $D \leftarrow \sum_{i=1}^{M} D(Q_i)$.
Step 1.	Find among all subbands codebooks the quantizer with the least GBFOS slope $Q_i = \arg\min_{Q_j, j=1,\ldots,M} \lambda(Q_j)$. Assign $\lambda_{min} \leftarrow \lambda_{min}(Q_i)$. If $\lambda_{min} = \infty$, quit. All of the TSVQ's have been pruned to their respective root nodes.
Step 2.	If $\lambda_{min}(Q_i) \leq \lambda_{min}$: Prune the minimum slope TSVQ Q_i using ROPA to get the highest rate ROPA codebook $\widehat{Q}_i$. Assign $R \leftarrow R - R(Q_i) + R(\widehat{Q}_i)$, $D \leftarrow D - D(Q_i) + D(\widehat{Q}_i)$, $Q_i \leftarrow \widehat{Q}_i$. If $R > R_{\text{target}}$, go to **Step 2.**
Step 3.	If $R > R_{\text{target}}$, go to **Step 1.** Otherwise, quit.

time. The algorithm does not switch to another source until the next GBFOS point is reached. Thus bit allocation by ROPA will contain the set of optimal GBFOS bit allocations. As described before, the additional multiresolution codebooks found by ROPA will lie close to the rate-distortion curve since only one of the source codebooks at a time is ever suboptimal. Every suboptimal multiresolution codebook will minimize the area between the rate-distortion curve and the convex hull of the rate-distortion curve.

Bit allocation with ROPA has the same subtlety as ROPA for pruning when there are multiple minimum slopes. In this case, each pruned subset must be examined. However, the order of examination does not matter since pruning one section of a codebook does not affect the pruning of another codebook and the area under the rate-distortion curve will still be minimal with respect to the convex hull. Therefore, the bit allocation solutions found by ROPA are not unique. For every n identical minimum slopes, there will be $n!$ possible solutions.

A pictorial example of ROPA bit allocation is shown in Figure 4.4. After each pruning iteration of the GBFOS algorithm, Figure 4.3(c), a number of multiresolution codebooks are discarded as possible solutions for bit allocation since they are suboptimal. Some of these suboptimal codebooks lie in the square shaded regions bounded by the optimal codebooks, Figure 4.4(a), while others lie outside of this region. ROPA bit allocation works by recursively pruning the trees in these shaded regions, identifying the suboptimal codebooks which lie closest to the optimal rate-distortion curve.

4.2.3 Optimal Bit Allocation for Wavelet Packets and PTSVQ

Thus far, bit allocation methods for the discrete wavelet transform and PTSVQ have been presented. Because wavelet packets offer a much richer set of spaces for signal decomposition and the possibility of better data representation leads to the possibility of better data compression, I next examine bit allocation for the wavelet packet transform. For instance, the DWT is just one possible basis, or equivalently one possible subtree, of a WPT. It may be that a different subtree can offer a better representation in the rate-distortion sense. The number of possible subtrees of any WPT is given recursively where, for a d-dimensional WPT of depth D, the number of possible subtrees is $S(D) = [S(D-1)]^{2^d} + 1$ and $S(0) = 1$. Thus a two-dimensional WPT of depth three (as will be used in Chapter 5) has 83522 possible subtrees. The optimal bit allocation problem is much more complicated since it is necessary to choose the best basis representation among all subtrees of the WPT for the data while also assigning the best bit allocation.

In this section, I present a bit allocation algorithm using a pruning approach for wavelet packets which will systematically find *all* best basis WPT/PTSVQ encoders that lie on the lower convex hull of the rate-distortion curve without having to evaluate every possible subtree of the WPT. I do this by assigning high-rate TSVQ's to each wavelet packet subband of the wavelet packet tree, Figure 4.5. I then prune the

TSVQ with the least slope and identify the best basis WPT subtree using several recursive properties of the WPT and its structure to simplify the best basis search. This procedure is repeated until the desired rate is achieved. Note that the WPT is not being pruned in this algorithm: only the TSVQ's assigned to each node of the WPT are pruned. Each time a TSVQ is pruned, the subtree of the WPT that represents the best basis is identified by examining only the subband node whose TSVQ was just pruned, and that node's ancestors in the WPT. The WPT subtrees are not nested as a function of rate since the WPT is not being pruned.

My approach is similar to the single-tree method [79], described in Section 4.1.4, which was the first algorithm proposed to address the problem of joint bit allocation and best basis selection, especially as the Lagrangian minimization technique is used for both algorithms. The main difference is that my scheme will systematically find *all* vertices on the lower convex hull of the rate-distortion curve. The single-tree method performs a search for a given rate by heuristically selecting a slope and finding the WPT quantizer assignments whose joint vertex lies on the lower convex hull intersecting the slope. The search continues until a WPT codebook with a reasonably close rate is found. It is possible that the single-tree method may not find the WPT codebook closest to the desired rate due to the heuristic search. Bit allocation by pruning ensures that the WPT codebook closest to the desired rate will be found. Furthermore, it also ensures that all WPT codebooks on the lower convex hull of the rate-distortion curve are identified starting from the highest rate possible to the rate 0 codebook. By having the optimal rate-distortion characteristics for all bit rates, a precise description of the WPT codebook by its rate-distortion characteristics alone can be used in the design of more complex encoding schemes.

Optimal bit allocation and best basis assignment can be done simultaneously while pruning TSVQ's since the wavelet packet decomposition is an orthonormal transform, and each wavelet packet space is the direct sum of the subspaces created by the decomposition. For a d-dimensional wavelet packet transform, the M-ary

wavelet packet tree T is structured so that interior nodes t_i have children nodes t_{Mi+j}, $j = 1, \ldots, M$, where $M = 2^d$, and the root node is t_0. The WPT can be quantized by a WPT codebook Q which represents a hybrid WPT/PTSVQ scheme where each wavelet packet at node t_i has an assigned PTSVQ Q_i. The WPT codebook Q is characterized by rate $R(Q)$ and distortion $D(Q)$, which, as shown previously, are the sum of the rate and distortion of the quantized subbands (which are assumed to be normalized by vector dimension as well as the fraction of the signal space) of a given wavelet packet decomposition. Since the original signal space can be represented with perfect reconstruction by the leaves of any subtree S of the wavelet packet tree T, the rate and distortion of the WPT codebook Q can be written as:

$$R(Q) = \sum_{t_i \in \tilde{S}} R(Q_i) \quad \text{and} \quad D(Q) = \sum_{t_i \in \tilde{S}} D(Q_i)$$

where $\tilde{S}$ refers to the leaf nodes of subtree S.

Using the Lagrangian minimization technique, the quantization cost is written as the average distortion plus a penalty for the rate of the quantizer:

$$J(Q) = J(D(Q), R(Q)) = \sum_{t_i \in \tilde{S}} D(Q_i) + \lambda \sum_{t_i \in \tilde{S}} R(Q_i).$$

The problem is to minimize the cost $J(Q)$ for a given subtree, and to find the subtree S^{opt} which minimizes the cost for all subtrees.

$$S^{\text{opt}} = \arg\min_{S \subset T} J(Q) = \arg\min_{S \subset T} \left[\sum_{t_i \in \tilde{S}} D(Q_i) + \lambda \sum_{t_i \in \tilde{S}} R(Q_i) \right].$$

Optimal bit allocation for WPT's necessarily determines the best basis since the optimal bit allocation is found by examining all possible basis representations. Again, there are two problems here: one is to find the best bit allocation knowing the basis; the other is to find the best basis for all bit allocations.

Assuming that the basis is known, I first address the bit allocation problem. By differentiating $J(D(Q), R(Q))$ with respect to the rate allocated to each subband

quantizer Q_j and setting the results to zero,

$$\frac{\partial J(D(Q),R(Q))}{\partial R(Q_j)} = \sum_{t_i \in \tilde{S}} \left[\frac{\partial D(Q_i)}{\partial R(Q_j)} + \lambda \frac{\partial R(Q_i)}{\partial R(Q_j)} \right] = 0,$$

the Lagrangian functional for any subtree $S \subset T$ is minimized when $\lambda = -\frac{\partial D(Q_i)}{\partial R(Q_i)}$.

As seen previously, λ is the same for all subbands regardless of scale or orientation. Again, optimal bit allocation is achieved when the quantizers of each subband have the same slope on the lower convex hull of their local rate-distortion curve. By enforcing the condition that all quantizers of the WPT have the same slope, the subtree that yields the minimum cost may be found.

I next discuss the search for the best basis subtree S^{opt} knowing that the bit allocation problem is solved if the vertex of all subband quantizers on their local rate-distortion curves intersect a common slope λ. Again, bit allocation is done by assigning high-rate TSVQ's (or other VQ's with known rate-distortion characteristics) to each wavelet packet subband and pruning the TSVQ with the least slope until the desired rate is achieved. Assuming that each subband quantizer lies on a vertex of the lower convex hull of its rate-distortion curve, each quantizer can intersect a slope which lies in the interval of the two slopes which form the vertex. Therefore, by always pruning the TSVQ with the least slope: that least slope value will intersect all subband quantizers; the slope that will be pruned next will also intersect all subband quantizers; and any slope value in between will intersect all subband quantizers too. The endpoints of the interval of slope values that intersect all subband quantizers will become an important test case to determine when there will be a switch between best bases. This will be discussed later in more detail.

As discussed previously, to find the best basis subtree each time a subband quantizer is pruned, the cost function $J(D(Q),R(Q))$ must be reevaluated for *every* possible subtree of the WPT for the given slope λ. This is computationally costly and inefficient. Instead, I seek a fast and efficient computational solution that exploits the structure and properties of the wavelet packet decomposition. Ideally, the rate-

distortion characteristics of the optimal WPT quantizer with the best basis determination and bit allocation could be obtained from one hybrid, or *best basis* quantizer which characterizes the entire WPT tree. Towards this end, I use two simplifications which will make the optimal characteristics of the entire WPT/PTSVQ tree available from only the root node of the WPT, while preserving the best basis decisions in the WPT structure itself.

To facilitate the following discussion and algorithm development, I now introduce the complete data structure for an M-ary WPT T. Each node t_i of the WPT has an associated TSVQ quantizer Q_i which consists of a number of nested GBFOS optimal PTSVQ's ordered from highest to lowest rate by associated rate $R(Q_i)$, distortion $D(Q_i)$, and slope $\lambda(Q_i)$. Furthermore, as each node will ultimately represent a best basis quantizer Q_i^{BB} for the branch of the WPT that it heads, the following information is also stored: the next minimum slope of the best basis quantizer $\lambda(Q_i^{\mathrm{BB}})$ (creating a path from the root to the minimum slope quantizer), the best basis decision B_i (where $B_i = 0$ if it is better to use the current node, or $B_i = 1$ if it is better to use the children nodes), and the rate and distortion associated with the best basis decision, $R(Q_i^{\mathrm{BB}})$ and $D(Q_i^{\mathrm{BB}})$, (permitting each parent node to have direct access to the optimal rate-distortion characteristics of all of its descendents from its children only, without requiring the specific details of which basis was selected). Again, the nodes of the WPT are labeled such that each interior node t_i has children nodes $t_{Mi+j}, j = 1, \ldots, M$, and the root node is t_0. The parent nodes of t_i are referred to as parent(t_i).

The first simplification for creating a single best basis quantizer uses two properties of WPT's (orthonormality and direct sum of spaces) to replace the quantizers of the children of any internal node t_i of the WPT by a *composite* quantizer Q_i^{comp}. The second simplification replaces the internal node's quantizer Q_i with a *merged* best basis quantizer Q_i^{BB} (created from Q_i and the composite quantizer Q_i^{comp}) which represents the best basis quantizer for node t_i. The use of these simplifications is

shown in Figure 4.6 where leaf node quantizers are grouped into composite quantizers, and composite quantizers are merged with their parent node's quantizer. This process continues until there is only one node left, that is the root node t_0 with merged quantizer Q_0^{BB}. This node contains the optimal rate-distortion curve, and hence bit allocations, for the entire WPT for all rates. To create these merged and composite WPT quantizers, it is necessary that the quantizers used for each subband lie on the lower convex hull of their local rate-distortion curve. This implies that the composite and merged quantizers Q_i^{comp} and Q_i^{BB} must also lie on the lower convex hull of their local rate-distortion curves.

If the component nodes' rate-distortion curves are convex and pruning is done by choosing the smallest slope quantizer, then the rate-distortion curve of the composite node is convex. Let Q_i^{comp} be a composite quantizer created from M merged quantizers $Q_{Mi+j}^{\mathrm{BB}}, j = 1, \ldots, M$. Q_i^{comp} is characterized by rate $R(Q_i^{\mathrm{comp}})$ and distortion $D(Q_i^{\mathrm{comp}})$:

$$R(Q_i^{\mathrm{comp}}) = \sum_{j=1}^{M} R(Q_{Mi+j}^{\mathrm{BB}}), \quad D(Q_i^{\mathrm{comp}}) = \sum_{j=1}^{M} D(Q_{Mi+j}^{\mathrm{BB}}),$$

where $R(Q_{Mi+j}^{\mathrm{BB}})$ and $D(Q_{Mi+j}^{\mathrm{BB}})$ are the rate and distortion of the M component quantizers. If one quantizer Q_{Mi+m}^{BB} is pruned to $\widehat{Q}_{Mi+m}^{\mathrm{BB}}$, and the slope is

$$\lambda(Q_{Mi+m}^{\mathrm{BB}}) = -\frac{D(\widehat{Q}_{Mi+m}^{\mathrm{BB}}) - D(Q_{Mi+m}^{\mathrm{BB}})}{R(\widehat{Q}_{Mi+m}^{\mathrm{BB}}) - R(Q_{Mi+m}^{\mathrm{BB}})},$$

then Q_i^{comp} is pruned to $\widehat{Q}_i^{\mathrm{comp}}$ with slope $\lambda(Q_i^{\mathrm{comp}}) = \lambda(Q_{Mi+m}^{\mathrm{BB}})$:

$$\begin{aligned}
R(\widehat{Q}_i^{\mathrm{comp}}) &= R(Q_i^{\mathrm{comp}}) - R(Q_{Mi+m}^{\mathrm{BB}}) + R(\widehat{Q}_{Mi+m}^{\mathrm{BB}}) \\
D(\widehat{Q}_i^{\mathrm{comp}}) &= D(Q_i^{\mathrm{comp}}) - D(Q_{Mi+m}^{\mathrm{BB}}) + D(\widehat{Q}_{Mi+m}^{\mathrm{BB}}) \\
\lambda(Q_i^{\mathrm{comp}}) &= -\frac{D(\widehat{Q}_i^{\mathrm{comp}}) - D(Q_i^{\mathrm{comp}})}{R(\widehat{Q}_i^{\mathrm{comp}}) - R(Q_i^{\mathrm{comp}})} \\
&= -\frac{D(\widehat{Q}_{Mi+m}^{\mathrm{BB}}) - D(Q_{Mi+m}^{\mathrm{BB}})}{R(\widehat{Q}_{Mi+m}^{\mathrm{BB}}) - R(Q_{Mi+m}^{\mathrm{BB}})} \quad = \quad \lambda(Q_{Mi+m}^{\mathrm{BB}}).
\end{aligned}$$

Hence the previous slope of a composite quantizer is given by the slope of the quantizer just pruned, and the next slope of a composite quantizer is given by the slope of the next quantizer to be pruned. Therefore, if the component quantizers are on the lower convex hull of their rate-distortion curves, and pruning is always done by choosing the smallest slope quantizer, then the composite node's quantizers will also lie the lower convex hull of their rate-distortion curve.

I next examine the creation of a merged quantizer Q_i^{BB} using an interior WPT node t_i with associated quantizer Q_i with rate $R(Q_i)$, distortion $D(Q_i)$, and slope $\lambda(Q_i)$ known, and composite quantizer Q_i^{comp} with rate $R(Q_i^{\text{comp}}) = \sum_{j=1}^{M} R(Q_{Mi+j}^{\text{BB}})$, distortion $D(Q_i^{\text{comp}}) = \sum_{j=1}^{M} D(Q_{Mi+j}^{\text{BB}})$, and slope

$$\lambda(Q_i^{\text{comp}}) = \min(\lambda(Q_{Mi+1}^{\text{BB}}), \ldots, \lambda(Q_{Mi+M}^{\text{BB}}))$$

known as well. Assuming that the quantizers Q_i and Q_i^{comp} are located on the lower convex hull of their rate-distortion curves, Figure 4.7(a), and that pruning is done by choosing the least-slope quantizer, the following merging procedure will create a "merged" rate-distortion curve where all quantizers Q_i^{BB} lie on the lower convex hull, Figure 4.7(b).

Again, both quantizers, Q_i and Q_i^{comp}, are located on vertices of their respective rate-distortion curves and are the results of pruning using the least slope. Assuming that Q_i^{BB} is also located on a vertex on the lower convex hull, λ_{min} is the minimum slope previously pruned (zero if evaluating the highest rate subband quantizers), and $\lambda(Q_i^{\text{BB}}) = \min(\lambda(Q_i), \lambda(Q_i^{\text{comp}}))$ is the next minimum slope to be pruned. That is, λ_{min} is the maximum slope of the slopes that are to the right of the current vertices, and $\lambda(Q_i^{\text{BB}})$ is the minimum slope of the slopes that are to the left of the vertices, Figure 4.8. It follows that both Q_i and Q_i^{comp} intersect the line created by the cost function for a slope $\lambda \in [\lambda_{min}, \lambda(Q_i^{\text{BB}})]$. Following [24, 79], the quantizer that incurs the least cost is the quantizer that should be used for encoding where the costs are computed as $J_\lambda(Q_i) = D(Q_i) + \lambda R(Q_i)$ and $J_\lambda(Q_i^{\text{comp}}) = \sum_{j=1}^{M} D(Q_{Mi+j}^{\text{BB}}) + \lambda R(Q_{Mi+j}^{\text{BB}})$

for any $\lambda \in [\lambda_{min}, \lambda(Q_i^{BB})]$.

It is possible that both quantizers lie on the lower convex hull of the merged rate-distortion curve, and that a single cost test may not identify both quantizers when they are associated with a different basis. To test for this, it is sufficient to use the two endpoints of the slope interval to evaluate costs. By first using $\lambda = \lambda_{min}$, the higher-rate quantizer is tested. By next using $\lambda = \lambda(Q_i^{BB})$, the lower-rate quantizer is tested. This order ensures that higher rate quantizers are identified first. By evaluating the costs twice, all vertices on the lower convex hull are detected.

The different cases that arise when merging are illustrated in Figures 4.9 and 4.10. An example of the need for two cost evaluations is shown in Figure 4.9(c) where two nodes are first being merged (i. e. during initialization). In this example, both Q_i and Q_i^{comp} lie on the lower convex hull of the merged rate-distortion curve. Bit allocation should begin with the assignment of the lowest distortion quantizer, Q_i^{comp}. If only the next minimum slope $\lambda(Q_i^{BB})$ were used to measure costs, Q_i^{comp} would not be identified as being located on the lower convex hull, which is incorrect. If only the previous minimum slope (which is zero when initializing) were used to measure costs, Q_i^{comp} would be identified. However, quantizer Q_i will be pruned before Q_i^{comp} since it has the next minimum slope. Since Q_i is not identified as being located on the lower convex hull when using only the previous minimum slope, Q_i would never be identified as being on the lower convex hull! Therefore, merging with only one slope to measure costs would misidentify one of the two quantizers on the lower convex hull. This situation can arise any time when switching from the current rate-distortion curve to the other, i. e. when switching from one best basis to another, see, for example, Figure 4.10(c).

Since two cost measurements can identify all quantizers that lie on the lower convex hull, I compute both costs each time a VQ is pruned. If both quantizers lie on the lower convex hull, I compute the slope of the "missing link," $\tilde{\lambda} = \frac{D(Q_i^{comp}) - D(Q_i)}{R(Q_i^{comp}) - R(Q_i)}$, between the two quantizers, and use it as the next local minimum slope for pruning

$\lambda(Q_i^{BB}) \leftarrow \tilde{\lambda}$. No actual pruning will be done since $\tilde{\lambda}$ is less than $\lambda(Q_i^{comp})$ and $\lambda(Q_i)$, but there will be a switch between curves (i. e. best bases). If the smallest slope quantizer is always pruned, and two cost measurements are used where the missing link slope is computed when necessary, the rate, distortion, and next slope of the quantizers of the merged node will be identified, and all quantizers will lie on the lower convex hull of the merged node's rate-distortion curve. This is desirable since the merged node may then be used as a component in a composite node.

The previous method is not the only way to identify all optimal quantizers. An alternative approach is to compute the missing link slope $\tilde{\lambda}$ every time a quantizer is pruned where $\tilde{\lambda} = \infty$ if the lower cost quantizer located on slope λ_{min} has a higher rate than the higher cost quantizer and $\tilde{\lambda} = \frac{D(Q_i^{comp}) - D(Q_i)}{R(Q_i^{comp}) - R(Q_i)}$ if the lower cost quantizer has a lower rate than the higher cost quantizer. Then, if $\tilde{\lambda} \in [\lambda_{min}, \lambda(Q_i^{BB})]$, both quantizers lie on the lower convex hull. If there is to be a switch from one rate-distortion curve to the other, $\tilde{\lambda}$ should be used as the next minimum slope for pruning, $\lambda(Q_i^{BB}) \leftarrow \tilde{\lambda}$. The slope computation requires two subtractions and one division, while the second cost computation requires one additional multiplication and addition. Since curve switching is relatively infrequent, I use the two-cost method. If the input to the WPT bit allocation program is a list of rate-distortion characteristics, the two-cost method may also be more computationally stable since a division is not required for the test case.

The WPT bit allocation algorithm that I introduce does not require that the previous simplifications be carried out in the manner just described creating a single node as shown in Figure 4.6 as this would require more work than is necessary. The physical merging of nodes is avoided by using the given WPT data structure permitting each WPT node to retain only the current optimal rate-distortion characteristics of the merged, or best basis quantizer that it represents. This information is directly available to each parent node so that its composite quantizer can be easily formed.

The WPT structure field B_i is used to specify whether the current or composite

node's quantizer should be used, indicating the optimal best basis structure. This decision is made once the costs for merging an interior node with its composite node have been computed. The best basis decision is made so that the lowest cost quantizer is used. Once an interior node has been modified, it is necessary to update the rate, distortion, and best basis decisions of its ancestor nodes' merged quantizer using the same decision criteria since any interior node is itself part of a composite or merged quantizer for all of its ancestors. Other nodes throughout the tree do not need to be reevaluated since their status stays the same as their next minimum slopes are larger than the previous minimum slope. The initialization and updating of internal nodes are illustrated in Figures 4.9 and 4.10.

My algorithm first initializes a WPT for the highest-rate quantizers Q_i, deciding which merged quantizers Q_i^{BB} lie on the lower convex hull of the rate-distortion curve. Then, until the desired rate is achieved, the lowest slope quantizer is pruned, and the rate-distortion values of only the composite and merged nodes containing the pruned node are recomputed while simultaneously selecting the optimal quantizer for the given slope. In this manner, each slope is examined only once, and only the portion of the WPT containing the pruned quantizer is examined. Furthermore, since this pruning method always finds the lower convex hull of the rate-distortion curve, the overall reported rate of the WPT is monotonically decreasing with increasing λ, and the overall rate-distortion curve contains all quantizers on the vertices of the lower convex hull.

Initialization is outlined in Table 4.3. A path through the WPT from the root node to the node with the minimum slope quantizer is created by computing $\lambda(Q_i^{\mathrm{BB}})$ as $\min(\lambda(Q_i), \lambda(Q_{Mi+1}^{\mathrm{BB}}), \ldots, \lambda(Q_{Mi+M}^{\mathrm{BB}}))$ for all nodes. When initializing internal nodes, it is assumed that the previous minimum slope of the rate-distortion curve, λ_{min}, is zero. The next minimum slope value for each internal node of the WPT is $\lambda(Q_i^{\mathrm{BB}})$. If it is found for λ_{min} that the cost of encoding with the node's quantizer Q_i is the same as the cost of encoding with the composite quantizer Q_i^{comp}, then the lower

rate quantizer is selected. Otherwise, the lower cost quantizer is chosen for encoding. Using the next minimum slope value, $\lambda(Q_i^{\mathrm{BB}})$, a test is done to determine if there will be a switch between rate-distortion curves, in which case the missing $\tilde{\lambda}$ is computed and stored as $\lambda(Q_i^{\mathrm{BB}})$. The best basis decision B_i reflects the selection of the lowest cost quantizer. The rate and distortion of the lowest cost (optimal) quantizer are stored in the given node as $R(Q_i^{\mathrm{BB}})$ and $D(Q_i^{\mathrm{BB}})$.

The main WPT bit allocation algorithm is described in Table 4.4, and its two subroutine calls are described in Tables 4.5 and 4.6. Bit allocation and best basis assignment continue until the desired rate, R_{target}, has been reached. The **min_slope_node** subroutine, outlined in Table 4.5, finds the WPT node t_i with the minimum slope quantizer by following the path indicated by the minimum value $\lambda_{min} = \lambda(Q_0^{\mathrm{BB}})$. This subroutine must check for the possibility that a missing link $\tilde{\lambda}$ is stored as $\lambda(Q_i^{\mathrm{BB}})$ in which case $\lambda(Q_i^{\mathrm{BB}})$ is less than $\lambda(Q_i)$ and $\lambda(Q_{Mi+j}^{\mathrm{BB}}), j = 1, \ldots, M$. Once the minimum slope node t_i is found, its quantizer Q_i is pruned if it is determined that the slope is not a missing link slope. Then it is determined whether or not it is better to encode using the given node's *new* pruned quantizer, or if it is better to encode using the nodes' children's composite quantizer. The ancestor nodes are updated using this new information since each merged or composite quantizer containing the pruned quantizer can be considered to have been pruned. The best basis assignment is determined by the best basis decisions B_i. The subtree structure is indicated the B_i values where a node is a subtree leaf node if $B_i = 0$ and all ancestors of the node have values of 1.

The **update** subroutine, described in Table 4.6, is very similar to the second step of the initialization procedure, except that the initialization procedure is performed on *all* interior nodes whereas updating needs to be performed only on the minimum slope node and its ancestors. When updating internal nodes, the previous minimum slope of the rate-distortion curve is $\lambda_{min} = \lambda(Q_0^{\mathrm{BB}})$. The next minimum slope value for each internal node t_i of the WPT is $\lambda(Q_i^{\mathrm{BB}})$. If it is found that the cost of encoding with

Table 4.3: Wavelet packet bit allocation initialization.

Step 1. For each leaf node t_i:
$\lambda(Q_i^{\text{BB}}) \leftarrow \lambda(Q_i)$,
$B_i \leftarrow 0$,
$R(Q_i^{\text{BB}}) \leftarrow R(Q_i)$,
$D(Q_i^{\text{BB}}) \leftarrow D(Q_i)$.

Step 2. For each interior node t_i,

(a) Find the previous and next minimum slopes:
$\lambda_{min} \leftarrow 0$,
$\lambda(Q_i^{\text{BB}}) \leftarrow \min(\lambda(Q_i), \lambda(Q_{Mi+1}^{\text{BB}}), \ldots, \lambda(Q_{Mi+M}^{\text{BB}}))$.

(b) "Merge" the current node with its children's "composite":
Define $R(Q_i^{\text{comp}}) \leftarrow \sum_{j=1}^{M} R(Q_{Mi+j}^{\text{BB}})$,
$D(Q_i^{\text{comp}}) \leftarrow \sum_{j=1}^{M} D(Q_{Mi+j}^{\text{BB}})$,
$J_\lambda(Q_i^{\text{comp}}) \leftarrow D(Q_i^{\text{comp}}) + \lambda R(Q_i^{\text{comp}})$,
$J_\lambda(Q_i) \leftarrow D(Q_i) + \lambda R(Q_i)$.
If $J_{\lambda_{min}}(Q_i^{\text{comp}}) = J_{\lambda_{min}}(Q_i)$, use the lower rate node:
If $R(Q_i^{\text{comp}}) < R(Q_i)$, $B_i \leftarrow 1$.
Otherwise, $B_i \leftarrow 0$.
If $J_{\lambda_{min}}(Q_i^{\text{comp}}) < J_{\lambda_{min}}(Q_i)$, use the composite, $B_i \leftarrow 1$.
If $J_{\lambda(Q_i^{\text{BB}})}(Q_i^{\text{comp}}) > J_{\lambda(Q_i^{\text{BB}})}(Q_i)$, prepare to switch curves:
$\lambda(Q_i^{\text{BB}}) \leftarrow -\frac{D(Q_i)-D(Q_i^{\text{comp}})}{R(Q_i)-R(Q_i^{\text{comp}})}$.
If $J_{\lambda_{min}}(Q_i^{\text{comp}}) > J_{\lambda_{min}}(Q_i)$, use node t_i, $B_i \leftarrow 0$.
If $J_{\lambda(Q_i^{\text{BB}})}(Q_i^{\text{comp}}) < J_{\lambda(Q_i^{\text{BB}})}(Q_i)$, prepare to switch curves:
$\lambda(Q_i^{\text{BB}}) \leftarrow -\frac{D(Q_i)-D(Q_i^{\text{comp}})}{R(Q_i)-R(Q_i^{\text{comp}})}$.

(c) Update the rate and distortion of the "merged" node:
If $B_i = 1$, use the composite node,
$R(Q_i^{\text{BB}}) \leftarrow R(Q_i^{\text{comp}})$ and $D(Q_i^{\text{BB}}) \leftarrow D(Q_i^{\text{comp}})$.
If $B_i = 0$, use the current node t_i,
$R(Q_i^{\text{BB}}) \leftarrow R(Q_i)$ and $D(Q_i^{\text{BB}}) \leftarrow D(Q_i)$.

Table 4.4: Wavelet packet bit allocation.

Step 1.	If $\lambda(Q_0^{\text{BB}}) = \infty$, quit. All of the TSVQ's have been pruned to their respective root node.
Step 2.	Find the node with the lowest slope λ_{min}. $\lambda_{min} \leftarrow \lambda(Q_0^{\text{BB}})$ and $t_i \leftarrow$ **min_slope_node**(t_0) (see Table 4.5).
Step 3.	If $\lambda_{min} = \lambda(Q_i)$, prune the quantizer Q_i associated with node t_i. Otherwise there will be a switch from one curve to another, and pruning isn't necessary.
Step 4.	Update node t_i and its ancestors, **update**(t_i, λ_{min}) (see Table 4.6).
Step 5.	The optimal WPT codebook has rate $R(Q_0^{\text{BB}})$ and distortion $D(Q_0^{\text{BB}})$, and the corresponding best basis has been found. If $R(Q_0^{\text{BB}}) > R_{\text{target}}$ go to **Step 1.**

Table 4.5: Wavelet packet bit allocation subroutine **min_slope_node**(t_0).

Step 1.	Find the node with the lowest slope λ_{min} of a WPT with root t_0. $t_i \leftarrow t_0$ and $\lambda_{min} \leftarrow \lambda(Q_0^{\text{BB}})$.
Step 2.	If $\lambda(Q_i) = \lambda_{min}$, this is the node whose TSVQ should be pruned. Return node t_i.
Step 3.	Find t_{Mi+j} such that $\lambda(Q_{Mi+j}) = \lambda_{min}$. (a) If such a child does not exist, the slope links two different rate-distortion curves. Return node t_i. (b) Otherwise, $t_i \leftarrow t_{Mi+j}$. Go to **Step 2.**

the given node's quantizer $(J_{\lambda_{min}}(Q_i))$ is the same as the cost of encoding with the node's composite children's quantizer $(J_{\lambda_{min}}(Q_i^{\text{comp}}))$, then the lower rate quantizer is selected. Otherwise, the lower cost quantizer is selected. A check is done to determine if there will be a switch between rate-distortion curves, in which case the missing $\tilde{\lambda}$ is computed and stored as $\lambda(Q_i^{\text{BB}})$. The best basis decision B_i reflects the selection of the lowest cost quantizer. The rate and distortion of the lowest cost (optimal) quantizer are stored in the given node as $R(Q_i^{\text{BB}})$ and $D(Q_i^{\text{BB}})$ allowing each parent node to have direct access to the optimal rate-distortion characteristics of the best basis decomposition without requiring the specific details of which basis was selected.

Table 4.6: Wavelet packet bit allocation subroutine **update**(t_i, λ_{min}).

Step 1. Update node t_i and its ancestors using λ_{min} as the previous slope for all updated nodes.
If t_i is an interior node, go to **Step 2.**
If t_i is a leaf node:
$\lambda(Q_i^{\text{BB}}) \leftarrow \lambda(Q_i)$, $B_i \leftarrow 0$,
$R(Q_i^{\text{BB}}) \leftarrow R(Q_i)$, $D(Q_i^{\text{BB}}) \leftarrow D(Q_i)$.
If $t_i = t_0$, return.
If $t_i \neq t_0$, $t_i \leftarrow \text{parent}(t_i)$, go to **Step 2.**

Step 2. If t_i is an interior node:
(a) Find the next minimum slope:
$\lambda(Q_i^{\text{BB}}) \leftarrow \min(\lambda(Q_i), \lambda(Q_{Mi+1}^{\text{BB}}), \ldots, \lambda(Q_{Mi+M}^{\text{BB}}))$.
(b) "Merge" the current node with its children's "composite":
Define $R(Q_i^{\text{comp}}) \leftarrow \sum_{j=1}^{M} R(Q_{Mi+j}^{\text{BB}})$,
$D(Q_i^{\text{comp}}) \leftarrow \sum_{j=1}^{M} D(Q_{Mi+j}^{\text{BB}})$,
$J_\lambda(Q_i^{\text{comp}}) \leftarrow D(Q_i^{\text{comp}}) + \lambda R(Q_i^{\text{comp}})$,
$J_\lambda(Q_i) \leftarrow D(Q_i) + \lambda R(Q_i)$.
If $J_{\lambda_{min}}(Q_i^{\text{comp}}) = J_{\lambda_{min}}(Q_i)$, use the lower rate node.
If $R(Q_i^{\text{comp}}) < R(Q_i)$, $B_i \leftarrow 1$.
Otherwise, $B_i \leftarrow 0$.
If $J_{\lambda_{min}}(Q_i^{\text{comp}}) < J_{\lambda_{min}}(Q_i)$, use the composite, $B_i \leftarrow 1$.
If $J_{\lambda(Q_i^{\text{BB}})}(Q_i^{\text{comp}}) > J_{\lambda(Q_i^{\text{BB}})}(Q_i)$, prepare to switch curves:
$\lambda(Q_i^{\text{BB}}) \leftarrow -\frac{D(Q_i) - D(Q_i^{\text{comp}})}{R(Q_i) - R(Q_i^{\text{comp}})}$.
If $J_{\lambda_{min}}(Q_i^{\text{comp}}) > J_{\lambda_{min}}(Q_i)$, use node t_i, $B_i \leftarrow 0$.
If $J_{\lambda(Q_i^{\text{BB}})}(Q_i^{\text{comp}}) < J_{\lambda(Q_i^{\text{BB}})}(Q_i)$, prepare to switch curves:
$\lambda(Q_i^{\text{BB}}) \leftarrow -\frac{D(Q_i) - D(Q_i^{\text{comp}})}{R(Q_i) - R(Q_i^{\text{comp}})}$.
(c) Update the rate and distortion of the "merged" node:
If $B_i = 1$, use the composite node,
$R(Q_i^{\text{BB}}) \leftarrow R(Q_i^{\text{comp}})$ and $D(Q_i^{\text{BB}}) \leftarrow D(Q_i^{\text{comp}})$.
If $B_i = 0$, use the current node t_i,
$R(Q_i^{\text{BB}}) \leftarrow R(Q_i)$ and $D(Q_i^{\text{BB}}) \leftarrow D(Q_i)$.
(d) Update the ancestors of node t_i.
If $t_i = t_0$, return.
If $t_i \neq t_0$, $t_i \leftarrow \text{parent}(t_i)$, go to **Step 2.**

4.2.4 Implementation

I have implemented in C bit allocation code for wavelets using GBFOS and ROPA. I have also implemented in C bit allocation code for wavelet packets and PTSVQ. The implementation of all algorithms is such that it can be used in a more general setting for other types of VQ since the inputs to each file are a list of precalculated rate-distortion characteristics for each quantizer.

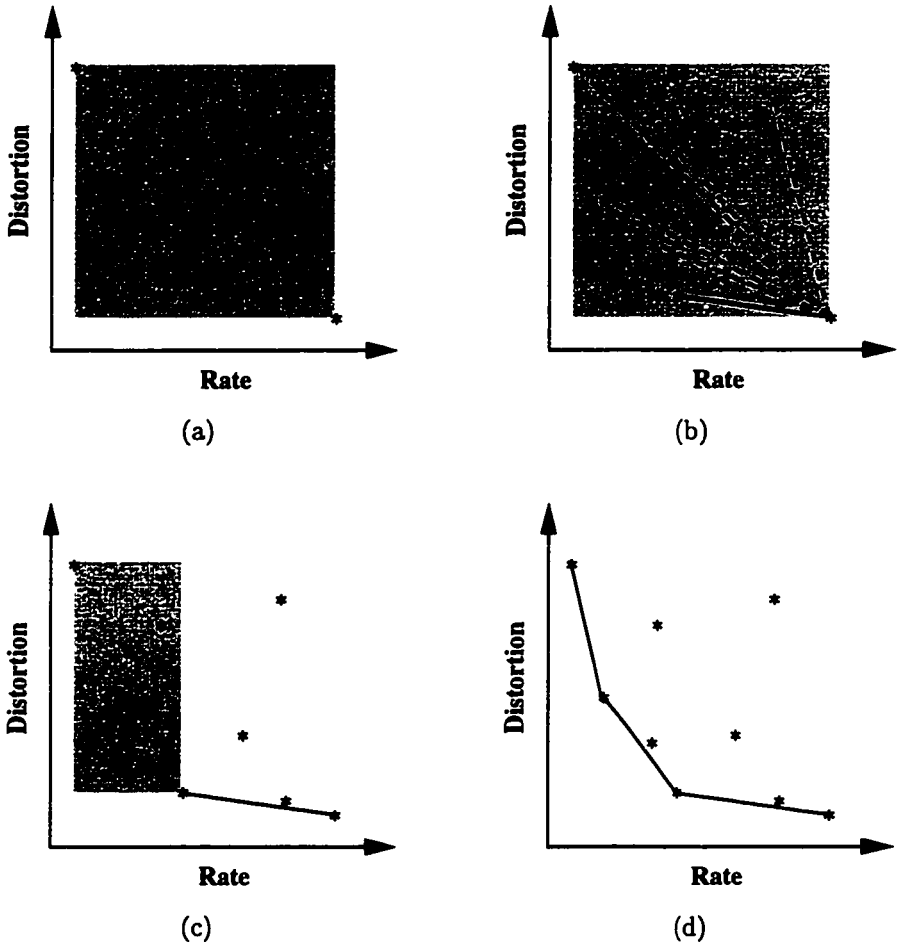


Figure 4.3: (a) At each iteration of the GBFOS algorithm, the space to be examined is limited to the shaded rectangular region between the rate 0 codebook and the entire multiresolution codebook. Each $*$ represents a possible pruned subtree where the $*$ in the upper left hand corner is the root node and the $*$ in the lower right hand corner is the large multiresolution codebook. (b) This demonstrates one iteration of the GBFOS bit allocation. The $*$ in the lower right hand corner of the shaded region is the current multiresolution codebook and the other $*$'s are possible pruned multiresolution codebooks. The next optimal multiresolution codebooks has the least slope as indicated by the solid line. (c) At the next iteration of the GBFOS algorithm a reduced area of the space must be examined which is limited to the shaded rectangular region between the root node and the optimal multiresolution codebook of the previous GBFOS iteration. (d) The end results of GBFOS bit allocation where the optimal multiresolution codebooks are linked by the solid line.

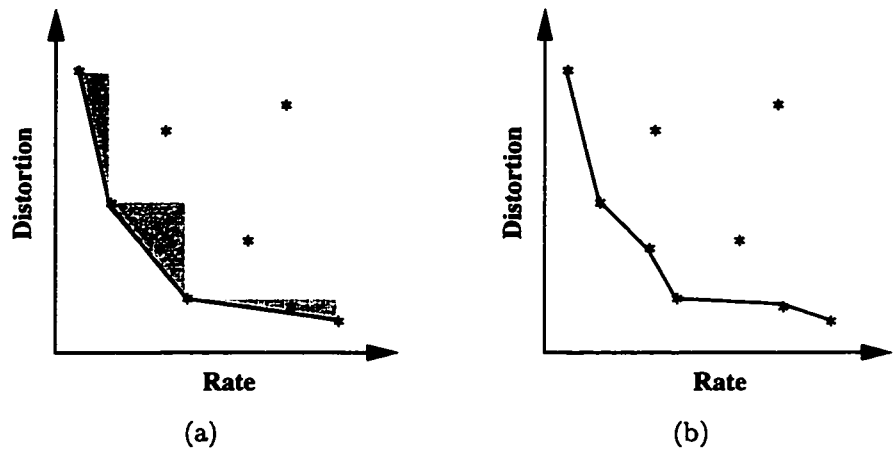


Figure 4.4: (a) After each pruning iteration of the GBFOS algorithm, Figure 4.3, a number of multiresolution codebooks are discarded as possible solutions for bit allocation since they are suboptimal. Some of these suboptimal codebooks lie in the square shaded regions bounded by the optimal codebooks, while others lie outside of this region. (b) ROPA works by recursively pruning the codebook tree in these shaded regions, identifying the suboptimal codebooks which lie closest to the optimal rate-distortion curve.

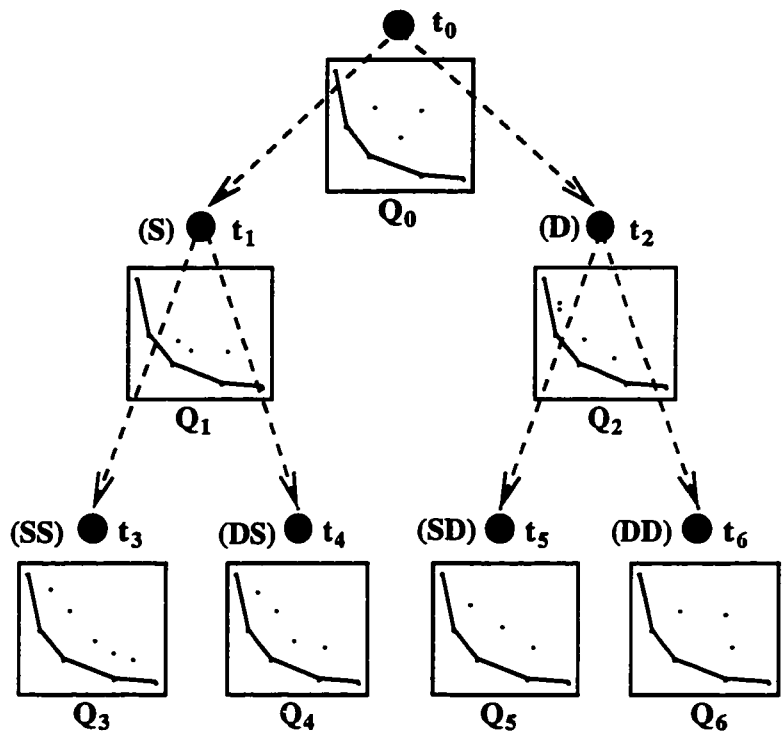
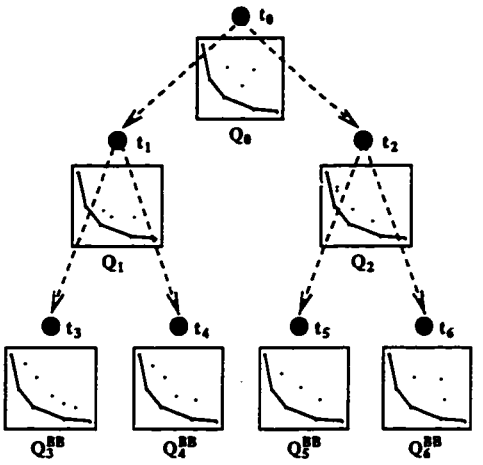
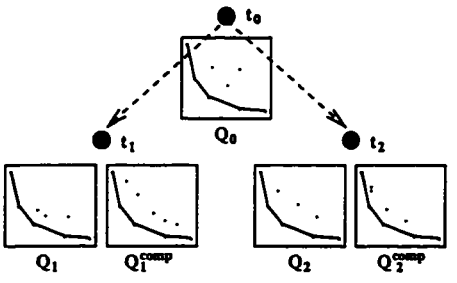


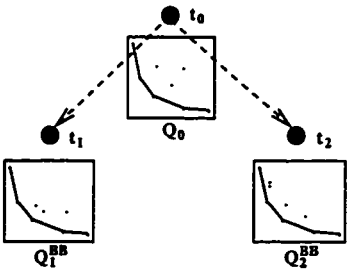
Figure 4.5: Wavelet packet bit allocation for a one-dimensional signal decomposed to two levels of wavelet packets. Each frequency subband of the wavelet packet tree represented by node t_i has an associated TSVQ Q_i. Bit allocation proceeds by finding each GBFOS optimal quantizer as the slope of the rate-distortion function increases and determining from that node up to the root node whether or not it is better to use the larger frequency subband or to use linear combinations of the smaller subbands. **S** and **D** refer to smooth and detail subbands.



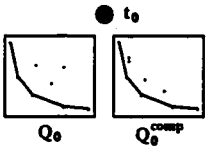
(a) The original WPT.



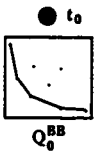
(b) Composite quantizers.



(c) Merged best basis nodes.



(d) Composite quantizer.



(e) Merged best basis node.

Figure 4.6: WPT bit allocation simplifications. (a) A WPT with root t_0. (b) Composite quantizers created from leaf node quantizers. (c) Best basis quantizers created by merging composite quantizers and the parent quantizer. (d) Composite quantizer created from leaf node quantizers. (e) A best basis quantizer created by merging the composite quantizer and the root node quantizer. Q_0^{BB} contains the optimal bit allocations for the entire WPT.

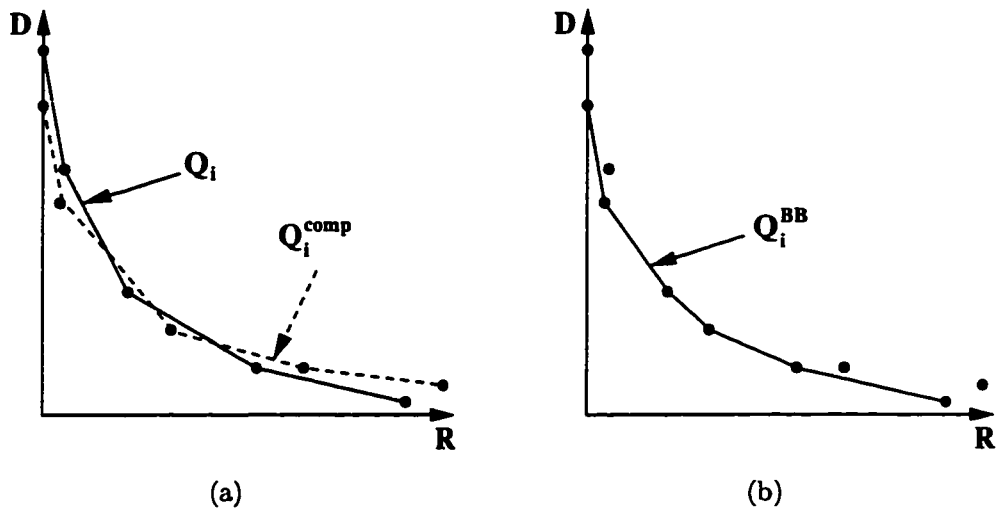


Figure 4.7: (a) The rate-distortion curves available to WPT node t_i before merging. (b) The rate-distortion curve of the node after merging the quantizers Q_i of node t_i with the quantizers of the composite node Q_i^{comp}.

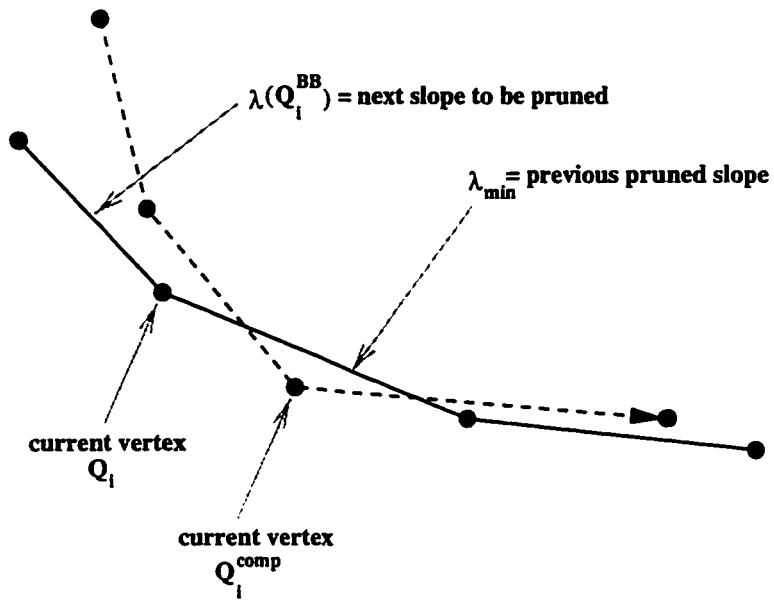


Figure 4.8: The previous and next slopes examined for WPT bit allocation where λ_{min} is the previous slope, or the maximum slope of the slopes that are to the right of the current vertices, and $\lambda(Q_i^{\text{BB}})$ is the next slope, or minimum slope of the slopes that are to the left of the vertices.

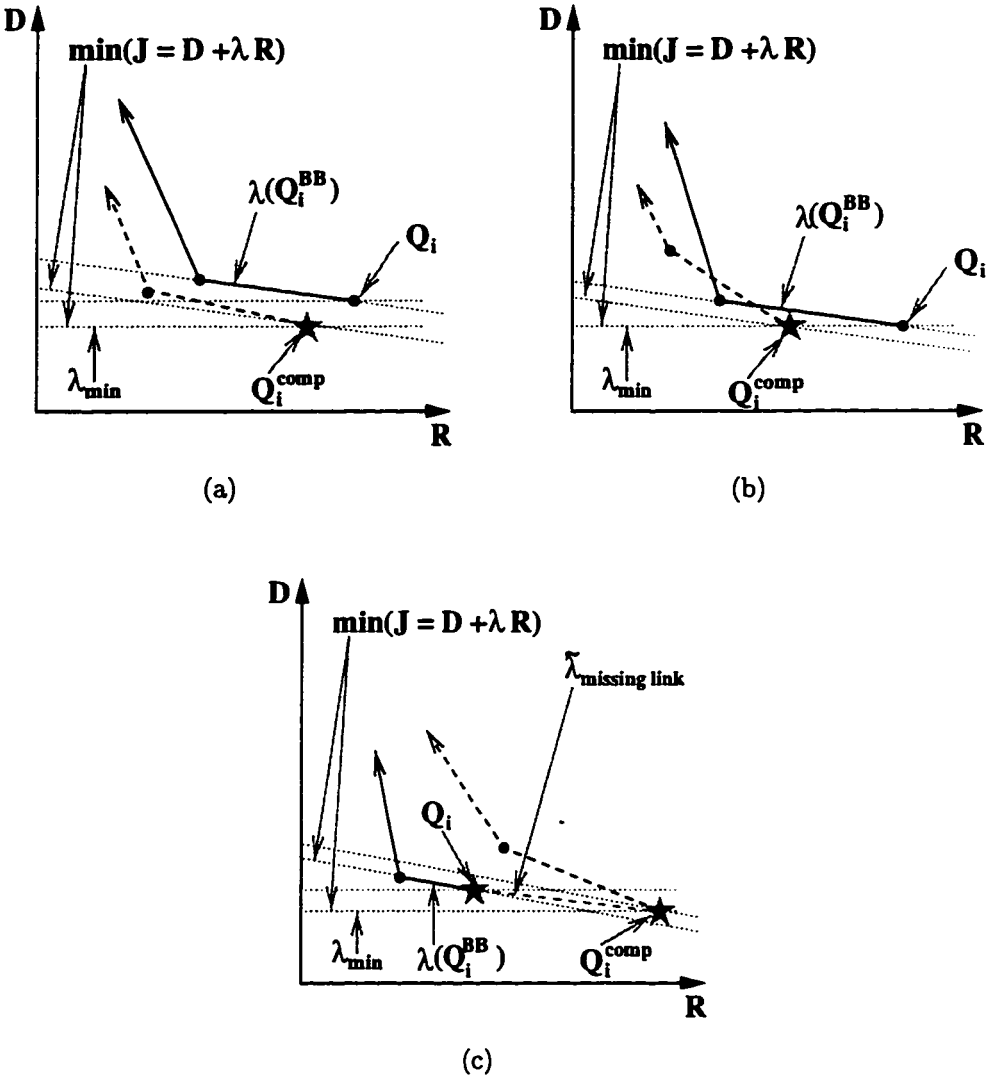


Figure 4.9: Node merging for WPT bit allocation during initialization. A star indicates that the quantizer lies on the lower convex hull of the rate-distortion curve. During initialization, the previous minimum slope λ_{min} is zero, and the next minimum slope of the current branch is $\lambda(Q_i^{BB})$. (a) If one quantizer has lower encoding costs for both λ_{min} and $\lambda(Q_i^{BB})$, then it should be used: e. g. both encoding costs of Q_i^{comp} are less than those of Q_i, the composite node should be used. (b) If the encoding costs are the same for λ_{min}, the lower rate quantizer should used: e. g. Q_i^{comp} has a lower rate, the composite node should be used. (c) If one quantizer has a lower cost for λ_{min} while the other has a lower cost for $\lambda(Q_i^{BB})$, the higher rate quantizer should be used, and the slope of the missing link $\tilde{\lambda}$ should be calculated and stored so that the higher rate quantizer will be used next as it also lies on the lower convex hull: e. g. Q_i^{comp} has a higher rate, the composite node should be used.

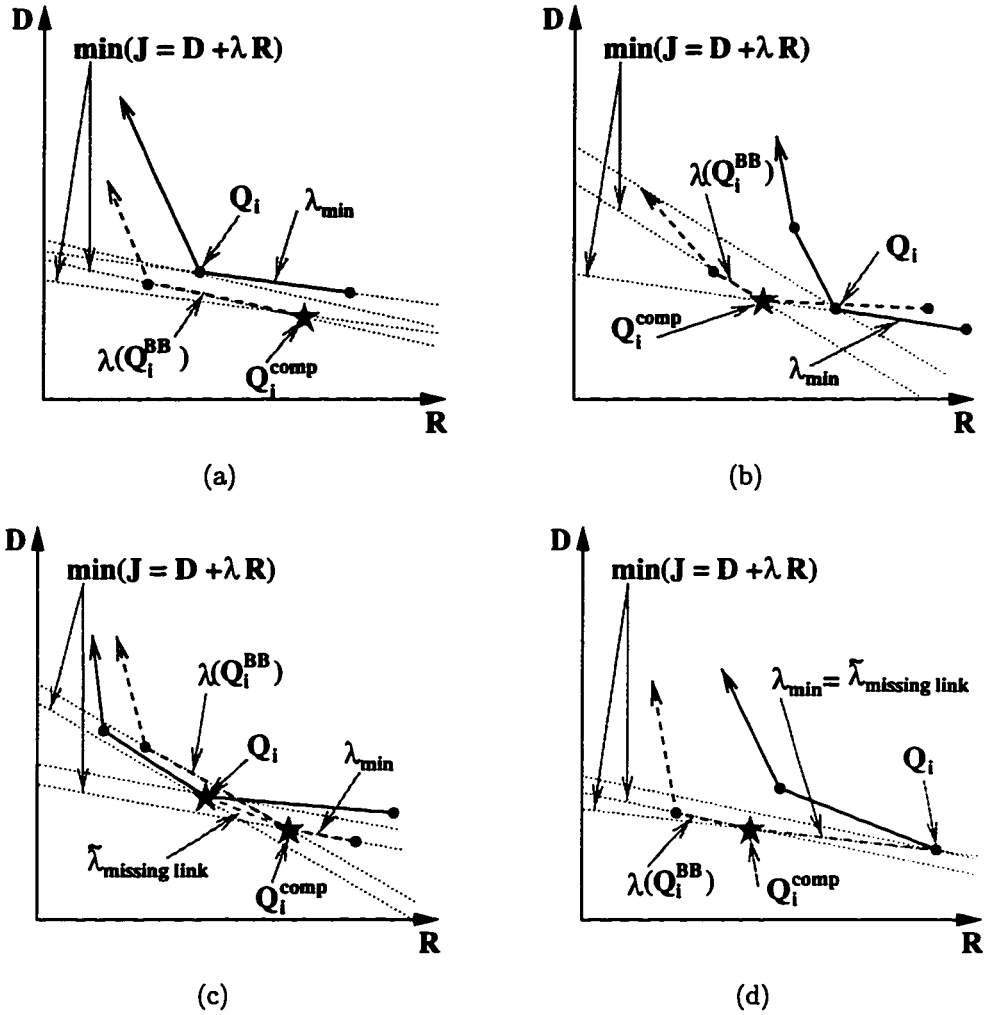


Figure 4.10: Node merging for the main WPT bit allocation algorithm. A star indicates that the quantizer lies on the lower convex hull of the rate-distortion curve. Before pruning, the minimum slope of the WPT is λ_{min}. After pruning, the minimum slope of the current branch is $\lambda(Q_i^{\text{BB}})$. (a) If one quantizer has lower encoding costs for both λ_{min} and $\lambda(Q_i^{\text{BB}})$, then it should be used: e. g. both encoding costs of Q_i^{comp} are less than those of Q_i, the composite node should be used. (b) If the encoding costs are the same for λ_{min}, the lower rate quantizer should used: e. g. Q_i^{comp} has a lower rate, the composite node should be used. (c) If one quantizer has a lower cost for λ_{min} while the other has a lower cost for $\lambda(Q_i^{\text{BB}})$, the higher rate quantizer should be used, and the slope of the missing link $\tilde{\lambda}$ should be calculated and stored so that the higher rate quantizer will be used next as it also lies on the lower convex hull: e. g. Q_i^{comp} has a higher rate, the composite node should be used. (d) If λ_{min} is a missing link slope $\tilde{\lambda}$, then λ_{min} is less than the next slope of the Q_i and Q_i^{comp}. Neither quantizer should be pruned, but the WPT should be updated using λ_{min} and $\lambda(Q_i^{\text{BB}})$ resulting in situation (b).

Chapter 5

COMPRESSION OF USC IMAGES

In this chapter I apply the wavelet/VQ compression schemes of Section 4.2 to grayscale images from the USC database for benchmarking purposes. Wavelet/VQ compression is applied to ocean science data in Chapter 6. The coding of color images for joint progressive transmission and low bit-depth display and printing is addressed in Appendix A.

For these experiments, five images (crowd, couple, man, woman1, and woman2) are used to train the PTSVQ's, and the test image is the standard "Lenna." I have used the S+WAVELETS package from the Statistical Science Division of MathSoft Inc., as well as the WaveLab .600 library for Matlab made available by Iain Johnstone *et al.* at Stanford University, with my VQ code to implement the DWT/PTSVQ and WPT/PTSVQ encoders. Bit allocation for the DWT is done using the optimal bit allocation algorithms based on GBFOS and ROPA, Sections 4.2.1 and 4.2.2. Results for this are presented in Section 5.1. Bit allocation for the WPT is done using the optimal algorithm introduced in Section 4.2.3. Results for this are presented in Section 5.2. Conclusions are presented in Section 5.3.

5.1 Wavelet Bit Allocation Results for the USC Images

To illustrate the differences between subbands, the distributions of coefficients of the training data decomposed by the DWT to three levels using the S8 wavelet are shown in Figure 5.1. The fine resolution detail coefficients have very peaked distributions, especially the diagonal components, and the variance of the distribution tends to

increase as the detail spaces become coarser. The coarse approximation coefficients have a much different distribution, more closely resembling the histogram of an image. This is in agreement with studies which have found that wavelet coefficients of images tend to have distributions that are highly peaked around zero [2].

The binary TSVQ's created from the high resolution data exhibit atypical behavior. To illustrate this, greedy tree growing, GBFOS pruning, and ROPA pruning for two subbands (the third-level coarse-approximation subband $\mathbf{V}_4$ and the first-level diagonal-content detail subband $\mathbf{W}_1^3$) using a vector dimension of 4, are shown in Figure 5.2.

First, the TSVQ's are created using the greedy growing technique described in Section 2.4. Generally, when a greedy-grown TSVQ is created, the first split of the tree reduces the distortion the most for the given increase in rate as compared to the rest of the tree, and the second split is the second best, and so on. The expected results for greedy tree growing are found for the coarse approximation data, Figure 5.2(a), where the rate 1 (bits per vector) codebook represents a significant drop in distortion. In the case of the high-resolution detail data, Figure 5.2(b), it is seen that the first split (i. e. the 1 bit per vector codebook) is not very good. The best tree growth in this case occurs for low-probability high-distortion data, and the worst tree growth occurs for high-probability low-distortion data. The greedy growing algorithm for the high resolution data concentrates on growing large branches that have very low-probability data, rather than short, compact trees with high probability data. This behavior is unexpected for TSVQ, and indicates that the greedy growing algorithm for TSVQ's is not operating well for the given data. It may be that the distribution of the data is not well suited for binary splitting, but that ternary, or higher-order M-ary, TSVQ's may be able to better adapt themselves to the data.

Secondly, the greedy-grown TSVQ's are pruned using GBFOS and the same training data. GBFOS for the typical greedy-grown TSVQ of the coarse approximation data yields many quantizers on the lower convex hull, Figure 5.2(c), indicating

that the greedy-growing algorithm is well suited to the data. GBFOS for the high-resolution detail data does not find many quantizers on the lower convex hull of the rate-distortion curve, especially at very low bit rates, Figure 5.2(d). The net result is that very few optimal quantizers are created, also indicating that the trees are not being grown well. This is a problem when allocating a small number of bits (less than 1 bpp) to the image as this requires low bit-rate quantizers for all of the subbands.

Thirdly, the greedy-grown TSVQ's are pruned using ROPA. ROPA for both the typical greedy-grown TSVQ of the coarse approximation data and the high-resolution detail data yields a large number of low bit rate quantizers, Figures 5.2(e) and 5.2(f). Hence bit allocation with ROPA will provide a large number of multiresolution codebooks at all bit rates.

For example, the number of codebooks found for DWT/PTSVQ encoding using bit allocation via GBFOS is shown in Table 5.1, and the number of additional codebooks found using bit allocation via ROPA is shown in Table 5.2. When small vector dimensions are used, ROPA finds relatively few additional suboptimal codebooks. When large vector dimensions are used, ROPA finds many more additional suboptimal codebooks. This indicates that as the vector dimension increases, the greedy-grown TSVQ's no not work as well in that the trees do not have many GBFOS subtrees, and that growing is inefficient as large branches are grown that do not optimally minimize rate-distortion performance.

I now evaluate the rate-distortion performance of DWT/PTSVQ compression using optimal bit allocation. Performance results as a function of different levels of decomposition for the training data are shown in Figure 5.3 where it is seen that for a fixed PTSVQ vector dimension, it is desirable to have more levels of decomposition of the signal. However, the gain decreases so rapidly that it may not be necessary to do more than two or three levels of decomposition. To study the performance as a function of vector dimension, five different three-level DWT/PTSVQ codecs are created using different vector dimensions for the PTSVQ's where each DWT/PTSVQ

Table 5.1: Codebook densities for DWT/PTSVQ bit allocation via GBFOS where there are three levels of DWT decomposition and R is the rate in bpp.

Dimension	$0 < R < 0.25$	$0.25 < R < 0.5$	$0.5 < R < 0.75$	$0.75 < R < 1$
1	69	60	112	28
2	901	1668	677	418
4	4548	1904	1892	4545
8	3112	2237	1690	2555
16	1446	1239	1280	1697

Table 5.2: Additional codebooks found by ROPA for DWT/PTSVQ bit allocation where there are three levels of DWT decomposition and R is the rate in bpp.

Dimension	$0 < R < 0.25$	$0.25 < R < 0.5$	$0.5 < R < 0.75$	$0.75 < R < 1$
1	29	17	15	0
2	462	549	261	183
4	5166	2445	2197	4014
8	10447	8389	7338	467
16	11670	31092	154	20314

encoder has the same vector dimension for all PTSVQ's. Results are shown in Figure 5.4(a) where it is seen that it is better to use larger vector dimensions, as is predicted by rate-distortion theory.

The DWT/PTSVQ codecs created above are next used to encode the test image "Lenna." The rate-distortion performance results of the test data are shown in Figure 5.4(b). Except for scalar quantization, the rate-distortion curves tend to rise sharply at very low bit rates and then suddenly flatten, showing no improvement at higher bit rates. The higher the dimension, the earlier this flattening occurs. This flattening has to do with the unsuitability of the PTSVQ's at higher bit rates. Since I use only five training images, and do not use overlapping training vectors, I have very little training data for the small subbands. This means that as the bit rate increases, the TSVQ's become overtrained, or overadapted, to the training data. The expected improvements for encoding performance when using higher dimensions are seen only at very low bit rates where the curve has not begun to taper off. Two examples of the compressed image with a first-order entropy of 0.248 bpp (a compression ratio of 32.26:1 and PSNR of 31.62 dB) and a first-order entropy of 0.128 bpp (a compression ratio of 62.46:1 and PSNR of 28.37 dB) using scalar quantization are shown in Figure 5.5. For comparison purposes, the EZW encoding scheme with arithmetic encoding [88] yields an image having a PSNR of 30.23 dB when the compression ratio is 64:1, and a PSNR of 33.17 dB when the compression ratio is 32:1.

5.2 Wavelet Packet Bit Allocation Results for the USC Images

The rate-distortion performance for training data of WPT/PTSVQ optimal bit allocation as a function of WPT depth is shown in Figure 5.6 where it is seen that PSNR results improve with increased tree depth. This is expected as a deeper tree permits more adaptability in signal representation and compression. However, the improvements decrease as the tree depth increases which was also is seen for DWT/PTSVQ

bit allocation. To study the performance as a function of PTSVQ vector dimension, five different three-level WPT/PTSVQ encoders are created using different vector dimensions for the PTSVQ's where each WPT/PTSVQ encoder has the same vector dimension for all PTSVQ's. Results are shown in Figure 5.7 where it is seen again that it is better to use larger vector dimensions. Note that some of the curves shown start increasing rapidly at higher bit rates. This is a reflection of overtraining of the PTSVQ's due to the finite amount of training data. All of the curves exhibit this behavior although it is not shown in these graphs. In addition, the number of codebooks found for WPT/PTSVQ encoding using optimal bit allocation is shown in Table 5.3 where is it seen that WPT/PTSVQ bit allocation generally produces more codebooks that DWT/PTSVQ bit allocation using GBFOS, as expected.

The WPT/PTSVQ codecs created above are next used to encode the test image "Lenna." The rate-distortion performance results as a function of WPT depth are shown in Figure 5.8. When scalar quantization is used, the results are exactly as expected, the best gain is achieved when using more levels of decomposition. However, this is true only at very low bit rates when using larger vector dimensions. The rate-distortion curves tend to taper off very rapidly as the bit rate increases indicating mismatch between the source and the encoder. The rate-distortion performance as a function of PTSVQ vector dimension is shown in Figure 5.9. When the depth of the tree is 0 (i. e. no levels of wavelet decomposition), the results are exactly as expected: the best gain is achieved when using larger vector dimensions. However, as the WPT depth increases, this it true only at very low bit rates. The rate-distortion curves tend to taper off very rapidly as the tree-depth and vector dimension increase, again indicating mismatch between the source and encoder.

A comparison of DWT/PTSVQ and WPT/PTSVQ optimal bit allocation for three levels of decomposition as a function of vector dimension is shown in Figure 5.10(a) for various vector dimensions. WPT/PTSVQ outperforms DWT/PTSVQ for all vector dimensions tried. This is expected since the DWT is a subset of the

Table 5.3: Codebook densities for WPT/PTSVQ bit allocation using GBFOS where there are three levels of WPT decomposition and R is the rate in bpp.

Dimension	$0 < R < 0.25$	$0.25 < R < 0.5$	$0.5 < R < 0.75$	$0.75 < R < 1$
1	66	162	292	405
2	1122	2574	4047	5855
4	4130	7919	11589	16201
8	3472	6409	9685	13318
16	1958	3838	6825	8363

WPT, and so WPT/PTSVQ should always do at least as well as the WPT/PTSVQ. Using the same codecs for the test image "Lenna," a comparison of DWT/PTSVQ and WPT/PTSVQ encoding schemes using three levels of decomposition is shown in Figure 5.10(b) as a function of vector dimension. Unexpectedly, the results for DWT/PTSVQ and WPT/PTSVQ are very close, and WPT/PTSVQ slightly outperforms DWT/PTSVQ at low bit rates. I believe this results from the WPT/PTSVQ scheme being overadapted to the training data at higher levels of decomposition, and less able to deal with test data. The WPT/PTSVQ bit allocation scheme almost always chooses to use the highest levels of decomposition. An example of the actual numerical bit allocations made for WPT of depth three using PTSVQ's with dimensions of four at 0.5 bpp is shown in Figure 5.11.

Three examples of the compressed image with a first-order entropy of 0.0755 bpp (a compression ratio of 105.97:1 and PSNR of 25.90 dB), a first-order entropy of 0.1253 bpp (a compression ratio of 63.84:1 and PSNR of 28.34 dB), and a first-order entropy of 0.2522 bpp (a compression ratio of 31.73:1 and PSNR of 31.76 dB) using scalar quantization are shown in Figure 5.12. For comparison purposes, the EZW encoding scheme with arithmetic encoding yields a PSNR of 27.54 dB when the compression ratio is 128:1, a PSNR of 30.23 dB when the compression ratio is 64:1, and a PSNR of 33.17 dB when the compression ratio is 32:1.

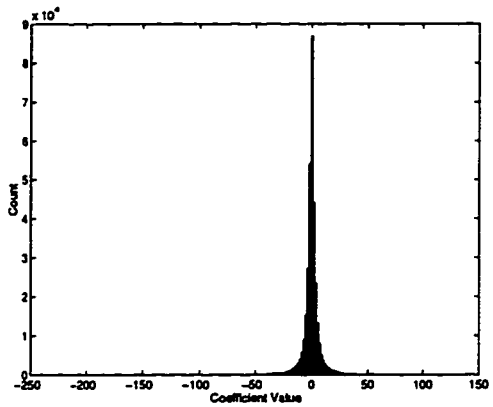
5.3 Conclusions

In this chapter I have presented results for bit allocation for DWT/PTSVQ using GBFOS and ROPA pruning methods as well as optimal bit allocation for WPT/PTSVQ based on the GBFOS pruning approach. As expected, ROPA found many more codebooks than GBFOS. For the same bit rates, WPT bit allocation generally found more codebooks too. It was shown that the greedy grown binary trees were not well suited to the detail subbands. If the greedy grown trees were initially better than those created here, ROPA bit allocation would probably not be needed. It would be worthwhile to investigate ternary TSVQ, or other types of VQ, to find an approach that is more suited to highly peaked data in terms of the density of codebooks found on the lower convex hull of the rate-distortion curve.

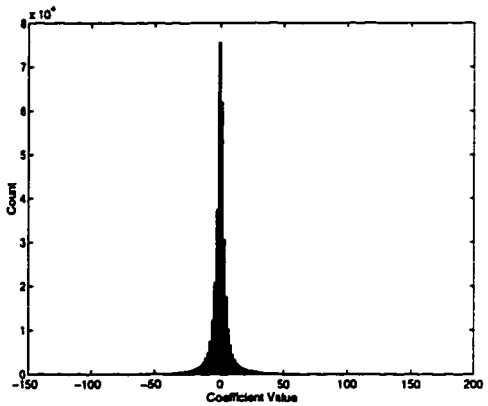
The results for the training data showed that WPT/PTSVQ outperformed DWT/PTSVQ, which is expected since the DWT is a subset of the WPT, and the WPT approach is inherently adaptive. It was also seen that results improved as the vector dimension and decomposition level increased. Rate-distortion theory predicts that it is better to code with larger dimensions, and more levels of decomposition permit better energy compaction and adaptation to the source by enabling more flexibility for bit allocation by the creation of multiple subband sources.

Unfortunately, the results for test data behaved as expected only at very low bit rates. I believe that this has to do with the DWT and the WPT, especially, overadapting themselves to the limited amount of training data, or equivalently, that the test image differs significantly from the training data. For example, the "crowd" image, which is a scene containing a number of people, is not similar to "Lenna," which can charitably be considered a portrait. Also, it was seen that as vector dimension increased, the performance was better only at low bit rates. Again, I believe that this has to do with overtraining since the high dimension curves flattened out rapidly for the test data.

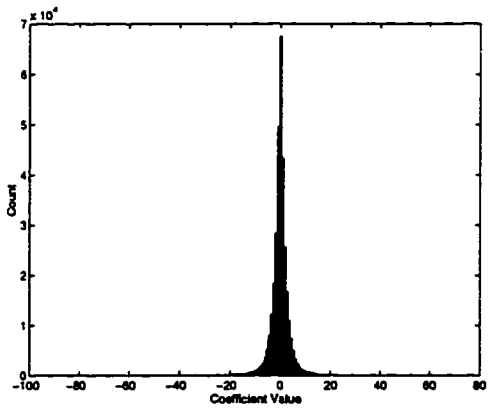
A comparison of my results to the EZW shows that my method lags the EZW by about 1.5 dB at most bit rates. It should be noted that my results are for the first-order entropy of the index stream, whereas the EZW results are computed using arithmetic coding of indices. It is possible that my results may be comparable to the EZW if I were also to use an arithmetic encoder. This is especially true since at very low bit rates, the improvement in SNR is greatly accelerated as the bit rate increases, and any shift of the rate-distortion curve to the left (as would result with better lossless coding) can improve reported SNR results by several dB.



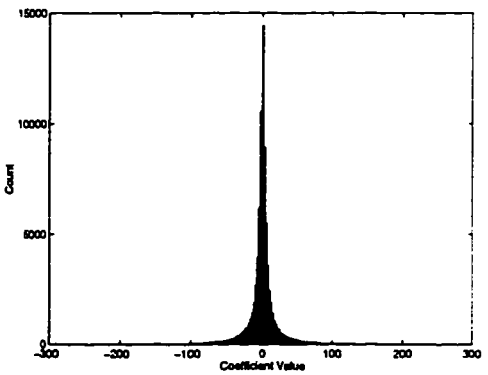
(a) Histogram of coefficients in horizontal detail space $\mathbf{W}_1^1$.



(b) Histogram of coefficients in vertical detail space $\mathbf{W}_1^2$.

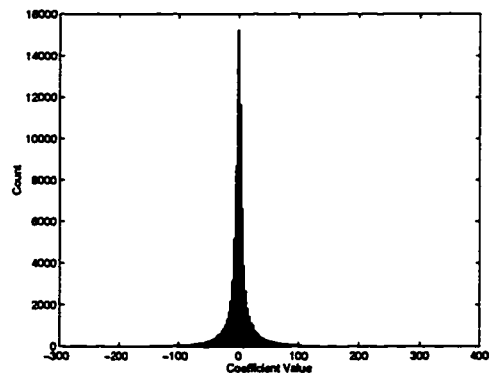


(c) Histogram of coefficients in diagonal detail space $\mathbf{W}_1^3$.

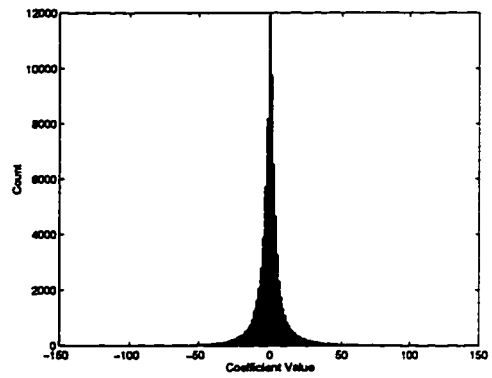


(d) Histogram of coefficients in horizontal detail space $\mathbf{W}_2^1$.

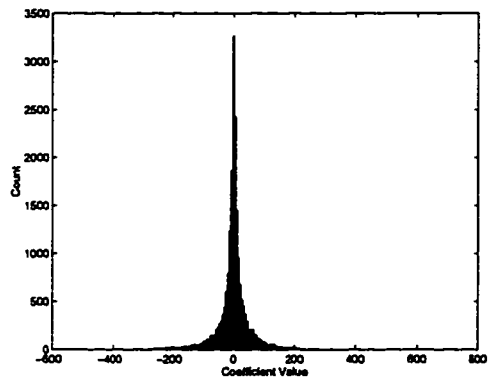
Figure 5.1: The distribution of wavelet coefficients found among the S8 wavelet transformation of the five training images where (a) is the horizontal component of the first level decomposition, (b) is the vertical component of the first level decomposition, (c) is the diagonal component of the first level decomposition, and (d) is the horizontal component of the second level decomposition.



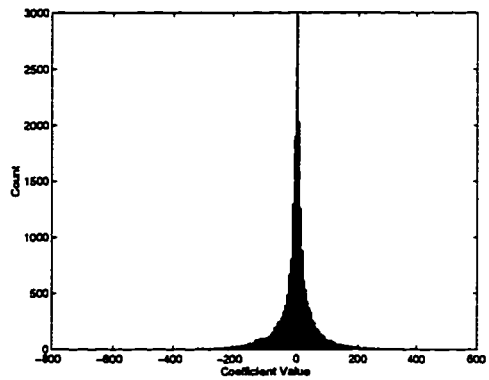
(e) Histogram of coefficients in vertical detail space $\mathbf{W}_2^2$.



(f) Histogram of coefficients in diagonal detail space $\mathbf{W}_2^3$.

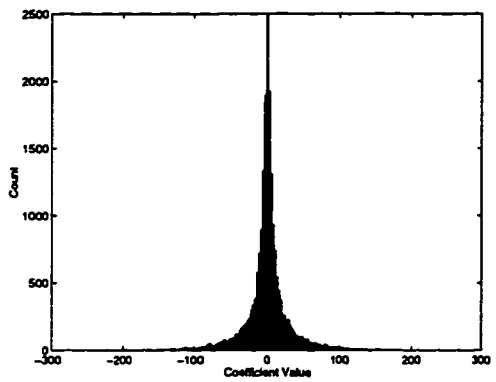


(g) Histogram of coefficients in horizontal detail space $\mathbf{W}_3^1$.

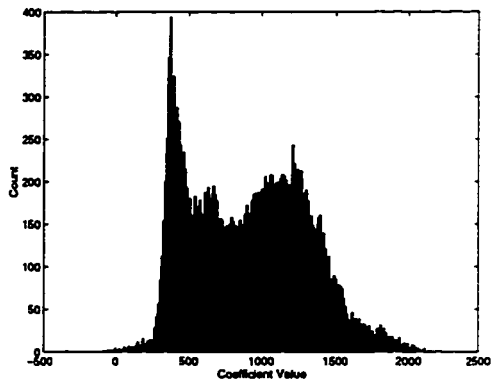


(h) Histogram of coefficients in vertical detail space $\mathbf{W}_3^2$.

Figure 5.1: *(Continued).* The distribution of wavelet coefficients found among the S8 wavelet transformation of the five training images where (e) is the vertical component of the second level decomposition, (f) is the diagonal component of the second level decomposition, (g) is the horizontal component of the third level decomposition, and (h) is the vertical component of the third level decomposition.

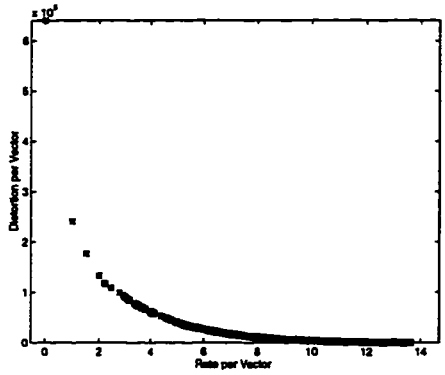


(i) Histogram of coefficients in diagonal detail space $\mathbf{W}_3^3$.

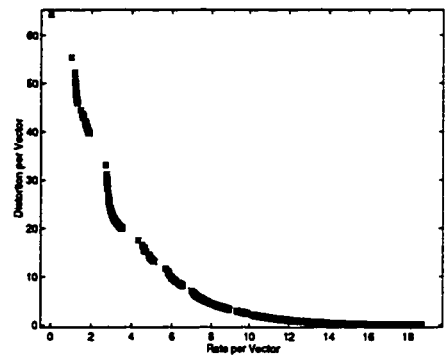


(j) Histogram of coefficients in coarsest resolution space $\mathbf{V}_4$.

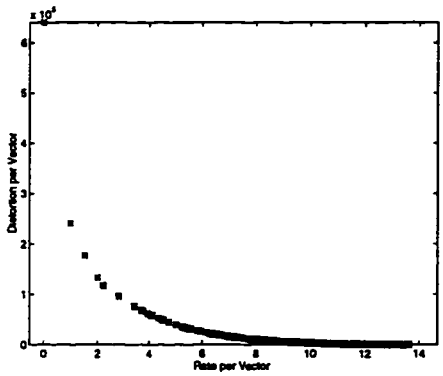
Figure 5.1: *(Continued).* The distribution of wavelet coefficients found among the S8 wavelet transformation of the five training images where (i) is the diagonal component of the third level decomposition, and (j) is the coarsest resolution image of the third level decomposition.



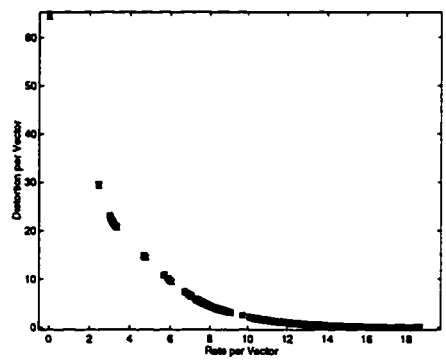
(a) Greedy growing, subband $\mathbf{V}_4$.



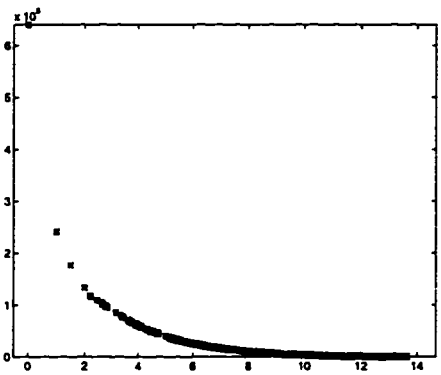
(b) Greedy growing, subband $\mathbf{W}_1^3$.



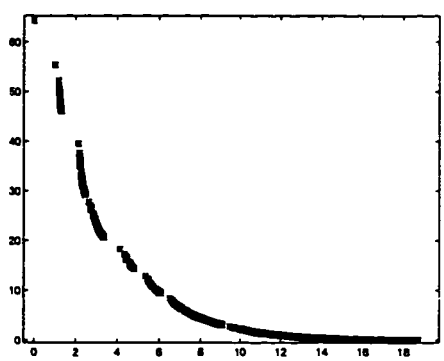
(c) GBFOS pruning, subband $\mathbf{V}_4$.



(d) GBFOS pruning, subband $\mathbf{W}_1^3$.

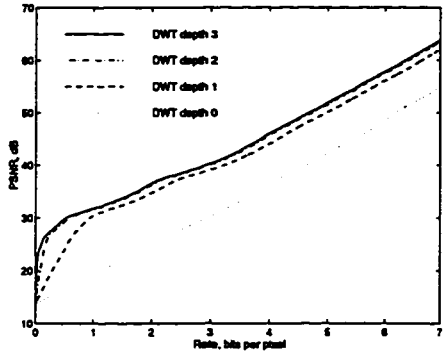


(e) ROPA pruning, subband $\mathbf{V}_4$.

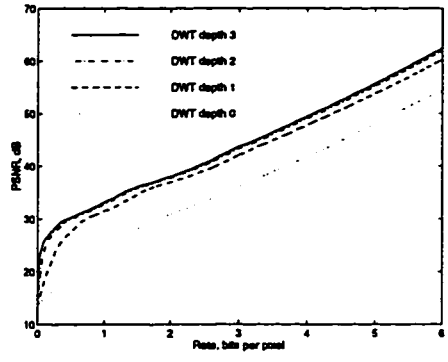


(f) ROPA pruning, subband $\mathbf{W}_1^3$.

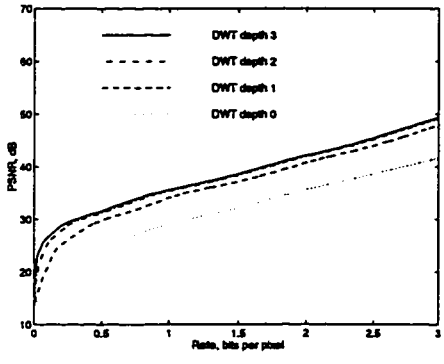
Figure 5.2: Growing and pruning results for TSVQ's from different subband sources of the training data. (a) Greedy growing, (c) GBFOS pruning, and (e) ROPA pruning for the TSVQ of the (low frequency) coarsest resolution subband of the third-level decomposition $\mathbf{V}_4$ using a vector dimension of 4. (b) Greedy growing, (d) GBFOS pruning, and (f) ROPA pruning for the TSVQ of the (high frequency) fine resolution detail subband of the first-level decomposition $\mathbf{W}_1^3$ using a vector dimension of 4.



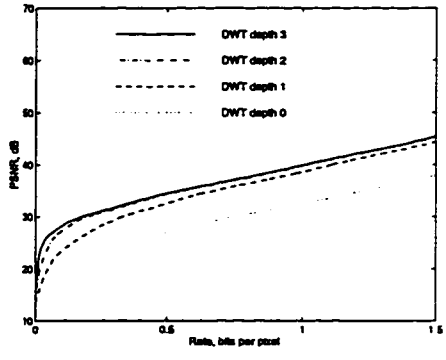
(a) Vector dimension 1.



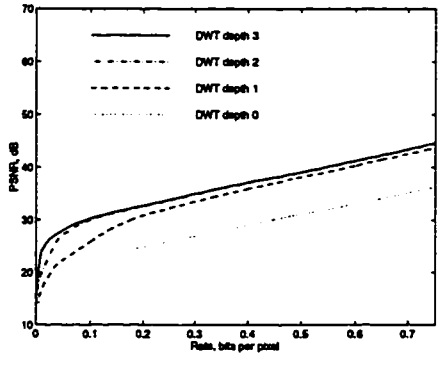
(b) Vector dimension 2.



(c) Vector dimension 4.

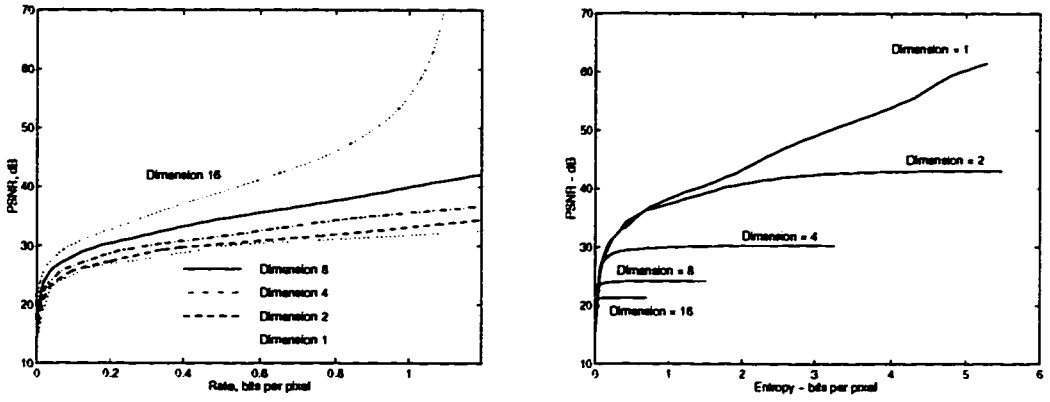


(d) Vector dimension 8.



(e) Vector dimension 16.

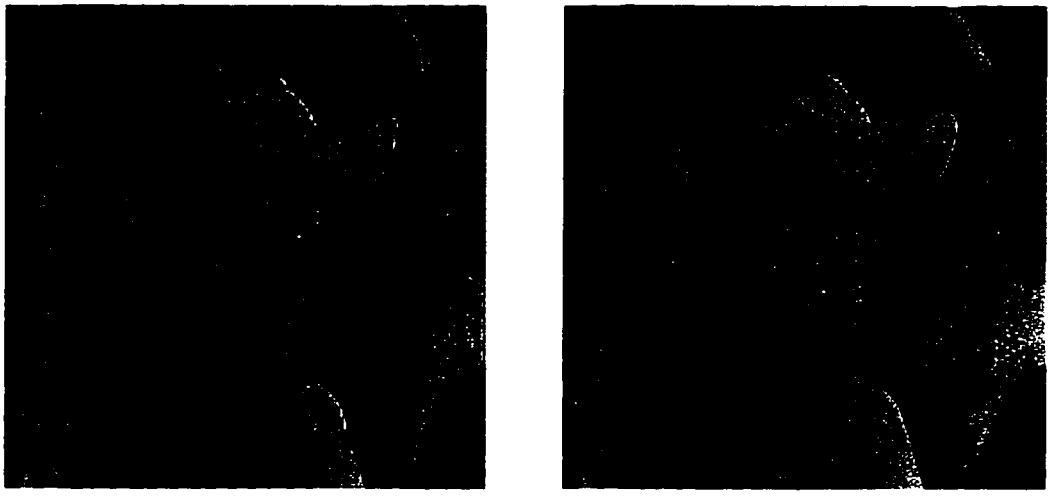
Figure 5.3: DWT/PTSVQ bit allocation results as a function of DWT depth for the training data. PSNR versus rate for the training data using different DWT depths ranging from 0 to 3. The same vector dimension is used for all subband PTSVQ's: (a) 1 (scalar quantization), (b) 2, (c) 4, (d) 8, and (e) 16 coefficients per vector.



(a) Training data.

(b) Test data.

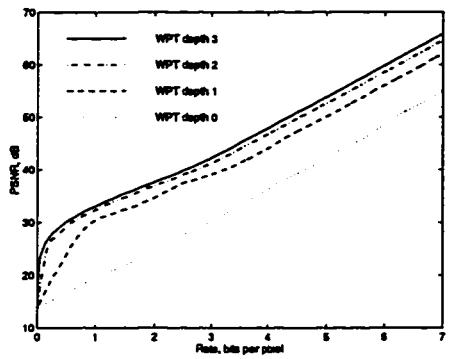
Figure 5.4: DWT/PTSVQ bit allocation and test results as a function of PTSVQ vector dimension using a three-level DWT decomposition. (a) PSNR versus rate for the training data where the PTSVQ vector dimensions are 1, 2, 4, 8, and 16 coefficients per vector and all PTSVQ's have the same vector dimension. (b) PSNR versus entropy for the test image "Lenna" where the vector dimensions are 1, 2, 4, 8, and 16 coefficients per vector and all PTSVQ's have the same vector dimension.



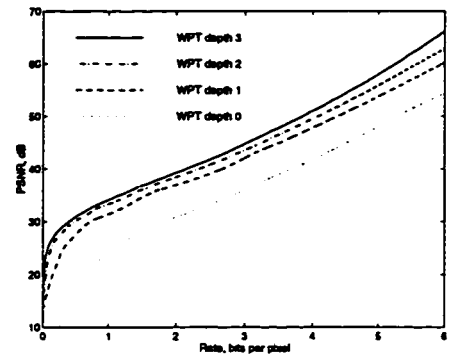
(a) 0.1281 bpp, 28.37 dB.

(b) 0.2480 bpp, 31.62 dB.

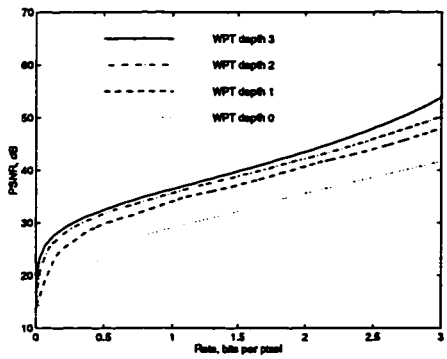
Figure 5.5: DWT/PTSVQ compression on the test image "Lenna." (a) Scalar quantization with a first-order entropy of (a) 0.1281 bpp (a compression ratio of 62.46:1) and a PSNR of 28.37 dB, and with a first-order entropy of (b) 0.2480 bpp (a compression ratio of 32.26:1) and a PSNR of 31.62 dB.



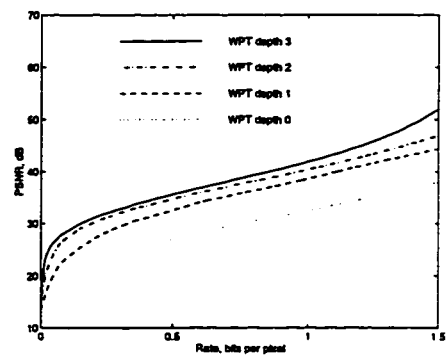
(a) Vector dimension 1.



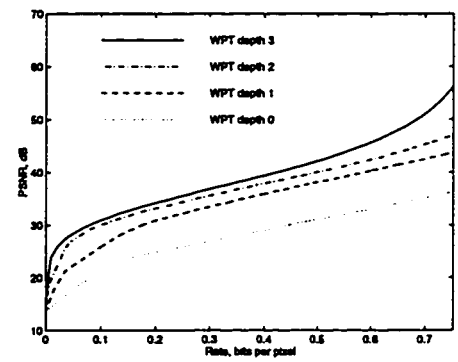
(b) Vector dimension 2.



(c) Vector dimension 4.

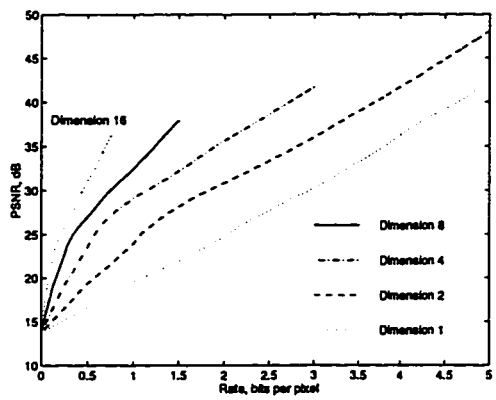


(d) Vector dimension 8.

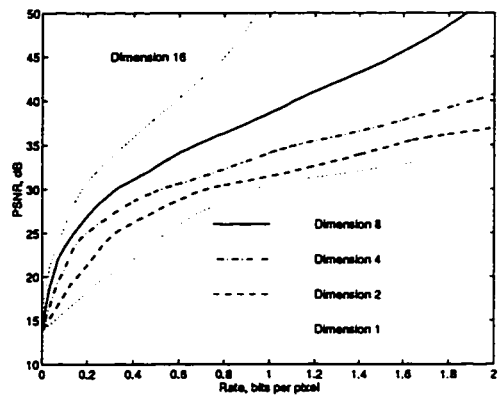


(e) Vector dimension 16.

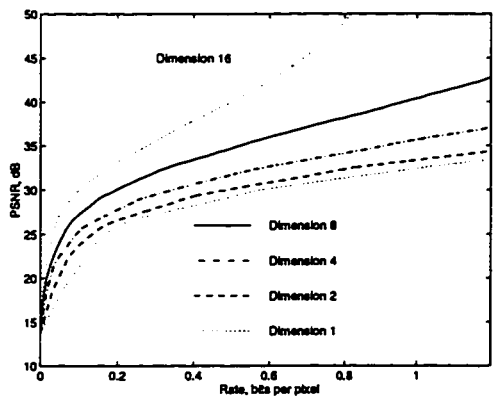
Figure 5.6: PSNR versus rate for WPT/PTSVQ bit allocation as a function of WPT depth ranging from 0 to 3 for the training data. The same vector dimension is used for all subband PTSVQ's: (a) 1 (scalar quantization), (b) 2, (c) 4, (d) 8, and (e) 16 coefficients per vector.



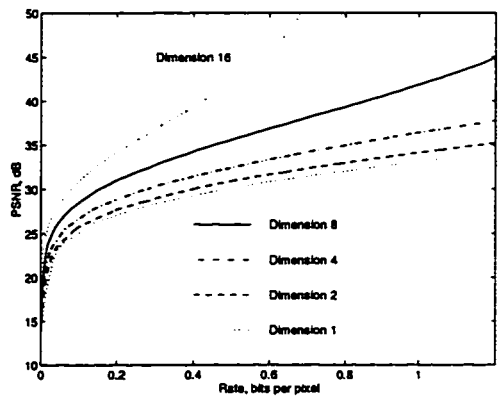
(a) WPT depth 0.



(b) WPT depth 1.

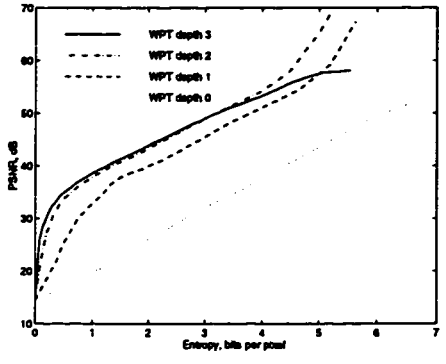


(c) WPT depth 2.

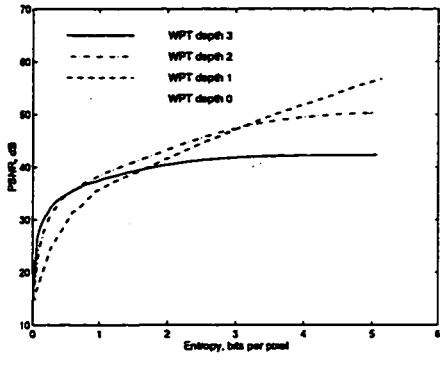


(d) WPT depth 3.

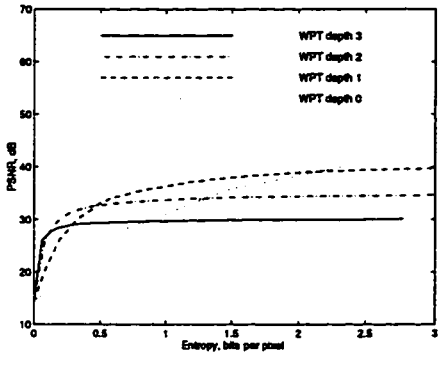
Figure 5.7: WPT/PTSVQ bit allocation results as a function of vector dimension. PSNR versus rate for training data using (a) depth-0, (b) depth-1, (c) depth-2, and (d) depth-3 WPT's with varying PTSVQ vector dimensions of 1, 2, 4, 8, and 16 coefficients per vector where all PTSVQ's in the WPT have the same vector dimension.



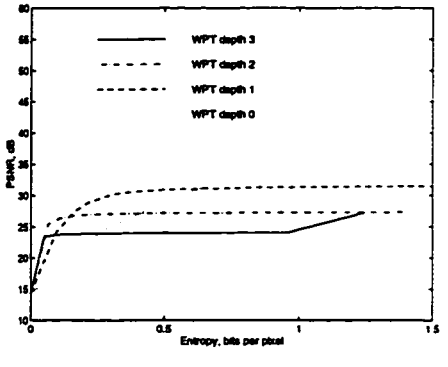
(a) Vector dimension 1.



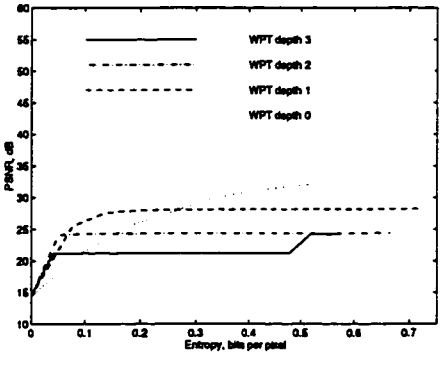
(b) Vector dimension 2.



(c) Vector dimension 4.

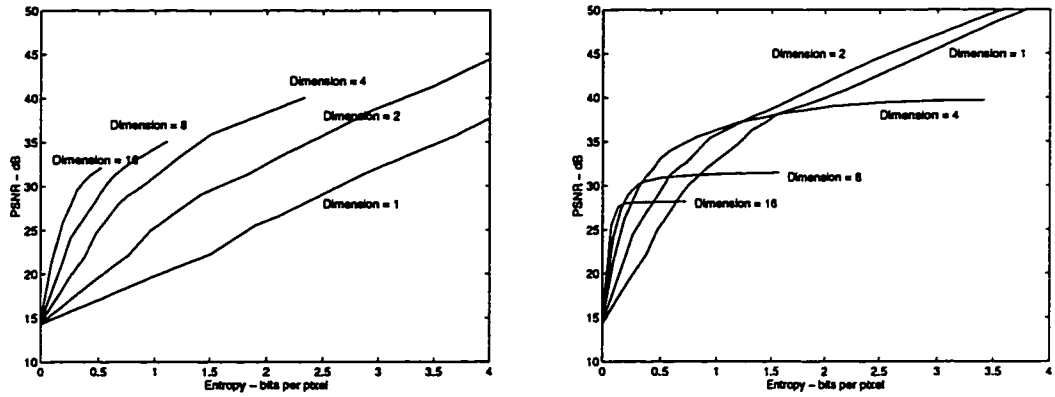


(d) Vector dimension 8.



(e) Vector dimension 16.

Figure 5.8: WPT/PTSVQ test image results as a function of WPT depth. PSNR versus entropy for the test image "Lenna" using different WPT depths ranging from 0 to 3. The same vector dimension is used for all subband PTSVQ's: (a) 1 (scalar quantization), (b) 2, (c) 4, (d) 8, and (e) 16 coefficients per vector.



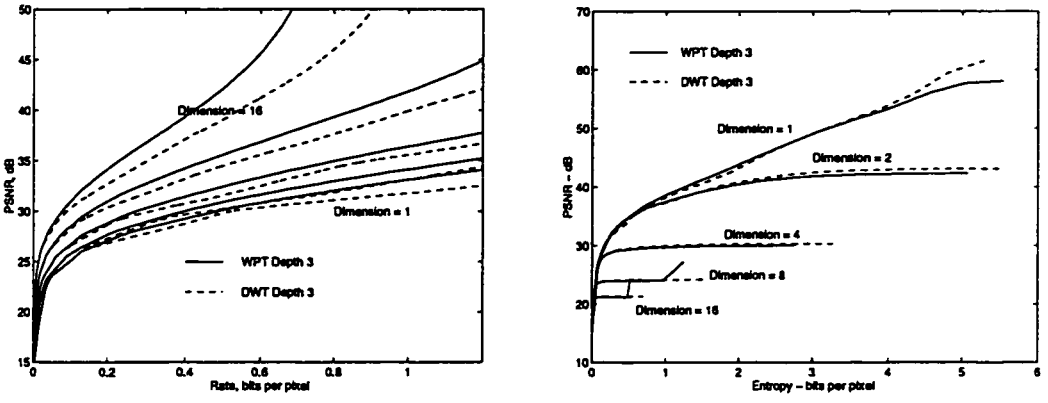
(a) WPT depth 0.

(b) WPT depth 1.

(c) WPT depth 2.

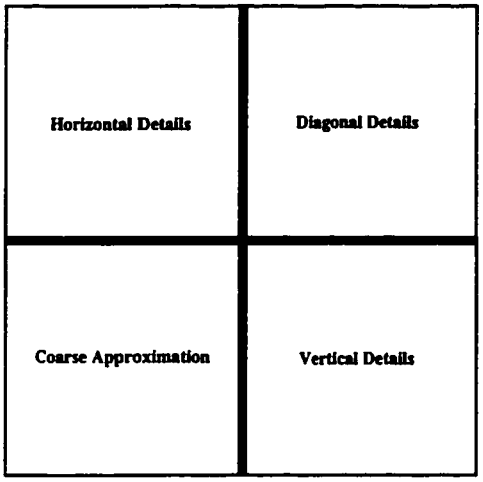
(d) WPT depth 3.

Figure 5.9: WPT/PTSVQ test image results as a function of vector dimension. PSNR versus entropy for the test image "Lenna" encoded using (a) depth-0, (b) depth-1, (c) depth-2, and (d) depth-3 WPT's with varying vector dimensions of 1, 2, 4, 8, and 16 coefficients per vector where all PTSVQ's in the WPT have the same vector dimension.



(a) Training data. (b) Test data.

Figure 5.10: A comparison of DWT/PTSVQ and WPT/PTSVQ bit allocation using three levels of decomposition. (a) PSNR versus rate for training data. Note that WPT/PTSVQ outperforms DWT/PTSVQ for the same vector dimensions (1, 2, 4, 8, 16 from the bottom of the graph to the top) as expected since the DWT is a subset of the WPT. (b) PSNR versus entropy for the test image "Lenna." Note that WPT/PTSVQ outperform DWT/PTSVQ only at low bit rates.

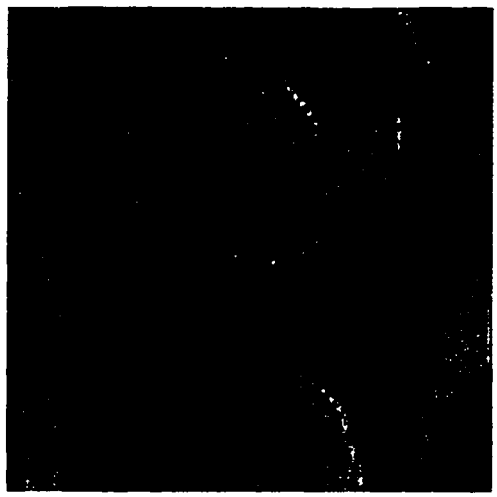


(a)

3.663	0	0	0	0	0	0	0
4.788	3.605	0	0	0	0	0	0
1.000	0	0	0	0	0	0	0
1.014	0	0	0	0	0	0	0
8.021	5.548	2.805	3.012	0	0	0	0
6.484	3.120	0	2.842	0	0	0	0
10.623	7.815	2.675	6.218	0	0	2.337	0
13.471	9.885	4.697	6.174	1.002	0	4.306	3.232

(b)

Figure 5.11: (a) For each subband decomposition, the resulting subbands are arranged so that the upper left-hand corner contains the horizontal detail, the upper right-hand corner contains the diagonal detail, the lower right-hand corner contains the vertical detail, and the lower left-hand corner contains the coarse approximation. (b) The numerical bit allocations (rate, bits per vector) made for a WPT of depth three and a vector dimension of four at a design rate of 0.5 bpp.



(a) 0.0755 bpp, 25.90 dB.

(b) 0.1253 bpp, 28.24 dB.

(c) 0.2522 bpp, 31.76 dB.

Figure 5.12: WPT/PTSVQ compression on the test image "Lenna." Scalar quantization with a first-order entropy of (a) 0.0755 bpp (a compression ratio of 105.97:1) and a PSNR of 25.90 dB, with a first-order entropy of (b) 0.1253 bpp (a compression ratio of 63.84:1) and a PSNR of 28.34 dB, and with a first-order entropy of (c) 0.2522 bpp (a compression ratio of 31.73:1) and a PSNR or 31.76 dB.

Chapter 6

COMPRESSION OF ACOUSTIC DATA

Fast retrieval, previewing, and browsing of large volumes of data in remotely located underwater sensors is an important problem that must be addressed as the amount of data to be stored and transmitted grows rapidly. One example is of remotely collected data from underwater monitors such as the University of Washington's Benthic Acoustic Measurement System. This is an autonomous sonar platform placed on the ocean floor which can operate for several months. Data are transmitted by an RS232 cable temporarily connected by a diver to a nearby ship [35]. Other examples are remotely operated vehicles such as the Massachusetts Institute of Technology's Odyssey II which uses a two-way acoustic modem and can operate, under certain conditions, for up to 40 hours [10].

The data collected by remote sensors are typically very large in quantity. Unfortunately, underwater transmission channels usually have very limited bandwidths, especially acoustic modems which typically operate at very low baud rates. Sending a 10 MByte file of data over the most recently engineered acoustic modems [89], which operate at best at 40 kbaud, would take 35 minutes. Quick look, previewing, and browsing capabilities as well as easy access to the data are vital operations needed to evaluate the relevancy and the quality of the data, both for scientific and operational purposes. Sending all of the data, although optimal, is clearly not practical. However, the large reduction in time needed to transmit data is an important potential gain that warrants a serious study of the feasibility of lossy compression of underwater acoustic data.

In this chapter, I use high frequency (300 kHz) acoustic scattering data from the

Benthic Acoustic Measurement System (BAMS) to study the effects of lossy compression. BAMS, which takes sea-bed scattering measurements in shallow water, was built for the Sediment TRansport Events on Shelves and Slopes (STRESS) project to monitor sediment movement caused by storms. The research goals of BAMS have expanded to cover a broad range of scientific research and applications including benthic ecology studies, ocean floor activity monitoring, detection and classification of mine-like objects, ocean temperature studies, and modeling of the physics of acoustic scattering, [35, 57, 58, 92].

Using six scans of the BAMS data, all containing mine-like objects, I examine the results of several compression schemes including PTSVQ and wavelets/PTSVQ and evaluate the effects of compression on the output of two scientific applications. The use of complex wavelets, wavelet packets, and other wavelets filters was examined in [19] and showed no improvement, in terms of energy compaction, over the DWT. Hence only wavelets/PTSVQ are used in this chapter. The first scientific application is previewing and target detection computed from the power of the pressure field backscatter measurements. The second application is ocean floor temperature measurements computed from changes in the correlation of the pressure field over time. The three goals for these experiments are: to allow quick and easy previewing of the data (so the data could be transmitted over an acoustic modem); to preserve data analysis as accurately as possible; and to preserve targets for accurate detection at high compression ratios.

In Section 6.1 descriptions of BAMS and the experimental data are presented. Properties of the data are discussed in Section 6.2. Descriptions of the two scientific analyses are in Section 6.3. The proposed compression experiments and their evaluation are presented in Section 6.4. Experimental results are presented in Section 6.5. Conclusions are presented in Section 6.6.

6.1 The Benthic Acoustic Measurement System and Data

The Benthic Acoustic Measurement System [58] is an autonomous sonar platform that is mounted on the ocean floor. It has two sonars: the first operates at 40 kHz, and the second, newer, sonar operates at 300 kHz. BAMS has on-board storage capacity of approximately 120 MByte for the 40 kHz sonar, and 2000 MByte for the 300 kHz sonar. Data are retrieved by a temporary connection of an RS232 cable from a nearby ship by a diver.

For my experiments the output of the 300 kHz sonar which scans a circular section of radius 50 to 75 meters is used. It is comprised of 360 transmissions, or pings, separated by a 1° rotation of the transmitter and receiver. The complex baseband sonar signal is the result of a large amount of digital signal processing (including filtering and decimation) done to the output of a 12-bit A/D converter operating at a frequency below the Nyquist frequency.

The BAMS data used here are part of the U. S. Office of Naval Research's ORCAS experiment that was conducted off of Orcas Island, Washington over a two month period to monitor benthic activity and detect mine-like objects [58]. Six scans are used for these experiments, each containing mine-like objects. They were taken over a 48 hour period (July 29 to July 31, 1995) with spacings of 5 to 12 hours between each set of measurements.

Of the data examined, each ping contains 2401 complex numbers. When the data are analyzed, the first 240 complex values of each ping are discarded since they result from either sampling done before the transmitter is turned on or because the echo is at first too loud when the transmitter is first turned on. The next 1359 values are used in the analyses and represent the first 50 meters of the circular section; however, the first 119 values of these are ignored since there has not yet been enough time for the echo to return from the ocean floor. Any remaining data samples beyond this point (i. e. the last 25 meters) are not considered since noise from the ocean

surface interferes too much with the signal for there to be any data of good quality. Following the convention of using data with a sample size that is a power of two, the first 240+112 values are skipped and the next 2048 values of each ping are considered for this work.

Each scan, stored in an ascii file, is about 23.44 MByte in size but compresses losslessly to about 6.5 MByte using the Unix `compress` utility. The ascii file format is as follows. First there is a two line header to indicate the scan number. Then there are 360 sets of complex data, one for each ping, with the following information: eight lines of header information including ping number, date, time, Xdcr position, and Inclinometer, followed by 2401 lines of complex data from two slightly offset channels, Channel 0 and 1, which comprise the receiver. Next there are 5 lines of information from a third channel, Channel 2, which is the transmitter monitor, followed by 91 lines of complex data for Channel 2. Finally there are three blank lines, leading to a total of 2508 lines per ping. Of all of the data, only the data from Channel 0 are used although either channel or a linear combination of the two channels (which is used to adjust the directionality of the vertical angle) could be used. The data from Channel 2 are used for matched filtering which denoises the ping data and improves the resolution of the data by 20, that is for every current data sample, there would be 20 reconstructed data samples along the ping. The matched filtering code had not yet been written at the time of these experiments.

6.2 Properties of the BAMS Data

As stated previously, six scans are used for these experiments, each containing mine-like objects. Each scan contains 360 pings of 2048 complex samples, given in complex Cartesian coordinates (X, Y), from Channel 0. The first four scans (1, 2, 3, and 4) are used as training data to create the VQ's by sampling every fourth ping where the ith ping of the training data is the ith ping of scan j where $j = i \bmod 4 + 1$, and

Table 6.1: Acoustic data range, and rate and entropy calculations.

	Training Data	Scan 5	Scan 6
Minimum Value (Real)	-97.70	-78.50	-94.40
Minimum Value (Imaginary)	-77.19	-76.47	-96.03
Maximum Value (Real)	77.06	86.03	118.24
Maximum Value (Imaginary)	90.10	87.93	103.99
Rate per Complex Number	30	30	30
Entropy per Complex Number	17.36	17.77	17.27

$i = 1, \ldots, 360$. The remaining two scans (5 and 6) are reserved for testing. Both test sets contain five targets: two spheres, two cylinders, and a "reference target," which are shown in Figure 6.1.

The range of the data and the associated rate and first order entropy are shown in Table 6.1. The values of the training and test data range from -97.70 to $+118.24$. Since all values are accurate to the second decimal place, approximately

$$\lceil \log_2(21594) \rceil = \lceil 14.4 \rceil = 15.0$$

bits are required for each value, or 30.0 bits are required to represent each complex number. The first order entropies for scans 5 and 6 are 17.77 and 17.27 bits per complex number respectively. Thus compression ratios (CR's) of approximately 1.62:1 and 1.67:1 respectively could be achieved by a simple lossless compression algorithm.

The real and imaginary components of the data are considered to be independent and identically distributed zero-mean Gaussian data. When working in the polar domain, the magnitude and phase components are assumed to be statistically independent where the magnitude is Rayleigh distributed and the phase is uniformly distributed [56]. The mean values and standard deviations of the data (in complex form (X, Y) and as power and phase (P, ϕ)) are shown in Table 6.2. For further illustration purposes, two pings of scan 5 are examined in detail: ping 42, shown in Figure 6.2, does not contain a target; while ping 84, shown in Figure 6.3, contains a

Table 6.2: Mean and standard deviation of the acoustic data.

	Mean (X,Y)	Std. Dev. (X,Y)	Mean (P,ϕ)	Std. Dev. (P,ϕ)
Training Data	(-0.001, -0.005)	(3.27, 3.28)	(-6.037, 0.087)	(17.58, 1.81)
Scan 5	(0.015, -0.007)	(3.44, 3.43)	(-4.401, 0.067)	(16.32, 1.82)
Scan 6	(-0.017, -0.008)	(3.38, 3.37)	(-6.498, 0.089)	(18.18, 1.81)
Ping 42	(-0.140, 0.152)	(2.87, 3.24)	(-5.051, 0.111)	(17.43, 1.82)
Ping 84	(0.246, 0.147)	(7.15, 7.47)	(-3.296, -0.012)	(17.02, 1.83)

portion of the reference target. Note that the mean and standard deviation of ping 84 differ significantly from those of the entire scan since it contains a target. In general, pings containing portions of targets will differ statistically from the majority of pings which do not contain targets. This in turn will cause more difficulty in the coding or compressing of pings containing targets due to statistical mismatch between the data and the codebooks.

6.3 Scientific Objectives of the BAMS Data

The first scientific application examined is target detection [58]. Targets are detected by visual inspection of backscatter plots which are computed from the power of the pressure field. The power of the complex data $X+iY$ is computed in polar coordinates as magnitude $M = \sqrt{X^2+Y^2}$ and power $P = 20\log_{10} M$. The scattering plots are computed over all 360 pings. Each ping is independently processed and divided into 40 non-overlapping regions (pixels) which cover a radial distance of one meter by an arc length of 1°. The output image has 360×40 pixels, and the annular region of the scattering plot ranges from 10 to 50 meters from the transmitter/receiver.

Specifically, let $R = 1,\ldots,360$ be the range of pings, and let $N = 1,\ldots,2048$ be the range of samples. The complex samples are defined to be $X(r,n) + iY(r,n), r \in R, n \in N$, and the range (in meters) of the bins, or processed pixels, is defined as

$B = 1, \ldots, 50$. The function $\Gamma : N \rightarrow B$ maps each time sample n to a spatial bin b based on the distance of the sample to the *base* of the sonar. This distance, r_{floor}, is computed as

$$r_{floor} = \sqrt{r_{sonar}^2 - h_{sonar}^2}$$

where r_{sonar} is the distance from the sonar to the ocean floor, and h_{sonar} is the height of the sonar (5.16 m). The distance from the sonar to the ocean floor is calculated as

$$r_{sonar} = \frac{c}{2}\left(t_0 - t_{trx} - \frac{\tau}{2} + \frac{(n-1)}{f_{samp}}\right)$$

where c is the speed of sound (1490.2 m/s), t_0 is the time of the first sample (0.002 s), t_{trx} is the time of transmission (0 s), τ is the pulse length (0.002 s), n is the sample number, and f_{samp} is the sampling rate (20 kHz).

The backscatter pressure strengths are computed as:

$$P(r,b) = 10\log_{10}\left(\sum_{n=n_b}^{n_{b+1}-1} \frac{X^2(r,n) + Y^2(r,n)}{A_1(n_{b+1} - n_b)} + A_2\right) - A_3(b) + A_4(b) - A_5$$

for $r \in R$ and $b \in B$ where n_b is the sample in bin b closest to the sonar, $(n_{b+1} - n_b)$ is the number of samples in bin b, A_1 and A_2 are constants used for calibration, $A_3(b)$ is an adjustment based on the angle of the sensor to the center of pixel b, $A_4(b)$ is an adjustment based on the radius to the center of pixel b, and A_5 is a constant which makes adjustments for instrument and ocean conditions.

The second scientific application examined is that of ocean floor temperature measurements [57] which are computed from the magnitude of correlation measurements which are in turn computed after the phase of two temporally adjacent scans is perturbed. The basic idea is that the speed of sound changes as the ocean temperature changes, and that this results in a change in detected phase. The phase perturbation which yields the largest magnitude of the correlation coefficient, ρ, is considered to be the actual phase change. From this phase change, the sound speed change and temperature change are computed. The sound speed changes, correlation magnitude

values, and temperature changes each make an image of the ocean floor. The phase ϕ of the complex data $X + iY$, is computed in polar coordinates where $\phi = \arctan \frac{Y}{X}$. The correlation computations depend on both the magnitude and phase of the complex data, and the correlation magnitude coefficients are normalized by the power of the two scans so that $0 \leq \rho \leq 1$. The correlation coefficients and sound speed changes are computed over all 360 pings of two temporally adjacent scans. Each ping is divided into 50 non-overlapping regions, or pixels, where each pixel covers a radial distance of 1 meter by an arc length of 1°. The output image has 360 × 50 pixels, and the range of the image is 0 to 50 meters from the transmitter/receiver.

Given the speed of sound, c, in m/s, and two temporally adjacent scans (X_i, Y_i), $i = 1, 2$, the temperature algorithm works by finding the sound speed change Δc (which is expressed as phase perturbations $\varepsilon(\Delta c, n)$) that maximizes the magnitude of the correlation coefficient ρ:

$$\Delta c(r, b) = \arg \max_{\Delta c} \rho(r, b, \Delta c)$$

for $r \in R$ and $b \in B$. The magnitude of the correlation coefficient ρ is computed as

$$\begin{aligned}
\rho(r, b, \Delta c) &= \frac{\|(X_1, Y_1)(\hat{X}_2, \hat{Y}_2)^*\|_{(r,n)}}{\|(X_1, Y_1)\|_{(r,n)} \|(\hat{X}_2, \hat{Y}_2)\|_{(r,n)}} \\
&= \left(\frac{\left(\sum_{n=n_b}^{n_{b+1}-1} X_1(r,n)\hat{X}_2(r,n) + Y_1(r,n)\hat{Y}_2(r,n) \right)^2}{\left(\sum_{n=n_b}^{n_{b+1}-1} X_1^2(r,n) + Y_1^2(r,n) \right) \left(\sum_{n=n_b}^{n_{b+1}-1} \hat{X}_2^2(r,n) + \hat{Y}_2^2(r,n) \right)} \right. \\
&\quad \left. + \frac{\left(\sum_{n=n_b}^{n_{b+1}-1} X_1(r,n)\hat{Y}_2(r,n) - \hat{X}_2(r,n)Y_1(r,n) \right)^2}{\left(\sum_{n=n_b}^{n_{b+1}-1} X_1^2(r,n) + Y_1^2(r,n) \right) \left(\sum_{n=n_b}^{n_{b+1}-1} \hat{X}_2^2(r,n) + \hat{Y}_2^2(r,n) \right)} \right)^{1/2}
\end{aligned}$$

where $\hat{X}_2(r,n) = X_2(r,n)\cos\varepsilon(\Delta c, n) - Y_2(r,n)\sin\varepsilon(\Delta c, n)$

and $\hat{Y}_2(r,n) = X_2(r,n)\sin\varepsilon(\Delta c, n) + Y_2(r,n)\cos\varepsilon(\Delta c, n)$.

The phase change $\varepsilon(\Delta c, n)$ in a given pixel b is related to sound speed by

$$\varepsilon(\Delta c, n) = (n + 1 - n_b) * \frac{2\pi f}{f_{samp}} \frac{\Delta c}{c},$$

where f = 300 kHz, f_{samp} = 20 kHz, and c = 1490.2 m/s. Note that the phase difference for each time sample n increases linearly with time, but nonlinearly with distance, from the near edge of its pixel. For the first pass of the algorithm, Δc ranges from -2 to 2 in steps of 0.1, which corresponds to a range of sound speed changes of ± 3 m/s [53]. For the second pass of the algorithm, the value of Δc is locally refined to be precise to the hundredth place. Ocean temperature is related to sound speed by the sound speed equation [25, 71] which can be approximated by the following nine term equation:

$$\begin{aligned} c \;=\; & 1448.96 + 4.591T - 5.304 \times 10^{-2}T^2 + 2.374 \times 10^{-4}T^3 \\ & +1.340(S - 35) + 1.630 \times 10^{-2}D + 1.675 \times 10^{-7}D^2 \\ & -1.025 \times 10^{-2}T(S - 35) - 7.139 \times 10^{-13}TD^3, \end{aligned}$$

where T is the temperature (° Celsius), S is the salinity (parts per thousand), and D is the depth (m). A plot of this nine-term sound speed equation is shown in Figure 6.4 where the depth is 20 meters and the salinity is 30.5 parts per thousand, reflecting the local conditions found for the ORCAS experiments. The area of interest is for temperatures of approximately 11.5° Celsius.

Note that for quick look viewing purposes, it would be very convenient if the scattering, correlation, and sound speed data could be computed and sent since they are so much smaller (each image is no larger than 18 kByte) than the original data size. However, previewing and browsing applications should also include the ability to "zoom" into areas of interest. Compression of the entire scan will enable the user to zoom into areas of interest.

6.4 Evaluation of the Effects of Acoustic Data Compression

Since I propose to apply lossy compression techniques to acoustic data, it must be ensured that these techniques produce compressed data with high visual quality and more importantly, give accurate results when analyzed. Similar work has been done in [51] where side scan sonar images are compressed using wavelet packets and a compression method following [33], where only the N largest coefficients (which are scalar quantized) and their positions are preserved. They report recognizable images with compression ratios of up to 108:1. Visual inspection and PSNR are used to measure image quality. In [62], sonar images are compressed using a one-dimensional discrete cosine transform (DCT) performed on small blocks of the data followed by filtering and vector quantization of the coefficients. The impact of compression is evaluated by examining the deflection index of synthetic data. A block-DCT transform has also been adopted in [95] for compression of sonar images from autonomous underwater vehicles to be sent over an acoustic modem. The authors report that line detection is not negatively affected for compression ratios of up to 110:1. A foreground/background approach to sonar image coding is taken in [84] where the foreground image is coded at a high bit rate using a block DCT-like scheme, and the background image is coded by retaining only its local statistics. Compression quality evaluation is done by visual examination of a test image, and quantitatively by the cross ambiguity measure which is the squared magnitude of the cross correlation between the original and compressed data. The authors report that their images are well preserved for compression ratios of up to 25:1. Unfortunately, it is difficult to extend the conclusions of previous work to my experiments as several papers report only the compression algorithm built into the hardware, e. g. [95], or the reported attempts to evaluate the effects of compression are done on such different data that the results do not generalize.

For my experiments I use the MSE as the distortion measure to be minimized in

the design of the vector quantizers and during bit allocation due to its tractability. I evaluate signal noise using the SNR since the data come from a known, stationary source. Since the data are to be computationally analyzed, SNR measurements are unfortunately not a good means to determine acceptability of compression since it is not known how the data will be used in the future. The use of metrics fails in that the uses or applications for which the data will be used are not considered. I am interested in evaluating the compression of acoustic data by evaluating the results of scientific analysis using lossily compressed data. For these compression experiments, I will use MSE and SNR calculations, visual inspection of scattering plots, and measurements of the distortion introduced into sound speed measurements to evaluate the compressed data.

Specifically, I will determine which scattering plots for scan 5 show no visible changes due to compression, as well as those in which the five targets can be identified although the plots are severely degraded due to high compression. This evaluation will be done by visual examination using the 8 bit color palette used by Jackson *et al.* [54], as well as a grayscale palette to view the targets. Image processing techniques, such as histogram equalization, are not used to enhance the backscatter plots, although one would expect that the application of adaptive image processing procedures to the scatter plots would aid target detection, and that the compression ratios reported would improve with the application of these image processing techniques.

Change in the correlation and sound speed plots will be evaluated by comparing the mean absolute difference in value to tolerances defined by [55]. For the correlation measurements, less change will be permissible where high correlation values are expected, and more change will be allowed where low correlation values are expected. Following these guidelines, the original correlation plot is divided into six regions (A through F) based on each correlation value ρ. The tolerances allowed for mean absolute difference in each region are shown in Table 6.3 where there are two sets of tolerances, one for viewing the data, and one that would permit further scientific

Table 6.3: Error tolerances for the mean absolute difference in correlation computations.

Region	Correlation values	Viewing Tolerance	Scientific Tolerance
A	$0.95 < \rho \leq 1$	± 0.01	± 0.005
B	$0.90 < \rho \leq 0.95$	± 0.02	± 0.01
C	$0.80 < \rho \leq 0.90$	± 0.03	± 0.015
D	$0.60 < \rho \leq 0.80$	± 0.05	± 0.025
E	$0.40 < \rho \leq 0.60$	± 0.1	± 0.05
F	$0 \leq \rho \leq 0.40$	± 0.2	± 0.1

use (i. e. temperature calculations). For the sound speed measurements, a tolerance level of 0.2 m/s will be considered acceptable as this corresponds to the errors found in the original application of the temperature algorithm [56].

6.4.1 Experiment 1: Compression of Complex Acoustic Data

For the first set of experiments I compress the raw, complex acoustic data using three methods. The first approach, simple averaging, is a "blind" approach to data compression. I assume that the data are oversampled, so that I can average N consecutive samples to reduce the size of the scan by N. I do this to show that data compression must be approached in an intelligent manner.

As stated previously, it is desired that the MSE be minimized during quantization. Since the real and imaginary parts of the data are statistically independent, the MMSE quantizer requires the minimization of:

$$E[((X,Y) - (\hat{X},\hat{Y}))^2] = E[(X - \hat{X})^2] + E[(Y - \hat{Y})^2].$$

The MSE is minimized when the codeword is chosen to be the centroid of each Voronoi region, i. e. $(\hat{X},\hat{Y}) = (E[X], E[Y])$. The data can be quantized as real vectors using the VQ algorithms described in Chapters 2 and 4. Thus the second approach is to apply PTSVQ directly to the complex data where the vector dimension is two, and

each vector (X, Y) represents a complex number $X + iY$.

The third approach is to apply PTSVQ to the complex data after transformation with the DWT. The real and imaginary components of each ping of the data are independently decomposed by the DWT to seven subbands using the orthonormal basis "S8." Again, the components are independent and may be processed separately. The use of complex wavelets, wavelet packets, and other wavelets filters was examined in [19] and showed no improvement, in terms of energy compaction, over this decomposition. Hence only wavelets/PTSVQ are used for the BAMS data. Seven two-dimensional PTSVQ's, one for each subband, are used to encode the data where each vector represents the corresponding wavelet coefficients from the real and imaginary components of the transformed ping. Each PTSVQ used is created from the process of pruning the larger TSVQ's during bit allocation by GBFOS, as described in Section 4.2.1. The overall compression method to encode one scan consists of transforming the scan's 360 complex pings by the DWT to seven subbands each. The seven PTSVQ's are then applied to each of the seven subbands of each of the 360 pings.

6.4.2 Experiment 2: Compression of Polar Acoustic Data

For the second set of experiments I consider the scientific analysis performed with the data in this approach to data compression. Since the first analysis is the creation of a scattering plot which is obtained from the power of the pressure field, and the second analysis is the creation of correlation and sound speed plots which are obtained from correlating the phase components of two scans, I convert the data from the complex Cartesian domain to the polar domain, magnitude and phase

$$(M, \phi) = (\sqrt{X^2 + Y^2}, \arctan \frac{Y}{X}) = M \exp(i\phi),$$

and power and phase

$$(P, \Phi) \;=\; 20 \log_{10}(M \exp(i\phi)) = 20 \log_{10} M + i\phi 20 \log_{10}(\exp)$$

$$= \quad (20\log_{10} M, \phi 20\log_{10}(\exp)).$$

By converting the data to its magnitude, power and phase components, compression can be optimized to preserve the power and phase of the data which should lead to more accurate scattering plots and perhaps temperature measurements. To see how the compression of the magnitude and power components affects the scattering plots, I have compressed just the magnitude component M and the power component P using PTSVQ and wavelets/PTSVQ. Experimentally, I found that backscatter results are poorer when compressing the magnitude M of the data. Since the scattering plots are computed from the power of the data, it is not surprising that it is better to compress the power of the data. Therefore, I present results for the compression of the data in the polar domain with the power and phase components, (P, Φ), since compression using this data representation promises high compression ratios.

Again, it is desired that the MSE be minimized during quantization. When working in the polar domain and the magnitude and phase parts of the data are statistically independent, it turns out that the MMSE quantizer requires biased conditional mean estimators to minimize

$$E[((M,\phi) - (\hat{M},\hat{\phi}))^2] = E[M^2] + E[\hat{M}^2] - 2E[cos(\phi - \hat{\phi})]E[M\hat{M}],$$

where the choice of the magnitude $\hat{M}$ is dependent on the choice of phase $\hat{\phi}$. An MMSE polar coordinate quantizer with a biased estimator for scalar quantization is presented in [94]. However, as I am quantizing (P, Φ), it happens that one must minimize

$$E[((P,\Phi) - (\hat{P},\hat{\Phi}))^2] = E[(P - \hat{P})^2] + E[(\Phi - \hat{\Phi})^2].$$

The MSE is minimized when the codeword is chosen to be the centroid of each Voronoi region: $(\hat{P}, \hat{\Phi}) = (E[P], E[\Phi])$. Thus, the data can be quantized as real vectors using, without modification, the VQ algorithms described in Chapter 2. For the remainder of this chapter, I refer to (P, Φ) as the polar domain.

I examine two methods to compress the data in the polar domain. The first approach is to apply PTSVQ directly to the data where the vector dimension is two, and each vector (P, Φ) represents the power and phase of the complex number $X + iY$. The second approach is to apply PTSVQ to the power and phase data after transformation with the DWT. The power and phase components of each ping of the data are independently decomposed by the DWT to seven subbands using the orthonormal basis "S8." Seven two-dimensional PTSVQ's, one for each subband, are used to encode the data where each vector represents the corresponding wavelet coefficients from the power and phase components of the transformed ping. Each PTSVQ used is created from the process of pruning larger TSVQ's during bit allocation by GBFOS. The overall compression method to encode one scan consists of transforming the scan's 360 pings of power and phase data by the DWT to seven subbands each. The seven PTSVQ's are then applied to each of the seven subbands of each of the 360 pings.

6.5 Results and Discussion

6.5.1 Effects of Compression on the Raw Data

I first examine the effects of compression in complex Cartesian coordinates over the entire scan. SNR versus entropy results for simple averaging, PTSVQ, and wavelets/PTSVQ for scan 5 is shown in Figure 6.5(a). Wavelets/PTSVQ clearly gives the best results for compression at low bit rates. The SNR tapers off at 5 bits, and is approximately 8 dB less than that of pure PTSVQ at high bit rates. This is due to the smaller set of training data available to design each TSVQ. It does not necessarily mean that the data compressed at higher rates using wavelets/PTSVQ are worse than the data compressed at the same rates using PTSVQ only. What is important is the rapid rise in SNR at very low bit rates. The simple averaging method shows very poor results overall as it degrades quite rapidly.

I next examine the effects of compression in the polar domain (P, Φ). SNR versus entropy in the polar domain is shown in Figure 6.5(b) for wavelets/PTSVQ and PTSVQ. While results for PTSVQ show a rapid and steady rise in SNR for all rates, wavelets/PTSVQ shows an improvement in SNR over PTSVQ below 9 bits. The same SNR plots, when computed in the complex Cartesian domain, Figure 6.5(c), show severe degradation in SNR below 4 bits per complex value for wavelets/PTSVQ and 6 bits for PTSVQ, and seen in this light does not outperform the previous methods examined at low bit rates except for the simple averaging approach.

6.5.2 Effects of Compression on an Individual Ping

I next examine the effects of compression on individual pings of data, specifically pings 42 and 84. While it is expected that the mean of the compressed data will differ insignificantly from the mean of the original data, it is also expected that the standard deviation of the compressed data will be less than that of the original data. In general, the more a signal varies statistically from the training data signal, the more difficult it is to code; this should be reflected in the coding results for ping 84 since it contains a target. I will first examine the change in standard deviation due to compression. Then, I will discuss the effects of each compression experiment on the real, imaginary, power, and phase components of the ping plus the target present in ping 84.

The errors of the standard deviation versus entropy for pings 42 and 84 are shown in Figures 6.6 and 6.7, respectively. As expected, the simple averaging method performs the worst of all the methods; the standard deviation has a very large error (above 10 percent) for all entropies below 10 bits. For ping 42, the standard deviation has low mean errors (within 10 percent) for both PTSVQ and wavelets/PTSVQ in the complex Cartesian domain for entropies above 1.5 bits. In the polar domain, results are similar to the results in the complex Cartesian domain except that the curves do not drop as fast. Low errors (below 10 percent) are found for entropies

above 2.0 bits for wavelets/PTSVQ and above 3.0 bits for PTSVQ. When ping 84 is coded, PTSVQ in both coordinate domains behaves similarly as when ping 42 is coded. The main difference is found when wavelets/PTSVQ is used to code ping 84 in the complex Cartesian domain, where the average error never drops below 9 percent although the error drops to that value by 2.0 bits. In the polar domain, the error is less than 10 percent by 2.0 bits, and it eventually settles to 1.5 percent at 8.0 bits. These larger error values are indicative of the difficulty in coding a ping with a target in it.

I next describe the effects of compression on ping 84 as the compression ratio increases. The quick decay seen in the standard deviation made by simple averaging is seen by a rapid decrease in magnitude throughout the entire data set and is already evident in Figure 6.8 which results from averaging 4 data samples at a time. In addition, one can see that noise is introduced into the reference target, as identified by the power spectrum in Figure 6.8(b), which will affect target detection. Due to the averaging, the high frequency components of the data are lost throughout the entire ping. Averaging of the data will directly affect the quality of the scattering and temperature measurements because there is less variation in the compressed data as well. There will also be more difficulty in target detection due to the loss of resolution.

In general, PTSVQ in the complex Cartesian domain tends to preserve data with large magnitudes and to set data with small magnitudes to a limited number of values. As the bit rate decreases, fewer bits are used for the high probability, low magnitude data which results in very poor coding of the last 1300 values of the data. This causes a complete loss of the power data and of the phase data which will cause high, incorrect, correlation values to be found in the outer half of the data, while forcing low power spectrum values in the outer half of the data. This progression of decay can be seen in Figures 6.9 and 6.10 in which the data are compressed to 6.48 bits and 3.56 bits per complex number respectively. As the bit rate decreases, the limited number of low magnitude codewords affects the phase and magnitude

components first, Figure 6.9(b). By 3.56 bits, PTSVQ is unable to preserve any of the small magnitude data, Figure 6.10(b).

Two versions of ping 84 compressed by wavelets/PTSVQ in the complex Cartesian domain at 4.39 and 2.25 bits per complex number are shown in Figure 6.11 and Figure 6.12 respectively. Wavelets/PTSVQ does an excellent job compressing the data that have a high magnitude combined with a strong frequency component, Figure 6.11(a). As the bit rate decreases, fewer bits are used to code the low magnitude wavelet coefficients, while relatively more bits are used to code the high magnitude wavelet coefficients. Since there is more space/frequency activity in the first half of the data (Figures 6.3(b) and 6.3(c)), coding by wavelets/PTSVQ results in better preservation of the target and the data surrounding it. Note that when the second half of the signal is completely lost, there is much more degradation around the target when using only PTSVQ (Figure 6.10(b)) than when using wavelets/PTSVQ (Figure 6.12(b)). The poor coding of the last half of the ping causes a complete loss of power and phase data which will cause incorrect correlation values to be found in the outer half of the data, while incorrect, low values of the power spectrum will be found in the outer half of the data. The target cannot be easily distinguished from noise below 0.1 bits per complex number.

Two versions of ping 84 coded at 7.82 and 6.82 bits per complex number by PTSVQ in the polar domain are shown in Figures 6.13 and 6.14. PTSVQ in the polar domain does not yield good compression of ping 84. The power of the target starts to attenuates rapidly below 8 bits per complex number, while the background signal tends to be better preserved. The target cannot be easily distinguished from noise below 6.82 bits per complex number.

Ping 84 coded at 2.67 and 0.69 bits per complex number by wavelets/PTSVQ in the polar domain is shown in Figures 6.15 and 6.16. Wavelets/PTSVQ does an excellent job encoding the data, although it reduces the power of the target in the encoded ping 84 by approximately two, at even the highest rates. This reduction

of power is not a problem for ping 42 which does not contain a target. As the bit rate decreases, the signal gradually degrades. The magnitude of the reference target begins to degrade below 5.0 bits, but the background degrades as well. The reference target can be seen in ping 84 at 0.1 bits and above. The signal is very poorly coded below 0.1 bits where the signal content includes only very low frequency components. As the rate increases from zero the signal looks much more like the original signal, and the improvement is graceful, without any notably abrupt jumps in quality. The quality of the coded signal at low bit rates can be seen in Figure 6.16(b) at 0.69 bits where the lowpass features of the signal are retained. The ping at rate 2.67 is shown in Figure 6.15(b) where the signal includes high frequency components. As the bit rate decreases, the gradual degradation of the signal will lead to a gradual degradation of the scattering plots and correlation plots as the high frequency details are lost.

6.5.3 Effects of Compression on the Backscatter Plots

I now examine the effects of the different compression schemes on the backscatter plots. The original backscatter plot for scan 5 is shown in Figure 6.17. Evaluation of visual preservation is done using a very high-contrast 8 bit color palette, so expected compression ratios for visual preservation are low. To avoid dramatic color changes when backscatter values change due to compression, a grayscale palette is used to view targets.

The best visual preservation of the backscatter plots of scan 5 is done by wavelets/PTSVQ in the polar domain, Table 6.4. It creates scattering plots which are visually unchanged at 3.08 bits per complex number which corresponds to a compression ratio of 5.77:1 and a drop of 24.43 dB in SNR. As the bit rate decreases, image degradation can be seen throughout the entire plot, and at high compression ratios (below 0.6 bits) striped rings appear. Note that all of the targets are visible at compression ratios below and including 126.07:1 (0.14 bits), Figure 6.18. The targets tend to diminish and disappear (especially the reference target and cylinder B) as

Table 6.4: Results of compression on scattering plots of acoustic scan 5.

Scattering Plots	Visually Lossless CR	Drop in SNR (dB)	Targets Detectable CR
Exp. 1: Simple Averaging	0	0	12.29
Exp. 1: PTSVQ	2.74	3.86	37.49
Exp. 1: Wavelets/PTSVQ	2.97	0.81	130.03
Exp. 2: PTSVQ	2.61	34.11	3.03
Exp. 2: Wavelets/PTSVQ	5.77	24.33	126.07

the compression ratio increases above 126:1.

Surprisingly, since compression in the polar domain is expected to produce superior results, compression by PTSVQ in the polar domain produces the worst backscatter plots for target detection. Although the scattering plots are visually unchanged at 6.82 bits per complex number which corresponds to a compression ratio of 2.61:1 and a drop of 34.11 dB in SNR, as the bit rate decreases, image degradation, which can be seen throughout the entire plot, progresses very rapidly. There is a distinct decrease of signal in the innermost parts of the plot, and a distinct increase of signal in the outermost parts of the plot. However, the targets are still visible at 5.86 bits per complex number, as seen in Figure 6.19, which gives a compression ratio of 3.03:1. The entire image is completely degraded at a compression ratio of 4.53:1 (3.93 bits); no targets can be made out at this or lower bit rates.

When compressing in the complex Cartesian domain, wavelets/PTSVQ produces the best results. It creates scattering plots which are visually unchanged at 5.98 bits per complex number which corresponds to a compression ratio of 2.97:1 and a drop in SNR of 0.81 dB. Degradation begins in the outer edges of the plot, and spreads to the center. The outer third of the image is completely degraded at 2.25 bits; the central third is degraded at 0.87 bits; and the inner third of the image has degraded at 0.14 bits per complex number, Figure 6.20, which is a compression ratio of 130.03:1.

Note that all of the targets are clearly visible at 0.14 bits and above. Below 0.14 bits, the targets, while still visible, become very dim and small, especially unburied sphere B. In addition, all of the background detail is gone.

As expected, the overall worst results arise by simple averaging in the complex Cartesian domain which creates scattering plots with notable visual degradation even when averaging only two samples together to produce a compression ratio of 1.83:1 (9.69 bits). Degradation of the scattering plots continues rapidly as the compression ratio increases and distortion is seen throughout the entire image. The image looks entirely degraded at a compression ratio of 10.81:1 (1.64 bits, size 14 sample average) although the targets are still visible at this and higher compression ratios, such as Figure 6.21 which has a compression ratio of 12.29:1 resulting from averaging 16 values at a time.

Compression by PTSVQ in the complex Cartesian domain creates scattering plots which are visually unchanged at 6.48 bits per complex number, a compression ratio of 2.74:1, which corresponds to a 3.86 dB drop in SNR. As the bit rate decreases, image degradation begins in the outer edges of the plot and spreads to the center. The outer one-third of the image is completely degraded at 3.5 bits; the central one-third is degraded at 2.1 bits; and the inner one-third of the image has degraded at 0.47 bits per complex number, Figure 6.22, a compression ratio of 37.49:1. It should be noted that all of the targets are visible at all compression ratios below and including 37.49:1. This is the highest compression ratio possible for a dimension of two.

6.5.4 Effects of Compression on the Correlation and Sound Speed Measurements

I next describe the experiments designed to measure the effects of compression on the second scientific application, sound speed measurements. Ideally, the best sound speed measurements using the 300 kHz sonar are found when the scans are sampled 7 or 8 minutes apart in time so that small phase changes can be accurately detected. Unfortunately, the 300 kHz sonar on BAMS is unable to take a scan with an interval

of less than 35 minutes. Furthermore, the scans taken during the ORCAS experiment have time lapses of at least 3 hours, and my two test scans (5 and 6) are separated by 11 hours and 36 minutes. Patrick Ahearn, a Master's student in Oceanography, is currently developing an accurate temperature algorithm for the 300 kHz data. An accelerated BAMS (XBAMS) system has been developed which is able to complete a scan in 6 minutes providing good high frequency data that will create useful correlation and sound speed plots. However, I did not have access to XBAMS data at the time that this experiment was done. Initial experiments with the BAMS test data give poor results, and the data and method are at this time considered suspect and perhaps of little value. Therefore, to conduct my experiments, it is necessary to create a synthetic data set that approximates a scan that would contain valid sound speed, and hence, temperature information.

I have tried to create a reasonable synthetic data scan using several methods. The first set of synthetic images I created was made by linear combinations of scans 5 and 6: λ scan 5 + $(1-\lambda)$ scan 6 where $0 \leq \lambda \leq 1$. The second set of synthetic images I created was made by adding noise to scan 5 in various ways such as adding zero-mean Gaussian noise to scan 5 with different variances from 0.3 to 20, or by thresholding scan 5 and then adding Gaussian noise, and finally by adding Gaussian noise to scan 5 with variances that were determined by the local variance of the data as measured by distance from the sonar platform. Unfortunately, neither of these methods produced a good synthetic image. In the first case, the scans are so far distant in time, that it is not possible to make a linear combination that approximates a scan that is 7 minutes from either of the scans. In the second case, the addition of Gaussian noise does not take into account the physical sound speed response one expects as the ocean temperature changes. In fact, it is not clear what a good synthetic scan should look like.

Therefore, I finally created a synthetic scan using the sound speed model itself. By assigning different sound speed changes across the ocean floor, I determined the

phase changes necessary to induce those particular changes, and added those phase changes to scan 5. Finally I added a small amount of zero-mean Gaussian noise to the synthetic scan to make it more realistic. The synthetic scan was created assuming there was a uniform sweep of sound speed change of ± 3 m/s over the entire 162 meter diameter, although only the inner 100 meters are examined. Thus the amount of sound speed change in the pings will range from no change to changes of up to 3 m/s across a ping. Noise was added to the magnitude and phase of the synthetic data as follows: zero-mean Gaussian noise with a variance of ten percent of the variance of the magnitude as a function of distance from the sonar (where this function was smoothed to avoid undue influence of the targets) was added to the magnitude of the synthetic signal, and zero-mean Gaussian noise with a variance of five percent of the variance of the phase was added to the phase of the synthetic signal. The noise was added in the above manner since the variance of the magnitude is, when ignoring targets, a decreasing function with respect to increasing distance from the sonar, while the variance of the phase of the signal does not vary with respect to the distance from the sonar. The resulting correlation and sound speed plots, with the added noise, are shown in Figures 6.23(a) and 6.23(b). Using scan 5 and this synthetic scan, I will be able to measure the effects of compression on the sound speed measurements.

I next report the effects of compression on the correlation and sound speed measurements using scan 5 and the synthetic scan to create correlation and sound speed plots. The mean absolute differences in the correlation plots computed from compressed versions of scan 5 and the synthetic scan, for regions A, B, and C, are shown in Figure 6.24. Acceptable tolerances are specified in Table 6.3. Results are summarized in Table 6.5.

Overall, the results are disappointing as the compression ratios are all very low and pretty much the same for the different compression methods. The best results for the correlation measurements are found in the complex Cartesian domain with

Table 6.5: Acoustic correlation measurement results due to compression.

Correlation Coefficient	Largest CR (Viewing Tolerance)	Largest CR (Scientific Tolerance)
Exp. 1: Simple Averaging	0	0
Exp. 1: PTSVQ	2.11	1.93
Exp. 1: Wavelets/PTSVQ	3.31	2.04
Exp. 2: PTSVQ	2.02	1.63
Exp. 2: Wavelets/PTSVQ	2.06	1.76

wavelets/PTSVQ which preserves the correlation plots within the viewing tolerances at compression ratios of 3.31:1 and below, and within the scientific tolerances at compression ratios of 2.04:1 and below. Excluding simple averaging which cannot preserve the data within the desired tolerances, the worst results arise when compressing with PTSVQ in the polar domain. The data are not preserved for compression ratios of 1.63:1 and higher, so that the correlation results remain within the specified tolerances. PTSVQ is able to give some compression with better results when applied to the data in the complex Cartesian domain, where it is able to preserve the correlation plots with good accuracy at compression ratios of 2.11:1 and 1.93:1 and below for viewing and scientific tolerances, respectively. Wavelets/PTSVQ applied in the polar domain (P, Φ) can preserve the correlation plots with good accuracy at compression ratios of 2.06:1 and 1.76:1 and below. These results do indicate, as seen previously, that wavelets/PTSVQ outperforms PTSVQ regardless of the data domain.

When visually examining the sound speed plots, it is seen, for the simple averaging case, that rings and spots are introduced at all compression ratios, even when averaging only 2 samples at a time. The plots rapidly degrade throughout the entire image as the step size increases. However, the uniform sound speed trend can be seen to 1.89 bits per complex value. For the other compression methods in the complex Cartesian domain, it is found that for both PTSVQ and wavelets/PTSVQ, the sound

Table 6.6: Acoustic sound speed measurement results due to compression.

Sound Speed	Largest CR (Tolerance = 0.2)	Drop in SNR (dB)
Exp. 1: Simple Averaging	1.83	0
Exp. 1: PTSVQ	3.87	9.05
Exp. 1: Wavelets/PTSVQ	4.81	3.58
Exp. 2: PTSVQ	3.07	37.31
Exp. 2: Wavelets/PTSVQ	3.38	18.04

speed plots degrade from the outer edges to the center, preserving the inner third so that the uniformly changing sound speed trend can be seen to 1.02 bits for PTSVQ and 0.68 bits for wavelets/PTSVQ. For the experiments in the polar domain, it is seen that for both PTSVQ and wavelets/PTSVQ, the sound speed plots degrade throughout the entire image. For PTSVQ, the uniformly changing sound speed trend can be seen for 3.38 bits per complex value and higher, while for wavelets/PTSVQ, the trend can be seen to 1.26 bits per complex value.

The mean absolute differences in the sound speed plots computed from compressed versions of scan 5 and the synthetic scan, as well as the associated drop in SNR, are shown in Figure 6.25. An acceptable tolerance was defined in Section 6.4 to be 0.2 to correspond with errors found in the original temperature application. Compression results, summarized in Table 6.6, are more promising for these sound speed measurements. Compression by wavelets/PTSVQ in the complex Cartesian domain shows that a compression ratio of 4.81:1 yields acceptable results. As expected, simple averaging yields the worst results, while wavelets/PTSVQ outperforms PTSVQ in both of the data domains.

6.6 Conclusions and Future Work

In this chapter, the benefit of compression on acoustic data used for target detection and ocean floor temperature studies was examined. The compression schemes examined were simple averaging, PTSVQ, and wavelets/PTSVQ in both the complex Cartesian and polar domains. Experimental results showed that wavelets/PTSVQ consistently outperforms PTSVQ in terms of SNR, backscatter plot quality, correlation and sound speed measurements. The improvement in performance is clearly worth the added complexity of using adaptive transform coding to compress the signal. Experimental results also show that it is not viable to use the simple approach of averaging for compression.

The decision of the domain in which compression should be done is a function of the application, as compression results vary from application to application. For example, for the scatter plots, compression of only the power of the signal, without phase preservation, leads to better target preservation since phase information has no value for backscatter calculations. Better compression results also arise when examining backscatter plots since these calculations are more robust as they represent an average value for each bin. This is evident from experiments which show that compression of over 100:1 of both phase and magnitude components of the data yields data that can be used for target detection. Thus, when examining only this application, one can expect to be able to get higher compression ratios with a better tailored compression algorithm. This is ideal for quick look and browse applications. However applications such as correlation and sound speed measurements are sensitive to the smallest changes in data. One can expect to be able to use lossy compression on the data with acceptable accuracy for compression ratios of perhaps no more than 4:1.

I conclude that the use of only one approach for the compression of acoustic data will not provide reasonable results when a variety of different scientific analyses are

to be applied to the data. Rather, compression should be tailored to the application or analysis for which the data will be used whether it be for previewing and browsing purposes, or for scientific analysis. I also conclude that, although SNR versus entropy measurements are a popular and easy way of measuring the success of compression experiments, they are not good predictors of compression performance for scientific data. Other measures are needed to predict the viability of different compression schemes.

Future work could involve the development of a hybrid measure that can be optimized to produce compressed data that can be used for several applications at once. For example a two application measure could take the form

$$D = D_A + \lambda D_B$$

where λ would control the weight given to application A and application B. This measure could be used during the VQ and bit allocation procedures to produce compressed data that would be useful for the chosen applications. A more detailed study of the data involving the higher resolutions available with matched filtering, and the use of XBAMS data with a temperature measurement analysis suitable for this data would provide more insight to a hybrid-measure compression approach. Unfortunately, at the time of these experiments, the new XBAMS data had yet to be examined, and an accurate temperature algorithm for the 300 kHz data was still under development.

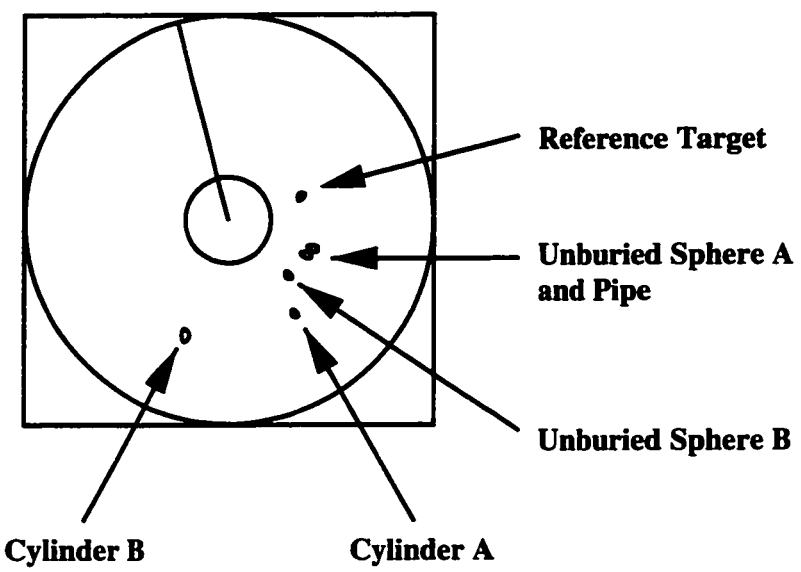
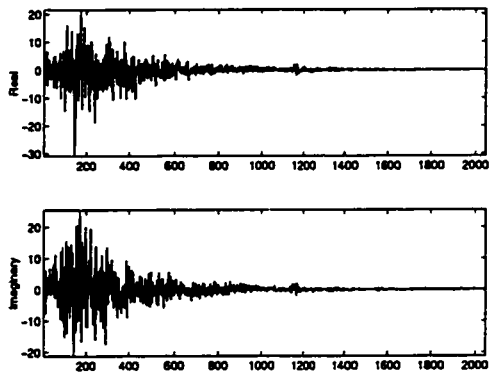
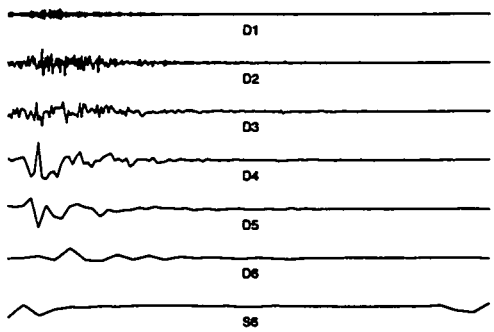


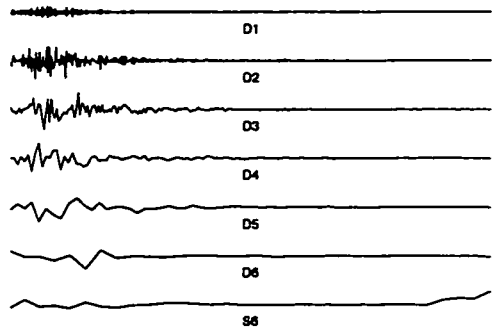
Figure 6.1: The five targets present in the acoustic test scans.



(a) Scan 5, ping 42. Real and imaginary components.

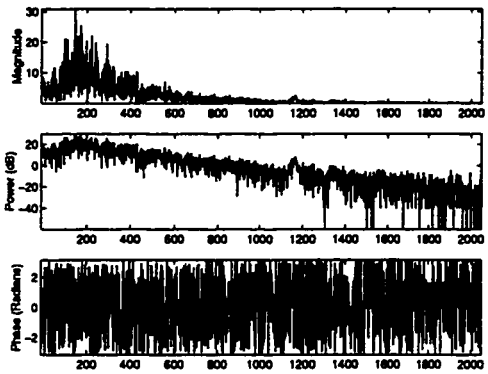


(b) Scan 5, ping 42. DWT of the real component.

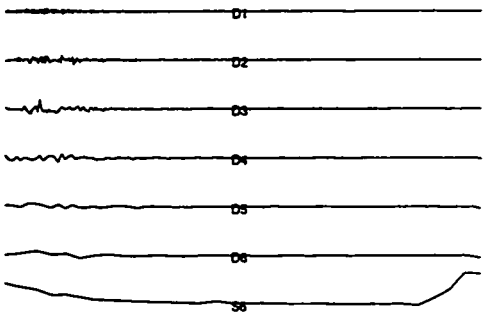


(c) Scan 5, ping 42. DWT of the imaginary component.

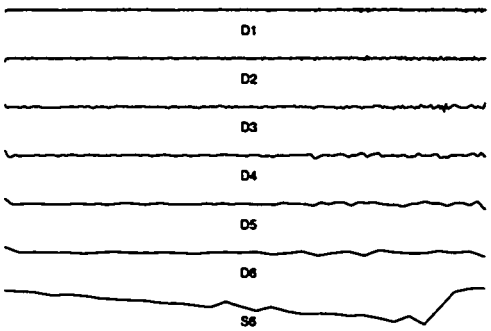
Figure 6.2: Scan 5, ping 42, (a) the real and imaginary components, (b) the DWT of the real component, and (c) the DWT of the imaginary component.



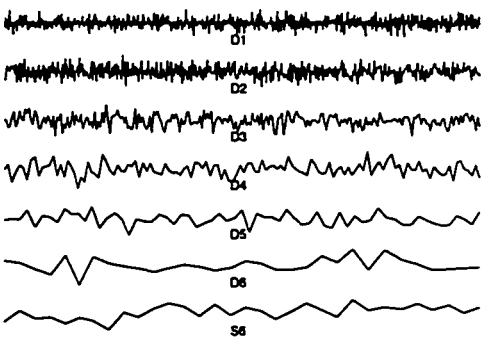
(d) Scan 5, ping 42. Magnitude, power, and phase components.



(e) Scan 5, ping 42. DWT of the magnitude component.

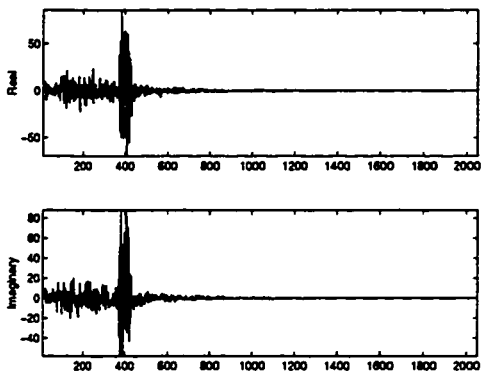


(f) Scan 5, ping 42. DWT of the power component.

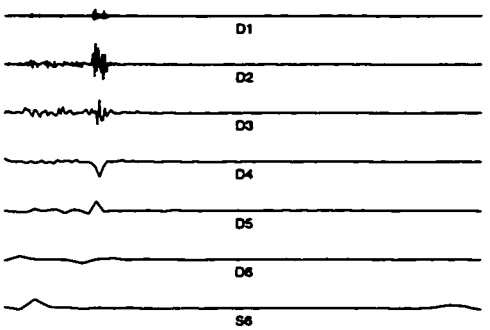


(g) Scan 5, ping 42. DWT of the phase component.

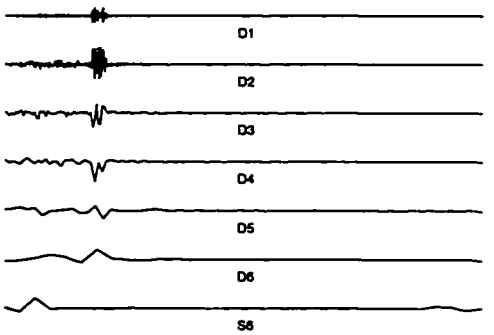
Figure 6.2: *(Continued).* Scan 5, ping 42, (d) the magnitude, power and phase components, (e) the DWT of the magnitude component, (f) the DWT of the power component, and (g) the DWT of the phase component.



(a) Scan 5, ping 84. Real and imaginary components.

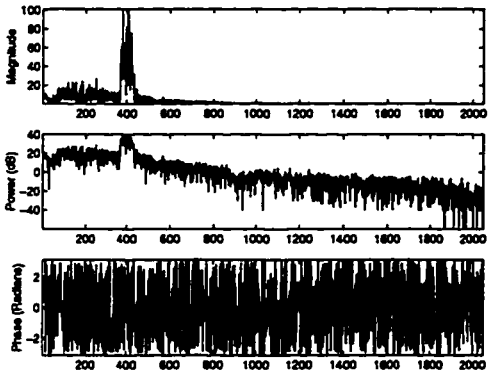


(b) Scan 5, ping 84. DWT of the real component.

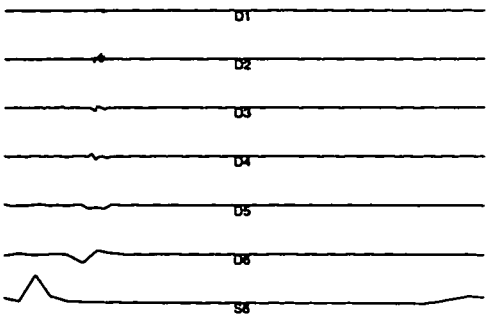


(c) Scan 5, ping 84. DWT of the imaginary component.

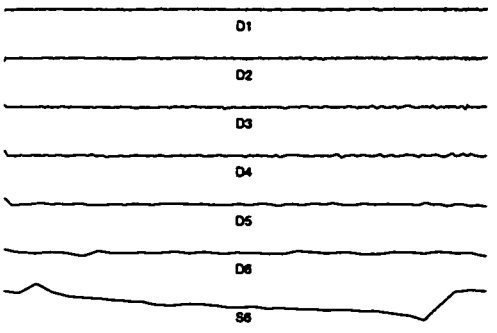
Figure 6.3: Scan 5, ping 84 (which contains a portion of the reference target) (a) the real and imaginary components, (b) the DWT of the real component, and (c) the DWT of the imaginary component.



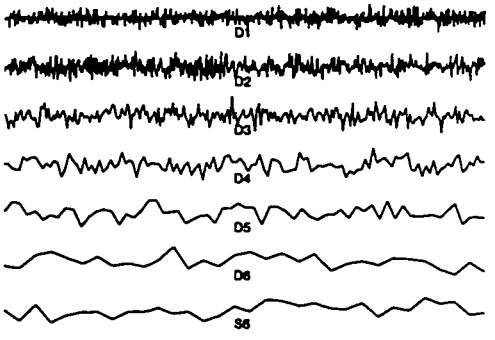
(d) Scan 5, ping 84. Magnitude, power, and phase components.



(e) Scan 5, ping 84. DWT of the magnitude component.



(f) Scan 5, ping 84. DWT of the power component.



(g) Scan 5, ping 84. DWT of the phase component.

Figure 6.3: *(Continued).* Scan 5, ping 84 (which contains a portion of the reference target) (d) the magnitude, power and phase components, (e) the DWT of the magnitude component, (f) the DWT of the power component, and (g) the DWT of the phase component.

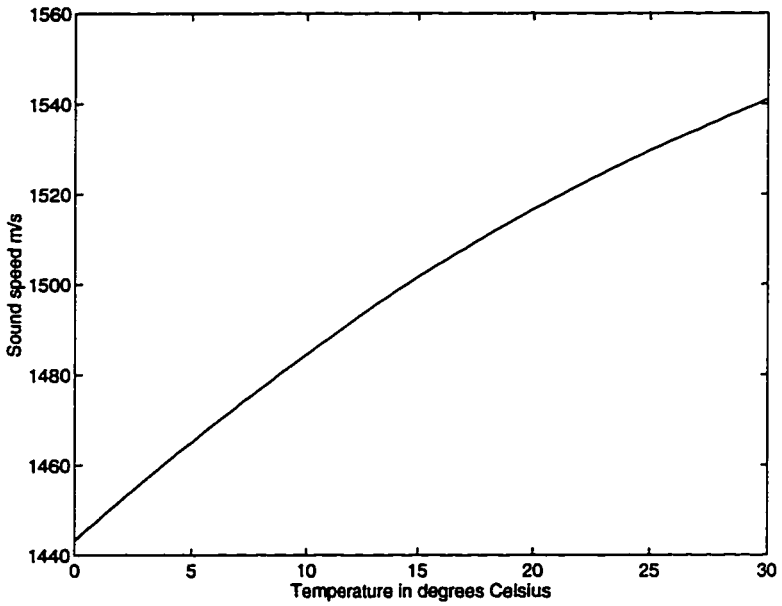
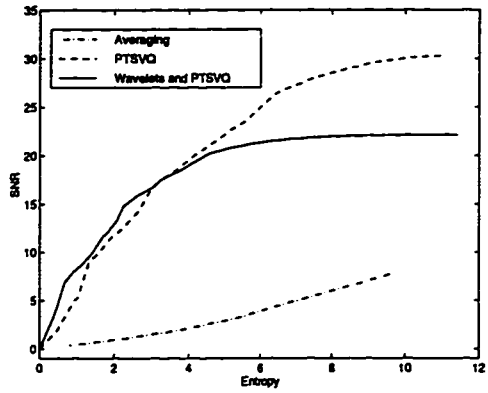
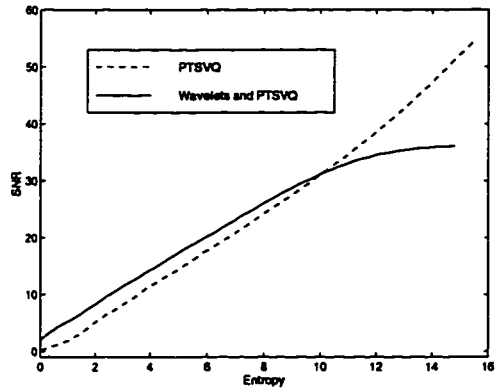


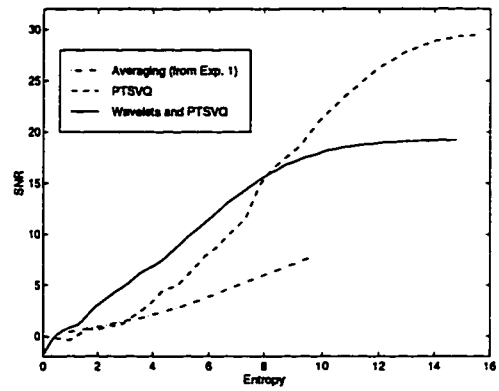
Figure 6.4: Sound speed as a function of temperature at a depth of 20 meters and salinity of 30.5 parts per thousand. The area of interest is for temperatures of approximately 11.5° Celsius.



(a) Experiment 1, scan 5 (X, Y).

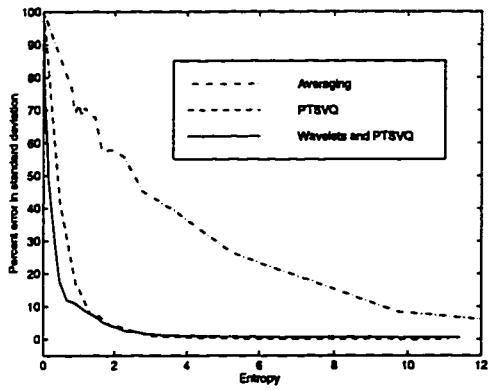


(b) Experiment 2, scan 5 (P, Φ).

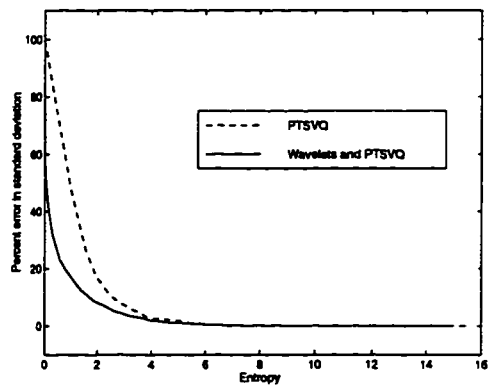


(c) Experiment 2, scan 5 (X, Y).

Figure 6.5: SNR versus entropy for acoustic scan 5. Experiment 1 is conducted in the complex Cartesian domain (X, Y), and Experiment 2 is conducted in the polar domain (P, Φ). (a) Experiment 1. (b) Experiment 2, computed in the polar domain, (P, Φ). (c) Experiment 2, computed in the complex Cartesian domain (X, Y) and compared to simple averaging.

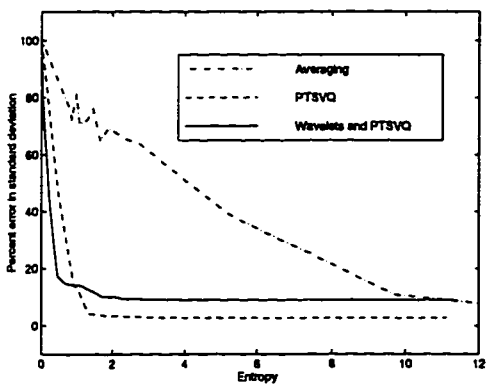


(a) Experiment 1, ping 42 of scan 5.

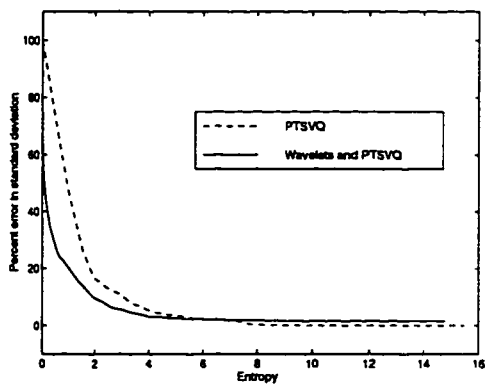


(b) Experiment 2, ping 42 of scan 5.

Figure 6.6: The average decrease in standard deviation versus entropy for ping 42 of scan 5: (a) Experiment 1 and (b) Experiment 2.

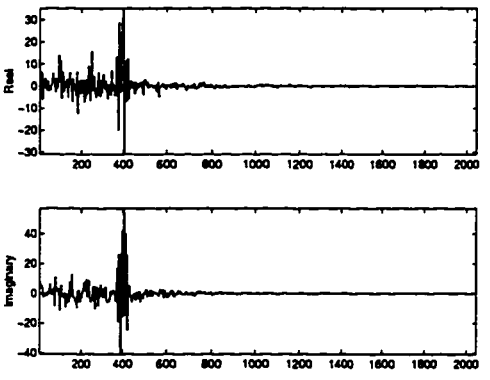


(a) Experiment 1, ping 84 of scan 5.

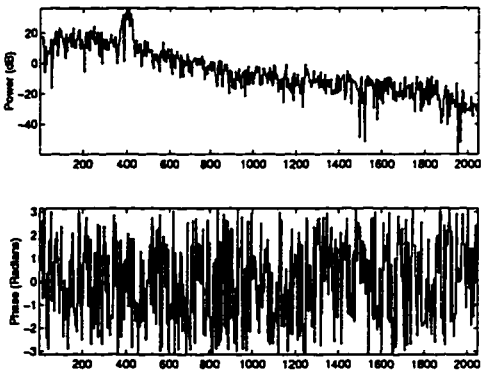


(b) Experiment 2, ping 84 of scan 5.

Figure 6.7: The average decrease in standard deviation versus entropy for ping 84 of scan 5 which contains a portion of the reference target. (a) Experiment 1 and (b) Experiment 2.

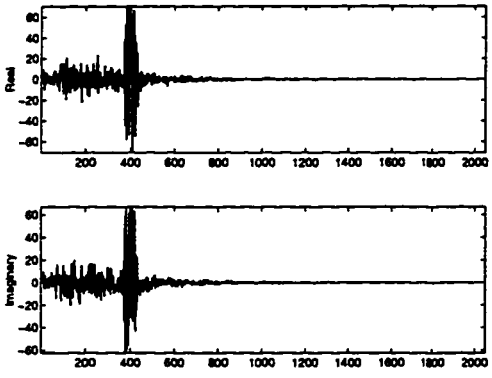


(a) Scan 5, ping 84. Real and imaginary components.

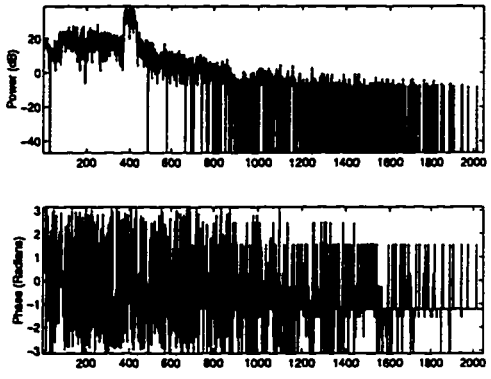


(b) Scan 5, ping 84. Magnitude, power and phase components.

Figure 6.8: Experiment 1: The compressed version of scan 5, ping 84 at 2.62 bits yielded by averaging 4 samples together: (a) the real and imaginary components, and (b) the magnitude, power, and phase components.

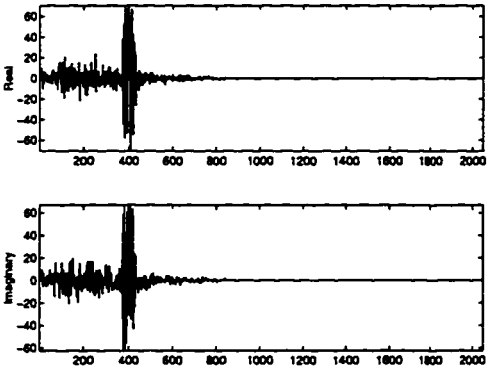


(a) Scan 5, ping 84. Real and imaginary components.

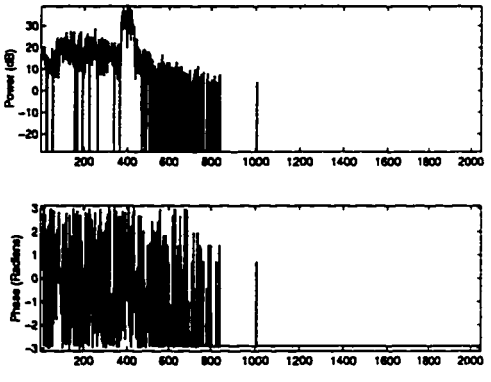


(b) Scan 5, ping 84. Magnitude, power, and phase components.

Figure 6.9: Experiment 1: The compressed version of scan 5, ping 84 at 6.48 bits using PTSVQ in the complex Cartesian domain: (a) the real and imaginary components, and (b) the magnitude, power, and phase components.

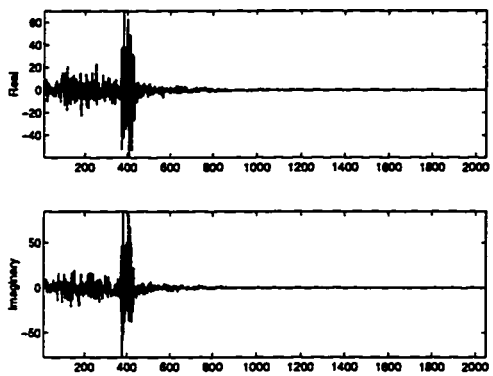


(a) Scan 5, ping 84. Real and imaginary components.

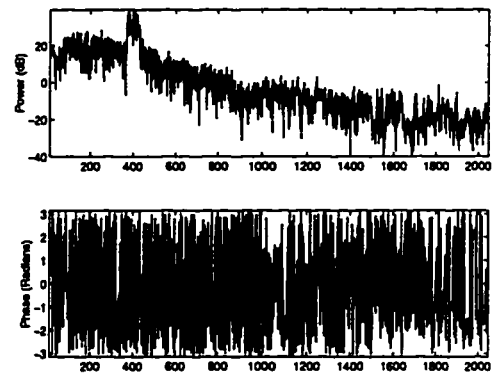


(b) Scan 5, ping 84. Magnitude, power, and phase components.

Figure 6.10: Experiment 1: The compressed version of scan 5, ping 84 at 3.56 bits using PTSVQ in the complex Cartesian domain: (a) the real and imaginary components, and (b) the magnitude, power, and phase components.

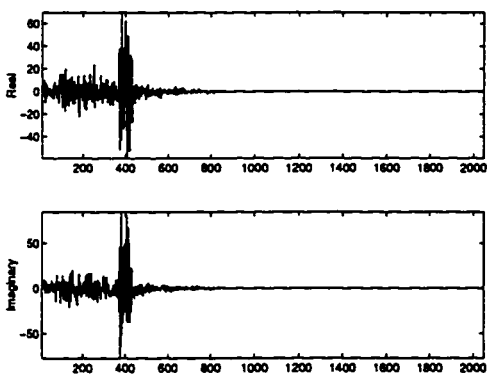


(a) Scan 5, ping 84. Real and imaginary components.

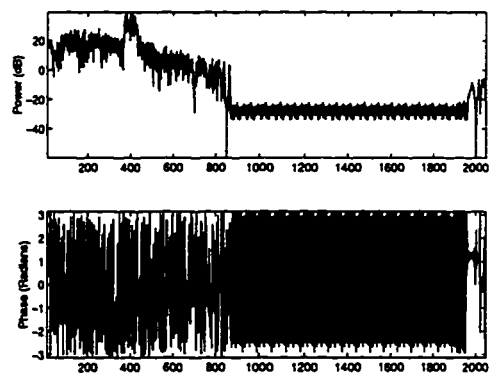


(b) Scan 5, ping 84. Magnitude, power, and phase components.

Figure 6.11: Experiment 1: The compressed version of scan 5, ping 84 at 4.39 bits using wavelets/PTSVQ in the complex Cartesian domain: (a) the real and imaginary components, and (b) the magnitude, power, and phase components.

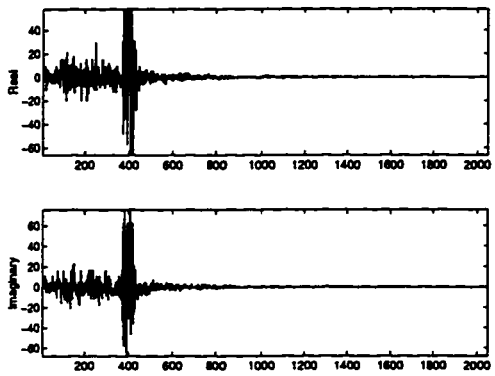


(a) Scan 5, ping 84. Real and imaginary components.

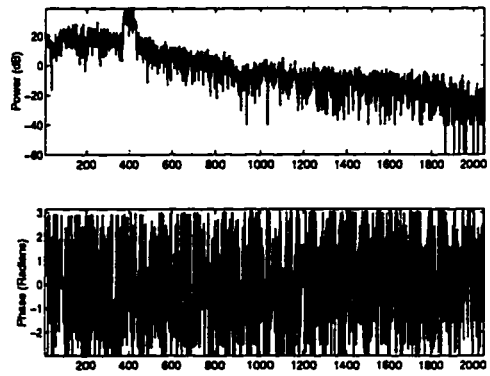


(b) Scan 5, ping 84. Magnitude, power, and phase components.

Figure 6.12: Experiment 1: The compressed version of scan 5, ping 84 at 2.25 bits using wavelets/PTSVQ in the complex Cartesian domain: (a) the real and imaginary components, and (b) the magnitude, power, and phase components.

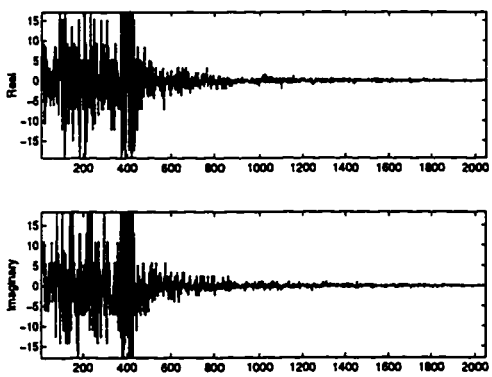


(a) Scan 5, ping 84. Real and imaginary components.

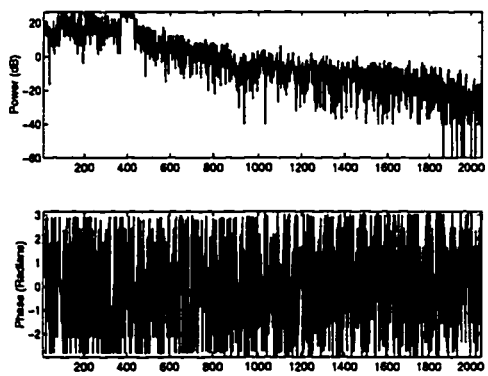


(b) Scan 5, ping 84. Magnitude, power, and phase components.

Figure 6.13: Experiment 2: The compressed version of scan 5, ping 84 at 7.82 bits using PTSVQ in the polar domain: (a) the real and imaginary components, and (b) the magnitude, power, and phase components.

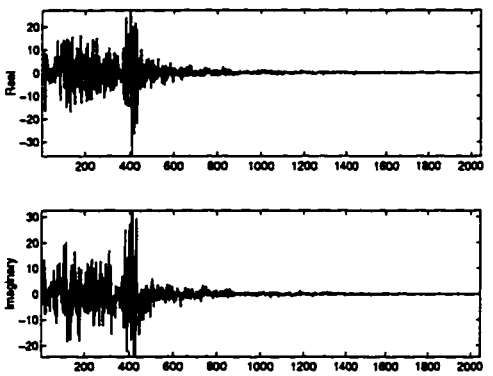


(a) Scan 5, ping 84. Real and imaginary components.

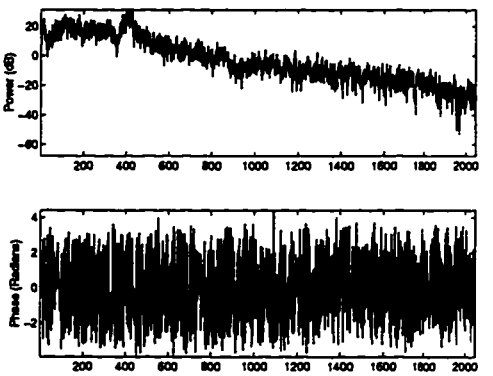


(b) Scan 5, ping 84. Magnitude, power, and phase components.

Figure 6.14: Experiment 2: The compressed version of scan 5, ping 84 at 6.82 bits using PTSVQ in the polar domain: (a) the real and imaginary components, and (b) the magnitude, power, and phase components.

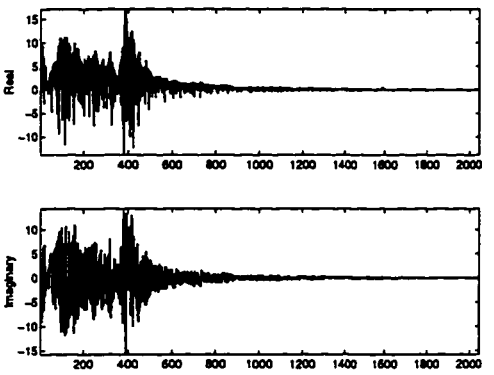


(a) Scan 5, ping 84. Real and imaginary components.

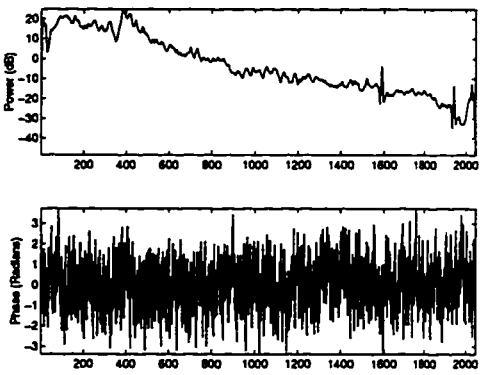


(b) Scan 5, ping 84. Magnitude, power, and phase components.

Figure 6.15: Experiment 2: The compressed version of scan 5, ping 84 at 2.67 bits using wavelets/PTSVQ in the polar domain: (a) the real and imaginary components, and (b) the magnitude, power, and phase components.



(a) Scan 5, ping 84. Real and imaginary components.



(b) Scan 5, ping 84. Magnitude, power, and phase components.

Figure 6.16: Experiment 2: The compressed version of scan 5, ping 84 at 0.69 bits using wavelets/PTSVQ in the polar domain: (a) the real and imaginary components, and (b) the magnitude, power, and phase components.

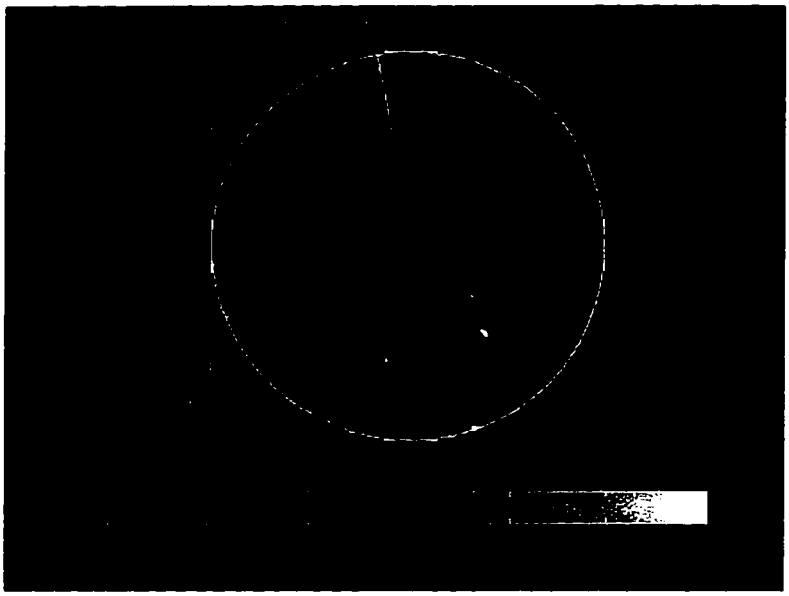


Figure 6.17: The original scattering plot of scan 5 computed from the power of the pressure field.

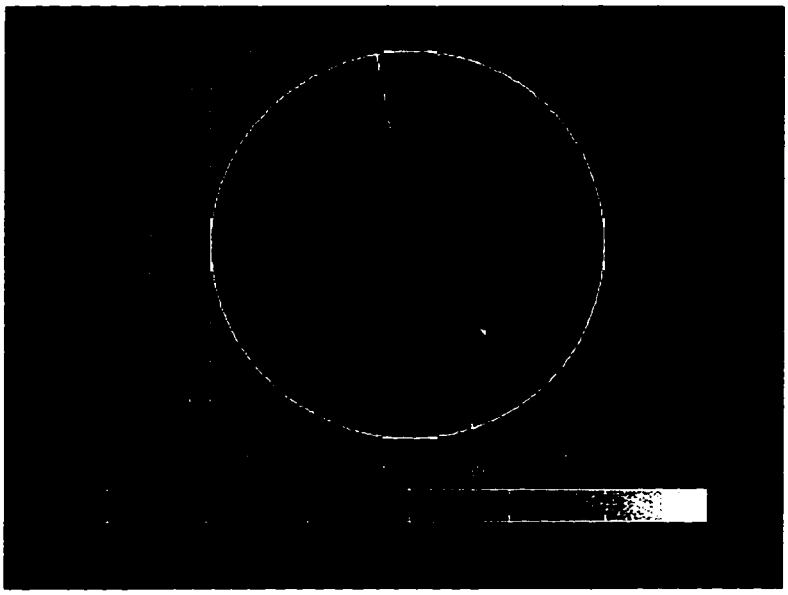


Figure 6.18: Experiment 2: The scattering plot for scan 5 created by using wavelets/PTSVQ in the polar domain (P, Φ). The entropy is 0.1410 bits per complex number, and the compression ratio is 126.07:1.

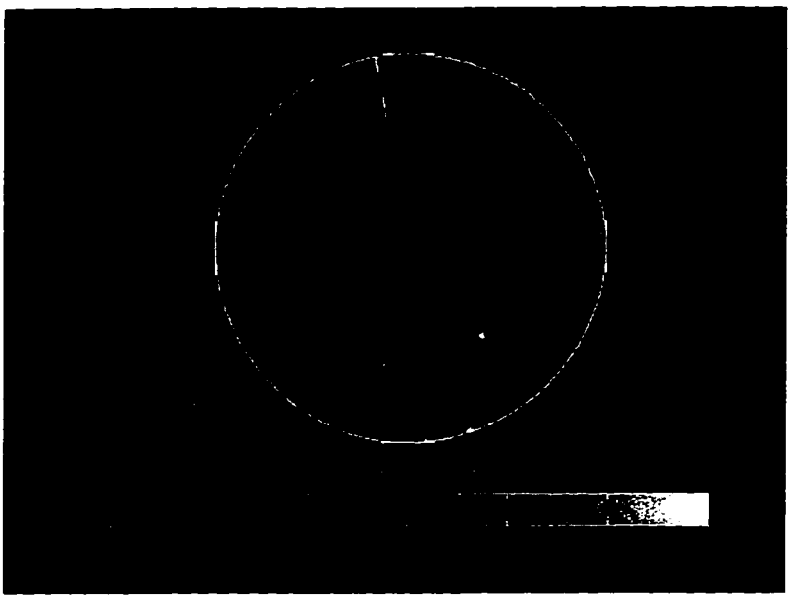


Figure 6.19: Experiment 2: The scattering plot for scan 5 created by using PTSVQ in the polar domain (P, Φ). The entropy is 5.86 bits per complex number, and the compression ratio is 3.03:1.

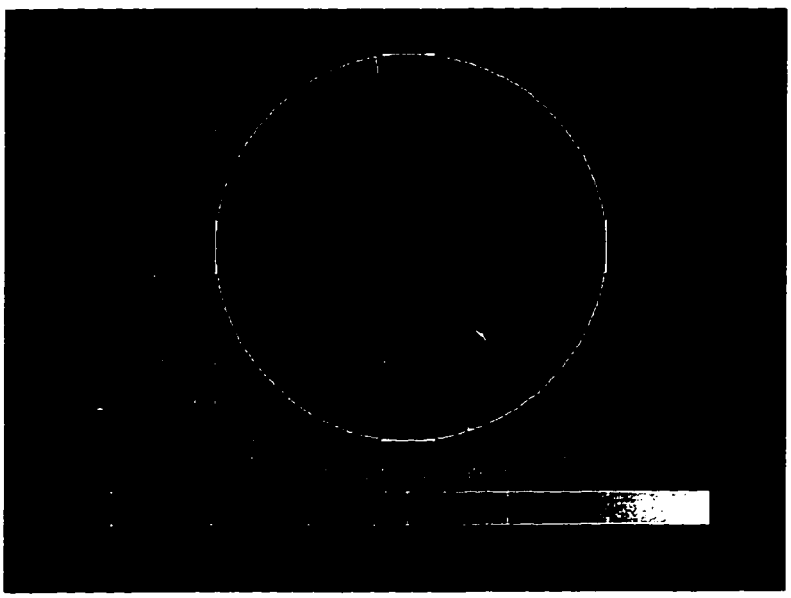


Figure 6.20: Experiment 1: The scattering plot for scan 5 created by using wavelets/PTSVQ in the complex Cartesian domain. The entropy is 0.14 bits per complex number, and the compression ratio is 130.03:1.

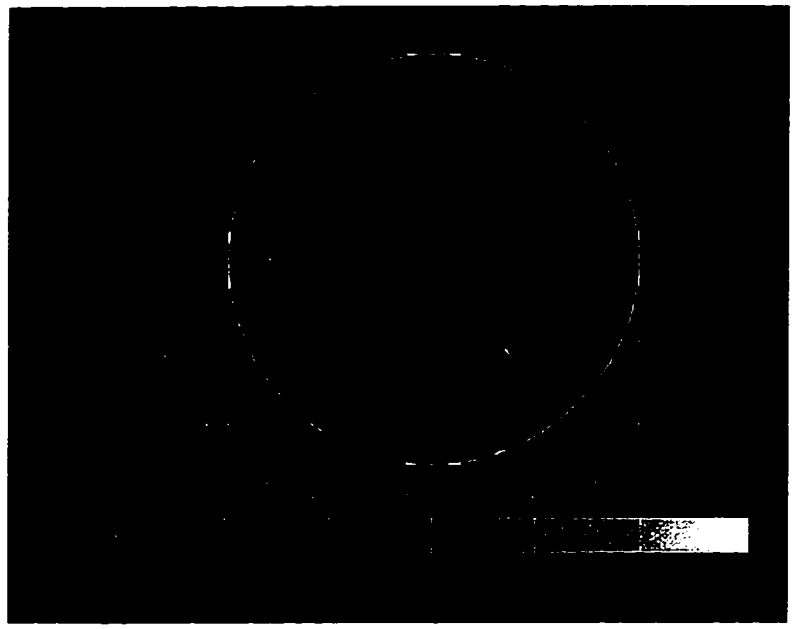


Figure 6.21: Experiment 1: The scattering plot for scan 5 created by averaging 16 samples at a time. The entropy is 1.4465 bits per complex number, and the compression ratio is 12.29:1

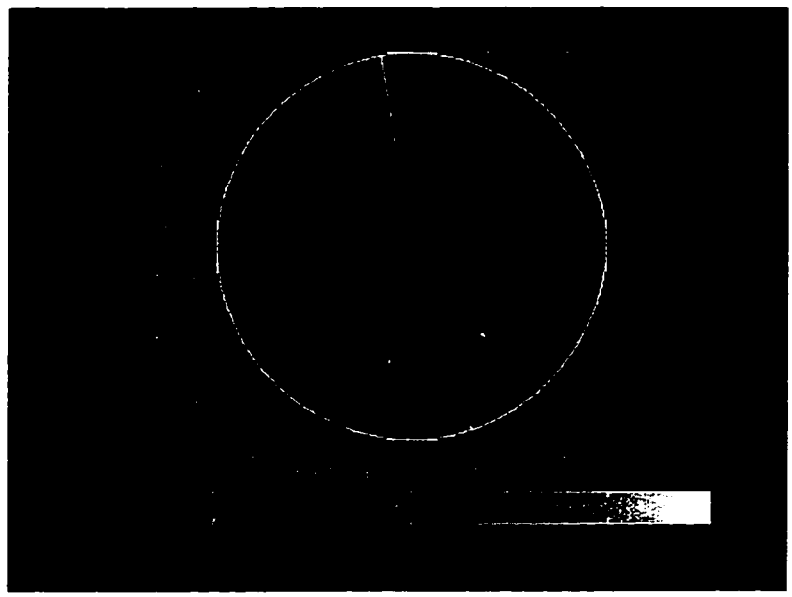
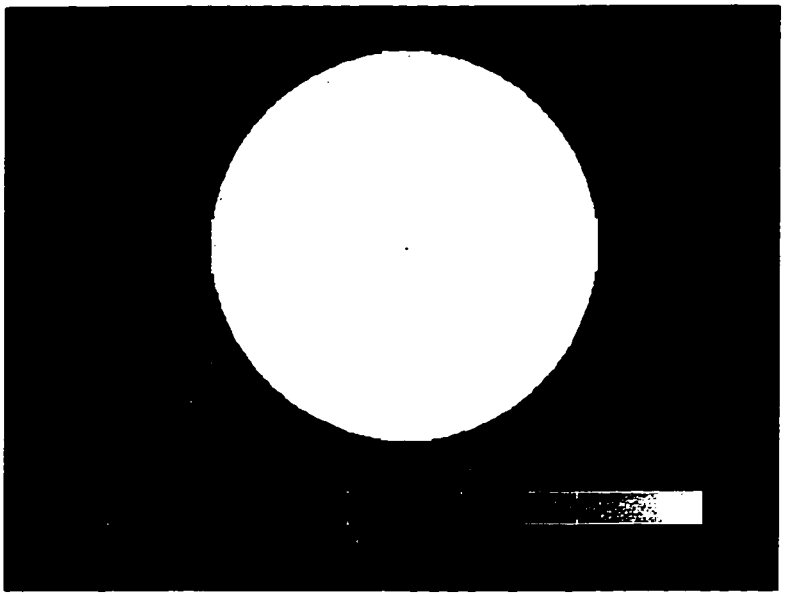
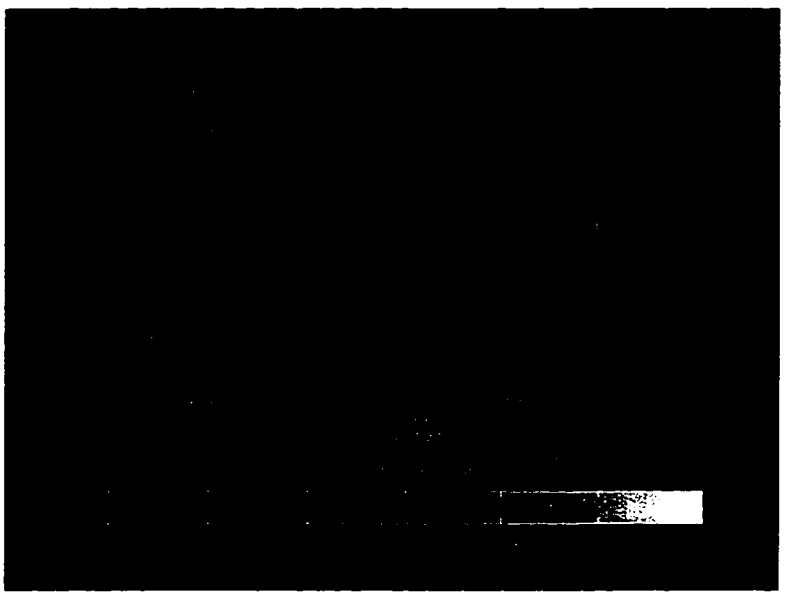


Figure 6.22: Experiment 1: The scattering plot for scan 5 created by using PTSVQ in the complex Cartesian domain. The entropy is 0.4741 bits per complex number, and the compression ratio is 37.49:1.

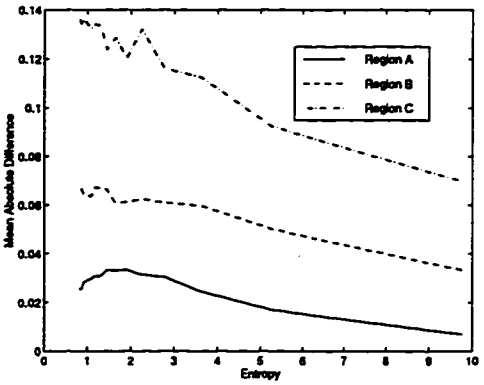


(a) Correlation plot.

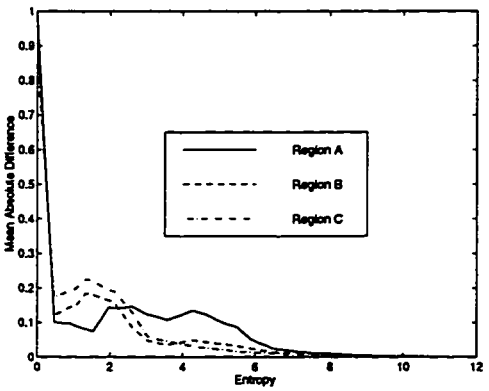


(b) Sound speed plot.

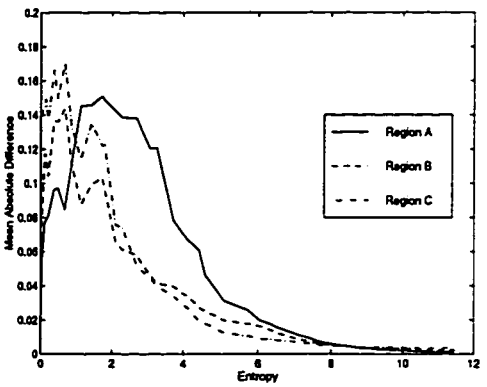
Figure 6.23: The original (a) correlation plot and (b) sound speed plots computed from scan 5 and the synthetic data.



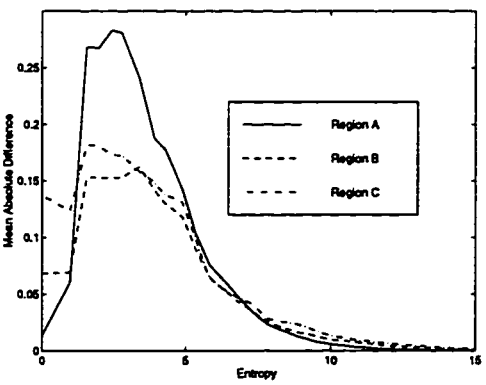
(a) Experiment 1, simple averaging.



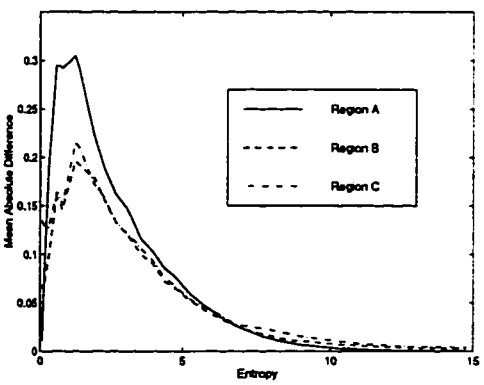
(b) Experiment 1, PTSVQ.



(c) Experiment 1, wavelets/PTSVQ.

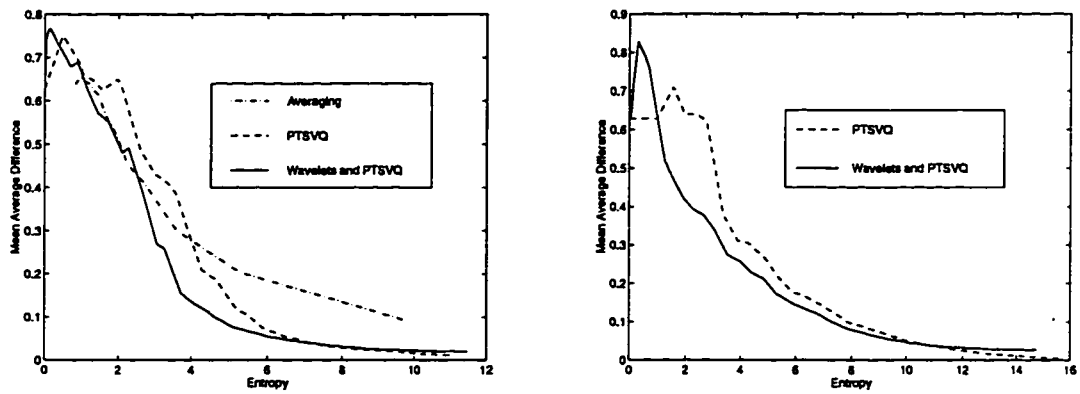


(d) Experiment 2, PTSVQ.



(e) Experiment 2, wavelets/PTSVQ.

Figure 6.24: The mean absolute difference in correlation for scan 5 and the synthetic scan, regions A, B, and C. (a) simple averaging in (X, Y), (b) PTSVQ in (X, Y), (c) wavelets/PTSVQ in (X, Y), (d) PTSVQ in (P, Φ), and (e) wavelets/PTSVQ in (P, Φ).



(a) Experiment 1.

(b) Experiment 2.

Figure 6.25: The mean absolute difference in sound speed measurements for scan 5 and the synthetic scan for (a) Experiment 1 and (b) Experiment 2.

Chapter 7

CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

In this dissertation the feasibility of lossy compression of image and earth science data was studied. Lossy compression was done by coding wavelet transform coefficients using vector quantization. Three optimal bit allocation methods for the wavelet transform were developed for this purpose, including one adaptive technique permitting joint bit allocation and signal space representation. These compression algorithms were applied to USC database images and underwater acoustic sonar data, and the results of compression were evaluated.

To develop these compression algorithms, standard lossy data compression techniques including full search vector quantization, the generalized Lloyd algorithm, a greedy growing algorithm for tree structured vector quantization, two tree pruning algorithms (the generalized Breiman Friedman Olshen and Stone algorithm and the recursive optimal pruning algorithm), as well as an overview of data quality measures were presented in Chapter 2. In Chapter 3, the discrete wavelet transform and wavelet packet transform for one- and two-dimensional signals were reviewed, and the relationships between distortion, rate, and entropy in the original and wavelet domains were derived to demonstrate compression and bit allocation feasibility in the wavelet domain.

In Chapter 4, current wavelet and quantization compression schemes were reviewed, and three bit allocation algorithms for wavelet subbands using a pruned tree-structured vector quantization approach were developed. The first two methods

use the GBFOS and ROPA tree pruning approaches to allocate bits among wavelet subbands of a predetermined basis as given by the discrete wavelet transform. The third approach (for wavelet packets) uses the GBFOS pruning approach to allocate bits among wavelet packet subbands while simultaneously determining the best basis decomposition.

In Chapter 5, wavelets/PTSVQ compression using the three different bit allocation techniques was used to compress several images from the USC database. It was found that the ROPA bit allocation approach found many more codebooks than the GBFOS bit allocation approach for the DWT, and that for the same bit rates, WPT bit allocation generally found more codebooks too. When comparing the two techniques, WPT/PTSVQ outperformed DWT/PTSVQ for training data which was expected since the DWT is a subset of the WPT, and the WPT approach is adaptive. As predicted by rate-distortion theory, PSNR results improved with increasing vector dimension and increasing level of wavelet decomposition as this, for the WPT, permitted better adaptation to the source. Unfortunately, the results for test data behaved as expected only at very low bit rates. Since the rate-distortion curves flattened out rapidly for the test data, I believe that DWT/PTSVQ and WPT/PTSVQ, especially, overadapted themselves to the limited amount of training data available.

In Chapter 6 the benefits of compression of acoustic data were examined. Two applications, target detection (a highly robust calculation) and ocean floor temperature studies (derived from sensitive correlation and sound speed measurements), were used to evaluate data compression quality. DWT/PTSVQ, PTSVQ, and an averaging compression algorithm were applied to several acoustic sonar scans in both the complex Cartesian and polar domains. Experimental results showed that wavelets/PTSVQ consistently outperformed PTSVQ and simple averaging (which was found to be unviable) in terms of SNR, backscatter plot quality, correlation and sound speed measurements. The improvement in performance is clearly worth the added complexity of using adaptive transform coding to compress the signal.

It was also found that the decision of the domain in which compression should be done is a function of the application, as compression results vary from application to application. When examining the target viewing application, it was found that high compression ratios can be sustained. This is ideal for quick look and browse applications. However the ocean temperature studies, which depend on measurements such as correlation and sound speed measurements, are sensitive to the smallest changes in data. For this case one can expect to be able to use lossy compression on the data with acceptable accuracy for compression ratios of perhaps no more than 4:1. I conclude that the use of only one approach for the compression of acoustic data will not provide reasonable results when a variety of different scientific analyses are to be applied to the data. Rather, compression should be tailored to the application or analysis for which the data will be used whether it be for previewing and browsing purposes, or for scientific analysis.

In addition to studying the problems of lossy compression of scientific data and optimal bit allocation for wavelets, an algorithm for the coding of color images that permits joint progressive transmission and low bit-depth display and printing is presented in Appendix A. This algorithm permits the embedding of the output of binary Floyd and Steinberg error diffusion into the output of grayscale and color error diffusion; the embedding of grayscale halftones into higher bit-depth grayscale and color halftones; and the embedding of color halftones into higher bit-depth color halftones. Preprocessing algorithms are also presented to organize color palettes for effective use of the color space when embedding halftones into color halftones.

Possible directions for future work for wavelet/VQ encoding and the compression of scientific data include:

- Different wavelet transforms such as biorthogonal wavelets which promise better data coding, but make bit allocation more complicated due to loss of orthonormality.

- A ROPA-like approach to WPT bit allocation to find the WPT codebooks close to the lower convex hull of the rate distortion curve.

- Coefficient modeling in each subband as in [2] to help overcome the dependence on training data.

- Other forms of VQ, such as predictive PTSVQ as used in [81], lattice VQ, universal VQ, or even ternary PTSVQ for the wavelet subbands to try to better encode the subbands with highly peaked distributions.

- Joint selection of vector dimension and bit allocation as in [29] to help balance the complexity of encoding and training with the quality of encoded data.

- The establishment of a freely available source of images to create a standard set of test and training images where the images are selected for certain characteristics, and their means of creation known. This will aid benchmarking comparisons which are difficult for the "Lenna" image since a number of different versions of this image at different resolutions created from color scanning and grayscale conversion or grayscale scanning are used throughout the imaging community.

- Development of other measures to predict the viability of different compression schemes such as hybrid distortion measures, or non-MSE based distortion measures. SNR measurements are a popular and easy way of measuring the success of compression experiments, but they are not necessarily good predictors of compression performance for scientific data.

- The evaluation of compression of other earth science data and applications.

BIBLIOGRAPHY

[1] H. Abut, editor. *Vector Quantization.* IEEE Press, Piscataway, NJ, May 1990.

[2] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using wavelet transformation. *IEEE Transactions on Image Processing*, 1(2):205–220, 1992.

[3] David Applegate and Bill Cook. Solving large-scale matching problems. Technical report, DIMACS, Rutgers Univ. Piscataway, NJ, 1991.

[4] N. Baaziz and C. Labit. Laplacian pyramid versus wavelet decomposition for image sequence coding. In *ICASSP-90*, volume 4, pages 1965–1968. IEEE Signal Processing Society, April 1990.

[5] R. Balasubramanian and J. Allebach. A new approach to palette selection for color images. *Journal of Imaging Technology*, 17(6):284–290, December 1991.

[6] R. Balasubramanian, C. A. Bouman, and J. P. Allebach. Sequential scalar quantization of color images. *Journal of Electronic Imaging*, 3(1):45–59, January 1994.

[7] R. Balasubramanian, C. A. Bouman, and J. P. Allebach. Sequential scalar quantization of vectors: an analysis. *IEEE Transactions on Image Processing*, 4(9):1282–1295, September 1995.

[8] M. R. Banham and B. J. Sullivan. A wavelet transform image coding technique with a quadtree structure. In *ICASSP-92*, volume 4, pages 653–656. IEEE Signal Processing Society, March 1992.

[9] M. Barlaud, P. Solé, M. Antonini, and P. Mathieu. A pyramidal scheme for lattice vector quantization of wavelet transform coefficients applied to image coding. In *ICASSP-92*, volume 4, pages 401–404. IEEE Signal Processing Society, March 1992.

[10] J. G. Bellingham, C. A. Goudey, T. R. Consi, J. W. Bales, D. K. Atwood, J. J. Leonard, and C. Chyrssostomidis. A second generation survey AUV. In *Proceedings of the 1994 Symposium on Autonomous Underwater Vehicle Technology*, pages 148 – 155. IEEE, July 1994.

[11] Jonathan N. Bradley and Christopher M. Brislawn. Image compression by vector quantization of multiresolution decompositions. Technical report, Los Alamos National Laboratory, Los Alamos, New Mexico, 87545, 1991.

[12] Jonathan N. Bradley and Christopher M. Brislawn. Applications of wavelet-based compression to multidimensional earth science data. In *Proceedings of the 1993 Data Compression Conference*, pages 224–233. IEEE Computer Society Press, April 1993.

[13] Jonathan N. Bradley and Christopher M. Brislawn. Proposed first-generation VQ bit allocation procedure. In *Proc. Symp. Criminal Justice Info. Services Tech.*, September 1993.

[14] Jonathan N. Bradley and Christopher M. Brislawn. Wavelet transform-vector quantization compression of supercomputer ocean models. In *1993 Space and Earth Science Data Compression Workshop*, pages 13–24. NASA, April 1993.

[15] Jonathan N. Bradley, Christopher M. Brislawn, and Tom Hopper. The FBI wavelet/scale quantization standard for gray-scale fingerprint image compression. In *SPIE Proceedings, Visual Info. Process. II*, volume 1961, Orlando, FL, April 1993.

[16] G. W. Braudaway. A procedure for optimum choice of a small number of colors from a large color palette for color imaging. In *Electronic Imaging '87: International Electronic Imaging Exposition and Conference*, pages 71–75, February 1987.

[17] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* The Wadsworth Statistics/Probability Series. Wadsworth, Belmont, California, 1984.

[18] M. Broja, K. Michalowski, and Olof Bryngdahl. Error diffusion concept for multi-level quantization. *Optics Communications*, 79(5):280–284, November 1990.

[19] A. G. Bruce, D. B. Percival, and J. R. Goldschneider. Algorithms for wavelet compression of acoustic signals. Naval Air Warfare Center Aircraft Division, Phase I Final Report N00421-95-C-1148, Data Analysis Products Division, MathSoft, Inc., 1700 Westlake Ave. N., Suite 500, Seattle, WA 98109, 1996.

[20] Andrew Bruce and Hong-Ye Gao. *S+WAVELETS User's Manual, Version 1.0.* StatSci, a division of MathSoft, Inc., 1700 Westlake Ave. N, Seattle, WA 98109, 1994.

[21] A. Buzo, A. H. Gray Jr., R. M. Gray, and J. D. Markel. Speech coding based upon vector quantization. *IEEE Transactions on Acoustics Speech and Signal Processing*, 28:562–574, October 1980.

[22] E. Cammarota and G. Poggi. Address vector quantization with topology-preserving codebook ordering. In *Proc. 13th GRETSI Symposium*, pages 853–856, September 1991.

[23] P. A. Chou, T. Lookabaugh, and R. M. Gray. Optimal pruning with applications to tree-structured source coding and modeling. *IEEE Transactions on Information Theory*, 35(2):299–315, March 1989.

[24] R. Coifman and V. Wickerhauser. Entropy-based algorithms for best basis selection. *IEEE Transactions on Information Theory*, 38(2):713–718, 1992.

[25] A. B. Coppens. Simple equations for the speed of sound in neptunian waters. *Journal of the Acoustical Society of America*, 69(3):862–863, March 1981.

[26] P. C. Cosman, R. M. Gray, and M. Vetterli. Vector quantization of image subbands: a survey. *IEEE Transactions on Image Processing*, 5(2):202–225, February 1996.

[27] P. C. Cosman, S. M. Perlmutter, and K. O. Perlmutter. Tree-structured vector quantization with significance map for wavelet image coding. In *Proceedings Data Compression Conference*, pages 33–41, April 1995.

[28] Criminal Justice Information Services. WSQ gray-scale fingerprint image compression specification. Technical report, Federal Bureau of Investigation, February 1993.

[29] V. Cuperman. Joint bit allocation and dimensions optimization for vector transform quantization. *IEEE Transactions on Information Theory*, 39:302–305, January 1993.

[30] E. A. B. da Silva, D. G. Sampson, and M. Ghanbari. Image coding using successive approximation wavelet vector quantization. In *Proceedings of ICASSP*, volume 4, pages 2201–2204, May 1995.

[31] I. Daubechies. *Ten lectures on wavelets.* Society for industrial and applied mathematics, Philadelphia, PA, 1992.

[32] Philippe Desarte, Benoit Macq, and Dirk T. M. Slock. Signal-adapted multiresolution transform for image coding. *IEEE Transactions on Information Theory*, 38(2):897–904, 1992.

[33] Ronald A. DeVore, Bjorn Jawerth, and Bradley J. Lucier. Image compression through wavelet transform coding. *IEEE Transactions on Information Theory*, 38(2):719–746, 1992.

[34] F. Dufaux, I. Moccagatta, B. Rouchouze, T. Ebrahimi, and M. Kunt. Motion-compensated generic coding of video based on a multiresolution data structure. *Optical Engineering*, 32(7):1559–1569, 1993.

[35] J. G. Dworski and D. R. Jackson. Spatial and temporal variation of acoustic backscatter in the STRESS experiment. *Continental Shelf Research*, 14(10/11):1221–1237, 1994.

[36] B. R. Epstein, R. Hingorani, J. M. Shapiro, and M. Czigler. Multispectral KLT-Wavelet data compression for Landsat thematic mapper images. In *Proceedings of the 1992 Data Compression Conference*, pages 200–208, May 1992.

[37] W. H. Equitz. A new vector quantization clustering algorithm. *IEEE Transactions on Acoustics Speech and Signal Processing*, 37(10):1568–1575, October 1989.

[38] N. Farvardin. A study of vector quantization for noisy channels. *IEEE Transactions on Information Theory*, 36(4):799–809, July 1990.

[39] R. Floyd and L. Steinberg. An adaptive algorithm for spatial gray scale. *SID Int. Sym. Digest of Tech. Papers*, pages 36–37, 1975.

[40] Jacques Froment and Stéphane Mallat. Second generation compact image coding with wavelets. In Charles K. Chui, editor, *Wavelets: A tutorial in theory and applications*, pages 655–678. Academic Press, Inc., San Diego, CA, 1992.

[41] R. S. Gentile, I. Walowit, and J. P. Allebach. Quantization and multilevel halftoning of color images for near-original image quality. *Journal of the Optical Society of America*, 7(6):1019–1026, June 1990.

[42] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression.* Kluwer Academic Publishers, Boston, MA, 1992.

[43] K. H. Goh, J. J. Soraghan, and T. S. Durrani. New 3D wavelet transform coding algorithm for image sequences. *Electronics Letters*, 29(4):401–402, 1993.

[44] J. R. Goldschneider, E. A. Riskin, and P. W. Wong. Embedded multilevel error diffusion. In *Proceedings of SID 1995*, volume XXVI, pages 829–832, May 1995.

[45] R. M. Gray. Vector quantization. *IEEE ASSP Magazine*, 1:4–29, April 1984.

[46] P. Heckbert. Color image quantization for frame buffer display. *Computer Graphics*, 16(3):297–307, July 1982.

[47] C. Herley, J. Kovačević, K. Ramchandran, and M. Vetterli. Arbitrary orthogonal tilings of the time-frequency plane. *IEEE Transactions on Signal Processing*, 41(12):3341–3359, December 1993.

[48] C.-M. Huang, Q. Bi, S. Stiles, and R. W. Harris. Fast full search equivalent encoding algorithms for image compression using vector quantization. *IEEE Transactions on Image Processing*, 1(3):413–416, July 1992.

[49] R. W. G. Hunt. *The Reproduction of Colour in Photography, Printing & Television.* Tolworth, Fountain Press, England, 4 edition, 1987.

[50] W.-J. Hwang and H. Derin. Multi-resolution multi-rate progressive image transmission. In *Twenty-Seventh Asilomar Conference on Signals Systems and Computers*, volume 1. IEEE Computer Society Press, November 1993.

[51] J. Impagliazzo, W. Greene, and Q. Huynh. Wavelet image compression algorithm for side scan sonar and teleradiology. In *Wavelet Applications II*, volume 1, pages 162–172. SPIE, April 1995.

[52] V. S. Iverson and E. A. Riskin. A fast method for combining palettes of color quantized images. In *Proceedings of ICASSP*, volume 5, pages 317–320. IEEE Acoustics Speech and Signal Processing Society, April 1993.

[53] D. R. Jackson. Unpublished Fortran Programs, 1995.

[54] D. R. Jackson. Unpublished Matlab Programs, 1995.

[55] D. R. Jackson. Private Communication, August 1996.

[56] D. R. Jackson. Private Communication, August 1997.

[57] D. R. Jackson and J. G. Dworski. An acoustic backscatter thermometer for remotely mapping seafloor water temperature. *Journal of Geophysical Research*, 97(C1):761–767, January 1992.

[58] D. R. Jackson and K. L. Williams. High-frequency sea-bed scattering measurements in shallow water. In *Proceedings of the Third European Conference on Underwater Acoustics*. FORTH/IACM, June 1996.

[59] M. H. Johnson, R. Ladner, and E. A. Riskin. Fast nearest neighbor search for ECVQ and other modified distortion measures. In *Proceedings of ICIP*, volume III, pages 423–426, 1996.

[60] I. Katsavounidis and C.-C. J. Kuo. Image compression with embedded wavelet coding via vector quantization. In *Proceedings of the SPIE*, volume 2569, pages 333–344. SPIE, July 1995.

[61] S.-Z. Kiang, R. L. Baker, G. J. Sullivan, and C.-Y. Chiu. Recursive optimal pruning with applications to tree-structured vector quantizers. *IEEE Transactions on Image Processing*, 1(2):162–169, April 1992.

[62] D. H. Kil and F. B. Shin. Reduced dimension image compression for remotely distributed underwater signal processing. In *Oceans '95*, volume 2, pages 1883–1888. IEEE, October 1995.

[63] B. W. Kolpatzik and C. A. Bouman. Optimized error diffusion for image display. *Journal of Electronic Imaging*, 1(3):277–292, July 1992.

[64] B. W. Kolpatzik and C. A. Bouman. Optimized universal color palette design for error diffusion. *Journal of Electronic Imaging*, 4(2):131–143, April 1995.

[65] E. L. Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart and Winston, New York, 1976.

[66] A. S. Lewis and G. Knowles. Video compression using 3D wavelet transforms. *Electronics Letters*, 26(6):396–398, 1990.

[67] A. S. Lewis and G. Knowles. A 64 kb/s video coder using the 2-D wavelet transform. In *Proceedings of the Data Compression Conference*, Snowbird, UT, 1991.

[68] J. Li, P. Y. Cheng, and C.-C. J. Kuo. On the improvements of embedded zerotree wavelet (EZW) coding. In *Proceedings of the SPIE*, volume 2501, pages 1490–1501. SPIE, May 1995.

[69] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28:84–95, January 1980.

[70] S. P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–136, March 1982. Previously an unpublished Bell Laboratories Technical Note (1957).

[71] K. V. Mackenzie. Nine-term equation for sound speed in the oceans. *Journal of the Acoustical Society of America*, 70(3):807–812, September 1981.

[72] Stéphane Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989.

[73] Stéphane Mallat and Wen Liang Hwang. Singularity detection and processing with wavelets. *IEEE Transactions on Information Theory*, 38(2):617–643, 1992.

[74] K. V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate Analysis.* Academic Press, 1979.

[75] Nader Moayeri, Ingrid Daubechies, Qing Song, and Hong Shen Wang. Wavelet transform coding using trellis coded vector quantization. In *ICASSP-92*, volume 4, pages 405–408. IEEE Signal Processing Society, March 1992.

[76] K. L. Oehler and R. M. Gray. Combining image compression and classification using vector quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):461–473, May 1995.

[77] M. Orchard. A fast nearest-neighbor search algorithm. In *Proceedings of ICASSP*, pages 2297–2300. IEEE Acoustics Speech and Signal Processing Society, 1991.

[78] M. T. Orchard and C. A. Bouman. Color quantization of images. *IEEE Transactions on Signal Processing*, 39(12):2677–2690, December 1991.

[79] K. Ramchandran and M. Vetterli. Best wavelet packet bases in a rate-distortion sense. *IEEE Transactions on Image Processing*, 2(2):160–175, 1993.

[80] K. Ramchandran, M. Vetterli, and C. Herley. Wavelets, subband coding, and best bases. *Proceedings of the IEEE*, 84(4):541–560, April 1996.

[81] E. A. Riskin. *Variable rate vector quantization of images.* PhD thesis, Stanford University, Stanford, CA, May 1990.

[82] E. A. Riskin. Optimum bit allocation via the generalized BFOS algorithm. *IEEE Transactions on Information Theory*, 37(2):400–402, March 1991.

[83] E. A. Riskin, R. Ladner, R.-Y. Wang, and L. E. Atlas. Index assignment for progressive transmission of full search vector quantization. *IEEE Transactions on Image Processing*, 3(3):307–312, May 1994.

[84] J. A. Saghri and A. G. Tescher. Sonar feature-based bandwidth compression. *Journal of Visual Communication and Image Representation*, 4(2):130–148, June 1993.

[85] A. Said. and W. A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3):243–50, June 1996.

[86] Masoud Sajadieh and Anastasios N. Venetsanopoulos. Wavelet vector quantization of images. In *ICASSP-92*, volume 4, pages 471–474. IEEE Signal Processing Society, March 1992.

[87] T. Senoo and B. Girod. Vector quantization for entropy coding of image subbands. *IEEE Transactions on Image Processing*, 1(4):526–533, October 1992.

[88] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41(12):3445–3462, December 1993.

[89] M. Stojanovic. Recent advances in high-speed underwater acoustic communications. *IEEE Journal of Oceanic Engineering*, 21(2):125–136, April 1996.

[90] Gilbert Strang. Wavelets and dilation equations: A brief introduction. *SIAM Review*, 31(4):614–627, 1989.

[91] Gilbert Strang. Wavelet transforms versus Fourier transforms. *Bulletin of the American Mathematical Society*, 28(2):288–305, 1993.

[92] D. Tang, G. Jin, D. R. Jackson, and K. L. Williams. Analyses of high-frequency bottom and subbottom backscattering for two distinct shallow water environments. *Journal of the Acoustical Society of America*, 96(5):2930–2936, November 1994.

[93] R. Ulichney. *Digital Halftoning.* MIT Press, Cambridge, MA, 1987.

[94] S. D. Voran and L. L. Scharf. Polar coordinate quantizers that minimize mean-squared error. *IEEE Transactions on Signal Processing*, 42(6):1559–1563, June 1994.

[95] X. Wen, W. Yuling, and Z. Weiqing. Sonar image processing system for an autonomous underwater vehicle AUV. In *Oceans '95*, volume 3, pages 1883–1886. IEEE, October 1995.

[96] Mladen V. Wickerhauser. Acoustic signal compression with wavelet packets. In Charles K. Chui, editor, *Wavelets: a tutorial in theory and applications*, pages 679–700. Academic Press, San Diego, CA, 1992.

[97] Mladen V. Wickerhauser. *Adapted Wavelet Analysis - from theory to software.* A. K. Peters, Ltd, 1994.

[98] P. W. Wong. Adaptive error diffusion and its application in multiresolution rendering. *IEEE Transactions on Image Processing*, 5(7):1184–1196, July 1996.

[99] K. Zeger and A. Gersho. Pseudo-gray coding. *IEEE Transactions on Communications*, 38(12):2147–2158, December 1990.

[100] Ya-Qin Zhang and Sohail Zafar. Motion-compensated wavelet transform coding for color video compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 2(3), September 1992.

Appendix A

EMBEDDED MULTILEVEL ERROR DIFFUSION

Multilevel halftoning [93] or multitoning is a process used to render an image on a device with a limited number of tones to produce an image with the illusion of more tones. Multilevel error diffusion is similar to the common binary Floyd and Steinberg error diffusion algorithm [39], but more than two output levels or colors are available for display. The additional output levels or colors can be used to greatly improve image quality.

The block diagram for Floyd and Steinberg's binary error diffusion algorithm [39] is shown in Figure A.1, where X is the input image, Y is the output halftoned image, $Q(\ )$ is a binary scalar quantizer, and H is the error filter. To achieve multilevel error diffusion for a grayscale image, the binary scalar quantizer $Q(\ )$ in Floyd and Steinberg's algorithm is simply replaced with a multiple-bin scalar quantizer as suggested by Ulichney [93]. For color images, a color palette, or vector quantizer of three dimensions (red, green, blue), replaces the binary scalar quantizer [16, 46, 64, 78].

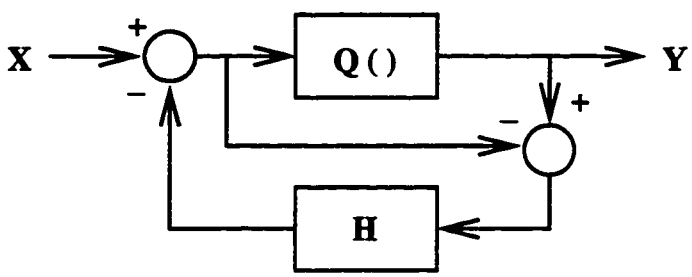


Figure A.1: Block diagram of the binary error diffusion algorithm where $Q()$ is the binary scalar quantizer and H is the error propagation mask.

Vector quantization (VQ) [1, 42, 45] is a lossy compression technique that has been used extensively for image compression. VQ design techniques have also been extensively used for color palette design and color image display [5, 16, 41, 46, 52, 78]. For my embedded multilevel error diffusion algorithm, I use uniform scalar quantizers when working with grayscale images and full-search vector quantizers when working with color images. The full-search VQ's are designed using the generalized Lloyd algorithm [69] although it is also possible to use sequential scalar quantization [6, 7].

Multilevel error diffusion has been widely used [16, 18, 41, 46, 78, 93] to either enhance displayed images by the use of more output values or to reduce the contouring effects that occur due to limited palette size. In [18], binary error diffusion is extended to three-level grayscale error diffusion. The authors note a marked improvement in image quality, but they warn that the choice of quantization levels can greatly affect image quality. In [46] the display of 24-bit color images using a limited color palette size is discussed. Methods for generating full-search color palettes and methods for quick full searches are introduced. Furthermore, it is shown that excellent image quality with color palettes of very small size (as few as four colors) is obtained when error diffusion is used. In [16] small color palettes for displaying color images are generated with the generalized Lloyd algorithm [69]. A modified version of multilevel error diffusion is used to reduce contouring introduced by the small palette size. In [78] tree-structured color palettes are developed for color image display. A modification of multilevel error diffusion that is adjusted to work with the parameters of their tree-structured palette is used to reduce contouring effects. In [41] the effects of multilevel error diffusion and halftoning on color image quality when using a restricted color-palette size are examined. As expected, the results confirm that multilevel error diffusion greatly enhances color-image quality for restricted color-palette sizes.

While multilevel error diffusion yields excellent visual results, an image must be independently processed for each different number of output levels required. It would be nice to have the multilevel error diffused image be embedded: the error diffused

image with N output levels should also "contain" the error diffused images for smaller numbers of output levels including the Floyd and Steinberg binary error diffused image. One benefit of embedding is that a Floyd and Steinberg error diffused image can be obtained for printing or display on a binary device by simply masking one bit off of the multilevel image or indices. A second benefit is progressive transmission capabilities: first one bit plane of the image, the Floyd and Steinberg error diffused image, can be sent to the user; later, more bit planes can be sent to the user if desired, leading to an improved image.

One possible application for embedded multilevel error diffusion is that of an image database or browsing system, such as the World Wide Web, in which images stored in a lossy, error diffused format are accessed by users with different display devices. In this example a user with an 8-bit color monitor can receive the image indices (progressively perhaps) and color palette to display an 8-bit color image; a user with a monochrome screen can receive one bit of the 8-bit color-image indices to display a binary error diffused version of the image; and a user with a 16-level LCD can receive four bits of the 8-bit color-image indices to display a 16-level grayscale image. Furthermore, each of these three users could then send the received image to a binary or multilevel, monochrome or color printer, as appropriate, to receive a high quality hardcopy output.

Note that there is a closely related approach in embedding multiple halftones where a halftone "contains" halftones with the same number of output levels but of different sizes [98]. For example, a bilevel halftone of size $N \times N$ can be designed that contains a smaller sized bilevel halftone of size $aN \times aN$, where a is a rational number less than 1. In this case, it is required that the two halftones of sizes $aN \times aN$ and $N \times N$ be related by simple down-sampling (i. e. one can obtain the small halftone by down-sampling the large one), and that both the large and small sized halftones be of high quality. It is evident that halftones embedded in size can also be progressively transmitted. This approach can be combined with the multilevel

embedding approach in this appendix for a general multilevel/multisize embedded halftoning algorithm which could be used for devices with different resolutions such as a 1200×1000 CRT or a 600×800 LCD.

In this appendix I present an algorithm to embed binary error diffused images into grayscale or color error diffused images; to embed grayscale error diffused images into grayscale or color error diffused images; and to embed color error diffused images into color error diffused images. Due to constraints on the palettes introduced by embedding, image quality in the higher bit-depth image may be reduced. To counteract the reduction in image quality, the embedding algorithm requires the use of ordered scalar and vector quantizers. While scalar quantizers have a natural ordering, vector quantizers as used for color quantization do not. In this appendix I present algorithms for color palette organization to be used as a preprocessing step to the embedded multilevel error diffusion algorithm.

This appendix is organized as follows. In Section A.1, embedded multilevel error diffusion is described. In Section A.2, palette organization for embedded multilevel error diffusion is described. Results are presented in Section A.3, and conclusions are presented in Section A.4.

A.1 Embedded Multilevel Error Diffusion

There are many approaches to embedding error diffused images into higher-level images. For example, an N-level multilevel error diffused image containing an M-level error diffused image can be produced by independently generating an M-level and an (N-M)-level error diffused image and appending these two images together so that the image is "embedded." By this method the embedded halftoning property has been achieved. However, the assignment of the N available colors to the amalgamated index is not clear or obvious. Such an image would have poor color and image quality. For example, if $M = \frac{N}{2}$, two M-level halftones would be generated, and the

same two halftones would be appended to make the N-level halftone. This halftone would have only M unique indices, and thus only M colors. At best it would look just like the M-level image. Clearly, this naive approach cannot make good use of the larger color space.

My embedded error diffusion algorithm embeds an M-level error diffused image into an N-level error diffused image where $M < N$ by a two stage algorithm. In the first stage, the M-level error diffused image is produced by using an M-bin scalar or vector quantizer. In the second stage M quantizers, each with $\frac{N}{M}$ output levels, are used to produce the N-level error diffused image where the output of the first stage determines which of the M quantizers to use. The block diagram is shown in Figure A.2, where $Q_1^1()$ is the M-bin scalar or vector quantizer used in the first stage, and $Q_1^2(), Q_2^2(), \ldots, Q_M^2()$ are the $\frac{N}{M}$-bin scalar or vector quantizers used in the second stage.

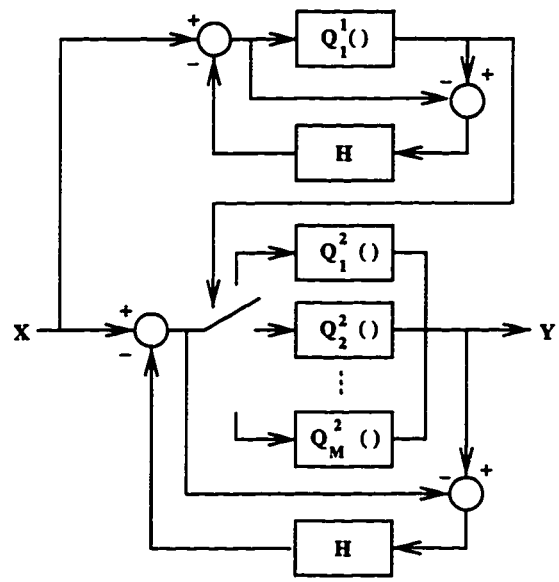


Figure A.2: Block diagram of the embedded multilevel error diffusion algorithm where $Q_1^1()$ is an M-bin scalar quantizer, and $Q_1^2(), Q_2^2(), \ldots, Q_M^2()$ are $\frac{N}{M}$-bin scalar or vector quantizers designed to embed an M-level halftone into an N-level halftone.

To create the second stage quantizers, it is assumed that an *ordered* N-bin quantizer $Q^2()$ with binary indices $\{0, 1, \ldots, N-1\}$ and associated codewords $\{q_0^2, q_1^2, \ldots, q_{N-1}^2\}$ has been designed for normal N-level error diffusion. Different disjoint subsets of these indices are used to form the M $\frac{N}{M}$-bin second-stage quantizers $Q_1^2(), Q_2^2(), \ldots, Q_M^2()$. The partitioned sets are such that all codewords in the same set share the common bits that satisfy the embedding requirement from the output of the first stage.

The constraint of subdividing the N-word palette into M different palettes will affect the quality of the output. Both the relative size of M to N as well as the assignment of colors to the subsets are affective factors. If $0 < \log_2 M < \frac{\log_2 N}{2}$, the M-level image is said to be *small* relative to the N-level image. If $\frac{\log_2 N}{2} < \log_2 M < \log_2 N$, the M-level image is said to be *large* relative to the N-level image. Palette organization and its effects on embedding for both cases will be discussed.

I next discuss in detail a special case of embedded multilevel error diffusion, embedded binary error diffusion [44], which was used to embed a binary error diffused image into a grayscale error diffused image. This is done for the purpose of motivating my approach to the ordering of color palettes.

Given a binary error diffused image, embedding can be done in any bit of the grayscale error diffused output depending on the division of quantizer $Q^2()$'s indices and codewords among quantizers $Q_1^2()$ and $Q_2^2()$. For example, the ordered binary and scalar quantizers used for 8-level embedded binary error diffusion are shown in Figure A.3(a) where the dots refer to quantizer outputs. The binary error diffused image is embedded into the most and least significant bit of the output since these bits are the most convenient. If the quantizer $Q_1^2()$ uses the codewords of $Q^2()$ with even indices $\{0, 2, \ldots, N-2\}$ and the quantizer $Q_2^2()$ uses the codewords of $Q^2()$ with odd indices $\{1, 3, \ldots, N-1\}$, as shown in Figure A.3(b), then the binary error diffused image is embedded in the least significant bit (LSB) of the grayscale output and can be obtained from the grayscale image simply by masking off the LSB of each

pixel value. If the quantizer $Q_1^2()$ uses the codewords of $Q^2()$ with the first $\frac{N}{2}$ indices $\{0, 1, \ldots, \frac{N}{2} - 1\}$ and the quantizer $Q_2^2()$ uses the codewords of $Q^2()$ with the second $\frac{N}{2}$ indices $\{\frac{N}{2}, \frac{N}{2} + 1, \ldots, N - 1\}$, as shown in Figure A.3(c), then the binary error diffused image is embedded in the most significant bit (MSB) and can be obtained from the grayscale image by thresholding. Note that after quantizer Q^2 is partitioned into two sets, the bin boundaries of the new quantizers Q_1^2 and Q_2^2 are not the same as the bin boundaries of Q^2.

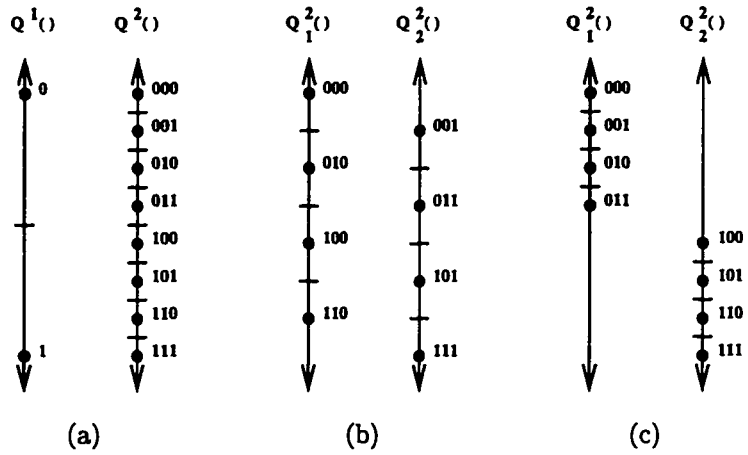


Figure A.3: (a) Scalar quantizers $Q^1()$ and $Q^2()$ used for grayscale embedded binary error diffusion where $Q_1^2()$ and $Q_2^2()$ are the partitions of $Q^2()$ for embedding in the (b) LSB and the (c) MSB. The dots refer to quantizer outputs.

A.2 Color Palette Organization

As stated previously, embedding requires the creation of smaller ordered quantizers from subsets of a larger ordered quantizer. While scalar quantizers have a natural ordering, vector quantizers as used for color quantization do not. Imposing an ordering, or organization, on a vector quantizer requires careful consideration. Binary index assignment for vector quantizers has been considered [22, 38, 83, 99] for joint

source/channel coding, progressive transmission, and in lossless compression of VQ indices. I adapt some of these binary index assignment techniques to order the color palettes.

Using two observations of the scalar quantizer example of Figure A.3 to motivate the ordering of color palettes. First, to embed in the LSB, the quantizer is divided into subsets that sparsely cover the *entire* grayscale space. The gray level of the embedded image is not correlated to the gray level of the output image. Secondly, to embed in the MSB, the scalar quantizer is divided into contiguous subsets that divide the grayscale space into separate, non-overlapping parts. In this case, the gray level of the embedded image influences the gray level of the output image; that is, a dark (or light) pixel in the embedded image results in a dark (or light) pixel in the higher bit-depth image.

Using these insights I develop two preprocessing techniques to order color palettes for LSB-like and MSB-like embedding. The first technique divides a color palette into subsets which sparsely cover the entire color space for LSB-like embedding. This will allow a search of a reduced set of the entire color space to find the best color representation for each output pixel. The second method, for MSB-like embedding, jointly orders both the M- and N-word palettes by dividing the N-word palette into subsets which divide the color space into separate, non-overlapping parts. The M-word palette is ordered so that each of its indices is a prefix to the subset of codewords in the N-word palette that occupies the same color space. This will permit the refinement of the color information of the smaller bit-depth halftone when creating the larger bit-depth color halftone.

Both of my techniques use Minimum Cost Perfect Matching (MCPM) [3, 65] from optimization theory, although other techniques, such as Principal Component Partitioning [74, 83], can be used in an analogous way. Given a graph of nodes $V_1, V_2, \ldots, V_N$ with associated costs $C_{i,j} = C_{j,i}$ between nodes V_i and V_j where $i \neq j$, the MCPM algorithm creates a perfect matching by minimizing the overall cost, where

$COST = \sum_{i=1}^{N/2} C_{i,j}$, of $\frac{N}{2}$ pairs of nodes $\{V_i, V_j\}$, where the $\frac{N}{2}$ pairings are disjoint and together comprise the entire set of nodes. The complexity of MCPM is approximately cubic with respect to the number of nodes. Good implementations exist which are efficient for the preprocessing of palettes especially since the number of nodes is very small. For a given quantizer $Q()$, it is assumed that the codewords q_i are the nodes of the graph, and that the costs between nodes are calculated as the increase in distortion due to merging one codeword with another, $C_{i,j} = \frac{W_i W_j}{W_{i,j}} \|q_i - q_j\|^2$ [37]. Note that the costs are weighted squared errors in the given color space and that the weights W_i are the number of training vectors that map into region i, and that $W_{i,j} = W_i + W_j$. Training data are required to compute the weights, and hence the costs. Ideally, the training data that are used to create the palette should also be used to order the palette.

A.2.1 Color Palette Organization for LSB Embedding

To embed in the LSB's, the N-word color palette is divided into subsets where each subset sparsely covers the color space. I begin by finding the minimum cost perfect matching for the N-word palette. I create two subsets with $\frac{N}{2}$ codewords by *breaking up* the $\frac{N}{2}$ MCPM pairs into two sets. The colors in each MCPM pair can be arbitrarily assigned to either set. Following the scalar quantizer example I use the following convention: the node from each pair that is closest to the origin is assigned to one set and the node that is furthest from the origin is assigned to the second set. MCPM is recursively applied to the equally-sized sets of codewords until M sets are found.

The Voronoi diagram of two dimensions (green and red) of the three-dimensional 256-word color palette of Section A.3 is shown in Figure A.4. The origin is in the upper left-hand corner. The two subset palettes with 128 colors that are produced after one iteration of the ordering algorithm for LSB embedding are shown in Figure A.5. Note that the two palettes have reproduction values throughout the entire input space and that, as expected, the Voronoi regions are larger than those for the original palette

since there are fewer reproduction vectors. It should be noted that the codewords of these new Voronoi regions are generally not centroids. They are centroids, however, with respect to the dense Voronoi regions of Figure A.4.

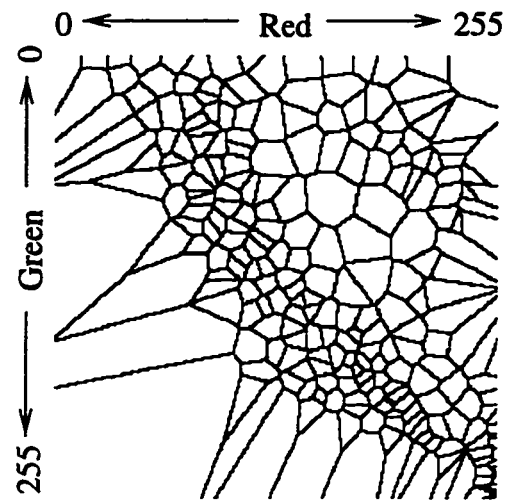


Figure A.4: The Voronoi regions of two dimensions (green and red) of the three-dimensional 256-word color palette of Section A.3. The origin is the upper left-hand corner.

A.2.2 Color Palette Organization for MSB Embedding

To embed in the MSB's, both color palettes are jointly ordered so that the index of each codeword in the M-word palette is the prefix of codewords with like colors in the N-word color palette. Assuming that there are two palettes of size $N = 2^K$ and $M = 2^{K-J}$, the N-word palette is split into M subsets of size $\frac{N}{M} = 2^J$ codewords where each subset covers a separate, non-overlapping region of the color space. To do this, I begin by finding the minimum cost perfect matching of the N-word palette. I create one set with $\frac{N}{2}$ codewords by *merging* the MCPM pairs [37,83]. MCPM is applied to the reduced-size set of codewords found at each iteration until the resulting reduced-size set has M members, where each member represents 2^J colors. At this time, the codewords of the N-word palette are indexed to indicate each perfect matching by

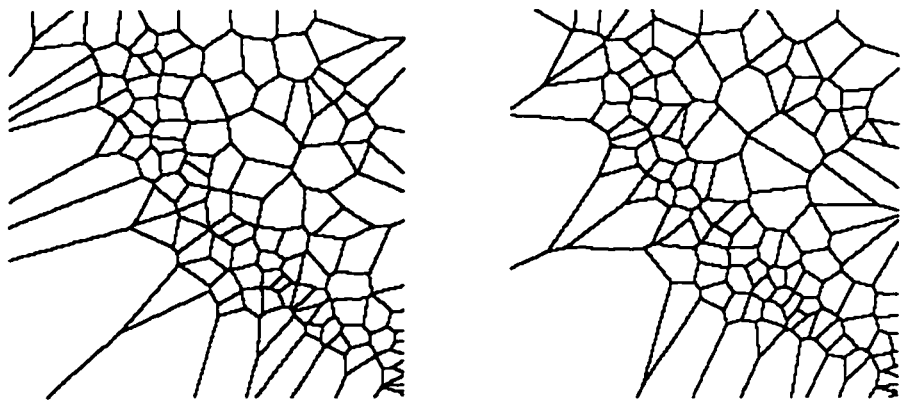


Figure A.5: The Voronoi regions of the two resulting subset palettes used for embedding in the LSB. The origin is the upper left-hand corner.

prefix. The last step of the algorithm is to assign indices to the M-word palette so that each color's index is a prefix to like colors in the N-word palette. MCPM is used to find a perfect matching between the M-word palette and the M-word set of the merged N-word palette. This matching correlates the colors of the two palettes, and final index assignments are made from this. Note that if there are no color palettes for the first embedding stages, color palettes can be created by using the merged palettes.

The two subset palettes of Figure A.4 with 128 colors that are produced after seven iterations of the ordering algorithm for MSB embedding are shown in Figure A.6. Note that the two palettes have reproduction values in entirely different parts of the input space and that in the dense regions, except for the edges, the Voronoi regions are the same as those for the original palette.

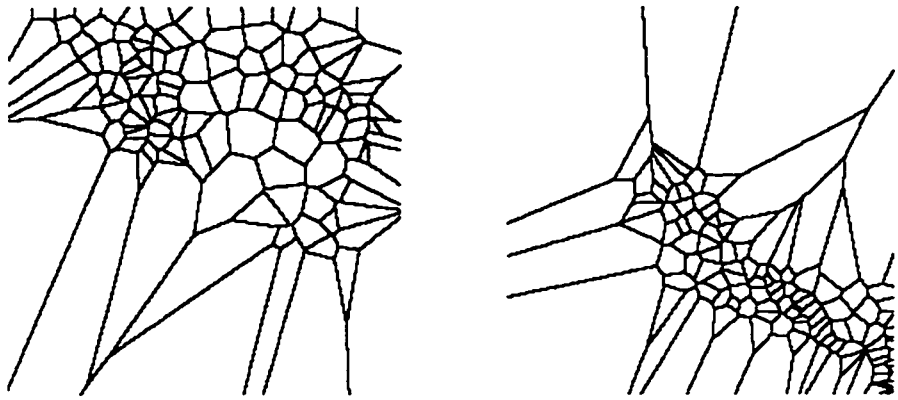


Figure A.6: The Voronoi regions of the two resulting subset palettes used for embedding in the MSB. The origin is the upper left-hand corner.

A.3 Results

A three-dimensional full-search color palette (red, green, blue) with 256 codewords was created using the generalized Lloyd algorithm [69]. The training images used are the airplane, baboon, sailboat, splash, and Tiffany 24-bit color images. Using the same training data, the color palette was ordered for both LSB and MSB embedding. The two ordered palettes are shown in Figure A.7 where each half, quarter, eighth, and so on, portion of the palette is a subset used as an embedding quantizer. Note that the subsets of the LSB ordered palette, Figure A.7(a), have a variety of colors while the subsets of the MSB ordered palette, Figure A.7(b), have a restricted range of colors. Grayscale and binary images are obtained by using the Y color component of the YIQ color space of the image [49]. Uniform scalar quantizers are used for grayscale and binary error diffusion. The error diffusion mask used is that of [39], although using optimized filters for certain input devices would likely yield better results [63, 64]. Results shown are for the 24-bit color Lenna image. The 8-bit color error diffused Lenna image is shown in Figure A.8.

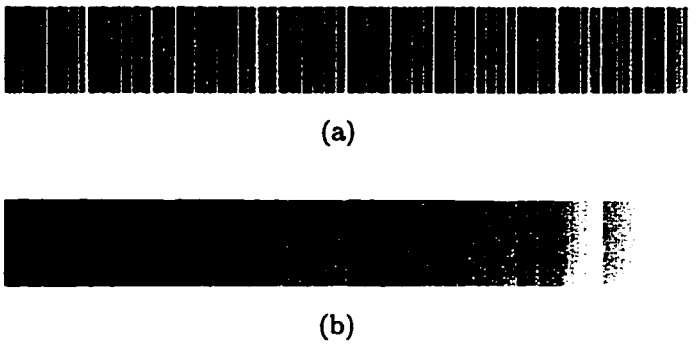


Figure A.7: The 8-bit color palette as ordered for (a) LSB embedding and (b) MSB embedding where each half, quarter, eighth, and so on, portion of the palette is a subset used as an embedding quantizer. Note that the subsets of the LSB ordered palette have a variety of colors while the subsets of the MSB ordered palette have a restricted range of colors.

A.3.1 Results of embedding in the LSB's

The results of embedding with an LSB ordered palette are shown in Figure A.9(a) where binary as well as 2-, 3-, ..., and 7-bit color images are embedded into the LSB's of an 8-bit color halftone. In general, I find that as the bit depth of the embedded image decreases, visual results improve. In fact, the 8-bit color error diffused image with the embedded binary halftone shows no visual difference when compared to the 8-bit color error diffused image. As the bit depth of the embedded image increases, the output image quality degrades by the introduction of inappropriate colors which are outside the natural color space. Similar results arise when working with grayscale images where it is seen that stippling or thumbprint patterns intensify and become more prevalent as the relative bit depth increases.

The differences in image quality can be explained by recalling that the embedding quantizers are created from subsets of the N-word palette each of which have color representations throughout the entire color space. The good visual quality that is found when M is small relative to N is due to embedding quantizers that densely cover

Figure A.8: The 8-bit color error diffused Lenna image. The palette of 256 colors was generated using the generalized Lloyd algorithm with 5 training images.

the entire color space making it possible to find good color representations for any given input pixel. In effect, this permits the color distribution of the original image to influence the choice of the N output colors while minimizing the influence of the distribution of the colors of the embedded halftone on this choice. The poor quality that results when M is large relative to N results from the embedding quantizers having very few color representations across the entire color space. Thus a given input color pixel may not find a good color representation among any of the M embedding quantizers.

A.3.2 Results of embedding in the MSB's

The results of embedding with an MSB ordered palette are shown in Figure A.9(b) where binary as well as a 2-, 3-, ..., and 7-bit color images (the same used in the LSB case) are embedded into the MSB's of an 8-bit color halftone. In general, I find that as the bit depth of the embedded image increases, visual quality improves. In fact, the 8-bit color error diffused image with the embedded 7-bit color halftone

(a)

(b)

Figure A.9: The 8-bit color error diffused images containing (from left to right) a binary halftone, a 2-bit, a 3-bit, ..., and a 7-bit color halftone when (a) embedding in the LSB's using the LSB palette arrangement and when (b) embedding in the MSB's using the MSB palette arrangement.

shows little visual difference from the original 8-bit error diffused image. As the bit depth of the embedded image decreases, visual quality degrades by the introduction of "thumbprint" artifacts and unnatural color use. Similar results arise when working with grayscale images where it is seen that thumbprint artifacts intensify and become more prevalent as the relative bit depth decreases. It should be noted that for the color images, the degradation is not continuous as is the case for embedding in the LSB's, but that as M goes to 2 the artifacts are present to a lesser extent. Enlarged portions of the 8-bit color halftone, Figure A.8, and the leftmost images of Figure A.9 are shown in Figure A.10 to illustrate the difference between embedding a binary image in the LSB and MSB using both the LSB and MSB palette orderings respectively [44]. Note in particular the introduction of yellow colors on the side of Lenna's face and lips as well as bluish specks in the hair and lips when embedding in the MSB's,

Figure A.10(c). Again it is seen that embedding a binary image in the LSB causes little or no visible difference in quality.

(a) (b) (c)

Figure A.10: Artifacts seen in enlarged portions of: (a) the 8-bit color error diffused image; (b) the 8-bit color error diffused image embedded with a binary image in the LSB's using the LSB palette arrangement (the leftmost image from Figure A.9(a)); and (c) the 8-bit color error diffused image embedded with a binary image in the MSB's using the MSB palette arrangement (the leftmost image from Figure A.9(b)).

The differences in image quality can be explained by recalling that the embedding quantizers are created by dividing the color space into M separate color regions linked to the M colors of the smaller palette where the output pixel of the first stage of embedding selects which of the M embedding quantizers to use. When M is small relative to N, the color space is crudely divided into a few quantizers with a large number of output levels in one section of the color space. As there is little color information in the embedded halftone, an input pixel cannot always find a good color representation since the pixel color may not be in the same color space as the embedding quantizer. However when M is very small, such as 2 or 4, there may be so many color representatives in each embedding quantizer that it is possible for a reasonable color match to be found most, but not all, of the time. The good results that arise when M is large relative to N, are due to the M embedding quantizers

representing a very small portion of the color space, and there being much color information in the embedded image. Each pixel color of the embedded image directs the choice of output image colors to a similar color. In effect, this permits the color distribution of the smaller bit-depth halftoned image to influence the choice of the N output colors while minimizing the influence of the color distribution of the original image. Thus, if the embedded image looks good, the color halftone containing it will look good as well since it is a refined version of the embedded image.

A.3.3 Limitations to Color Palette Use

Note that for the scalar quantizer example, embedding can be done in both the MSB's and LSB's when using the same ordered scalar quantizer. This implies that a color palette, when ordered for either MSB or LSB embedding for all embedding possibilities (e. g. an 8-bit palette ordered for embedding any of 1-, 2-, ..., or 7-bit images), could be used to do both MSB and LSB embedding too. If the palette were designed for LSB embedding, it would be necessary to correlate the colors of the smaller palettes to the MSB subsets of the larger LSB ordered palette, and this could be done by using the centroid of each larger subset. Alternatively, LSB embedding can be done directly off of the MSB ordered palette. In practice however, it is not possible to use an LSB ordered palette to do embedding in the MSB since the recursive creation of quantizers that span the color space destroys any groupings of similar colors which are originally found by MCPM. It is possible to use an MSB ordered palette to do embedding in the LSB, since the similar color groupings easily break down into LSB subsets that have colors throughout the entire color space. In fact, since MCPM is used as the first step of both palette ordering techniques, the LSB subsets for embedding a binary image are identical for either ordering technique. However, in this case, the color image quality degrades much more rapidly as M increases than when embedding in the LSB's using a palette ordered for LSB embedding. Since color palettes have no natural ordering, the results vary for each ordering imposed.

The best results are achieved by using palettes which are ordered for the desired embedding scheme.

A.3.4 Limitations to Embedded Error Diffusion

I next discuss limits to the amount of embedding that can be done in a single image. To embed as many halftones as possible, an N-level "fully-embedded" error diffused image [44] could be produced, which contains the $\frac{N}{2}$-, $\frac{N}{4}$-, ..., and 2-level fully-embedded error diffused images, where $N = 2^K$, so that if one bit is removed from the N-level output image, the $\frac{N}{2}$-level fully-embedded error diffused image is preserved. To generate this image, $K = \log_2(N)$ stages of embedding must be done where the embedding quantizers at each stage have only two output levels. Color image generation by a series of thresholding operations makes little sense. However, fully-embedded grayscale images can be created, and these can be embedded into a color halftone. For example, a 16-level fully-embedded grayscale image could be embedded into the LSB's of an 8-bit color image. This would be useful for progressive transmission as well as for a browsing system where different types of color and grayscale monitors (2-, 4-, 16- or 256-level displays) would be used to view the image.

When making fully-embedded grayscale images, the best results are obtained when embedding only in the MSB's. This produces an image that retains some of the thumbprint artifacts of binary error diffusion but is of good visual quality, improving with each successive level of embedding. This can be viewed as a refinement of the binary image as the bit-depth increases. The thumbprint artifacts are retained because the multilevel image is always dark (light) where the binary image is black (white). The worst results are obtained when full embedding is done in the LSB's, which produces a grainy image that gets worse with each successive level of embedding. Other combinations of LSB/MSB embedding at different stages of the algorithm produce images that vary in quality between the MSB's-only embedded image and the LSB's-only embedded image.

A.4 Conclusions

I have presented an algorithm for embedding the output of binary Floyd and Steinberg error diffusion into the output of grayscale and color error diffusion; for embedding grayscale halftones into higher bit-depth grayscale and color halftones; and for embedding color halftones into higher bit-depth color halftones. I have also presented preprocessing algorithms to organize color palettes for effective use of the color space when embedding halftones into color halftones. I have shown that embedding relatively small halftones works best for grayscale and color images when the N-word palette is ordered for LSB embedding, and that embedding relatively large halftones works best when the N-word palette is ordered for MSB embedding.

To embed an M-level halftone into an N-level halftone with good visual results, I draw the following conclusions. If M is small relative to N, the N-level palette should be ordered in an LSB-like arrangement where palette subsets have colors throughout the entire color space. The M-level image has little color or grayscale information which can be used, so it can easily be hidden in the N-level halftone. If M is large relative to N, and the smaller bit-depth image is a grayscale image while the larger bit-depth image is color, then the M-level grayscale halftone has a relatively large amount of grayscale information which cannot be easily integrated into the color halftone, and there is no clear solution. However, if both halftones are to be color (or grayscale) images, then the palette should be ordered in an MSB-like arrangement where subsets divide the color space into separate, non-overlapping parts. The M-level halftone has color information which can be used to create an N-level image that is a refined version of the M-level image.

VITA

Jill R. Goldschneider

Master of Science in Electrical Engineering awarded June 1991 from the University of Washington. Thesis Topic: *"Performance Analysis of Banyan Based Copy Networks."*

Bachelor of Science in Electrical and Computer Engineering, with High Honors, awarded June 1989 from the University of California, Santa Barbara.

Publications

"Wavelet Packet Bit Allocation," in preparation.

"Embedded Multilevel Error Diffusion," J. R. Goldschneider, E. A. Riskin, P. W. Wong. To Appear, July 1997, IEEE Transactions on Image Processing. Special Issue on Color.

"Lossy Compression of Acoustic Backscatter Data," J. R. Goldschneider, A. G. Bruce, D. B. Percival. Proceedings of SPIE AeroSense '97, April 1997.

"Algorithms for Wavelet Compression of Acoustic Signals," A. G. Bruce, D. B. Percival, and J. R. Goldschneider. NAVAIR Phase I Final Report N00421-95-C-1148, MathSoft Inc., 1700 Westlake Ave. N., Suite 500, Seattle, WA 98109, 1996.

"Embedded Color Error Diffusion," J. R. Goldschneider, E. A. Riskin, P. W. Wong. Proceedings of ICIP, September 1996. Vol. 1, pp. 565-568.

"Bit Allocation via Recursive Optimal Pruning with Application to Wavelet/VQ Image Compression." J. R. Goldschneider, E. A. Riskin. Proceedings of ICASSP, May 1996. Vol. IV, pp. 2339-2342.

"Embedded Multilevel Error Diffusion," J. R. Goldschneider, E. A. Riskin, P. W. Wong. Proceedings of SID, May 1995. Vol. XXVI, pp. 829-832.

"On Vector Quantization for Fast Facet Edge Detection," M. Y. Jaisimha, J. R. Goldschneider, A. Mohr, E. A. Riskin, R. M. Haralick. Proceedings of ICASSP, April 1994. Vol. V, pp. 37-40.