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.

U·M·I

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

Order Number 9410668

**Image coding using adaptive recursive interpolative DPCM with
entropy-constrained trellis coded quantization**

Gifford, Eric Allan, Ph.D.

The University of Arizona, 1993

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

**IMAGE CODING USING ADAPTIVE RECURSIVE
INTERPOLATIVE DPCM WITH ENTROPY-CONSTRAINED
TRELLIS CODED QUANTIZATION**

by
Eric Allan Gifford

A Dissertation Submitted to the Faculty of the
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
In Partial Fulfillment of the Requirements
For the Degree of
DOCTOR OF PHILOSOPHY
WITH A MAJOR IN ELECTRICAL ENGINEERING
In the Graduate College
THE UNIVERSITY OF ARIZONA

1 9 9 3

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

As members of the Final Examination Committee, we certify that we have
read the dissertation prepared by Eric Allan Gifford

entitled Image Coding Using Adaptive Recursive Interpolative
DPCM With Entropy-Constrained Trellis Coded Quantization

and recommend that it be accepted as fulfilling the dissertation
requirement for the Degree of Doctor of Philosophy

B. R. Hunt

11-8-93

Date

Michael W. Maxwell

11/8/93

Date

R. Schomenger

11/8/93

Date

Date

Date

Final approval and acceptance of this dissertation is contingent upon
the candidate's submission of the final copy of the dissertation to the
Graduate College.

I hereby certify that I have read this dissertation prepared under my
direction and recommend that it be accepted as fulfilling the dissertation
requirement

B. R. Hunt

11-8-93

Dissertation Director

Date

Dr. Bobby R. Hunt

STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: _____

Eric Gifford

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my friend and advisor Dr. Bobby R. Hunt for his continued support and guidance over the past four years, without which this work and my degree would not have been possible. I would like to thank Dr. Michael Marcellin for his insight and persistence. I would like to thank my fellow graduate students Glen, Phil, Dave, Jim, and of course Mari aka "Ralph".

On a personal level, I would like to acknowledge a great debt of gratitude to my parents for their continual support, encouragement, and confidence. I would like to thank Debbie for just being Debbie.

TABLE OF CONTENTS

LIST OF FIGURES	7
LIST OF TABLES	9
ABSTRACT	10
1 INTRODUCTION	12
1.1 Problem Statement	12
1.2 Historical/Contemporary Uses of Imagery	14
1.3 Image Properties	16
1.4 Data Compression Algorithms	18
1.5 Motivation for Adaptive RIDPCM-TCQ	28
2 RIDPCM	30
2.1 Interpolation Theory	30
2.2 RIDPCM Algorithm	32
2.3 Testing/Results	43
3 ADAPTIVE CLASSIFICATION	45
3.1 Problem Statement	45
3.2 Previous Implementations of Adaptive RIDPCM	46
3.3 Current Implementation of Adaptive RIDPCM	49
3.4 Adaptivity: Feature Extraction	51
3.5 Adaptivity: Classifier Training	54
3.6 Adaptivity: Classification	55
3.7 Classifier Implementation	57
4 QUANTIZATION	62
4.1 Quantization Theory	62
4.2 Vector vs Scalar Quantizers	64
4.3 TCQ	70
4.4 ECTCQ	79
4.4.1 Uniform Codebook TCQ	82
4.4.2 Optimized ECTCQ	84
4.4.3 Entropy Coding	86

TABLE OF CONTENTS – *Continued*

5	RATE ALLOCATION	87
5.1	Theory	87
5.2	Adaptive RIDPCM (ARIDPCM) Model	88
5.3	Integer Programming Solution for TCQ	95
5.4	Conjugate Gradient Solution for ECTCQ	96
5.5	Encoding Rate Computation	97
6	ADAPTIVE RIDPCM-TCQ	99
6.1	Implementation	99
6.2	Decoder Complexity	101
6.3	Performance	103
7	ADAPTIVE RIDPCM-ECTCQ	108
7.1	Implementation	108
7.2	Decoder Complexity	109
7.3	Performance	110
8	RESULTS	116
8.1	Visual Quality	120
8.2	Other Compression Systems	126
9	CONCLUSIONS AND FUTURE RESEARCH	133
	REFERENCES	138

LIST OF FIGURES

1.1	Test Images:(a)Lena, (b)Paglady, (c)Urban26, (d)Urban28	19
1.2	Digital Communication System	22
1.3	Source Encoder	22
2.1	RIDPCM Encoder	35
2.2	RIDPCM in 2D	36
2.3	RIDPCM Decoder	37
2.4	2-D Interpolation Filter	39
2.5	1-D Interpolation Filter	39
2.6	RIDPCM Performance	41
2.7	RIDPCM at 1 bit: (a)Lena, (b)Urban26, (c)Lena Difference Image, (d)Urban26 Difference Image	42
3.1	Adaptive RIDPCM Performance	50
3.2	Subimage Classifier	58
3.3	Classification Images: (a)Lena, (b)Paglady, (c)Urban26, (d)Urban28	61
4.1	Voronoi Regions	67
4.2	Scalar Quantizer	69
4.3	4-State Machine	72
4.4	State Transition Diagram	72
4.5	4-State Trellis	73
4.6	Rate 1/2 Convolution Coder	75
4.7	TCQ Performance for Generalized Gaussian Sources	80
4.8	Rate Distortion Function	80
4.9	Uniform Codebook TCQ Performance	83
4.10	Entropy-Constrained TCQ Performance	83
5.1	Encoding Performance for TCQ and ECTCQ, $\alpha = 0.75$	91
5.2	Distortion Model for ECTCQ	91
5.3	Estimated Variances	92

LIST OF FIGURES – *Continued*

6.1	Adaptive RIDPCM-TCQ Encoder	100
6.2	Encoder Quantization Algorithm	102
6.3	Adaptive RIDPCM-TCQ Decoder	104
6.4	1 bit ARIDPCM-TCQ: (a)Lena, (b)Urban26, (c)Lena Difference Image, (d)Urban26 Difference Image	106
6.5	Adaptive RIDPCM-TCQ Performance	107
7.1	Adaptive RIDPCM-ECTCQ Performance	113
7.2	1 bit ARIDPCM-ECTCQ: (a)Lena, (b)Urban26, (c)Lena Difference Image, (d)Urban26 Difference Image	114
8.1	RIDPCM, ARIDPCM-TCQ, ARIDPCM-ECTCQ: Lena	117
8.2	RIDPCM, ARIDPCM-TCQ, ARIDPCM-ECTCQ: Paglady	117
8.3	RIDPCM, ARIDPCM-TCQ, ARIDPCM-ECTCQ: Urban26	118
8.4	RIDPCM, ARIDPCM-TCQ, ARIDPCM-ECTCQ: Urban28	118
8.5	Lena at 0.5 bits: (a)RIDPCM, (b)ARIDPCM-ECTCQ, (c)RIDPCM Difference Image, (d)ARIDPCM-ECTCQ Difference Image	122
8.6	Difference Image Histogram: RIDPCM	124
8.7	Difference Image Histogram: ARIDPCM-ECTCQ	124
8.8	Lena at 0.5 bits: (a)ARIDPCM-TCQ, (b)ARIDPCM-ECTCQ, (c)TCQ Difference Image, (d)ECTCQ Difference Image	125
8.9	ARIDPCM-ECTCQ Lena: (a)1.0 bits, (b).75 bits, (c).54 bits, (d).33 bits	127
8.10	Lena: ARIDPCM-ECTCQ, DCT-ECTCQ, DWT-ECTCQ	129

LIST OF TABLES

2.1	Optimal Filter Coefficients	32
2.2	RIDPCM Computations	40
3.1	Trained Classifier: Portrait and Urban Images	59
3.2	Classification: Portrait and Urban Images	60
4.1	TCQ Performance for Gaussian Source Using Doubled and Optimized Codebooks	78
4.2	Computational Requirements of TCQ and VQ for Memoryless Sources	78
5.1	Integer Programming Solution for Lena at 1 Bit	96
5.2	Conjugate Gradient Solution for Lena at 1 Bit	97
7.1	Predicted vs. Actual Encoding Statistics, Lena at 1-Bit	111
7.2	Residual Correlation of Error Sequences	112
8.1	ECTCQ Rate Table, Lena 1-Bit	119
8.2	ECTCQ Rate Table, Urban28 1-Bit	120

ABSTRACT

The goal of image coding is to represent images with a minimum amount of distortion at a given encoding rate. Image coding algorithms comprise methods for generating uncorrelated sequences and quantizing the uncorrelated sequences. The earliest encoding algorithms, such as Differential Pulse Code Modulation, are prediction based and must be considered primitive when compared to the more recent transform coders, such as Discrete Cosine Transform or Discrete Wavelet Transform. Judged only by SNR performance, the contemporary transform coders are far superior to the predictive coders. However, the computational complexity of the transform coders is much greater than predictive coders. In general, the improvement of hardware has diminished the importance of computational complexity. Thus, little research has been devoted to improving the performance of predictive coders. Furthermore, in a few applications such as remote decoding or real-time video decoding, the complexity of the decoder is still a constraint.

In this dissertation, I have developed a predictive image coder having minimal decoder complexity and providing SNR's in the range of the most advanced transform coders. The image coder utilizes the Recursive Interpolative DPCM algorithm as a kernel in conjunction with an adaptive rate allocation scheme and entropy-constrained trellis coded quantization. The Adaptive RIDPCM-ECTCQ image coder

is a high performance, low decoder-complexity alternative to contemporary transform coders.

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

Image data compression systems have exhibited an increasing level of sophistication and performance during the past twenty five years of research and development. The earliest compression systems, such as Differential Pulse Code Modulation (DPCM), must be considered as primitive and showing only modest compression performance when compared to some of the more recent schemes involving transforms, such as discrete Cosine or Wavelet transform. This increased performance does not come without cost, however. The computational complexity of modern compression algorithms, such as those based on transforms, is much greater than the computational complexity of DPCM. This is true for the decoder as well as the encoder. Transform based algorithms require much more memory than DPCM, as well as requiring many more computations per pixel of the image coming into the compression algorithm.

Judged only by performance, e.g., compression ratio and Signal-to-Noise-Ratio (SNR), modern compression systems favor the usage of transform-based algorithms. The implicit assumption is that computational costs of transforms can be met by

the continued improvement in VLSI semiconductor digital logic. Experience has certainly justified this assumption. However, a “contrarian” view leads to the question: are the transform-based algorithms the only approach to achieving high-performance data compression? One strong distinction between DPCM and transform-based systems is their domain of operation. DPCM operates in the domain space of the original image, whereas transform techniques operate in the domain (e.g., Fourier, Wavelet) constructed by the particular transform. Thus, another way to rephrase the question above is: can space-domain methods of image data compression be as effective in performance as transform-domain methods? A related question also arises: what is the relative computational complexity of advanced space-domain compression methods, as compared to corresponding transform-domain methods?

In this dissertation I present results that contribute to answering these questions. A particularly effective interpolative compression technique that is easily made adaptive has been developed. Combining this adaptive interpolative coding with the most recent advance in data quantization, the trellis coded quantizer (TCQ), yields an image data compression system, which operates solely in the space-domain of the original image, with performance in the range of the most advanced transform domain schemes. The computational complexity of the encoder in this adaptive space domain scheme is moderate and the decoder complexity minimal in comparison to the competing transform-domain techniques, emphasizing the possibilities for efficient data compression without transforming out of the original image space. The algorithm is

particularly well suited for applications placing stringent complexity limitations on the system decoder.

1.2 Historical/Contemporary Uses of Imagery

In its earliest form, imagery consisted of black and white photographs. The quality of the photographs was, by today's standards, very poor, being limited to black and white, and by the construction of the camera and quality of the film. Furthermore, the utility of the photograph was severely constrained, by today's standards. Copies or enlargements were only available through photographic processing. Even with these restrictions, photographs were a very important part of the public, commercial, and military culture. From personal pictures to newspapers to historical documents to the first military reconnaissance pictures taken from hot air balloons, imagery has played an ever increasing and complex role.

A major breakthrough in handling image data was the advent of the digital computer and associated digital processing algorithms. Now, images in standard photographic format could be scanned into the digital format of the computer thus creating seeming endless opportunities. The images could be stored in memory forever without fading, tearing, or getting lost, and retrieval required minimal effort. The same images could be transmitted over a network to as many other computers as desired in very little time, where the image could then be printed, viewed on the screen, or stored for later use.

Digital image data may also be processed or transformed to improve its overall visual quality or to extract or enhance specific information or features in the imagery. There exist noise suppression, edge enhancement and contrast enhancement algorithms for enhancing the visual quality of images. There are algorithms for identifying line, area, and point features as well as advanced pattern recognition and artificial intelligence algorithms for identifying specific objects or groups of objects.

The function of these algorithms fall into two major areas: techniques for improving the visual or aesthetic quality of the images, and techniques for providing very specific enhancement of information for analysis. The military and commercial business interests provide a great demand for high quality digital imagery that can be stored, transmitted, and processed efficiently and with a great deal of flexibility.

Within the military, the intelligence agencies use a tremendous amount of image processing in satellite and aerial reconnaissance, enhanced video is being employed in smart bombs, pre-programmed maps for missile guidance, etc. Reconnaissance photographs may be noise filtered and enhanced to improve overall visual quality. They may also be processed to identify specific features such as runways, planes, thermal properties of waterways near industrial complexes, etc. Typically, military systems place a premium on reliability and performance, and have correspondingly high price tags for the research and development, and hardware required to implement such systems.

Commercial business applications are typically targeted towards the visual or aesthetic quality of the images. The object of any processing may be to retain or restore as much of the original image quality as possible. This might include digitized portrait photographs, satellite image communications, HDTV, or interactive video. In business applications, performance per dollar is crucial.

The different functions of visual quality and analysis, and the different demands placed by the military and business interests, create a very wide range of image processing applications. Common throughout image processing, and of primary importance to holding costs down and maintaining image quality, is the area of data compression. By compressing image data, it can be stored, transmitted, and sometimes processed in a much more efficient manner. One cost of compression can be a loss of image quality. The problem is to maximize compression and image quality while holding algorithmic complexity and cost down.

1.3 Image Properties

It is often said that “A picture is worth a thousand words.” This might be true, but unfortunately a picture, in terms of computer memory, may also cost far more than a thousand words. A typical grey scale image of size 512 by 512 pixels and an original quantization of eight bits per pixel requires a quarter of a megabyte of memory to store. By today's standards this might not appear to be all that much, and might not be if your desire is to only have a couple of images available. But

what if you would like to have digitized versions of all the renaissance art works on your computer, or if your application requires detailed aerial imagery of all of South America, or hyperspectral imagery, e.g., approximately 200 images taken at varying wavelengths of the same scene, or an interactive video application. Under any of these scenarios the volume of data is immense and if not impossible to store, transmit or process, at the very least costly and cumbersome. The volume of data provides the motivation for developing encoding algorithms that can represent the images in a reduced number of bits while preserving the integrity of the image.

The inherent structure of natural or man-made images provides the means for achieving image compression. Of course, images of any interest, convey information. Far from being random, images are comprised of structure and features that characterize the image. Properties such as the variance, entropy, texture, edge concentration, contrast etc. are measurable quantities of imagery. The features and structure convey a measure of correlation between any pixel and its neighbors. Correlated data represents redundant information and therefore directly encoding correlated data wastes bits. The effectiveness of an algorithm in removing the redundancy from an image, prior to coding, is central to its performance. Additionally, the structure and features are spatially varying. Hence, the correlation changes throughout the image. By adapting compression algorithms to the local statistics, it is possible to improve performance.

The features discussed, and their spatially varying nature, are visible in Figure 1.1, comprising the four images used for testing purposes in this dissertation. The first two (“lena” and “paglady”), are portrait type images, the second two (“urban26” and “urban28”), are aerial photographs of urban areas in the Bay Area near San Francisco. These images are typical of those used in business and military applications.

1.4 Data Compression Algorithms

Effective data compression algorithms are vital to efficient storage, transmission, and processing of data, while maintaining image quality. The importance and complexity of efficient compression algorithms can be hard to convey to the user since they are essentially transparent. Simply put, an image is compressed, processed, and then decompressed with the output being visually and/or mathematically as “close” to the original as possible. In colloquial terms, you want to maximize the quality of the reconstructed image while minimizing the bit rate of the compressed image.

By definition, a lossless system requires the reconstructed signal to be exactly equal to the original signal. In a lossy system, the output is a distorted version of the input signal. Intuitively, it is possible to get far greater compression from a lossy system than a lossless one. In many image compression systems small to large errors can be allowed in the reconstructed image without seriously degrading the image quality or impairing the analysis of the image. The human visual system may not

Figure 1.1: Test Images:(a)Lena, (b)Paglady, (c)Urban26, (d)Urban28

even detect many small errors and depending on the purpose of the system, large errors may even be tolerable.

The relation between the amount of achievable compression and the distortion incurred in the reconstructed image is direct and prescribed by fundamental limits of information theory. These limits on data compression were set forth by Claude Shannon [1] in theorems relating to lossless as well as lossy compression. For lossless compression, a code can be found with arbitrarily small error probability provided that the rate exceeds the entropy of the source. Therefore, the entropy is a lower bound on the error free coding of the source. For lossy compression, the rate distortion function for a given source determines the minimum distortion for a given rate.

Figure 1.2 is a block diagram of a typical digital communication system. A signal is input to the transmitter where the signal is coded by a source encoder. The encoded signal passes through a channel encoder, which codes the signal in accordance with the constraints of the channel. The encoded signal is then passed through a modulator and into the channel. All real channels are noisy and corrupt the signal. The signal emerges from the channel and is provided as input to the receiver where it is demodulated and sent to the channel decoder. The decoder produces estimated source codewords which are provided to the source decoder for reconstruction of the quantized signal. If there is no channel noise, then any distortion is created by the source encoder/decoder (e.g., data compression). If a lossless compression system is employed, then the entire communication system is lossless.

Figure 1.3 is a block diagram of the source encoder comprising a data compression coder and a data compaction coder. By definition, the compaction encoder refers to the lossless compression and the compression coder refers to the lossy compression. Typically, there is only a single encoding algorithm which is referred to as a compression algorithm if the system is lossy and as a compaction algorithm if the system is lossless.

Compression algorithms can be divided into the classes: predictive algorithms, transform algorithms, and Vector Quantization algorithms. Predictive algorithms were developed first and follow from the correlation properties of imagery. For example, the pixel values on the flesh of a human face are similar as are the values in a lake or wheat field. Knowing the value of one pixel allows you to make an intelligent prediction at the values of neighboring pixels. Any single prediction might be inaccurate, but statistically you can do much better than randomly selecting a value. Typically, predictive algorithms are conceptually and computationally very simple. The newer transform algorithms are much more mathematically based and typically provide much better performance. Transform algorithms decorrelate a signal by projecting the signal onto orthogonal basis functions. Transform algorithms entail many more computations per pixel in both the encoder and the decoder than do the basic predictive algorithms. In general, this is not viewed as a deterrent because of the recent increases in hardware capabilities. Vector Quantizers map input blocks of data

Figure 1.2: Digital Communication System

Figure 1.3: Source Encoder

directly into code vectors. According to Shannon's theory for source coding [1], VQ is optimal but the complexity associated with superior performance is prohibitive.

The simplest coding scheme is pulse code modulation or PCM. In PCM each pixel value is simply quantized, assigned a codeword, and transmitted independent of all other pixel values. The decoder simply utilizes a look-up table (LUT) that provides the quantization level for each codeword to match the incoming codewords with its quantized value. This algorithm does not utilize the correlation of the image and is very inefficient.

Differential Pulse Code Modulation (DPCM) [2] - [6] utilizes the correlation of the signal to form a prediction of the actual pixel values. Taking the difference between the original and predicted signal values creates a difference signal. The difference signal is quantized and transmitted. At the decoder, the signal is reconstructed by forming the same prediction signal as at the encoder, and summing the prediction values and the quantized difference signal. The error in the reconstructed signal is equal to the quantization error of the difference signal. Therefore minimizing the quantization error, minimizes the overall reconstruction error.

Ideally, the encoder completely decorrelates the signal. The major drawback to prediction encoders including DPCM is that the difference signal is not completely decorrelated. However, the variance of the difference signal is much smaller than the variance of the original signal. The distortion in a quantized signal is modeled as $D = \sigma^2 2^{-2R}$ where σ^2 is the variance of the signal and R is the bit rate. Since

the variance of the difference signal is less than the variance of the original signal, a smaller rate (fewer bits) is required to achieve the same distortion.

Transform coders in general, and the DCT in particular, perform well because they decorrelate the signal more effectively than prediction codes. The Discrete Cosine Transform or DCT [7] operates on a $N \times N$ block of image pixels to create a $N \times N$ block of transform coefficients. These coefficients are quantized and transmitted. The decoder performs an inverse DCT on the quantized coefficients to produce the reconstructed image. Equations 1 and 2 are the DCT pair.

$$\theta(k, l) = \frac{2}{N} \alpha(k) \alpha(l) \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} x(m, n) \cos \frac{\pi k(2m+1)}{2N} \cos \frac{\pi l(2n+1)}{2N} \quad (1.1)$$

$$x(m, n) = \frac{2}{N} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} \alpha(k) \alpha(l) \theta(k, l) \cos \frac{\pi k(2m+1)}{2N} \cos \frac{\pi l(2n+1)}{2N} \quad (1.2)$$

where $x(m, n)$ is the $(m, n)^{th}$ pixel data point, $\theta(k, l)$ is the $(k, l)^{th}$ transform coefficient, $\alpha(0) = \frac{1}{\sqrt{2}}$, and $\alpha(j) = 1, j \neq 0$. Note: the transform kernels are separable so that the 2-D DCT can be efficiently performed in two steps using 1-D DCT's.

The signal is represented as a sum of cosines (i.e., orthogonal basis functions). Each coefficient represents a different frequency component of the signal. With the DCT, a vast majority of the energy in a signal is represented by the first few coefficients with the DC component dominating all others. By intelligently allocating bits to the individual components based on the strength of each coefficient, the distortion of the reconstructed image for a given rate will be lower for the DCT than

for DPCM. Visual quality is also better because the quantization errors in the coefficients are distributed evenly throughout each DCT block by inverse transforming the quantized coefficients whereas quantization errors in DPCM are localized by the prediction. Once again, this improvement in performance has an attendant cost, a sharp increase in encoder and decoder complexity and memory requirements. The DCT and inverse DCT require far more computations than DPCM.

The wavelet coder [8], [9], [10] is another transform coder that is becoming popular for its excellent performance. The wavelet transform provides a method for looking at a signal at various scales and analyzing it with various resolutions. The wavelet transform uses short windows at high frequencies and long windows at low frequencies, so called constant Q filters. Mathematically, the WT is a signal decomposition onto a set of basis functions (i.e., wavelets). The wavelets are obtained from a single prototype wavelet by shifting and scaling. The prototype is a band-pass filter, and the constant Q property is maintained by scaling the prototype.

To maintain constant Q, $\frac{\Delta f}{f} = c$ where f is the center frequency of the filter. Therefore, the time resolution becomes arbitrarily good at high frequencies, while the frequency resolution become arbitrarily good at low frequencies. The continuous WT is:

$$CWT_X(\tau, a) = \frac{1}{\sqrt{|a|}} \int x(t) h^*\left(\frac{t - \tau}{a}\right) dt \quad (1.3)$$

where $h(t)$ is the prototype wavelet and a is the scale factor.

As applied to signal compression, the WT is used to form successive approximations together with “added detail” to represent the signal. In a subband coder using the WT, the original signal is filtered into a low-pass and a high-pass signal. The high-pass signal is subsampled and transmitted, the low-pass signal is subsampled and filtered into its high and low pass components, etc. The subband coding system corresponds to a decomposition onto an orthonormal basis (wavelets). This sub-band encoder provides excellent signal decorrelation. Combined with an efficient method for encoding the individual sub-bands, this method produces excellent SNR’s at very low rates. The wavelet transform does require iterative filter applications both to decompose the signal in the encoder and to reconstruct the quantized signal at the decoder.

Vector Quantization or VQ [11],[12] is a system for mapping sequences of vectors into discrete code vectors. As detailed in Shannon’s theory of source coding, vector quantizers can always outperform scalar quantizers. However, scalar quantizers have been much more widely used in compression systems due to their computational simplicity. True vector quantizers are highly computationally intensive; prohibitively so in many practical applications.

A vector quantizer $\mathcal{Q}$ of dimension k and size N is a mapping from a point in $\mathcal{R}^k$, into a finite set $\mathcal{C}$ (Codebook) containing N code vectors. Thus

$$\mathcal{Q} : \mathcal{R}^k \rightarrow \mathcal{C} \quad (1.4)$$

where $\mathcal{C} = (y_1, y_2, \dots, y_N)$ and $y_i \in \mathcal{R}^k$ for $i = 1, 2, \dots, N$. The rate of the vector quantizer is $r = \frac{\log_2 N}{k}$ bits/sample and hence the codebook size is $N = 2^{rk}$.

The design of any vector quantizer entails selecting an encoding rule and a codebook. For an optimal system the average distortion between the input sequence and the selected code vector must be minimized. The average distortion is:

$$D = \sum_i d(x_i, Q(x_i))p_X(x_i). \quad (1.5)$$

Given the codebook size N , distortion measure $d(\cdot)$, and the p.d.f. of the source, we must select the encoding rule and codebook that minimizes D .

The encoding rule is completely defined by partitions $\mathcal{R}_i$ of the input space $\mathcal{R}^k$. Any source vector in $\mathcal{R}_i$ is mapped to the i^{th} code vector. For a given codebook, the optimal encoding rule is the nearest neighbor rule such that partitions are given by:

$$\mathcal{R}_i = \{x : d(x, y_i) < d(x, y_j); \forall j \neq i\} \quad (1.6)$$

if $\exists x$ s.t. $d(x, y_i) = d(x, y_j)$, x can be put in either $\mathcal{R}_i$ or $\mathcal{R}_j$

For a given partition, the optimal code vectors are the centroids of the partitions:

$$E[x|x \in \mathcal{R}_i]. \quad (1.7)$$

For a given set of training data, the optimal codebook is solved iteratively using the Lloyd algorithm [11].

According to Shannon's source coding theorems [1], vector quantizer will always outperform scalar quantizers. However, the encoding complexity of VQ is exponential

in rate and vector length. The trade-off for increased performance is a sharp increase in encoder complexity.

1.5 Motivation for Adaptive RIDPCM-TCQ

In general, as data compression systems have evolved to maximize performance the computational complexity of the encoder and decoder have increased greatly. In most applications, the substantial performance gains justify this increase in complexity. Thus, very little research has been done to investigate whether predictive algorithms can be improved to performance levels comparable to transform coders. Additionally, the specific problem of designing a high quality compression system, having the constraint of minimal decoder complexity, is posed. This particular problem is germane to certain military applications, and real-time video decoding.

The DPCM algorithm is rejected because although simple, its performance is inferior. The DCT and Wavelet coders are rejected due to decoder complexity in spite of their superior performance. The VQ algorithm is rejected due to the exponential cost of its encoder. The RIDPCM algorithm is selected because its performance can be increased to levels comparable with transform coders without greatly increasing the complexity of its decoder. In fact, the military has adopted the RIDPCM system as part of the National Image Transmission Format (NITF) for use in applications with strict decoder complexity constraints. The adaptive RIDPCM system with TCQ

maintains a decoder complexity far less than transform coders and exhibits performance far exceeding RIDPCM, and in fact, within the range of the most advanced and complex coding systems, as we will demonstrate in the remaining chapters.

CHAPTER 2

RIDPCM

2.1 Interpolation Theory

Interpolation is a non-causal prediction employing past and future signal values to estimate the present value. With respect to images, past and future values designate a spatial relationship to the present pixel. Therefore, provided subimage blocks of the image are available for processing, interpolation is quite feasible.

For a general 1-D stochastic signal the interpolation process is given in [13].

$$\hat{s}(t + \lambda) = \sum_{k=-N}^N a_k s(t + kT) \quad 0 < \lambda < T \quad (2.1)$$

where $s(t)$ is random process and a_k are prediction coefficients. The orthogonality principle yields,

$$E\{[s(t + \lambda) - \sum_{k=-N}^N a_k s(t + kT)]s(t + nT)\} = 0 \quad |n| \leq N \quad (2.2)$$

from which it follows that

$$\sum_{k=-N}^N a_k R(kT - nT) = R(\lambda - nT) \quad |n| \leq N \quad (2.3)$$

where $R(\cdot)$ is the autocorrelation function.

For example, take the case of interpolating a value in between two other values such that $N = 1$ and $s(t) = 0$ i.e. the inserted zero. For a first order filter,

$$a_1 = \frac{R(1)}{R(0) + R(2)} \quad (2.4)$$

and for a second order filter,

$$a_1 = \frac{R(1) - a_2(R(2) + R(4))}{R(0) + R(2)} \quad (2.5)$$

and

$$a_2 = \frac{R(3) - a_1(R(2) + R(4))}{R(0) + R(6)} \quad (2.6)$$

Assume the signal has a correlation that can be modelled by α^n where $\alpha = 0.9$ such that $R_0 = 1$, $R_1 = 0.9$, and $R_2 = 0.81$. For a first order filter, $a_1 = \frac{0.9}{1.81} \approx 0.5$. Intuitively, it is very reasonable that a good estimate of a point lying directly between two points is their average. Table 2.1 shows the correlations and associated optimal filter coefficients for “lena”. In all of the RIDPCM algorithms investigated, a first order filter with coefficients equal to 0.5 is used. The performance is near optimal, and multiplication by 0.5 is just a right shift operation for binary data. In the more general case, several points lying between the two known points could be estimated by varying λ between 0 and T. This would require computing the correlations and coefficients at each of these values, and then interpolating to the estimated signal in one step.

Data compression is realized by subtracting the interpolated values from the actual values and coding the difference signal. The variance of the difference signal should

Table 2.1: Optimal Filter Coefficients

	Original	Subsample	1 st Recursion	2 nd Recursion
R_1	<i>0.97</i>	<i>0.68</i>	<i>0.85</i>	<i>0.93</i>
R_2	<i>0.93</i>	<i>0.44</i>	<i>0.73</i>	<i>0.85</i>
R_3	<i>0.89</i>	<i>0.28</i>	<i>0.62</i>	<i>0.79</i>
1 st order h_1	<i>0.50</i>	<i>0.47</i>	<i>0.50</i>	<i>0.52</i>
2 nd order h_1	<i>0.56</i>	<i>0.47</i>	<i>0.50</i>	<i>0.52</i>
2 nd order h_2	<i>-.06</i>	<i>0.00</i>	<i>0.00</i>	<i>-.03</i>

be significantly smaller than that of the original signal and, analogous to DPCM, fewer bits are required for quantization at a given distortion.

2.2 RIDPCM Algorithm

Interpolated DPCM, the non-causal analogy to DPCM, was first proposed by Hunt for optical data compression in 1977 [14] [15]. Interpolative DPCM (IDPCM) differs from DPCM in that the algorithm uses a non-causal rather than a causal prediction filter to create the difference signal. As applied to digital imagery, the original image is subsampled, and the subsamples quantized and transmitted. The quantized subsample image is interpolated up to the original image size. The interpolated pixels are subtracted from their corresponding pixel values in the original image to create a difference sequence. The difference sequence is quantized at a lower bit rate and

transmitted. At the receiver, the quantized subsample image is interpolated up to the original image size, and the quantized difference values are added to their corresponding interpolated pixels in the interpolated image to reconstruct the quantized image.

The longer the subsample spacing, the greater the percentage of the image that is represented by difference values. Since the difference values are quantized at a significantly lower bitrate, increasing the subsample spacing will increase the amount of compression achieved by the algorithm. However, performing the interpolation in a single step for a larger subsample spacing will result in larger interpolation errors and consequently larger quantization errors.

The IDPCM algorithm was modified in [16] so that instead of performing the interpolation in a single step, the interpolation is executed recursively (RIDPCM) thereby providing better estimates of the pixels and consequently better compression. Figure 2.1 is a block diagram of the RIDPCM compression system. Figure 2.2 is a two dimensional representation of the compression process. In RIDPCM, the original image of size $(2^n \times 2^n)$ is subsampled by a factor of 2^l in each direction. The resulting subsample image, of size $(2^{n-l} \times 2^{n-l})$ is quantized and transmitted. The quantized subsample image is interpolated up by a factor of two. The interpolated pixels are subtracted from their corresponding pixels in the original image to form a sequence of error values. The error values are quantized at a reduced rate and transmitted. The quantized error values are added to their interpolated values to obtain a new

quantized image of size $(2^{n-l+1} \times 2^{n-l+1})$. This image is interpolated up by a factor of two, the error values are computed, quantized, and transmitted, and a new quantized image of size $(2^{n-l+2} \times 2^{n-l+2})$ is computed. This process continues recursively until the quantized image is equal in size to the original image. Since all operations are performed using quantized data, the decoder can compute the same quantized image.

The decoder (Figure 2.3) complexity of RIDPCM is minimal. If linear interpolation is used, the only computations required (on a per pixel basis) are one addition and one shift (division by 2) to interpolate each pixel value, and a single addition to add the quantized errors to the interpolated values.

The recursive interpolation allows RIDPCM to realize higher SNR's at lower bit rates. By computing the estimate of the original image at every recursion, the variance of the difference signal is decreased, thus facilitating better quantization. This is caused by the proximity of the pixels used to form the interpolated value. In the first recursion, the interpolated values are formed from subsample pixels which may have little or no correlation thus causing a fairly high variance of difference values. Whereas, in the last iteration, the interpolated values are formed from quantized pixels immediately adjacent to the interpolated pixel.

The overall encoder complexity of RIDPCM is much less than contemporary algorithms. The simple 3×3 2-D linear filter employed to perform the interpolation is separable into a 1-D filter shown in Figures 2.4 and 2.5 respectively. In Figure 2.4, only the coefficients having value $1/4$ are multiplied by non-zero pixels and in

Figure 2.1: RIDPCM Encoder

- (a) Original Image, subsample spacing of 4 i = Original Image Pixel
 (b) Quantized Subsample Image x = Subsample of Original Image
 (c) Quantized Image with inserted zeros * = Quantized Subsamples
 (d) Quantized Intermediate Image 0 = Inserted zero values
 (e) Quantized Image with inserted zeros 1 = 1st recursion quantized pixels
 (f) Quantized Image 2 = 2nd recursion quantized pixels

Figure 2.2: RIDPCM in 2D

Figure 2.3: RIDPCM Decoder

Figure 2.5, only the coefficients having value $1/2$ are multiplied by non-zero pixels. Therefore, the effective sum of the coefficients for each filter is one and thus energy preserving. An intelligent application of the 1-D filter reduces the number of computations significantly from directly convolving the image by the 2-D filter. The interpolation is executed by applying the 1-D filter horizontally to only the interpolated pixels in every other row. Thereafter, the filter is applied vertically to every interpolated pixel in every other row. Overall, $2/3$ of the pixels require a single filtering operation with the remaining $1/3$ requiring two operations. The result is equivalent to applying the 2-D filter directly. Each execution of the filter requires only 1 add and 1 shift. A total of $6r^2$ additions are required to form the difference vector and the intermediate quantized image. The value $r^2 = \frac{512^2}{K^2}$ where K is the subsample length. This process is repeated for the second and third iterations with the number of pixels quadrupling at each recursion. At the third or last recursion, the quantized image does not have to be computed. Table 2.2 illustrates the computational complexity of RIDPCM. The number of computations per pixel is $\frac{63}{K^2}$ shifts + $\frac{141}{K^2}$ adds or approximately 0.98 shifts and 2.2 adds per pixel.

The decoder complexity of RIDPCM is minimal. A LUT (no arithmetic operations) is needed to decode the binary codewords coming out of the channel, and only one add and one shift are required to interpolate each pixel value, and a single add to sum the quantized difference signal and the interpolated value.

$1/4$	$1/2$	$1/4$
$1/2$	1	$1/2$
$1/4$	$1/2$	$1/4$

Figure 2.4: 2-D Interpolation Filter

$1/2$	1	$1/2$
-------	-----	-------

Figure 2.5: 1-D Interpolation Filter

Table 2.2: RIDPCM Computations

	Interpolation	Difference Vector	Quantized Image
1 st	$3r^2(\text{shift}, \text{add})$	$3r^2(\text{add})$	$3r^2(\text{add})$
2 nd	$12r^2(\text{shift}, \text{add})$	$12r^2(\text{add})$	$12r^2(\text{add})$
3 rd	$48r^2(\text{shift}, \text{add})$	$48r^2(\text{add})$	no op's
Total	$63r^2\text{shifts} + 141r^2\text{adds}$		

The recursive nature of the interpolation algorithm provides some opportunities for intelligent bit allocation. At each recursion the variance of the difference values is decreasing. Thus, the difference values at each recursion can be quantized with fewer and fewer bits and achieve the same level of distortion. The freedom to reduce the bit rate at each recursion greatly enhances the compression capabilities of the system. The number of quantized difference values increases by a factor of 4 at each recursion with the final quantization accounting for 75% of all the pixels. Thus, quantizing the last set of difference values at a very low rate reduces the overall bit rate substantially without having a significantly deleterious effect on the overall image quality. The quantization rate for RIDPCM is [17]:

$$R_{avg} = (1/K^2)(BPP_{ss} + 0.75 \sum_i 4^i BPP_i) \quad (2.7)$$

where BPP_{ss} is the rate for quantizing the subsamples and BPP_i is the rate for quantizing the i^{th} recursion.

Figure 2.6: RIDPCM Performance

Figure 2.7: RIDPCM at 1 bit: (a)Lena, (b)Urban26, (c)Lena Difference Image, (d)Urban26 Difference Image

2.3 Testing/Results

Figure 2.6 plots the performance of RIDPCM on the four test images. The bit rates were “hand tuned” towards maximizing SNR. The SNR and visual quality of the reconstructed images are far inferior to the performance of contemporary transform coders. Figure 2.7 shows the reconstructed image and difference images for “lena” encoded at 1 bit, and “urban26” encoded at one bit.

Although RIDPCM provides significant improvement over IDPCM [16], there is still room for additional gains. First, although the bit rates can be set to take advantage of the decreasing variance of the difference signal at each recursion, the bit rates can not be adjusted to take advantage of spatially varying properties of the image. Uniform areas of the image contain very little information and can be quantized with far fewer bits without causing substantial distortion whereas highly detailed regions require more bits to represent them adequately. Under the current scheme, either a large number of bits are wasted on areas with low information content or severe distortion is incurred in the detailed regions of the image. The answer to this problem is to make the bit allocation adaptive as a function of scene content.

A second deficiency in current implementations of RIDPCM is the quantization employed to code the difference signal. Typically, a uniform quantizer is used for the subsample values and an optimum Lloyd-Max scalar quantizer is employed for

the difference values. These scalar quantizers encode each individual sample separately. Since there are a large number of difference values at each recursion for a typical image, a quantizer that encodes long vectors of data would provide superior performance.

A third problem with the classic RIDPCM algorithm is rate allocation. To achieve a desired bit rate with good SNR and visual quality for a given image, the rates must be iteratively hand tuned. This is an extremely inefficient method. The problem is greatly exacerbated when adaptivity and entropy constrained quantization are added to the algorithm. An automated method for determining the optimum rate allocation for any image is required.

CHAPTER 3

ADAPTIVE CLASSIFICATION

3.1 Problem Statement

In the context of image processing, adaptivity relates to changing the processing algorithm in accordance with changing local statistics or signal properties. Adaptivity can be employed in contrast enhancement, prediction filters, or rate allocation. In a contrast enhancement algorithm, a 512 by 512 image might be partitioned into 32 by 32 blocks, the minimum and maximum grey levels in each block measured, and the pixels in each block stretched in accordance with the local statistics. In speech processing, the predictor might change at each sample depending on estimates of the local statistics. In regards to rate allocation, an adaptive algorithm might be able to separate the image into different classes, each class requiring a different bit rate to achieve a desired quantization distortion. Such an algorithm would allow more bits to be allocated to high variance classes and fewer bits to be provided to low variance classes, thereby improving the overall SNR of the reconstructed image for a given rate.

3.2 Previous Implementations of Adaptive RIDPCM

An adaptive RIDPCM algorithm was presented in [16] . The basic RIDPCM algorithm is not changed by the addition of the adaptive algorithm. The adaptive algorithm assigns each pixel in the image to one of J classes. Each of the classes represents a varying level of image structure, detail, or information. Within each recursion of the interpolative DPCM algorithm, the quantization rate can be adjusted according to the characteristics of the particular class. Accordingly, high information and detailed classes are assigned more bits than low detail, low information classes. In this particular algorithm, the rates are still selected by hand for each image and each rate. This is a very inefficient and suboptimal method.

This particular adaptive algorithm works as follows. The image is divided into subimages, the corners of which are defined by the subsamples. If the sample length is eight, then each subimage is eight by eight. A measure of subimage activity is calculated for each subimage at each recursion. The measure is the sum of the absolute magnitudes of the difference between the original image and the interpolated values within each subimage, the theory being that early recursions will have larger difference errors than late recursions and high activity subimages will have larger difference errors than low activity subimages. The measure is compared to hand-tuned thresholds and assigned to either a low, average, or high detail class. The tables corresponding to the three classes for each recursion are hand picked. The

classification of each subimage may change at each recursion. A Lloyd-Max scalar quantizer is employed to quantize the difference samples. For a 128 x 128 version of “lena”, at an encoding rate of 0.5 bits, the reconstructed image has Peak-SNR (PSNR) of 26.26 dB.

Another adaptive RIDPCM system was presented in [17] . The adaptive algorithm works as follows. The subimages are defined in the same manner as in [16]. Features extracted from each subimage are given as input to a classifier which assigns each subimage into one of J classes. The classifier is developed via a supervised training algorithm which requires significant involvement and skill of an analyst. Overhead bits must be transmitted to identify the class of each subimage. The decoder must know what class each subsequent pixel is labeled to be able to extract the proper number of bits from the transmitted bitstream to form the quantized difference value.

To develop an effective classifier, a set of features which reduces the dimensionality of the data while preserving or enhancing our ability to discriminate between varying classes of imagery must be identified. Three features were considered in a series of tests: variance, entropy, and fractal dimension.

The second step in developing a classifier is selecting a training algorithm. A neural network architecture in conjunction with the back propagation algorithm was used to train the classifier. An artificial neural network is composed of multiple layers of processing elements which are interconnected by weighted links [18], [19],

[20]. At each node, a weighted sum of inputs is passed through a functional non-linearity to produce the output of the node. A semilinear feedforward multilayer network, theoretically capable of creating arbitrarily shaped classification regions, is employed. The back-propagation algorithm, employed to train the network, uses the gradient descent method to minimize the error between the known output and the actual output of the net. The output of the network is generated by presenting training data to the input of the net and their known classes to the output of the net. The resulting error is propagated back through the net to update the network weights. The training data is presented until the error converges. The result of training is a set of centroids representing the n classes of subimages. The supervision occurs in assigning the classes of the training data.

As applied to ARIDPCM, the analyst must view each subimage in the training data set and assign it to a class. If a different number of classes is desired, the task must be done again. This can be a very tedious and difficult task.

The trained classifier consists of the J centroids generated by the training algorithm for a particular class of imagery, e.g. urban, portrait. In the compression system, an image is divided into eight by eight subimages, the features are extracted from each subimage, and each subimage is assigned the class label of the nearest class, where nearest is judged by euclidean distance. This produces a classification map for every compressed image.

The results of adding this particular adaptive scheme to RIDPCM are shown in Figure 3.1. These results are strongly biased by the fact that the classifier was trained on subimages from “lena” and then tested on “lena”. Furthermore, the network training was very intensive. The neural network parameters, such as momentum and learning rate, were tuned specifically for “lena”.

3.3 Current Implementation of Adaptive RIDPCM

In this dissertation, the adaptive algorithm has a structure similar to the previous algorithms but employs different features and an unsupervised training algorithm. To implement the adaptive algorithm, the image is divided into subimages, the corners of which are defined by the subsamples. Features extracted from each subimage are provided as input to a classifier which assigns each subimage into one of J classes. The classifier is developed using an unsupervised k-means training algorithm. The adaptive algorithm implemented in conjunction with RIDPCM functions along these lines. If there are I recursions and J classes there will be $I \cdot J + 1$ sequences created by the algorithm. The variance and length of each sequence are input to the rate allocation system and determine the number of bits an individual sequence will be assigned.

Figure 3.1: Adaptive RIDPCM Performance

3.4 Adaptivity: Feature Extraction

The feature extraction process is crucial to the performance and complexity of the classifier and hence the adaptive algorithm. The goal is to find a set of features which reduces the dimensionality of the data while preserving or enhancing our ability to discriminate between varying classes of imagery. Four features are considered in a series of tests: the previously mentioned variance, entropy, and fractal dimension features, and an edge density feature.

The variance feature is just the variance of a subimage. For a $K \times K$ subimage, the variance is calculated

$$\sigma^2 = (1/K)^2 \sum_i \sum_j (I(i, j) - \bar{I})^2 \quad (3.1)$$

where $\bar{I}$ is the subimage mean. The variance is a good indicator of the activity level in a subimage, separating the high detail and contrast regions from the low detail and contrast regions [21].

The entropy feature measures the amount of information contained in a subimage. The entropy represents the minimum theoretical bit rate needed to code the data in the subimage without error using first order coding techniques. To compute the entropy, the probability density function of the subimage must be calculated. The estimate of the entropy is calculated

$$H(\rho) = - \sum_i p(i) \log_2(p(i)) \quad (3.2)$$

The fractal dimension feature is a measure of roughness as perceived by a human viewer. Fractal dimension is based upon the continuity of an individual pixel value and surrounding pixel values. By incorporating the spatial relationship between the pixels, fractal dimension measures scene complexity and can discriminate between detail and edge scenes. The algorithm for computing the fractal dimension is given in [22] and [23]. A uniform scene has a fractal dimension of two whereas a completely random scene has a fractal dimension of three.

The edge density feature measures the number of edge pixels in a subimage. A 2-by-2 Roberts gradient operator [2] is applied in two orthogonal directions to each subimage. The magnitude of the sum of the two orthogonal components for each pixel is compared to a threshold. All pixels having magnitudes exceeding the threshold are declared “edge” pixels. The maximum filtered value occurs at an edge from zero to 255 and equals $\sqrt{2} \times 255$. The threshold is set at 10% of the maximum. Adjusting the threshold effects the sensitivity of the measure. The edge density is normalized between zero and one.

The performance of any classifier is highly dependent on the feature set. To determine the best set of features, the four individual features and combinations of the features were used to train and test classifiers. The performance of each feature set was judged on three criteria: a comparison of the original image and its classification map, the number of subimages assigned to each class, and the quality of the compression in an adaptive RIDPCM system. The classifier labels each subimage

as belonging to one of n classes. Each class is then assigned a grey level value corresponding to its class label. A visual comparison of the original and classification images indicates how well the classifier and feature set are identifying image structure and discriminating between classes. Classifying the subimages creates a distribution of the classes for each image. This distribution should reflect the structure and frequency content of the image. A good feature set won't classify all the subimages into a single class but will reflect the composition of the image. The quality of the compression realized for the Adaptive RIDPCM algorithm was compared for each feature set. A qualitative evaluation or judgement based on these three criteria revealed that the best overall compression is achieved by separating the subimages according to the number and strength of edges.

The quantization of edges is very important to minimizing the distortion of the compressed image and to the perceived image quality. Additionally, uniform or background regions typically have very few strong edges. The variance, entropy, and fractal dimension features all incorporate the edge properties to a varying degree, and those subimages with a lot of strong edges will have higher variance, entropy and fractal dimension. These three features were found to perform adequately but do not separate the classes as sharply as desired.

The edge density feature sharply discriminates between the subimages with varying degrees of edge activity and detail. For portrait type images, those subimages with very sharp edges occupy a fairly sparse class, those with significant edge activity

occupy a more populated class, and so forth until the large majority of the subimages fall in the lowest class. For urban images, the high frequency content of the image is reflected in the higher populations of the higher edge density classes. The distributions of the classes reflects the structure and detail found in the two types of imagery.

By only employing a single, relatively computationally simple feature such as edge density the number of computations for training the classifier and more importantly, the number of computations for classifying the subimages in each image, is small.

3.5 Adaptivity: Classifier Training

In general, there are two types of training algorithms; supervised and unsupervised. Supervised and unsupervised relate to the involvement of a human analyst. In supervised algorithms, the analyst is an integral part of the overall system. The analyst's abilities and time to a large part determine the performance of the algorithm. In unsupervised algorithms, an analyst is not involved. Clearly, if the performance of a supervised algorithm and an unsupervised algorithm are comparable, the unsupervised one will be preferable. The neural network algorithm is an example of a supervised procedure. A k-means clustering algorithm is an example of an unsupervised method of training.

The K-means algorithm is a very simple yet robust clustering algorithm [24], [25]. Given a desired number of clusters and their dimension, the algorithm initializes

random cluster centroids. The set of extracted features from each subimage of a set of training images is presented to the clustering algorithm. Each subimage is assigned to the closest cluster centroid, where distance is measured in euclidean distance. After the first presentation of the training data, the cluster centroids are recomputed by averaging feature vectors in each cluster. The training data is presented, classified, and the cluster centroids recomputed until the algorithm converges.

The difference between the neural network and k-means training algorithms is the amount of analyst input required. Whereas the k-means algorithm requires no supervision, the performance of the neural net is highly dependent on the analysts identification of the subimage classes. For every class of training data and every different number of classes, the analyst must accurately identify a large number of training images. This involvement could possibly be justified, if the classification maps generated by the neural net algorithm were substantially better than those produced by the k-means algorithm. An evaluation of the classification maps generated by the two algorithms has not demonstrated any significant difference between the two methods.

3.6 Adaptivity: Classification

The result of training either classifier is a set of centroids representing the J classes of subimages. The actual classifier used in compression simply assigns to each subimage a class label representing the closest centroid. This produces a classification

map for every compressed image. Analysis of the original image and its classification map provides insight into the performance of the classifier and selected features.

The costs of adding adaptivity include training a classifier, extracting features from and classifying each subimage, and transmitting the class label for each subimage. The number of computations for extracting features depends on the number and type of features. Classification entails computing a q -dimensional euclidean distance with J different classes and selecting the smallest value for each subimage, where q is the number of features and J is the number of classes. To use all the bits efficiently, the number of classes is usually chosen as a power of two such that the number of bits required to transmit the class label for a given subimage is $\log_2 J$. Therefore, the additional bits required for an adaptive system is $\frac{\log_2 J}{K^2}$ /pixel. For $J=4$ classes and $K= 8 \times 8$ subimages, this is approximately 0.03 bits/pixel. Huffman coding may be used to reduce this rate. Decoder complexity is not effected by the adaptive rate allocation scheme.

By classifying the subimages into four classes, thirteen different sequences are formed, 1 subsample sequence and 4 sequences in each of the three recursions. The variance and number in each sequence are provided to the rate allocation scheme which returns an optimal rate table for a given compression rate. Each sequence is quantized at the specified rate. The adaptive rate allocation greatly improves the overall performance of the system. The encoder complexity is increased substantially but the decoder is not affected.

3.7 Classifier Implementation

Figure 3.2 depicts the implementation of the classifier training and classification algorithms. The training algorithm makes use of a set of training images, specifically six images from the USC data base (portraits) or six images from an urban data base (urban), a feature extraction scheme and a clustering algorithm. The sets of training and test images are mutually exclusive. The training images are partitioned into 8 x 8 subimages which are provided to the feature extractor. A subsample length of 8 was found to be appropriate. At this length the pixel-to-pixel correlation is approximately 0.6 and the subsamples account for only 1/64 of the image. Increasing the subsample length provides no noticeable improvement. The edge density, a scalar, is computed for each subimage and input to a k-means clustering algorithm. The desired number of classes (e.g., four) are also provided as input. The clustering algorithm randomizes the initial cluster centroids, assigns each subimage to the nearest centroid, and recomputes the cluster centroids as the average of the elements assigned to each cluster. This repeats until convergence is achieved. The output of the training algorithm is four one-dimensional centroids representing the classes. The smallest edge density centroid is class A and represents those subimages with minimal edge content. The largest edge density centroid is class D and represent those subimages with the highest edge density. Table 3.1 shows the results of training the classifier on the USC images and the urban images respectively.

Train

Test

Figure 3.2: Subimage Classifier

Table 3.1: Trained Classifier: Portrait and Urban Images

	Portrait		Urban	
	Centroid	%	Centroid	%
Class A	<i>0.01</i>	<i>56.7%</i>	<i>0.046</i>	<i>14.5%</i>
Class B	<i>0.218</i>	<i>15.2%</i>	<i>0.266</i>	<i>17.4%</i>
Class C	<i>0.463</i>	<i>13.8%</i>	<i>0.467</i>	<i>31.5%</i>
Class D	<i>0.757</i>	<i>10.4%</i>	<i>0.67</i>	<i>32.7%</i>

For the portrait images, Class A is very populous, accounting for approximately 56.7% of all subimages, and has a very low edge density of 1%. Class D is relatively small at 10.4% but has a very high edge density of 76%. For the urban images, class A is much smaller at 14.5% but has a larger centroid of 4.6%. Class D accounts for 32.7% of the subimages and has an edge density of 67%.

The class definitions i.e., centroids, are nearly the same for the two classes of imagery. The relative frequency of each class is substantially different for the two types of imagery. The portrait images contain large uniform areas and a small percentage of sharp image structure. The distribution is therefore skewed towards the edge density classes. The urban imagery contains very few low frequency subimages and a great number of high frequency subimages, with a corresponding skew to higher edge density classes.

Table 3.2: Classification: Portrait and Urban Images

	Lena	Pagladay	Urban26	Urban28
Class A	<i>73%</i>	<i>35.6%</i>	<i>27.5%</i>	<i>9.4%</i>
Class B	<i>14.7%</i>	<i>19.8%</i>	<i>23.5%</i>	<i>14.1%</i>
Class C	<i>11.1%</i>	<i>17.5%</i>	<i>26.6%</i>	<i>33.6%</i>
Class D	<i>4.2%</i>	<i>27.1%</i>	<i>22.5%</i>	<i>42.8%</i>

The testing or classification algorithm comprises a single test image, the feature extractor, and the trained classifier. The image is divided into 4096 subimages, the edge density feature of each subimage is measured, and input to the classifier. Each subimage is assigned to the closest class and labeled. The subimage labels are sent to the encoder and decoder. Table 3.2 represents the classification results for the portrait and urban test images respectively.

Figure 3.3 is a montage of the classification maps or images for the four test images. Each image is classified using the appropriately trained classifier. The darkest subimages are class A and the lightest class D. Comparing the classification images to the original images in Figure 1.1, the classifier appears to label the subimages in a manner consistent with the visual structure of the image. For example, the extensive background in “lena” is all class A whereas the sharp edges on the hat and feather are labeled class D.

Figure 3.3: Classification Images: (a)Lena, (b)Paglady, (c)Urban26, (d)Urban28

CHAPTER 4

QUANTIZATION

4.1 Quantization Theory

Typical signals such as speech, waveforms, and imagery are continuous both in time and in amplitude. However, modern communication systems such as the one depicted in Figure 1.2 are digital, discrete in time and amplitude, having A/D converters at the input and D/A converters at the output. The representation of a signal in discrete rather than continuous time falls under the area of sampling theory. The process of mapping the continuous valued amplitude signal into a finite valued signal is called amplitude quantization.

Representing a continuous valued signal by elements selected from a finite codebook introduces distortion into the signal. Distortion is typically measured by $MSE = \frac{1}{N} \sum_{i=1}^N (x_i - \hat{x}_i)^2$. When quantizing a number of sequences or coefficients, the problem is to minimize the distortion for a given codebook size.

$$\min \sum_{i=1}^N (x_i - \hat{x}_i)^2 \tag{4.1}$$

$$\ni \log_2 M = R$$

where N is the number of input values, x_i is the input signal value, $\hat{x}_i$ is the quantized signal value where $\hat{x}_i = Q(x_i)$, M is the number of codewords in the codebook, and R is the rate.

The quantization function $Q()$ has two interrelated parts. The first is the function for mapping between the input and output space. The function can be a scalar or vector quantizer, and can be a process with or without memory. The second part is selecting the M codewords in the codebook.

The mapping function and codebook are effected by a number of factors. The type of data, system constraints, and the definition of “optimal” all effect the structure of the quantization. Uniform, Gaussian, and Laplacian sources will have very different optimal codebooks [26]. Practical computational complexity and memory constraints may constrain the type of algorithms that are acceptable. Typically, “optimal” refers to minimizing the MSE of the quantized signal. However, for the application of compressing and quantizing image data for optimal visual quality, MSE is not always viewed as a very good distortion measure. This is attributable to the types of errors induced by quantization and their effects both on visual quality and on the MSE measure. Granular errors are small in magnitude and appear as fine grain noise throughout the image. Granular errors are caused by the inability of the smallest quantizer level to represent smooth regions of the image well. Granular noise is most visible in constant or low detail areas of the imagery. Contouring occurs when groups of neighboring pixels are quantized to the same value. For low rate quantizers

(few levels), contouring can be a significant problem. Overload errors are large in magnitude and are the result of the inability of the largest quantizer level to represent edges. Overload errors are typically fewer in number than granular, or contouring errors but contribute disproportionately to the distortion measure. Visually, the granular and contouring errors are more offensive than overload errors in high detail imagery. Therefore, a quantizer design that minimizes the MSE might be sub-optimal in the context of visual quality. A lot of research has been done to identify a better distortion measure that is computationally simple for visual processing [27], [28], [29], but to date no such measure that is completely satisfactory has been found.

4.2 Vector vs Scalar Quantizers

Quantization encompasses scalar, vector, and trellis quantization. Scalar quantizers map each input point to an output point independent of all others. Scalar quantization is very simple but fails to achieve the performance promised by the rate-distortion function. To achieve this theoretical bound on performance, data must be encoded in long sequences instead of sample-by-sample.

The theoretical performance bound on quantization is a function of the granular, entropy, and memory gains associated with a particular quantizer [30], [31]. The granular gain is determined by the shape of the Voronoi regions. The Voronoi regions are shaped by the nearest neighbor mapping associated with the MSE distortion measure, and thus by the positioning of the output points in N -space of the quantizer.

The optimal shape is found by minimizing the second moment of the region and can be shown to converge asymptotically to an N -dimensional sphere [30]. The granular gain is independent of the particular source density. The entropy gain relates to matching the shape of the quantizer codebook to the source density. The entropy gain can be achieved in three ways: vector length and position, codebook weighting, and entropy coding. With respect to a uniform scalar quantizer, the available entropy gain is 2.81 dB and 5.63 dB for the Gaussian and Laplacian sources respectively. Long vector quantizers can exploit the correlation of a source and thus achieve the memory gain directly. Otherwise, the source can be decorrelated and then quantized.

Scalar quantizers are restricted to forming N -dimensional rectangular Voronoi regions. This restriction costs scalar quantizers 1.53 dB in gain from the theoretical optimum [30]. Scalar quantizers can achieve all of the entropy gain by using optimal (Lloyd-Max) codebooks and entropy coding. Vector quantizers form optimal Voronoi regions as $N \rightarrow \infty$ and thus can outperform scalar quantizer by as much as 1.53 dB. Vector quantizers can realize the entropy gain directly as $N \rightarrow \infty$ without having to design optimal codebooks or employ entropy coding. For smaller N , the entropy gain can be realized by shaping the codebooks and entropy coding the vectors. The performance of vector quantizers is far superior to scalar quantizers as N gets very large. However, the encoding complexity of VQ is exponential in N , 2^{RN} .

For the Uniform source, the optimal codebook distribution is uniform and thus there is no entropy gain available. Therefore, any performance difference between

scalar and vector quantization is attributable to the shape of the Voronoi regions. Figure 4.1 is an example of a 2-D uniform case. The scalar quantizer forms rectangular Voronoi regions and has a $SNR = 6.02 \times rate$ dB. The length two VQ creates hexagonal regions and has a $SNR = 6.02 \times rate + 0.17$ dB. As the dimension $N \rightarrow \infty$, the Voronoi regions of the scalar quantizer become N-dimensional hypercubes and the SNR remains the same, while the Voronoi regions of VQ converge to the optimal spherical regions and an $SNR = 6.02 \times rate + 1.53$ dB, thus achieving the rate distortion bound [32]. For any source, the maximum gain over scalar quantization attributable to Voronoi region shaping is 1.53 dB.

In theory, infinite state trellis quantizers can also provide spherical Voronoi regions and the 1.53 Db gain. Typically, finite state trellis quantizers are employed to achieve reasonable encoding complexity at the cost of suboptimal Voronoi regions. The number of states in the trellis and the efficiency of the particular encoding algorithm determine how much of the 1.53 dB can be achieved.

A scalar quantizer and an alternate representation are shown in Figure 4.2. A scalar quantizer is a function that deterministically maps each element in a subset of the real line to a particular value in a finite subset of the real line. If the real line is segmented into M disjoint intervals

$$\mathcal{X}_i = (x_i, x_{i+1}), \quad i = 1, 2, \dots, M$$

with

$$-\infty = x_1 < x_2 < \dots < x_{M+1} = +\infty$$

Scalar Quantizer, $N = 2$

Vector Quantizer, $N = 2$

Figure 4.1: Voronoi Regions

and y_i is the output of a point in $\mathcal{X}_i$, a scalar quantizer maps every element in the input sequence x lying in the interval $\mathcal{X}_i$ into the output point y_i .

The simplest scalar quantizer is the uniform quantizer in which the intervals $\mathcal{X}_i$ are all the same length and the output points y_i are equal to the center of the interval $\mathcal{X}_i$. Figure 4.2 is an example of a 2-bit uniform quantizer where $\text{length}(\mathcal{X}_i) = 1$.

A better performing, although more complex, scalar quantizer is the Lloyd-Max quantizer [2] which provides optimal performance for a scalar quantizer. The source is modeled as a sequence of realizations of a random variable X with p.d.f. $f(x)$ and using the MSE as a distortion measure,

$$D = E[(X - Y)^2] = \sum_{i=1}^M \int_{x_i}^{x_{i+1}} (x_i - y_i)^2 f_X(x) dx. \quad (4.2)$$

This optimization problem is solved by setting the partial derivatives of D wrt $x_i, i = 1, 2, \dots, M$ and $y_i, i = 1, 2, \dots, M$ equal to zero. This yields:

$$x_i = \frac{y_i + y_{i+1}}{2}, \quad i = 1, 2, \dots, M \quad (4.3)$$

and

$$y_i = \frac{\int_{x_i}^{x_{i+1}} x f_X(x) dx}{\int_{x_i}^{x_{i+1}} f_X(x) dx}, \quad i = 1, 2, \dots, M \quad (4.4)$$

as necessary conditions for optimality. Solutions for x_i and y_i can be computed iteratively [2]. Note, if $f_X(x)$ is uniform, the solution simplifies to the uniform quantizer.

Vector quantizers, as described in Chapter 1.2, can achieve the rate distortion bound provided the dimension or length N of the encoded vector is allowed to grow without bound. The crux of the problem with traditional vector quantizers is that

Alternate Representation

Figure 4.2: Scalar Quantizer

the complexity of a vector quantizer is an exponential function of sequence length. Unfortunately for problems of practical interest very long vectors are required to achieve high SNR's but the complexity of such encoders is not feasible.

4.3 TCQ

A moderate complexity and particularly effective trellis quantizer is TCQ [26]. TCQ can provide near optimal Voronoi regions to capture the granular gain, and using optimal codebooks can achieve a large portion of the entropy gain. Entropy-constrained TCQ [33] achieves the remainder of the entropy gain. Entropy-constrained TCQ using a 4-state trellis has performance within 0.55 dB of the rate distortion function for a Gaussian source [33]. If the number of states is allowed to get large, the performance will converge to the rate distortion function. The encoding for an L -state trellis requires only 4 multiplications, $4+2L$ additions, L comparisons, and 4 rate $(R-1)$ scalar quantizations per data sample. The number of computations is not a function of trellis length [26]. The decoding for TCQ can be done using only look-up-tables requiring no arithmetic operations. Applications of TCQ to image coding are described in [34], [35], [36]. Because of their excellent performance, moderate encoder complexity, and very low decoder complexity, we have chosen to use TCQ and ECTCQ in our advanced RIDPCM systems.

Trellis coded quantization (TCQ) is a highly efficient encoding algorithm that provides results very close to the rate distortion function. A trellis is simply a transition

diagram (with a time index) for a finite state machine. Figures 4.3 and 4.4 show a 4-state machine and its associated state transition diagram. In the state machine, t_1 and t_0 are binary storage elements and “+” is a modulo adder. r is the binary input to the machine, s_1 and s_0 are the outputs, and t_1t_0 is the state of the machine. In the transition diagram each circle represents a state, each arrow represents a state transition with the labels on each arrow indicating the input required to initiate that transition and the associated output.

Figure 4.5 depicts a 4-state trellis. Each heavy dot is a state or node, and each branch signifies a state transition. Selecting a sequence of states or state transitions is equivalent to selecting a path through the trellis. In general, up to 2^R branches may enter and leave each node of the trellis. Thus any path through the trellis starting from a specified initial state can be given by a sequence of R -bit codewords. In TCQ, only two branches enter and leave each state in the trellis such that the path through the trellis can be specified by only one bit per stage.

In TCQ, the trellis states are populated with codewords selected from a structured codebook. The codebook has 2^{R+1} values, which for a 4-state trellis are split into four subsets, D_0, D_1, D_2, D_3 . The subsets are populated by arranging the 2^{R+1} codewords in ascending order, assigning the 1st codeword to D_0 , the 2nd codeword to D_1 , and so on until each subset has 2^{R-1} codewords.

The trellis is generated by a rate 1/2 convolution coder. Figure 4.6 shows a feedforward implementation of a 4-state, rate 1/2 convolutional coder. The states in

Figure 4.3: 4-State Machine

Figure 4.4: State Transition Diagram

Figure 4.5: 4-State Trellis

the convolutional coder specify the state in the trellis, and the outputs of the coder specify a subset D_i . To encode a sequence at rate R , one bit is used to trace a path through the trellis from subset to subset. The remaining $R-1$ bits are used to select the best codeword from the subset D_i .

The actual encoding is done using the Viterbi algorithm. The Viterbi algorithm is an application of forward dynamic programming that facilitates encoding the trellis by making L local decisions at each stage, where L is the number of states in the trellis. Given an input sequence x , the Viterbi algorithm computes the distortion associated with each path leaving the initial state. The distortion at the i^{th} stage for a given path will be $(x_i - y_i)^2$ where y_i is the closest codeword from the subset D_i specified by the trellis. Whenever two paths pass through the same node in the trellis, only the smallest cumulative distortion path (survivor path) is retained. The number of survivor paths equals the number of states in the trellis. At the end of the input sequence, a global decision is made by retaining the survivor path with the lowest cumulative distortion. The quantized sequence is decoded by providing the sequence of bits specifying the path through the trellis to the convolutional coder. The output of the convolutional coder selects the proper subset D_i and the sequence of rate $R-1$ codewords selects the correct codeword from the subset D_i .

For a given number of states, the Viterbi algorithm provides the optimal solution for the trellis. By increasing the number of states in the trellis, the number of survivor paths is increased and the MSE of the best path will be lower. For the doubled

Figure 4.6: Rate 1/2 Convolution Coder

alphabet coders, the encoding for an L -state trellis requires only 4 multiplications, $4+2L$ additions, L comparisons, and 4 rate $(R-1)$ scalar quantizations per data sample. The number of computations is not a function of trellis length [26]. In VQ, the number of computations is an exponential function of block length, see Table 4.2.

Doubled and optimized codebooks for different sources are found via a clustering algorithm. The subsets are initially populated with the output points from a rate $R+1$ Lloyd-Max quantizer. A data sequence of length 100,000 is quantized. Each codeword is recomputed as the average of the input sequence values assigned to that codeword. The sequence is quantized and the new codewords computed until the algorithm converges. This algorithm is repeated for all desired integer rates and source distributions.

The subsample and error sequences are modeled by generalized gaussian sources. The generalized gaussian distribution is:

$$p(x) = Ae^{-B(|x|/s)^\alpha} \quad (4.5)$$

where A is a normalization parameter, B is a scale parameter, s is the standard deviation, and α is the exponent of the distribution. The parameters A and B are given by,

$$A = \frac{\alpha B^{1/\alpha}}{2s\Gamma(1/\alpha)} \quad (4.6)$$

and,

$$B = \frac{\Gamma(3/\alpha)^{(\alpha/2)}}{\Gamma(1/\alpha)} \quad (4.7)$$

where the gamma function is defined in [13] as:

$$\Gamma(b+1) = \int_0^{\infty} y^b e^{-y} dy \quad b > -1 \quad (4.8)$$

If $\alpha = 2$ the distribution is Gaussian, and if $\alpha = 1$ the distribution is Laplacian. Models of $\alpha = 1.5$ and $\alpha = 0.75$ are selected for the subsample and error sequences respectively. Histograms of the image subsamples and interpolation error sequences indicate that the data are reasonably modeled as generalized Gaussian with exponential parameters of 1.5 and 0.75, respectively. Thus, their p.d.f.'s are somewhat more peaked and heavy tailed than those Gaussian and Laplacian densities, respectively.

Table 4.1 displays the SNR performance in dB of TCQ for a Gaussian source using doubled and optimized codebooks [26]. At 1 bit, 4-state TCQ outperforms the Lloyd-Max quantizer by 0.6 dB and is within 1 dB of the rate distortion function (RDF). The improvement relative to scalar quantization increases with rate and number of trellis states. At high rates for the Gaussian source, TCQ is within 2 dB of the rate distortion function. Additional trellis states improves the performance of TCQ slightly. Figure 4.7 shows the performance of TCQ with doubled and optimized codebooks for the generalized gaussian sources of $\alpha = 0.75$ and $\alpha = 1.5$. Table 4.2 displays the number of computations for TCQ as well as VQ and other coding algorithms. In TCQ, the number of computations is only a function of the number of states L and the rate R . In VQ, the complexity is an exponential function of rate R and most importantly block length N .

Table 4.1: TCQ Performance for Gaussian Source Using Doubled and Optimized Codebooks

	Trellis Size (states)								
Rate	4	8	16	32	64	128	256	Lloyd-Max	RDF
1	5.00	5.19	5.27	5.34	5.43	5.52	5.56	4.40	6.02
2	10.56	10.70	10.78	10.85	10.94	10.99	11.04	9.30	12.04
3	16.19	16.33	16.40	16.47	16.56	16.61	16.64	14.62	18.06

Table 4.2: Computational Requirements of TCQ and VQ for Memoryless Sources

Coder Type	Multiplies	Adds	Compares
TCQ(doubled alphabet)	4	$2L + 4$	$L + 4(R - 1)$
VQ	2^{RN}	2^{RN}	$(2^{RN} - 1)/N$

An infinite dimensional Vector Quantizer can achieve the rate distortion bound in theory. However, the complexity of this theoretical encoder is also infinite, and in fact the complexity of VQ having performance close to the rate distortion bound is too high to be practical. The drop in performance of TCQ as compared to the infinite dimensional VQ is relatively small as shown in Table 4.1. For VQ having a small enough dimension to be computationally practical, TCQ outperforms VQ for a given block length[32].

TCQ achieves most of the performance advantages of vector quantization i.e. support and Voronoi regions, without the exhaustive search requirements of VQ. The incorporation of doubled and structured codebooks and the Viterbi algorithm into the trellis structure create a low complexity encoder (TCQ) that provides excellent MSE performance. Figure 4.8 shows the performance of an optimum scalar, 8-state TCQ and 8-state entropy-constrained TCQ (ECTCQ) with doubled and optimized codebooks for the gaussian source, and the rate distortion function. The performance of TCQ is much better than Lloyd-Max quantization but is still inferior to the rate distortion bound. Using entropy constrained TCQ instead of fixed rate TCQ captures the remaining performance available.

4.4 ECTCQ

Instead of directly binary coding the codewords, a scheme that entropy codes the codewords achieves the same distortion with a lower average rate for a given trellis

Figure 4.7: TCQ Performance for Generalized Gaussian Sources

Figure 4.8: Rate Distortion Function

length N and codebook size M . The entropy coded system reduces the rate from $R = \frac{(\log_2 M)}{N}$ to the codebook entropy. The problem is to find the codebook which minimizes the average distortion between the source and the coded output subject to an entropy constraint.

Entropy-constrained TCQ was first developed in [37] to improve the high rate performance of TCQ. Very good performance at high rates was achieved but coding was not possible at rates below one bit. In TCQ, one bit is used to trace the path through the trellis and $R-1$ bits are used to select a codeword from a subset. Therefore, even if the entropy of the codebook was zero, one bit would still be required to trace through the trellis. ECTCQ was extended to low rates in [33]. Instead of using one bit to specify a path and $R-1$ bits to select a codeword, all R bits are used to select a codeword from a “superset” at each stage. From the structure of the trellis, it is evident that given a current state, the next codeword must be selected from either $D_0 \cup D_2$ or $D_1 \cup D_3$ depending on the current state. The two supersets are accordingly $A_0 = D_0 \cup D_2$ and $A_1 = D_1 \cup D_3$ which are mutually exclusive sets. Therefore, using R bits to select a codeword from a superset uniquely determines the next state in the trellis.

A variable length code is generated for each superset with an encoding rate of:

$$H(X|A) = - \sum_{i=0}^1 P(A_i) \sum_{x \in A_i} P(x|A_i) \log_2 P(x|A_i) \quad (4.9)$$

The entropy $H(X|A)$ can assume any non-negative value.

4.4.1 Uniform Codebook TCQ

The easiest method for entropy coding TCQ is to use uniform codebooks and minimum MSE encoding. In fixed rate TCQ the number of codewords is 2^{R+1} where R is the rate. In entropy coded TCQ the number of codewords is $\sim 2^{R+5}$ with an expected encoding rate of R . The entropy or rate is controlled by the spacing of the uniform codewords (i.e., the step size). For small step sizes, the entropy of the codebook will be large and the MSE of an encoded sequence small. As the step size increases, the codebook entropy decreases and the MSE of encoding increases.

Sweeping the step size from small to large values generates a rate/step size curve from which step sizes can be selected for any desired rate. When encoding real data, the rate will deviate from the expected rate. Entropy codes are probabilistic and developed for an assumed model. Any mismatch between the distribution of the source and the model will effect the actual encoding rate and the performance of the quantizer. The encoding performance for uniform codebooks for the Generalized Gaussian sources of $\alpha = 0.75$ and $\alpha = 1.5$ are plotted in Figure 4.9. The performance at high rates (> 2.5 bits) is very good but deteriorates at lower rates. The $\alpha = 0.75$ codebook was designed at low rates to demonstrate the deficiency of uniform codebooks at low rates. However, uniform codebooks will not be used at low rates. The performance gains over fixed-rate TCQ are evident from the rate distortion functions plotted in Figures 4.7 and 4.9.

Figure 4.9: Uniform Codebook TCQ Performance

Figure 4.10: Entropy-Constrained TCQ Performance

4.4.2 Optimized ECTCQ

Uniform codebooks and simple minimum MSE encoding are not sufficient for a low rate entropy constrained TCQ system. A significant improvement can be realized by minimizing the MSE subject to a constraint on average codeword length i.e., a rate constraint. The codebook design algorithm is a modification of the generalized Lloyd algorithm implemented in [38] for designing codebooks for vector quantizers. Two codebooks are designed, one for each superset used in the trellis.

Optimum codebooks are designed by minimizing the following functional:

$$J = E[\rho(x, \hat{x})] + \lambda E[l(\hat{x})] \quad (4.10)$$

where $\rho(x, \hat{x})$ is the cost (MSE) of quantizing x by $\hat{x}$, λ is a Lagrange multiplier, and $l(\hat{x})$ is the number of bits used to code $\hat{x}$. The entropy constraint is introduced by setting $l(\hat{x}) = -\log_2 P(\hat{x})$, the probability of selecting $\hat{x}$. The entropy rate of the codebook is controlled by the value of the Lagrange multiplier. If the Lagrange multiplier is small the portion of the functional value due to the length constraint is small and thus the rate will be high. If λ is large, the rate constraint has a larger effect on the functional and the rate will be low.

There is no deterministic method for selecting λ for a desired rate. Similar to the uniform codebooks, λ is swept from small to large values to generate rate versus λ curve from which λ 's can be selected for any desired rate. Developing codebooks with precise rates requires considerable fine tuning of the λ values.

The algorithm for minimizing the functional is a variant of the generalized Lloyd algorithm for vector quantizer design.

1. Initialize the codebook with uniform and equiprobable codewords
2. Quantize the training sequence using TCQ and the functional J as the distortion measure for choosing the correct codeword from a subset.
3. Compare J^i , the i^{th} functional value, with the previous iteration J^{i-1} . if $\frac{(J^{i-1}-J^i)}{J^i} < \epsilon$ stop, else continue.
4. Compute new codewords, probabilities, and lengths. Each codeword is recomputed as the centroid of all the values encoded with that codeword. The probabilities are computed as the probability of choosing that particular codeword conditioned on the probability of the particular superset. If a codeword is unused, it is given zero probability and infinite length and is never selected again.
5. Using the new codewords and lengths, go back to step 2.

The result of the training algorithm is a codebook having codewords with non-zero probabilities and finite lengths. The encoding rule for optimized ECTCQ is slightly different from fixed rate TCQ. When a codeword is selected from a subset, instead of selecting the codeword with the smallest MSE, the codeword having the smallest functional value J is chosen.

The actual rate of the encoded sequence is computed using the actual coded probabilities of the codewords, not the design probabilities. If there is substantial mismatch

between the training model and the source, the rate can deviate substantially from the expected rate.

Figure 4.10 shows the encoding performance for optimized ECTCQ for the two generalized Gaussian sources. The performance at low rates is far superior to uniform codebook TCQ. At high rates, optimum codebooks converge to the uniform codebooks. Entropy-constrained TCQ provides near rate distortion function performance at all positive rates with only modest computational complexity. A 4-state system encoding a Gaussian source is within 0.55 dB of the rate distortion function at all positive rates (see Figure 4.8).

4.4.3 Entropy Coding

To actually realize the rates for the entropy constrained systems, the bit streams from the trellis must be encoded with either a Huffman code or an Arithmetic code. The Huffman coder [2] is simpler than the arithmetic coder [39], but the performance of the arithmetic coder is superior. All contemporary source coding algorithms employ either the Huffman or arithmetic coder. The complexity of implementing either a Huffman or arithmetic code is roughly equivalent for all compression systems and can thus be viewed as a constant computational cost when comparing encoding algorithms. In this work, neither algorithm was implemented in conjunction with the encoder.

CHAPTER 5

RATE ALLOCATION

5.1 Theory

Rate allocation is a very important element of any data compression system. Typically, there are a number of sequences of equal length or coefficients that must be quantized such that the average rate is equal to some preselected rate. The problem is posed as:

$$\begin{aligned} \min \quad & D \\ \ni \quad & \frac{1}{N} \sum_i R_i = R \end{aligned} \tag{5.1}$$

For a simple system, such as quantizing N DCT coefficients and assuming a quantization distortion model of $D = \epsilon_i \sigma_i^2 2^{-2R_i}$, the problem is [7] :

$$\begin{aligned} \min \quad & \sum_{i=1}^N \epsilon_i \sigma_i^2 2^{-2R_i} \\ \ni \quad & \frac{1}{N} \sum_i R_i = R \end{aligned} \tag{5.2}$$

The distortion in each coefficient is independent of the distortion in any other coefficient. Using the Lagrange multiplier method [7]

$$R_k = R + \frac{1}{2} \log_2 \frac{\sigma_k^2}{[\prod_{j=0}^{N-1} \sigma_j^2]^{\frac{1}{N}}} \tag{5.3}$$

This must be modified slightly to assure $R_k > 0$. Given such a rate allocation the error variance for all of the coefficients will be equal. For any other allocation, the error variances will not be equal and the distortion will increase.

5.2 Adaptive RIDPCM (ARIDPCM) Model

In Adaptive RIDPCM, the image is partitioned into $IJ+1$ sequences (I recursions, J classes). The subsample pixels are ordered sequentially left-to-right, top-to-bottom into a sequence. At each of the I recursions, the error values are separated into J sequences according to class label. After quantizing, the error values in each of the J sequences are unsorted according to class label into their original positions.

The rate allocation problem is then to quantize a number of sequences of varying lengths at a given average rate with minimal distortion. Mathematically, the problem can be stated as follows. Given the sequences to be encoded, having lengths $m_0, m_{i,j}, i = 1, \dots, I, j = 1, \dots, J$ and quantizers that yield MSEs of $d_0(R_0)$ and $d_{ij}(R_{ij})$ when operating on the 0^{th} and $(i,j)^{th}$ sequence at R_0 and R_{ij} bits per sample, respectively, choose the encoding rates such that

$$D = m_0 d_0(R_0) + \sum_i \sum_j m_{ij} d_{ij}(R_{ij}) \quad (5.4)$$

is minimized, subject to the constraint that

$$\frac{1}{M} (m_0 R_0 + \sum_i \sum_j m_{ij} R_{ij}) = R$$

where $M = m_0 + \sum_i \sum_j m_{ij}$ is the total number of samples to be encoded.

Histograms of the image subsamples and interpolation error sequences indicate that the data are reasonably modeled as generalized Gaussian with exponential parameters of 1.5 and 0.75, respectively. Thus, their p.d.f.'s are somewhat more peaked and heavy tailed than those Gaussian and Laplacian densities, respectively.

The MSE of most quantizers can be modeled as

$$d(R) = \epsilon(R)\sigma^2 2^{-2R} \quad (5.5)$$

where σ^2 is the variance of the sequence to be encoded and $\epsilon(R)$ is a smooth function that converges to some constant γ for large R and is 1.0 when $R = 0.0$ [7]. The model chosen for $\epsilon(R)$ in this work is given by:

$$\epsilon(R) = (\gamma + (1 - \gamma)2^{-\beta R}). \quad (5.6)$$

The value of γ can be determined by examining Figure 5.1 (in which 2^{-2R} is plotted along with the actual MSE performance for TCQ and ECTCQ respectively for the unit variance generalized Gaussian source with $\alpha = 0.75$) and noting that γ is the asymptotic difference between 2^{-2R} and the actual MSE performance of the particular quantizer of interest. For example, we see from Figure 5.1 that for TCQ, $\gamma \approx 4.7$ dB, or $\gamma \approx 2.93$. Similarly, for ECTCQ, $\gamma \approx 0.8$. For the $\alpha = 1.5$ source, we find that $\gamma = 2.4$ for TCQ and $\gamma = 1.1$ for ECTCQ.

For the $\alpha = 1.5$ source (subsample sequence), it was found that the encoding rate is always high enough to let $\epsilon(R) = \gamma$. Thus, $\epsilon_0(R_0) = 2.4$ for TCQ, and $\epsilon_0(R_0) = 1.1$ for ECTCQ.

For the $\alpha = 0.75$ source (interpolation residuals), the parameter β can be found by minimizing (w.r.t. β)

$$\sum_R (\epsilon(R)2^{-2R} - D(R))^2 \quad (5.7)$$

where the sum is over the rates achievable by the quantizer and $D(R)$ is the actual distortion obtained at those rates. This procedure yields $\beta = 0.53$ for TCQ and $\beta = 12.7$ for ECTCQ. Figure 5.2 plots the actual performance of TCQ and ECTCQ along with the modeled performance for the $\alpha = 0.75$ source.

With reference to equation (5.4), we now see that $d_0(R_0) = \gamma\sigma_0^2 2^{-2R_0}$, with γ equal 2.4 or 1.1 for TCQ or ECTCQ respectively. Also $d_{ij}(R_{ij}) = (\gamma + (1 - \gamma)2^{-\beta R})\tilde{\sigma}_{i,j}^2 2^{-2R_{ij}}$ where γ and β are 2.93 and 0.53 (for TCQ) or 0.8 and 12.7 (for ECTCQ) respectively, and $\tilde{\sigma}_{i,j}^2$ is the variance of the interpolation error data in the i^{th} recursion and j^{th} class. If σ_0^2 and $\tilde{\sigma}_{i,j}^2$ were known, the optimization problem of (5.4) could then be solved using standard techniques. Unfortunately, since the interpolation error values are computed recursively using previously quantized data, the value of $\tilde{\sigma}_{i,j}^2$ depends on the quantization errors incurred in previous recursions. Therefore, to optimize the rate allocation, we must develop an estimate for the variance of each error sequence that incorporates the dependencies prescribed by the Adaptive RIDPCM algorithm.

For the general case of the i^{th} recursion and j^{th} class, let X_{kl} = actual pixel value, P_{kl}^i = pixel value interpolated from the original image if no quantization errors are introduced in previous recursions, $\tilde{P}_{kl}^i$ = pixel value interpolated assuming full quantization in previous recursions, $e_{kl}^i = X_{kl} - P_{kl}^i$, $\tilde{e}_{kl}^i = X_{kl} - \tilde{P}_{kl}^i$ and q_{kl}^i = the

Figure 5.1: Encoding Performance for TCQ and ECTCQ, $\alpha = 0.75$

Figure 5.2: Distortion Model for ECTCQ

Figure 5.3: Estimated Variances

quantization error incurred in quantizing $\tilde{e}_{kl}^i$. It then follows that $\hat{e}_{kl}^i = \tilde{e}_{kl}^i + q_{kl}^i =$ quantized version of $\tilde{e}_{kl}^i$ and $\hat{X}_{kl} = \tilde{P}_{kl}^i + \hat{e}_{kl}^i = X_{kl} + q_{kl}^i =$ quantized version of X_{kl} . Figure 5.3 depicts the adaptive algorithm and the estimated variances.

At each recursion, one-third of the interpolated values are computed using each of the following three equations:

$$\tilde{P}_{kl}^i = \frac{\hat{X}_{k,l-1}^{i-1} + \hat{X}_{k,l+1}^{i-1}}{2}, \quad (5.8)$$

$$\tilde{P}_{kl}^i = \frac{\hat{X}_{k-1,l}^{i-1} + \hat{X}_{k+1,l}^{i-1}}{2}, \quad (5.9)$$

and

$$\tilde{P}_{kl}^i = \frac{\hat{X}_{k-1,l-1}^{i-1} + \hat{X}_{k-1,l+1}^{i-1} + \hat{X}_{k+1,l-1}^{i-1} + \hat{X}_{k+1,l+1}^{i-1}}{4} \quad (5.10)$$

depending on whether the nearest neighbors are along a row, column, or the four diagonals¹. Note that we assume all pixels in (5.8)-(5.10) are from class j . Near subimage boundaries a few pixels may actually be from a different class. The effect of this is of secondary importance and is ignored here. Substituting,

$$\hat{X}_{kl}^{i-1} = X_{kl}^{i-1} + q_{kl}^{i-1} \quad (5.11)$$

in (5.8) - (5.10) yields

$$\tilde{P}_{kl}^i = P_{kl}^i + \frac{q_{k,l-1}^{i-1} + q_{k,l+1}^{i-1}}{2}, \quad (5.12)$$

$$\tilde{P}_{kl}^i = P_{kl}^i + \frac{q_{k-1,l}^{i-1} + q_{k+1,l}^{i-1}}{2}, \quad (5.13)$$

and

$$\tilde{P}_{kl}^i = P_{kl}^i + \frac{q_{k-1,l-1}^{i-1} + q_{k-1,l+1}^{i-1} + q_{k+1,l-1}^{i-1} + q_{k+1,l+1}^{i-1}}{4}. \quad (5.14)$$

Thus,

$$\tilde{e}_{kl}^i = e_{kl}^i + \frac{q_{k,l-1}^{i-1} + q_{k,l+1}^{i-1}}{2}, \quad (5.15)$$

$$\tilde{e}_{kl}^i = e_{kl}^i + \frac{q_{k-1,l}^{i-1} + q_{k+1,l}^{i-1}}{2}, \quad (5.16)$$

and

$$\tilde{e}_{kl}^i = e_{kl}^i + \frac{q_{k-1,l-1}^{i-1} + q_{k-1,l+1}^{i-1} + q_{k+1,l-1}^{i-1} + q_{k+1,l+1}^{i-1}}{4}. \quad (5.17)$$

Computing an average value of $E[(\tilde{e}_{kl}^i)^2]$ over the three types of pixels yields,

$$E[(\tilde{e}_{kl}^i)^2] = \frac{1}{3}(E[(e_{kl}^i)^2] + 2E[(e_{kl}^i)(\frac{q_{k,l-1}^{i-1} + q_{k,l+1}^{i-1}}{2})] + E[(\frac{q_{k,l-1}^{i-1} + q_{k,l+1}^{i-1}}{2})^2])$$

¹Note that for computational purposes, interpolated pixels of the type in (5.10) can be computed by averaging a pixel computed by (5.8) with a pixel computed by (5.9). Thus each type of interpolated pixel can be computed using one addition and one shift, as claimed previously

$$\begin{aligned}
& + \frac{1}{3}(E[(e_{kl}^i)^2] + 2E[(e_{kl}^i)(\frac{q_{k-1,l}^{i-1} + q_{k+1,l}^{i-1}}{2})] + E[(\frac{q_{k-1,l}^{i-1} + q_{k+1,l}^{i-1}}{2})^2]) \\
& + \frac{1}{3}(E[(e_{kl}^i)^2] + 2E[(e_{kl}^i)(\frac{q_{k-1,l-1}^{i-1} + q_{k-1,l+1}^{i-1} + q_{k+1,l-1}^{i-1} + q_{k+1,l+1}^{i-1}}{4})] \\
& + E[(\frac{q_{k-1,l-1}^{i-1} + q_{k-1,l+1}^{i-1} + q_{k+1,l-1}^{i-1} + q_{k+1,l+1}^{i-1}}{4})^2]). \tag{5.18}
\end{aligned}$$

Assuming that the error sequence and the quantization errors are zero-mean, stationary, and uncorrelated², we get

$$E[(\tilde{e}_{kl}^i)^2] = E[(e_{kl}^i)^2] + \frac{5}{12}E[(q_{kl}^{i-1})^2] \tag{5.19}$$

Recognizing that (5.19) is the variance of the interpolation sequence in the i^{th} recursion and j^{th} class, and that $E[(e_{kl}^i)^2]$ is the variance of the same sequence if no quantization is present (i.e., $\tilde{\sigma}_{i,j}^2 = E[(\tilde{e}_{kl}^i)^2]$ and $\sigma_{i,j}^2 = E[(e_{kl}^i)^2]$), we have,

$$\tilde{\sigma}_{i,j}^2 = \sigma_{i,j}^2 + \frac{5}{12}\epsilon_{i-1,j}(R_{i-1,j})\tilde{\sigma}_{i-1,j}^2 2^{-2R_{i-1,j}} \tag{5.20}$$

where the distortion model has been substituted for $E[(q_{kl}^{i-1})^2]$. The rate allocation problem can now be restated as follows. Find $R_0, R_{i,j} i = 1, \dots, I, j = 1, \dots, J$ such that

$$D = m_0\epsilon_0(R_0)\sigma_0^2 2^{-2R_0} + \sum_{i=1}^I \sum_{j=1}^J m_{i,j}\epsilon_{i,j}(R_{i,j})\tilde{\sigma}_{i,j}^2 2^{-2R_{i,j}} \tag{5.21}$$

is minimized, subject to the constraint that

$$\frac{1}{M}(m_0 R_0 + \sum_{i=1}^I \sum_{j=1}^J m_{i,j} R_{i,j}) = R$$

and

$$R_0, R_{i,j} \geq 0$$

²It is well known that this assumption is invalid at low encoding rates. However, our experimental results indicate that the ultimate results (equation(5.20)) is quite accurate for this application.

where

$$M = m_0 + \sum_{i=1}^I \sum_{j=1}^J m_{ij},$$

$$\epsilon_{i,j}(R_{i,j}) = (\gamma + (1 - \gamma)2^{-\beta R_{i,j}}),$$

and

$$\tilde{\sigma}_{i,j}^2 = \sigma_{i,j}^2 + \frac{5}{12} \epsilon_{i-1,j}(R_{i-1,j}) \tilde{\sigma}_{i-1,j}^2 2^{-2R_{i-1,j}}.$$

A closed form solution to this problem could not be found. Iterative programming algorithms were used to solve for the optimum rate allocation.

5.3 Integer Programming Solution for TCQ

Because TCQ is a fixed rate system, the rates $R_0, R_{i,j}$ must be integer valued. Adding an integer rate constraint to equation 5.21 changes the problem from a general optimization problem to an integer programming problem. This problem is solved using an integer programming algorithm with a non-linear objective function and linear constraints [40]. Due to the integer constraint, the error variances of the quantized sequences will not be equal and the distortion will be greater than or at best equal to the original problem. By adding the integer constraint, the feasible solution space has decreased so that the minimum of the objective function can only get larger.

For example, prescribing a rate of 1 bit for “lena” produces Table 5.1 which corresponds to a rate of 0.98 bits per pixel. Encoding of “lena” using Adaptive

Table 5.1: Integer Programming Solution for Lena at 1 Bit

	Recursion		
	1	2	3
Class A	2 bits	2 bits	0 bits
Class B	3	3	1
Class C	4	4	2
Class D	5	4	3
	Subsample = 4		

RIDPCM with fixed rate TCQ gives a quantized rate of 0.98 bits per pixel and a transmitted rate of 1.01 bits per pixel at 36.67 dB PSNR.

5.4 Conjugate Gradient Solution for ECTCQ

In theory the rates in ECTCQ can take any rate greater than zero. In this work however, codebooks for 0 to 12.7 bits at intervals of one-tenth were designed. The problem could be reformulated to constrain all rates to tenths, but the error in simply rounding the unconstrained rates to the nearest tenth is small. Therefore the optimization problem in equation 5.21 having a non-linear objective function with linear constraints is solved. Since the objective function D has an analytic gradient, conjugate gradient methods are employed to solve the problem [41]. The rates are then rounded to the nearest tenth.

Table 5.2: Conjugate Gradient Solution for Lena at 1 Bit

	Recursion		
	1	2	3
Class A	1.68 bits	1.21 bits	0.36 bits
Class B	2.93	2.29	1.34
Class C	3.41	2.78	1.78
Class D	3.72	3.32	2.35
	Subsample = 4.15		

For example, selecting a rate of one bit for “lena” produces the rate allocation in Table 5.2. Rounding the rates to the nearest tenth and coding the image produces a quantized rate of 0.97 bits and a transmitted rate of 1.0 bits per pixel at 39.78 dB PSNR.

5.5 Encoding Rate Computation

The overall transmitted bit rate for the compression system is made up of several components. The primary component of the overall rate is the number of bits required to quantize the image subsamples and interpolation errors. For an N by N image and I recursions, the subsample length is $K = 2^I$. The number of pixels in the subsample sequence is then $\frac{N^2}{K^2}$. The number of interpolated pixels in the first recursion is three times the number in the subsample sequence, and each successive recursion produces

four times as many interpolated pixels as the previous recursion. Thus, the average encoding rate is given by

$$\begin{aligned} R &= \frac{N^2}{K^2} [R_0 + 3R_1 + 12R_2 + 48R_3] / N^2 \\ &= (1/K^2) (R_0 + 3 \sum_{i=1}^I 4^{i-1} R_i) \end{aligned} \quad (5.22)$$

where

$$R_i = \frac{\sum_{j=1}^J m_{i,j} R_{i,j}}{\sum_{j=1}^J m_{i,j}}, \quad (5.23)$$

$R_{i,j}$ is the bit rate and $m_{i,j}$ is the length of data from the j^{th} class at the i^{th} recursion.

A very small component of the transmitted bit rate is due to overhead bits. The overhead includes bits for transmitting the rate table, subimage labels, and the mean and variance of each quantized sequence at each recursion. The subimage label bits dominate the overhead. Transmitting the subimage labels for a 512 by 512 image with I recursions (subsample length of $K = 2^I$) and J classes requires $(N^2/K^2) \log_2 J$ bits. For typical values of $N=512$, $I=3$, and $J=4$, 8,192 bits, or 0.03 bits/pixel, are required. Huffman coding will reduce this number in most cases.

CHAPTER 6

ADAPTIVE RIDPCM-TCQ

The adaptivity, rate allocation, and fixed rate TCQ are integrated into a compression coder exhibiting performance superior to RIDPCM with a decoder no more complex.

6.1 Implementation

Figure 6.1 is a block diagram of the encoder of Adaptive RIDPCM with TCQ. The original image is first put through the feature extractor to measure the edge density in each subimage. The classifier assigns each subimage to one of the four predetermined classes. The classification information is coded and sent to the decoder. The class labels and original image provide the input to the rate allocation system which delivers the rates to the subsample TCQ and the interpolation residual TCQ. The rate table is also coded and sent to the decoder.

Once the rate allocation is completed, the image is coded recursively. The original image is downsampled by a factor of K , e.g. 8, and the subsamples quantized, assigned codewords, and sent to the decoder. The quantized subsampled image is upsampled by a factor of two and then interpolated using filter h , see Figure 2.5. The original image is downsampled by a factor of $K/2^i$ where i is the recursion index. The

Figure 6.1: Adaptive RIDPCM-TCQ Encoder

interpolated pixel values are then subtracted from their corresponding pixel values in the original image to form a sequence of interpolation residuals. Referring to Figure 6.2, the sequence of interpolation residuals is sorted into four separate sequences by class. Each of the four sequences is quantized using TCQ at the rate prescribed by the allocation table. The four quantized sequences are assigned codewords and sent to the decoder. The quantized pixel values are unsorted into their original order and added to the interpolated pixel values to form an intermediate quantized image. The intermediate quantized image is upsampled, filtered, subtracted from corresponding pixels in the original image, quantized and summed to form the next intermediate quantized image. This continues recursively until the entire image is quantized.

Clearly, the encoder complexity has increased substantially. Extracting and classifying features, replacing scalar quantizers with TCQ, and using a integer program to perform the rate allocation requires a significant number of computations.

6.2 Decoder Complexity

Although the encoder complexity has increased significantly, the complexity of the decoder (Figure 6.3) remains minimal. The encoded bit sequence for the subsamples includes one bit for tracing through the trellis and $R-1$ bits for selecting the proper codeword from the designated subset where R is the number of bits allocated to the subsamples. The decoder uses the same convolutional coder as the encoder to

Figure 6.2: Encoder Quantization Algorithm

generate the quantized subsample sequence from the bit sequence. The quantized sequence forms the quantized subsample image.

In each successive recursion, zero's are inserted between each pixel and interpolated values for each zero are computed. At each recursion, each class of pixels is encoded in a separate bit sequence. As with the subsample sequence, the convolutional coder decodes each bit sequence into a sequence of quantized pixel values. Using the transmitted class labels for each pixel to match the quantized pixels with their corresponding interpolated pixels, the quantized pixel values are added to the interpolated values to form the next quantized image.

The convolution coder requires only a shift register and a look-up table (i.e., no arithmetic operations) to decode each codeword into the correct quantization level. Essentially, the number of operations per pixel has not changed from the original algorithm i.e., 2 adds and 1 shift.

6.3 Performance

Adaptive RIDPCM-TCQ outperforms standard RIDPCM on "lena" by approximately 3 dB PSNR at 1 bit and 3.5 dB at .3 bits. The visual quality of the encoded images shown in Figure 6.4 is greatly enhanced with the edges being much sharper and the background noise reduced substantially. The results are similar for paglady. For the urban images, ARIDPCM-TCQ outperforms RIDPCM but the margin is significantly smaller. This is attributable to the large amount of high frequency

Figure 6.3: Adaptive RIDPCM-TCQ Decoder

structure in the urban images. The higher edge density classes contain far more subimages than they do in the portrait type images. This causes the rate allocation scheme to assign much lower rates to all thirteen sequences to meet the preselected rate. Many pixels are quantized with 0 bits i.e., the error values are not coded, and TCQ has no advantage over scalar quantizers in this case. By observing the encoding performance, Figure 6.5, you can see that as the rate increases the adaptive system outperforms RIDPCM by a wider margin. The adaptive system has more flexibility and more pixels are being quantized at non-zero rates.

The advantages of Adaptive RIDPCM-TCQ are improved SNR and visual performance, especially edge representation, at all rates and images tested. However, the degree of improvement is affected by the type of imagery and compression rate. For portrait type images, the improvement is uniform from 0.3 to 1.0 bits. For higher frequency urban images, the improvement is minimal at low rates but improves significantly near 1 bit. And although the encoder complexity has risen, the decoder complexity remains minimal.

The disadvantages are the additional encoder complexity and approximately 0.03 bits of additional overhead. As compared to transform coders, the performance is still inferior but the decoder complexity is much simpler.

Figure 6.4: 1 bit ARIPDCM-TCQ: (a)Lena, (b)Urban26, (c)Lena Difference Image, (d)Urban26 Difference Image

Figure 6.5: Adaptive RIDPCM-TCQ Performance

CHAPTER 7

ADAPTIVE RIDPCM-ECTCQ

The adaptivity, rate allocation, and entropy constrained TCQ are integrated into a source encoder exhibiting performance vastly superior to RIDPCM, superior to the fixed rate system, and in the range of the most complex transform coders available to date.

7.1 Implementation

The encoder for Adaptive RIDPCM-ECTCQ is exactly the same as the fixed rate encoder shown in Figure 6.1 with TCQ replaced by ECTCQ. At rates greater than 3 bits, uniform codebooks are used. At lower rates, optimized codebooks are employed. Of course, the codewords for the subsamples and residual sequences are entropy coded with a Huffman or arithmetic coder instead of the fixed rate coding associated with TCQ. Since the encoding is probabilistic, the actual rate depends on the particular sequence produced by the ARIDPCM algorithm for a given image. The rate is computed as $\sum_{i=1}^N \hat{p}_i \ln p_i$ where p_i is the codeword probability and $\hat{p}_i$ is the relative frequency that the i^{th} codeword was selected to encode that particular sequence conditioned on the probability of the associated superset. If the distribution of the input sequence exactly fits the model used to generate the codebooks, then

$p_i \simeq \hat{p}_i$ for very long encoded sequences and the actual encoding rate should be very close to the expected encoding rate. For image data, it is very unlikely that the subsample sequence or any residual sequence will have a distribution that matches a generalized gaussian model perfectly. Therefore, the selection of an appropriate model is very important to quantizer performance as regarding the accuracy of the actual rate and the MSE of the quantized signal. Using a generalized gaussian of $\alpha = 1.5$ for the subsamples and an $\alpha = 0.75$ for the residual sequences performs well.

7.2 Decoder Complexity

The decoder for the entropy constrained TCQ system is the same as the decoder for the fixed rate system except that either a Huffman or arithmetic decoder must be used to decode the encoded bit sequences. Relative to the fixed rate system, the decoder complexity has increased. However, all high performance contemporary coders employ entropy coding and thus bear a similar computational burden in the decoder. When comparing decoder complexity between systems, the Huffman or arithmetic decoder complexity is viewed as a constant and can be ignored. The true comparison is between the decoder complexity of the RIDPCM system and the decoder complexity of the other advanced compression systems.

7.3 Performance

In the development of the rate allocation system, the distortion function was dependent on a model for the quantization error in a sequence, a model for epsilon, and a derivation for the predicted quantized variances. An assumption that the error sequence and the quantization errors are uncorrelated was made. If these models, derivations, and assumptions are valid, the actual quantized variances for each sequence should be close to the predicted quantized variances and the MSE for each sequence should be about equal. Table 7.1 shows the predicted quantized variances, actual quantized variances and associated squared errors for each sequence. The actual variances are very close to the predicted variances which validates the model and assumptions made in the algorithm. The error terms are approximately equal, the deviations being caused by the rounding of the rates and the non-ideal properties of the actual sequences of image data. Table 7.2 shows the correlation of the subsample and interpolation error sequences. The correlation of the subsampled image is 0.68. The correlation of the error sequences ranges from 0.02 to 0.26. The residual correlation in the higher edge density classes is less than the residual correlation in the lower edge density classes.

The performance of Adaptive RIDPCM-ECTCQ is shown in Figure 7.1. The performance for “lena” is much better than the performance for the other images. The only reason for this is the structure and frequency content of “lena”. This image

Table 7.1: Predicted vs. Actual Encoding Statistics, Lena at 1-Bit

sequence	Predict σ^2	Actual σ^2	error
1	1680	1680.4	6.32
2	57.45	57.67	4.71
3	442.45	442.91	6.99
4	887.45	882.34	6.57
5	1354.45	1352.52	6.56
6	23.87	24.53	4.25
7	160.53	161.93	5.75
8	344.61	344.77	6.02
9	759.61	760.97	6.50
10	12.49	13.45	7.33
11	52.25	53.97	8.01
12	95.43	97.16	6.94
13	211.56	211.88	7.61

Table 7.2: Residual Correlation of Error Sequences

	Recursion		
	1	2	3
Class A	0.16	0.22	0.26
Class B	0.11	0.13	0.04
Class C	0.11	0.13	0.04
Class D	0.10	0.13	0.02
	Subsample = 0.68		

is simply much easier to compress than “paglady”, “urban26”, or “urban28”. The performance for the entropy coded system is between one and four dB better at all rates than the performance for the fixed rate system. This improvement in MSE is reflected in the visual quality of the images. The granular noise has decreased significantly. The edge quality and reproduction of high frequency detail is much sharper throughout the test imagery. Figure 7.2 shows “lena” quantized at 1.0 bits and the corresponding difference image, and urban26 quantized at 1.0 bits and the corresponding difference image. The visual quality is much improved. The difference image exhibits a lot less structure than one resulting from RIDPCM. Ideally the difference image is uniformly random for optimal compression.

Adaptive RIDPCM-ECTCQ provides substantial MSE distortion gains as well as visual quality over fixed rate TCQ. The addition of the entropy coder does increase

Figure 7.1: Adaptive RIDPCM-ECTCQ Performance

Figure 7.2: 1 bit ARIDPCM-ECTCQ: (a)Lena, (b)Urban26, (c)Lena Difference Image, (d)Urban26 Difference Image

the complexity of the decoder but is necessary to compete on a performance basis with other contemporary compression coders which also use similar entropy codes. The complexity of the decoder is still significantly less than coders meeting or exceeding the performance of ARIDPCM with ECTCQ.

CHAPTER 8

RESULTS

The comparative performance (SNR, visual, and complexity) of RIDPCM, Adaptive RIDPCM-TCQ, and Adaptive RIDPCM-ECTCQ are evaluated in this chapter. The performance of Adaptive RIDPCM-ECTCQ as compared to other high performance compression systems is provided. Lastly, the overall performance advantages and disadvantages of Adaptive RIDPCM-ECTCQ are analyzed.

The standard measure of performance for a compression system is the SNR or PSNR of the quantized signal at a given bit rate. Figures 8.1-8.4 are functions measured in PSNR for each of the four test images for the three RIDPCM systems. The improvement from RIDPCM, adding adaptivity and fixed rate TCQ, to adaptivity and entropy-constrained TCQ is consistent and pronounced. On “lena” and paglady, fixed rate TCQ provides 2-4 dB of improvement over the base system. The largest improvement coming at the lowest rate. On the two urban images, fixed rate TCQ provides 0-1.5 dB improvement. This seemingly great disparity can be attributed to the vast difference in image structure of the two types of imagery.

Tables 8.1 and 8.2 are the rate tables for “lena” at 1 bit and urban28 at 1 bit. Notice, the rates for each sequence are greater for “lena” than urban28 and approximately 60% of urban28 is allocated only one-tenth of a bit. The performance gains for

Figure 8.1: RIDPCM, ARIDPCM-TCQ, ARIDPCM-ECTCQ: Lena

Figure 8.2: RIDPCM, ARIDPCM-TCQ, ARIDPCM-ECTCQ: Paglady

Figure 8.3: RIDPCM, ARIDPCM-TCQ, ARIDPCM-ECTCQ: Urban26

Figure 8.4: RIDPCM, ARIDPCM-TCQ, ARIDPCM-ECTCQ: Urban28

Table 8.1: ECTCQ Rate Table, Lena 1-Bit

	Recursion		
	1	2	3
Class A	1.7 bits	1.2 bits	0.4 bits
Class B	2.9	2.3	1.3
Class C	3.4	2.8	1.8
Class D	3.7	3.3	2.3
	Subsample = 4.1		

TCQ over scalar quantization are larger at higher rates and clearly at zero bits they are equivalent. Therefore, the structure and frequency content has a great impact on the performance of the adaptive algorithm and the performance gain realized by using TCQ instead of scalar quantization.

Adaptive RIDPCM-ECTCQ outperforms TCQ by 1-3 dB and RIDPCM by 3-7 dB for the portrait images and outperforms TCQ .5-1.5 dB and RIDPCM by .5-3 dB. The performance increase is attributable to two factors. The most obvious is that entropy-constrained TCQ simply outperforms fixed rate TCQ. Additionally, since the entropy-constrained codebooks are available for every tenth of a bit, the rate allocation system allocates bits in a more efficient manner thus increasing the performance of the compression system.

Table 8.2: ECTCQ Rate Table, Urban28 1-Bit

	Recursion		
	1	2	3
Class A	1.1 bits	0.7 bits	0.1 bits
Class B	1.5	1.3	0.4
Class C	1.8	1.6	0.7
Class D	2.1	1.9	1.1
	Subsample = 2.3		

8.1 Visual Quality

Although SNR is the standard measure for evaluating and comparing system performance, another measure of an image coder is the visual quality of the image [42], [29], [28]. The difficulty with evaluating visual quality is that it is not an easily measurable quantity but is highly subjective and dependent upon the end use of the image, e.g., aesthetic appreciation, or quantitative analysis. Certain types of errors are more or less offensive depending on how the image is viewed. A person viewing a picture, painting, or video might be very sensitive to granular noise or contouring errors whereas an analyst trying to identify specific structure in an image might be sensitive to overload errors or blurring in the imagery. Unfortunately, there is often a trade off between the performance on edges and detail, and granular noise and contouring. It is well known that SNR and visual quality are not perfectly correlated.

Therefore systems that maximize SNR do not necessarily maximize image quality. Currently, almost all systems are designed to maximize SNR because MSE is the only viable distortion measure. Depending on the encoding algorithm, this may or may not produce visual quality of an equally high level. Transform coders such as the DCT and Wavelet spread the errors evenly across many pixels thus typically producing better image quality at the same rate and SNR than predictive codes.

Figure 8.5 shows “lena” and the associated difference image coded at 0.5 bits using the RIDPCM and the Adaptive RIDPCM-ECTCQ algorithms. The RIDPCM algorithm induces serious blurring in the feather and brim of the hat. The entire image has a significant amount of granular noise. The adaptive ECTCQ algorithm provides much sharper edges and detail, and reduces the granular noise substantially. Some contouring or blotchiness is evident in the smooth regions of the image. Although these contoured regions have a smaller MSE than the noisy regions in RIDPCM, they can be more offensive to the eye from an aesthetic point of view. The overall image quality has improved but maybe not as much as the SNR’s would lead one to believe. At 0.5 bits, the difference is approximately 4.5 dB. This can be attributed to the fact that a lot of the SNR increase is realized by the vast improvement of the edges. However, at a cursory glance the edge and detail quality is not as noticeable as the background noise or contouring.

The performance improvement is reflected in the difference images. Using RIDPCM, there is a great deal of image structure reflected in the difference image. The

Figure 8.5: Lena at 0.5 bits: (a)RIDPCM, (b)ARIDPCM-ECTCQ, (c)RIDPCM Difference Image, (d)ARIDPCM-ECTCQ Difference Image

outline of the hat, feather, eyes, and shoulder are easily identifiable with relatively small errors in the background. The adaptive and entropy-constrained system tries to evenly distribute the errors throughout the image with less structure and more noise. Ideally, from the point of view of SNR, the difference image would be completely random. The difference between the two systems is very noticeable at one bit. The difference images shown in Figures 2.7 and 7.2 represent the difference images for RIDPCM and ARIDPCM-ECTCQ. Using RIDPCM, the difference image is a very sharp outline of all the major structure and detail in the image. The adaptive entropy-constrained coder removes all of the image structure and produces an almost “white” difference image. Figures 8.6 and 8.7 are difference image histograms of “lena” coded at one bit for RIDPCM and ARIDPCM-ECTCQ respectively. The histogram for RIDPCM has much longer tails than that for ECTCQ. The large errors occurring at edges in RIDPCM do not occur in the adaptive system. Figure 8.8 shows “lena” and the associated difference image coded at 0.5 bits using the Adaptive RIDPCM-TCQ and the Adaptive RIDPCM-ECTCQ algorithms. The entropy constrained system performs uniformly better with respect to edge reproduction, noise suppression and contouring. The difference images support these claims. Figure 8.9 shows “lena” compressed at 1.0, 0.75, 0.54, and 0.33 bits using the Adaptive RIDPCM-ECTCQ system. At 1.0 bits the PSNR is 39.78 dB and visual quality is very good. The large edges in the shoulder and hat are very sharp with no streaking

Figure 8.6: Difference Image Histogram: RIDPCM

Figure 8.7: Difference Image Histogram: ARIDPCM-ECTCQ

Figure 8.8: Lena at 0.5 bits: (a)ARIDPCM-TCQ, (b)ARIDPCM-ECTCQ, (c)TCQ Difference Image, (d)ECTCQ Difference Image

or blurring. The feather detail is crisp, and no granular noise or contouring is perceived. At 0.75 bits the PSNR is 37.95 and the overall image quality remains high. The edge and detail structure is very good. Some blotchiness is noticeable in the background and skin. At 0.54 bits the PSNR is 35.78 and the overall image quality has deteriorated noticeably. The detail and edge reproduction in the feather, hat, and shoulder are still very good, but the contouring evident in the background and especially her skin, is visually annoying. At 0.33 bits the PSNR is 33.17 and the image is very blotchy or mottled in places such as the background, shoulder, face, and texture in the hat. The edges remain quite sharp.

These results also suggest that at the higher rates, SNR and visual quality are highly correlated but at the lower rates the design criterion of maximizing SNR reproduces edges and high detail regions well by sacrificing the smoother and lower frequency regions to the detriment of the overall image quality. As stated above, the design algorithm is dependent on the MSE distortion measure, and one must hope that this translates into good image quality. These results seem to provide more evidence that this is not always true.

8.2 Other Compression Systems

A standard data compression system is JPEG [43] which employs 8 x 8 DCT's to encode the image. The DCT coefficients are normalized depending upon the specific application. The DC coefficient is coded using DPCM. The remaining 63 coefficients

Figure 8.9: ARIDPCM-ECTCQ Lena: (a)1.0 bits, (b).75 bits, (c).54 bits, (d).33 bits

are sorted using a zig-zag pattern into a 1-D vector in order of decreasing energy and coded using runlength codes. All of the codes are entropy coded using a Huffman code. The decoder performs an inverse entropy code, denormalizes each coefficient, and then performs an inverse DCT. The DCT alone costs about 8 operations per pixel. The performance of JPEG on a 512 x 512 “lena” is 37.93 dB PSNR at 1 bit and 34.69 dB PSNR at 0.5 bits [43].

An image coder employing the DCT and ECTCQ is detailed in [34]. The encoder uses the DCT to transform 16 x 16 blocks of an image. Like coefficients in each block are collected into sequences and encoded using ECTCQ. At the decoder, a 16 x 16 inverse DCT is needed to reconstruct the quantized image.

A discrete wavelet transform coder using ECTCQ is presented in [35]. The wavelet decomposition is achieved using 9-tap and 7-tap filters. The lowest frequency subimage is encoded with a 2-D DCT with block size 4 x 4. Like coefficients are collected into sequences and encoded using ECTCQ. Each of the high frequency sub-images is treated as a sequence and encoded using ECTCQ. At the decoder, a 4 x 4 inverse DCT must be performed to reconstruct the quantized lowest frequency sub-image. Thereafter, the quantized image is reconstructed using the 9 and 7-tap filters.

Figure 8.10 plots the encoding performance of “lena” for Adaptive RIDPCM-ECTCQ, JPEG, DCT-ECTCQ, DWT-ECTCQ. ARIDPCM-ECTCQ outperforms JPEG at all rates. At rates above 0.7 bits the Adaptive RIDPCM system outperforms the DCT coder and at rates below 0.7 bits the DCT coder is better. The Wavelet coder

Figure 8.10: Lena: ARIDPCM-ECTCQ, DCT-ECTCQ, DWT-ECTCQ

exhibits substantially better performance at all rates compared to these and all other coders. The visual quality of the Wavelet coder is correspondingly superior. The visual quality of the DCT is also superior to ARIDPCM-ECTCQ. The DCT distributes the quantization errors better than the interpolative algorithm and although the SNR at higher rates is similar the visual quality is superior.

However, SNR and visual quality are only two criteria for an encoding system. Decoder complexity is also a constraint in some systems. These three systems all employ ECTCQ which requires roughly the same number of computations at the decoder for each system. Whereas RIDPCM only requires 2 adds and 1 shift to decode the signal, JPEG requires 8 operations per pixel, the DCT-ECTCQ coder must perform an inverse DCT at the decoder at a cost somewhat in excess of 8 operations per pixel, and the DWT must perform an inverse DCT on the lowest frequency sub-image and then reconstruct the quantized image using repeated applications of the 9-tap and 7-tap 1-D filters. The number of operations to decompose the original image is 16 operations per pixel. The operations for creating the remaining images and DCT coding the lowest frequency image bring the total number of operations to near 20 operations per pixel. If the number of computations and memory requirements of the decoder are tightly constrained, the DCT and DWT although providing superior performance might not be viable options. In such a case, the Adaptive RIDPCM-ECTCQ coder could provide a high performance, low complexity solution.

Another encoding algorithm presented in [44] details an encoder using a three-component image model. The perceptually-motivated model comprises a primary component accounting for strong edges, a smooth component representing background, and a texture component for texture. The original image is filtered using a space-variant filter to create a “stressed” image. The difference between the original and stressed images represents the texture component. The primary component is derived from the stressed image by measuring the maximum curvature energy. The difference between the stressed image and primary component creates the smooth component. The primary component is encoded using chain codes representing the intensity and geometric information of the line features. The smooth and texture components are coded using a sub-band coder. The performance in SNR lies between the DCT-ECTCQ and DWT-ECTCQ coders shown in figure 8.10.

Another subband coder is given in [45]. Sixteen subbands are created by 32 point 1-D QMF filtering. The lowest frequency image is transform coded and then entropy coded. The remaining subimages are entropy coded using zero-memory quantizers. The decoder performs a variable length decoding, an inverse 2-D DCT on the lowest frequency image, and a 2D inverse QMF filtering. The performance is similar to the DCT-ECTCQ coder.

A hierarchical image coder having the property that image encoding and transmission is progressive was developed in [46]. The decoder can stop decoding at any time and still reconstruct a quantized image at a rate less than the target encoding

rate. The algorithm is based on three concepts: 1) hierarchical subband decomposition, 2) prediction of the absence of significant information across scales, and 3) hierarchical entropy-coded quantization. The algorithm performance is similar to the DCT-ECTCQ coder.

Another transform coding algorithm is presented in [47]. The image is partitioned into subblocks that are transformed using a 2-D DCT. The sequences of like DCT coefficients are encoded using tree codes. The performance lies between the DCT-ECTCQ and DWT-ECTCQ but the encoder complexity for any tree search is quite substantial. The decoder must still take an inverse DCT. An adaptive entropy-coded predictive vector quantization algorithm for coding images is presented in [48]. The performance is quite good lying just below the DWT-ECTCQ algorithm but the encoder has the complexity associated with all vector quantizers with the additional complexity of adaptivity and prediction.

CHAPTER 9

CONCLUSIONS AND FUTURE RESEARCH

In the introduction to this dissertation two questions motivating this research were posed. Can space-domain methods of image data compression be as effective in performance as transform-domain methods? What is the relative computational complexity of advanced space-domain compression methods, as compared to corresponding transform-domain methods.

The ARIDPCM-ECTCQ coder has been shown to have performance in terms of SNR similar to the DCT-ECTCQ coder but still inferior to the DWT-ECTCQ coder. For the DCT and DWT transform coders the SNR values are also a good measure of visual quality. For the RIDPCM algorithm, the SNR seems to overstate the visual quality of the image especially at lower rates. Whereas the transform coders evenly distribute the errors at all encoding rates, the predictive coder localizes the quantization errors. Therefore the visual quality of transform codes degrades gracefully with rate. For the interpolative coder, as the rate decreases the edge and detail regions maintain high quality but the background and smooth regions deteriorate badly thus causing an overall deterioration of image quality. On the sole criterion of performance, SNR and visual quality at the same bit rate, the Adaptive RIDPCM-ECTCQ image coder is not competitive with advanced transform coders.

Predictive or interpolative coders are just not capable of the performance achieved by transform coders. Transform coders provide better signal decorrelation and provide the advantage of distributing the coding errors across many pixels. In DCT, specific quantization errors are distributed across the block in which they occur. In DWT, the errors are distributed throughout the image.

However, the computational complexity of these advanced transform coding algorithms is much greater than the interpolative coder. The encoder complexity of Adaptive RIDPCM-ECTCQ has increased significantly with the inclusion of adaptive rate allocation and entropy-constrained TCQ. Compared to these other coding systems, the system encoder would be of moderately high complexity. Straight RIDPCM would be considered minimal complexity and Vector Quantization would be considered very high complexity. Of particular interest is the relative complexity of the decoders. The best performing coder is the DWT-ECTCQ which requires between 16 and 20 operations per pixel not including the entropy constrained decoding. The DCT-ECTCQ coder requires somewhat more than 8 operations per pixel again disregarding the entropy coding. By comparison, the ARIDPCM-ECTCQ algorithm requires only 3 operations per pixel not including the entropy coder. The wavelet coder outperforms the interpolative coder in SNR and visual quality but its encoder complexity is 6 to 7 times greater. The DCT coder maintains performance comparable to the interpolative coder at the cost of 3 times greater decoder complexity.

Therefore, if decoder complexity is an issue, Adaptive RIDPCM-ECTCQ can be a viable encoder. If the application required a remote, mobile unit for decoding transmitted compressed imagery there would be a constraint on complexity and memory requirements in the decoder but no requirements in the encoder. The algorithm may also be suitable for real-time video decoding of images stored in compressed form. Video could be stored on floppy, hard, or optical disk then decoded in real-time. In general, the high performance wavelet or subband coders are probably the wave of the future, but for very specific tasks, a low-complexity and reasonably high performance encoder might be very valuable.

Within the basic framework of the RIDPCM algorithm, I feel the addition of adaptive rate allocation and entropy-constrained TCQ has improved the SNR performance to very near the achievable upper bound. Slight improvements might be made by using a more complicated classifier, e.g. more or different features, more classes, etc. or by using more states in TCQ but with the cost of additional encoder complexity. Additional trellis states might also improve the visual quality of the reconstructed image slightly by decreasing the contouring effects. Higher state trellises were not investigated for two reasons. First, the improvements would be marginal (see Table 4.1), and second training codebooks for higher state ECTCQ would take a very long time. Although additional states does not increase decoder complexity.

It may be possible to improve the visual quality provided by the algorithm at the cost of reduced SNR performance. The obvious but most difficult way would

be to find a better distortion measure, one that is mathematically tractable and computationally simple [27]. This has been a focus of a lot of research for many years with limited success. A simpler method that might achieve some increase in visual quality would be to use the frequency response of the human eye [49] to select coefficients to weight the relative importance of the quantization distortion between the subsamples, and the first, second, and third recursion error sequences for rate allocation [50]. For this purpose, each recursion of the image can be considered as containing a different portion of the frequency spectrum. The center frequency of each band could be employed to select a weight from the MTF of the eye. As the center frequency increases, the MTF and the weight decreases. This would cause the algorithm to weight the distortion in each successive recursion less and less thereby allocating more bits to the earlier recursions. This would be very simple to implement. The SNR of the reconstructed image would have to decrease. The effect on perceived image quality is uncertain but promising.

Another approach to improving visual quality would be to weight the subimage classes. By de-emphasizing the high frequency classes and emphasizing the low frequency classes, the visual quality of the reconstructed image might improve at the cost of lower SNR.

The goal was to maximize the performance of a space domain compression system while maintaining an encoder of feasible complexity and most importantly a decoder of minimal complexity. The encoder complexity has increased substantially but is

still comparable with other contemporary encoders while the performance of the algorithm has improved dramatically and the decoder complexity remains minimal.

REFERENCES

- [1] R. Blahut, *Principles and Practice of Information Theory*. Addison-Wesley, 1991.
- [2] A. K. Jain, *Fundamentals of Digital Image Processing*. Prentice-Hall, 1989.
- [3] A. K. Jain, "Image Data Compression: A Review," *Proc. of the IEEE*, vol. 69, Mar. 1981.
- [4] A. Netravali, "On Quantizers for DPCM Coding of Picture Signals," *IEEE Transactions on Information Theory*, vol. IT-23, May 1977.
- [5] V. Algazi and J. DeWitte, "Theoretical Performance of Entropy-Coded DPCM," *IEEE Transactions on Communications*, vol. COM-30, May 1982.
- [6] L. Zetterbert, S. Ericsson, and C. Couturier, "DPCM Picture Coding with Two-Dimensional Control of Adaptive Quantization," *IEEE Transactions on Communications*, vol. COM-32, Apr. 1984.
- [7] N. S. Jayant and P. Noll, *Digital Coding of Waveforms*. Prentice-Hall, 1984.
- [8] O. Rioul and M. Vetterli, "Wavelets and Signal Processing," *IEEE SP Magazine*, Oct. 1991.
- [9] S. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 11, July 1989.
- [10] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image Coding Using Wavelet Transform," *IEEE Transactions on Image Processing*, vol. 1, pp. 205–220, Apr. 1992.
- [11] A. Gersho and R. Gray, *Vector Quantization and Signal Compression*. Kluwer, 1992.
- [12] R. Gray, "Vector Quantization," *IEEE Trans. on Acoustics, Speech and Signal Proc.*, Apr. 1984.

- [13] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*. McGraw Hill, 1991.
- [14] B. R. Hunt, "An Optical Analogy to DPCM Digital Image Data Compression," *SPIE Vol. 119: Applications in Digital Image Processing*.
- [15] B. R. Hunt, "Optical Computing for Image Bandwidth Compression: Analysis and Simulation," *Applied Optics*, vol. 15, pp. 2944–2951, Sept. 1978.
- [16] D. Y. Fu, "Adaptive Digital Image Data Compression by Recursive IDPCM," M.S. thesis, The University of Arizona, 1985.
- [17] T. Allan, "Adaptive Digital Image Data Compression using RIDPCM and a Neural Network for Subimage Classification," M.S. thesis, The University of Arizona, 1992.
- [18] Y.-H. Pao, *Adaptive Pattern Recognition and Neural Networks*. Addison-Wesley, 1989.
- [19] J. Hertz, A. Krogh, and R. Palmer, *Introduction to the Theory of Neural Computation*. Addison-Wesley, 1991.
- [20] R. P. Lippman, "An Introduction to Computing with Neural Nets," *IEEE Trans. on Acoustics, Speech and Signal Proc.*, Apr. 1987.
- [21] W. H. Chen and C. H. Smith, "Adaptive Coding of Monochrome and Color Images," *IEEE Transactions on Communications*, vol. COM-25, pp. 1285–92, Nov. 1977.
- [22] S. Peleg and J. Naor, "Multiple Resolution Texture Analysis and Classification," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, pp. 518–523, July 1984.
- [23] T. Peli, V. Tom, and B. Lee, "Multi-Scale Fractal and Correlation Signatures for Image Screening and Natural Clutter Suppression," *SPIE Visual Communications and Image Processing IV*.
- [24] J. A. Hartigan, *Clustering Algorithms*. Wiley-Interscience, 1975.
- [25] M. Anderberg, *Cluster Analysis for Applications*. Academic Press, 1973.

- [26] M. Marcellin and T. Fischer, "Trellis Coded Quantization of Memoryless and Gauss-Markov Sources," *IEEE Transactions on Communications*, vol. 38, Jan. 1990.
- [27] D. Sakrison, "On the Role of the Observer and a Distortion Measure in Image Transmission," *IEEE Transactions on Communications*, vol. COM-25, Nov. 1977.
- [28] J. Limb, "Distortion Criteria of the Human Viewer," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-9(12), Dec. 1979.
- [29] J. Saghri, P. Cheatham, and A. Habibi, "Image Quality Measure Based on a Human Visual System Model," *Optical Engineering*, vol. 28(7), July 1989.
- [30] M. V. Eyuboglu and J. G. D. Forney, "Lattice and trellis quantization with lattice- and trellis-bounded codebooks - high rate theory for memoryless sources," *IEEE Transactions on Information Theory*, vol. 39, pp. 46-59, Jan. 1993.
- [31] T. Lookabaugh and R. Gray, "High-Resolution Quantization Theory and the Vector Quantizer Advantage," *IEEE Transactions on Information Theory*, vol. 35, pp. 1020-1033, Sept. 1989.
- [32] M. W. Marcellin, "Trellis Coded Quantization: An Efficient Technique for Data Compression," Ph.D. dissertation, Texas AM University, 1987.
- [33] M. Marcellin, "On Entropy-Constrained Trellis Coded Quantization," *Accepted for publication in IEEE Transactions on Communications*.
- [34] M. Marcellin, P. Sriram, and K. Tong, "Transform Coding of Monochrome and Color Images using Trellis Coded Quantization," *IEEE Transactions Circuits Systems Video Technology*, Aug. 1993.
- [35] P. Sriram and M. Marcellin, "Image coding using Wavelet Transforms and Entropy-Constrained Trellis Coded Quantization," *submitted to IEEE Transactions on Image Processing*.
- [36] P. Sriram and M. Marcellin, "Image Coding using Wavelet Transforms and Entropy-Constrained Trellis Coded Quantization," in *Conf. on Acoust., Speech, and Signal Proc.*, pp. V554-557, Nov. 1993.

- [37] T. R. Fischer and M. Wang, "Entropy-Constrained Trellis Coded Quantization," *IEEE Transactions on Information Theory*, vol. 38, pp. 415–425, Mar. 1992.
- [38] P. Chou, T. Lookabaugh, and R. Gray, "Entropy-Constrained Vector Quantization," *IEEE Trans. on Acoustics, Speech and Signal Proc.*, vol. 37, Jan. 1989.
- [39] I. Witten, R. Neal, and J. Cleary, "Arithmetic Coding for Data Compression," *Communications of the ACM*, vol. 30, June 1987.
- [40] J. Ecker and M. Kupferschmid, *Introduction to Operations Research*. Wiley, 1988.
- [41] M. Avriel, *Nonlinear Programming: Analysis and Methods*. Prentice-Hall, 1976.
- [42] J. Mannos and D. Sakrison, "The Effects of a Visual Fidelity Criterion on the Encoding of Images," *IEEE Transactions on Information Theory*, vol. IT-20, July 1974.
- [43] M. Rabbani and P. Jones, *Digital Image Compression Techniques*. SPIE Press, 1991.
- [44] X. Ran and N. Farvardin, "Low Bit-Rate Image Coding using a Three-Component Image Model," tech. rep., SRC technical report, Univ of Maryland.
- [45] N. Tanabe and N. Farvardin, "Subband Image Coding using Entropy-Coded Quantization over Noisy Channels," *IEEE Journal on Selected Areas in Communications*, vol. 10, pp. 926–943, June 1992.
- [46] J. M. Shapiro, "An Embedded Wavelet Hierarchical Image Coder," in *Conf. on Acoust., Speech, and Signal Proc.*, Nov. 1992.
- [47] W. Pearlman, P. Jakatadar, and M. Leung, "Adaptive Transform Tree Coding of Images," *IEEE Journal on Selected Areas in Communications*, vol. 10, June 1992.
- [48] J. Modestino and Y. Kim, "Adaptive Entropy-Coded Predictive Vector Quantization of Images," *IEEE Transactions on Signal Processing*, vol. 40, Mar. 1992.

- [49] C. Hall and E. Hall, "A Nonlinear Model for the Spatial Characteristics of the Human Visual System," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-7, Mar. 1977.
- [50] M. Perkins and T. Lookabaugh, "A Psychophysically justified Bit Allocation Algorithm for Subband Image Coding Systems,"