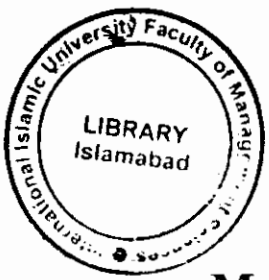
Acc. No. (TH) T-1172

Predictive Reflected Residual Vector Quantization



Developed by

Muhammad Shoaib
Ch. Muhammad Imran

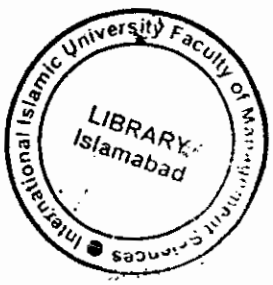
Supervised by

Dr. M.Sikandar H. Khiyal

Department of Computer Science
International Islamic University, Islamabad
(2005)

Acc. No. (PHC) T-1172

In the name of ALMIGHTY ALLAH,
The most beneficent, the most
Merciful.



Department of Computer Science
International Islamic University, Islamabad

Dated: 29/01/2005

Final Approval

It is certified that we have read the project report submitted by Ch. Muhammad Imran and Muhammad Shoaib and it is our judgment that this project is of sufficient MS standard to warrant its acceptance by International Islamic University, Islamabad for MS degree in Computer Science.

COMMITTEE

External Examiner

Dr. Asmatullah Khan
Head,
Electrical Engineering Department,
Comsats Institute of Information Technology, Islamabad.

Internal Examiner

Asim Munir
Faculty member,
Department of Computer Science,
International Islamic University, Islamabad.

Supervisor

Dr. M.Sikandar H. Khiyal
Head,
Department of computer science
Faculty of Applied Sciences,
International Islamic University, Islamabad.

A dissertation submitted to the

DEPARTMENT OF COMPUTER SCIENCE,

INTERNATIONAL ISLAMIC UNIVERSITY, ISLAMABAD

as a partial fulfillment of the requirements for the award of the degree of MS in Computer Science.

Dedication

Dedicated to those teachers and parents who not only make professionals but also human beings.

Declaration

We hereby declare that this software, neither as a whole nor as a part thereof has been copied out from any source. It is further declared that we have developed this software entirely on the basis of our personal efforts made under the sincere guidance of our teachers and supervisor.

No portion of the work presented in this report has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning.

Ch. Muhammad Imran
66-CS/MS/2002

Muhammad Shoaib
78-CS/MS/2002

Acknowledgements

All praise to the Allah Almighty, the most merciful, the most gracious, without His help and blessings, we were unable to complete the project.

Thanks to our project supervisor Dr. M.Sikandar H. Khiyal Whose sincere efforts helped us to complete our project successfully.

Without the great help, motivation and righteous guidance of Dr. Asmat it was nearly impossible to complete this project.

Thanks to our families who helped us during our most difficult times and it is due to their unexplainable care and love that we are at this position today.

Ch. Muhammad Imran
66-CS/MS/2002

Muhammad Shoaib
78-CS/MS/2002

Project in Brief

Project Title:	**Predictive Reflective Residual Vector Quantization.**
Objective:	**To build an efficient and robust Technique for image compression**
Undertaken By.	**Ch. Muhammad Imran** **66-CS/MS/2002** **Muhammad Shoaib** **78-CS/MS/2002**
Supervised By:	**Dr. M.Sikandar H. Khiyal** **Head, Department of Computer Science, International Islamic University, Islamabad**
Date Started:	**1st September, 2003**
Date Completed	**30th September, 2004**
Technologies Used:	**GCC on Linux and Matlab 6.5**
System Used:	**Pentium IV**

Abstract

Image communications is primarily constrained due to large bandwidth requirements. Therefore, researchers worked on various compression algorithms to achieve low bit rate. It was stated that images and video sequences are highly correlated sources and their correlation should be exploited in a given compression algorithm Differential pulse code modulation (DPCM) has emerged as a mean of exploiting the correlation among the image pixels. Later on, DPCM was improved upon by predictive vector quantization (PVQ). PVQ employs block-by-block prediction and results in satisfactory performance at low bit rates. However, its design is complicated and recently an asymptotic closed-loop (ACL) was proposed to stabilize the design. In this thesis, we attempted to replace the VQ with a multistage VQ structure in a hope to further reduce the stress on the closed-loop design. The multistage VQ structure that we employed is commonly referred to as reflected residual vector quantization (RRVQ). RRVQ works by imposing an additional symmetry constraint on the multistage codebook design. RRVQ has been quite popular where large block length vector quantization is needed due to their very low codebook search capability. Our proposed design goal in replacing VQ with RRVQ in a PVQ design is our wish to use large block length like 16 x 16 or 32 x 32 size vectors to grab any linear/non-linear correlation among the vector components. The way to incorporate RRVQ within PVQ structure has been proposed and simulation results are discussed.

TABLE OF CONTENTS

TABLE OF FIGURES

Chapter 1
Introduction

1. Introduction

This introductory chapter has threefold purpose. First, to place the work of this dissertation into its proper place within the general framework of complete communication system design and implementation. Second, to provide a brief review of the theory of source block coding, and to discuss the practical issues that utilization of source block code structures called residual quantization (RQ) and reflected residual quantization (RRQ) that examined and improved in this dissertation.

This chapter is comprised of four sections. Section 1.1 introduces a complete and very general communication system model and shows how the general design problem can be separated into smaller design tasks. Section 1.2 provides brief reviews of the theory that motivates the use of source block codes. Section 1.3 reviews several source code structure and discuss the influence of practical issues on the design and utilization of these structures. Finally, Section 1.4 introduces residual quantization and presents the goals and objectives of this dissertation.

1.1. A Communication System Model

Consider the block diagram of the communication system model shown in figure 1.1. The basic building blocks of this model are source–user pair, the encoder-decoder pair and the channel. This model is important for two reasons. First, the various elements are suitably idealized from their physical components to allow the model to be sufficiently general for most communication systems. Second, if the model components are statistically charcterizable, then the model proves amenable to productive analysis. We proceed to describe briefly each of the components of this communication system model.

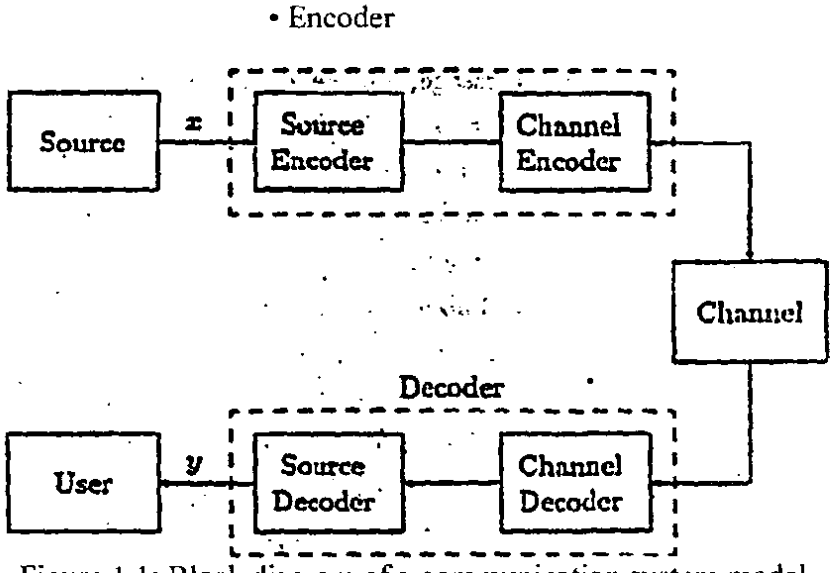


Figure 1.1: Block diagram of a communication system model.

The function of a communication system is to convey "useful" information from the source to the user. Often, the user does not require an exact reproduction of the message produced by the

source. For example, distortion, which does not degrade the intelligibility of speech, does not hinder a speech communication system's ability to convey the pertinent information. In general, a fidelity criterion may be associated with each source-user pair that measures the effect of any distortion of the received message relative to the actual transmitted message. Ideally, fidelity criteria should measure the effect of transmission errors on the users ability to use the received message relative to the useful-ness of the source's intended message. Unfortunately, such fidelity measures are unknown for many source-user pairs, or are very complex and difficult to use. Less descriptive, but more tractable fidelity measures may be denned by assigning functional values to the various errors that the communication system may make.

To give a mathematical description of an information source it is necessary to quantitatively specify the rate at which the source produces information. One pf the major contributions of Shannon's [1] work was the introduction of the entropy measure. Entropy may be interpreted as quantifying the average information content per source message, and hence, the average information rate of the source. Information sources may be separated into two major classes: discrete alphabet sources or continuous alphabet sources. A discrete alphabet source produces source symbols from a finite set of possible symbols. A continuous alphabet source produces source symbols from a continuum of possible symbols. The entropy of a discrete alphabet source is called absolute entropy and is different from the differential or relative entropy of a continuous alphabet source. To determine the entropy of either a discrete or continuous source, it is necessary to have a probabilistic description of the source output. The characterization of a source-user pair as a mathematical entity is complete if the joint probability distribution functions of the source outputs are known, and if a quantitative fidelity criterion is specified. These concepts are reviewed more thoroughly in Chapter 2.

Before proceeding to the encoder and decoder of Figure 1.1, we discuss briefly the channel. The channel is a physical medium that spatially or temporally links the source to the user. Channels can be separated into two major classes: noiseless channels and noisy channels. A noiseless channel is one in which each channel output results from a specific and known channel input. A noisy channel is one in which the signal is perturbed by noise during transmission, and the user is left uncertain as to what the channel input was.

For our purposes, the only parameter of a channel that interests us is its capacity. Shannon proved that the capacity of a channel is a number C that specifies the maximum rate at which information can be sent through the channel with arbitrarily low probability of error. Shannon

was able to define n definite capacity for both the noiseless and noisy channels. Since it is not possible to reconstruct the transmitted signal with certainty by any operation on the received signal, it may seem surprising that a capacity could be defined for a noisy channel. However, the channel capacity and the fundamental theorems of information theory guarantee the existence of methods which are optimal (in the sense of achieving arbitrarily low probability of error) in combating the channel noise. This is the job of the channel encoder and channel decoder of the block diagram in Figure 1.1. The channel encoder adds a certain amount of redundancy to combat the particular noise structure of the channel. Shannon [1] proved that if the information rate of the source H is less than or equal to the capacity C of the channel (noiseless or noisy), then an encoder-decoder pair can be found which transmits information at the rate $H \leq C$ with as small a frequency of error as desired. Conversely, Shannon proved that if H is greater than C, any attempt to transmit information at the rate H will lose or destroy information at a rate no less than the difference H-C. This last situation is of special interest to the work of this dissertation. We consider the case for which the source-user pair and the channel are assumed fixed, and the entropy of the source H exceeds either a given or desired channel capacity C. Although Shannon proved that information must unavoidably be lost under these circumstances, is it possible to control which information is lost? It is a generalization of the concepts of information theory. Which collectively is called rate-distortion theory, that answers this question in a rigorous yet relatively straightforward manner.

Rate-distortion theory associates with most source-users pairs a function D(R) called the distortion-rate function, which has the following significance. A communication system that is constrained to operate with a channel capacity of C and a source output rate of $H > C$ must somehow reduce the output, rate of the source to a value R that satisfies $R \leq C$. The distortion-rate function D(R) specifies the best possible fidelity that may be achieved when the rate of the information source is reduced to R. In other words, it is possible to control which information is lost, or more importantly, which information is communicated such that the resulting average distortion is as close as desired to the theoretically minimum value D(R).

Whenever the entropy, of the source exceeds the channel capacity, the encoder and decoder of Figure 1.1 may be separated into source/channel sub-blocks. The source encoder is given the important function of reducing the information rate of the source to a Value R not greater than the capacity of the channel. An "ideal" source encoder operates such that the resulting average distortion is minimized to a value arbitrarily close to the theoretically optimal value D(R). The

channel encoder then employs an "ideal" channel code to achieve, an arbitrarily low probability of error in the reproduction of the source encoder's output.

With source/channel encoder and decoder sub blocks, the communication system model has the advantage of a complete separation of the tasks of source coding and channel coding. Shannon [1] proved that performance arbitrarily close to the theoretical optimum could be achieved by designing each subsystem independently, which usually is the approach in practice. The source encoder employs a source code that is optimal with respect to the given source-user pair and fidelity criterion without any regard to the detailed nature of the channel. The channel encoder employs a channel code that is optimal with respect to the channel without any regard to the detailed nature of the source-user pair and fidelity criterion. In this dissertation we assume that we are given a source-user pair and fidelity criterion, and that we have the freedom to choose a source encoder and source decoder. The rest of the communication system model is assumed fixed and is lumped into some type of "ideal" channel that we assume to be error free in the sense that the output of the source encoder can be communicated with arbitrarily low probability of error. The general problem addressed in this work is the design of "good" source encoder-decoder pairs, and in particular, the design of "good" source coders we call residual quantizers. In the next two sections we briefly discuss what is meant by the word "good" in the present context. Section 1.2 discusses some of the theoretical issues that determine "good" source coders, and Section 1.3 discusses some of the practical issues that determine "good" source coders.

1.2. Vector Quantizers

In this dissertation we consider the communication system model of Figure 1.1 to be a discrete-time system. The source produces a sequence of source symbols $x(t_i)$ drawn from the source alphabet A_x, in discrete time. That is $x(t_i) = x_{ti} \in A_x$. The coding and transmission of the source output also occur in discrete time. The source is described by the joint probability density functions $p(x_t)=p(x_{t1}, x_{t2},\ldots, x_{tn})$ for all $n=\{1,2,\ldots\}$ and for all $t = (0,\pm1,\pm2\ldots\}$. We assume the source is stationary, that is $p(x_{t+k}) = p(x_t)$ holds for all integers k, all t, all $x_t \in R^n$, and all n. Thus, we ignore the t index and write the probability density function $p(x_t)$ as $p(x)$. Each n-dimensional vector x produced by the source is called a source vector.

We construct a particular type of source encoder called n vector quantizer (VQ) by choosing an indexed set $A= \{y_1, y_2,\ldots, y_N\}$, where A is called a code book of size JV and each of the selected $y_j \in R^n$ is called a code vector of dimensionality n. The vector quantizer

encoder operates by blocking the sequence of source output samples into contiguous source vectors of length n. The encoder maps each source vector x into whichever code vector $y_j \in A$ minimizes a given distortion function $d_n(x, y_j)$, and transmits the index of the selected code vector to the decoder. Since the decoder has a copy of the codebook, the decoder simply outputs the code vector corresponding to the received index associated with every possible codebook .A is an average distortion.

$$D(A) = \int \min_{y_j \in A} d_n(x, y_j) p(x)\, dx. \qquad (1.1)$$

The output rate of the vector quantizer is defined as $R = n^{-1} \lg N$ bits per sample (the entropy of the vector quantizer's output cannot exceed this value). Where the index of each selected code vector is encoded in a straightforward manner into binary number requiring $n^{-1} \lg N$ binary digits.

As described in Section 1.1, rate-distortion theory associates with the given source p(x) and the given distortion measure $d_n(x, y_j)$ with distortion-rate function D(R) which describes the lowest possible average distortion of any encoding scheme with an output rate of R. In particular, Shannon's Noisy Source Coding Theorem and its converse (to be described in more detail in Chapter 2) state that for a fixed rate of R bits per symbol, if the block length n is sufficiently large, then there exists a code book such that the average distortion of the encoded sequence is arbitrarily close to D(R). More precisely, there exists a code book A' such that $D(A') < D(R) + \epsilon$, where ϵ is an arbitrarily small positive number. The Noisy Source Coding Theorem and its converse have important practical consequences in communication theory. Other types of source code Structures may or may not be theoretically capable of achieving the distortion-rate bound *D(R)*. but these theorems guarantee that the use of block codes will be optimal if the block length is sufficiently large.

Even though rate-distortion theory guarantees the existence of vector quantizers with codebooks, which give nearly optimal performance, the theory provides no methods of determining "good" codebooks. It is interesting to note that although the Noisy Source Coding Theorem and its converse were first proved by Shannon [1] in the 1940s, it was not until about 1980 that vector quantizer code book design methods were widely developed and put into practice [2,4]. The delay was not a result of a lack of interest, but probably a result of the unavailability of the necessary computation and storage resources required to design and

implement block codes. In the next section we briefly review a few of the methods used to generate code books and discuss the important practical issues that determine the usefulness of a given code book structure in a vector quantizer.

1.3. Design and Instrumentation of Vector Quantizers

In 1980 a vector quantizer codebook design method known as the Generalized Lloyd Algorithm (GLA) or LBG method (named for the authors Linde, Buzo and Gray) was introduced [26]. Although we do not present the GLA algorithm in detail until Chapter 2, we find it useful to describe briefly the algorithm here. The GLA method uses a set of statistically representative samples of the source output, called a training set, to design the codebook. The code vectors produced by this method are obtained by clustering the training set into subsets of similar vectors and choosing the centroid of each cluster as a code vector. A characteristic of the GLA design method is that the vectors in each cluster are "closer" in the sense of the distortion measure $d_n(\cdot)$ to their centroid than to any other cluster's centroid.

A key issue in using codebooks designed with the GLA algorithm is the complexity of the vector quantizer implementation. The GLA code vectors have no natural order or structure, so every source vector requires an exhaustive search of the codebook to locate a code vector, which minimizes $d_n(\cdot)$. The implementation costs of these exhaustive search vector quantizers (ESVQs), measured in terms of computation and memory requirements, grow exponentially with the product of quantizer dimension and output rate. Specifically, for an ESVQ code book comprised of N code vectors, each of dimensionality n, the number of vector distortion computations needed to quantize a source vector is N. A vector distortion computation may be arbitrarily complex, but for a single-letter fidelity criterion a vector distortion computation requires n total of n scalar distortion calculations. If a scalar distortion calculation quantifies a measure of computation, then the computation cost (C) for quantizing each source sample is

$$C = \text{N scalar distortion calculations.} \tag{1.2}$$

A codebook with N *code* vectors requires (by definition) $\beta = \log_2 N$ bits to communicate the code vector selected for each source vector. Since one index is transmitted per source vector, the transmission rate is given by $R = \beta/n$ bits per source sample. Hence, the computation cost given by equation (1.2) can lie expressed as

$$C = 2^{\beta} = 2^{nR} \text{ scalar distortion calculations per sample.} \tag{1.3}$$

Assuming that one scalar memory location is used for each element of a code vector, the memory cost *(M)* required to store the codebook at either, the encoder or decoder is defined by the exponential relationship

$$M = n2^{\beta} = n2^{nR} \text{ scalar memory locations.} \tag{1.4}$$

Equations (1.3) and (1.4) express that at a fixed data rate (R constant) the computation and memory costs increase exponentially with increasing vector length *n*, or at a fixed block length (n constant) the costs increase exponentially with increasing rate *R*. Because rate-distortion theory requires a block length that is "sufficiently large" (for a fixed rate) before the existence of an optimal codebook is guaranteed, these incompatible theoretical and practical requirements limit the performance of practical ESVQs to values far from the optimum *D(R)*.

The exponential growth in computation and memory costs of ESVQ is a result of the lack of structure in the ESVQ codebook. Many researchers have suggested imposing different structures on the codebook in the hope that computationally economical searches can be used instead of exhaustive searches. The choice of a particular structure is necessarily ad hoc, in the sense that there is no motivation for any particular structure in rate-distortion theory. Examples of structured vector quantizers proposed in the literature include product code VQ [2], classified VQ [3], lattice VQ [4, 5], hierarchical VQ, and tree-structured VQ [6,7]. The different structural constraints lead to various compromises between quantizer design complexity, implementation complexity and performance. For example, lattice VQs have small computation and storage requirements, and perform quite well on uniformly distributed sources. Unfortunately, lattice VQs give poor performance for non-uniform data sources such as speech and imagery sources. Tree-structured vector quantizers perform well on these real data sources and have computation costs that grow only linearly with increasing vector length; however, the memory requirements are at least as great as those of ESVQs.

Most structured VQs are fairly successful at reducing the computation cost, but very little work has been done to reduce the memory requirements of VQ. One notable exception is a codebook structure that we introduce in the next section. This structure has been proposed to reduce *both* computation and memory expense.

1.4. Residual Quantizers

Imposing a multistage structure on the VQ has been suggested to hold down both the memory and computation costs [7, 8, 9, and 10]. This class of structured vector quantizers is known variously as multistage, multistep, Cascade, residual or successive approximation VQ. In this dissertation, we refer to these quantizers as residual quantizers (RQs). Instead of using one very large codebook, a residual quantizer uses a sequence of smaller codebooks. As shown in Figure 1.2, each stage of a residual quantizer encodes the residual error of the preceding stage. The final quantized value of the source vector is the sum of the code vectors selected at each of the stages.

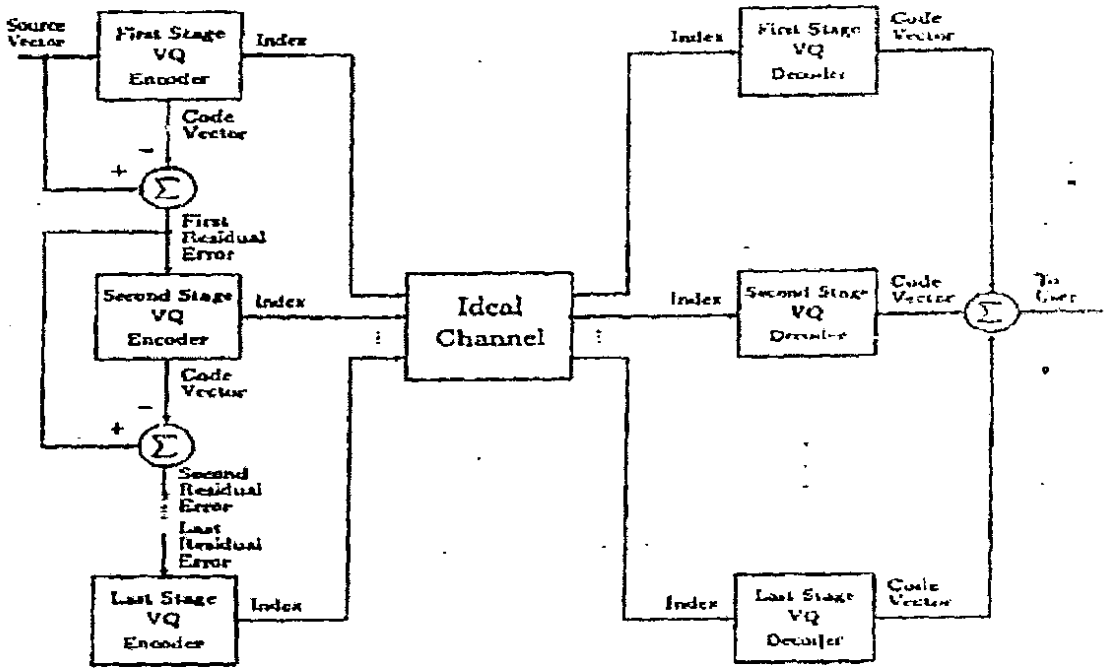


Fig 1.2. Residual Vector Quantizer

The computation and storage costs of a P-stage RQ with Np code vectors comprising the codebook of the pth stage are determined by the sums

$$C_{RQ} = \sum_{p=0}^{P-1} N^p \text{ scalar distortion calculations per sample.} \quad (1.5)$$

$$M_{RQ} = n\sum_{p=0}^{P-1} N^p \text{ scalar memory calculations.} \quad (1.6)$$

instead of the products

$$C_{ESVQ} = \prod_{p=0}^{P-1} N^p \text{ scalar distortion calculations per sample} \quad (1.7)$$

$$M_{ESVQ} = n\prod_{p=0}^{P-1} N^p \text{ scalar memory locations} \quad (1.8)$$

for an ESVQ that has the same number of code vectors as the number of unique code vector combinations determined by the product $N^0 \times N^1 \times N^2 x,\ldots,xN^{p-1}$.

Juang and Gray [8] were the first to propose the residual quantizer structure. They designed and tested two-stage RQs with each codebook size varying from 2 to 1024 code vectors. Using the GLA algorithm and a set of source training set vectors, they first designed the first stage codebook. Then using a residual training set formed from the source training set vectors and the first stage codebook, they used the GLA algorithm to design the second stage codebook. In their speech coding experiments, they reported a slight loss in performance relative to single stage ESVQs when using the two stages RQ.

Baker [27] proposed a modified RQ structure that was a generalization of Juang and Gray's [8] RQ to a product code structure. He seems to have done some preliminary investigations with RQ structures having more than two stages: however, he gave no results and made the discouraging statement "that it is not advantageous to iteratively vector quantize image waveform residuals.

Makhoul, Roucos, and Gish [9] conducted more extensive experiments with the RQ structure. Their results showed that the performance of RQs designed with the method of Juang and Gray [8] becomes relatively worse (compared to the performance of ESVQs) as the number of stages increases. They suggested using rotations of the residual vectors between stages to improve performance. The rotations improved the signal-to-quantization-noise ratio by about 1 db.

They conjectured that the reason RQs with more than two stages perform so poorly is because the training set residuals are pooled together before the GLA algorithm is applied to design the codebook of the next stage. They hypothesized that this pooling destroys the dependencies that exist in the initial training set clusters, and prevents the GLA algorithm from exploiting these dependencies when constructing the source code. Their final conclusion is that RQs should be limited to only two stages.

Makhoul, et al. [9], make the general observation that if some method of reducing the storage requirements of VQ without a concomitant major reduction in performance is not found, then storage cost is ultimately the major limitation in the practical use of VQ.

It is at this point that we ask ourselves a few questions about the RQ codebook structure:

1. What is the nature of the structure in RQ encoders and decoders, and can this structure be made explicit for possible analytical work to improve RQ performance?
2. Similar to ESVQ codebooks, are there certain conditions that are necessary for minimum distortion of RQ codebooks?

3. If such conditions exist, can design procedures be developed which yield RQ codebooks that satisfy these conditions?

The remainder of this dissertation essentially provides affirmative answers to each of these questions.

Gabor and Gyorfi [28] note that the design process involves two complementary steps: the choice of the structural constraints, and the design of the system given the structural constraints. The former is mainly a heuristic and experimental process where the designer should consider all the technological and financial limitations involved. The reduced computation and memory costs justify our interest in the RQ structure. However, once a structural constraint is accepted, it is the task of the designer to make the constraint explicit, and maximize performance under it. This is precisely the task accomplished in this dissertation. We make the structure of RQ codebooks explicit and optimize performance under the residual constraint.

This dissertation is outlined as follows. In Chapter 2, we provide background material necessary for understanding concepts used in subsequent chapters. Topics covered include a more detailed description of information theory, image compression and the theory of fixed-level quantization. In Chapter 3, we make explicit the structure of residual quantizers and derive necessary conditions for minimum mean squared error scalar RQs and minimum distortion vector RQs. In general, one of the derived necessary conditions requires a computationally expensive encoding strategy. We call residual quantizers that use this encoding strategy exhaustive search RQs. In Chapter 4, we give an algorithm to design of PEC-RRVQ.

In chapter 5, we will discus implementation of coder and decoder for PRRVQ. In Chapter 6, we will show certain results and comparisons of PEC-RRVQ with EC-RRVQ and comparison of EC-RRVQ with certain exhaustive search RVQ techniques. Finally, in Chapter 6 we summarize the results of the dissertation and suggest future research topics.

Chapter 2
Background

2. Background

Everyday, an enormous amount of information is stored, processed, and transmitted digitally. Companies provide business associates, investors, and potential customers with financial data, annual reports, inventory, and product information over the Internet. Order entry and tracking, two of the most basic on-line transactions, are routinely performed from the comfort of one's own home. Cable television programming on demand is on the verge of becoming a reality. Digital image and video compression is now essential. Internet teleconferencing, because much of this on-line information is graphical or pictorial in nature, the storage and communications requirements are immense. Methods of compressing the data prior to storage and transmission are of significant practical and commercial interest. High Definition Television (HDTV), satellite communications and digital storage of movies would not be feasible without a high degree of compression [10].

Image compression addresses the problem of reducing the amount of data required to represent a digital image. The-underlying basis of the reduction process is the removal of redundant data. From a mathematical viewpoint, this amount to transforming a 2-d pixel array into a statiscally uncorrelated data set. The transformation is applied prior to storage or transmission of image. At some later time, the compressed image is decompressed to reconstruct the original image or an approximation of it.

Interest in image compression dates more back than 35 years. The initial research efforts in this field were on the development of analog methods for reducing the video transmission bandwidth, a process called bandwidth compression. The advent of digital computer and subsequent development of advanced integrated circuits however caused interest to shift from analog to digital compression approaches. With relatively recent adoption of several key international image compression standards, the field has undergone significant growth through the practical application of the theoretic work that began in the 1940s, when C.E.Shannon [1] and other first formulated the probabilistic view of information and its representation, transmission, and compression.

Currently, image compression is organized as an "enabling technology". In addition to the areas just mentioned, image compression is the natural technology for handling the increased spatial resolutions of today's imaging sensors and evolving broadcast television standards. Furthermore, image compression plays a major role in many important and diverse applications, including televideo conferencing, remote sensing (the use of satellite imagery for weather and other earth-resource applications), document and medical imaging, facsimile transmission (FAX), and the control of remotely piloted vehicles in military, space and hazardous waste management applications, in short, an ever-expanding number of applications depend on the efficient manipulation, storage, and transmission of binary, gray-scale, and color images.

In this chapter, we examine both the theoretic and practical aspects of the image compression process. Section 2.1 through 2.3 constitutes an introduction to the fundamentals that collectively from the theory of this discipline. Section discusses the basics of image compression. Section 2.2 describes the data redundancies that may be exploited by image compression algorithms.

Section 2.4 through 2.5 covers the practical aspects of image compression, including both the principal techniques in use and standards that have been instrumental in increasing the scope

and acceptance of this discipline. Compression techniques fall into two broad categories: information preservation and lossy. Section 2.4 addresses these methods. Which are particularly useful in image archiving (as a storage of legal or medical records). These methods allow an image to be compressed and decompressed with or without losing information. Section 2.5 describes method in the second category, which provides higher level of data reduction but result in a less than perfect reproduction of the original image. Lossy image compression is useful in applications such as broadcast television, videoconferencing, and facsimile transmission, in which a certain amount of error is an acceptable trade-off for increased compression performance. Section 2.5 deals with existing and proposed lossy image compression standards known as Vector Quantization.

2.1.FUNDAMENTALS:

The term data compression refers to the process of reducing the amount of data required to represent a given quantity of information. A clear distinction must be made between data and information. They are not synonymous. Infect, data are the means by which information is conveyed. Such amount of data may be used to represent the same amount of information. Such might be the case, for example, if a long-winded individual and someone who is short and to the point were to relate the same story. Here, the information of interest is the story; words are the data used to relate the information. If the two the individuals use a different number of words to tell the same basic story, two different versions of the story are created, and at least one includes nonessential data. That is, it contains data (or words) that either provide no relevant information or simply restate that which is already known. It is thus said to contain data redundancy.

Data redundancy is a central issue in digital image compression. It is not an abstract concept but a mathematically quantifiable entity. If n1 and n2 denote the number of information-carrying units in two data sets that represent the same information, the relative redundancy Rd of the first data set (the one characterized by n1) can be defined as

$$RD = 1 - 1\backslash CR \qquad (2.1)$$

Where CR, commonly called as compression ratio, is

$$CR = n1\backslash n2 \qquad (2.2)$$

For the case n2 = n1, CR=1 and RD= 0, including that (relative to the second data set) the first representation of the information contains no redundant data. When $n2 \ll n1$, $CR \rightarrow \infty$ and $RD \rightarrow 1$, implying significant compression contains no redundant data. Finally, When the case $n2 \gg n1$, $CR \rightarrow 0$ and $RD \rightarrow -\infty$, indicating that the second data set contains much more data than the original representation. This is of course, undesirable case of data expansion. In general, CR and RD lie in the open intervals $(0, \infty)$ and $(-\infty, 1)$, respectively. A practical compressior. ratio, such as 10 (or 10:1), means that the second or compressed data set. The corresponding redundancy of 0.9 implies that 90% of the data in the first data set is redundant.

2.2. Redundancy

Redundancy is the repetition of same data with in the image. In digital image compression, three basic redundancies can be identified are exploited; coding redundancy, interpixel

redundancy, and psychovisual redundancy. Data compression is achieved when one or more of these redundancies are reduced or eliminated [11].

2.2.1. Coding Redundancy

We developed the technique for image enhancement by histogram processing on the assumption that the gray levels of an image are random quantities. We showed that great deal of information about the appearance on an image could be obtained from histogram of these gray levels. In this section, we utilize a similar formulation to show how the gray-level histogram of an image also can provide a great deal of insight into the construction of codes to reduce the amount of data used to represent it.
Let us assume, once again in, that a discrete random variable rk in the interval [0,1] represent the gray levels on an image and that each rk occurs with probability pr(Rk).

$$Pr(Rk) = nk/n \quad k=0, 1, 2,\ldots, L-1 \tag{2.3}$$

Where L is the number of gray levels, nk is the number of times that the kth gray level appears in the image, and n is the total number of pixels in the image. If the number of bits used to represent each value of rk is l(Rk), then the average number of bits required to represent each pixel is

$$L\,avg = \sum l(Rk)pr(Rk). \tag{2.4}$$

That is the average length of the code words assigned to the various gray-level values is found by summing the product of the number of bits used to represent each gray level and the probability that the gray level occurs. Thus the total number of bit required to code an MXN image is MNL avg.
Representing the gray levels of an image with a natural m-bit binary code reduces the right-hand side of Equation (2.4) to m bits, That is , Lavg = m when m is substituted for l(Rk). Then the constant m may be taken outside the summation, leaving only the sum of the pr(Rk) for 0<=k<=L-1, which, of course equals 1.
Example: (A simple illustration of variable-length coding)
An 8-level image has the gray-level distribution shown in Table 2.1. If a natural 3-bit binary code [see code 1 and $l_1(Rk)$= 3 bits for all Rk. If Code 2 in Table 2.1 is used, however, the average number of bits required to code the image is reduced to 2.7 bits.

Table 2.1 Example of variable-length coding.

Rk	Pr(Rk)	Code1	$l_1(Rk)$	Code2	L2(Rk)
R0=0	0.19	000	3	11	2
R1=1/7	0.25	001	3	01	2
R2=2/7	0.21	010	3	10	2
R3=3/7	0.16	011	3	001	3
R4=4/7	0.08	100	3	0001	4
R5=5/7	0.06	101	3	00001	5
R6=6/7	0.03	110	3	000001	6
R7=1	0.02	111	3	000000	6

$$L_{avg} = \sum_{K=0}^{7} l_2(Rk)\, p_r(Rk)$$

$=2(0.19) + 2\,(0.25) + 2\,(0.21) + 3\,(0.16) + 4\,(0.08) + 5\,(0.06) + 6\,(0.03) + 6\,(0.02)$

$=2.7$ bits.

From Equation (2.2), the resulting compression ratio CR is 3/2.7 or 1.11. Thus approximately 10% of the data resulting from the use of code is redundant. The exact level of redundancy can be determined from Equation (2.1).

$R_D=1-1/1.11=0.099$.

In the preceding example, assigning fewer bits to the more probable gray levels than to the less probable ones achieves data compression. This process commonly is referred to as variable-length coding. If the gray levels of an image represent each gray level [that is, the code fails to minimize Equation (2.4)], the resulting image is said to contain coding redundancy. In general, coding redundancy is present when the codes assigned to a set of events (such as gray-level values) have not been selected to take full advantage of the probabilities of the events. It is almost always present when an image's gray levels are represented with a straight or natural binary code. In this case, the underlying basis for the coding redundancy is that images are typically composed of objects that have a regular and somewhat predictable morphology (shape) and reflectance, and are generally sampled so that the objects being depicted are much larger than gray levels are more probable than others (that is, the histograms of most images are not uniform). A natural binary coding of their gray levels assigns the same number of bits to both the most and least probable values, thus failing to minimize Equation. (2.4) and resulting in coding redundancy.

2.2.2. Inter-pixel Redundancy

If the gray levels in images are not equally probable variable-length coding can be used to reduce the coding redundancy that would result from a straight or natural binary encoding of their pixels. The coding process, however, would not alter the level of correlation between the pixels within the images. In other words, the codes used to represent the gray levels of each image have nothing to do with the correlation between pixels. These correlations result from the structural or geometric relationships between the objects in the image.

Autocorrelation coefficients computed along one line of each image.

$$\gamma(\Delta n) = \frac{A(\Delta n)}{A(0)} \tag{2.5}$$

Where

$$A(\Delta n) = \frac{1}{N-\Delta n} \tag{2.6}$$

The scaling factor in Equation (2.6) accounts for the varying number of sum terms that arise for each integer value of Δn. Of course, Δn must be strictly less than N, the number of pixels on a line. The variable x is the coordinate of the line used in the computation.

These illustrations reflect another important form of data redundancy, one directly related to the inter-pixel correlations within an image. Because the value of any given pixel can be reasonably, predicted from the value of its neighbors. The information carried by individual pixels is relatively small. Much of the visual contribution of a single pixel to an image is redundant; it could have been guessed on the basis of the values of its neighbors. A variety of names, including spatial redundancy, geometric redundancy, and inter-frame redundancy, have been coined to refer to these inter-pixel dependencies. We use the term inter-pixel redundancy to encompass them all.
In order to reduce the inter-pixel redundancies in an image, the 2-D pixel array normally used for human viewing and interpretation must be transformed into more efficient (but usually "non visual") format. For example, the differences between adjacent pixels can be used to represent an image. Transformations of this type (That is, those that remove inter-pixel redundancy) are referred to as mappings. They are called reversible mappings in the original image elements can be reconstructed from the transformed data set.

2.2.3.Psychovisual Redundancy

We noted in Section 2.1 that the brightness of a region, as perceived by the eye, depends on factors other than simply the light reflected by the region. For example, intensity variations (Mach bands) can be perceived in an area of constant intensity. Such phenomena result from the fact that eye does not respond with equal sensitivity to all visual information. Certain information simply has less relative importance than other information in normal visual processing. This information is said to be psychovisually redundant. It can be eliminated without significantly impairing the quality of image perception.
That psychovisual redundancies exist should not come as surprise, because human perception of the information in an in image normally does not involve quantitative analysis of every pixel value in the image. In general, an observer searches for distinguishing features such as edges or textural regions and mentally combines them into recognizable groupings. The brain then correlates these groupings with prior knowledge in order to complete the image interpretation process.
Psychovisual redundancy is fundamentally different from the redundancies discussed earlier. Unlike coding and inter-pixel redundancy, psychovisual redundancy is associated with real or quantifiable visual information. Its elimination is possible only because the information itself is not essential for normal visual processing. Since the elimination of psychovisually redundant data results in a loss of quantitative information, it is commonly referred to as quantization. This terminology is consistent with normal usage of the word, which generally means the mapping of a broad range of input valued to a limited number of output values, as discussed in Section 2.4. As it is an irreversible operation (visual information is lost). Quantization results in lossy data compression.

2.3. Fidelity Criteria

As noted previously, removal of psycho visually redundant data results in a loss of real or quantitative visual information. Because information of interest may be lost, a repeatable and reproducible means of quantifying the nature and extent of information loss is highly desirable. Two general classes of criteria are used as for such an assessment: (1) objective fidelity criteria and (2) subjective fidelity criteria.

When the level of information loss can be expressed as a function of the original or input image and the compressed and subsequently decompressed output image, it is said to be based on an objective fidelity criterion. A good example is the root-mean-square (rms) error between an input and output image. Let $f(x, y)$ denote an estimate and approximation of f (x, y) that results from compressing and subsequently decompressing the input. For any value of x and y, the error e (x, y) between f (x, y) and $f(x, y)$ can be defined as

$$e(x, y) = f(x, y) - f(x,y) \tag{2.7}$$

So that the total error between the two images is

$$\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\, f(x,y) - f(x,y) \,] \tag{2.8}$$

Where the images are of size M x N. The root-mean-square error, rms, between f(x, y) and $f(x,y)$ then is the square root of the squared error averaged over the M x N array, or

$$e_{rms} = \left[1/MN * \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\, f(x,y) - f(x,y) \,]^2 \right]^{1/2} \tag{2.9}$$

A closely related objective fidelity criterion is the mean-square signal-to-noise ratio of the compressed-decompressed image. If $f(x,y)$is considered [by a simple rearrangement of terms in Equation 2.7] to be the sum of original image f(x, y) and a noise signal e(x, y), the mean-square signal-to-noise ratio of the output image, denoted SNR_{rms}, is

$$SNR_{rms} = \frac{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)^2}{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\, f(x,y) - f(x,y) \,]^2} \tag{2.10}$$

The rms value of the signal-to-noise ratio, denoted SNR_{rms}, is obtained by taking the square root of Equation (2.9).
The objective performance measure used in all experiments is the peak-signal-to-quantization noise ratio (PSNR). PSNR is defined as

$$PSNR = -10 \log_{10} \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} (x(i,j) - x(i,j))^2}{(N^2)(255)^2} \tag{2.11}$$

Where N x N is the size of the image and x(i,j) and $\hat{x}$(i,j) represent the original and coded values, respectively, at the ith row and the jth column.

2.4.The Two Kingdoms of Techniques.

Data-compression techniques can be divided into two major families; lossy and loss less. Lossy data compression concedes a certain loss of accuracy in exchange for greatly increased compression. Lossy compression proves effective when applied to graphics images and digitized voice. By their very nature, these digitized representations of analog phenomena are not perfect to begin with, so the idea of output and input not matching exactly is a little more acceptable. Most lossy compression techniques can be adjusted to different quality levels, gaining higher accuracy in exchange for less effective compression. Until recently, lossy compression has been primarily implemented using dedicated hardware. In the past few years, powerful lossy-compression programs have been moved to desktop CPUs, but even so the field is still dominated by hardware implementations.

Loss less compression consists of those techniques guaranteed to generate an exact duplicate of the input data stream after a compress/expand cycle. This is the type of compression used when storing database records, spreadsheets, or word processing files. In these applications, the loss of even a single bit could be catastrophic.

2.4.1.Noiseless Source Coding

If the original signal is digital and can be perfectly reconstructed from the coded signal or data, then the coding scheme is called noiseless coding or lossless coding or Entropy coding. Noiseless coding is often required in some systems, for example in coding binary computer programs for storage or transmission: A single bit in error has disastrous consequences. If the noiseless coding results in a digital sequence with a smaller communications rate or storage rate then than the original signal, then the noiseless coding is called noised less data compression. Noiseless data compression is also referred as data compaction. An example of noiseless data compression of the ASCII code is the Morse code representation (use short vectors for more common or likely letters and less characters for less common or less likely letters).

In many cases the rate of a code used to encode discrete amplitude source exceeds the source's entropy. For example, the code used at the output of a scalar quantizer, or analog-to-digital converter, typically employs a fixed input of bits to represent the value of each output sample. If the quantizer's output values do not occur with equal regularity, then this fixed-length code is inefficient. We may seek a code with an average rate below that of the fixed-length code. Encoding schemes that attempt to give the lowest possible average rate (for a discrete amplitude source) without introducing any distortion .is called distortion less encoding schemes. Practical systems that realize noiseless coding are called entropy coders. Entropy coding involves variable-rate and variable length mappings of codewords. The instantaneous rate of an entropy encoder varies about its average entropy and one must be concerned with buffer overflow and underflow problems in the implementation.

$$\lim_{n \to \infty} n^{-1} \log N(n, D) = R(D) \qquad (2.15)$$

Equation (2.15) expresses that for each D > 0, the rate-distortion function R (D) is the minimal exponential rate at which the size of a code book must be increased with increasing block length n in order for the code to remain D-admissible.

2.5. Quantization

This section is the last of the preliminary background and review sections. The work of this dissertation is strongly influenced by the ideas presented here. In fact, much of this work may be considered analogous to the scalar work of Lloyd and Max, and the vector work of Linde, Buzo and gray; the major difference being our acceptance of a particular structural constraint which we call a residual structure on the quantizer's encoder and decoder.
The purpose of this section is to discuss in more detail some of the particulars of quantization. Included in this discussion is a review of conditions necessary for the optimality of fixed-level quantizers: both fixed-level scalar quantizers and fixed-level vector quantizers. Fixed-level quantizers have a fixed number of representative output values; other classes of quantizers (for example, entropy-constrained quantizer) place no constraint on the number of levels and may even have an infinite number of possible output values. Fixed-level scalar quantizers are of interest because the derivation of necessary conditions for minimum mean squared error of scalar quantizers preceded and motivated the establishment of necessary conditions for minimum distortion vector quantizers.

2.5.1. Lloyd-Max Scalar Quantizers

An N-level scalar quantizer of the real-valued random variable V consists of a finite indexed set or real numbers $A = (y_1, y_2, ..., y_N\}$, and a corresponding set $\{X_0, X_1, ..., X_N\}$. The $y_j \in A$ are called the quanta or reconstructed levels, and the xj are called the partition boundaries or decision levels. The Xj specify a set of partition intervals where the quantizer output is yj if x falls within the Interval Sj: $\{x_{j-1} < x <= x_j\}$. The collection of partition intervals forms the partition $P = \{S_1, S_2, ..., S_N\}$. It is usually implicitly assumed that $yj \in Sj$. A scalar quantizer is specified by a mapping y = Q(x), called the quantizer characteristic.
The quantizer represented in Figure 2.1 has decision levels and reconstruction levels no uniformly spaced, and is called non-uniform. A quantizer with equidistant decision levels and reconstruction levels is called uniform, as shown by an example in Figure 2.2. Although uniform quantization is most commonly used in practice, it does not necessarily represent the most effective conversion.

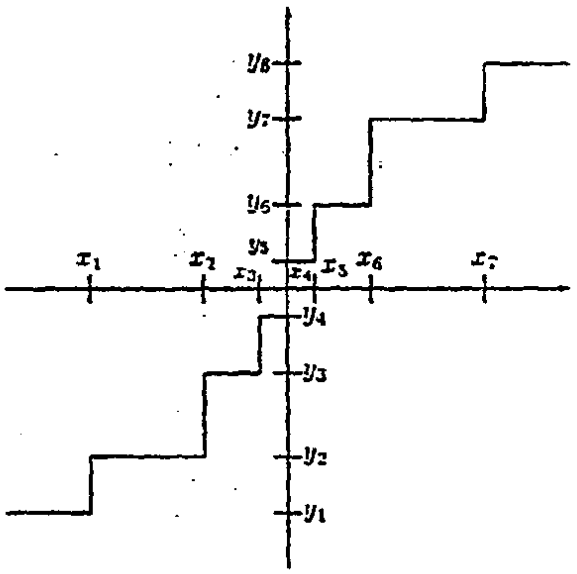


Figure 2.1: Nonuniform scalar quantizer.

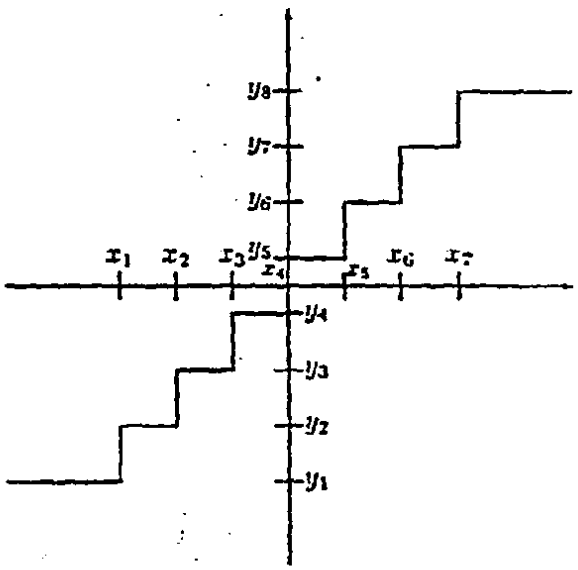


Figure 2.2: Uniform scalar quantizer.

A nonuniform quantizer, which uses smaller partition intervals where the probability of Occurrence of x is relatively higher, and larger intervals otherwise, should yield a smaller average distortion.

Necessary Conditions for Minimum Mean Squared Error

To determine which X_j and Y_j minimize the average distortion of a nonuniform scalar quantizer for a given source, we express the mean squared error as

$$D_{MSE} = \sum \int_{x_{j-1}}^{x_j} (x-y_j)^2 f(x)d(x) \qquad (2.16)$$

Where f(x) is the probability density function of A'. Necessary conditions for the optimality of a nonuniform scalar quantizer with a finite number of quantization levels follow from the minimizations

$$\frac{\partial_{DMSE}}{\partial x_j}=0;\ j=:\ 1,2...N-1 \qquad \frac{\partial_{DMSE}}{\partial y_j}=0;j=0.1...N. \tag{2.17}$$

and were first derived in an unpublished paper by Lloyd and later in a published paper by Max. For an N-Level scalar quantizer to give minimum mean squared error, implies that the quantizer partition boundaries necessarily satisfy

$$x_j = \frac{y_j + y_{j+1}}{2} \quad \text{for } 1<=j<=N-1, \tag{2.18}$$

Where $x_0 = -\infty \qquad x_N = \infty$

and the quantizer quanta necessarily satisfy

$$y_j = \frac{\int_{x_{j-1}}^{x_j} x\, f(x)d(x)}{\int_{x_{j-1}}^{x_j} f(x)d(x)} \qquad \text{for } 1<=j<=N. \tag{2.19}$$

Above Conditions are also sufficient if f(x) is log-concave. Equation (2.18) expresses that the optimal boundary points of the quantizer partition intervals lie halfway between quanta values. Equation (2.19) expresses that the optimal quanta are the centroid or conditional means of their partition intervals. Quantizers satisfying both equations are called Lloyd-Max quantizers. Max also derived necessary conditions for the optimality of uniform quantizers. We next review; design algorithm which was developed to yield optimal Lloyd-Max nonuniform quantizers.

Lloyd's "Method I" Design Algorithm

Lloyd gave two different methods, which he called "Method I", and "Method II" to design nonuniform scalar quantizers with quanta and partition boundaries that minimize mean squared error. Since Method I has been generalized to yield a vector quantizer design method, we briefly describe this algorithm.
Starting with either an initial guess of the partitions or an initial guess of the quanta, Method I iterates by first satisfying the condition not initially guessed and then the other, repeating the iteration until eventually both conditions are satisfied simultaneously.
Lloyd's methods have been used to design optimal scalar quantizers for sources characterized by one of a variety of probability density functions.

2.5.2 Exhaustive Search Vector Quantizers.

Before reviewing conditions necessary for the optimality of vector quantizers and a vector quantizer design algorithm, it is helpful to investigate the structure of exhaustive search vector quantizers.

Let x be a random vector in n-dimensional Euclidian space R^n described by a distribution function F on R^n. An N-level vector quantizer of R^n consists of the following: 1). A finite indexed subset $A= \{y_1, y_2,..., y_N\}$ of R^n called a code book, where each $y_j \in A$ is called a code vector, 2) a partition $P =\{S_1,S_2,...,S_N\}$ of R^n where the equivalence classes or cells S_j of P satisfy

$$\bigcup_{j=1}^{N} S_j = R^n, \quad (2.20)$$

$$S_j \cap S_k = \varnothing \text{ for } j \neq k;$$

And 3) a mapping Q: $R^n \rightarrow A$ that defines the relationship between the codebook and partition as

$$Q(x) = y_j \text{ if and only if } x \in S_j \quad (2.21)$$

for $1 \leq j \leq N$, where $y_j \in A$ and $Sj \in P$. In practice, the quantizer mapping Q is realized as a composition of two separate functions: the encoder mapping ξ and the decoder mapping D. The encoder mapping $\xi : R^n \rightarrow J$ is defined as

$$\xi(x) = j \text{ if and only if } x \in Sj, \quad (2.22)$$

where $x \in R^n$ and j is a member of the *index set* $J = \{1.2,..., N)$. Each $j \in J$ is called a *channel* code word and is either stored in some medium or transmitted through the channel. The decoder mapping $D : J \rightarrow A$ is defined as

$$D(j) = y_j \quad (2.23)$$

Where $j \in J$ and $y_j \in A$. The encoder and decoder mappings define the quantizer mapping as $Q(x) = D(\xi(x))$. Specification of the triple $(A.Q.P)$ determines a vector quantizer. In the next section, we present a condition necessary for the optimality of the codebook A, and a condition necessary for the optimality of the partition P.

Vector Generalizations of the Lloyd-Max Conditions

For a fixed quantizer dimensionality n and fixed codebook size N the vector quantizer design problem is to determine the triple {A.Q.P} that minimizes the expected value of the distortion

$$D(x,y) = E\{d(x,y)\} = \sum E\{d(x, y) \mid x \in S_j\} Pr(x \in S_j). \quad (2.24)$$

For any given code book A, the partition that minimizes the average distortion satisfies

$$d(x, y_j) < d(x, y_k) \text{ for all } k \quad (2.25)$$

Where $x \in Sj$. Any partition that satisfies above Equation is called a Voronoi partition of R^n. Since any input vector may have more than one nearest neighbor (where nearest-neighbor means a code vector closest to x in the sense of d), a Voronoi partition is not in general unique. In the case of a tie between two or more code vectors, the input vector may be assigned to any of the corresponding cells with some arbitrary tie-breaking rule. An arbitrary codebook A and its associated Voronoi partition is denoted by the subscript V in the triple (A, Q, Pv). The quantizer (A, Q, Pv) is called a Voronoi quantizer.

For any fixed partition *P*, the codebook that minimizes the average distortion satisfies

$$E\{d(x, y_j) \mid x \in S_j\} = \min E\{d(x, u) \mid x \in S_j\} \quad (2.26)$$

for $1 \leq j \leq N$. Analogous to the points that satisfy , any yj that satisfies above equation is called a centroid of Sj. The points yj that satisfy above equation improved to exist if Pr ($x \in S_j$ Also, if d(x,y) is strictly convex in y then the centroid y, is unique otherwise, the centroid of a cell is not in general unique and some arbitrary rule for resolving ties may be necessary. A tie-breaking rule together with above equation defines a codebook of centroids for any given partition. An arbitrary partition P and its associated code book of centroids is indicated by the subscript C in the triple (Ac, Q, P) If the quantizer simultaneously satisfies both the Voronoi partition condition and the centroid code book condition, then the resulting quantizer (Ac, Q, Pv) satisfies a fixed-point condition .

The Generalized Lloyd Algorithm

In 1980, Linde, Buzo, and Gray generalized Lloyd's Method I to develop a vector quantizer design algorithm, known variously as the LBG or Generalized Lloyd Algorithm (GLA). Although very similar, there are two significant differences between Method I and the GLA algorithm. First, Method I requires an analytically satisfied probability density function to describe the source; but for many information sources encountered in practice, the multidimensional source probability density function is either unknown or is not easily specified analytically. The GLA algorithm circumvents this difficulty by substituting a training set for the probability density function. This substitution is proven in the limit of large training set size to produce asymptotically equivalent designs. Second, Method I require an explicit description of the quantizer partition. Unfortunately, the specification of the boundaries of an arbitrary partition of R^n for $n > 1$ can be extremely complex. The GLA algorithm avoids this complication by exploiting the observation that any code book A and the nearest-neighbor rule partitions a training set the same as a Voronoi partition Pv of R^n that is associated with the triple (A, Q, Pv). [12]

The GLA algorithm can be described as follows. Let $T = \{x_1, x_2, \ldots, x_L\}$ be a training set of L sample vectors, where each sample vector x_i is drawn according to the probability distribution function F on R^n. The GLA algorithm improves (in the sense of reducing the *average* distortion) a Voronoi quantizer for the training set T. The GLA algorithm starts with some initial codebook A and then iterates by first replacing the code book with the centroids of the training set vectors that are in each of the cells of Pv that is associated with A. The algorithm then determines the new Voronoi partition by a nearest-neighbor mapping of the training set to the new codebook of centroids, and the entire process is repeated. Each iteration of the algorithm either reduces the average distortion or leaves it unchanged. If the distortion is unchanged by iteration, both the centroid codebook condition and nearest-neighbor partition condition are simultaneously satisfied, and the resulting quantizer (A, Q, Pv) is fixed-point with respect to the training set T.

Chapter 3
Residual Vector Quantizers

3. Residual Vector Quantization

Vector Quantization (VQ) is a powerful technique for data compression of speech, image, and video signals. Source coding theorems, as well as other well-known results in rate distortion theory, imply that one can always do better in the sense of achieving better rate-distortion performance if one can code vectors of samples as units, instead of separately as individual scalars. However, an issue of recognized importance for unconstrained VQ is that the size of the codebook grows exponentially as a function of the product of the vector and bit rate. Even for simple distortion measures, the encoding complexity can quickly become unmanageable. Similarly, the memory required storing the codebook at both the encoder and decoder side also grows exponentially. To overcome the complexity barrier, many researchers have suggested imposing certain structural constraints on the VQ codebook design. Residual VQ (RVQ) is one of the simple and efficient types of structurally constrained VQ designs. An RVQ consists of Multiple VQ stages, each operating on the residual of the previous stage.

A simple two-stage RVQ was proposed by Juang and Gray in 1982 for coding of speech signals. This RVQ encoder utilizes a computationally inexpensive, but sub optimal sequential single-path search. The performance of this computationally cheap RVQ was found to degrade significantly as the number of stages grows beyond two. In 1989, Barnes and Frost introduced a jointly optimized RVQ (JORVQ) design. In their design, an attempt was made to minimize the overall quantization error of the RVQ in lieu of merely optimizing the individual stages in isolation. They demonstrated that a sequential single-path search through VQ stages could not, in general, utilize all the available codevectors. They employed M-search, an efficient multi-path tree search algorithm, to search the stage codebooks. The design resulted in improvement. However, the increase in performance comes at the expense of additional computations. It was later shown in that the rate-distortion performance of a JORVQ can be further improved by including entropy encoding. The method was referred to as Entropy-constrained JORVQ (EC-JORVQ). Experimental results, reported in, show that EC-JORVQ outperforms single-stage entropy-constrained VQ (ECVQ) [13].

Although multi-stage JORVQ and EC-JORVQ do provide improved rate-distortion performance over the initial Juang and Gray sequential single-path design, the fact remains that these are computationally expensive designs. In order to keep the search complexity manageable while having a jointly optimized RVQ design, Barnes suggested an alternative in the form of a binary JORVQ with the advantage of using single-path search. The encoder and decoder of this binary JORVQ perform a reflecting or folding operation on the residual vectors between VQ stages. The folding operation forces certain symmetry on the JORVQ codebook. The symmetry of this JORVQ structure makes the sequential single-path search of stage codebooks optimal. This new binary JORVQ is referred to as Reflected RVQ (RRVQ). The experimental results, reported in [3], have shown that the imposition of reflection constraint led to an unavoidable increase in distortion as compared to multi-path JORVQ. However, since structured systems are inherently less random or more ordered, we expect that imposition of structure will also reduce the output entropy.

The RVQ emerges as a practical scheme for implementing VQ for large vector sizes. In particular, it was shown in [3] that entropy-constrained (EC-RVQ), a.k.a. entropy-

constrained jointly-optimized RVQ (EC-JORVQ), can outperform JPEG both in rate-distortion performance and decoding complexity, but generally requires large encoding complexity. Later on, EC-RVQ utilizing conditional entropy-constrained, and subband coding frame-works was also introduced with better gains [4]. An option for lowering RVQ encoding complexity cost is the imposition of additional structure on the RVQ stage codebooks to make the code more submissive to sequential single path searches. Multiple-stage VQ's with stage codebooks comprised of lattice VQ's and reflected RVQ (Ref-RVQ) and CEC-RVQ [25] are examples of this option.
In this Thesis, we design and analyze the performance of predictive RRVQ under entropy constraint (EC-RRVQ). Specifically, we try to answer the question: to what degree would entropy coding or optimization under an entropy constraint improve RRVQ rate-distortion performance. Our technical discussion begins with an overview of the RVQ design and provides definitions. Then, EC-RRVQ theory is fully explained and the algorithm de-sign is presented. In this chapter we formulate an explicit description of the structure of residual quantizers, and give a problem statement for the design of minimum distortion residual quantizers. We develop a structure called an equivalent quantizer that is useful in the derivation of necessary conditions for the optimality of residual quantizers.

3.1 Residual Quantizer Structures

Let x^0 be a random vector described by a probability distribution function F on R^n. A P-stage residual quantizer consists of a finite sequence of P quantizers $\{(A^p, Q^p, P^p); 0 \le p \le P\text{-}1\}$, ordered such that (A^0, Q^0, P^0) quantizes the source vector x^0 and (A^p, Q^p, P^p) quantizes the residual vector x^p of $(A^{p-1}, Q^{p-1}, P^{p-1})$ for $1 \le p \le P - 1$. The code vectors comprising the code book Ap and the cells comprising the partition P^0 are indexed with the subscripts j^p. Where j^p is a member of the p^{th} *index set* $J^p = \{1,2, ..., N^p\}$. The number of code vectors comprising A^p is indicated by N^p, which we consider a fixed preassigned number. We sometimes find it necessary to index the code vectors and partition cells of the p^{th} stage by the superscript p; that is, $A^p = \{y^p_1, y^p_2, ..., y^p_{Np}\}$ and $P^p = \{S^p_1, S^p_2, ..., S^p_{Np}\}$.
The mapping Q° applied to the input x° yields an output random vector $Q^0(x^0)$. The difference of x^0 and $Q^\circ(x^0)$ produces the residual random vector $x^1 = x^0 - Q^0(x^0)$. In general, the mapping Q^p applied to the input x^p yields an output $Q^p(x^p)$ and the residual $x^{p+1} = x^p - Q^p(x^p)$ for $0 \ p <. \ P$ -1. The random vectors $\{Q^p: 0 \le p \le P\}$ and the quantizer mappings $\{Q^p: 0 \le p \le P\text{-}1\}$ are related by the expression

$$X^0 = \sum_{p=0}^{P-1} Q^p(x^p) + x^p \qquad (3.1)$$

where x^p is the residual error of the last quantizer stage and is called the total residual error.
The sequence of quantizer triples $\{(A^p, Q^p, P^p); 0 \le p \le P\text{-}1\}$ can be separated into a sequence of quantizer mappings $\{Q^0, Q^1, ..., Q^{p-1}\}$, a sequence of codebooks $\{A^0, A^1, ..., A^{p-1}\}$, and a sequence of partitions $\{P^0, P^1, ..., P^{p-1}\}$. Each map Q^p in the sequence of quantizer mappings $\{Q^0, Q^1, ..., Q^{p-1}\}$, is realized by a composition of an encoder

mapping E^p and a decoder mapping E^p. The p^{th} encoder mapping $\varepsilon^p: R^n \rightarrow J^p$ is defined by

$$\varepsilon^p(x^p) = J^p \quad \text{if and only if } x^p \in S_{jp} \tag{3.2}$$

where $x^p \in R^n$, $S_{jp} \in P^p$ and $j^p \in J^p$.
For each source output x^o the indexes selected by the sequence of encoder mappings are concatenated to form the P-tuple $(j^0, j^1, ..., j^{p-1})$. Each P-tuple is an element of the Cartesian product of the index sets $(j^0, j^1 .... ..j^{p-1}) \in \{j^0 x j^1 x ... x j^{p-1}\}$
mappings $D^p : J^p \rightarrow A^p$ defined by

$$D^p(j^p) = y_{jp} \tag{3.3}$$

map each component index $j^p \in J^p$ to its corresponding code vector $J^p \in A^p$. The decoder sums the selected code vectors to form the residual quantizer's representation of the source vector x^0 by

$$X^0 = \sum_{p=0}^{P-1} y_{jp} \tag{3.4}$$

$$X^0 = \sum_{p=0}^{P-1} Q^p(x^p) \tag{3.5}$$

Where $Q^p(x^p) = D^p(\varepsilon^p(x^p))$

3.2. Formulation of the Residual Quantizer Optimization Problem

Let the distortion that results from representing x^0 with x^0 be expressed by $d(x^o, x^o)$. The expected value or average of the distortion is

$$D(x^o, x^o) = E\{d(x^a, x^o)\}. \tag{3.6}$$

A P-stage residual quantizer is said to be optimal for F if it gives a locally or globally minimum value of the average distortion. The design problem of residual quantizer performance optimization can be stated as follows:
Choose the codebooks $\{A^0, A^1, ..., A^{p-1}\}$ and partitions $\{P^0, P^1, ..., P^{p-1}\}$ that minimize the average distortion

$$D(x^o, x^o) = E\{d(x^a, x^o)\} \tag{3.7}$$

$$= E\{d[x^0, Q^p(x^p)]\} \tag{3.8}$$

$$= \int_{x_0 \in R_n} \ldots \int_{x_{p-1} \in R_n} d[x^0, Q^p(x^p)] dF(x^0, x^1, ..., x^{p-1}) \tag{3.9}$$

The minimization of (3.9) is complicated by the fact that $D(x^0,\hat{x}^0)$ requires knowledge of the joint probability distribution function $dF(x_0,x_1,\ldots, x_{p-1})$. This depends in a complicated fashion upon the sequence of codebooks and Partitioning boundaries. To avoid using (3.9), we find it useful to introduce the concept of an equivalent quantizer. The multistage residual and single stage equivalent quantizer's are identical in the sense that they produce the same representation of the source output, and they have the same expected value of distortion. We show in the next section that the equivalent quantizer allows us to express the expected value of the distortion in terms of the known source distribution F(x0). And avoid dealing explicitly with the complicated probabilistic interdependencies that exist among the stages of the residual quantizer.

3.3 Equivalent Quantizers

For our purposes, an equivalent quantizer is specified by the triple $\{A^c,Q^c,P^c\}$ consisting of .an equivalent code book, equivalent mapping, and equivalent partition. We first define the equivalent codebook A^c.

From the finite sequence of code books $\{A^0,A^1,\ldots,A^{p-1}\}$. We form a set, denoted by $\{A^0, A^1,\ldots, A^{p-1}\}$, of all ordered sums of the code vectors of the codebooks. That is,

$$\{A^0,A^1 \ldots\ldots A^{p-1}\} = \{\ (Y^0_1+Y^1_1+\bullet\bullet\bullet+Y_1^{p-2}+ Y_1^{p-1}),$$

$$(Y^0_1+Y^1_1+\bullet\bullet\bullet+Y_1^{p-2}+ Y_2^{p-1}),$$

$$\vdots$$

$$(Y^0_1+y^1_1+\bullet\bullet\bullet+Y_1^{p-2}+ Y_{Np-1}^{p-1}),$$

$$(Y^0_1+Y^1_1+\bullet\bullet\bullet+Y_2^{p-2}+ Y_1^{p-1}),$$

$$\vdots$$

$$(Y^0_{N^0}+Y^1_{N^1}+\bullet\bullet\bullet+Y^{p-2}_{N^{P-2}}+Y^{p-1}_{N^{P-1}})\} \tag{3.10}$$

where the number of elements in $\{A^0,A^1\ldots\ldots A^{p-1}\}$ is given by

$$N^c=\prod_{p=0}^{P-1}N^p \tag{3.11}$$

Each summation in $\{A^0, A^1,\ldots, A^{p-1}\}$ represents one of the possible output values of the residual quantizer. Each summation also represents a path through a tree structure that may be associated with the residual quantizer. To define a simple indexing scheme to refer to these values, (or paths), let b be a one-to-one but otherwise arbitrary function that

maps each index P-tuple $(j^0, j^1,..., j^{p-1}) \in \{J^0xJ^1x...xJ^{p-1}\}$ to an *equivalent index* $j^e \in J^e = \{1,2,3,...,N^e\}$.

Using the function ø and the set $\{j^0,j^1,...,j^{p-1}\}$ we form a new indexed set by setting

$$y^0_{j0}+y^1_{j1}+\bullet\bullet\bullet+y_{jP-1}{}^{P-1}=y^e o(j^0,j^1,...., j^{p-1}) \quad (3.12)$$

$$=y^e_{je} \quad (3.13)$$

For all $(j^0,j^1,....,j^{p-1}) \in \{j^0xj^1x...xj^{p-1}\}$. This set is called the equivalent codebook and is denoted by

$$A^e=\{y^e_{je;}\ 1\le j^e\le N^e\}, \quad (3.14)$$

Where the y^e_{je} are called the equivalent code vectors. We note that in general the equivalent code vectors are not necessarily unique.

The $(J^e)^{th}$ equivalent cell of the residual quantizer is the subset $S^e_{je} \in R^n$ such that all $x^0 \in S^e_{je}$ are mapped by the residual quantizer into y^e_{je}. That is,

$$S^e_{je}=\left\{ x^0, \quad \sum_{p=0}^{P-1} Q^p(x^p) = y^e_{je} \right\} \quad (3.15)$$

The equivalent partition $V = \{S^e_1, S^e_2, S^e_3,.... S^e_{Ne}\}$ of a residual quantizer is the collection of all equivalent cells.

An *equivalent quantizer mapping* $Q^e: R^n \rightarrow A^e$ is defined by

$$Q^e(x^0) = y^e_{je} \qquad \text{if and only if } x^0 \in S^e_{je} \quad (3.16)$$

For $1 \le j^e \le N^e$, where $x^0 \in R^n$, $S^e_{je} \in P^e$ and $y^e_{je} \in A^e$. The equivalent quantizer's map Q^e is constrained to produce the same representation of the source that the sequence of residual quantizer mappings $\{Q^p; 0 \le p \le P\text{-}1\}$ does. That is, the output of the equivalent quantizer is required to satisfy

$$Q^e(x^0)= \sum_{p=0}^{P-1} Q^p(x^p) \quad (3.17)$$

The average distortion of the equivalent quantizer is

$$D(x^0,x^0) = E\{d[x^0,Q^e(x^0)]\}, \quad (3.18)$$

$$= \sum \int s^e_{je}\, d[x^0,Q^e(x^0)]\, dF(x^0), \quad (3.19)$$

and expresses the distortion of the RQ in terms of the known source distribution $F(x^0)$. Because (3.17) constrains the equivalent quantizer and residual quantizer to produce the same representation of the source's output, the average distortions of the two quantizers are equal, that is

$$E\{d[x^0,Q^e(x^0)]\} = E\{d\,[x^0, \sum_{P=0}^{p-1} Q^p(x^p)\,]\,\} \tag{3.20}$$

By (3.19) and (3.20) the distortion of the residual quantizer may be expressed as

$$E\{d\,[x^0, \sum_{p=0}^{P-1} Q^p(x^p)\,]\,\} = \sum_{j^e \in J^e} \int_{S^e_{je}} d[x^0,Q^e(x^0)]\, dF(x^0) \tag{3.21}$$

Which is much simpler and more amenable to analysis than Equation (3.9).

3.4 Necessary Conditions for the Optimality of Residual Quantizers

Because the explicit representation of an arbitrary partition of R^n can be very complicated, the difficult problem of describing the cells S^e_{je} in (3.21) confront us. We circumvent this difficulty by attacking the optimization problem in the same manner that led to the development of the GLA design method (4,22]. That is, we first investigate in Section 3.4.1 the problem of optimizing *scalar* residual quantizers for the special case of mean squared error. The results of the scalar problem motivate an approach, which we use in Section 3.4.2 to determine conditions necessary for the optimality of *vector* residual quantizer, which hold for a broad class of distortion measures.

3.4.1 Necessary Conditions for Minimum Mean Squared Error of Scalar Residual Quantizers

Let x^0 be a continuously varying random variable described by the probability distribution function F on Rn with corresponding density function $fx^0 = F'$. Choosing squared error $d(x^\circ, x^\circ) = (x^\circ - x^\circ)^2$ as the distortion measure, the design problem of scalar residual quantizer optimization can be stated as follows:

Choose the codebooks $\{A^0, A^1, \ldots, Ap\text{-}1\}$ and partitions $\{P^0, P^1, \ldots, P^{P-1}\}$ that minimize the mean squared error

$$D(x^\circ, x^\circ) = E\{(x^0 x^0)^2\} \tag{3.22}$$

$$= E\{[x^\circ - Q^e(x^\circ)]^2\} \tag{3.23}$$

$$= E\{[X^0 - \sum_{p=0}^{P-1} Q^p(x^p)\,]\}^2 \tag{3.24}$$

$$= E\{(x^0)^2\} \tag{3.25}$$

Equation (3.25) follows from (.3.1) and (3.5), and expresses that minimum mean squared error requires minimum mean squared total residual error.

Optimum Quanta

Assuming that the stage wise partitions $\{P^0, P^1, ..., P^{P-1}\}$ are fixed, we seek the stagewise code books $\{A^0, A^1, ..., A^{P-1}\}$, that minimize (3.25). Fixed stagewise partitions imply that the equivalent partition P^e is also fixed. Optimization of $\{A^0, A^1, ..., A^{P-1}\}$ for a fixed $\{P^0, P^1, ..., P^{P-1}\}$ corresponds to optimization of A^e for a fixed P^e, where the $y^e_{je} \in A^e$ are responsible as sums of code vectors selected from P code books consisting of N^p; $0 \le p \le P\text{-}1$ code vectors. The set of all such A^e that corresponds to a fixed P and N^p; $0 \le p \le P\text{-}1$ forms the class of allowed equivalent codebooks. We seek a codebook A^e within this class that minimizes the expected distortion thus, optimization of the equivalent codebook. A^e within the class of allowed single stage code books for a fixed P^e, corresponds to optimization of $\{A^0, A^1, ..., A^{P-1}\}$ for a fixed $\{P^0, P^1, ..., P^{P-1}\}$. Since the construction of each $y^e_{je} \in A^e$ is given explicitly by Equation (3.13), we choose an optimal A^e within this class by choosing the best stagewise quanta used to form the $y^e_{je} \in A^e$.

It is useful in the following derivation to define a set of functions that identify the p^{th} component quantum that is used to construct each $(j^e)^{th}$ equivalent quantum. Functions that fulfill this role are $\gamma^e: J^e \rightarrow J^p$ for $0 \le p \le P\text{-}1$ which are defined as

$$\gamma^p(j^e) = j^p \text{ if and only if } j^e \varnothing (j^0, j^1, ..., j^{p-1}) \tag{3.26}$$

The maps γ_p are component-wise inverse mappings of the function $\varnothing$, and allow us to express the $(j^e)^{th}$ equivalent quantum as

$$y^e_{je} = \sum_{p=0}^{P-1} y_{\gamma p(je)} \tag{3.27}$$

Using Equations (3.16), (3.19), and (3.27) we express the mean squared error of the scalar residual quantizer as

$$D(x^0, x^0) = \sum_{j^e \in J^e} \int s^e_{je} \left(x^0 - \sum_{p=0}^{P-1} y_{\gamma p(je)} \right)^2 f_{x0}(x^0)\, dx^0 \tag{3.28}$$

To determine the best stagewise quanta, we minimize (3.28) with respect to the $(k^p)^{th} \in \{1,2,..., N^p\}$ quantum of the $p^{th} \in \{0,1, ..., P\text{-}1\}$ stage by setting the partial derivative with respect to y_{kp} equal to zero, i.e.

$$\partial / \partial y_{kp} \left[\sum_{j^e \in J^e} \int s^e_{je} \left(x^0 - \sum_{p=0}^{P-1} y_{\gamma p(je)} \right)^2 f_{x0}(x^0)\, dx^0 \right] = 0 \tag{3.29}$$

$$\sum_{j^e \in J^e} \int s^e_{je} 2\Big(x^0 - \sum_{P=0}^{p-1} y_{\gamma p(je)}\Big)\Big[\partial / \partial y_{kp} \Big(\sum_{p=0}^{p-1} y_{\gamma p(je)}\Big)\Big] f_{x0}(x^0)\, d\, x^0 = 0 \quad (3.30)$$

The partial derivative in (3.30) is evaluated as

$$\partial / \partial y_{kp}\Big(\sum_{p=0}^{p-1} y_{\gamma p(je)}\Big) = \begin{cases} 1 \text{ if } j^e \in H_{kp} \\ 0 \text{ otherwise} \end{cases} \quad (3.31)$$

where $H_{kp} = \{j^e : \gamma^p(j^e) = k^p\}$ That is, H_{kp} is the subset, of all indexes in J^e corresponding to equivalent quanta that contain y_{kp} in their construction. Using (3.31) we write (3.30) as

$$\sum_{j^e \in H_{kp}} \int s^e_{je}\Big(x^0 - \sum_{p=0}^{P-1} y_{\gamma p(je)}\Big) f_{x0}(x^0)\, d\, x^0 = 0. \quad (3.32)$$

To make explicit the fact that all equivalent quanta y^e_{je} with $j^e \in H_{kp}$ have y_{kp} in their construction, we let

$$\sum_{p=0}^{P-1} y_{\gamma^p(j^e)} = \sum_{\substack{p=0 \\ p!=0}}^{P-1} y_{\gamma p(je)} + y_{kp} \quad (3.33)$$

Substituting (3.33) into (3.32) we obtain

$$\sum_{j^e \in H_k^p} \int s^e_{je}\Big(x^0 - \sum_{p=0}^{p-1} y_{\gamma^p(j^e)} - y_{kp}\Big) f_x^0(x^0)\, d\, x^0 = 0. \quad (3.34)$$

which we solve for y_k^p

$$y_k^p = \frac{\sum_{j^e \in H_k^p} \int s^e_j \Big(x^0 - \sum_{p=0}^{P-1} y_{\gamma^p(j^e)} - y_k^p\Big) f_x^0(x^0)\, d\, x^0}{\sum_{j^e \in H_k^p} \int s^e_j f_x^0(x^0)\, d\, x^0} \quad (3.35)$$

$$\sum_{\substack{p=0 \\ p!=P}}^{P-1} y_{\gamma^p(j^e)} \quad (3.36)$$

In Equation (3.35) the expression differs from the construction of the $(j^e)^{th}$ equivalent quantum in that the p^{th} node of the $(j^e)^{th}$ path through the tree is removed. It is important to note that since $p \in \{0,1, ..., P - 1\}$, the removed node is not necessarily at the end of the path through the tree and hence does not correspond to a pruning. We interpret Equation (3.36) as the $(j^e)^{th}$ path with one node removed and the two remaining portions of the path reconnected or "grafted" back together. For each possible path y^e, through the tree structure, we denote the corresponding pth grafted path or pth grafted branch by the superscript p=0,1,...,P-1 as

$$Y^{p}{}_{j^e} = \sum_{\substack{p=0 \\ p!=P}}^{P-1} y_{Y^{p}(j^e)} \qquad \text{for } 1 \le j^e \le N^e \tag{3.37}$$

In terms of the grafted branches, (3.35) becomes

$$y_k^p = \frac{\sum_{j^e \in H_k^p} \int s^{e,e}_{j^e}\left(x^0 - y^p{}_{j_e}\right) f_x^0(x^0)\, d\, x^0}{\sum_{j^e \in H_k^p} \int s^{e,e}_{j^e} f_x^0(x^0)\, d\, x^0} \tag{3.38}$$

Equation (3.38) is a key expression. It is similar in form to the centroid expression, which is a necessary condition for the optimality of the quanta of single-stage Lloyd-Max quantizers.

Optimum Partitions

Assuming that the codebooks $\{A^0, A^1,..., A^{P-1}\}$ are fixed. We seek the equivalent partition P^e that minimizes (3.25). The fixed stagewise codebooks imply that the equivalent codebook A^e is also fixed. Once we determine an optimal P^e for the fixed A^e, we determine a sequence of stagewise partitions $\{P^0, P^1,...., P^{p-1}\}$ from the class of allowed stagewise partitions. The allowed class of stagewise partitions consists of collections of P partitions with N^p :$0 \le p \le$ P-1 cells in each partition. Since P^e and the derived $\{P^0, P^1,..., P^{p-1}\}$. By definition, partition R the same way, optimization of P^e for a fixed A^e corresponds to optimization of $\{P^0, P^1,..., P^{p-1}\}$ for a fixed $\{A^0, A^1,..., A^{p-1}\}$.
We make the reasonable assumption that an optimum equivalent partition $P^e = \{S_1^e, S2^e,..., S^e{}_N{}^e,\}$ consists of a collection of *connected,* cells. That is each equivalent cell is a finite or semi-infinite interval of R, which we denote? by $S^e_j{}^e = (x^e_j{}^{e-1}, x^e_j{}^e)$. The partition P^e may be described by the N^{e+1} points. $\{ x^e{}_0, x^e{}_1, ..., x^e{}_N{}^e \}$. Without loss of generality, we assume that the one-to-one function $\varphi : \{J^o \times J^1 \times ... \times J^{p-1}) \rightarrow J^e$ and the equivalent partitions are labeled such that the following ordering property is satisfied

$$x^e{}_0 < y^e{}_1 < y^e{}_2 < ... < x^e{}_N{}^{e-1} < y^e{}_N{}^e < x^e{}_N{}^e \tag{3.39}$$

Where $x^e{}_{0=}$ - infinity and $x^e{}_N{}^e$ = infinity

For j° = 1,2, ..., N^0, where the P-tuple $(k^0, k^1, ..., k^{p-1})$ vary over all members of the Cartesian product set $\{J^0 \times J^1 \times ... \times J^{p-1}\}$ and the (P −1)-tuples $(k^1, k^2, ... k^{p-1})$ vary over all members of the Cartesian product set $\{J^1 \times J^2 \times ... \times J^{p-1}\}$. The subset of all $x^0 \in R$ such that Equation (3.46) holds for each $j^0 \in J^0$ defines the optimal equivalence class S_j^0, where all $x^0 \in S_j^0$ map as $Q^0(x^0) = y_j^0$ and also $Q^e(x^0) = y^e_{je}$ in accordance with the optimal partitioning rule Equation (3.42). In words, Equation (3.46) identifies S_j^0 as the subset of R that is Voronoi with respect to the terminating nodes of all paths in $\{A^0 + A^1 + ... + A^{p-1}\}$ which contains y_j^0 in their construction.

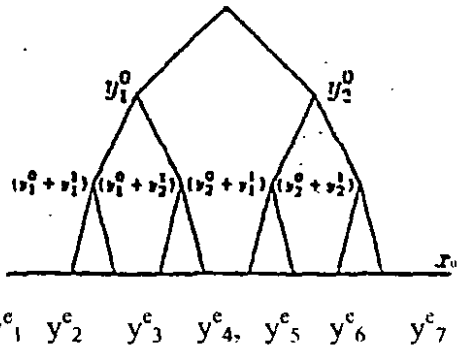


Figure 3.1: Example of an unentangled tree of a three stage, two quanta per stage scalar residual quantizer.

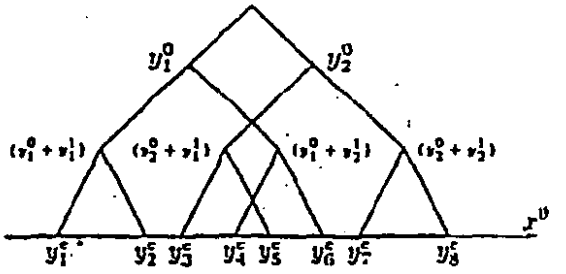


Figure 3.2 Example of a partially entangled tree of a three stage, two quanta per stage scalar residual quantizer.

Figures 3.1 and 3.2 represent three stages, two quanta per stage scalar residual quantizer by their tree structures. Each level of the trees (except, the root nodes which represent the origins) represents a particular stage of the residual quantizers. The value of the nodes gives either an intermediate or final quantizer level constructed by the decoders. In particular, Figure 3.3 represents an unentangled *tree*, while Figure 3.4 represents an entangled tree. The degree of entanglement affects the form of the stagewise equivalence classes S_{jp}. For example, in Figures 3.5 and 3.6 the nodes of the first layer have been labeled with the corresponding quanta of the first code book $A^0 = \{y^0_1, y^0_2\}$ Equation (3.65) expresses that for optimal stagewise encoding, the equivalence classes $\{S^0_1, S^0_2\}$ of the partition P^0 must be the union of all equivalent quantizer Voronoi cells that correspond to the terminating nodes of the sub-trees of y^0_1 *and* y^0_2 respectively

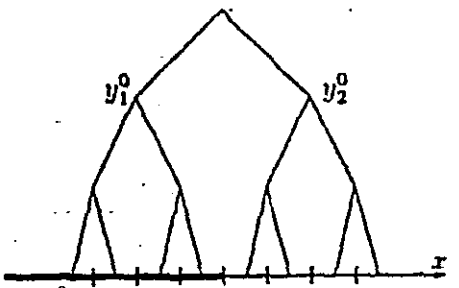


Figure 3.3: The equivalence class S^0_1 (indicated by thick lines) and S^0_2 (indicated by thin lines) in an unentangled tree.

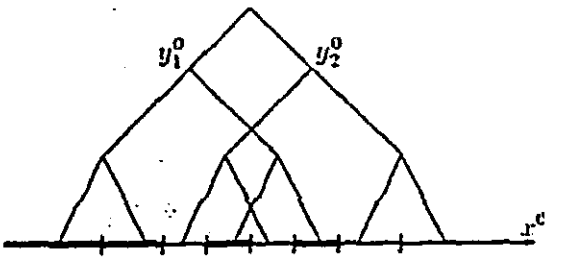


Figure 3.4: The equivalence class S^0_1 (indicated by thick lines) and S^0_2 (indicated by thin lines) in an entangled tree.

Comparing the unentangled tree of Figure 3.3 and the entangled tree of Figure 3.4 we see a fundamental difference in the construction of the equivalence classes $\{S^0_1, S^0_2\}$ for the two different trees. The equivalence class S^0_1 in Figure 3.3 is a connected interval of R and may be distinguished from the equivalence class S^0_2 by a single boundary point; while the equivalence class S^0_1 in Figure 3.4 is a union of three disjoint intervals. An encoder of greater complexity is required to distinguish the equivalence classes of entangled trees than the encoders of unentangled trees. We proceed to determine the optimal equivalence classes S_j^p for the remaining stages.

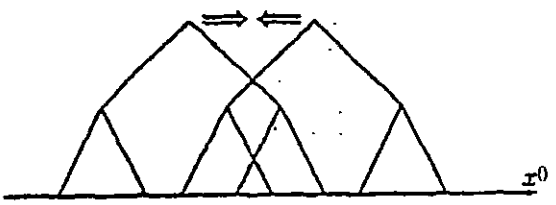


Figure 3.5: The translations of the sub-trees of $\{A^0 + A^1 + A^2\}$ to form the smaller tree $\{A^1 + A^2\}$

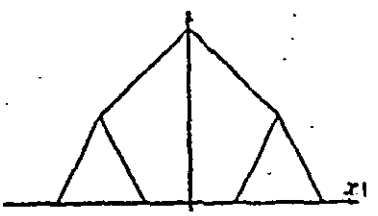


Figure 3.6: The tree $\{A^1 + A^2\}$

When the residual x^1 is formed by the difference $x^1 = x^0 - Q^\circ(x^0)$. The tree structure $\{A^0+A^1+...+A^{P-1}\}$ is modified by subtracting the first component of each path of the tree to produce the smaller tree structure $\{A^1+A^2+...+A^{P-1}\}$. The difference $x^1 = x^\circ - Q^\circ(x^\circ)$ causes shift or translation of each of the sub-trees corresponding to nodes in the first layers of $\{A^0+A^1+...+A^{P-1}\}$. Each of the translated sub-trees superimposes to form the smaller tree $\{A^1+A^2+...+A^{P-1}\}$. Where the root node of the smaller tree occurs at the origin of the residual. To determine the optimal partition P^1 of the residual x^1 we assume that x° has been optimally mapped by Q° and we fix y_j^1 to determine S_j^1. That is, we write

$$d(x^\circ, Q^\circ(x^0)+ y_j^1 + \sum_{p=2}^{P-1} y_{kp}) \le d(x^\circ, Q^\circ(x^0)+\sum_{p=o}^{P-1} y_{kp}) \qquad (3.47)$$

For $j^1 = 1,2, \ldots, N^1$, where the (P - 1)-tuple $\{k^1,k^2,\ldots,k^{P-1}\}$ varies over all members in the Cartesian product-set $\{J^1 \times J^2 \times \ldots \times J^{P-1}\}$ and likewise for the (P- 2)-tuple $\{k^2,k^3,\ldots,k^{P-1}\}$. Assuming that $d(x,y)$ is translation or position invariant in the sense that $d(x,y) = d(x-z, y-z)$ for any $x, y, z \in R$. we subtract $Q^\circ(x^\circ)$ from every term in (3.47) to obtain

$$d(x^0 - Q^0(x^0), y_j^1 + \sum_{p=2}^{P-1} y_{kp}) \le d(x^\circ - Q^\circ(x^0), \sum_{p=0}^{p-1} y_{kp}) \qquad (3.48)$$

Substituting $x^1 = x^0 - Q(x^0)$ into (3.48), we get

$$d(x^1, y_j^1 + \sum_{p=2}^{P-1} y_{kp}) \le d(x^1, \sum_{p=0}^{P-1} y_{kp}) \qquad (3.49)$$

the subset of all $x^1 \in R$ that satisfy (3.48) defines each equivalence class $S_j^1 : 1 \le j^1 \le N^1$, where all $x^1 \in S_j^1$ map as $Q^1(x^1) = y_j^1$ and all corresponding $x^0 = x^1 + Q^\circ(x^0)$ map as $Q^c(x^0) = y_j^c$ in accordance with the optimal partitioning rule (3.42). Similar to the result for P°, (3.49) identifies S_j^1 *as the subset* of R that is Voronoi with respect to the terminating nodes of all paths in the smaller tree $\{A^1 \oplus A^2 \oplus \ldots \oplus A^{P-1}\}$ that contains y_j^1 in their construction.

In general, this procedure can be repeated to determine $P^p = \{S_j^1 : 1 \le j^p \le N^p\}$ for $0 \le p \le$ P-1 by identifying all x^p such that

$$d(x^p, y_j^p + \sum_{p=p+1}^{P-1} y_{kp}) \le d(x^p, \sum_{p=p}^{P-1} y_{kp}) \qquad (3.50)$$

holds true for each $j^p \in J^p$, where $(k^p, k^{p+1}, \ldots, k^{P-1})$ varies over all members in $\{J^p \times J^{p+1} \times \ldots \times J^{P-1}\}$ and likewise for $(k^{p+1}, k^{p+2}, \ldots, k^{P-1})$. The optimal equivalence class S_j^p is the subset of R that is closest (in the sense of $d(x, y) = (x-y)^2$) to the terminating nodes of all paths of the tree $\{A^1 \oplus A^2 \oplus \ldots \oplus A^{P-1}\}$, which contain y_j^p in their construction.

3.4.2 Necessary Conditions for Minimum Distortion of Vector Residual Quantizers

We give vector generalizations of the scalar conditions necessary for the optimality of scalar residual quantizers. We also generalize the results of Section 3.4.1 to distortion measures more general than simple menu squared error. The class of distortion functions considered in tins section arc assumed to be nonnegative real-valued functions that satisfy the following requirements:

1. For any x, y, z $\in R^n$, d(x, y) is translation or position invariant in the sense that $d(x, y) = d(x - y, y - z)$.
2. For any fixed $x \in R^n$, $d(x, y)$ is a convex function of y. that is for $y_1.y_2 \in R^n$, $\lambda \in (0, 1)$, $d(x, \lambda y_1 + (1-\lambda)y_2) \leq \lambda d(x, y_1) + (1-\lambda)d(x, y_2)$. If the inequality is strict then $d(x, y)$ is strictly convex in y.
3. For any fixed x, if y(k) a= $(y_1(k), y2(k),...,y_n(k))) \rightarrow \infty$ as $k \rightarrow \infty$ (that is, $y_i(k)$ diverges for some i), then $d(x, y(k)) \rightarrow \infty$ also.

Properties (1) and (2) are key assumptions, property (3) is a technical condition imposed to avoid pathological cases . Assumption (1) is a necessary constraint resulting from the residual structure imposed on the vector quantizer. All distortion measures that depend on the difference (x - y) satisfy (1). since $d[(x-z)-(y-z)]=d(x-y)$.

For the general class of distortion measures determined by these assumptions, we proceed to derive necessary conditions for the optimality of RQ code vectors in R^n and necessary conditions for the optimality of stagewise partitions of R^n for each of the residual quantizer stages.

Chapter 4
Analysis and Design

4. Analysis and Design

Digital images and video signals exhibit large temporal correlations that can be exploited in a compression algorithm to bring down the bandwidth requirements. There certain compression algorithms reported in literature for this purpose. One such classical algorithm is differential pulse code Modulation (DPCM). The basic idea behind DPCM scheme is to predict value of a current input pixel based on neighboring pixel values using certain prediction coefficients. The difference between predicted value and actual value of the pixels emerge as differential or Residual image, which is much less correlated than original image. The Residual image is then coded and sent. The schematic diagram of the DPCM coder/decoder is shown in fig 4.1.a and 4.1.b.

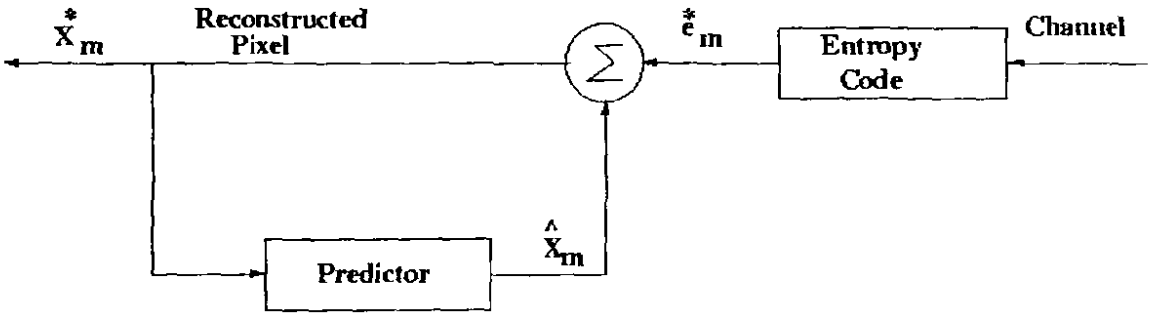


Fig.4.1.a. DPCM CODER

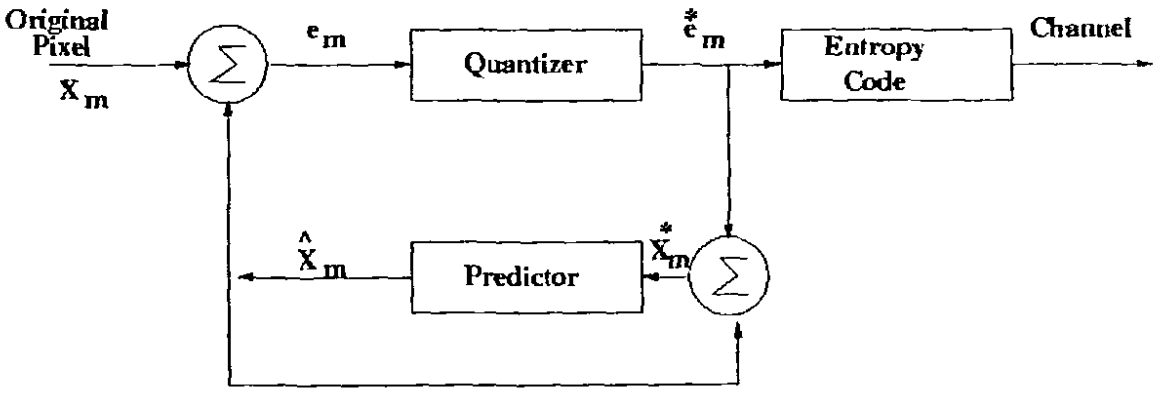
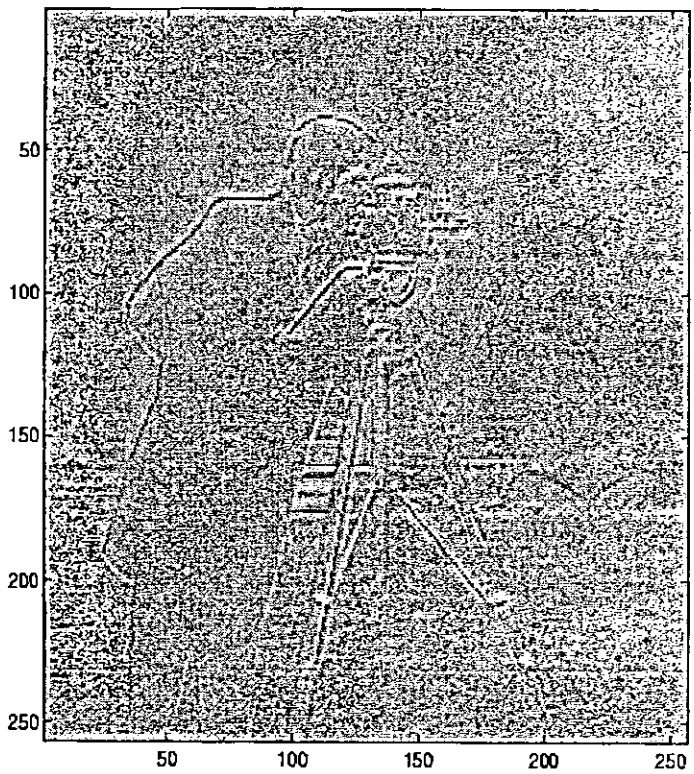


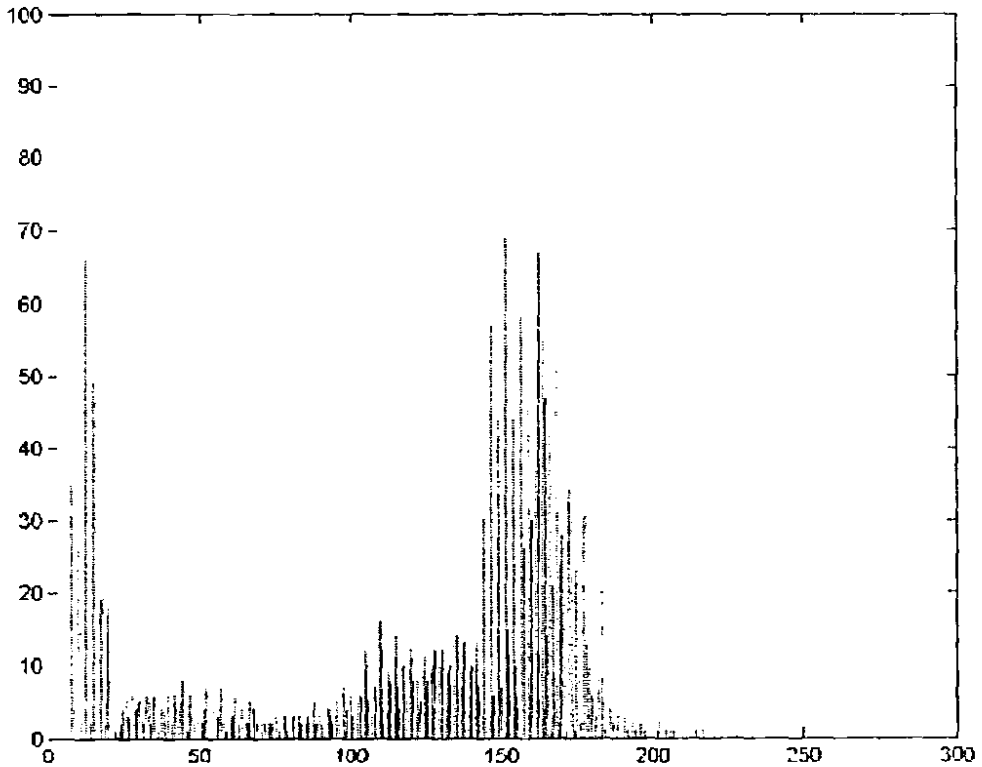
Fig.4.1.b. Differential pulse code Modulation DECODER

The Fig 4.2.a and 4.2.b shows the original cameraman image and Residual or Differential image. The differential image exhibits variance in the range of 42 as apposed to 433 for original Cameraman image. This illustrates that DPCM has been very effective in reducing spread of pixel values by at most 10:1 ratio. This reduction is very helpful in providing compression. Due to its simple structure DPCM has made its place in the standard algorithms like JPEG [14] for still image coding, H.261 [15] for videophones and videoconference communications, and MPEG [16] for interactive media applications. However performance of DPCM has certain drawbacks, which prevent its uses in certain circumstances. The two main drawbacks are channel error sensitivity and poor rate distortion performance at low bit rates [17].

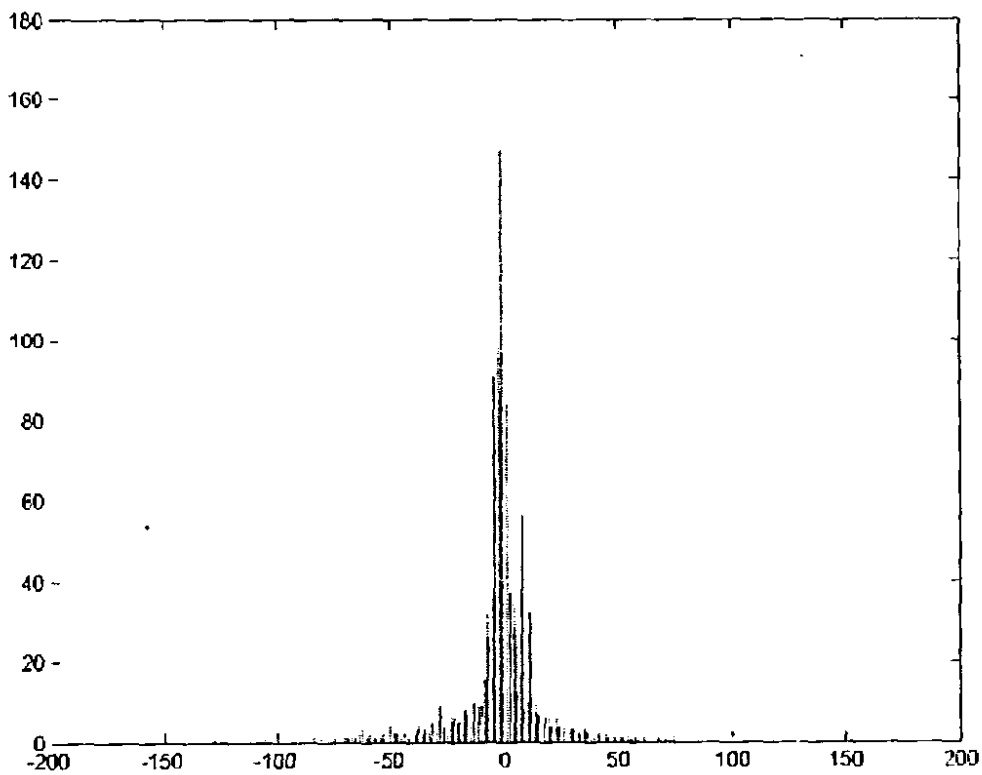
Fig. 4. 2. (a) Original Cameraman image



4.2.(b). The prediction error/ differential image.



4.2(c). Gray-level histogram of the original Cameraman image with variance 433.05.



4.2(d). Histogram of the Prediction errors with variance 43.33.

Another approach that generalizes DPCM is to code prediction Residual image using Vector Quantization (VQ). Generally, there are several arguments in support of VQ for image and video coding. Shannon rate-distortion theorem indicates that VQ always perform better than conventional vector Quantization for the simple reason that Quantization block of samples, as one unit is a general case of quantizing individual samples in isolation. The DPCM structure that involves VQ as its Quantization module is referred to as predictive Vector Quantization (PVQ) proposed in [18]. It was stated that although PVQ structure is simple and well understood, its design is problematic due to feed back loop involved and standard method often fail to produce optimal or even good predictors and quantizers. The design of vector quantizer in a feed back loop is in need of representative training set of prediction error image. However the representative training set has dependence on both predictor and quantizer. The predictor and quantizer have to be optimized jointly with respect to each other in order to come up with a reason able prediction error image suited for training quantizer.

Two simple approaches for PVQ design were introduced earlier by Cuperman and Gresho [19]. The first approach referred as open loop solves the vector quantizer design problem by assuming no feedback, and operates directly on original source vectors. An improvement was suggested by using second approach called closed-loop design. In this case, an iterative design is employed for updating the training set and quantizer given a fixed predictor. Later on, closed-loop design algorithm was modified further modified by Chang and Gray [20], where both vector predictor and vector quantizer are jointly optimized. The joint design algorithm provides an improvement over the previously stated designs however; such design approaches exhibit significant stability problems especially at low bit rates. The stability of closed-loop design is analyzed and a modified design with the name of asymptotic closed-loop (ACL) [21] starts as an open loop and then tries to simulate closed loop behavior over a long run.

Vector Quantization (VQ) used in PVQ is a powerful technique for data compression of speech, image, and video signals. Vector Quantization takes advantage of linear or nonlinear correlations that exists among the vector components. Therefore larger the vector size is used grater the compression rate can be achieved. However an issue of recognized importance for large block implementation of VQ is that size of codebook associated with associated with VQ grows exponentially as a product of Vector dimension an bit rate. With an increase in code book size comes an overwhelming increase in search complexity. To overcome the complexity barrier, many researches have suggested imposing certain constraints on the VQ codebook design. The relief in search complexity was obtained by replacing VQ with multistage VQ referred to as residual vector Quantization. The argument is that multistage VQ can be designed sequentially and its stage codebooks are smaller in size and thus require less search complexity. Researches also considered reducing further the search complexity by applying additional structural constraints on multistage VQ, and proposed multistage VQ with stage code books comprised of lattice VQ's [22] and reflected RRVQ [23]. Our focus in the thesis is the utilization of RRVQ.

A reflected RVQ (RRVQ) is a multistage structure with binary stage codebooks. The encoder and decoder of RRVQ perform reflecting and folding operations on residual vectors between stages. This reflecting operation forces a certain symmetry on the resultant codebooks, which in turn makes the sequential search of stage codebook

optimal. The sequential search ability of RRVQ makes it an ideal candidate for large block VQ implementations. Recently an entropy-constrained RRVQ (EC-RRVQ) design was introduced [24].

In order to extract linear and most of the non-linear correlations among image source, we suggest the use of large-block RRVQ with the current PVQ structure. The purpose of large block RRVQ is to take advantage of linear/nonlinear correlations present among the block of pixels and the predictive structure with the feedback loop to exploit the remaining intra-block dependencies. Our goal in this thesis is to simulate the behavior of RRVQ in a feedback loop that exploits correlation in a given image source in order to produce high compression ratios with lower complexity. A new design algorithm for RRVQ in a predictive environment has been proposed in this thesis, which is referred to as Predictive Reflected Residual Vector Quantization (PRRVQ). The rest of the chapter includes sections for the design parts of the technique.

4.1.Reflected Residual Vector Quantization:(RRVQ)

RRVQ is a Special form of RVQ in which only two code vectors are allowed in one stage. A p-stage residual quantizer consist of a finite sequence of P vector quantizers $\{(C^p, p^p);\ 1 \le p \le P\}$. We index the code vectors of pth-stage as $\{y^p_0, y^p_1, y^p_2, \ldots, y^p_{N-1}\}$ and voronoi cells of the pth-stage as $\{S^p_0, S^p_1, S^p_2, \ldots, S^p_{N-1}\}$. The code vectors comprising the codebook C^p and the cells comprising the partition P^p are indexed with the subscript j^p . The quantized representation $\aleph^l$ of the input source vector x^l is formed by the sum of the selected stage code vectors.

$$\aleph^l = \sum_{p=1}^{P} y^p_{j^p} \qquad (4.1)$$

For adopting a jointly optimized approach, an RVQ is represented with an equilent quantizer, referred to as direct sum quantizer (C^e, P^e). The elements of C^e are the elements of the set of all possible sums of stage code vectors. That is $C^e = C^1 + C^2 + \ldots + C^P$ .C^e can be interpreted as the terminating node of a path through tree structure, which is associated with the Residual vector quantizer mentioned earlier, corresponding to the direct sum quantizer. A tree structure of three stages two code vectors/ stage RVQ is illustrated in Fig 1. The root node the tree represents x^l. The leaf nodes represent the set C^e of the direct sum code vectors. The intermediate node represents the partial sums of the direct sum code vectors and branches represent stage code vectors. If two code vectors $\{y^p_0, y^p_1\}$ are allowed as in RRVQ in a given stage, then the Voronoi boundary is a plane of equal distortion between two code vectors.

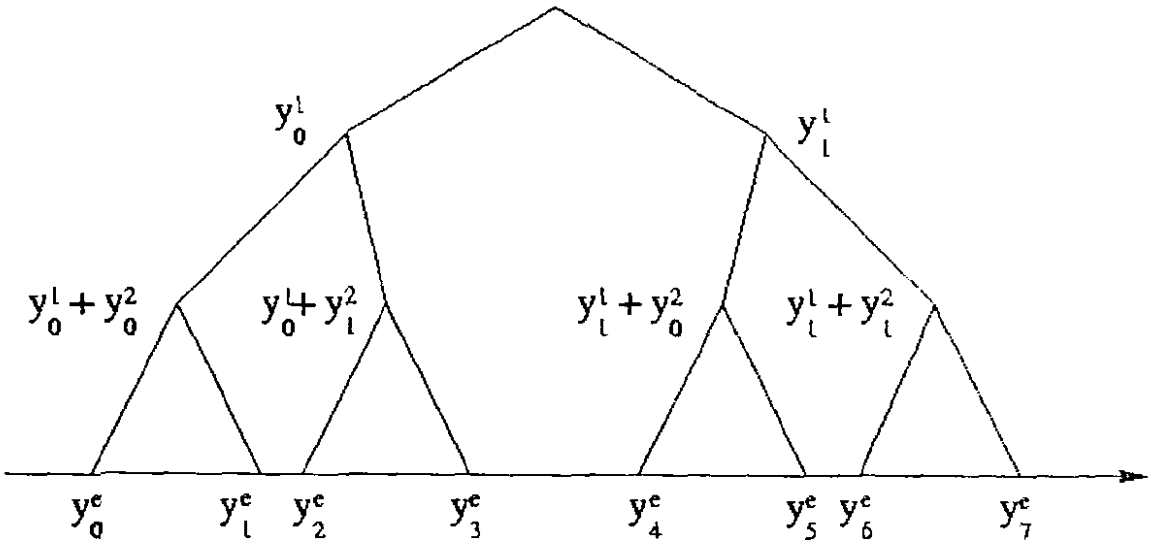


Fig. 4.3. Three-stage binary RVQ tree structure.

Voronoi boundary can be specified by midway point m^p between the two given code vectors for that particular stage p as

$$m^p = \tfrac{1}{2}(y^p{}_0 + y^p{}_1) \tag{4.2}$$

The normal vector n^p is defined to be the line joining two code vectors $\overline{y^p{}_0,\ y^p{}_1}$. The equation of the plane through the midway point m^p perpendicular to the n^p is

$$\overline{n^p . m^p}\ z^p = 0 \tag{4.3}$$

Where z^p is the any point in the plane. In order for this hyper-plane to specify also the boundary between adjacent children of the two code vectors. We reflect the input vectors of the pth-stage to one side of the hyper plane boundary, and by convention, we reflect all $x^{p,\ which}$ belongs to voronoi cell $S^p{}_1$ to second voronoi cell $S^p{}_0$. After reflection we subtract $y^p{}_0$ from reflected input vector forming a reflected residual vector. Then the residual vectors that represent the next stage code vectors will lie in the reflected residual space. If we are to unreflect all the reflected stage codebooks, the resulting direct sum codebook has the desired symmetry properties.

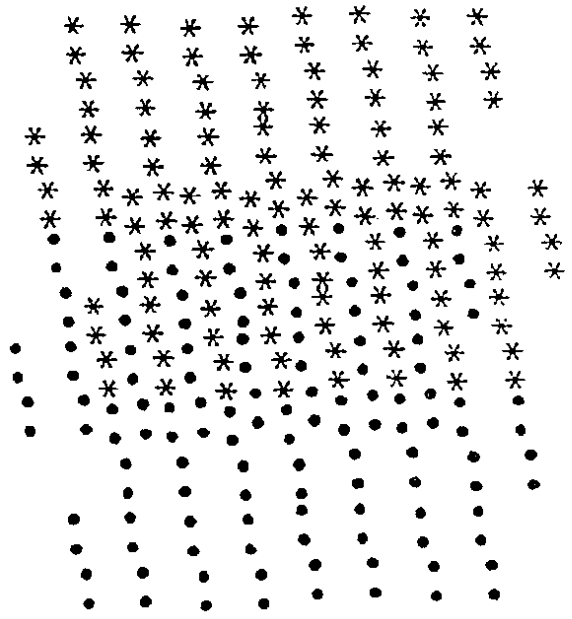


Fig. 4.4.a. Gaussian source coded with a binary 8-stage, two-dimensional quantizer.
(a) Equivalent code vector constellation of RVQ.

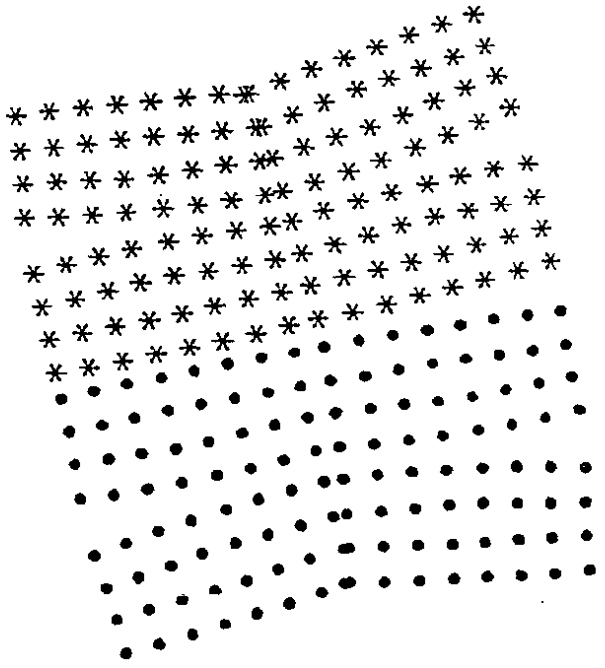


Fig 4.4. Gaussian source coded with a binary 8-stage, two-dimensional quantizer.
(b) Code vector constellation of Ref-RVQ.

To illustrate the structure of Ref-RVQ codebook, Fig 4.4.(a) shows the code vector constellation for (8-stages, 2 code vector/stage) two-dimensional RVQ designed for Gaussian source. Similarly Fig 4.4.(b) Shows the code vector constellation (8-stages, 2 code vector/stage) two-dimensional Ref-RVQ designed for Gaussian source. All the direct sum code vectors that involve the first code vector of the first RVQ stage in their construction are represented as dots. On the other hand, asterisks are used for the direct sum code vectors constructed with second code vector of the first stage RVQ. Fig 4.2.(a) indicates severe code vector diffusion for RVQ, where as Ref-RVQ (Fig 4.2.(b)) shows no diffusion and hyper plane boundaries are evident. The presence of Voronoi cells with hyper-plane boundaries in Ref-RVQ codebook makes the sequential single path optimal.

4.2.Entropy Constrained Reflected RVQ (EC-RRVQ)

For Entropy constrained design algorithm, the distortion trade of squared error with code word rate. In case of residual VQ (RVQ), we will have the Lagrangian

$$J_\lambda = E[d(x^l, C^e)] + \lambda L(C^e) \tag{4.4}$$

Where L (C^e) is the length associated with a direct sum code-vector. For the case of developing an entropy-constrained design for the Ref-RVQ, we have binary stages i.e., 2 code-vectors/stage. This will form the Voronoi region boundary between the two code vectors as plane. For an EC-Ref-RVQ, we need to work with the plane of equal Lagrangian as opposed to the plane of equal distortion used in fixed-rate Ref-RVQ design. We define a plane of equal Lagrangian as

$$\begin{aligned} &\|x^p - y^p{}_0\|^2 + \lambda L(y^p{}_0 | C^{p-1}, C^{p-2}, \ldots, C^1) \\ &\quad = \|x^p - y^p{}_1\|^2 + \lambda L(y^p{}_1 | C^{p-1}, C^{p-2}, \ldots, C^1) \end{aligned} \tag{4.5}$$

We restate the above Equation as the more familiar normal plane equation $n.x^p = d$ as

$$\begin{aligned} \frac{(y^p{}_0 - y^p{}_1)}{\|y^p{}_0 - y^p{}_1\|} . x^p = \; & \frac{\|y^p{}_0\|^2 - \|y^p{}_1\|^2}{2\|y^p{}_0 - y^p{}_1\|} \\ & + \frac{\lambda L(y^p{}_0 | C^{p-1}, C^{p-2}, \ldots, C^1) - L(y^p{}_1 | C^{p-1}, C^{p-2}, \ldots, C^1)}{2\|y^p{}_0 - y^p{}_1\|} \end{aligned} \tag{4.6}$$

The shortest distance from $y^p{}_1$ to the plane will be given by

$$|d| - n.y^p{}_1 \tag{4.7}$$

Thus the new midpoint in case of entropy-constrained Ref-RVQ (EC-Ref-RVQ) will be given by

$$m^p = \frac{y^p{}_1 + \|y^p{}_0 - y^p{}_1\|}{2}{}_n + \frac{\lambda L(y^p{}_0 | C^{p-1}, C^{p-2}, \ldots, C^1) - L(y^p{}_1 | C^{p-1}, C^{p-2}, \ldots, C^1)}{2\|y^p{}_0 - y^p{}_1\|} \quad (4.8)$$

By having a look at Equation.(4.8) one notice that unlike fixed-rate Ref-RVQ the midpoint for a given stage will not be equidistant from the two codevectors, but will be offset by an amount dependent on the difference of lengths between the two. The midpoint will move in the direction of the larger length code vector because of the third term in the Equation. (4.8).

An important complexity-reducing feature of EC-Ref-RVQ is its potential to use stage-conditional entropy tables of relatively sizes, where conditioning is performed on previous stages. With the use of smaller Markov model order x , a large reduction in entropy-tables storage can be obtained. The length for a direct sum code vector is given by

$$L(C^e) = L(C^0) + L(C^1 | C^0) + \ldots + L(C^p | C^{p-1}, C^{p-2}, .., C^0) \quad (4.9)$$

For a given Markov model order x the above Equation. can be approximated as

$$L(C^e) = L(C^0) + L(C^1 | C^0) + \ldots + L(C^p | C^{p-1}, C^{p-2}, .., C^{p-m}) \quad (4.10)$$

Where P-m >>0

4.3. ASYMPTOTIC CLOSED-LOOP DESIGN

The asymptotic closed-loop design was originally proposed for video coding. In this section we will adopt ACL design for incorporating reflected residual vector Quantization in a PVQ structure. The encoding and decoding operation for PRRVQ is essentially same as being used in earlier PVQ implementations. However, the design of RRVQ stage codebooks in a feedback loop needs attention. The design of stage codebooks under ACL approach can be best explained by first introducing some mathematical notations.

Given a set of source vectors, $X:\{x_n\}^N{}_{n=0}$, the training set of prediction errors at an iteration i-1 is generated by

$$e^{(i)}{}_n = x_n - Pred[\aleph^{(i-1)}{}_{n-1}], \quad n=1,2,3,\ldots,N \quad (4.11)$$

The quantizers at iteration i – 1, are denoted by $Q^{i-1}{}_1$, $Q^{i-1}{}_2$,..., $Q^{i-1}{}_p$., where for example the $Q^{i-1}{}_p$ notation represents pth stage quantizer at the i - 1th iteration.. Training set of prediction errors is generated for the next iteration i as .

$$T_{(i)} = \{ e^{(i)}{}_n \}^N{}_{n=1}.$$

Where $e^{(i)}{}_n = x_n - Pred[\aleph^i{}_{n-1}]$, and reconstruction vectors are produced as

Chapter 5
Implemetation

5. Implementation

Implementation basically consists of three modules all the three modules work independently but has an impact on other module.
Modules are

1. Blocking code
2. Prediction code
3. Encoder/Decoder

First of all a training set is created using sample image in Matlab. The set is then converted into blocks of different sizes depending on the requirements.
The blocked training set is then handed over to the prediction module a code written in matlab. This module is also used to reverse the process at the end of cycle.
The blocked predicted data in the form of training set is given to Encoder/Decoder pair. This Module quantizes the data.

5.1.Blocking Code

It is a program written in whose only purpose is to convert the input into blocks. The data is converted into blocks in order to utilize block prediction instead of pixel prediction. The code also unblocks the data after its quantization. The same code and process is applied again to reverse the blocking data into an unblocked image structure.

The main portion of the blocking code is given below.

```
blocks_per_col = rows/blockheight;
 blocks_per_row = cols/blockwidth;
 num_cols = blocks_per_row*blockwidth;
 num_rows = blocks_per_col*blockheight;
 num_blocks = blocks_per_col*blocks_per_row;
 num_pixels = rows*cols;
 vector_length = blockheight*blockwidth;

 /* allocate memory for the raw image and the block image  */
 if (!(raster_image = (DATA *) calloc(num_pixels, sizeof(DATA))) ||
    !(blocked_image = (DATA **) calloc(num_blocks, sizeof(DATA *)))) {
  fprintf(stderr,"%s: %s\n",programname,NOMEMORY);
  exit(10);
 }
 /* allocate memory for the block image elements */
 for(i=0; i<num_blocks; i++) {
  if (!(blocked_image[i] = (DATA *) calloc(vector_length,sizeof(DATA)))) {
    fprintf(stderr,"%s: %s\n",programname,NOMEMORY);
    exit(11);
  }
```

```
}

/* read contents of inputfile into raster_image array */
clearerr(inputfile);
if (fread(raster_image,sizeof(DATA),num_pixels,inputfile)!=num_pixels ||
    feof(inputfile) || ferror(inputfile) ) {
  fprintf(stderr,"%s: %s: %s\n",programname,inputname,NOREAD);
  exit(12);
}

/* create the block vectors and write them to the output file */
for(i=0; i<num_blocks; i++) {
  for(j=0; j<vector_length; j++) {
    k = (i%blocks_per_row)*blockwidth + (j%blockwidth) +
        ( (i/blocks_per_row)*blockheight + (j/blockwidth) ) * cols;
    blocked_image[i][j] = raster_image[k];
  }
  if (fwrite(blocked_image[i], sizeof(DATA), vector_length, outputfile)
        != vector_length) {
    fprintf(stderr,"%s: %s: %s\n",programname,outputname,NOWRITE);
    exit(13);
  }
}
```

5.2. Prediction Code

Prediction code is the block of code written in matlab and is used to create an initial prediction based training set or a prediction based sample image. The prediction may be one two or three dimensional where increasing dimension result in increase in performance. Prediction is block based instead of pixel. The predicted training set is provided to Encoder/Decoder for quantization.

```
size=6144*512;
fid=fopen('blk_training_set.512','rb');
fid1=fopen('a1.512','wb');
fid2=fopen('a2.512','wb');

input=fread(fid,[1,size],'uchar');
input=double(input);

R11=zeros(8,8);
R12=zeros(8,8);

R21=zeros(8,8);
R22=zeros(8,8);

R10=zeros(8,8);
R20=zeros(8,8);
```

```
RR(1:8,1:8)=R10(:,:);
RR(9:16,1:8)=R20(:,:);

inverse=inv(RL);
A=inverse*RR

for i=1:8
 fwrite(fid1,A(1:8,i),'float');  % check this
 fwrite(fid2,A(9:16,i),'float');
end
 fclose(fid);
 fclose(fid1);
 fclose(fid2);
```

5.3.Encoder/Decoder

Encoder/Decoder is based on the vector quantization technique called Entropy constrained Reflected Residual Predictive vector Quantization. The module first time takes the input in the form of predictive training set as open loop. Then afterward work is done asymptotically and prediction is done similar to the closed loop i.e. prediction module is embedded in the Encoder/Decoder module and this module creates input for the next phase or cycle of Encoder/Decoder.
The Encoder/Decoder functionality is achieved using a set of functions.

5.3.1. parse_command_line()

The function parse command takes input from user on command line and extract information training set name, code book sequence, no of vectors per stage etc and make available this information for rest of the program.

```
parse_command_line( argc, argv, ts_file, nvps_file, cbk_file);
```

5.3.2. get_nvps( )

The function gets number of the vectors per stage of the Encoder and Decoder as provided by user.

```
get_nvps(nvps_file);
```

5.3.3. get_ts()

The function is used to get training set from disk as specified by the user. Fuction uses file handling techniques to read the file and get training set.

```
get_ts(ts_file);
```

5.3.4. allocate_memory()

The Function is dynamically used to allocate memory for the storage of training set and other temperary results during the Encoder/Decoder functionallity.

allocate_memory();

5.3.5. get_cbks();

The Function get_cbk is used to get previously created codebooks so that code vectors can be used for decoding.

get_cbks(cbk_file);

5.3.6. mwrite_cbks();

The function mwrite_cbks() is to write codebooks after each Encoding stage so that these can be used for next stages.

mwrite_cbks(ts_file,cbk_file,0);

5.3.7. read_or_write_tables()

The Function read_or_write_tables() is used to

read_or_write_tables(0);

5.3.8. rvq_entropy_encode()

The Function rvq_entropy_encode() is used to simulate the functionality of the Encoder based on EC-RRVQ.

rvq_entropy_encode(in_lambda,start_index);

5.3.9. minimize_lagrangian()

The function minimize_lagrangian() is used to minimize the Lagrangian.

minimize_lagrangian(l,ts_file,index)

5.3.10. read_or_write_mid(index)

The function read_or_write_mid(index) is used to read midpoints of previous stage or is used to write midpoint of current stage so that they can be utilized at later stages.

read_or_write_mid(index)

5.3.11. rvq_entropy_decode()

This function entropy decodes the image.

rvq_entropy_decode()

5.3.12. rvq_entropy_decode3()

This function is used to Encode and then decode an image using the results or codebook of training set.
rvq_entropy_decode3()

5.3.13. Encoding portion

The main portion of the module is Encoder and Decoder. The code for Encoder is as follows

```
Initialize tppsizes ans disps arrays*/
 for(i=0;i<MAXNUMSTAGES;++i) disps1[i]=disps2[i]=tppsizes1[i]=tppsizes2[i]=0;

 /* Determine mid_point displacement for each stage*/
 tmid_size=0;
 for(n=0;n<min(order,num_primed_stages);n++){
  for(i=1,j=0;j<n;j++) i*=nvps[j];
  tmid_size+= (i*vs);
  mid_disps[n]=tmid_size;
 }
 for(k=order;k<num_primed_stages;k++){
  for(i=1,l=k-order;l<=k;l++) i*=nvps[l];
  tmid_size+= (i*vs);
  mid_disps[k]=tmid_size;
 }

 /* Determine length displacement for each stage*/
 tratio_size=0;
 for(n=0;n<min(order,num_primed_stages);n++){
  for(i=1,j=0;j<n;j++) i*=nvps[j];
  tratio_size+= (i*1);
```

```
  ratio_disps[n]=tratio_size;
 }
 for(k=order;k<num_primed_stages;k++){
  for(i=1,l=k-order;l<=k;l++) i*=nvps[l];
  tratio_size+= (i*1);
  ratio_disps[k]=tratio_size;
 }

 /*******Determine last stage flag displacement*******/
  tlast_stage_flag_size=0;
 for(n=0;n<min(order,num_primed_stages);n++){
  for(i=1,j=0;j<n;j++) i*=nvps[j];
  tlast_stage_flag_size+= (i*1);
  last_stage_flag_disps[n]=tlast_stage_flag_size;
 }
 for(k=order;k<num_primed_stages;k++){
  for(i=1,l=k-order;l<=k;l++) i*=nvps[l];
  tlast_stage_flag_size+= (i*1);
  last_stage_flag_disps[k]=tlast_stage_flag_size;
 }



 j=0;
 /*Determine size of tpp1 and tpp2 arrays*/
 for(tpp1_size=0,i=1,j=0,k=0;k<min(order,num_primed_stages);++k){
   i*=nvps[k];tppsizes1[k]=i;j+=i;disps1[k]=j;
 }/*k*/
 tpp1_size+=j;
 for(k=order;k<num_primed_stages;++k){
   for(i=1,l=k-order;l<=k;++l) i*=nvps[l];
   tppsizes1[k]=i;tpp1_size+=i;disps1[k]=tpp1_size;
 }/*k*/


 if((order>0)&&(order<num_primed_stages-1)){
  for(tpp2_size=0,k=order+1;k<num_primed_stages;++k){
    for(i=1,l=k-order;l<k;++l) i*=nvps[l];
    tppsizes2[k-1]=i;tpp2_size+=i;disps2[k-1]=tpp2_size;
  }/*k*/
 }/*if*/

 /*Allocate memory to hold old probabilities while determining new ones*/
 tpp1= (float *)calloc(sizeof(float),tpp1_size);
 if(tpp1==0){
   perror("Cannot allocate memory for probabilities: ");
   exit(-1);
 }/*if*/

 /*Allocate memory to hold old probabilities while determining new ones*/
 ttpp1= (float *)calloc(sizeof(float),tpp1_size);
 if(ttpp1==0){
   perror("Cannot allocate memory for probabilities: ");
   exit(-1);
 }
```

```
if((order>0)&&(order<(num_primed_stages-1))){
  tpp2= (float *)calloc(sizeof(float),tpp2_size);
 if(tpp2==0){
    perror("Cannot allocate memory for probabilities: ");
    exit(-1);
  }/*if*/
}/*if*/

/*Initialize the tpp1 and tpp2 arrays*/
 for(i=0;i<tpp1_size;++i) {*(tpp1+i)=0.0;*(ttpp1+i)=0.0;}
 if((order>0)&&(order<(num_primed_stages-1)))
   for(i=0;i<tpp2_size;++i) *(tpp2+i)=0.0;

/*Store old probabilities*/
  for(i=0;i<tpp1_size;++i) *(ttpp1+i)=*(pp1+i);

/*Compute conditional probabilities*/
if((order>0)&&(order<(num_primed_stages-1))){
for(i=order+1;i<num_primed_stages;++i){
  for(k=0;k<nvps[i];++k)
    for(j=0;j<tppsizes2[i-1];++j){
      if(*(pp2+disps2[i-2]+j)==0.0)
        *(pp1+disps1[i-1]+j+tppsizes2[i-1]*k)=0.0;
      else
        *(pp1+disps1[i-1]+j+tppsizes2[i-1]*k)/=*(pp2+disps2[i-2]+j);
      }/*j*/
}/*i*/
}/*if*/


for(i=0; i<num_ts_vecs; i++)
  flag[i]=200;

for(i=0; i<num_ts_vecs; i++){

/***initialize****/
  vrs[i]=num_primed_stages-1;

/*Initialize es*/
  es=0;

/*Initialize causal residual vector to be source vector.*/
  char_ptr = ts_buf + i*vs;

/*Initialize index buffers */
  for(j=0;j<(MAXNPATHS*MAXNVPS);++j){
    *(vi+j)=0;
    *(ci+j)=0;
    *(pi+j)=0.0;
  }/*j*/


/*Initialize the sp buffer*/
  for(j=0;j<(num_primed_stages*MAXNPATHS);++j) *(sp+j)=0.0;
```

```
/*Clear the spaths buffer*/
  for(j=0;j<num_primed_stages*MAXNPATHS;++j) *(spaths+j)=0;

  for(k=0;k<vs;k++) fvec[k]= (float)*char_ptr++;


dtable(i,0,&zero,&one,fvec,1,cb[0],nvps[0],ci,vi,pi,lagrangian,distortion,lambda,&zero,&zero,&zero);

for(m=0;m<min(npaths[0],nvps[0]);++m) {
   *(tci+m)=*(ci+m);
   *(tvi+m)=*(vi+m);
   *(sp+m)=*(pi+m);
   *(spaths+m)=*(ci+m);
}/*m*/

if(vrs[i]==0){
  if(*pi!=0.0)
    dt3+=-log((double)*pi)/log(2.0);
}/*if*/

/*for(m=0;m<min(npaths[0],nvps[0]);++m)
  subvecs(fvec,cb[0]+(*(tci+m))*vs,fbuf+m*vs);*/

/*Initialize n*/
  n=nvps[0];

/*Encode through primed stages.*/
for(stage=1;stage<=vrs[i];stage++){

/*Compare with each code vector.*/

dtable(i,stage,disps1+stage-1,tppsizes1+stage-1,fvec,min(npaths[stage-
1],n),cb[stage],nvps[stage],ci,vi,pi,lagrangian,distortion,lambda,mid_disps+stage-1,ratio_disps+stage-
1,last_stage_flag_disps+stage-1);

/*Update n*/
if(n<MAXNPATHS) n*=nvps[stage];

/*Keep the best npaths*/

 for(m=0;m<min(npaths[stage],n);++m){
    *(tci+stage*MAXNPATHS+m)=*(ci+m);
    *(tvi+stage*MAXNPATHS+m)=*(vi+m);
    *(sp+stage*MAXNPATHS+m)=*(pi+m);
 }/*m*/



 if(l!= 0.0){

   /***** here I make the change to determine the last stage ****/


          if(stage==0) {if(*(last_stage_flag)==1)  vrs[i]=stage;}
    else if(stage<=order){
```

```
    for(t1=1,t2=0,k=0;k<stage;++k){
      t2+=((int)*(tci+stage))*t1;
      t1*=nvps[k];
    }
    if(*(last_stage_flag+last_stage_flag_disps[stage-1]+t2+t1*0)==1) vrs[i]=stage;
  }
  else{
    for(t1=1,t2=0,k=stage-order;k<stage;++k){
      t2+=((int)*(tci+stage))*t1;
      t1*=nvps[k];
    }
    if(*(last_stage_flag+last_stage_flag_disps[stage-1]+t2+t1*0)==1) vrs[i]=stage;
  }

        /*****************************************************************/
}

if(l!= 0.0 && stage==vrs[i]){

  /***** here I make the change to determine the last stage ****/



        if(stage==0) {if(vrs[i]==stage) *(last_stage_flag)=1;}
  else if(stage<=order){
    for(t1=1,t2=0,k=0;k<stage;++k){
      t2+=((int)*(tci+stage))*t1;
      t1*=nvps[k];
    }
    if(vrs[i]==stage) *(last_stage_flag+last_stage_flag_disps[stage-1]+t2+t1*0)=1;
  }
  else{
    for(t1=1,t2=0,k=stage-order;k<stage;++k){
      t2+=((int)*(tci+stage))*t1;
      t1*=nvps[k];
    }
    if(vrs[i]==stage) *(last_stage_flag+last_stage_flag_disps[stage-1]+t2+t1*0)=1;
  }

        /*****************************************************************/
}



if(stage==vrs[i]){
 if(*pi!=0.0) dt3+=-log((double)*pi)/log(2.0);
}/*if*/

es=stage;

if(stage!=vrs[i]){

 /*Find the best M-paths by tracing back the RVQ structure*/
 for(m=0;m<min(npaths[stage],n);++m){
   *(spaths+stage*MAXNPATHS+m)=*(tci+stage*MAXNPATHS+m);
```

```
    trace[0][m]=*(tvi+stage*MAXNPATHS+m);
    for(j=0,k=stage-1;k>=0;--k){
      *(spaths+k*MAXNPATHS+m)=*(tci+k*MAXNPATHS+trace[j][m]);
      trace[j+1][m]=*(tvi+k*MAXNPATHS+trace[j][m]);
      ++j;
    }/*k*/
  }/*m*/


}/*if*/

}/*stage*/



FINE:;
/*Calculate average distortion as we encode.*/
 new_cum_dist += *distortion;

/*Find best path by tracing back the RVQ structure*/
*(P_tuple+i*num_stages+vrs[i])=*(tci+vrs[i]*MAXNPATHS);
tcc[vrs[i]][*(tci+vrs[i]*MAXNPATHS)]+=1;
n=0;
trace[n][0]=*(tvi+es*MAXNPATHS);
for(stage=vrs[i]-1;stage>=0;--stage){
  best_index=*(tci+stage*MAXNPATHS+trace[n][0]);
  trace[n+1][0]=*(tvi+stage*MAXNPATHS+trace[n][0]);
  *(P_tuple+i*num_stages+stage)=best_index;
  tcc[stage][best_index]+=1;
  ++n;
}/*stage*/

/* Compute the probabilities of all paths */

for(inc=0,t1=1,t2=0,j=0;j<min(order,vrs[i]+1);++j){
   t2+=*(P_tuple+i*num_stages+j)*t1;
   *(tpp1+inc+t2)+=unit; t1*=nvps[j];inc+=t1;
}/*j*/
for(k=order;k<=vrs[i];++k){
  for(t1=1,t2=t3=0,l=k-order;l<=k;++l){
    t2+=*(P_tuple+i*num_stages+l)*t1;
    t1*=nvps[l];t3+=t1;
  }/*l*/
  *(tpp1+inc+t2)+=unit;inc+=t1;
}/*k*/

if((order>0)&&(order<vrs[i])){
for(inc=0,k=order+1;k<(vrs[i]+1);++k){
  for(t1=1,t2=t3=0,l=k-order;l<k;++l){
    t2+=*(P_tuple+i*num_stages+l)*t1;
    t1*=nvps[l];t3+=t1;
  }/*l*/
  *(tpp2+inc+t2)+=unit;inc+=t1;
}/*k*/
}/*if*/
```

Chapter 6
Results

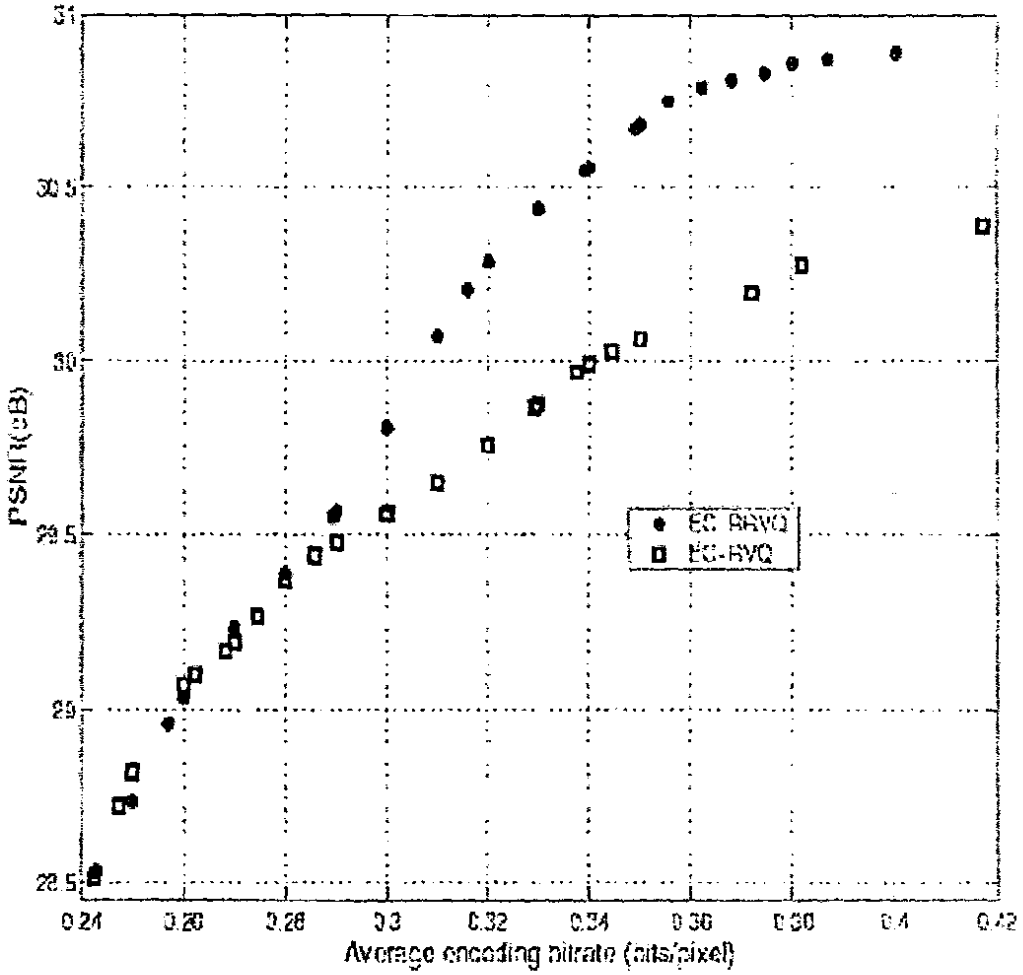


Fig 6.1. Rate-distortion performance of EC-RRVQ and EC-RVQ with 32 stages for test image LENA at m=1. The vector size is 8 x 8.

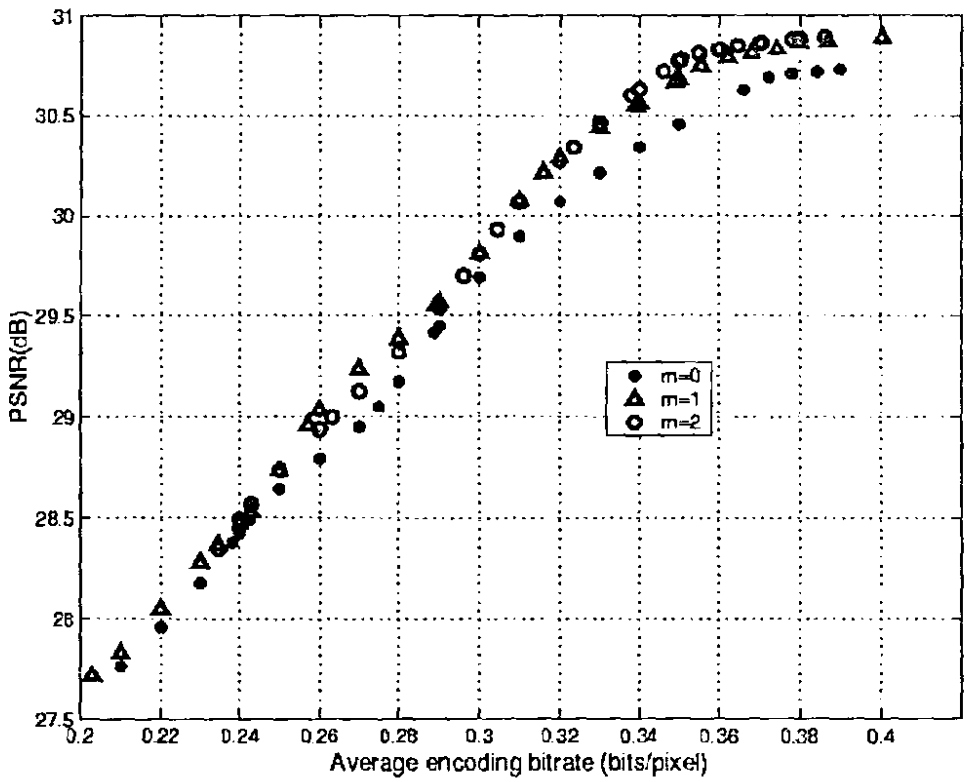


Fig. 6.2. Rate-distortion performance of EC-RRVQ with 32 stages for the test image LENA at increasing values of m. The vector size is 8 x 8.

With increasing value of m performance increases but becomes static at a certain bit rate where all the curves meet each other. Due to less high frequencies image like LENA can be better coded by technique.

Fig. 6.4. Image LENA coded using EC-RVQ at a bit rate of 0.179 bpp with PSNR of 28.03 dB of dimension 8 x 8 and m=1.

Fig. 6.5. Image LENA coded using EC-RRVQ at a bit rate of 0.177 bpp with PSNR of 28.15 dB of dimension 8 x 8 and m=1.

6.1.2. Blocks of size 16 x 16

For the 16 x 16 vector dimension, the training set was composed of 350,000 vectors. The same work as in the previous subsection is done with vectors of size 16 x 16. For the Markov model order experiment, the peak bit rate was 0.25 bpp meaning 64 fixed rate stages were designed. Fig. 6 shows that for rates between
0.1 and 0.2 bpp, there exits a difference of approximately 0.5 dB on average between EC-RRVQ with m=0 and m=1. Also, for the range of rates between 0.1 and 0.2, the difference between EC-RRVQ with m=1 and m=2 is 0.2 dB. In contrast, for rates less than 0.1 bpp, all curves provide similar performance.
A comparison is made between EC-RRVQ and EC-RVQ for the same set of data and dimension of 16 x 16 for m=1. It can be seen from Fig. 7 that for rates grater than 0.12 bpp, the gap between the EC-RRVQ and the EC-RVQ curves is about 0.6 dB on average while for rates less than 0.12 bpp, the two curves will join. For subjective comparison, Fig. 8 and 9 are provided. Fig 8 shows the test image LENA coded using EC-RVQ at a bit rate of 0.215 bpp with PSNR of 28.39 dB of dimension 16 x 16 and m=1. Fig 9 shows the test image LENA coded using EC-RRVQ at a bit rate of 0.201 bpp with PSNR of 29 dB of dimension 16 x 16 and m=1.
A comparison is made between EC-RRVQ and CEC-RVQ for the same set of data and vector dimension of 16 x 16. It can be seen from figure 10 that CEC-RVQ coded image is blocky in nature. The Reason is the fact that CEC-RVQ with 64 residual stages, for the simple fact that it was in need of more than 32 multipaths and a very large training set of data. The EC-RRVQ coded BARBARA image provides no structural Artifacts and image is intact.

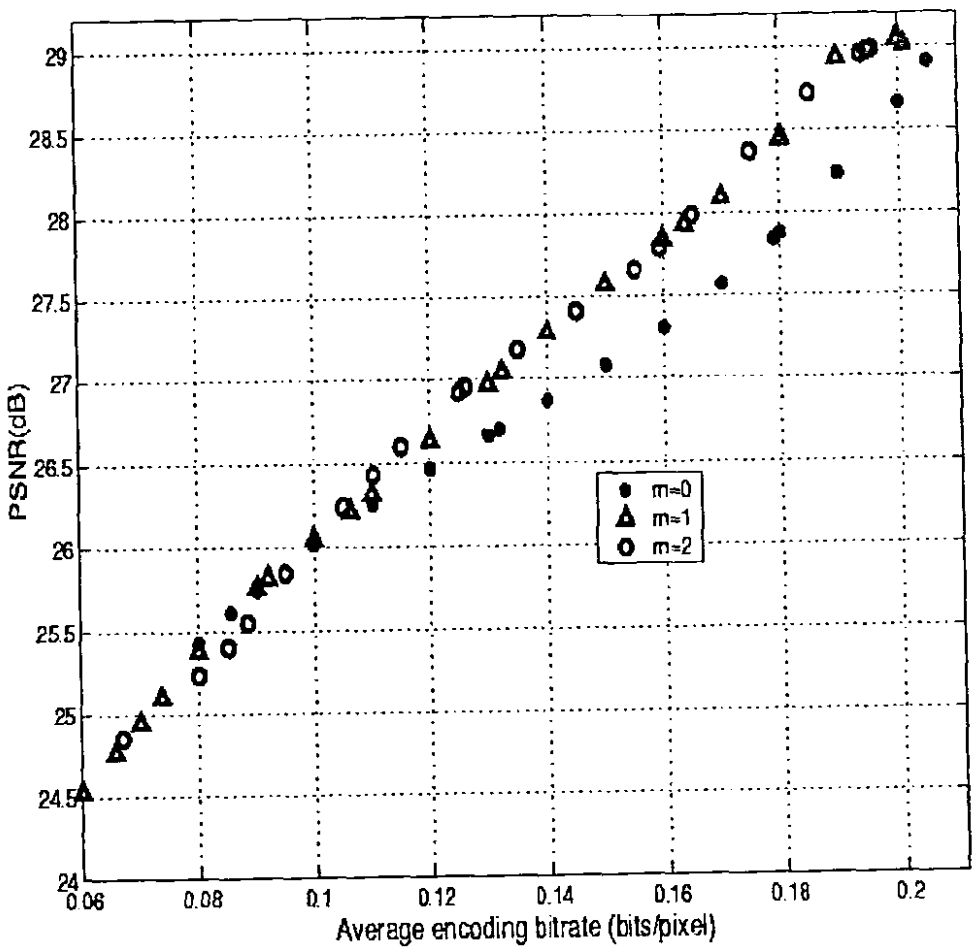


Fig.6. 6. Rate-distortion performance of EC-RRVQ with 64 stages for the test image LENA at increasing values of m. The vector size is 16 x 16.

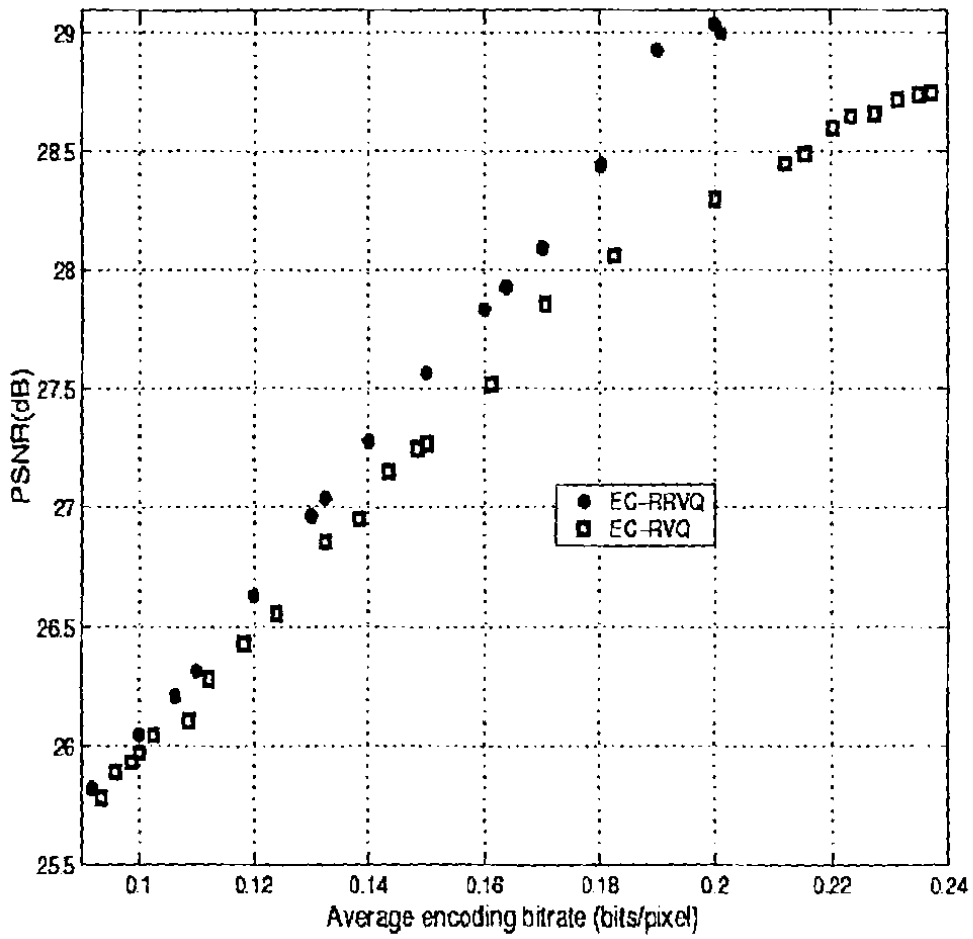


Fig. 6.7. Rate-distortion performance of EC-RRVQ and EC-RVQ With stages for the test image LENA at m=1. The vector size is 16 x 16.
Both the curves are coincided at lower bit rates but at higher bite rates EC-RRVQ outperforms EC-RVQ.

Fig. 6.8. Image LENA coded using EC-RVQ at a bit rate of 0.215 bpp with PSNR of 28.39 dB of dimension 16 x 16 and m=1.

Fig. 6.9. Image LENA coded using EC-RRVQ at a bit rate of 0.201 bpp with PSNR of 29 dB both of dimension 16 x 16 and m=1.

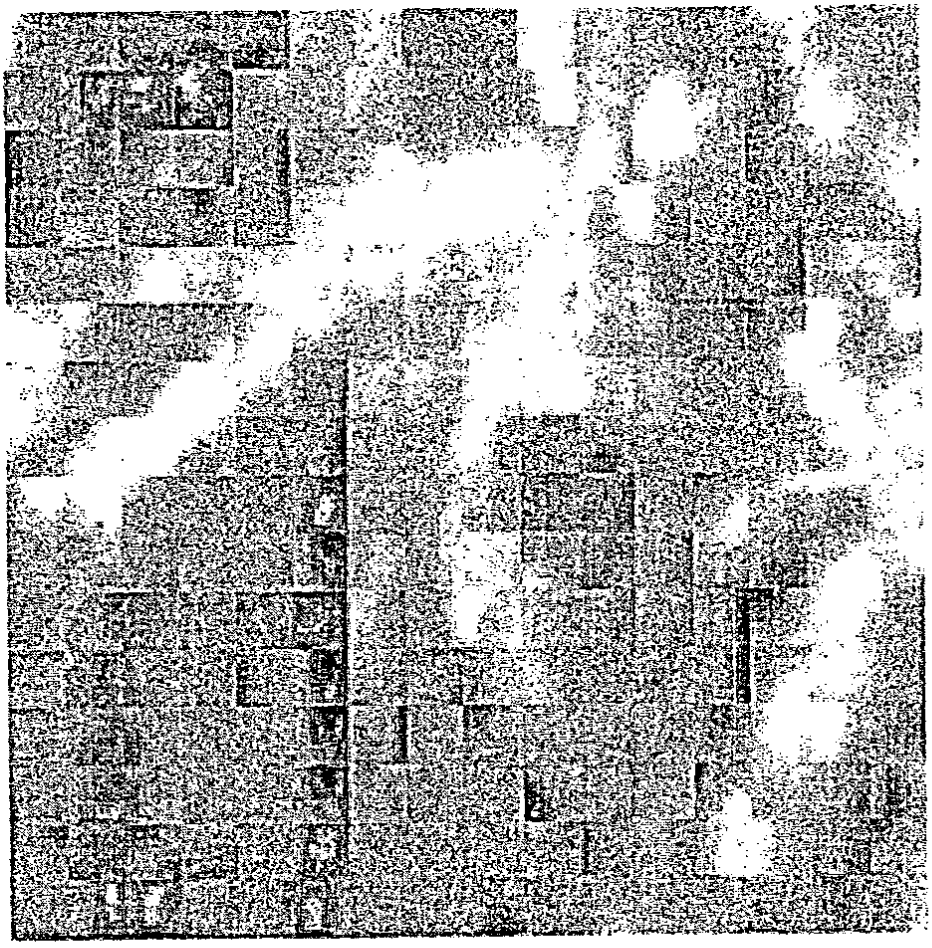
Fig 6.10. Image Barbara coded using 20- path CEC-RVQ at a bit rate of 0.175 bpp with PSNR of 21.17 dB of dimension 16 x 16.

6.1.3. Blocks of size 32 x 32

Here we report on the experiments conducted for designing 32 x 32 dimension EC-RRVQ design. Since the expense of design effort increases linearly with increasing residual stages, the EC-RRVQ design for very large dimension VQ was only feasible EC-RRVQ structure. However we are not able to design CEC-RVQ for 32 x 32 dimension with 128 residual stages, for the simple fact that it was not in need of more than 32 multipaths and a very large training set data. For the 32 x 32 dimension EC-RRVQ system design the training data of 400,000 32 x 32 vectors seems adequate for the reason that it is stage by stage optimization with no inter-stage dependencies. This characteristic is very desirable otherwise one may never design very large dimensional VQ due to insufficient training set data.
Shown in figures 6.11, 6.12, 6.13, 6.14 is the subjective quality comparison of BARBARA image coded at various bit rates using 32 x 32 dimension EC-RRVQ wit h 128 binary stages. The coding results describe the fact that BARBARA image coded at very low bit rates like 0.02 and 0.01 is still recognizable.

Subjective quality comparison for 32 x 32 dimensions EC-RRVQ
Fig 6.11. Shows the original BARBARA image.

Subjective quality comparison for 32 x 32 dimensions EC-RRVQ
Fig .6.13. Shows the coded BARBARA image at 0.02 bpp with PSNR 20.86 dB using EC-RRVQ of dimensions 32 x 32.



Subjective quality comparison for 32 x 32 dimensions EC-RRVQ
Fig 6.14. Shows the coded BARBARA image at 0.01 bpp with PSNR 19.50 dB using EC-RRVQ of dimensions 32 x 32.

6.2. SIMULATION RESULTS for PEC-RRVQ

In this section we compare the performance of PRRVQ with unpredictive RRVQ with an entropy-constraint. The training set for an 8 x 8 vector dimension contained no more than 500,000 vectors and 32 fixed rate RRVQ stages were designed giving 0.05 bpp as a peak bit rate. First experiments were performed for obtaining satisfactory performance as a function of Markov model order. Fig. 15 shows that for rate below 0.1 bpp there seems to be no difference among m=0, m=1 and m=2 Entropy-constrained-RRVQ (EC-RRVQ) curves. However, third markov model order m=3 for EC-RRVQ has an edge of about 0.05 dB for rates above 0.15 bpp. Also, the Fig. 15 provides comparison between Predictive Entropy-constrained RRVQ (PEC-RRVQ) with unpredictive EC-RRVQ. We observe that for rates between 0.05 and 0.15, the first-order PEC-RRVQ provides only a slight improvement of about 0.2 dB difference with that of EC-RRVQ with m=3. The second-order PEC-RRVQ emerges as the most successful predictive quantizer design. The Fig. 6.7 shows that the second-order PEC-RRVQ outperforms both first-order PEC-RRVQ and unpredictive EC-RRVQ. Specially, in the middle region for rates between 0.05 and 0.2 bpp, the second-order PEC-RRVQ (PRRVQ-2) provides about 0.05 dB improvements.

The Fig. 6.16 reveals the visual quality obtained for BARBARA image coded at dimension 8 x 8 with Predictive EC-RRVQ and 14-path conditional EC-RVQ employing 32 binary stages. The conditional entropy-constrained residual vector quantization (CEC-RVQ) was earlier proposed in [12].

The CEC-RVQ makes use of conditioning model to extract linear and non-linear correlations present in an image and has been considered as one of the most successful means of providing improved rate-distortion performance at low bit-rates. The comparison of rate-distortion results does not show a big difference. While examining the coded images we observe that high-frequency texture present on the tablecloth and on trouser of BARBARA image is well preserved in conditional EC-RVQ coding. However, the low-frequency portion of the image like the books and face is adequately reconstructed by PEC-RRVQ coding. This is due to the fact that high-frequency texture is highly non-linear in nature and cannot be predicted by a linear prediction model as employed in PEC-RRVQ. Therefore, to code images with texture, we need to employ non-predictive EC-RRVQ but at a higher dimension to get effective coding with minimum complexity.

Proceedings

2004 International Networking and Communication Conference

INCC 2004

June 11 – 13, 2004

Pakistan

Edited by
Shahab M Baqai
Ishtiaq A Bhatti

Organized by
Lahore University of Management Sciences, Pakistan

Sponsored by
Pakistan Telecommunication Company (PTCL), Pakistan

IEEE COMMUNICATIONS SOCIETY

Fig 6.17. Image Barbara coded using PEC-RRVQ at a bit rate of 0.282 bpp with PSNR of 24.61 dB, of dimension 8 x 8.

References

7. REFERENCES

[1] C.E Shannon "A mathematical theory of communication", Bell Systems Technical Journal 27, pp.379-423.

[2] M. J. Sabin and R.M Gray, " Product Code vector Quantizer for waveform and voice coding" IEEE Transactions on Acoustics, speech and signal processing ASSP –32(3).

[3] B. Ramamurthi and A. Gresho, "Classified vector Quantization on images" IEEE Transactions on Communications.

[4] J. H Conway and N. J. A. Sloane, " Fast Quantizing and decoding Algorithms for lattice quantizers and codes" IEEE Transactions on Information Theory.

[5] K. Sayood, J. D. Gibson and M. C. Rost, "An Algorithm for uniform vector Quantization Design" IEEE Transactions on Information Theory.

[6] A. Buzo, A. H. Gray, Jr., R.M. Gray, J.D. Markel, "speech coding based on Vector Quantiztion", IEEE Transactions on Acoustics, speech and signal processing ASSP – 28(5).

[7] R.M. Gray, "Vector Quantization " IEEE ASSP Magazine, pp.4-20.
[8] B.H. Juang and A.H. Gray, "Multiple stage quantization for speech coding" In proceedings of the IEEE International Conference on Acoustics, speech and signal processing.

[9] J.Makhoul, S. Roucos, and H.Gish, " Vector Quantization in speech coding". Proceedings of the IEEE 73(11) pp.1551-1581, Nov. 1985

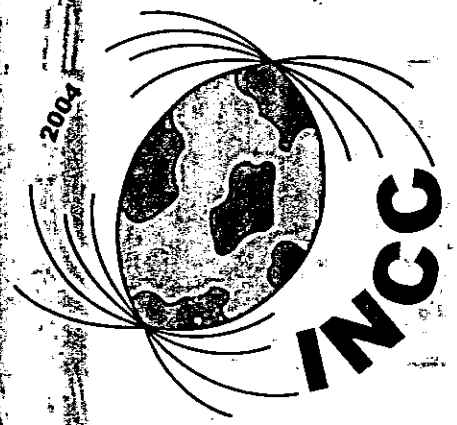
[10] B.Hammer, A.V. Brandt, and M. Schiein, "Hierarical encoding of image sequences using multistage vector Quantization ", in proceedings of the IEEE Conference on Acoustics, speech and signal processing.
[11] www.Autosopy.com/Image and Video Compression Techniques.htm
[12] Digital image processing by Raefel Gonzales
[13] M. A. U. Khan and W. A. H. Mousa, Desin and Analysis of entropy-constrained reflected residual vector quantization, la Proceedings of IEEE.
[14] Gao W., Compression Techniques for Multi-media Data, Electronics Industry Publishing House, Beijing, 1994.
[15] I2Draft revised recommendation h.261-video code for audiovisual service at 64kbits/s, li Tech. Rep., CCITT, 1981.
[16] I3Mpeg video committee draft, le Tech. Rep., MPEG Video CD Editorial Committee, 1990.
[17] Chen k. and Ramabadran T. V., i.Near-lossless compression of medical image trough entropy-coded dpcm. Is IEEE Trans. Medical Image, pp. 538Œ548, 1994.
[18] Hosam Khalil and Kenneth Rose, Predictive vector quantizer design using deterministic annealing, li IEEE 2003.

[19] V. Cuperman and A. Gersho, ieVector predictive coding of speech at 16kbits/s, Ir IEEE Trans. Commun. vol. COM-33, pp. 685Œ696, July 1985.
[20] P.C. Chang and R. M. Gray, Gradiant algorithm for designing predictive vector quantizers) IEEE Trans. Acoust. Speech, Signal Processing, vol. ASSP-34, pp. 679Œ690. Aug 1986.
[21] H. Khalil, K. Rose, and S. L. Regunathan, I. Asymptotic closed-loop approach to predictive vector quantizer design with application in video coding, I IEEE Trans. on Image Processing, vol. 10, pp. 15Œ23, Jan 2001.
[22] J. Pan and T.R. Fischer, Two-stage vector quantization-lattice vector quantization,la IEEE trans. on info. Theory, vol. 41, no. 1, pp. 155Œ163, Jan. 1995.
[23] C. F. Barnes, Residual Quantizers, Ph.D. thesis, Brigham Young University, Provo, Utah, 1989.
[24] M. A. U. Khan and W. A. H. Mousa, Image coding using entropy-constrained reflected residual vector quantization, la Proceedings of IEEE Int. Conf. On Image Processing (ICIP), vol. I, pp. 253Œ256, Sept.
2002.
[25] F. Kossentini, W. C. Chung, and M. J. T. Smith, i. Conditional entropy constrained residual vq with application to image coding,In IEEE Trans. Image Processing, vol. 5, pp. 311Œ320, Feb 1996.
[26] Y. Linde, A. Buzo, R. M. Gray, "An algorithm for vector quantizer design " IEEE Transactions on Communications
[27] R. L. Baker, Vector Quantization of Digital images, Stanford University.
[28] G.Gabor and Z.Gyorfi, Recursive source coding, Springer Verlag. NewYork.

Research Publication

IEEE

IEEE COMMUNICATIONS SOCIETY



INCC 2004

International Networking and Communications Conference

11th - 13th JUNE

Lahore, Pakistan

Organized by:

LUMS

Lahore University of Management Sciences

2004 International Networking and Communication Conference

INCC 2004

Table of Contents

Technical Sessions

Saturday, June 12, 2004, Session I – Ad hoc Networks

Saturday, June 12, 2004, Session II – Network Security

Saturday, June 12, 2004, Session III – Communications

Saturday, June 12, 2004, Session IV – Policy & Regulation

Sunday, June 13, 2004, Session I – Quality of Service

Sunday, June 13, 2004, Session II – Optical Communications

Sunday, June 13, 2004, Session III – Signal Processing

Sunday, June 13, 2004, Session IV – Miscellaneous

Design and Analysis of Predictive Reflected Residual Vector Quantization

Mohammad A. U. Khan[1], Ch. M. Imran[2], M. Shoaib[2], M. Sikander H. Khiyal[2]
[1] Department of Electrical Engineering, COMSATS Institute of Information Technology, Sector H-8/1, Islamabad.
mohammad_a_khan@yahoo.com
[2] Department of Computer Science, International Islamic University, Islamabad, Pakistan.
imran_chuadry@hotmail.com

Abstract - **Image communications is primarily constrained due to its large bandwidth requirements. Therefore, researchers worked on various compression algorithm to achieve low bit rate. It was stated that images and video sequences are highly-correlated sources and their correlation should be exploited in a given compression algorithm Differential pulse code modulation (DPCM) has emerged as a mean of exploiting the correlation among the image pixels. Later on, DPCM was improved upon by predictive vector quantization (PVQ). PVQ employs block by block prediction and results in satisfactory performance at low bit rates. However, its design is complicated and recently an asymptotic closed-loop (ACL) was proposed to stabilize the design. In this paper, we attempted to replace the VQ with a multistage VQ structure in a hope to further reduce the stress on the closed-loop design. The multistage VQ structure that we employed is commonly referred to as reflected residual vector quantization (RRVQ). RRVQ works by imposing an additional symmetry constraint on the multistage codebook design. RRVQ has been quite popular where large block-length vector quantizations is needed due to their very low codebook search capability. Our proposed design goal in replacing VQ with RRVQ in a PVQ design is our wish to use large block length like 16×16 or 32×32 size vectors to grab any linear/non-linear correlation among the vector components. The way to incorporate RRVQ within PVQ structure has been proposed and simulation results are discussed.**

I. Introduction

Compression of digital images and video signals to reduce their storage and transmission bandwidth requirements is of great interest in the implementation of communication systems. Low bit rates, particularly in the range of 16-32 kbits/s, are gaining an importance due to the Internet and wireless mobile communications. Digital images and video signals exhibit large temporal correlations that can be exploited in a compression algorithm to bring down the bandwidth requirements. There are several compression algorithms reported in the literature for this purpose. One such classical algorithm is differential pulse code modulation (DPCM). The basic idea behind DPCM scheme is to predict the value of a current input pixel based on neighboring pixel values, using certain *prediction coefficients*. The difference between the predicted value and the actual value of the pixels emerge as a differential or residual image, which is much less correlated than the original image. The differential image is then entropy coded and sent. The schematic diagram of the DPCM coder/decoder is shown in Fig. 1. The Fig. 2 displays the original Cameraman image and its differential image. The differential image exhibits variance in the range of 42 as oppose to 433 for original Cameraman image. This illustrates that DPCM has been very effective in reducing spread of the pixel values by almost 10:1 ratio. This reduction is very helpful in providing compression. Due to its simple structure DPCM has made its place in the standard algorithms like JPEG [1] for still image coding, H.261 [2] for videophones and video-conference communications, and MPEG [3] for interactive media applications. However, the performance of DPCM has several drawbacks which prevent its uses in some circumstances. The two main drawbacks are the channel error sensitivity and poor rate-distortion performance at low bit rates[4].

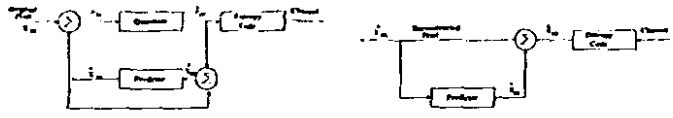
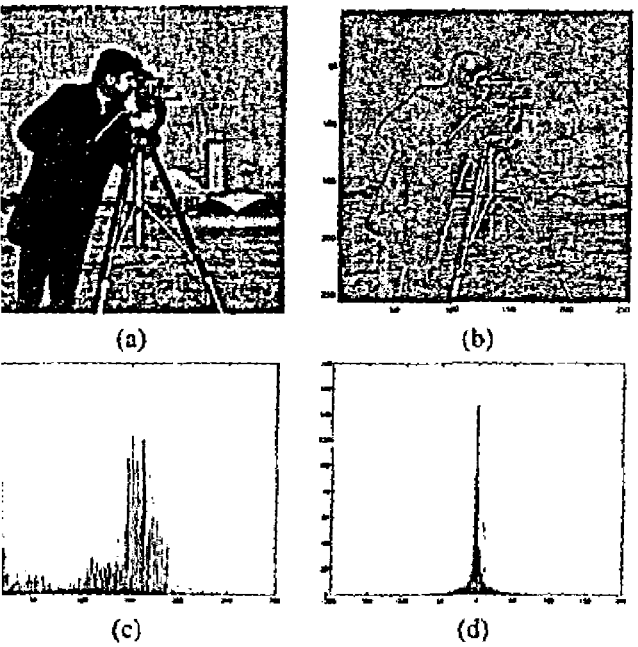


Fig. 1. Basic differential pulse coded modulation (a) encoder, (b) decoder

Another approach that generalizes DPCM is to code prediction residual image using Vector quantization (VQ). Generally, there are several arguments in support of VQ for image and video coding. Shannon rate-distortion theorem indicates that VQ will always perform better than conventional quantization for the simple reason that quantizing block of samples as one unit is a general case of quantizing individual samples in isolation. The DPCM structure that involves VQ as its quantization module is referred to as predictive vector quantization (PVQ). Detailed analysis of PVQ structure is proposed in [5]. It was stated in [5] that although the PVQ structure is simple and well-understood, its design is problematic due to feed-loop involved, and standard methods often fail to produce optimal or even good predictors and quantizers. The design of vector quantizer in a feedback loop is in need of a representative training set of prediction error image.

0-7803-8325-7/04/$20.00 ©2004 IEEE



(a) (b) (c) (d)

(a) Original Cameraman image. (b) The prediction error/ differential
(c) Gray-level histogram of the original image with variance 433.05.
istogram of the prediction error with variance 43.33.

ever, the representative training set has dependence on the predictor and the quantizer. The predictor and the tizer has to be optimized jointly with respect to each in order to come up with a reasonable prediction error e suited for training quantizers.

vo simple approaches for PVQ design were introduced r by Cuperman and Gersho [6]. The first approach red to as *open-loop* solves the vector quantizer design lem by assuming no feedback, and operates directly riginal source vectors. An improvement was suggested sing second approach called *closed-loop* design. In case, an iterative design is employed for updating raining set and the quantizer given a fixed predictor. on, closed-loop design algorithm was further modified hang and Gray [7], where both vector predictor and r quantizer are jointly optimized. The joint design ithm provides an improvement over the previously l designs however, such design approaches exhibits icant stability problems especially at low bit rates. stability of the closed-loop design is analyzed and a fied design with the name of *asymptotic closed-loop*) algorithm is proposed in [8]. Specifically, the ACL as an open-loop and then tries to simulate closed-loop ior over a longer run.

ctor quantization (VQ), used in PVQ, is a powerful ique for data compression of speech, image, and video s. Vector quantization takes advantage of linear or non- correlations that exist among the vector components. fore larger the vector size used better compression can hieved. However, an issue of recognized importance rge block VQ implementation is that the size of the ook associated with VQ grows exponentially as a function of the product of the vector dimension and bit rate With increase in codebook size comes an overwhelming increase in search complexity. To overcome the complexity barrier, many researchers have suggested imposing certain structural constraints on the VQ code book design. The relief in search complexity was obtained by replacing VQ with a multistage VQ referred to as residual vector quantization (RVQ). The argument is that multistage VQ can be designed sequentially and its stage codebooks are smaller in size thus requires less search complexity. Researchers also considered reducing further the search complexity by applying additional structural constraints on the multistage VQ , and proposed Multiple-stage VQ's with stage code books comprised of lattice VQ's [9] and reflected RVQ (RRVQ) [10]. Our focus in this paper is on the utilization of RRVQ.

A reflected RVQ (RRVQ) is a multistage structure with binary stage code books. The encoder and decoder of RRVQ perform a *reflecting* operations on the residual vectors between stages. This reflecting operation forces a certain symmetry on the resultant RRVQ codebook which in turn makes the sequential search of stage codebook optimal. The sequential search ability of RRVQ makes it an ideal candidate for large block VQ implementations. Recently, an entropy-constrained RRVQ (EC-RRVQ) design was introduced and its simulation results are presented in [11].

In order to extract linear and most of the non-linear correlations among image source, we suggest the use of large-block RRVQ with the current PVQ structure. The purpose of large-block RRVQ is to take advantage of linear/non-linear correlations present among the block of pixels and the predictive structure with the feed-back loop to exploit the remaining intra-block dependencies. Our goal in this paper is to simulate the behavior of RRVQ in a feedback loop that exploits correlation in a given image source in order to produce high compression ratios with lower complexity. A new design algorithm for RRVQ in a predictive environment has been proposed in this paper which is referred to as Predictive reflected residual vector quantization (PRRVQ). The paper is organized as follows. Section II describes reflected residual vector quantization and its generalization to include entropy coding is presented in Section III. Predictive reflected residual vector quantization (PRRVQ) design is proposed in Section IV. Simulation results are discussed in Section V.

II. Reflected RVQ (Ref-RVQ)

A P-stage residual VQ consists of a finite sequence of P vector quantizers $\{(C^p, \mathcal{P}^p);\ 1 \le p \le P\}$. We index the code vectors of *pth*-stage as $\{y_0^p, y_1^p, y_2^p, \cdots, y_{N-1}^p\}$ and Voronoi cells of *pth*-stage as $\{S_0^p, S_1^p, S_2^p, \cdots, S_{N-1}^p\}$. The codevectors comprising the codebook C^p and the cells comprising the partition P^p are indexed with the subscript j^p. The quantized representation $\hat{x}^1$ of the input source vector

x^1 is formed by the sum of the selected stage codevectors,

$$\hat{x}^1 = \sum_{p=1}^{P} y_{j_p}^p, \tag{1}$$

For adopting a jointly optimized approach, an RVQ is represented with an equivalent quantizer, referred to as *direct sum* quantizer $(C^e, \mathcal{P}^e)$. The elements of C^e are the elements of the set of all possible sums of stage codevectors, that is, $C^e = C^1 + C^2 + \cdots + C^P$. C^e can be interpreted as a terminating node of a path through a tree structure, which is associated with the residual quantizer mentioned earlier, corresponding to the direct sum quantizer. A tree structure of three stages, two codevectors/stage RVQ is illustrated in Fig.3. The root node of the tree represents x^1. The leaf nodes represents the set C^e of the direct sum codevectors. The intermediate nodes represent partial sums of the direct sum codevectors and the branches represent stage codevectors. If two codevectors $\{y_0^p, y_1^p\}$ are allowed in a

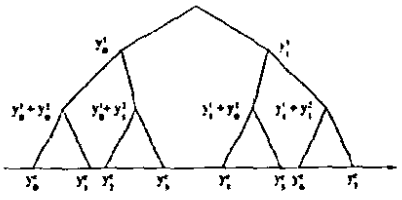


Fig. 3. Three-stage binary RVQ tree structure.

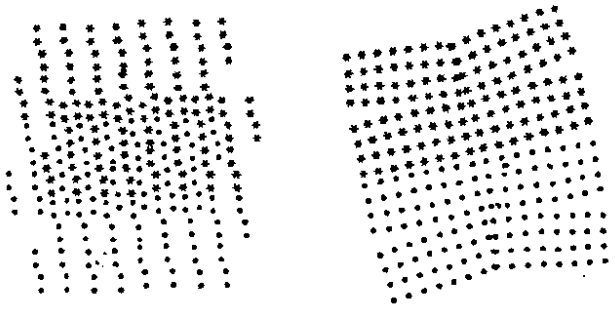
given stage, then the Voronoi boundary is a plane of equal distortion between two codevectors. This boundary can be specified by a midway point m^p between the two given codevectors for that particular stage p as

$$m^p = \frac{1}{2}(y_0^p + y_1^p) \tag{2}$$

The normal vector, n^p, is defined to be the line joining two codevectors $\overrightarrow{y_0^p, y_1^p}$. The equation of the plane through the midway point m^p perpendicular to n^p is

$$n^p . \overrightarrow{m^p z^p} = 0 \tag{3}$$

where z^p is any point in the plane. In order for this hyper-plane to specify also the boundary between adjacent children of the two codevectors, we *reflect* the input vectors of the *pth*-stage to one side of the hyper-plane boundary, and by convention, we reflect all x^p which belong to Voronoi cell S_1^p to second Voronoi cell S_0^p. After reflection, we subtract y_0^p from the reflected input vector forming a *reflected* residual vector. Then, residual vectors that represent the next stage codevectors, will lie in the *reflected residual space*. If we are to unreflect all the reflected stage codebooks, the resulting direct sum codebook has the desired symmetry properties. To illustrate the structure of Ref-RVQ codebook, Fig.II(a) shows the codevector constellation for (8-stages, two codevectors/stage) two dimensional RVQ designed for Gaussian source. Similarly, Fig.II(b) shows the codevector constellation for (8-stages, two codevectors/stage) two dimensional



(a) (b)

Fig. 4. Gaussian source coded with a binary 8-stage, two-dimensional quantizer. (a) Equivalent code vector constellation of RVQ. (b) Code vector constellation of Ref-RVQ.

Ref-RVQ designed for Gaussian source. All the direct sum codevectors that involve first codevector of first RVQ stage in their construction are represented as dots. On the other hand, asterisks are used for the direct sum codevectors constructed with second codevector of first stage RVQ. Fig.II(a) indicates severe codevector diffusion for RVQ, whereas the Ref-RVQ (Fig.II(b)) shows no diffusion and hyper-plane boundaries are evident. The presence of Voronoi cells with hyper-plane boundaries in Ref-RVQ codebook makes the sequential single-path search optimal.

III. Entropy-Constrained Reflected RVQ (EC-Ref-RVQ)

For entropy-constrained design algorithm, the distortion trades off squared error with the codeword rate. In case of residual VQ (RVQ), we will have the Lagrangian

$$J_\lambda = E[d(x^1, C^e)] + \lambda L(C^e), \tag{4}$$

where $L(C^e)$ is the length associated with a direct sum codevector. For the case of developing an entropy-constrained design for the Ref-RVQ, we have binary stages i.e., 2 codevectors/stage. This will form the Voronoi region boundary between the two codevectors as plane. For an EC-Ref-RVQ we need to work with the plane of equal Lagrangian as opposed to the plane of equal distortion used in fixed-rate Ref-RVQ design. We define a plane of equal Lagrangian as

$$\begin{aligned} &\|x^p - y_0^p\|^2 + \lambda L(y_0^p | C^{p-1}, C^{p-2}, \cdots, C^1) \\ &\quad = \|x^p - y_1^p\|^2 + \lambda L(y_1^p | C^{p-1}, C^{p-2}, \cdots, C^1). \end{aligned} \tag{5}$$

We restate the above Eq. as the more *familiar normal plane* equation $n \cdot x^p = d$ as

$$\underbrace{\frac{(y_0^p - y_1^p)}{\|y_0^p - y_1^p\|}}_{n} \cdot x^p = \frac{\|y_0^p\|^2 - \|y_1^p\|^2}{2\|y_0^p - y_1^p\|} + \frac{\lambda(L(y_0^p | C^{p-1}, \cdots, C^1) - L(y_1^p | C^{p-1}, \cdots, C^1))}{2\|y_0^p - y_1^p\|} \tag{6}$$

The shortest distance from y_1^p to the plane will be given by

$$|d| - \boldsymbol{n} \cdot \boldsymbol{y}_1^p. \tag{7}$$

Thus the new midpoint in case of entropy-constrained Ref-RVQ (EC-Ref-RVQ) will be given by

$$m^p = y_1^p + \frac{\|y_0^p - y_1^p\|}{2} n + \frac{\lambda(L(y_0^p|C^{p-1}, \cdots, C^1) - L(y_1^p|C^{p-1}, \cdots, C^1))}{2\|y_0^p - y_1^p\|} n \tag{8}$$

By having a look at Eq. (8) one notice that unlike fixed-rate Ref-RVQ the midpoint for a given stage will not be equidistant from the two codevectors, but will be offset by an amount dependent on the difference of lengths between the two. The midpoint will move in the direction of the larger length code vector because of the third term in the Eq. (8).

An important complexity-reducing feature of EC-Ref-RVQ is its potential to use *stage-conditional* entropy tables of relatively sizes, where conditioning is performed on previous stages. With the use of smaller Markov model order m, a large reduction in entropy-tables storage can be obtained. The length for a direct sum codevector is given by

$$L(C^e) = L(C^0) + L(C^1|C^0) + \cdots + L(C^P|C^{P-1}, C^{P-2}, \cdots, C^0). \tag{9}$$

For a given Markov model order m the above Eq. can be approximated as

$$L(C^e) = L(C^0) + L(C^1|C^0) + \cdots + L(C^P|C^{P-1}, C^{P-2}, \cdots, C^{P-m}), \tag{10}$$

where $P - m >> 0$.

IV. Asymptotic Closed-loop Design

The asymptotic closed-loop design was originally proposed for video coding. In this section we will adopt ACL design for incorporating reflected residual vector quantization in a PVQ structure. The encoding and decoding operation for PRRVQ is essentially same as being used in earlier PVQ implementations. However, the design of RRVQ stage codebooks in a feedback loop needs attention. The design of stage codebooks under ACL approach can be best explained by first introducing some mathematical notations.

Given a set of source vectors, $X : \{x_n\}_{n=0}^N$, the training set of prediction errors at an iteration $i-1$ is generated by

$$e_n^{(i)} = x_n - Pred[\hat{x}_{n-1}^{(i-1)}], \quad n = 1, 2, 3, \cdots, N. \tag{11}$$

The quantizers at iteration $i-1$, are denoted by $Q_1^{i-1}, Q_2^{i-1}, \cdots, Q_P^{i-1}$, where for example the Q_P^{i-1} notation represents Pth stage quantizer at the $i-1$th iteration.. Training set of prediction errors is generated for the next iteration i as , $T_{(i)} = \{e_n^{(i)}\}_{n=1}^N$ where, $e_n(i) = x_n - Pred[\hat{x}_{n-1}^i]$, and reconstruction vectors are produced as $\hat{x}_n^i = Pred[\hat{x}_{n-1}^{(i)}] + [Q_1^{(i-1)}(e_{n,1}^{(i)}) + Q_2^{(i-1)}(e_{n,2}^{(i)}) + \cdots + Q_P^{(i-1)}(e_{n,P}^{(i)})]$.

Having collected the next iteration training vectors we optimize a new set of quantizers, $Q_1^i, Q_2^i, \cdots, Q_P^i$ for the ith iteration. The optimized quantizers are then in turn used to generate the new set of reconstruction vectors,

$$\hat{x}_n^i = Pred[\hat{x}_{n-1}^{(i-1)}] + [Q_1^{(i)}(e_{n,1}^{(i)}) + Q_2^{(i)}(e_{n,2}^{(i)}) + \cdots + Q_P^{(i)}(e_{n,P}^{(i)})]. \tag{12}$$

The ACL employed here is in fact a repetition of three basic steps, that is, first calculate e_n^i for all $n = 1; 2, \cdots, N$, then design the stage quantizers, and finally calculate $\hat{x}_n^{(i)}$ for all $n = 1, 2, 3, \cdots, N$.

Note that the stage quantizers $Q_1^i, Q_2^i, \cdots, Q_P^i$ are used to encode exactly the same prediction error vector used for its design. Neglecting possible problems of local optima, this is the best match for these vectors. We are thus assure that the resulting reconstruction is improved, and this results in better prediction for the next iteration. Subject to the high rate assumption that smaller prediction errors lead to smaller quantization errors, we obtain monotonic convergence through out the process.

It is emphasized that the design is open-loop in nature due to the fact that prediction errors for all elements of the sequence are calculated before quantization. At each iteration, the reconstructed set, on which the prediction for the next iteration will be based, is generated by applying the optimized quantizers and predictor based on the previous fixed set. Since the new reconstructed set will better approximate the original input sequence, the distortion at each iteration is generally decreasing, and we expect the process to converge. At convergence, further iterations do not modify the training set. The quantizer is hence assume to have converged, i.e., $Q_j^{(i+1)} = Q_j^{(i)}$, where $j = 1, 2, 3, \cdots, P$, which immediately ensures that the reconstruction sequence is unchanged, i.e., $\hat{x}_n^{(i+1)} = \hat{x}_n^{(i)}$ as well as the prediction sequence $Pred[\hat{x}_{n-1}^{(i)}] = Pred[\hat{x}_{n-1}^{(i-1)}]$. The procedure is thus open-loop in nature, yet it asymptotically converges to the closed loop performance.

V. Simulation Results

In this section we compare the performance of PRRVQ with unpredictive RRVQ with an entropy-constraint. The training set for an 8×8 vector dimension contained no more than 500,000 vectors and 32 fixed rate RRVQ stages were designed giving 0.05 bpp as a peak bit rate. First experiments were performed for obtaining satisfactory performance as a function of Markov model order. Fig. 7 shows that for rate below 0.1 bpp there seems to be no difference among $m = 0$, $m = 1$ and $m = 2$ Entropy-constrained-RRVQ (EC-RRVQ) curves. However, third markov model order $m = 3$ for EC-RRVQ has an edge of about 0.05 dB for rates above 0.15 bpp. Also, the Fig. 7 provides comparison between Predictive Entropy-constrained RRVQ (PEC-RRVQ) with unpredictive EC-RRVQ. We observe that for rates between 0.05 and 0.15, the first-order PEC-RRVQ provides only a slight improvement of about 0.2 dB difference with that of EC-RRVQ with $m = 3$. The second-order PEC-RRVQ emerges as the most successful predictive quantizer design. The Fig. 7 shows

that the second-order PEC-RRVQ outperforms both first-order PEC-RRVQ and unpredictive EC-RRVQ. Specially, in the middle region for rates between 0.05 and 0.2 bpp, the second-order PEC-RRVQ (PRRVQ-2) provides about 0.05 dB improvement.

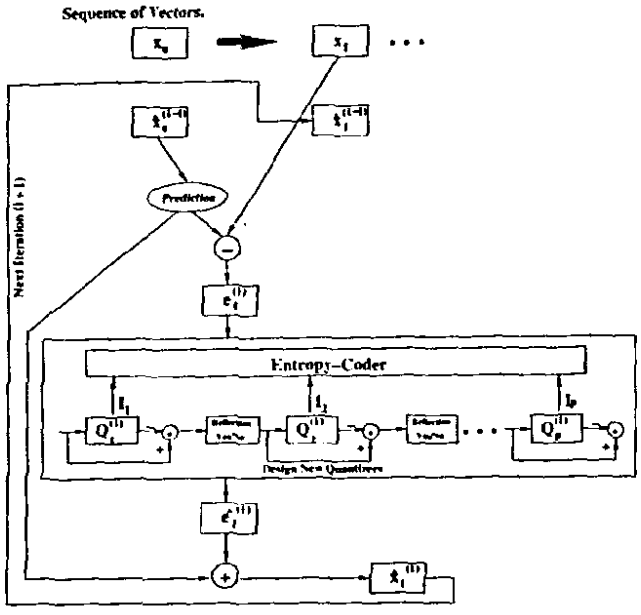
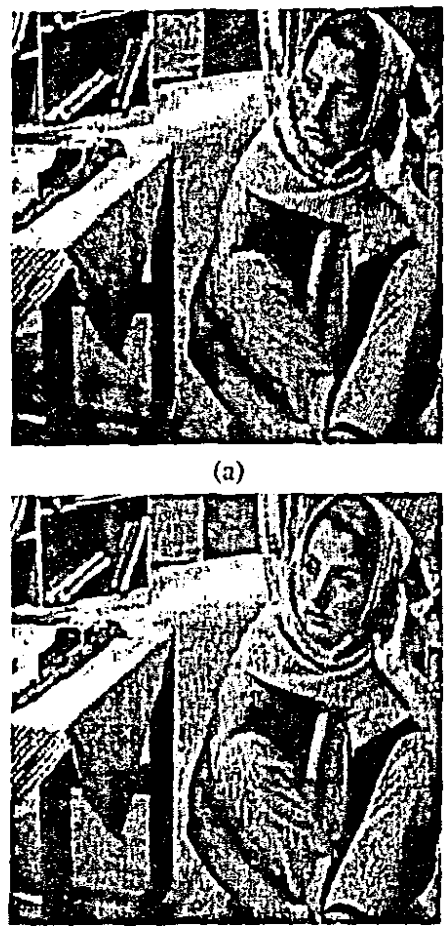


Fig. 5. Proposed asymptotic closed-loop (ACL) procedure for the design of predictive EC-RRVQ (PEC-RRVQ).

The Fig. 6 reveals the visual quality obtained for BARBARA image coded at dimension 8 × 8 with predictive EC-RRVQ and 14-path conditional EC-RVQ employing 32 binary stages. The conditional entropy-constrained residual vector quantization (CEC-RVQ) was earlier proposed in [12]. The CEC-RVQ makes use of conditioning model to extract linear and non-linear correlations present in an image and has been considered as one of the most successful means of providing improved rate-distortion performance at low bit-rates. The comparison of rate-distortion results does not show a big difference. While examining the coded images we observe that high-frequency texture present on the table cloth and on trouser of BARBARA image is well-preserved in conditional EC-RVQ coding. However, the low-frequency portion of the image like the books and face is adequately reconstructed by PEC-RRVQ coding. This is due to the fact that high-frequency texture is highly non-linear in nature and can not be predicted by a linear prediction model as employed in PEC-RRVQ. Therefore, to code images with texture, we need to employ non-predictive EC-RRVQ but at a higher dimension to get effective coding with minimum complexity.

VI. Concluding Remarks

Predictive Reflected residual vector quantization (PRRVQ) has been designed and compared with other well-known predictive algorithms. The strength of PRRVQ lies in its ability to use large vector sizes in a predictive loop and can be designed with smooth convergence behavior. The low side of PRRVQ algorithm is its inability to represent well the images with lot of texture or high-frequency contents.



(a) (b)

Fig. 6. Image Barbara coded using (a) CEC-RVQ at a bit rate of 0.28 bpp with PSNR of 24.54 dB (b) PEC-RRVQ at a bit rate of 0.282 bpp with PSNR of 24.61 dB, both of dimension 8 × 8.