

# Lossless Image Coding using Conditional Entropy-Constrained Vector Quantization

T4205



*Developed By*  
**Kishwar Rani Naushahi**

*Supervised by*  
**Dr. M. Sikander H. Khiyal**

Department Of Computer Science, Faculty of Applied Sciences  
International Islamic University, Islamabad  
(2006)

27/7/00

MS  
621.367  
NAL

1. Image processing - digital techniques

2  
ma

~~MS~~



MS

Accession No TH-4205

KJL

**Department of Computer Science**  
**International Islamic University, Islamabad**

25 Nov 2006

Dated: ~~15 Nov 2006~~

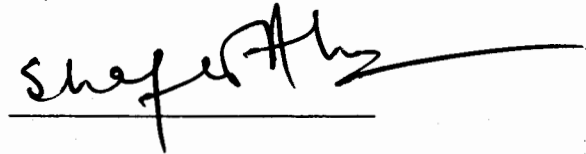
Final Approval

It is certified that we had read the project report submitted by Ms Kishwar Rani Naushahi, Reg no;142-CS/MS/2003 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

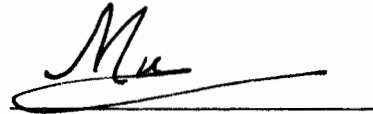
**External Examiner**

Mr. Shaftab Ahmed  
Faculty member  
Bahria University,  
Islamabad.



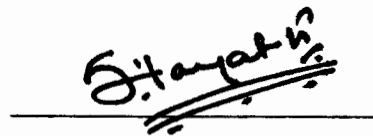
**Internal Examiner**

**Ms Muneera Bano**  
Lecturer  
Department of Computer Science  
Faculty of Applied Sciences,  
International Islamic University, Islamabad



**Supervisor**

Dr. M. Sikander H. Khiyal  
Head of DCS,  
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 requirement for the award of the  
degree of MS in Computer Science.

Dedicated to my Parents  
Whose prayers and attention  
Made me able to reach at this extent

## **Declaration**

We hereby declare that this software, neither as a whole nor as a part there of has been copied out from any source. It is further declared that we have developed this software entirely on the basis of our personal effort 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 degree or qualification of this or any other university or institute of learning.

Kishwar Rani Naushahi

142-CS/MS/2003

## **Acknowledgement**

All praise to Allah Almighty, the most merciful, the most gracious, without his help and blessing, I was unable to complete the project.

Thanks to our project supervisor Dr. M. Sikander H. Khiyal whose sincere efforts helped us to complete my project successfully.

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

Thanks to my family who helped us during our most difficult times and it is due to their unexplainable care and love that I am at this position today.

Kishwar Rani Naushahi

142-CS/MS/2003

## Project in Brief

Project Title	Lossless Image Coding using Conditional Entropy-Constrained Vector Quantization
Objective technique	To Build an lossless compression using vector quantization
Undertaken By	Kishwar Rani Naushahi 142-CS/MS/2003
Supervised By	Dr. M. Sikandar H. Khiyal Head, Department of Computer Science, International Islamic University, Islamabad
Date Started	1 <sup>st</sup> September 2004
Date Completed	31 <sup>st</sup> July 2005
Technology used	GCC compiler on Linux, C language, Matlab
System used	Pentium IV



## Abstract

Most data that is inherently discrete needs to be compressed in such a way that it can be recovered exactly, without any loss. Lossless compression requires that the reproduced reconstituted bit stream be an exact replica of the original bit stream.

Lossless coding guarantees that the decompressed image is absolutely identical to the image before compression. This is an important requirement for some application domains, e.g. medical imaging, where not only high quality is in demand, but unaltered archiving is a legal requirement. Entropy constrained residual vector quantization (EC-RVQ) has been shown to be a competitive compression technique. Its design procedure is an iterative process which typically consists of three steps: encoder update, decoder update, and entropy coder update.

In this research, an entropy constrained vector quantization is proposed for lossless compression of image. The method consists of first quantizing the input image using conditional entropy constrained vector quantizer and then coding the residual images using entropy coder. Experimental results show perfect reconstruction of gray scale images.

## Table of Contents

<b>1. Introduction</b>	1
1.1 Overview	1
1.2 Communication System Model	3
1.3 Compression	4
1.3.1 Lossy/Lossless Coding	4
1.4 Vector Quantization	5
1.4.1 VQ is better than scalar Quantization	6
1.4.2 Why Entropy coding with Quantization?	6
1.4.3 Major Issues in VQ	7
<b>2. Fundamentals</b>	8
2.1 Background	9
2.1.1 Coding redundancy	10
2.1.2 Inter-pixel Redundancy	10
2.1.3 Psycho visual Redundancy	11
2.2 Fidelity Criteria	12
2.3 Image Compression Models	13
2.3.1 The Source Encoder and Decoder	13
2.3.2 The Channel Encoder and Decoder	15
2.4 Image Compression Techniques	15
2.4.1 Lossless Coding	15
2.4.2 Lossy Coding	16
<b>3. Compression Techniques</b>	17
3.1 Lossless coding techniques	18
3.1.1 Run length encoding	18
3.1.2 Huffman encoding	19
3.1.3 Entropy coding (Lempel/Ziv)	20
3.1.4 Area coding	20
3.2 Lossy coding techniques	21
3.2.1 Transform coding (DCT/Wavelets/Gabor)	22
3.2.2 Vector quantization	22
3.2.3 Segmentation and approximation methods	23
3.2.4 Spline approximation methods	24
3.2.5 Fractal coding	25
3.3 Efficiency and quality of different lossy compression techniques	26
3.4 Source Coding	27
3.4.1 Lossy / Lossless Coding	27
3.4.2 Variable Rate	28
3.4.3 Compression Means	29
3.4.4 Performance Limits	29
<b>4. Vector Quantization</b>	31
4.1 Definitions	31
4.2 Optimality conditions	32
4.3 Brief description	32
4.3.1 main advantages	32

4.3.2	main disadvantages .....	33
4.3.3	Practical constraints .....	33
4.4	Quantization .....	33
4.4.1	Fundamental of Vector Quantization .....	34
4.4.2	Vector Quantizer Design .....	35
4.4.2.1	Random Coding .....	35
4.4.2.2	Pruning .....	36
4.4.2.3	Pairwise Nearest Neighbor Design .....	36
4.4.2.4	Product Codes .....	37
4.4.2.5	Splitting .....	38
4.5	The LBG Algorithm .....	38
4.5.1	Basic steps of LBG .....	39
4.5.2	Remarks on LBG .....	39
4.5.3	Initialization in LBG .....	40
<b>5.</b>	<b>Lossless using EC-RVQ .....</b>	<b>41</b>
5.1	Definitions .....	41
5.1.1	Definition of Probability .....	41
5.1.2	Definition of Model .....	41
5.1.3	Definition of Entropy .....	41
5.2	Entropy .....	42
5.2.1	Memory-less Entropy .....	42
5.2.2	High-Order Entropy .....	42
5.3	Codebook Design .....	43
5.4	VQ using Entropy Constrained .....	43
5.5	Existing Frame Work .....	45
5.5.1	Uniqueness of Existing Framework .....	46
<b>6.</b>	<b>Entropy Coding .....</b>	<b>47</b>
6.1	Data Compression .....	47
6.2	Idea of Arithmetic Coding .....	49
6.3	Models for Arithmetic Coding .....	53
6.3.1	Fixed Model .....	54
6.3.2	Adaptive Models .....	54
<b>7.</b>	<b>Methodology and Framework .....</b>	<b>55</b>
7.1	High level structure of single stage EC-VQ .....	55
7.1.1	System Encoder .....	56
7.1.2	System Decoder .....	57
7.2	Major Components of System .....	57
7.2.1	Design of Vector Quantizer .....	57
7.2.1.1	Measuring Criteria .....	58
7.2.1.2	Encoding using Entropy Constrained .....	58
7.2.2	Specification of Entropy Coder .....	59
7.2.3	Multiplexer-Demultiplexer .....	59
7.2.4	Decoding Process .....	59
<b>8.</b>	<b>Implementation .....</b>	<b>60</b>
8.1	Preparing a Training set .....	60
8.1.1	Scaling of images .....	60

---

8.1.2 Concatenating the images.....	60
8.2 Blocking code.....	60
8.3 CEC-VQ Process.....	62
8.3.1 Codebook Generation.....	62
8.3.1.1 Generating codebook on MSE basis.....	63
8.3.1.2 Generation of code book on Entropy base.....	67
8.3.2 Encoding using entropy constraints.....	69
8.4 Entropy Coding.....	71
8.5 Decoding.....	71
8.6 Unblocking.....	71
8.7 Verifying the Perfection Reconstruction.....	71
8.8 Display Voronoi Regions.....	72
<b>9. Results and Analysis.....</b>	<b>74</b>
9.1 Experimental Techniques.....	74
9.1.1 MSR.....	74
9.1.2 SNR.....	74
9.1.3 PSNR.....	74
9.2 Testing Measures.....	75
9.3 Experimental Results.....	75
9.4 Future Work and Conclusion.....	81

**Reference and Bibliography**

## **Chapter 1**

---

# **Introduction**

## 1. Introduction

In this age of information, we see an increasing trend toward the use of digital representation for audio, speech, images, and video. Much of this trend is being fueled by the exploding use of computers and multimedia computer application. Man's fascination for images has existed and has joined his activities ever. Digital image processing is a rather recent topic if we compare it with human beings' interest towards visual stimulus. To acquire a digital image (to *digitize* it), roughly two processes must be accomplished: *sampling* (get measurements of the luminance level, equally distributed along the bi dimensional space) and *quantization* (representation of the measured value with an integer). Digital image processing, however, usually generates huge loads of data, so transmission and storage needs have motivated efforts to obtain more efficient compression techniques. Data compression is the mapping of a given set of data into a bit stream, with the sole goal of diminishing the number of bits needed to represent the data set, trying to lose the minimum quantity (or nothing) of information.

At the heart of these algorithms is *quantization*, a field of study that has matured over the past few decades. In simple term Quantization is the mapping of a large set of possible inputs into a smaller set of possible outputs. In scalar quantization (SQ), the inputs are individual numbers. Scalar quantization includes such operations as "rounding to the nearest integer." The possible outputs, in this case the integers, are called quantization levels or reproduction levels or reconstruction levels. Obviously, quantization is an irreversible process, since it involves discarding information. if it done wisely, the error introduced by the process can be held to minimum.

The generalization of this notion (quantization) is called vector quantization, commonly denoted as VQ, but it involves quantizing blocks of samples together. Blocks of samples, which we view as vectors, are represented by codevectors stored in a codebook – a process called encoding. The code book is typically a table stored in a digital memory, where each table entry represents a different codevector. The Key element in designing a VQ is determining, how to generate and design codebook for a given input source.

### 1.1 Overview

Compressing image data by using Vector Quantization (VQ) will measure up to Training Vectors by way of Codebook. Consequently, a catalog of position with minimum distortion will achieve. The research presents the Splitting solution to implement the

Codebook, which improves the image quality by the average Training Vectors, then splits the average result to Codebook that has minimum distortion. The result from this presentation will give the better quality of the image than using Random Codebook.

A common approach to lossless image coding is to pre-process the data, in a way that removes statistical dependencies among the input symbol and code those symbols with entropy coder. High order entropy coding is a powerful technique for exploiting high order statistical dependencies. Entropy constrained residual vector quantization (EC-RVQ) is relatively new method for image compression that combines multistage residual vector quantization(RVQ) and entropy-constrained optimization.

The method, which we call conditional entropy-constrained residual vector quantization (CEC-RVQ), employs a high-order entropy conditioning strategy that capture local information in the neighboring vectors. The complexity of this design is relatively low, due mainly to the efficiency of the multistage structure of the residual vector quantizer, but also effectiveness of the searching technique used to locate the best conditioning spatial stage region support.

EC-RVQ is a memory less vector quantizer that is designed to minimize the average distortion subject to constraints on the first-order entropy of the vector quantizer output. The advantage of EC-RVQ over the other VQ methods is achieved mainly by exploiting the statistical dependencies among the VQ stages [10].

The efficiency of the RVQ structure, CEC-RVQ can outperform EC-RVQ in rate-distortion, performance, encoding complexity, and memory. CEC-RVQ can also achieve better compression performance than some of the best previously reported finite-state and predictive VQ techniques of classes. Much of the performance gain due to the strategy of locating the best conditioning stage symbols given a limit on entropy coder complexity and memory [5].

## 1.2 Communication System Model

Consider the block diagram of communication system model shown in figure 1.1. The basic building blocks of this model are source-user pair, the encode-decoder pair and the channel. This model is important for two reasons. First, the various elements are suitable idealized from their physical components to allow the model to be sufficiently general for most communication system. Second, if the model is statistically characterizable, then

the model proves amenable to productive analysis. We proceed to describe briefly each of this communication system models.

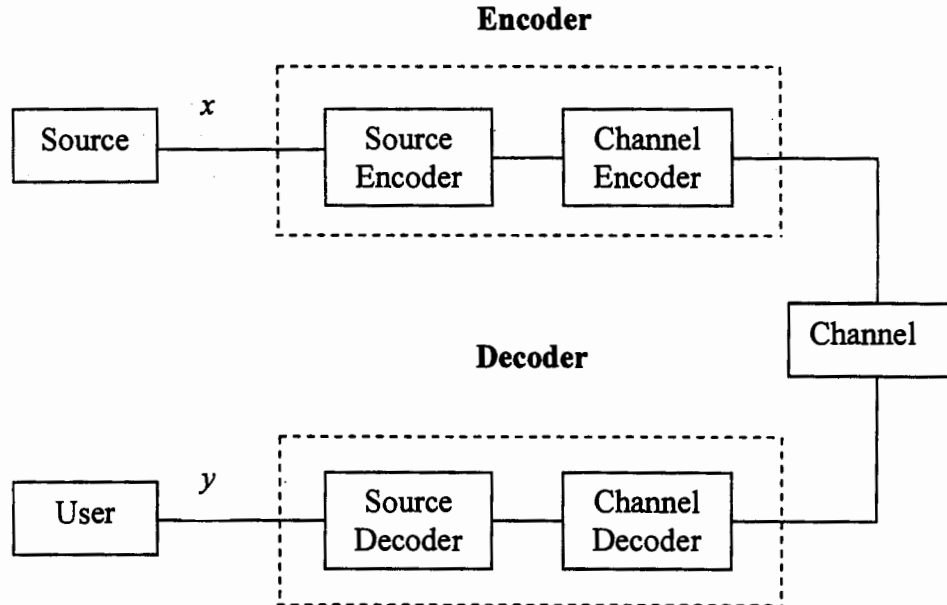


Figure 1.1 Block diagram of a communication system model

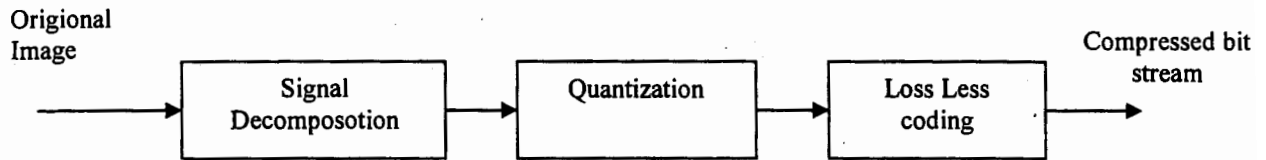
The function of a communication system is to convey “useful” information from the source to the user. Often, the user does not require an exact reproduction of the message produced by the source. For example, distortion, which does not degrade the intelligibility of speech, does not hinder a speech communication 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 usefulness of the source’s intended message. Unfortunately, such fidelity ensures are unknown for many source-user pairs, or are very complex and difficult to use. Less descriptive, but more tractable fidelity measure may be defined by assigning functional values to the various errors that the communication system may make.

### 1.3 Compression

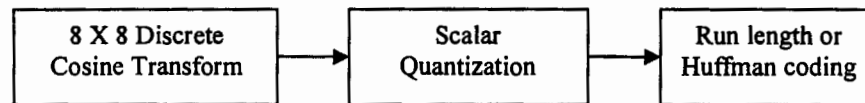
Image compression maps as shown in figure 1.2, an original image into a bit stream suitable for communication over or storage in a digital medium. The number of bits required to represent the coded image should be smaller than that required for original image so that one can use less storage space or communication time.



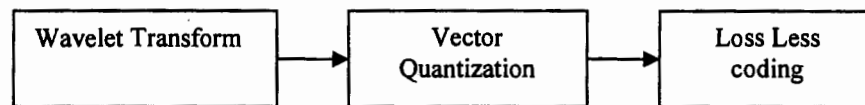
There are two type of compression lossless compression, also called noiseless coding; data compaction or entropy coding refers to algorithms which allow the original pixel intensities to be perfectly recovered from compressed representation. Lossy compression algorithms do not allow that.



(a) A General System for Image Compression



(b) The JPEG System for Image Compression



(c) The compression System considered in this work

Figure 1.2: Image compression System

### 1.3.1 Lossy/Lossless Coding

In lossless coding, often referred to as entropy coding, the coded message is perfectly reconstructed from the coded data. Any source that is inherently digital is subject to lossless coding. Compression is achieved by exploiting skewed symbol probability sets and inters symbol dependencies in the source to be coded. The family of entropy coders is big, and can be categorized into coders where the symbol probabilities need to be known, or not need to be known, in order to design the coder. Huffman coding [11] and arithmetic coding [12] are well-known examples belonging to the first group.

In lossy coding, or source coding with a fidelity criterion, the objective is to reconstruct the signal with as little distortion as possible according to some distortion criterion. Lossy coding, also referred to as quantization, introduces a new dimension to the coding problem.

## 1.4 Vector Quantization

Scalar quantization is used primarily for analog-to-digital conversion, VQ is used with sophisticated digital signal processing, where in most cases the input signal already has some form of digital representation and the desired output is a compressed version of the original signal. In VQ, the inputs are vectors, rather than scalars.

A vector quantizer  $Q$  of dimension  $k$  and size  $N$  is a mapping from a vector in  $k$ -dimensional Euclidean space,  $R^k$ , into a finite set  $C$  containing  $N$  output or reproduction points, called code vector or codewords as shown in figure 1.3. Thus,

$$Q: R^k \rightarrow C,$$

Where  $C = (y_1, y_2, \dots, y_N)$  and  $y_i \in R^k$  for each  $i \in I = \{1, 2, \dots, N\}$  set  $C$  is called codebook.

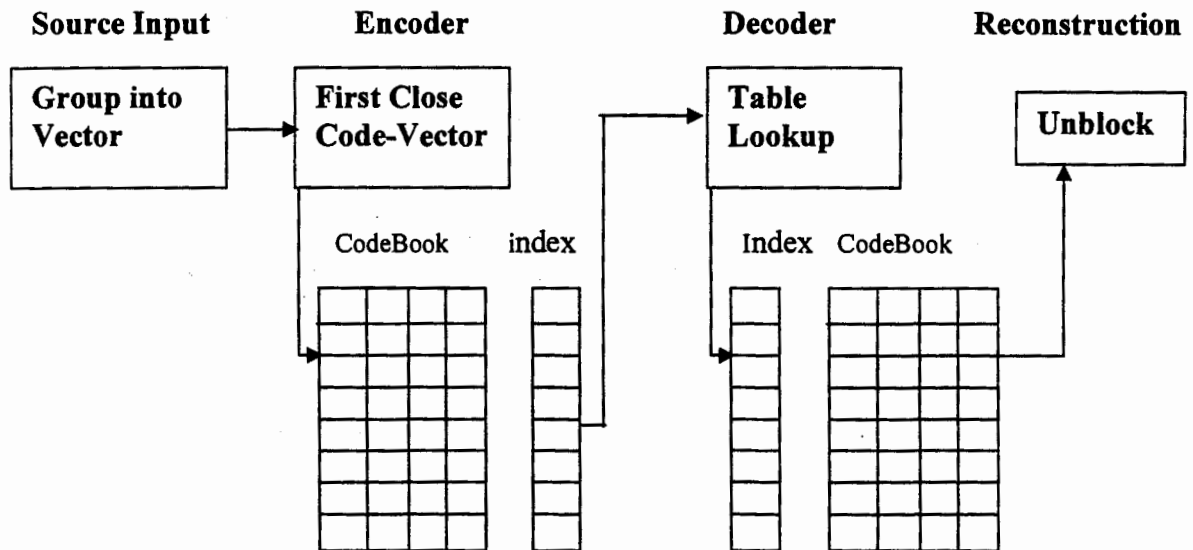


Figure 1.3. Block diagram of Vector Quantization

### 1.4.1 VQ is better than scalar Quantization

Scalar quantizer treats each pixel independently and does not use correlation between neighboring pixels. Whereas in vector quantizer, Image (data) divided into vectors (blocks) as the result correlation among pixels in vectors is exploited. So, Block size should be appropriate:

- Too large block : correlation is loss
- Too small block : More code vectors
- If no interpixel correlation, then no gain

### 1.4.2 Why Entropy coding with Quantization?

Entropy coding is now being used frequently in conjunction with vector quantization for image coding. Its use is motivated by the fact that the probability distribution of VQ coded images is generally skewed or non-uniform. While the average bit rate can most often be reduced by entropy coding the VQ codeword, improvement in rate-distortion performance is usually attainable by embedding the entropy coding in the design process such that both the VQ codebook and entropy coder are optimized jointly.

By generalizing the entropy-constrained scalar quantization design to vector case, introduced an iterative descent algorithm for the design of entropy-constrained vector quantization.

Entropy constrained VQ method consist of first quantizing the input image using a high order entropy-constrained VQ and then coding the image using a first order entropy coder. The distortion measure used in the entropy constrained optimization is essentially the first order entropy of image.

Ordinary VQ strives to minimize average distortion subject to a constraint on the number of codewords. This is equivalent to minimizing average distortion for a given rate, when rate is measured by the log of the number of codewords. This is suitable optimization formulation when the system is fixed rate and no subsequent entropy coding take place. If we intend to entropy code the output of a VQ, then it make more sense to design the quantizer with the entropy coding in mind, that is to minimize the average distortion subject to a constraint on the quantization output entropy rather than the number of codewords.

### 1.4.3 Major Issues in VQ

- **Generation (construction) of codebook**
  - concerns what needs to be included in the codebook
- **Design of codebook**
  - concerns structuring codebook entries to minimize search time

## **Chapter 2**

---

# **Fundamentals**

## 2. Fundamentals

Everyday, an enormous amount of information is stored, processes and transmitted digitally. Companies provide business associates, investors and potential customers with financial data, annual reports, and inventory and product information over the internet. Other entry tracking two of the most basic on-line transaction is 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 communication 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 communication and digital storage of movies would not be feasible without a high degree of compression. 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. Mathematical viewpoint, this amount to transforming a 2-d pixel array into a statistically 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 an effort in this field was 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 images compression standards, the field has undergone significant growth through the practical application of the theoretic work that began in the 1940's, when C.E.Shannon 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 area just mentioned, image compression is the natural technology for handling the increased spatial resolution 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, document and medical images, facsimile transmission, and the control of remotely piloted vehicles in

military. in short, an ever expanding number of applications depend on the efficient manipulation, storage and transmission of binary, gray scale and color images.

## 2.1 Background

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. Infact, 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 relates the same story. Here, the information of interest is the story; words are the data used to relate the information. If the individual use a different number of words to tell the same basic story, to different versions of the story are created, and at least one includes nonessential data that is it contain either provide no relevant information or simply restate that which is already known. It is thus said to be data redundancy.

Data redundancy is a central issue in digital image compression. It is not an abstract concept but a mathematically quantifiable entity. If  $n_1$  and  $n_2$  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 can be defined as

$$RD = 1 - 1 / CR \text{ ..... (2.1)}$$

Where CR, commonly called as compression ratio, is

$$CD = n_1/n_2 \text{ ..... (2.2)}$$

For the case  $n_2=n_1$ ,  $CR =1$  and  $RD=0$ , including that the first representation of the information conations no redundant data, when  $n_2 \ll n_1$ ,  $CD \rightarrow \infty$  and  $RD \rightarrow 1$ , imply significant compression contains no redundant data. Finally, when the case  $n_2 \gg n_1$ ,  $CR \rightarrow 0$  and  $RD \rightarrow -\infty$ , indicating that the second data set contain much more data than the original representation. This is of course, is the normally undesirable case of data expansion. In general CR and RD lie in the open intervals  $(0, \infty)$  and  $(-\infty, 1)$  respectively. A practical compression ratio, such as 10(or 10:1), means that the second or compresses data set. The corresponding redundancy of 0.9 implies that 90% of the data in the first data set is redundant.

In digital image compression, three basic redundancies can be identified are exploited: coding redundancy, inter pixel redundancy and psycho visual redundancy. Data compression is achieved when one or more of these reduced or eliminates.

### 2.1.1 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 hoe 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  $r_k$  in the interval  $[0,1]$  represent the gray levels on an image and that each  $r_k$  occurs with probability  $pr(r_k)$ .

$$Pr(r_k) = n_k / n, \quad k=0,1,2,\dots,L-1 \quad (2.3)$$

Where  $L$  is the number of gray levels,  $n_k$  is the number of times that the  $k^{\text{th}}$  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  $r_k$  is  $l(r_k)$ , then average number of bits represent each pixel is

$$L_{\text{avg}} = \sum l(r_k)pr(r_k) \quad (2.4)$$

That is the average length of code words assigned to the various gray-level values is found by summing the products if the number of bits used to represent each gray level and the probability that the gray levels occurs. Thus the total number of bit required to code an  $M \times N$  and  $MNL_{\text{avg}}$ .

Representing the gray levels of an image with a natural  $m$ -bit binary code reduces to  $m$  bits, that is,  $L_{\text{avg}} = m$  when is substituted for  $l(r_k)$ . Then the constant  $m$  may be taken outside the summation, leaving only the sun of the  $pr(r_k)$  for  $0 \leq k \leq L-1$ , which of course equal 1.

### 2.1.2 Inter-pixel Redundancy

If the gray levels in image 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 pixel 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. This correlation result from the structural or geometric relationship between the objects in the image. Autocorrelation coefficient computes along one line of each image.

$$\gamma(\Delta n) = A(\Delta n) / A(0) \quad (2.5)$$

where

$$A(\Delta n) = (1/N) - \Delta n \quad (2.6)$$

The scaling factor accounts for the varying number of sum terms that arise for each integer value of  $\Delta n$ ,  $\Delta n$  must be strictly less than  $N$ , the number of pixels on a line.

### 2.1.3 Psycho visual Redundancy

We notice in inter pixel redundancy 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 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 psycho-visually redundant. It can be eliminated without significantly impairing the quality of image perception.

The psychovisual redundancies exist should not come as surprise, because human perception of the information in an image normally does not involve quantitative analysis of every pixel value of image. In general, an observer searches for distinguishing features such as edge or textual regions and mentally combines them into recognizable grouping. The brain then correlates these groupings with prior knowledge in order to complete the image interpretation process.

Since the elimination of psychovisually redundant data result 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 values to limited number of output values. As it is irreversible operation, Quantization results in lossy data compression.

## 2.2 Fidelity Criteria

As noted previously, removal of psycho visually redundant 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 (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 criteria. A good example is the root mean square error between an input and output image. Let  $f(x,y)$  denote an estimate and approximation of  $f(x,y)$  that result 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 define as.

$$e(x,y) = f(x,y) - f(x,y) \quad (2.7)$$

so that the total error between two images is

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

where the images are of the size  $M \times N$ . The root mean square error,  $e_{rms}$ , between  $f(x,y)$  and  $f(x,y)$  then is the square root of squared error averages over the  $M \times N$  array, or

$$e_{rms} = \left[ (1/MN) \times \sum \sum [f(x,y) - f(x,y)]^2 \right]^{1/2} \quad (2.9)$$

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

a closely related objective fidelity criterion is the mean square signal-to-noise ratio of the compresses-decompresses image. If  $f(x,y)$  is considered 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  $SNR_{rms}$ .

### 2.3 Image Compression Models

In section 2.2 we discusses individually three general techniques for reducing or compressing the amount of data required-to represent an image. However these three techniques typically are combined to form practical image compression systems. In this section we examine overall characteristics of such a system and develop a general model to represent it.

A compression system consists of two distinct blocks; an encoder and a decoder. An input image  $f(x,y)$  fed into the encoder, which creates a set of symbols from the input data. After transmission over the channel, the encoded representation is fed to the decoder,

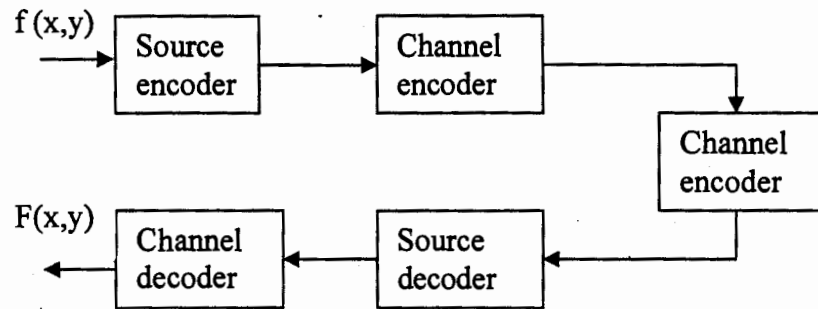
where a reconstructed output image  $f(x,y)$  is generated. In general  $f(x,y)$  may or may not be an exact replica of  $f(x,y)$ . If it is, the system is error free or information preserving; if not, some level of distortion is present in the reconstructed image.

Both the encoder and decoder, figure 2.1; consist of two relatively independent function or sub blocks. The encoder is made up of a source encoder, which removes input redundancies and a channel encoder, which increase the noise immunity of the source encoder's output. As would be expected, the decoder includes a channel decoder followed by a source decoder. If the channel between the encode and decoder is noise free, the channel encoder and decoder are omitted and the general encoder and decoder become the source encoder and decoder respectively.

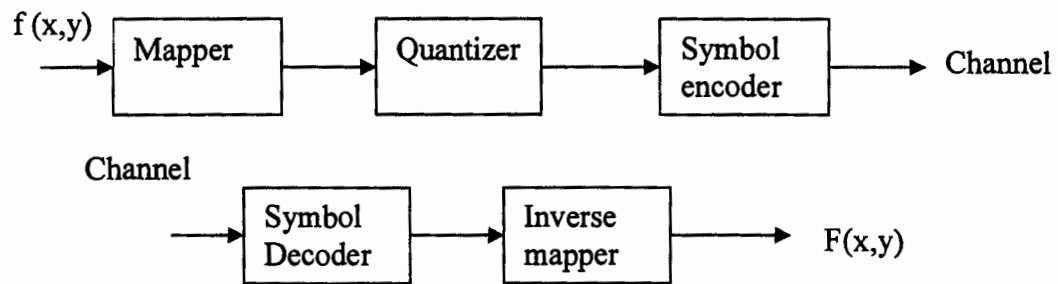
### **2.3.1 The Source Encoder and Decoder**

The source encoder is responsible for reducing or eliminating any coding, inter-pixel or psycho visual redundancies in the input image. The specific application and associated fidelity requirements dictate the best encoding approach to use in any given situation. Normally, the approach can be modeled by a series of three independent operations.

As shown in Figure 2.2(a), In the first stage of the source encoding process, the mapper transforms the -input data into a format designed to reduce inter-pixel redundancies in the input image. This operation generally is reversible and may or may not reduce directly the amount of data required to represent the image.



**Figure 2.1 A General Compression Model**



**Figure 2.2 (a) Source Encoder (b) Source Decoder**

An image is represented by a set of transform coefficients. Here, the mapper transforms the image into an array of coefficient  $s$ , making its interpixel redundancies more accessible for compression in later stages of encoding process.

The second stage or quantizer block reduces the accuracy of the mappers output in accordance with some pre-established fidelity criterion. This reduces the psycho visual redundancies of the input image.

In the third and final stage of the source encoding process, the symbol coder creates a fixed or variable length code to represent the quantizer output and maps the output in accordance with the code. The term symbol coder distinguishes this coding operation from the overall source encoding process. In most cases, a variable length code is used to represent the mapped and quantized data set. It assigns the shortest code words to most frequently occurring output values and thus reduces coding redundancy. Source encoding process as three successive operation but all three operations are not necessarily included in every compression system.

### 2.3.2 The Channel Encoder and Decoder

The channel encoder and decoder play an important role in the overall encoding-decoding process when the channel is noisy or prone to error. They are designed to reduce the

impact of channel noise by interesting a controlled form of redundancy into the source encoded data. As the output of the source encoder contains little redundancy, it would be highly sensitive to transmission noise without the addition of this “controlled redundancy”. One of the most useful channel encoding techniques was devised by R, W.Hamming.

## 2.4 Image Compression Techniques

Image compression techniques can be divided into two major families; lossy and loss less. Lossy image compression concedes a certain loss of accuracy in exchange for greatly increased compression. Loss less compression consist 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 file.

### 2.4.1 Lossless 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, lossless, entropy coding. Lossless is often required in some systems e.g in coding binary computer programs for storage or transmission.

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 problem in the implementation.

Depending on the memory structure of the source, noiseless source coding typically achieves in practice a data rate deduction or data compression ratio of about 5:1, and often achieves only about 2:1. these compression ratios are not not sufficient in many circumstances to reduce the output rate of discrete amplitude source to values not greater than a given

### 2.4.2 Lossy Coding

If the original data or signal cannot be accurately reconstructed from the coded data or signal then the technique is called Noisy source coding. Loss can be suitably controlled using a fidelity criteria or distortion measure. Noisy source coding techniques have more compression rates then noiseless source coding techniques.

Shanon Noisy Source Coding Theorem and its converse imbue the distortion rate function with its operational significance. The distortion  $D(A)$  of code book  $A$  is less than or equal to some constant  $D$ , then we say the codebook is  $D$ -admissible.

The size of a code book is defined by the number of  $n$ -dimensional code vectors which comprise it.

## **Chapter 3**

---

# **Compression Techniques**

### 3. Compression Techniques

One of the important aspects of image storage is its efficient compression. To make this fact clear let's see an example.

An image, 1024 pixel x 1024 pixel x 24 bit, without compression, would require 3 MB of storage and 7 minutes for transmission, utilizing a high speed, 64 kb/s, ISDN line. If the image is compressed at a 10:1 compression ratio, the storage requirement is reduced to 300 KB and the transmission time drops to under 6 seconds. Seven 1 MB images can be compressed and transferred to a floppy disk in less time than it takes to send one of the original files, uncompressed, over an AppleTalk network.

In a distributed environment large image files remain a major bottleneck within systems. Compression is an important component of the solutions available for creating file sizes of manageable and transmittable dimensions. Increasing the bandwidth is another method, but the cost sometimes makes this a less attractive solution.

Platform portability and performance are important in the selection of the compression/decompression technique to be employed. Compression solutions today are more portable due to the change from proprietary high end solutions to accepted and implemented international standards. JPEG is evolving as the industry standard technique for the compression of continuous tone images.

Two categories of data compression algorithm can be distinguished: **lossless and 'lossy'** Lossy techniques cause image quality degradation in each compression/decompression step. Careful consideration of the human visual perception ensures that the degradation is often unrecognizable, though this depends on the selected compression ratio. In general, lossy techniques provide far greater compression ratios than lossless techniques.

Here we'll discuss the roles of the following data compression techniques:

- Lossless coding techniques
  - Run length encoding
  - Huffman encoding
  - Entropy coding (Lempel/Ziv)
  - Area coding
- Lossy coding techniques
  - Transform coding (DCT/Wavelets/Gabor)
  - Vector quantization
  - Segmentation and approximation methods

- Spline approximation methods (Bilinear Interpolation/Regularisation)
- Fractal coding (texture synthesis, iterated functions system [IFS], recursive IFS [RIFS])
- Efficiency and quality of different lossy compression techniques

### 3.1 Lossless coding techniques

Lossless coding guaranties that the decompressed image is absolutely identical to the image before compression. This is an important requirement for some application domains, e.g. medial imaging, where not only high quality is in demand, but unaltered archiving is a legal requirement. Lossless techniques can also used for the compression of other data types where loss of information is not acceptable, e.g. text documents and program executables.

Some compression methods can be made more effective by adding a 1D or 2D delta coding to the process of compression. These deltas make more effectively use of run length encoding, have (statistically) higher maxima in code tables (leading to better results in Huffman and general entropy coding), and build greater equal value areas usable for area coding.

Some of these methods can easily be modified to be lossy. Lossy element fits perfectly into 1D/2D run length search. Also, logarithmic quantization may be inserted to provide better or more effective results.

#### 3.1.1 Run length encoding

Run length encoding is a very simple method for compression of sequential data. It takes advantage of the fact that, in many data streams, consecutive single tokens are often identical. Run length encoding checks the stream for this fact and inserts a special token each time a chain of more than two equal input tokens are found. This special input advises the decoder to insert the following token  $n$  times into his output stream.

Following is a short example of this method:

Clock	Input	Coder Output	Decoder Output
1	A		
2	B	A	
3	C	B	A
4	C	∅	B
5	C	∅	∅
6	C	∅	∅



7	C	∅	∅
8	D	5C	∅
9	E	D	CCCCC
10	∅	E	D
11	∅	∅	E

In this example, there are 9 tokens going into the coder, but just 7 going out. The affectivity of run length encoding is a function of the number of equal tokens in a row in relation to the total number of input tokens. This relation is very high in un-dithered two tone images of the type used for facsimile. Obviously, affectivity degrades when the input does not contain too many equal tokens. With a rising density of information, the likelihood of two following tokens being the same does sink significantly, as there is always some noise distortion in the input.

Run length coding is easily implemented, either in software or in hardware. It is fast and very well verifiable, but its compression ability is very limited.

### 3.1.2 Huffman encoding

This algorithm, developed by D.A. Huffman, is based on the fact that in an input stream certain tokens occur more often than others. Based on this knowledge, the algorithm builds up a weighted binary tree according to their rate of occurrence. Each element of this tree is assigned a new code word, whereat the length of the code word is determined by its position in the tree. Therefore, the token which is most frequent and becomes the root of the tree is assigned the shortest code. Each less common element is assigned a longer code word. The least frequent element is assigned a code word which may have become twice as long as the input token.

The compression ratio achieved by Huffman encoding uncorrelated data becomes something like 1:2. On slightly correlated data, as on images, the compression rate may become much higher, the absolute maximum being defined by the size of a single input token and the size of the shortest possible output token (max. compression = token size[bits]/2[bits]). While standard palletized images with a limit of 256 colors may be compressed by 1:4 if they use only one color, more typical images give results in the range of 1:1.2 to 1:2.5.

### 3.1.3 Entropy coding (Lempel/Ziv)

The typical implementation of an entropy coder follows J. Ziv/A. Lempel's approach. Nowadays, there is a wide range of so called modified Lempel/Ziv coding. These algorithms all have a common way of working. The coder and the decoder both build up

an equivalent dictionary of met symbols, each of which represents a whole sequence of input tokens. If a sequence is repeated after a symbol was found for it, then only the symbol becomes part of the coded data and the sequence of tokens referenced by the symbol becomes part of the decoded data later. As the dictionary is build up based on the data, it is not necessary to put it into the coded data, as it is with the tables in a Huffman coder.

This method becomes very efficient even on virtually random data. The average compression on text and program data is about 1:2, the ratio on image data comes up to 1:8 on the average GIF image. Here again, a high level of input noise degrades the efficiency significantly. Entropy coders are a little tricky to implement, as there are usually a few tables, all growing while the algorithm runs. LZ coding is subject to patents owned by IBM and Unisys (formerly Sperry).

#### **3.1.4 Area coding**

Area coding is an enhanced form of run length coding, reflecting the two dimensional character of images. This is a significant advance over the other lossless methods. For coding an image it does not make too much sense to interpret it as a sequential stream, as it is in fact an array of sequences, building up a two dimensional object. Therefore, as the two dimensions are independent and of same importance, it is obvious that a coding scheme aware of this has some advantages. The algorithms for area coding try to find rectangular regions with the same characteristics. These regions are coded in a descriptive form as an Element with two points and a certain structure. The whole input image has to be described in this form to allow lossless decoding afterwards.

The possible performance of this coding method is limited mostly by the very high complexity of the task of finding largest areas with the same characteristics. Practical implementations use recursive algorithms for reducing the whole area to equal sized sub rectangles until a rectangle does fulfill the criteria defined as having the same characteristic for every pixel.

This type of coding can be highly effective but it bears the problem of a nonlinear method, which cannot be implemented in hardware. Therefore, the performance in terms of compression time is not competitive, although the compression ratio is.

## 3.2 Lossy coding techniques

In most of applications we have no need in the exact restoration of stored image. This fact can help to make the storage more effective, and this way we get to lossy compression methods. Lossy image coding techniques normally have three components:

- *image modeling* which defines such things as the transformation to be applied to the image
- *parameter quantization* whereby the data generated by the transformation is quantized to reduce the amount of information
- *encoding*, where a code is generated by associating appropriate codewords to the raw data produced by the quantizer.

Each of these operations is in some part responsible of the compression. Image modeling is aimed at the exploitation of statistical characteristics of the image (i.e. high correlation, redundancy). Typical examples are transform coding methods, in which the data is represented in a different domain (for example, frequency in the case of the Fourier Transform [FT], the Discrete Cosine Transform [DCT], the Kahrunen-Loewe Transform [KLT], and so on), where a reduced number of coefficients contains most of the original information. In many cases this first phase does not result in any loss of information.

The aim of quantization is to reduce the amount of data used to represent the information within the new domain. Quantization is in most cases not a reversible operation: therefore, it belongs to the so called 'lossy' methods.

Encoding is usually error free. It optimizes the representation of the information (helping, sometimes, to further reduce the bit rate), and may introduce some error detection codes.

In the following sections, a review of the most important coding schemes for lossy compression is provided. Some methods are described in their canonical form (transform coding, region based approximations, fractal coding, wavelets, hybrid methods) and some variations and improvements presented in the scientific literature are reported and discussed.

### 3.2.1 Transform coding (DCT/Wavelets/Gabor)

A general transform coding scheme involves subdividing an  $N \times N$  image into smaller  $n \times n$  blocks and performing a *unitary transform* on each sub image. A unitary transform is a reversible linear transform whose kernel describes a set of complete, orthonormal discrete basic functions. The goal of the transform is to decorrelate the original signal, and this decorrelation generally results in the signal energy being redistributed among only a small

set of transform coefficients. In this way, many coefficients may be discarded after quantization and prior to encoding. Also, visually lossless compression can often be achieved by incorporating the HVS contrast sensitivity function in the quantization of the coefficients.

Transform coding can be generalized into four stages:

- *image subdivision*
- *image transformation*
- *coefficient quantization*
- *Huffman encoding.*

For a transform coding scheme, logical modeling is done in two steps: a segmentation one, in which the image is subdivided in bidimensional vectors (possibly of different sizes) and a transformation step, in which the chosen transform (e.g. KLT, DCT, Hadamard) is applied.

Quantization can be performed in several ways. Most classical approaches use 'zonal coding', consisting in the scalar quantization of the coefficients belonging to a predefined area (with a fixed bit allocation), and 'threshold coding', consisting in the choice of the coefficients of each block characterized by an absolute value exceeding a predefined threshold. Another possibility, that leads to higher compression factors, is to apply a vector quantization scheme to the transformed coefficients.

The same type of encoding is used for each coding method. In most cases a classical Huffman code can be used successfully. The JPEG and MPEG standards are examples of standards based on transform coding.

### 3.2.2 Vector quantization

A vector quantizer can be defined mathematically as a transform operator  $T$  from a  $K$ -dimensional Euclidean space  $R^K$  to a finite subset  $X$  in  $R^K$  made up of  $N$  vectors. This subset  $X$  becomes the vector codebook, or, more generally, *the codebook*.

Clearly, the choice of the set of vectors is of major importance. The level of distortion due to the transformation  $T$  is generally computed as the most significant error (MSE) between the "real" vector  $x$  in  $R^K$  and the corresponding vector  $x' = T(x)$  in  $X$ . This error should be such as to minimize the Euclidean distance  $d$ .

An optimum scalar quantizer was proposed by Lloyd and Max. Later on, Linde, Buzo and Gray resumed and generalized this method, extending it to the case of a vector quantizer.

The algorithm that they proposed is derived from the KNN cauterization method, and is performed by iterating the following basic operations:

- subdivide the training set into  $N$  groups (called 'partitions' or 'Voronoi regions'), which are associated with the  $N$  codebook letters, according to a minimum distance criterion;
- the centroids of the Voronoi regions become the updated codebook vectors;
- compute the average distortion: if the percent reduction in the distortion (as compared with the previous step) is below a certain threshold, then STOP.

Once the codebook has been designed, the coding process simply consists in the application of the  $T$  operator to the vectors of the original image. In practice, each group of  $n$  pixels will be coded as an address in the vector codebook, that is, as a number from 1 to  $N$ .

The LBG algorithm for the design of a vector codebook always reaches a local minimum for the distortion function, but often this solution is not the optimal one. A careful analysis of the LBG algorithm's behavior allows one to detect two critical points: the choice of the starting codebook and the uniformity of the Voronoi regions' dimensions. For this reason some algorithms have been designed that give better performances. With respect to the initialization of the LBG algorithm, for instance, one can observe that a random choice of the starting codebook requires a large number of iterations before reaching an acceptable amount of distortion. Moreover, if the starting point leads to a local minimum solution, the relative stopping criterion prevents further optimization steps.

### 3.2.3 Segmentation and approximation methods

With segmentation and approximation coding methods, the image is modeled as a mosaic of regions, each one characterized by a sufficient degree of uniformity of its pixels with respect to a certain feature (e.g. grey level, texture); each region then has some parameters related to the characterizing feature associated with it.

The operations of finding a suitable segmentation and an optimum set of approximating parameters are highly correlated, since the segmentation algorithm must take into account the error produced by the region reconstruction (in order to limit this value within determined bounds). These two operations constitute the logical modeling for this class of coding schemes; quantization and encoding are strongly dependent on the statistical

characteristics of the parameters of this approximation (and, therefore, on the approximation itself).

Classical examples are *polynomial approximation* and *texture approximation*. For polynomial approximation regions are reconstructed by means of polynomial functions in  $(x, y)$ ; the task of the encoder is to find the optimum coefficients. In texture approximation, regions are filled by synthesizing a parameterized texture based on some model (e.g. fractals, statistical methods, Markov Random Fields [MRF]). It must be pointed out that, while in polynomial approximations the problem of finding optimum coefficients is quite simple (it is possible to use least squares approximation or similar exact formulations), for texture based techniques this problem can be very complex.

#### **3.2.4 Spline approximation methods (Bilinear Interpolation/Regularisation)**

These methodologies fall in the more general category of image reconstruction or sparse data interpolation. The basic concept is to interpolate data from a set of points coming from original pixel data or calculated in order to match some error criteria. The problem of interpolating a set of sparse data is generally ill posed, so some regularization algorithm must be adopted in order to obtain a unique solution. The problem is well documented, and many interpolation algorithms have been proposed.

In order to apply this kind of technique to image coding, a good interpolant must be used to match visual criteria. Spline interpolation provides a good visual interpolant, notwithstanding its requiring a great computational effort. Bilinear interpolation is easier to implement, while maintaining a very good visual quality. Regularization involves the minimization of an energy function in order to obtain an interpolant which presents some smoothness constraints; it can be combined with non-continuities along edges in order to preserve contour quality during reconstruction. Generally all interpolants computations require the solution of very large linear equation sets, even if related to very sparse matrices. This leads to the use of recursive solution such as relaxation (suitable for a large parallel implementation), or to the use of gradient descent algorithm.

The use of an interpolation algorithm for image coding is more interesting when related to techniques such as two source decomposition, where the image is modeled as the sum of two sources; one is the stationary part (it can be considered related to the low frequency content), the second is the residual content coming from non-stationeries such as edges. The first of the two sources is coded by means of a prediction scheme that can be one of the previous described interpolants. The second source (the residual) can be coded trough

the use of a classical coding method (transform coding, vector or scalar quantization etc). Moreover two source decomposition is a very effective coding scheme as far as it shows a low tile effect that affects all block coding techniques when compression factors become higher.

### **3.2.5 Fractal coding (texture synthesis, iterated functions system [IFS], recursive IFS [RIFS])**

Fractal parameters, including fractal dimension, lacunarity, as well as others described below, have the potential to provide efficient methods of describing imagery in a highly compact fashion for both intra- and interframe applications. Fractal methods have been developed for both noisy and noise free coding methods.

Images of natural scenes are likely candidates because of the fractal structure of the scene content, but results are reported to be applicable to a variety of binary, monochrome, and color scenes.

In the mid-eighties Dr Michael Barnsley reported the use of "Iterated Function System" for image compression and synthesis. Using sets of affine transformations developed for a given image, and a principal result known as the "collage theorem", intraframe compressions in excess of 10,000:1 and interframe compression in excess of 1,000,000:1 were reported. The collage theorem states that if an image can be covered (approximately) with compressed affine transformations of itself, then the image can be (approximately) reconstructed by computing the attractor (in the sense of non linear dynamic systems) of this set of affine transformations.

This convergence was extremely slow, about 100 hours on a Cray, unless assisted by a person and was presented as an illustration of a scientific possibility, not as a commercial reality. In 1987 Barnsley and Sloan formed Iterated Systems Inc. to develop a product that would function in a commercial environment. By the end of 1988 Iterated Systems had developed the patented technique called the 'Fractal Transform' which has become the basis of their current product range. The development allowed a real world image to be reduced to a set of fractal equations based on the image being processed, rather than a huge library of pre-calculated, reference, fractal patterns. Image compression algorithms which are noise free have been reported to be developed from this transform for real time automatic image compression at ratios between 10:1 and 100:1

Researchers at BellCore have developed a compression method that incorporates a Peano Scan (the Peano curve is a "space filling" fractal curve) with a fractal based coding

schema for intraframe compression, so making it possible to archive high picture quality with bit rates of less than one bit per pixel. A fractal based method was reported earlier by Walach while the concepts of using a Peano Scan had been used for storage, compression and display. The BellCore researchers have combined these two fractal concepts into an efficient implementation which is now being incorporated into a VLSI chip. Compression results show good quality imagery at a rate of 0.8-1 bit per pixel. The technique may be implemented in an adaptive fashion since a local estimate of fractal dimension provides an objective measure of image complexity.

### 3.3 Efficiency and quality of different lossy compression techniques

The performances of lossy picture coding algorithms is usually evaluated on the basis of two parameters:

1. the compression factor (or analogously the bit rate) and
2. the distortion produced on the reconstruction.

The first is an objective parameter, while the second strongly depends on the usage of the coded image. Nevertheless, a rough evaluation of the performances of a method can be made by considering an objective measure of the error, like MSE or SNR.

For the methods described in the previous pages, average compression ratios and SNR values obtainable are presented in the following table:

Method	VQ	DCT-SQ	DCT-VQ	AP	SplineTSD	Fractals
Bit Rate (bpp)	0.8-0.4	0.8-0.3	0.3-0.08	0.3-0.1	0.4-0.1	0.8-0.0
SNR (dB)	36-30	36-31	30-25	image dependent	36-32	image dependent

#### *Comparison of Different Compression Methods*

During the last years, some standardization processes based on transform coding, such as JPEG, have been started. Performances of such a standard are quite good if compression factors are maintained under a given threshold (about 20 times). Over this threshold, artifacts become visible in the reconstruction and tile effect affects seriously the images decoded, due to quantization effects of the DCT coefficients.

On the other hand, there are two advantages: first, it is a standard, and second, dedicated hardware implementations exist. For applications which require higher compression factors with some minor loss of accuracy when compared with JPEG, different techniques should be selected such as wavelets coding or spline interpolation, followed by an



efficient entropy encoder such as Huffman, arithmetic coding or vector quantization. Some of these coding schemes, are suitable for progressive reconstruction (Pyramidal Wavelet Coding, Two Source Decomposition, etc). This property can be exploited by applications such as coding of images in a database, for previewing purposes or for transmission on a limited bandwidth channel.

### 3.4 Source Coding

Source coding, or data compression, is the art of finding efficient digital representations for a source. In theory the goal is to use as few bits as possible for a digital representation of the source. Getting more practical, the source and application at hand put many other constraints on the coder, such as delay and complexity constraints. If the source at hand is inherently digital, a perfect reconstruction is often required. Considering a real-time speech coding application, a digital representation will inevitably incur distortion. These examples split source coding into two main paradigms: lossless and lossy source coding,

#### 3.4.1 Lossy/Lossless Coding

In lossless coding, often referred to as entropy coding, the coded message is perfectly reconstructed from the coded data. Any source that is inherently digital is subject to lossless coding. Compression is achieved by exploiting skewed symbol probability sets and inter symbol dependencies in the source to be coded. The family of entropy coders is big, and can be categorized into coders where the symbol probabilities need to be known, or not need to be known, in order to design the coder. Huffman coding and arithmetic coding are well-known examples belonging to the first group. The latter group is also referred to as universal source coding, where Ziv-Lempel coding is a famous example [31]. Lossless coding can also be used for analog sources when applied in combination with lossy algorithms. One such example is the family of MPEG audio coders, where the resulting bits from the lossy coding are fed to an entropy code. For an introduction and an overview of lossless source coding see [20]. In lossy coding, or source coding with a fidelity criterion, the objective is to reconstruct the signal with as little distortion as possible according to some distortion criterion. Lossy coding, also referred to as quantization, introduces a new dimension to the coding problem. The toleration of a distortion enables tailoring of the coding scheme according to the intended consumer of the coded signal. Lossy source coding, and in particular vector quantization, is further treated below. For an introduction to lossy source coding, and for a comprehensive historical overview see [20].

---

### 3.4.2 Variable Rate

A coding scheme can also be categorized into being of fixed or variable-rate. As the name suggests, a variable-rate scheme operates with a rate that is varying over time. Variable-rate schemes can be categorized into source controlled schemes and network controlled schemes. The rate of a source controlled scheme is continuously adapted according to the characteristics of the source, while the rate of network controlled schemes are more slowly changing in order to adapt to varying network conditions. The family of entropy coders, discussed above, operates with a variable rate in order to adapt to varying source characteristics. For speech coding, lossy variable-rate schemes, adapting to source characteristics, have also been proposed, [20].

Multi-rate and embedded coders are designed to operate at varying rates in order to adapt to varying network conditions. In short, multi-rate coders consist of a battery of coders designed for different rates. For each transmission the rate is selected by the network. Embedded or layered coders are also intended for operation at multiple rates, but designed to enable a change of rate anywhere along the network path, by dropping part of the bit stream. Traditionally, many source coding algorithms for real-time communications have operated with a fixed rate, due to network restrictions. Lately, new networks have opened up the area for variable-rate coding to a higher extent. This has triggered new research on embedded and multi-rate coding .

### 3.4.3 Compression Means

The basic means for reducing the rate in a lossy source coder, are two-fold, and are often categorized into removal of statistical redundancies and reduction of perceptual irrelevancies. Redundancy removal is a means to reduce the rate of a coder by exploiting properties of the source to be coded. Many real-world sources show a dependency between source samples, the source is redundant. Redundancy removal is performed by distributing reconstruction vectors in accordance with the multi-dimensional source distribution, and/or by employing de-correlation methods such as linear prediction.

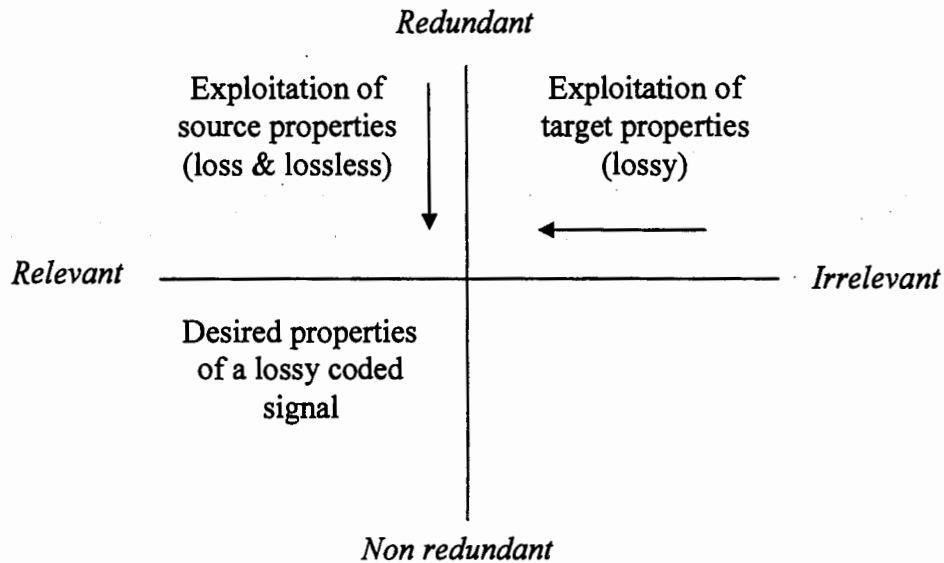


Figure 3.1 Coding Dimension : An illustration of the dimensions to be exploited to achieve compression in lossy and lossless source coding.

Irrelevancy is about exploitation of properties of the intended consumer of the coded source, for example hearing properties [20]. Irrelevancy removal is achieved by representing the source with a reduced precision, ideally such that the intended consumer will not perceive the distortion. The key to irrelevancy removal is the distortion criterion. The criterion employed should be chosen such that it correlates well with the perceived distortion of the intended consumer.

#### 3.4.4 Performance Limits

Theoretical limits for the performance of lossy source coding algorithms have been much studied. In 1959, Shannon published the classical rate-distortion theory [32]. Rate-distortion theory bounds the minimum required rate, given a distortion criterion and a source probability density function, without considering the structure of the code. It has been proven that the rate-distortion bound is theoretically approachable with many different code structures, such as block codes [32], although at the cost of infinite block lengths. Another asymptotic lossy source coding bound is given by high-rate theory [33]. High-rate theory considers finite blocks of samples under the assumption of high rate or small quantization errors. In general, this theory requires an asymptotically high rate, such that the pdf of the source is close to uniform over each decision region. High-rate theory for scalar coding was introduced in, and the generalization and development for blocks of samples, i.e. for vector quantization, are much due to.

T-H-4205

Rate-distortion theory could be considered impractical as it does not give any guidelines for how to design tractable coders, and blocks of infinite length are usually required for optimal performance. High-rate theory considers finite block lengths, and the point density gives a hint on how to distribute code vectors. Furthermore, coders with rather low rates have shown to follow the rules of high-rate theory [33]. Thus, high-rate theory is attractive as a tool for designing practical system.

## **Chapter 4**

---

# **Vector Quantization**

## 4. Vector Quantization - overview

*Vector quantization* (VQ) has since about 1980 become a popular technique for source coding of image and speech data. The popularity of VQ is motivated primarily by the theoretically optimal performance; no other source coding technique at equivalent delay can achieve better performance than optimal VQ. However, direct use of VQ suffers from a serious complexity barrier. Many authors have proposed constrained VQ structures to overcome the complexity, for example *multistage VQ* [1], *tree-structured VQ* [2-5], *vector sum VQ* [6], *gain-shape VQ* [7], etc. Each of these solutions has disadvantages, in most cases a reduced performance.

### 4.1 Definitions

A VQ  $Q$  of size  $N$  and dimension  $d$  is a mapping from a vector in the  $d$ -dimensional Euclidean space  $\mathbb{R}^d$  into a finite reproduction set  $C = \{c_1, c_2, c_3, \dots, c_N\}$

$$Q : \mathbb{R}^d \rightarrow C \quad \text{.....} \quad 4.1$$

The set  $C$ , denoted the *codebook*, contains  $N$  *codevectors*  $c_k$   $k = 1, 2, \dots, N$ , each a vector in  $\mathbb{R}^d$ . The index  $k$  of the codevectors is denoted *codeword*. The rate  $R$  of the quantizer is defined as  $\log_2 N/d$  [bits per sample]. The definition of  $Q$  in (2.1) partitions  $\mathbb{R}^d$  into  $N$  disjoint regions, each with a corresponding codevector  $c_k$ .

The vector quantizer can be decomposed in two components, the encoder and the decoder. The encoder  $E$  maps from  $\mathbb{R}^d$  to the index set  $I = \{1, 2, \dots, N\}$

$$E : \mathbb{R}^d \rightarrow I \quad \text{.....} \quad 4.2$$

and the decoder  $D$  maps the index set into the reproduction set  $C$ , i.e.,

$$D : I \rightarrow \mathbb{R}^d \quad \text{.....} \quad 4.3$$

With this notation, the quantization operation can be written as a cascade of the encoder and decoder:

$$Q(x) = D(E(x)) \quad \text{.....} \quad 4.4$$

The mean squared error criterion is only one of many possible distortion measures, but it has the advantage of being widely used and is mathematically simple.

$$MSE = E[x - Q(x)]^2 \quad \text{.....} \quad 4.5$$

## 4.2 Optimality conditions

In VQ design, the aim is to find encoder and decoder rules to minimize the chosen distortion measure. For the squared Euclidean distance measure (with a decoder  $D(i) = \mathbf{c}_i$ ), it can be readily shown that for a fixed partition  $\Omega_k$  of the input space, the codevectors  $\mathbf{c}_k = \{ \mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \dots, \mathbf{c}_N \}$  should be chosen as the centroid of the vectors in the region,  $\mathbf{c}_k = E[\mathbf{x} | \mathbf{x} \in \Omega_k]$  to minimize the expected distortion is often called *the centroid condition*. If instead the set of codevectors is fixed, the partition should be the *nearest neighbor partition*.

both the encoder and the decoder are completely specified by the codebook  $C$ , so finding optimal encoder and decoder rules is equivalent to finding the optimum set of codevectors  $\{ \mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \dots, \mathbf{c}_N \}$

The centroid condition and the nearest neighbor partition are necessary but not sufficient for a VQ to be optimal in the mean square sense. Sufficient conditions for a globally optimal VQ have never been presented (except for some special cases), and a quantizer fulfilling the necessary conditions may be far from optimal. This makes VQ design a delicate problem.

The new definition of the nearest neighbor partition shows that to find the optimum codevector to a given input vector  $\mathbf{x}$ , it suffices to find a codevector whose Voronoi neighbors all have greater distance to the input vector.

## 4.3 Brief description

### 4.3.1 Some main advantages:

- Exploit dependency that may exist within an input vector.
- Ability to generate non-cubic multi-dimensional partitions of input which provides better compaction of the input space
- Ability to track high order statistical characteristics of the input

### 4.3.2 Some main disadvantages:

- Encoding complexity and memory requirements increase exponentially with vector size (under a given rate) and with bit rate
- Lack of robustness: sensitivity to channel noise

- Conventional VQ is severely limited to modest vector and codebook size Different more robust methods needed

#### 4.3.3 Practical constraints:

- Encoding complexity and memory requirements increase exponentially with vector size (under a given bit- rate) and with bit-rate
- Codebook grows exponentially as a function of vector size  $N$  and bit-rate  $r$ .
- Other problem is lack of robustness and sensitivity to channel noise

#### 4.4 Quantization

Quantization is a mature topic that has been studied for many years. Early works in this area addressed the quantization of 1-D. interestingly; much of this work is relevant in the context of quantizing images. This simple case is scalar quantization. Here the dynamic range of the samples is identified and level is assigned to cover the range. A more advances form of quantization is called vector quantization. It involves extra blocks or groups of pixels from within the image and quantizing then as separate entities.

Image compression using vector quantization (VQ) is a lossy compression technique. It is defined as mapping  $Q$  of  $K$ -dimensional Euclidean space  $R^k$  into a finite subset  $Y$  of  $R^k$  Thus,

$$Q=R^k \rightarrow Y \text{ .....(4.6)}$$

where  $y=(x_i; i=1,2,\dots,N)$  is the set of reproduction vectors and  $N$  is the number of vectors in  $Y$ .

A vector quantizer is composed of two parts, encoder and decoder. An encoder will compare each input vector with every codevector in the codebook and generate index which represent the minimum distortion codevector from the input vector. A decoder takes the index's to locate the codevector in codebook and generate the output vectors.



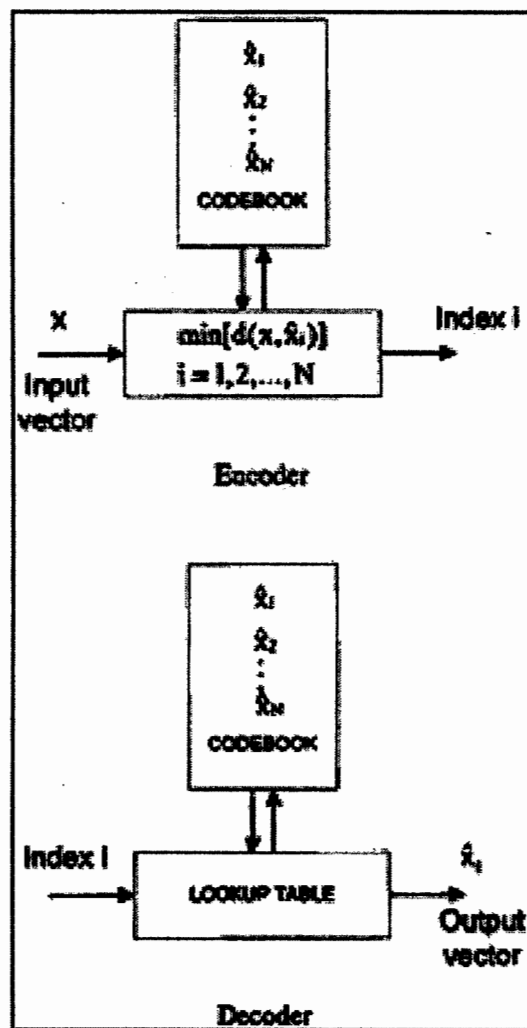


Figure 4.1 Block diagram of VQ

A codebook is the set of finite codevector for representing the input vector. The popular technique in codebook design is the Linde-Buzo-Gray (LBG) algorithm [4]. The whole image a partitioned into subblocks and all subblocks are used to training this codebook.

#### 4.4.1 Fundamental of Vector Quantization

A quantizer considered so far perform quantization on individual pixels. This quantizer can be generalized by considering new symbols composed of groups of pixels. The procedure for extracting vectors from an image typically consists of partitioning the image into contiguous blocks and then un-wrapping the pixels within the blocks to form vectors. These extracted vectors are then coded by comparing them to codevectors stored in codebook. Assume that the input vector extracted from the image is  $x$  and the codebook consist of  $N$  codevectors, here  $i=0,1,\dots,N-1$ , the comparison consist of computing the distortion  $d(x, \hat{x}_i)$  for all  $i$ , and choosing as the codevector.

### 4.4.2 Vector Quantizer Design

The necessary conditions for optimality provide the basis for iteratively improving a given vector quantizer if the iteration continues to convergence a good ( hopefully close to optimal )quantizer can be found the iteration begins with a vector quantizer consisting of its codebook and the corresponding optimal (NN ) partition and then finds the new codebook which is optimal for that partition this new codebook and its NN partition are then a new vector quantizer with average distortion no greater (and usually less )than the original quantizer Although each of these steps of optimizing a partition for a codebook and a codebook for a partition is simple and straightforward the simultaneous satisfaction of both conditions is not easy to achieve there are no known closed form solutions to the problem of optimal quantization the repeated application of the improvement step however yields an iterative algorithm which at least reduces ( or leaves unchanged) the average distortion at each step in effect we design a sequence of vector quantizers which continually improve in performance if the algorithm is effective We begin with the problem of obtaining the initial codebook for improvement since this too is a problem of vector quantizer design in fact if the initial codebook is good enough it may not be worth the effort to run further improvement algorithms there are a variety of techniques for generating a codebook that have been developed in cluster analysis (for pattern recognition )and in vector quantization (for signal compression )We survey several of the most useful.

#### 4.4.2.1 Random Coding

Perhaps the simplest conceptual approach towards filling a codebook of  $N$  code word is to randomly select the code words according to the source distribution, which can be viewed as a MonteCarlo codebook design. The obvious variation when designing based on a training sequence is to simply select the first  $N$  training vectors as code words if the data highly correlated, it will likely produce a better codebook if one takes say every  $K^{\text{th}}$  training vector . Tills technique has often been used in the pattern recognition literature and was used in the original development of the k-means technique. one can be somewhat more sophisticated and randomly generate a codebook using not the input distribution but the distribution which solves the optimization problem defining Shannon s distortion rate function. In fact, the Shannon source coding theorems imply that such a random selection will on the average yield a good code. Unfortunately the codebook will have no useful structure and may turn out quite awful. Observe that here random coding means

only that the codebook is selected at random once selected it is used in the usual deterministic (nearest neighbor) fashion this is the same sense that random is used in information theory

#### 4.4.2.2 Pruning

Pruning refers to the idea of starting set and selectively eliminating (pruning) training vectors as candidate code vectors until a final set of training vectors remains as the codebook. In one such method a sequence of training vectors is used to populate a codebook recursively as follows put the first training vector in the codebook. Then compute the distortion between the next training vector and the first code word. If it is less than some threshold, continue. If it is greater than the threshold, add the new vector to the codebook as codeword. With each new training vector, find the nearest neighbor in the code book. If the resulting distortion is not within some threshold, add the training vector to the codebook. Continue in this fashion until the codebook has enough words. A typical choice of threshold value for the MSE distortion measure is proportional to  $N^{-2/k} = 2^{-2r}$  where  $r$  is the rate of code. This technique is well known in the statistical clustering literature.

#### 4.4.2.3 Pairwise Nearest Neighbor Design

A more complicated, but better, means of finding a codebook from a training sequence is the pairwise nearest neighbor (PNN clustering proposed by Equitz) similar algorithms have also been used in the clustering literature. This is also a form pruning as it begins with the entire training sequence of  $L$  vectors, and ends with collection of  $N$  vectors, unlike previous design technique, however, the final vector need not to be in the training sequence. The technique involves more computation than the preceding methods, but it is faster than the Generalized Lloyd Algorithm which attempts to optimize the codebook.

Suppose that training sequence has  $L$  vectors, each of which is considered to be a separate cluster containing a single vector. The goal will be to merge vectors together into groups or clusters until we have the desired number, say  $N$ . The codebook will then contain the centroids of these clusters. In this manner we will have a partition of training sequence into the correct number of cells and have the optimal codebook for this partition. The partition might be a nearest, neighbor partition, however. The induced vector quantizer would then replace this partition by the NN partition. The partition is contained as follows. First compute the distortion between all pairs of vectors. The two training vectors having the smallest distortion are combined into a single cluster and

represented by their centroid. We now have  $L-1$  clusters, one containing two vectors and the rest containing single vector. at each step clusters may have more than one vector. Suppose now that we have  $K$  clusters with  $N < K \leq L-1$  and we wish to merge two of the clusters to have a good set of  $K-1$  clusters. This single step merging can be done in an optimal fashion as follows: for every pair of clusters drawn from the full collection of  $K$  clusters, compute the increase in average distortion resulting if the two clusters and their centroids are replaced by the merged two clusters and corresponding centroids are replaced by the merged two clusters and the corresponding centroid. This computation is much easier than it sounds in the case of squared error. When the best pair of clusters for merging is found, they are merged to form a codebook with  $K-1$  vectors. Continue in this way until only  $N$  vectors remain. Thus, for example, each of the  $K$  pairs of clusters consists of two vectors  $R_i = \{x_i(l); l=1,2,\dots,L_i\}$  and  $R_j = \{x_j(l); l=1,2,\dots,L_j\}$ . The contribution of these two clusters to the average distortion if they are not merged is

$$\Delta_{ij} = \sum d(x_i(l), \text{cent}(R_i)) + \sum d(x_j(l), \text{cent}(R_j)) \quad (4.7)$$

while the contribution if they are merged is

$$\Delta'_{ij} = \sum d(x_i(l), \text{cent}(R_i \cup R_j)) + \sum d(x_j(l), \text{cent}(R_i \cup R_j)) \geq \Delta_{ij} \quad (4.8)$$

The pair of clusters  $R_i, R_j$  for which  $\Delta'_{ij} - \Delta_{ij}$  is the smallest is merged. That is two clusters are merged which thereby cause the least increase in the overall average distortion. Each merge is optimal but the overall procedure need not be optimal, need not produce the optimal codebook of the given size.

#### 4.4.2.4 Product Codes

In some cases a product codebook may provide a good initial guess. for example if one wishes to design a codebook for a  $k$  dimensional VQ with codebook size  $2^{kr}$  for some integral resolution  $R$ , then one can use the product of  $k$  scalar quantizers with  $2^R$  words each. Thus if  $q(x)$  is scalar quantizer, then  $Q(x_0, \dots, x_{k-1}) = (q(x_0), \dots, q(x_{k-1}))$ , the Cartesian product of the scalar quantizers, is vector quantizer. this technique will not work if  $R$  is an integer. In general other product structures can be used e.g. one could first design a one dimensional quantizer  $q_1$  from scratch (perhaps using a uniform quantizer as an initial guess) one could then use  $(q(x_0), q(x_1))$  as an initial guess to design a good two dimensional quantizer  $q_2(x_0, x_1)$  one could then initiate a three dimensional VQ design with the product  $(q_1(x_0), q_2(x_1, x_2))$  as an initial guess One could continue in this way to construct higher Dimensional quantizers until the final size reached.

#### 4.4.2.5 Splitting

Linde et al. introduced a technique that resembles the product code initialization in that it grows large codebook from small ones, but differs in that it does not require an integral number of bits per symbol. The method is called splitting algorithm and it produces increasingly larger codebooks of a fixed dimension the globally optimal resolution 0 codebook of a training sequence is the centroid of the entire sequence. The one code word say  $y_0$ , in this codebook can be "split" into two code words  $y_0$  and  $y_0 + \epsilon$  where  $\epsilon$  is a vector of small Euclidean norm. One choice of  $\epsilon$  is to make it proportional to the vector whose  $i$ th component is the standard deviation of the  $i$ th component of the set of training vectors. Another choice is to make it proportional to the eigenvector, corresponding to the largest eigen value of the covariance matrix of the training set. This new codebook has two words and can be no worse than the previous codebook since it contains the previous codebook. The iterative improvement algorithm can be run on this codebook to produce a good resolution 1 code. When complete, all of the code words in the new codebook can be split, for an initial guess for a resolution 2 codebook. One continues in this manner, using a good resolution  $r$  codebook to form an initial resolution  $r+1$  codebook by splitting. This algorithm provides a complete design technique from scratch on a training sequence and will later be seen to suggest a vector analog to successive approximation quantizers.

#### 4.5 The LBG Algorithm

The classical method for designing VQ codebooks is the generalized Lloyd algorithm, so named because it is the vector extension of Lloyd's scalar quantizer design algorithm. In the data compression literature, this algorithm is also known as the LBG algorithm, after Linde, Buzo, and Gray, whose landmark paper on VQ popularized this method. Interestingly, the same design algorithm is known in the pattern recognition literature as the k-means algorithm. The intended applications associated with the k-means and LBG algorithms are different, but the algorithms are essentially the same. The first step in the design algorithm involves obtaining a training set. This is done by collecting a set of representative images and extracting training vectors that are considered to be typical and representative of the vectors that will be input to the system. To achieve good performance over a wide range of images, the training set should be large. As a rule of thumb, a (least 1000) training vectors should be used to design a codebook with  $N$  code-vectors.

The LBG design procedure is a two-step process that can be illustrated using a simple example. Assume we wish to design a codebook with three code-vectors. For simplicity, let us assume the vectors only have two elements, (i.e. a two-dimensional vector). In a sense, these initial code-vectors represent our best guess of the codebook. Many techniques have been suggested for obtaining an initial codebook, such as the splitting algorithm, the min-max algorithm, and the pair-wise nearest neighbor. (PNN) algorithm. Perhaps the simplest of them is to randomly take  $N$  vectors from the training set and use them as the initial vectors.

#### 4.5.1 Basic steps of LBG:

*Step 1:*

Start with a training set of vectors (get a large quantity of representative vectors: train on one set, test with others).

*Step 2:*

Start with an initial codebook of size  $M$  (selected from training set); example: randomly selected vectors from training set.

*Step 3:*

Vector quantize each training vector using current codebook (cluster training data).

*Step 4:*

Use centroid of clusters as the updated codebook

- centroid = mean of cluster for mse and for a stationary and ergodic input since time/ space averages replace statistical averages
- centroid = center of mass

*step 5:*

Repeat from Step 3 until distortion between old and new codebook is smaller than a selected small threshold

#### 4.5.2 Remarks on LBG:

LBG guaranteed to converge and finds a locally optimal quantizer for a training set (may not be locally optimal for the input  $x$ ).

Final resulting codebook depends on initial choice, algorithm influenced by choice of initial codebook (cluster centers), and by the choice and geometrical properties of training data.

Local optimal design for fixed number of levels  $M$ . In coding, VQ usually used in conjunction with entropy coding.

- Limit the entropy of quantized signal rather than number of quantization levels in the design process
- Entropy constrained VQ (EC-VQ)

#### **4.5.3 Initialization in LBG**

Most important issue since it can significantly affect the performance of designed codebook. Several codebook initialization methods proposed.

Popular ones:

1. Random selection from training set
2. Binary splitting for LBG codebook design

uses fixed perturbations of the current code vectors (centroids) to create more code vectors: twice as many at each step.

## **Chapter 5**

---

# **Lossless Using EC-RVQ**



## 5. Lossless using EC-RVQ

Loss less compression of image data require two consecutive steps

- Decorrelation
- Coding

In decorrelation step the redundancy resulting from dependency among the pixels or data values is reduced through the use of various mapping techniques because of mapping distortions. The extent of decorrelation may be limited in applications where error free reconstruction is required (its measuring criteria).

In the coding step, the memory less entropy coding is widely used to exploit the statistical redundancy of the data. Information theory [30] indicates that the coding efficiency can be improved by using higher order conditional entropy coding.

### 5.1 Definitions

Some definition concerning with conditional constrain model.

#### 5.1.1 Definition of Probability

Let  $S = (s_1, \dots, s_n)$  a finite-length sequence with  $|S| = n$  over  $A = \{a_1, \dots, a_m\}$ . Also let  $|S|a_i$  the frequency of  $a_i$  in  $S$ . Then we define  $P(a_i) := |S|a_i / n$  as the PROBABILITY of  $a_i$  (in  $S$ ).

From the definition, we can directly conclude that  $P(a_i)$  is always contained in the interval  $[0,1)$  for any symbol, whereas the sum over all such probabilities is always  $\sum P(a_i) = 1$ .

#### 5.1.2 Definition of Model

Let  $A$  an alphabet. A MODEL  $M$  is a function

$$M : A \rightarrow [0,1) : a_i \rightarrow PM(a_i),$$

which maps a probability  $P_M(a_i)$  to each symbol  $a_i \in A$  [19].

#### 5.1.3 Definition of Entropy

Let  $S$  a sequence over alphabet  $A = \{a_1, \dots, a_m\}$ . The ENTROPY  $H_M(S)$  of the sequence  $S$  under model  $M$  is defined as

$$H_M(S) = \sum P(a_i) \log_2 (1/P_M(a_i)) \quad \text{where } i=1, \dots, m \dots \dots \dots (5.1)$$

The unit of the entropy is [bits/symbol] because the formula only refers to probabilities as relative frequencies rather than absolute ones.

Considering a model as perfect, one obtains the *correct* probability distribution leading to the natural form of the entropy:

$$H(S) = \sum P(a_i) \log_2 (1/P(a_i)) \quad (a \in A) \dots \dots \dots (5.2)$$

A natural measure of how much information is contained in a given sequence of data is called the ENTROPY.[19] This kind of entropy is depended on the input data only and *no* subject to interpretation. However the interested reader might wish to know that most of the literature about Arithmetic Coding.

## 5.2 Entropy

It is well-known that the minimum bit rate required for representing a source is determined by the entropy of the source. Two commonly used digital source models in image coding are the statistically independent source and the Markov source, which can be measured by memory-less entropy and high-order conditional entropy, respectively.

### 5.2.1 Memory-less Entropy

For an independent source, the entropy is given by

$$H(X) = -\sum P(x) \log_2 P(x) \quad \text{.....(5.3)}$$

Where the unit of  $H(X)$  is bit/pixel. The entropy  $H(X)$  establishes a lower bound on the average bit rate for a memoryless source, and gives a measure of the amount of information carried by random variable  $X$ . However, for most of the image sources, there always exists some kind of statistical dependency among the neighboring pixels, even if the source has been decorrelated. This kind of redundancy cannot be reduced simply by applying memoryless entropy coding.

### 5.2.2 High-Order Entropy

In high-order block entropy coding, a block of pixels can be combined and coded by a single code. For an  $L$ -dimensional vector  $\mathbf{x}$ , a measure of the average number of bits that would be required to encode each block of  $L$ -tuple is defined as

$$H_L(\mathbf{x}) \Delta H_L(X_1, \dots, X_L) = -\sum P(\mathbf{x}) \log_2 P(\mathbf{x}) \quad \text{.....(5.4)}$$

where  $X$  is a random vector with possible values  $\mathbf{x}$  obeying probability  $P(\mathbf{x})$ . The summation is over all the  $P$  possible  $L$ -tuples for each pixel with  $K$ -bit quantization.

Conditional coding is another form of high-order entropy coding in which one assumes that  $L-1$  components from  $L$ -dimensional vector  $\mathbf{x}$ , i.e.,  $x_1, x_2, \dots, x_{L-1}$ , have already been received by the receiver. Component  $x_L$  then can be coded more efficiently by utilizing this conditional information. Conditional entropy be written as:

$$H(X_L|X_1, \dots, X_{L-1}) = -\sum P(\mathbf{x}) \log_2 P(x_L|x_1, \dots, x_{L-1}), \quad \text{.....(5.5)}$$

and can be shown to have the following property:

$$H(X_L|X_1, \dots, X_{L-1}) \leq H(X_{L-1}|X_1, \dots, X_{L-2}) \leq H(X_2|X_1) \leq H(X_1) \Delta H(X) \quad \text{.....(5.6)}$$

Thus, the higher the order of conditional probability, the lower the resulting entropy. Note that Equation (6) is satisfied with the equality if and only if the pixels are statistically independent.

### 5.3 Codebook Design

In the Image data compression process VQ type, Codebook is an essential part of the image quality, because the image quality is the result of the comparison between Training Vectors with Codebook. Compressing image data by using Vector Quantization (VQ) [26]-[28], will compare Training Vectors with Codebook. The result is an index of position with minimum distortion. The implementing Random Codebook will reduce the image quality.

The Random Codebook and Split Codebook [9]. Split Codebook is obtained from the average of the Training Vectors, and then split the average into 2 vectors. And then one vector is added by the average value from itself. While the other one is subtracted as shown in figure 2, observe the average distortion which is the result of the comparison between the Training Vectors and each of the two vectors whether it is equal, less than, or more than the decision value. If it is more than the decision value, each vector will be split into two. Otherwise, stop, and follow the VQ process. Split Codebook is the average amount of Training Vectors population. The result is reducing scatter data better than random codebook. Where as Searching for the best Codebook from studying the training set, Linde-Buzo-Gray's Algorithm (LBG) is the famous algorithm for Codebook and M-Search algorithm leading to approximately 60 vectors Lagrangian calculation per input vector [1]. to reduce the complexity, non exhaustive stage searching algorithm are usually used, leading to good balance between complexity and encoding accuracy. In particular, the dynamic M-search algorithm [29], which is shown to generally perform better than conventional M-search algorithm, is used to search CEC-RVQ [1].

### 5.4 VQ using Entropy Constrained

By generalizing the entropy-constrained scalar quantization design to the vector case, introduce the interactive descent algorithm for the design of entropy constrained vector quantizer (EC-VQ)[7]. Chou applied EC-VQ to image coding and showed that the entropy constrained optimization yield a significant performance gain, more recently, the entropy-constrained optimization was applied to residual VQ, which also known as

multistage VQ. [21][22]. This form of VQ, which is called entropy constrained residual VQ(EC-RVQ) [23]-[25], consist of cascade of VQ stages, where each stage operates on the input/output difference of previous stage. The individual stage symbols are entropy coded based on model using probabilities that are conditioned on previous stage output symbols.

For the number of reasons stemming from the memory and computationally efficient structure of RVQ, EC-RVQ can outperform EC-VQ and usually achieves image compression results competitive with those of JPEG and sub band coding [23], [24].

Like EC-VQ, EC-RVQ is a memory less vector quantizer. This is because the EC-RVQ design algorithm [23]-[25], minimizes the distortion subject to constraints on the first order or zero order entropy of vector quantizer input.

However better rate distortion performance can generally be achieved by incorporating memory into the vector quantizer.

The extend, EC-RVQ to a vector quantizer that exploit the memory by using higher order conditional entropy codes. Unlike EC-VQ, which conditions on the output of the previous stages, the higher order conditional EC-RVQ introduce here [7], takes advantage of the information available in previously coded vectors by conditional over the spatial stage region of support. While conditional EC-VQ is severely impaired by the exponential dependence of memory and complexity on the number of conditioning symbols and the VQ codebook size, CEC-RVQ is not as sensitive.

Entropy constrained residual vector quantization (EC-RVQ) has been shown to be a competitive compression technique. Its design procedure is an iterative process which typically consists of three steps: encoder update, decoder update, and entropy coder update. a new algorithm for the EC-RVQ was designed [11]. The main features of this algorithm are: ( i ) In the encoder update step, we propose a variation of the exhaustive search encoder that significantly speeds up encoding at no expense in terms of the rate-distortion performance. ( ii ) In the decoder update step, a new method that simultaneously updates the codebooks of all stages. The method is to form and solve a certain least square problem and we show that both tasks can be done very efficiently. ( iii ) The Lagrangian of rate-distortion decreases at every step and thus this guarantees the convergence of the algorithm.

The well known generalized BFOS algorithm used to prune the tree to find the best stage order subject to a limit on the number of conditional probabilities, used as a measure of

performance [1]. The PNN algorithm was shown to be successful in reducing the size of stage statistical model by one order of magnitude.

Residual VQ has been implied not only for data as well as video compression. A correspondence builds on the asymptotic closed-loop approach to predictive vector quantizer design, and extends it to the design of predictive multistage vector quantizer for low bit rate video coding. The design approach resolves longstanding shortcomings, in particular, design stability and empty-cell problems. Simulation results show substantial gains over traditional design approaches [10].

### 5.5 Existing Frame Work

The hybrid technique of quantization and entropy coding of the residual signal has been shown to yield good compression performance. This is due to the fact that quantization often produces a structure where high order statistical dependences can be exploited. Moreover the output alphabet of the quantizer can be made smaller than that of the original signal; the complexity of high order statistical modeling is reduced. This is especially the case when structurally constrained quantizers are employed. In particular, the structure of the multistage residual vector quantization (RVQ) used to be very successful in providing more accurate estimates of the statistical modeling. Multistage RVQ produces multi resolution approximation of the input signal, and allows high order statistical conditioning to be performed between the stage sub-signals.

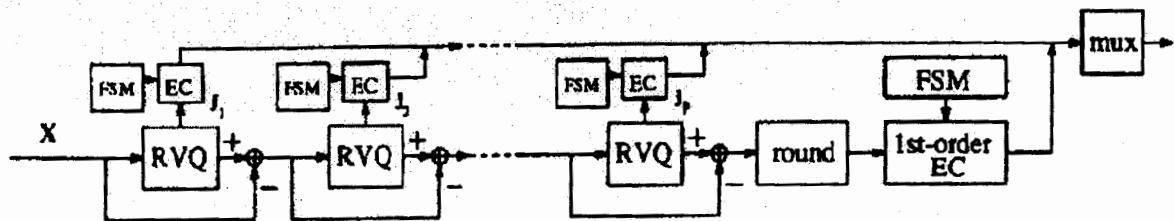


Figure 5.1. High level structure of P-Stage EC-RVQ encoder

As shown in figure 1, EC-RVQ employs a quantize the input signal, where the output of the stage RVQ is then fed to statistical-model-driven entropy coder (EC). The high order stage statistical model is represented by a finite state machine (FSM) where the state transitions are based on previously coded symbols. The quantized signal is rounded to the nearest integer, and the residual signal, formed by subtracting the rounded quantized signal from the original one, is then coded using first order entropy coder.

Empirical work has shown that using higher order entropy coding does not lead to significant reduction in output entropy of the residual signal.

### 5.5.1 Uniqueness of Existing Framework

There are two important ideas, unique to this framework, that exemplify the novelty of this lossless approach.

First, since the overall system is lossless it is potentially better to employ entropy of the residual signal as a distortion measure in the design of the CEC-RVQ. Using conventional distortion measures such as square error measure does not lead to minimization of the residual entropy. To elaborate, let  $x$  be the input and  $x'$  be the output of the CEC-RVQ. The new distortion measure used in the design of the CEC-RVQ is  $d(x, x') = -\log_2[\text{pr}(I(x-x'))]$ , where  $I(a)$  is the integer closest to the real  $a$ . The distortion is essentially the self-information of the integer converted residual signal, and is used as estimate of the length of the codeword that would be used to encode the symbol  $I(x-x')$ . In other words CEC-RVQ designed to minimize such a distortion measure also minimizes the entropy of the residual signal.

The second idea is that only entropy is a measure of performance. Since the distortion measure is the entropy, the CEC-RVQ design algorithm produces an operational entropy-entropy curve where each point represents a pair of entropies.

## **Chapter 6**

---

# **Entropy Coding**

## 6. Entropy Coding

The state of the art in data compression is arithmetic coding, not the better known Huffman method. Arithmetic coding gives greater compression, is faster for adaptive models, and clearly separates the model from the channel encoding.

Arithmetic coding is superior in most respects to the better-known Huffman method. It represents information at least as compactly-sometimes considerably more so. Its performance is optimal without the need for blocking of input data. It encourages a clear separation between the model for representing data and the encoding of information with respect to that model. It accommodates adaptive models easily and is computationally efficient. Yet many authors and practitioners seem unaware of the technique. Indeed there is a widespread belief that Huffman coding cannot be improved upon.

We aim to rectify this situation by presenting an accessible implementation of arithmetic coding and by detailing its performance characteristics. We start by briefly reviewing basic concepts of data compression and introducing the model-based approach that underlies most modern techniques. We then outline the idea of arithmetic coding using a simple example, before presenting programs for both encoding and decoding. In these programs the model occupies a separate module so that different models can easily be used. Next we discuss the construction of fixed and adaptive models and detail the compression efficiency and execution time of the programs, including the effect of different arithmetic word lengths on compression efficiency. Finally, we outline a few applications where arithmetic coding is appropriate.

### 6.1 Data Compression

To many, data compression conjures up an assortment of ad hoc techniques such as conversion of spaces in text to tabs, creation of special codes for common words, or run-length coding of picture data (e.g., see [14]). This contrasts with the more modern model-based paradigm for coding, where, from an input string of symbols and a model, an encoded string is produced that is (usually) a compressed version of the input. The decoder, which must have access to the same model, regenerates the exact input string from the encoded string. Input symbols are drawn from some well-defined set such as the ASCII or binary alphabets; the encoded string is a plain sequence of bits. The model is a way of calculating, in any given context, the distribution of probabilities for the next input



symbol. It must be possible for the decoder to produce exactly the same probability distribution in the same context. Compression is achieved by transmitting the more probable symbols in fewer bits than the less probable ones.

For example, the model may assign a predetermined probability to each symbol in the ASCII alphabet. No context is involved. These probabilities can be determined by counting frequencies in representative samples of text to be transmitted. Such a fixed model is communicated in advance to both encoder and decoder, after which it is used for many messages.

Alternatively, the probabilities that an adaptive model assigns may change as each symbol is transmitted, based on the symbol frequencies seen so far in the message. There is no need for a representative sample of text, because each message is treated as an independent unit, starting from scratch. The encoder's model changes with each symbol transmitted, and the decoder's changes with each symbol received, in sympathy.

More complex models can provide more accurate probabilistic predictions and hence achieve greater compression. For example, several characters of previous context could condition the next-symbol probability. Such methods have enabled mixed-case English text to be encoded in around 2.2 bits/character with two quite different kinds of model. Techniques that do not separate modeling from coding so distinctly, like that of Ziv and Lempel, do not seem to show such great potential for compression, although they may be appropriate when the aim is raw speed rather than compression performance[17].

The effectiveness of any model can be measured by the entropy of the message with respect to it, usually expressed in bits/symbol. Shannon's fundamental theorem of coding states that, given messages randomly generated from a model, it is impossible to encode them into less bits (on average) than the entropy of that model [16].

A message can be coded with respect to a model using either Huffman or arithmetic coding. The former method is frequently advocated as the best possible technique for reducing the encoded data rate. It is not. Given that each symbol in the alphabet must translate into an integral number of bits in the encoding, Huffman coding indeed achieves "minimum redundancy." In other words, it performs optimally if all symbol probabilities are integral powers of  $\frac{1}{2}$ . But this is not normally the case in practice; indeed, Huffman coding can take up to one extra bit per symbol. The worst case is realized by a source in which one symbol has probability approaching unity. Symbols emanating from such a source convey negligible information on average, but require at least one bit to transmit

[15]. Arithmetic coding dispenses with the restriction that each symbol must translate into an integral number of bits, thereby coding more efficiently. It actually achieves the theoretical entropy bound to compression efficiency for any source, including the one just mentioned.

In general, sophisticated models expose the deficiencies of Huffman coding more starkly than simple ones. This is because they more often predict symbols with probabilities close to one, the worst case for Huffman coding. For example, the techniques mentioned above that code English text in 2.2 bits/ character both use arithmetic coding as the final step, and performance would be impacted severely if Huffman coding were substituted.

## 6.2 The Idea of Arithmetic Coding

In arithmetic coding, a message is represented by an interval of real numbers between 0 and 1. As the message becomes longer, the interval needed to represent it becomes smaller, and the number of bits needed to specify that interval grows. Successive symbols of the message reduce the size of the interval in accordance with the symbol probabilities generated by the model. The more likely symbols reduce the range by less than the unlikely symbols and hence add fewer bits to the message, detail practical implementation can be reference [14][18].

Before anything is transmitted, the range for the message is the entire interval  $[0, 1)$ , denoting the half-open interval  $0 \leq x < 1$ . As each symbol is processed, the range is narrowed to that portion of it allocated to the symbol. For example, suppose the alphabet is  $\{a, e, i, o, u, !\}$ , and a fixed model is used with probabilities shown in Table I.

Table I. Example Fixed Model for Alphabets  $\{a,e,i,o,u,! \}$

Symbol	Probability	Range
a	0.2	[0, 0.2)
e	0.3	[0.2, 0.5)
i	0.1	[0.5, 0.6)
o	0.2	[0.6, 0.8)
u	0.1	[0.8, 0.9)
!	0.1	[0.9, 1.0)

Imagine transmitting the message *eaii!*. Initially, both encoder and decoder know that the range is  $[0, 1)$ . After seeing the first symbol, *e*, the encoder narrows it to  $[0.2, 0.5)$ , the range the model allocates to this symbol. The second symbol, *a*, will narrow this new range to the first one-fifth of it, since *a* has been allocated  $[0, 0.2)$ . This produces  $[0.2, 0.26)$ , since the previous range was 0.3 units long and one-fifth of that is 0.06. The next symbol, *i*, is allocated  $[0.5, 0.6)$ , which when applied to  $[0.2, 0.26)$  gives the smaller range  $[0.23, 0.236)$ . Proceeding in this way, the encoded message builds up as follows:

<b>Initially</b>	$[0, 1)$
<b>After seeing e</b>	$[0.2, 0.5)$
<b>a</b>	$[0.2, 0.26)$
<b>i</b>	$[0.23, 0.236)$
<b>i</b>	$[0.233, 0.2336)$
<b>!</b>	$[0.23354, 0.2336)$

Figure 1 shows another representation of the encoding process. The vertical bars with ticks represent the symbol probabilities stipulated by the model. After the first symbol has been processed, the model is scaled into the range  $[0.2, 0.5)$ , as shown in Figure 1a. The second symbol scales it again into the range  $[0.2, 0.26)$ . But the picture cannot be continued in this way without a magnifying glass! Consequently, Figure 1b shows the ranges expanded to full height at every stage and marked with a scale that gives the endpoints as numbers.

Suppose all the decoder knows about the message is the final range,  $[0.23354, 0.2336)$ . It can immediately deduce that the first character was *e!* since the range lies entirely within the space the model of Table I allocates for *e*. Now it can simulate the operation of the encoder:

<b>Initially</b>	$[0, 1)$
<b>After seeing e</b>	$[0.2, 0.5)$

This makes it clear that the second character is *a*, since this will produce the range

<b>After seeing a</b>	$[0.2, 0.26)$
-----------------------	---------------

which entirely encloses the given range [0.23354, 0.2336). Proceeding like this, the decoder can identify the whole message. It is not really necessary for the decoder to know both ends of the range produced by the encoder.

Instead, a single number within the range--for example, 0.23355--will suffice. (Other numbers, like 0.23354, 0.23357, or even 0.23354321, would do just as well.) However, the decoder will face the problem of detecting the end of the message, to determine when to stop decoding. After all, the single number 0.0 could represent any of a, aa, aaa, aaaa, ..... To resolve the ambiguity, we ensure that each message ends with a special EOF symbol known to both encoder and decoder. For the alphabet of Table I, will be used to terminate messages, and only to terminate messages. When the decoder sees this symbol, it stops decoding.

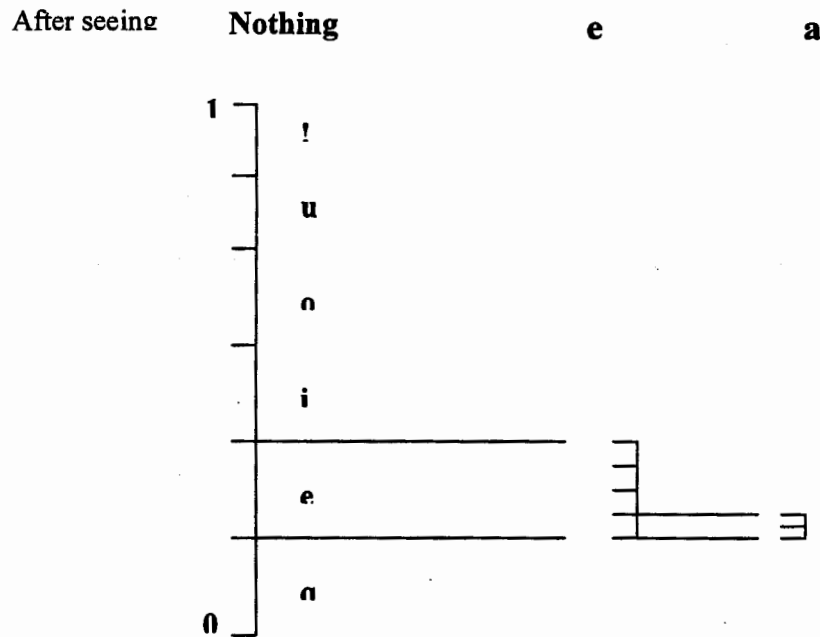


Figure 6.1.a Representation of Arithmetic coding process

Relative to the fixed model of Table I, the entropy of the five-symbol message eaii! Is

$$\begin{aligned}
 & -\log 0.3 - \log 0.2 - \log 0.1 - \log 0.1 - \log 0.1 \\
 & = -\log 0.00006 \approx 4.22
 \end{aligned}$$

(using base 10, since the above encoding was performed in decimal). This explains why it takes five decimal digits to encode the message. In fact, the size of the final range is  $0.2336 - 0.23354 = 0.00006$ , and the entropy is the negative logarithm of this figure. Of course, we normally work in binary, transmitting binary digits and measuring entropy in bits. Moreover, as noted earlier, more sophisticated models give much better performance in general.

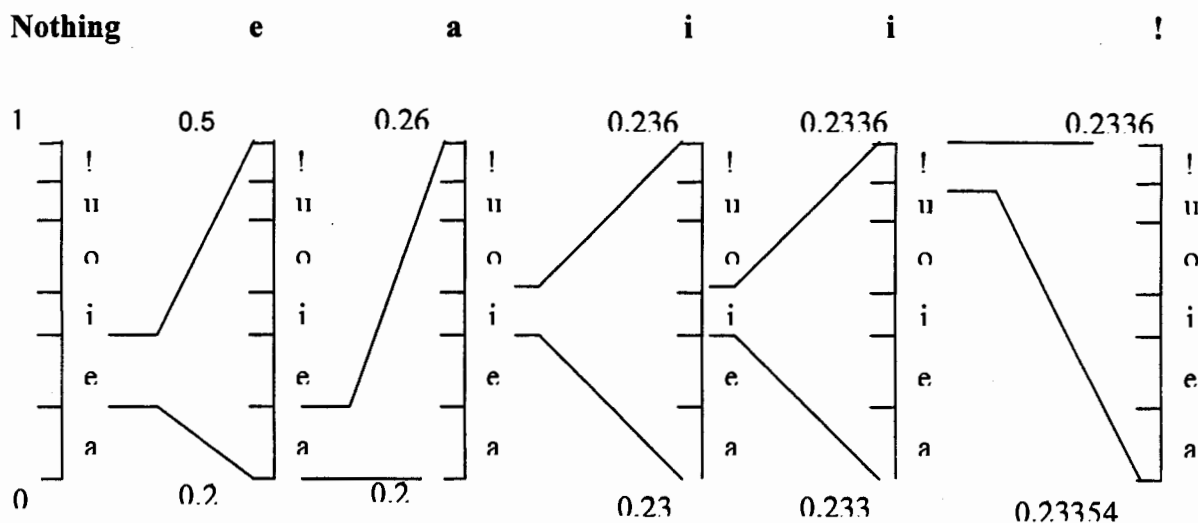


Figure 6.1.b Representation of arithmetic coding  
Process with the interval scaled up at each stage

### 6.3 Models for Arithmetic Coding

The program in [coding] must be used with a model that provides a pair of translation tables `index-to-char[]` and `char-to-index[]`, and a cumulative frequency array `cum-freq []`. The requirements on the latter are that

- $\text{cum-freq}[i - 1] > \text{cum-freq}[i]; l$
- an attempt is never made to encode a symbol  $i$  for which  $\text{cum-freq}[i - 1] = \text{cum-freq}[i]$ ; and
- $\text{cum-freq}[0] \leq \text{Max-frequency}$ .

Provided these conditions are satisfied, the values in the array need bear no relationship to the actual cumulative symbol frequencies in messages. Encoding and decoding will still

work correctly, although encodings will occupy less space if the frequencies are accurate. (Recall our successfully encoding *eaii!* according to the model of Table I, which does not actually reflect the frequencies in the message.)

### 6.3.1 Fixed Model

The simplest kind of model is one in which symbol frequencies are fixed. The fixed model has symbol frequencies that approximate; however, bytes that did not occur in that sample have been given frequency counts of one in case they do occur in messages to be encoded. Frequencies have been normalized to total. The initialization procedure simply computes a cumulative version of these frequencies, having first initialized the translation tables. Execution speed would be improved if these tables were used to reorder symbols and frequencies so that the most frequent came first.

An exact model is one where the symbol frequencies in the message are exactly as prescribed by the model. For example, the fixed model is close to an exact model for the particular excerpt of the Brown Corpus from which it was taken. To be truly exact, however, symbols that did not occur in the excerpt would be assigned counts of zero, rather than one (sacrificing the capability of transmitting messages containing those symbols). Moreover, the frequency counts would not be scaled to a predetermined cumulative frequency. [14]

The exact model can be calculated and transmitted before the message is sent. Under quite general conditions, this will not give better overall compression than adaptive coding (which is described next).

### 6.3.2 Adaptive Model

An adaptive model represents the changing symbol frequencies seen so far in a message. Initially all counts might be the same (reflecting no initial information), but they are updated, as each symbol is seen, to approximate the observed frequencies. Provided both encoder and decoder use the same initial values (e.g., equal counts) and the same updating algorithm, their models will remain in step. The encoder receives the next symbol, encodes it, and updates its model. The decoder identifies it according to its current model and then updates its model. Updating the model is quite expensive because of the need to maintain cumulative totals.

Entropy coding is now being used frequently in conjunction with vector quantization (VQ) for image coding. Its use is motivated by the fact that probability distribution of VQ

## 7. Methodology and Framework

It's about that how the system is designed as well as what possibly can be the module of this system. A detailed description is given about the general design of the system and then the system shown in parts and the respective detail about those parts are also given. Framework called conditional entropy constrained using vector quantization (CEC-VQ). Our quantizer design absolutely supports by Entropy.

### 7.1 High level structure of single stage EC-VQ

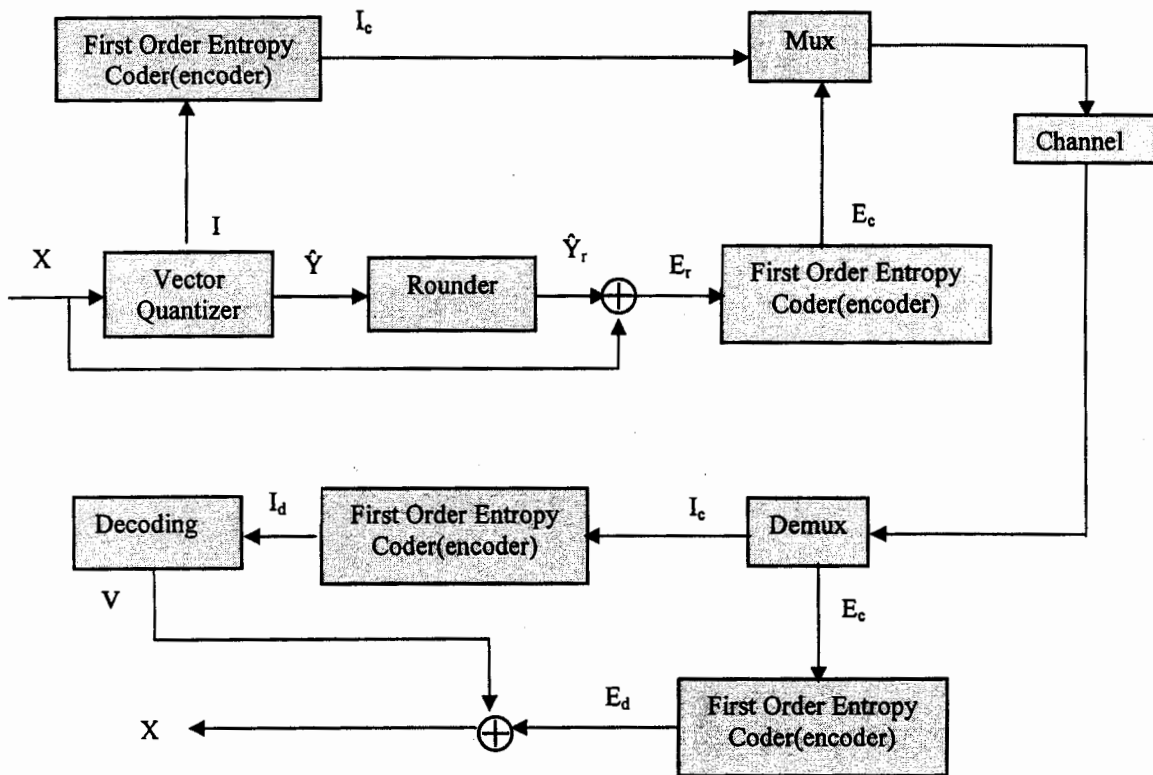


Figure 7.1 High level structure of Framework

The system shown in figure 1 is the general diagram of the system. The first portion of is the encoder. The input to the system is an image from which the input vectors  $X$  are created. The input  $X$  is given to the vector quantizer. The output of the vector quantizer is a quantized output  $\hat{Y}$  and indexes  $I$ . The quantized output  $\hat{Y}$  is used to calculate the error vectors  $E_r$  i.e by subtracting the quantized output  $\hat{Y}$  from the original input  $X$  The error vectors  $E_r$  and the indexes  $I$  are given to the first order entropy coder i.e is the arithmetic coders (encoder). The outputs of the arithmetic coder (encoder) are the compressed

indexes  $I_c$  and the compressed error vector  $E_c$  are combined by multiplexer and passed onto the channel.

From the channel this combined compressed indexes  $I_c$  and compressed error vectors  $E_c$  are separated by the de-multiplexer. These separated inputs are given to the first order entropy coders i.e. arithmetic coders (decoders). The output of the arithmetic decoders are the decompressed indexes  $I_d$  and decompressed error vectors  $E_d$ . The decompressed indexes  $I_d$  are given as input to the decoder which performs a table look-up to get the input vectors  $X$ . The input vectors obtained  $X$  and the error vectors  $E_d$  are combined to get the original input image.

### 7.1.1 System Encoder

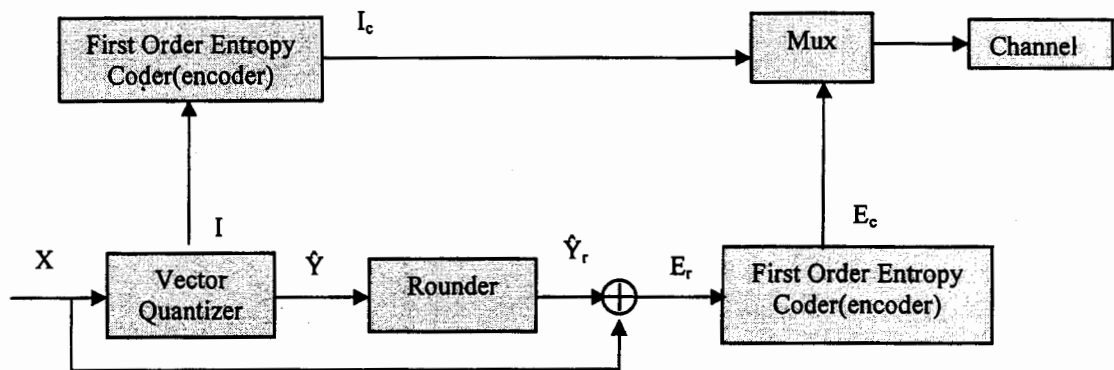


Figure 7.2. System Encoder- CEC-VQ

#### Description

The input to the system is an input image on the basis of which input vectors  $X$  are created by the process of blocking in which we pick blocks of size  $4 \times 4$  from the image. The input vectors are then passed to the vector quantizer where the best code words are selected on the basis of minimum entropy from the code book present at the vector quantizer with respect to each input vector and thus indexes  $I$  of the best code words and the quantized output  $\hat{Y}$  are obtained. The quantized output  $\hat{Y}$  is used to calculate the error vectors  $E_r$  i.e. by subtracting the quantized output  $\hat{Y}$  from the original input  $X$ . The indexes  $I$  and the error vectors  $E_r$  are passed onto the first order entropy coder i.e. the arithmetic coder (encoder) from where we get the output i.e. compressed indexes  $I_c$  and the compressed error vectors  $E_c$  which are then combined using the multiplexer to be passed onto the channel.

### 7.1.2 System Decoder



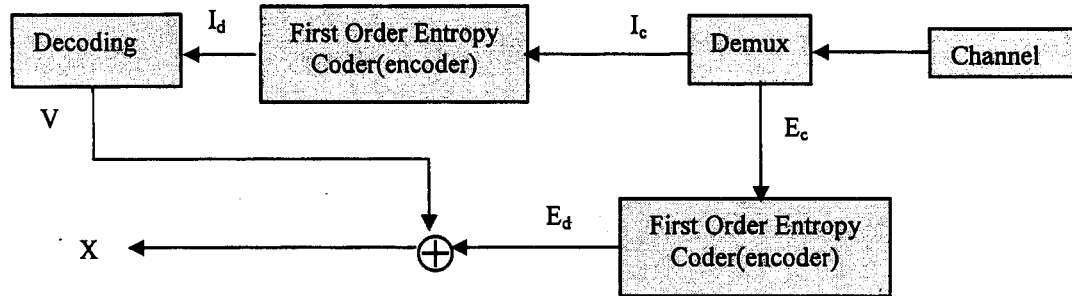


Figure 7.3. System decoder- CEC-VO

### Description

The compressed output of the encoder is then given to the decoder for further processing. The combined compressed output of encoder i.e compressed indexes  $I_c$  and the compressed error vectors  $E_c$  are separated using de-multiplexer. The compressed indexes  $I_c$  and the compressed error vectors  $E_c$  are then again passed onto the first order entropy coders i.e arithmetic coder (decoder). The output is decompressed indexes  $I_d$  and decompressed error vectors  $E_d$ . The  $I_d$  is then given to the decoding process i.e getting the input vectors  $X$  by mapping the indexes  $I_d$  to the codewords of the codebook, where codebook is the same as present on the vector quantizer encoder. Then the decompressed error vectors  $E_d$  and the input vector  $X$  are combined to get the resultant image which is same as the input image.

## 7.2 Major Components of System

System consist of major components as:

- Vector Quantizer
- Entropy Coder
- Multiplexer-Demultiplexer
- Decoding

### 7.2.1 Design of Vector Quantizer

Here, we discuss the task of how to design a VQ, or how to populate the codebook, which can be performed according to several different principles. Three usable codebook principles are trained codebook, random codebook, and lattice codebook. The basic goal for any design approach is to choose the codebook,  $C$ , and the partitioning, such that the statistical average is small or even minimized.

Firstly, we generate the codebook on the basis of Mean Square Error (MSE) as distortion measure, initial codebook. The next step is to improve this MSE based Codebook iteratively, on the basis of Entropy. It forms the codebooks using the generalized Lloyd algorithm (GLA). It runs the GLA for codebooks of size  $2^n$ ,  $n = 0, 1, 2, \dots$  until the final size is reached. The performance of a VQ is often given as the distortion.

### 7.2.1.1 Measuring Criteria

The distortion is usually quantized as the statistical average distortion between a random input vector,  $X$ , and the reproduction vector,  $Y$ .

$$MSE = D = E[d(X, Y)] = \sum_{i=1}^M \int_{s_i} f_X(X) d(X, Y) dx. \quad (7.1)$$

Where the distortion measure  $d$  is a non-negative cost-function and  $f_X(X)$  is the probability density function (pdf) of the source. In practice, the statistical average distortion is often evaluated as the time average over some set of vectors.

$$D = \frac{1}{N} \sum_{n=1}^N d(X_n, Y_n) \quad (7.2)$$

The new distortion measure used in the design of the CEC-RVQ is Entropy

$$Entropy = d(X, Y) = -\log_2 [pr(I(X-Y))] \quad (7.3)$$

To elaborate, let  $X$  be the input and  $Y$  be the output of the CEC-RVQ. The new distortion measure used in the design of the CEC-RVQ is  $d(X, Y)$ , where  $I(a)$  is the integer closest to the real  $a$ . The GLA continues until the change in distortion is less than threshold.

### 7.2.1.2 Encoding using Entropy Constrained

We have to find the best code word from codebook corresponds to Test image being compressing. For this find the difference of each test image vector with all the code words and then find their entropies. Search the difference in the stored errors, if difference matches with any of the stored error take its probability to calculate entropy otherwise assign a huge value to entropy. Choose the codeword, having the least entropy as the best codeword.

### **7.2.2 Specification of Entropy Coder**

In our system, first order entropy coder is Fixed-Model Arithmetic Coder. More complex models can provide more accurate probabilistic predictions and hence achieve greater compression. The effectiveness of any model can be measured by the entropy of the message with respect to it, usually expressed in bits/symbol. Shannon's fundamental theorem of coding states that, given messages randomly generated from a model, it is impossible to encode them into less bits (on average) than the entropy of that model.

The simplest kind of model is one in which symbol frequencies are fixed. The fixed model has symbol frequencies that approximate. An exact model is one where the symbol frequencies in the message are exactly as prescribed by the model.

### **7.2.3 Multiplexer-Demultiplexer**

First order entropy coder generates encoded Indices and error of image being compressed. Multiplexer combine theses output and send it to channel, at other receiver side demultiplex separate this merged information. This process take 1 index and its corresponding error then next index and its correspondence till end of indices.

### **7.2.4 Decoding Process**

In decoding process, we have to decode the encoded or quantized vectors, for this purpose we need the codebook, indices and errors. We got the form of image same as that the original image provided to the encoding process.

## Chapter 8

---

# Implementation

## 8. Implementation

Our System Implementation consist of the following main modules, The output of one module serves as an input to other module

1. Preparing a Training set
2. Blocking code
3. CEC-VQ Process
4. Entropy coding (Arithmetic Coder)
5. Decoding
6. Unblocking
7. Verifying the Perfect Reconstruction

First of all a training set is created using sample images in Matlab. The set is then converted into blocks of different size depending on requirements.

The Blocked training set is then handed over to CEC-VQ module, a module code written in C. firstly, EC-VQ module generates a code book using training set by enforcing conditional entropy constrains. Secondly, generate indices and residual of Test Image, called encoding.

### 8.1 Preparing a Training set

We have used two training sets for compression .One training set contains 4-images and other has 8-images.

#### 8.1.1 Scaling of images

We are using gay scale 512 x 512 and 8-bpp raw images. We have to scale many of the images to 512 x 512 .For this we have used the Matlab function like `imresize ()` and perform bicubic interpolation.

#### 8.1.2 Concatenating the images

For Training set we have concatenate a set of raw images using the 'cat' command in at Linux Terminal. `Cat image1 image2 image 3 ..... image8 > TrainingSet.`

### 8.2 Blocking code

it is a program written in Cm whose only purpose is to convert the input into blocks. The blocks of 4x4 or 8x8 of training set and test image has created. One block is called 1 vector. Blocking is a fundamental step, where ever we applying vector quantization technique.

SYNOPSIS

*Block -i inputfile -o outputfile -r rows -l column -h blockheight -w blockwidth*

## DESCRIPTION

Block takes a raw image `inputfile` and creates `outputfile` which is a list of vectors where each vector is a block from `inputfile`. The `inputfile` has dimensions `rows` and `columns`, and the block has dimensions `blockheight` and `blockwidth`. If no `outputfile` is named, then the name of the `outputfile` defaults to `inputfile.TS`.

The `training_data` now has four images, so there are  $4 \times 512 = 2048$  rows and 512 columns:

*Block -i training\_data -r 1536-1512 -h 4 -w 4 -o training\_data.TS*

The output is a file called `training_data.TS` where `TS` is for training set. Now the training data are ready to use.

Main portion of the blocking code is

```

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);
  }
}

fclose(inputfile);
fclose(outputfile);
exit(0);

```

### 8.3 CEC-VQ Process

The main process CEC-VQ consists of two sub-major process.

- Code Book Generation
- Encoding using entropy constrains

#### 8.3.1 Codebook Generation

Codebook generation process is the most important step in the design of entropy constrained vector quantizer.

We have to first generate the codebook on the basis of Mean Square Error (MSE) as distortion measure, initial codebook. The next step is to improve this MSE based Codebook iteratively, on the basis of Entropy. The codebook is updated in 4 to 6 iterations depending upon the size of the codebook used and the training set, the main program for codebook generation is

#### SYNOPSIS

```

Stdvq -t trainingset -c codebook -d dimension -f codebooksize -h threshold -a addoffset
-s speedup -W

```

#### DISCRIPTION

it forms the codebooks using the generalized Lloyd algorithm(GLA). It run the GLA for codebooks of size  $2^n$   $n = 0,1,2,\dots$  until the final size is reached. The final size can be any integer value. Each increase in the size of the codebook is done by splitting codewords from the next smallest codebook, (perturbed versions of the old codewords). The GLA continues to until the change in distortion is less than threshold .The GLA will

abort if there are cells, which cannot be filed. If there are empty cells, the Lloyd iteration tries to split the most populous cells only ,(individual cell distortion is not considered ). To speed up the Lloyd algorithm we have used partial distortion for speedup. It first generates the codebook using Mean square as a distortion measure and then update the codebook on the basis of entropy. There is one flag (-W). If the flag is not specified, then only the final codebook is written . If the flag is specified, then all intermediary codebooks are written as well .Each codebook has the following format:

TYPE	SIZE	DESCRIPTION
Long	1	number of codewords (size)
Integer	1	vector dimension (dimension)
Double	size* dimension	codewords

#### CALLS

- Lloyd ()
- splitcodewords()
- entropyCB.c
- writecodebook()

Use the “stdvq” program to make a codebook with 512 codewords using the vector dimension 16.

```
Stdvq -t trainingset .TS -c codebook -d 16 -f 512
```

This will output a codebook called “codebook” with 512 codewords of dimension 4. Important Functions:

#### 8.3.1.1 Generating codebook on MSE basis:

```
for(i = 1; i < codebooksize;)
{ /* run the GLA for codebook of size i */
  if ( (distortion = method(codebook,i)) < 0) {
    exit(13);
  }
  /* if distortion is zero, no need to continue.
  note that lloyd can and will change codebooksize in such a case */
  if (distortion == 0) break;
  /* display the distortion of the training set to the codebook of size i */
  printf("%s %-7d: %f\n",DISTORTION,i,distortion);
  fflush(stdout);
  /* write the codebook of size i if requested */
  if(write_all_codebooks) if(!writecodebook(codebook,i)) exit(14);
  /* find the number of new codewords that need to be made (j-i) */
  if ((j = 2*i) > codebooksize) j = codebooksize;
  /* split the codewords */
  splitcodewords(codebook,i,j,0);
  /* increment the codebook size */
  i = j;
```



```

}
/* it may be that distortion is 0, so we can exit early */
if (distortion == 0) {
printf("%s %-7d: %f\n",DISTORTION,i,distortion);
fflush(stdout);
if(!writecodebook(codebook,codebooksize)) exit(15);
}

```

### **Lloyd0(codebook, size)**

#### SYNOPSIS

`lloyd0(codebook, size)`

#### DESCRIPTION

`lloyd0` performs the generalized Lloyd algorithm for the given codebook with `size` codewords. Lloyd exits with a codebook if the percent change in distortion from one pass to the next is less than `threshold`. If there are any empty cells, Lloyd will try to split the most populous cells. Lloyd will attempt to split cells up to `size` times unless the distortion is zero. If the distribution is zero, then those codewords are returned and the global variable `codebook size` is modified since large codebooks can no longer be made. This ensures that the zero distortion codebook is returned to the user, but allows that the program terminate normally. If the `write_all_codebooks` option is not selected that the program will terminate.

#### RETURN VALUE

`lloyd` returns a positive number, i.e. the distortion, if a codebook is found. A negative number is returned if there is not enough memory for temporary storage and computations or if a codebook cannot be found. There are two possibilities for the latter case: there are empty cells which could not be filled after `size` attempts, or there are permanently empty cells, zero distortion, and the `write_all_codebooks` option was not selected.

#### PARAMETERS

- `codebook` contains the code words
- `size` is the current number of words in the codebook.

#### CALLS

`splitcodewords()`

#### **Selection of bestcodewords**

We found the distortion by MSE and the codeword giving minimum distortion will be selected as a best codeword.

```

/* read in vector and find the closest codeword */
while (fread(tempvector, sizeof(DATA), dimension, trainingfile) ==
      dimension && !feof(trainingfile) && !ferror(trainingfile)) {
    bestdistortion = HUGE; /* keep convention that ties go to lower index */
    bestcodeword = 0;
    for (i = 0; i < size; i++){ /* find the best codeword */
        tempdistortion = 0.0;
        for (j = 0; j < dimension; j++) {
            temp = ((double) tempvector[j]) - codebook[i][j];
            tempdistortion += temp*temp;
            if (tempdistortion > bestdistortion) j = dimension;
        }

        if (tempdistortion < bestdistortion) {
            bestdistortion = tempdistortion;
            bestcodeword = i;
        }
        /* if the bestdistortion is 0.0, the best codeword is found */
        if (bestdistortion == 0.0) i=size;
    }
    fwrite(&bestcodeword, sizeof(long), 1, best);
    count[bestcodeword]++;
    for (j = 0; j < dimension; j++){
        centroid[bestcodeword][j] += (double) tempvector[j];
    }
    distortion += bestdistortion;
} /* all training vectors have been encoded */

    fclose(best);
/* normalize the distortion */
numbervectors = 0;
for (i = 0; i < size; i++) {
    numbervectors += count[i];
}
if(numbervectors == 0) {
    fprintf(stderr, "%s: %s: %s\n", programname, trainingname, NOTFOUND);
    return(-1.0);
}
distortion /= (double) numbervectors;
if(distortion < 0) {
    fprintf(stderr, "%s: %s: %s\n", programname, OVERFLOWED, ABORT_STDVQ);
    return(-1.0);
}

/* if distortion = 0.0 or if change in distortion < threshold AND
   if there aren't any empty cells, exit */
if ( (emptycells == 0) &&
      ((distortion == 0.0) ||
       ( (olddistortion - distortion)/distortion < threshold)) ) {

```

```

    /* if distortion is 0, let the program exit gracefully */
    if(distortion == 0 && size < codebooksize) {
        fprintf(stderr, "%s %d\n", STOP, size);
        size = codebooksize;
    }
    return(distortion);
}
/* Find the number of empty cells */
emptycells = 0;
for (i = 0; i < size; i++) {
    if (count[i] == 0) ++emptycells;
}
/* no empty cells, find new centroids and reinitialize for next pass */
if (emptycells == 0) {
    for (i = 0; i < size; i++) {
        for (j = 0; j < dimension; j++) {
            codebook[i][j] = centroid [i][j] / (double) count[i];
            centroid[i][j] = 0.0;
        }
        count[i] = 0;
    }
    olddistortion = distortion;
}
/* consolidate the nonempty codewords at the beginning of the
   array with the most populous cells first. */
for(n = 0; n < size - emptycells; n++) {
    j = 0;
    bestcodeword = 0;
    for (i = 0; i < size; i++) {
        if (count[i] > j) {
            j = count[i];
            bestcodeword = i;
        }
    }
    /*i*/
    for (j = 0; j < dimension; j++) { /* find centroid */
        codebook[n][j] = centroid[bestcodeword][j] /
            (double) count[bestcodeword];
        centroid[bestcodeword][j] = 0.0;
    }
    /*j*/
    count[bestcodeword] = 0;
}
/*n*/
/* split the required number of codewords */
splitcodewords(codebook, size-emptycells, size, pass)

```

### 8.3.1.2 Generation of code book on Entropy base:

This function is called from stdvq.c. it takes MSE-based codebook generated through the lloyd0 iteration. The size is the final size of the codebook. Main steps followed are given as:

- It calculate and store the probability of error. The error is actually the difference between the MSE based codebook's best index for the given vector.

*/\*Read the vector 1 by 1 from the training file\*/*

```
while (fread(tempvector, sizeof(DATA), dimension, trainingfile) ==
        dimension && !feof(trainingfile) && !ferror(trainingfile) ){
    a1++;
    fread(&best,sizeof(long),1,bestM);
    for(j=0;j<dimension;j++){
        errorj = (int)(( double) tempvector[j]) - codebook[best][j]);
        if(errorj<min)
            min=errorj;
        if(errorj>max)
            max=errorj;
        a=searche(first,errorj,0);
        if(a==0)
            push(&first,errorj);
    }
}
```

- now we have to find the best codeword on the basis of entropy. For this find the difference of each vector with all the code words and then find their entropies.
- To find the entropy search the difference in the stored errors. If difference matches with any of the stored error take its probability to calculate entropy otherwise assigns a huge value to entropy.
- Choose the codeword, having the least entropy as the best codeword.

```
while (fread(tempvector, sizeof(DATA), dimension, trainingfile) ==
        dimension && !feof(trainingfile) && !ferror(trainingfile) ){
    bestentropy=HUGE;
    for(i=0;i<size;i++){
        entropy=0.0;
        for(j=0;j<dimension;j++){
            errorj=(int)(( double) tempvector[j]) - codebook[i][j]);
            a=searche(first,errorj,1);
            if(a==0){
                entropy+=10000;
            }
            else{
                prob=(double)(a)/(double)(512*512*4);
                entropy -= (log(prob)/M_LN2);
            }
        }
    }
}/*j*/
```

```

        entropy=entropy/(double)dimension;
        if(entropy<bestentropy){
            bestcodeword=i;
            bestentropy=entropy;
        }
    }/*i*/
    fwrite(&bestcodeword,sizeof(long),1,best2);
    count[bestcodeword]++;
    for (j = 0; j < dimension; j++){
        centroid[bestcodeword][j] += (double) tempvector[j];
    }
    distortion += bestentropy;
}/*while*/

```

- **now update the codebook**

```

for (i = 0; i < size; i++) {
    for (j = 0; j < dimension; j++) {
        if(count[i]!=0){
            codebook[i][j] = centroid [i][j] / (double) count[i];
        }
    }/*j*/
    count[i] = 0;
}/*i*/

```

- now for the next improvement iteration of the codebook we have to update the stored errors and probabilities using the updated entropy based codebook and best indices selected earlier on the basis of entropy.
- Continue the iteration to update codebook until percent change in distortion is less than threshold.

```

if ( ((distortion == 0.0) || ( (olddistortion - distortion)/distortion < 0.0001))) {
    fprintf(stderr, "%s %d\n", "STOP", size);
    fclose(bestM);
    return(distortion);
}

```

- write the final entropy-based codebook

```

if(!writecodebook(codebook,codebooksize)) exit(15);

```

### **Splitcodewords()**

It is called by Lloyd0 function. Takes a codebook and creates oldsize-newsize new codewords from the codewords. The old codewords are not modified.

### **Perturb()**

```

Perturb(codebook[i], codebook[i+oldsize], scale);

```

It is called by splitcodeword function. Perturb takes old codeword and changes it slightly to form newcodeword. Oldcodeword is not altered in the process. Scale is available to

change so that a split many times if necessary and still have each resulting new codeword be different.

### Write codebook()

It is called `stdvq.c`. write codebook to a file of a given size. If it fails it return FALSE, otherwise TRUE is returned.

### 8.3.2 Encoding using entropy constraints

to encode the test image, the image first be arranged into vectors using the “block” program. Then the “vq” program is used to encode the test image.

#### SYNOPSIS

```
Vqe -c codebook -I inputfile -o outputfile -s sppedup -D
```

#### DESCRIPTION

vqe encodes the inputfile using the codeword from the codebook that yield the lowest entropy, reproduction file is placed in outfile.

#### CALL

#### Fullsearch()

Fullsearch(codebook, count, cell distortion)

This function is called to perform encoding on the basis of entropy. The main steps are as follows:

- We have already stored the errors and their probabilities of the training set computes from the final codebook.
- Now we have to find the best codeword on the basis of entropy. For this find the difference of each test image vector with all the code words and then find their entropies.
- To find the entropy search the difference in the stored errors. If difference matches with any of the stored error take its probability to calculate entropy otherwise assigns a huge value to entropy.
- Choose the codeword, having the least entropy as the best codeword.

```
while (fread(tempvector, sizeof(DATA), dimension, inputfile) ==
        dimension && !feof(inputfile) && !ferror(inputfile) ){
    bestentropy=HUGE;
    for(i=0;i<codebooksize;i++){
        entropy=0.0;
        for(j=0;j<dimension;j++){
            errorj=(int)((double) tempvector[j] - codebook[i][j]);
            a=searche(first,errorj,1);
```

```

        if(a==0){
            entropy+=10000;
        }
        else{
            prob=(double)(a)/(double)(512*512*4);
            entropy -= (log(prob)/M_LN2);
        }
    }/*j*/
    entropy=entropy/(double)dimension;
    if(entropy<bestentropy){
        bestcodeword=i;
        bestentropy=entropy;
    }
}/*i*/
}/*end while*/

```

- store the best indices and their probabilities on the way.
- Also we had stored the errors and their probabilities. These errors are found by calculating the difference between the quantized test image vector and the original test image vector.

## 8.4 Entropy Coding

Best indices and errors stored during encoding process are now compressed using arithmetic coder. We have used fixed-model arithmetic coder which are provided with the probabilities of errors and indices.

For detail see chapter 6.

## 8.5 Decoding

In decoding process, we have to decode the encoded or quantized vectors, for this purpose we need the codebook, indices and errors. We got the form of image same as that the test image provided to the encoding process. The image is now ready to perform the unblocking operation.

```

while(fread(&de,sizeof(int),1,inputfile)==1){
    fread(err,sizeof(int),dimension,errorfile);
    for (i = 0; i < dimension; i++) {
        tempvector[i] = (DATA)((int)codebook[de][i]+(int)err[i]);
        /* write the data to the output file */
        if (fwrite(tempvector,sizeof(DATA),dimension,outputfile) != dimension) {
            fprintf(stderr,"%s: %s: %s\n",programname,outputname,NOWRITE);
            return(-1.0);
        }
    }/*i*/
}/*end while*/

```

## 8.6 Unblocking

finally, the unblock program is used to arrange the encoded/decoded vectors into a decoded image. It is the reverse process of blocking.

It's a reverse process of Blocking in same manner.

```
Unblock -i lena.TS.encoded -r 512 -512 -h 2 -w 2 -o lena.final
```

## 8.7 Verifying the Perfection Reconstruction

To verify that we have obtained perfect reconstruction we have calculated MSE, SNR and PSNR. Some portion of the code is given as:

```
/* get the headers and see if they match */
file1=fopen(filename1,"r");
if (file1==NULL) {
    fprintf(stderr, "rawmse: can not open file '%s'\n", filename1);
    exit(1);
}
file2=fopen(filename2, "r");
if (file2==NULL) {
    fprintf(stderr, "rawmse: can not open file '%s'\n", filename2);
    exit(1);
}
framesize=width*height;
data1=(unsigned char *)malloc(framesize);
data2=(unsigned char *)malloc(framesize);
if (data1==NULL || data2==NULL) {
    fprintf(stderr, "rawmse: can not allocate memory\n");
    exit(1);
}
/* read data */
if (fread(data1, framesize, 1, file1)!=1) {
    fprintf(stderr, "rawmse: can not read data from file '%s'\n", filename1);
    exit(1);
}
if (fread(data2, framesize, 1, file2)!=1) {
    fprintf(stderr, "rawmse: can not read data from file '%s'\n", filename2);
    exit(1);
}
if (fread(data2, framesize, 1, file2)!=1) {
    fprintf(stderr, "rawmse: can not read data from file '%s'\n", filename2);
    exit(1);
}

void calc_err(unsigned char *data1, unsigned char *data2,
             long framesize, double *err, double *energy)
{
```



```

long i;
double er, en;
for (er=0, en=0, i=0; i<framesize; i++) {
    er=(i*er+square(data1[i]-data2[i]))/(i+1);
    en=(i*en+square(data1[i]))/(i+1);
}
*err=er;
*energy=en;
return;
}

```

## 8.8 Display Voronoi Regions

### SYNOPSIS

```
voronoi_fs -c codebook -o output -r rows -l columns
```

### DESCRIPTION

Voronoi\_fs takes a full search codebook that has two dimensions and creates an output image that shows the edges of the voronoi regions. The codeword are scaled to fit 90% of the region defined by the user as row by columns. Some portion of the code is given as:

```

/* read the codebook */
for(i = 0; i < codebooksize; i++) {
    if (!(codebook[i] = (double *) calloc(dimension,sizeof(double)))) {
        fprintf(stderr,"%s: %s\n",programname,NOMEMORY);
        exit(13);
    }
    if (fread((char *) codebook[i],sizeof(double),dimension,codebookfile)
        != dimension || feof(codebookfile) || ferror(codebookfile) ) {
        fprintf(stderr,"%s: %s: %s\n",programname,codebookname,NOREAD);
        exit(14);
    }
}
}/*i*/
fclose(codebookfile);
/* normalize the tree data to the rows by cols range */
normalize_codebook(codebook);
/* encode the image */
if(!voronoi_diagram(codebook)) {
    exit(15);
}
}

```

## **Chapter 9**

---

# **Result and Analysis**

## 9. Results and Analysis

We have designed a loss less image compression system using vector quantization which is of course a lossy compression technique. Our architecture is an extension of CEC-RVQ to CEC-VQ by converting it into single stage. Practically, goal of work was to obtain the perfect reconstruction of image. Decompressed image should be same as that of original one. We have obtained 100% lossless image compression. Testing in our system done by different images and codebook of different size e.g., 256, 128, 64, 32 etc.

### 9.1 Experimental Techniques

Experimental results are used to study the effects of lossless compression. The techniques we used are as:

#### 9.1.1 MSE(Mean Square Error)

$$\text{MSE} = \frac{1}{N \cdot M} \cdot \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (X(i, j) - \hat{X}(i, j))^2 \quad (9.1)$$

#### 9.1.2 SNR (Signal to Noise Ratio)

$$\text{SNR} = 10 \cdot \log_{10} \frac{\sigma^2}{\text{MSE}} \quad (9.2)$$

#### 9.1.3 PSNR (Peak signal-to-Noise Ratio)

$$\text{PSNR} = 10 \cdot \log_{10} \frac{255^2}{\text{MSE}} \quad (9.3)$$

To test that the system is lossless our results should be as given

Experiment Techniques	result
MSE	0
SNR	Inf db
PSNR	Inf db

Table 1. MSE, SNR, PSNR of CEC-VQ for the image LENA

We have tested the system and it is found that perfect reconstruction is obtained as the above mentioned results.

## 9.2 Testing Measures

We gave different test images and examined the reconstructed images for black box testing. Favorite test image which we have used during research development was "Lena". For rest of experiments we used different images like Barbara, Elaine, Aya, Tiffany, etc.

For white box testing we used codebooks of different sizes such as 256, 128, 64, 32, 16 etc. we carried out most of our work using the codebook size of 256. Different compression ratio and bit-rates (in bpp) are obtained by changing the size of codebook.

We have made the codebook using different number of images in Training Set. We have tested the system using training set containing for 4-Images and then employed the training set containing 8-images.

### Bit Rate:

Bit-rate (in bpp) = size of compressed image in bits / Number of Pixel in original image

### Compression Ratio:

Compression ratio = Original size of image / Compressed image size

## 9.3 Experimental Results

### Specification

Code book size = 256k, 128k, 64k, 32k, 16k

Training set = 4-images, 8-images

Test image = "Lena"

Code book size		256k	128k	64k	32k	16k
bit rate (bpp)	4 images T-Set	4.925	4.995	5.084	5.17	5.34
	8 images T-Set	4.913	4.956	5.084	5.17	5.34

Table 2. Bit rate (bpp) of LENA (Compressed image )  
using 4 and 8 training set of image.

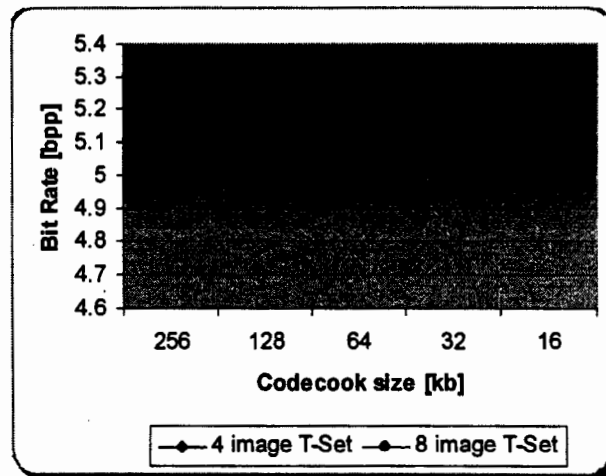


Figure9.1 Bit rate of LENA (Compressed image )  
using 16, 32,64, 128, 256 size of codebook

Different test images were tested successfully for their perfect reconstruction. Some of the images tested so far includes Ano, Aben, Barbara, Elaine, girl1, girlb.

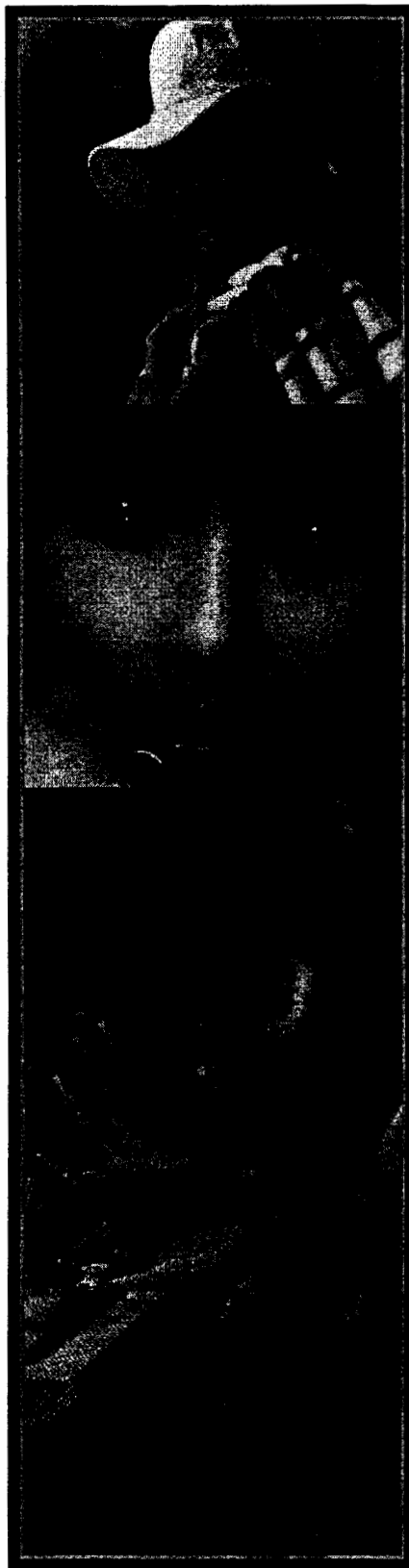


Figure 9.2 Training Set-4 images

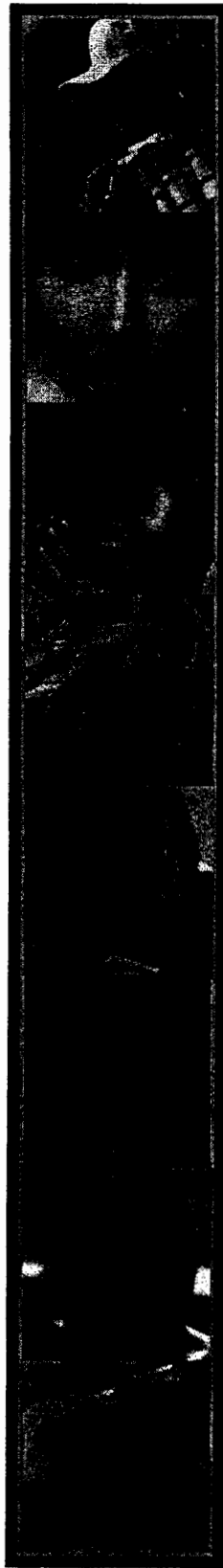


Figure 9.3 Training Set-8 images



Figure 9.4 Lena-Original Image





Figure 9.5 Lena-reconstructed image

#### 9.4 Future Work and Conclusion

In previous chapter of this dissertation we presented a thorough study of what is lossless image compression and how we have used it in our implementation. The method we presented here has shown a good compression of gray scale images. However it would be unreasonable to suggest that no further improvement can be made in our method i.e lossless image compression using entropy constrained vector quantization.

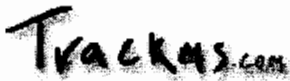
- Expending the training set with different size of block, 8x8, 12x12, 16x16 etc.

- Using different test images rather we were reserved on grayscale of head and shoulder images. The work can be further enhanced by including medical, tele-images.
- The codebook generation process in our work is a bit, it could be more efficient.

## Reference

- [1] Faouzi Kossentini and Mark J. T. Smith, "High order entropy constrained Residual VQ for Lossless compression of images", *IEEE trans*, 1995
- [2] G. G Langdon and J. Rissanen, "Compression of black-white images with arithmetic coding," *IEEE Trans. Comm*, vol.29, no. 6, pp. 858-867, 1982.
- [3] S. M. Lei, T. C. Chen, and K. H. Tzou, "Subband HDTV coding using higher-order conditional statistics", *IEEE JSAC*, vol. 11, pp. 65-76, Jan. 1993.
- [4] X. Ginesta and S. P Kim, "Semi-adaptive context-tree based lossless image compression, in *ICIP*, (Austin, TX), Nov. 1994.
- [5] K. Popat and R. W Picard, "Exaggerated consensus in lossless image compression," in *ICIP*, (Austin, TX), Nov. 1994.
- [6] S. Yu, M. N. Wernick, and N. P. Galatsanos, "lossless compression of multidimensional medical image data using binary-decomposed high-order entropy coding," in *ICIP*, (Austin, TX), Nov. 1994.
- [7] Faouzi Kossentini, Wilson C.Chung, "Conditional Entropy-constrained Residual VQ with Application to Image Coding", *IEEE Transactions on image processing*. Vol.5,no.2, (1996).
- [8] F. Kossentini, W. Chung, and M. Smith, "Image coding using high-order conditional entropy – constrained residual VQ," in *ICIP*, (Austin, TX), Nov. 1994.
- [9] Nopparat Pantaena, M. Sangworasil, C. Nantajiwakornchai and T. Phanprasit, "Image compression using vector quantization"
- [10] Hosam Khalil and Kenneth Rose, "Predictive multistage vector quantizer design using asymptotic closed loop optimization." *IEEE Transaction on image processing*, vol.10, no. 11, Nov 2001.
- [11] Yun Gong, Michael K.H. Fang, "On Entropy Constrained Residual Vector Quantization Design." *IEEE, data compression conference* March, 1999
- [12] *Vector Quantization and Signal Compression*, Allen Gersho & Robert M.Gray,
- [13] *Hand Book of Image and Video Compression*
- [14] Ian H. Witten, Radford M. Neal, and John G. Cleary , "Arithmetic coding or Data Compression" communication of the ACM, June 1987.
- [15] Gallager, R.G. Variations on a theme by Huffman. *IEEE Trans. Inf. Theory* IT-24, 6 (Nov. 1978), 668-674. Presents an adaptive Huffman coding algorithm and derives new bounds on the redundancy of Huffman codes.
- [16] Shannon, C.E. and Weaver, W. *The Mathematical Theory of Communication*. University of Illinois Press, Urbana, Ill., 1949. A classic book that develops communication theory from the ground up.

- [17] Welch, T.A. A technique for high-performance data compression. *Computer* 17, 6 (June 1984), 8-19. An improved implementation of this method is widely used in UNIX systems under the name compress.
- [18] Paul G. Howard and Jeffrey Scott Vitter, "Practical Implementation of Arithmetic Coding." A technical report, April 1992.
- [19] Eric Bodden, Malte Clasen, Joachim Kneis, "Arithmetic Coding Revealed" A guided tour from theory to praxis, Translated and updated version, May 2004.
- [20] Fredrik Norden "A short tour on vector quantization", March 2004.
- [21] B.H Juang and A.H Gray, "Multiple stage vector quantization for speech coding of images, " *Visual commun, Image processing IV*, vol.SPIE-1199, pp.970-978, 1989.
- [22] A. Gersho and R. M. Gray, "Vector Quantization and Signal Compression. Boston Kluwer1992.
- [23] F. Kossentini, M. Smith and C.Barnes, "Entropy constrained residual vector quantization," in proc. IEEE int.conf. Acoust., speech, signal processing, Minneapolis, MN, vol. V, Apr.1993
- [24] Image coding using entropy constrained RVQ," *IEEE Tran. Image Processing*. Mar, 1996.
- [25] Necessary conditions for the optimality of variable rate residual vector quantizers, *IEE Tran. Inform Theory*, pp. Nov 1995
- [26] Y. Linde, A. Buzo, and R. M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Trans. on Commun.*, vol. COM-28, pp. 84-95, 1980.
- [27] C. H. Lee, "Study on Vector Quantization for Image Compression," *Ph.D. dissertation, Institute of Computer and Information, NCTU, Hsinchu, Taiwan*, 1995.
- [28] M. Nasrabadi and R.A. King, "Image coding using vector quantization: A Review," *IEEE Trans. Commun.*, vol. COM-36, pp. 957-971, Aug 1980.
- [29] Faouzi Kossentini and M. Smith,"a fast searching technique for residual vector quantizer," *signal processing letters*, vol.1, pp.144-116, July 1994.
- [30] R. G. Gallager,"*Information theory and reliable communicatio*", John Wiley & Sons, inc, Newyork, 1968.
- [31] J. Ziv and A. Lempel, "A universal algorithm for data compression," *IEEE Transactions on Information Theory*, vol. 23, no. 3, pp. 337(343, 1977).
- [32] C. E. Shannon, "Coding theorems for a discrete source with a fidelity criterion," in *IRE National Convention Records*, 1959, pp. 142-163.
- [33] T. D. Lookabaugh and R. M. Gray, "High-resolution quantization theory and the vector quantizer advantage," *IEEE Transactions on Information Theory*, vol. 35, no. 5, pp. 1020-1033, 1989.



Help Site Map Contact us

Go

Powered by ASCI

Home New Submission In Process Published Rejected Tools Logout

Welcome sikandar

Papers: In Process

- 70-ITJ-2K6
- 161-ITJ-2K6
- 38-ITJ-2K6

70-ITJ-2K6

Title Lossless Image Coding Using Conditional Entropy Constrained Vector Quantization

Payment Clear

- Status
- Checked Galley Proof Received Back.
  - Ready for Press.

Send Revised Copy | Send Galley Proof Corrections

Download Corner

- ⌘ Acceptance
- ⌘ Invoice
- ⌘ Receipt of Payment
- ⌘ Comments
- ⌘ Galley Proof
- ⌘ Acrobat Reader

Message Board for 70-ITJ-2K6

Query

Text input field for query

POST

CLEAR

## Receipt of Payment

**Asian Network for  
Scientific Information**  
308-Lasani Town,  
Sargodha Road,  
Faisalabad, Pakistan

Tel: 0092-0412001145-46  
Fax: 0092-21-5206036  
Tax ID: 2476222-9

May 02, 2006

To: Dr. Malik Sikandar H. Khiyal  
Department of Computer Science,  
International Islamic University,  
Islamabad, Pakistan  
Ph: 92 51 9257951

Received a sum of 2000 Rs with thanks from Dr. Malik Sikandar H. Khiyal as publication cost of  
Article # 70-ITJ-2K6 through Bank Draft.



Muhammad Imran Pasha  
Account Manager

# Lossless Image Coding Using Conditional Entropy Constrained Vector Quantization

<sup>1</sup>Kishwar R. Naushahi, <sup>2</sup>Mohammad A.U.Khan, <sup>1</sup>M. Sikander H. Khiyal  
<sup>1</sup>Department of Computer Science, International Islamic University, Islamabad.  
<sup>2</sup>Department of Electrical Engineering, COMSATS Institute of Information  
Technology, H-8 Islamabad Pakistan

---

**Abstract** - Lossless coding guarantees that the decompressed image is absolutely identical to the image before compression. This is an important requirement for some application domains, e.g. medical imaging, where not only high quality is in demand, but unaltered archiving is a legal requirement. In this paper, an entropy constrained vector quantization is proposed for lossless compression of image. The method consists of first quantizing the input image using conditional entropy constrained vector quantizer and then coding the residual images using entropy coder. Experimental results show that the new method outperforms standard entropy constrained vector quantization to achieve lossless compression while also requiring lower encoding complexity and memory requirements.

**Keywords:** Lossless Compression, Vector Quantization, Entropy-Constrained.

---

## INTRODUCTION

Most data that is inherently discrete needs to be compressed in such a way that it can be recovered exactly, without any loss. The design of a data compression system consists of two distinct stages: the modeling and the coding. In the modeling part the structure is selected which determine way the events are to be conditioned, and then relative frequencies of the conditioned events are gathered.

The coding is easily implemented on all sources, stationary or not and the complexity of the model has no effect on the coding unit (Langdon and Rissanen, 1982). Lossless compression requires that the reproduced reconstituted bit stream be an exact replica of the original bit stream. Examples include text, experimental results, statistical database and images. The useful algorithms recognize redundancy and inefficiencies in the encoding and are most effective when designed for the statistical properties of the bit stream.

Lossless compression schemes can be crudely classified as follows:

- Predictive schemes with statistical modeling, in which differences between pixels and their surround are computed and their context modeled prior to coding,

- Transform based coding, in which images are transformed into the frequency or wavelet domain prior to modeling and coding.
- Dictionary based schemes, in which strings of symbols are replaced with shorter (more probable) codes,
- Ad hoc schemes (such as run length encoding).

## Lossless coding techniques

- Run length encoding
- Huffman encoding
- Entropy coding (Lempel/Ziv)
- Area coding
- 

Run length coding is easily implemented, either in software or in hardware. It is fast and very well verifiable, but its compression ability is very limited. The basic idea in Huffman encoding is to assign short codewords to those input blocks with high probabilities and long codewords to those with low probabilities. The compression ratio achieved by Huffman encoding uncorrelated data becomes something like 1:2. On slightly correlated data, as on images, the compression rate may become much higher, the absolute maximum being defined by the size of a single input token and the size of the shortest possible output token ( $\text{max. compression} = \frac{\text{token size}[\text{bits}]}{2[\text{bits}]}$ ). While standard

---

**Corresponding Author:** M.sikandar H. Khiyal, Department of Computer Science, International Islamic University, Islamabad, Pakistan, Tel: 92 51 9257951

palletized images with a limit of 256 colors may be compressed by 1:4

There is a wide range of so called modified *Lempel/Ziv coding*. These algorithms all have a common way of working. The coder and the decoder both build up an equivalent dictionary of Meta symbols, each of which represents a whole sequence of input tokens. If a sequence is repeated after a symbol was found for it, then only the symbol becomes part of the coded data and the sequence of tokens referenced by the symbol becomes part of the decoded data later. As the dictionary is build up based on the data, it is not necessary to put it into the coded data, as it is with the tables in a Huffman coder. This method becomes very efficient even on virtually random data.

**Huffman vs. Entropy coding:** A Huffman encoder takes a block of input characters with *fixed* length and produces a block of output bits of *variable* length. It is a fixed-to-variable length code. Lempel-Ziv, on the other hand, is a variable-to-fixed length code. The design of the Huffman code is optimal (for a fixed block length) assuming that the source statistics are known a priori. The Lempel-Ziv code is not designed for any particular source but for a large class of sources. Surprisingly, for any fixed stationary source, the Lempel-Ziv algorithm performs just as well as if it was designed for that source. Mainly for this reason, the Lempel-Ziv code is the most widely used technique for lossless file compression.

Area coding is an enhanced form of run length coding, reflecting the two dimensional character of images. This is a significant advance over the other lossless methods.

**Vector Quantization with Lossless:** Vector quantization is a lossy process so, why this lossy compression technique used to achieve lossless? It's a fact that quantization often produce a structure where high order statistical dependencies can be exploited. Moreover the output of quantizer can be made smaller than that of the original signal; the complexity of high order statistical modeling is reduced. This is especially the case when structurally constrained quantizers are employed (Yu et al., 1994). Entropy coding is now being used frequently in conjunction with vector quantization for image coding. Its use is motivated by the fact that the probability distribution of VQ coded images is generally skewed or non-uniform. While the average bit rate can most often be reduced by entropy coding the VQ codewords, improvement in the rate distortion performance is usually attainable by embedding the entropy coding in the design process

1

such that both the VQ codebook and entropy coder are optimized jointly (Kossentini and Wilson, 1996).

## MOTIVATION

The advantage of residual VQ over other VQ methods (Faouzi and Mark, 1995), is achieved mainly by exploiting the statistical dependencies among the VQ stages. The hybrid technique of quantization and entropy coding of the residual signal has been shown to give up good compression performance.

RVQ, at subsequent stages, input residuals are quantized and output residuals are computed, leading to successive refinement in the accuracy of the overall representation. Each stage VQ index or symbol is then mapped into a variable-length codeword based on probabilities that are conditioned on previous stage output.

This method outperforms standard-constrained residual vector quantization while also requiring lower encoding complexity and memory requirements.

Entropy constrained residual vector quantization (EC-RVQ) has been shown to be a competitive compression technique. Its design procedure is an iterative process which typically consists of three steps: encoder update, decoder update, and entropy coder update.

An adaptive arithmetic coder was used to encode the output of stage RVQ's and the residual images, the comparison ratios were slightly larger. Even better compression performance may be attained by using larger vector sizes and exploiting any statistical dependencies between the multistage images and the residual one. The entropy is used as a measure so that the comparison was fair. Preliminary experimental results are encouraging further study.

Like EC-VQ, EC-RVQ is memoryless vector quantizer. This is because the EC-RVQ design algorithm minimizes the distortion subject to a constraint on the first order or zero order conditional entropy of the vector quantizer output (Kossentini and Wilson, 1996).

## PROPOSED FRAMEWORK

We have designed a lossless image compression CEC-VQ architecture, based on a single stage vector quantizer and entropy coder.

The advantage of high-order entropy coding over the normally used first-order entropy coding has been revealed in Shannon's information theory. The high-order entropy of a source can be effectively exploited by using either the joint probability of  $(L + 1)$  symbols or the conditional probability of the current



symbol with the knowledge of its L previous symbols (Lei et al., 1993).

In particular, the structure of vector quantizer used in (Kossentini et al., 1995) has been shown to be very successful in providing more accurate estimates of

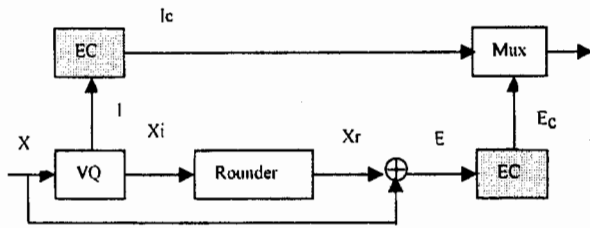


Fig 1: Proposed Lossless CEC-VQ coder

the statistical dependencies of the original signal while also reducing drastically the complexity of high order statistical modeling in multistage RVQ.

As shown in Figure 1, we employ a CEC-VQ to quantize the input signal, where the output of VQ is then fed into a Entropy Coder(EC).The quantized signal is rounded to the nearest integer, and the residual signal, form by subtracting the rounded quantized signal from the original one, is then coded using a entropy coder. Empirical work has shown that using higher order entropy coding does not lead to significant reduction in output entropy of the residual signal (Faouzi and Mark, 1995). Conventional distortion measures does not lead to minimization of the residual entropy, the overall system is lossless, its better to use the entropy of the residual signal as a distortion measure in the design of CEC-VQ. We used Entropy as measure of distortion.

The distortion measures used in the design of CEC-RVQ is,  $d(x,y) = -\log_2[\text{pr}(l(x-y))]$ , where  $l(\alpha)$  is the integer closest to the real  $\alpha$  (Faouzi and Mark, 1995). VQ designed to minimize such a distortion measure also minimize the entropy of the residual signal.

The second idea is that only entropy is a measure of performance(Faouzi and Mark, 1995). In figure 1,As illustrated, the encoding component consist of a single stage VQ, producing the output indices I. Associated with a fixed number of vector codebook. Initially, vector quantizer generates a codebook that have MSE distortion measure. This codebook has provided for next step of generation progression to minimize the entropy.

Our model based upon a single stage quantizer so it has reduced the complexity of entropy coder that construct by multiple stage in RVQ. Thus, the complexity and memory of the entropy coder grow exponentially with the number of conditioning symbols

and the output alphabet sizes of the stage VQs (Kossentini et al., 1994).

## RESULTS

CEC-VQ was examined carefully in the context of image coding. Several images of size 512x512 were taken from the USC database to design the CEC-RVQ codebook. We implemented an image codec where 4x4 blocks of residuals are used as vector. For this, we used a set of 4 and 8, head and shoulder gray level images.

Table 1: Results of Measuring Techniques

Measuring techniques	Required result
MSE	0
SNR	Inf db
PSNR	Inf db



Fig 2(a): Original Image



Fig 2(b): Reconstructed image

Table 1 shows the different objective measures calculated to prove the validity of our results. After compressing the input image and then decompressing the output image, the fidelity criteria were extracted using the MSE, SNR, and PSNR.

Obviously, the values in the above table indicate a perfect reconstruction, as shown in fig. (b) of the original image fig. 2(a)

Different images were tested successfully for their perfect reconstruction. Some of the images tested so far includes Ano, Aben, Barbara, Elaine, girl1, girlb. Perhaps even more significant is the fact that improvement in bit rate can be achieved without the enormous storage and complexity requirements that accompany higher-order conditional EC-VQ, finite state VQ and other predictive schemes of this type (Kossentini and Wilson, 1996). Experimental Resulted Bit rate validate it in this research.

Table 2. Bit rate (bpp) of LENA (Compressed image ) using 4 and 8 training set of image.

Bit rate (bpp)	Code book size				
	256 k	128 k	64 k	32 k	16 k
4 images	4.925	4.995	5.084	5.17	5.34
T-set					
8 images	4.913	4.956	5.084	5.17	5.34
T-set					

Table 2 shows the reduced bit rate that was achieved after the compression of 8-bit gray scale images of Lena. In Table 2, we can clearly justify that in a training set of 8 or 4 images, the compression is exactly the same. It is only when the size of the codebook increases then a fractional difference is introduced.

The work proposed by (Faouzi and Mark, 1995) was further built upon thus leading to an entirely new framework. They used high order entropy; whereas we have introduced lossless compression using entropy constrained VQ. This method is more efficient and has a lower complexity as compared to the work done by the afore-mentioned authors.

## CONCLUSIONS

The goal of our work was to obtain perfect reconstruction of an image. Here, the decompressed image should be a replica of the original. We have obtained 100% lossless image compression. Our system was tested on different images using codebooks of different sizes such as 256, 128, 64, 32 etc.

The codebook generation was a time consuming process. There is room for improvement in the sense that it could be made more efficient. This framework has experimented on 8-bit gray level images. Enhancements can be made for 16-bit and colored images.

In this work, we only consider 4x4 vectors; relatively large vector size provides an information theoretic benefit while still requiring manageable complexity and memory and the block sizes can be increased.

## REFERENCES

- Kossentini, F. and Mark J. T. Smith, (1995), High order entropy constrained Residual VQ for Lossless compression of images, in *ICIP, (Austin, TX)*, pages 2338-2341.
- Kossentini, F., Wilson C. Chung, (1996), Conditional Entropy-constrained Residual VQ with Application to Image Coding, *IEEE Transactions on image processing*, 5(2): 311-320.
- Kossentini, F., W. Chung, and M. Smith, (1995) Subband image coding with jointly optimized quantizers, in *ICASSP, (Detroit, MI)*.
- Kossentini, F., W. Chung, and M. Smith, (1994), Image coding using high-order conditional entropy constrained residual VQ, in *ICIP, (Austin, TX)*, 613-617.
- Langdon, G.G. and Rissanen, 1982]G. G Langdon and J. Rissanen, (1982), Compression of black-white images with arithmetic coding, *IEEE Trans. Comm*, 29(6):858-867.
- Lei, S.M., T. C. Chen, and K. H. Tzou, (1993), Subband HDTV coding using higher-order conditional statistics, *IEEE JSAC*, vol. 11, pages 65-76.
- Yu, S., M. N. Wernick, and N. P. Galatsanos, (1994) lossless compression of multidimensional medical image data using binary-decomposed high-order entropy coding, in *ICIP, (Austin, TX)*, pages. 351-355.