1993

A comparative study of image compression techniques within a noisy channel environment

Ahmed Yehia Banafa
Lehigh University

AUTHOR:

Banafa, Ahmed Yehia

TITLE:

A Comparative Study of Image Compression Techniques Within a Noisy Channel Environment

DATE: May 30, 1993

A Comparative Study of Image Compression Techniques within a Noisy Channel Environment

by

Ahmed Yehia Banafa

A Thesis

Presented to the Graduate and Research Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Electrical Engineering

Lehigh University

1993

This thesis is accepted and approved in partial fulfillment of the requirements for the Master of Science

May 12, 1993

Date

Thesis Adviser

Chairperson of Department

Acknowledgements

I would like to express my deep thanks and appreciating to my father-in-law who support me during the past two years.

I want to specially thank my family here in the U.S. and in Saudi Arabia for their continue encouragement and support.

Special thanks to the following friends who help me in making this dream real, Muhammed Khan (U. of Kuwait), Stephen Corbesero (Lehigh system manager) and Ahmed Balamesh (U. of Michigan).

Finally, Special Thanks to the man who stand behind all this study: my Adviser Prof. Weiping Li.

LIST OF TABLES

LIST OF FIGURES

Abstract

Image compression is no longer a theoretical curiosity, it's now a practical requirement in many data systems.

A detailed review of some important image compression techniques is presented, with more concentration on Vector Quantization (VQ), Disceret Cosine Transform (DCT), and Vector Transform Coding (VTC).

In practice, compressed data are transmitted through different media, where these data are exposed to many kinds of channel noise.Therefore, understanding the effects of channel noise on compression techniques is very important in many aspects, for instance, it helps in designing better encoder and decoder, optimization of the coding schemes, and more. In this thesis a study of channel noise (with uniform distribution) effects on three specific coding schemes, VQ, DCT, and VTC, is presented. Depending on this study, a ranking of the three techniques is presented.

Further study can be considered with different noise distribution, e.g., Gaussian distribution. The same procedure in evaluating these three techniques can be applied to other coding schemes.

Chapter 1

Introduction

The aim of this study is to present the effect of applying channel noise on three specific coding techniques, used currently in digital image compression.

Finally, comparing the performance of these three techniques under such kind of noise, and suggesting future directions for researches in this issue.

1.1 The information revolution

We live in a world in which electronic communication is so commonplace that we pick up our cordless telephones without a second thought. Yet the importance of such communication in today's world is so crucial that we cannot imagine modern society without it. We are now in an era of change, which some

people refer to as the information age, much like the era - more than 100 years ago - when the world underwent drastic changes because of the industrial revolution. From now on, the prosperity and continued development of modern nations will depend primarily on the originating and disseminating of *information,* rather than of manufactured goods [44]. For example the Gulf War in 1990, demonstrate the importance of having the lead in information transfer through, different places and times, furthermore, the rules of T.V. in bringing the actions and atmosphere of that war to our living room, were very clear.

This gives an idea about how is the information effecting our daily life, besides, almost every day we hear about, or read about, new concepts such as Electronic-mail, ISDN[1], Teleconferencing, HDTV[2], MULTI-MEDIA[3], and more.

Furthermore, images are a form of information that has a specially relevant role for human [4], this shows the importance of better understanding of processing digital images (including compression), to get the optimum possible output.

---

[1] Integrated Services Digital Network.

[2] High Definition TeleVision

[3] Utilization of video, speech, written words in one environment , Multimedia combines components from many sources. The creation part of the process involves a wide variety of equipments. The playback system requires a CD-ROM drive, an audio board, and speakers, along with a PC with an appropriate interface and display and enough storage [8] .

1.2 Historical Background

Interest in image compression dates back more than 25 years. The initial focus of research efforts in this field was on the development of analog methods for reducing video transmission bandwidth, a process called *bandwidth compression*. The advent of the digital computer and subsequent development of advance integrated circuits, however, caused interest to shift from analog to digital compression approaches. With the adoption of several key international image compression standards, the field is now poised for significant growth through the practical application of the theoretic work that began in the 1940's when C. E. Shannon and others first formulated the probabilistic view of information and it's representation, transmission, and compression [19] .

Over the years, the need for image compression has grown steadily. Currently, it is recognized as an "enabling technology. " For example, image compression has been and continues to be crucial to the growth of multimedia computing. In addition, it is the natural technology for handling the increasing spatial resolutions of today's imaging sensors and evolving broadcast television standards. Furthermore, image compression plays an extremely important role

in many important and diverse applications, including televideo-conferencing, remote sensing[4], document and medical imaging, facsimile transmission (FAX), and the control of remotely piloted vehicles in military, space, and hazardous waste control applications. In short, an ever-expanding number of applications depend on efficient manipulation, storage, and transmission of binary, gray - scale , or color images [13].

[4] Remote Sensing is the use of satellite imagery for weather and other earth-resources application [19] .

1.3 Need for compressing digital images

Image compression is no longer a theoretical curiosity- it is now a practical requirement in many data systems, with the advance in electronic technology, it is now possible to implement in hardware many image compression techniques that formally were executed only on large-scale general purpose digital computers[16] .

In the last few years there has been a growing interest in applying image compression techniques to actual image and communication systems in the commercial sector (compression of newspaper page for transmission), in the military (video compression for remotely piloted vehicles), and in government agencies, such as NASA (image compression for spacecraft). In each potential application there is a need to learn what compression techniques are available how they operate, and what the implementation considerations are for each technique [16].

Moreover, with the continuing growth of modern communications technology, demand for image transmission and storage is increasing rapidly. Advances in computer technology for mass storage and digital process have paved the way for implementing advance data compression techniques to improve the efficiency of transmission and storage of images [9].

Therefore, with the understanding of the fact that an enormous amount

of data is produced when *for example* a 2-D light intensity function is sampled and quantized to create a digital image. In fact, the amount of data generated may be so great that it results, in impractical storage, processing, and communications requirements [13].

Furthermore, image compression address the problem of reducing the amount required to represent a digital image. The underlying basis of the reduction process is the removal of redundant data. From a mathematical viewpoint, this amount to transforming a 2-D pixel array into a statistically uncorrelated data set. The transformation is applied prior to storage or transmission of image. At sometime, the compressed image is decompressed to reconstruct the original image or approximation to it [13].

1.3.1 Types of compression

Data (*including images*) - compression techniques, can be divided into two major families: *Lossy* and *Lossless* .

Lossy data compression concedes a certain loss of accuracy in exchange for greatly increased compression. Lossy compression proves effective when applied to graphics, images, and speech. Most lossy compression techniques can be adjusted to different quality levels, giving a higher accuracy in exchange for less effective compression [16].

Lossless compression consists of those techniques that 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 data base, records, and word processing files. In these applications the loss of even a single bit could be catastrophic, as for example in the case of medical records [16].

1.4 Fundamentals of digital image compression

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. In fact, data are the means by which information is conveyed. Various amounts of data may be used to represent the same amount of information [13].

Data redundancy is a central issue in digital image compression. It is not an abstract concept but a mathematically quantifable entity. If n_1 and n_2 denote a number of information carrying units in two data sets that represent the same information, the *relative data redundancy* R_D of the first data set (the one characterized by n_1) can be defined as

$$R_D = 1 - \frac{1}{C_R} \qquad (1.1)$$

where C_R,, commonly called the compression ratio, is

$$C_R = \frac{n_1}{n_2} \qquad (1.2)$$

For the case $n_2 = n_1$, $C_R = 1$ and $R_D = 0$, indicating that (relative to the second data set) the first representation of the information contains no redundant data.

When $n_2 \ll n_1$, $C_R \to \infty$ and $R_D \to 1$, implying significant compression and highly redundant data. In final case, $n_2 \gg n_1$, $C_R \to 0$ and $R_D \to \infty$, indicating that the second data set contains much more data than the original representation. This of course, is normally undesirable case of data expansion. In general, C_R and R_D lie in the open intervals (0, ∞) and (- ∞ , 1), respectively. A practical compression ratio such as 10 (or 10 : 1) means that the first data set has 10 information carrying units (e.g., bits) for every 1 unit in the second or compressed data set. The corresponding redundancy of 0.9 implies that 90 percent of the data in the first data set is redundant.

In digital image compression, three basic data redundancies can be identified and exploited : *coding* redundancy, *interpixel* redundancy, and *psychovisual* redundancy. Data compression is achieved when one or more of these redundancies are reduced or eliminated [13].

1.4.1 Coding redundancy

A code is a system of symbols (letters, numbers, bits, and the like) used to represent a body of information or set of events. Each pieces of information or event is assigned a sequence of *code symbols*, called a *code word*. The number of symbols in each code word is it's *length*.

In general coding redundancy is present when the codes assigned to a set of events (e.g., gray- level values) have not been selected to take full advantage of the probabilities of the events [13].

1.4.2 Interpixel redundancy

It is another important form of data redundancy, one directly related to the interpixel correlation within an image. Because the value of any given pixel can be reasonably predicated from the value of its neighbors, the information carried by individual pixels is relatively small. Much of the visual contribution of a single pixel to an image is redundant; it could have been guessed on the basis of it's neighbors values. A variety of names have been coined to refer to these interpixel dependencies, including spatial redundancy, geometric redundancy, and intraframe redundancy. The term interpixel redundancy encompasses them all [13].

1.4.3 Psychovisual redundancy

This type of redundancy needs more explanation, it can be defined as follows: The brightness of a region, as perceived by 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 *psychovisually redundant*. It can be eliminated without significantly impairing the quality of image perception [13].

Psychovisual redundancy is fundamentally different from the redundancies discussed earlier. Unlike coding and interpixel redundancy, psychovisual redundancy is associated with real quantifiable visual information. Its elimination is possible only because the information itself is not essential for normal visual processing. Since the elimination of psychovisually redundant data results in a loss of quantitative information, it commonly referred as *quantization*. This terminology is consistent with the 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 an irreversible operation (visual information is lost), quantization results in lossy data compression [13].

1.4.4 Fidelity criteria

As noted previously, removal of psychovisually redundant data results in a loss of real or quantitative visual information. Because information of interest may be lost, a reproducible means quantifying the nature and extent of information loss is highly desirable. Two general classes of criteria are used as basis for such an assessment:

(1) *Objective fidelity criteria.*

(2) *Subjective fidelity criteria.*

Objective fidelity criteria:

When the level of information loss can be expressed as a function of the original image and the compressed and the decompressed output image, it said to be based on an *objective fidelity criterion*. A good example is the root-mean-square (rms) error between an input and output image.

Let $f(x,y)$ represent an input image and let $\hat{f}(x,y)$ denote an approximation of $f(x,y)$ after compressing and decompressing. For any value of x, y, the error $e(x,y)$ between $f(x,y)$ and $\hat{f}(x,y)$ can be defined as

$$e(x,y) = \hat{f}(x,y) - f(x,y) \tag{1.3}$$

so that the total error between the two images is

$$\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\hat{f}(x,y) - f(x,y)] \tag{1.4}$$

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

$$e_{rms} = [\frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\hat{f}(x,y) - f(x,y)]^2]^{\frac{1}{2}} \tag{1.5}$$

A closely related objective fidelity criterion is the mean-square signal-to-noise ratio of the compressed-decompressed image. If $\hat{f}(x,y)$ is considered (by a simple arrangement of the terms in Eq. 1.3) to be the sum of the original image $f(x,y)$, and the noise signal $e(x,y)$, the *root-mean-squared signal-to-noise ratio* of the output image, denotes SNR_{rms}, is

$$SNR_{rms} = \sqrt{\frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\hat{f}(x,y)^2}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}[\hat{f}(x,y) - f(x,y)]^2}} \quad (1.6)$$

Subjective fidelity criteria:

Although objective fidelity criteria offer a simple and convenient mechanism for evaluating information loss (the quality of reproduction), most decompressed images ultimately are viewed by human beings. Consequently, measuring image quality by the subjective evaluations of human observer often is more appropriate. This can be done by showing a "typical" decompressed image to an appropriate group of viewers and then averaging their evaluations. The evaluations may be made using an absolute rating scale or by means of side-by-side comparisons of $f(x,y)$ and $\hat{f}(x,y)$. Table 1.1 shows one possible absolute rating scale. For the case of side-by-side comparisons the rating could be like {-3,-2,-1,0,1,2,3}, representing the subjective evaluation { *much worse, worse, slightly worse, the same, slightly better, better, much better*}, respectively. In either case, the evaluations are said to be based on *subjective fidelity criteria* [13].

Value	Rating	Description
1	Excellent	An image of extremely high quality, as good as you could desire.
2	Fine	An image of high quality, providing enjoyable viewing. Interference is not objectionable.
3	Passable	An image of acceptable quality, interference is not objectionable.
4	Marginal	An image of poor quality; you wish you could improve it. Interference somewhat objectionable.
5	Inferior	A very poor image, but you could watch it. Objectionable interference is definitely present.
6	Unusable	An images so bad that you could not watch it.

Table 1.1 : Television Allocations Study Organization Rating Scale (From Frendendall and Behrend [1960]).

1.5 Image Compression Models

As Fig 1.1 shows, an image compression system consists of two distinct structural blocks: an encoder and decoder. An input image $f(x,y)$ is 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 $\hat{f}(x,y)$ is generated [13].

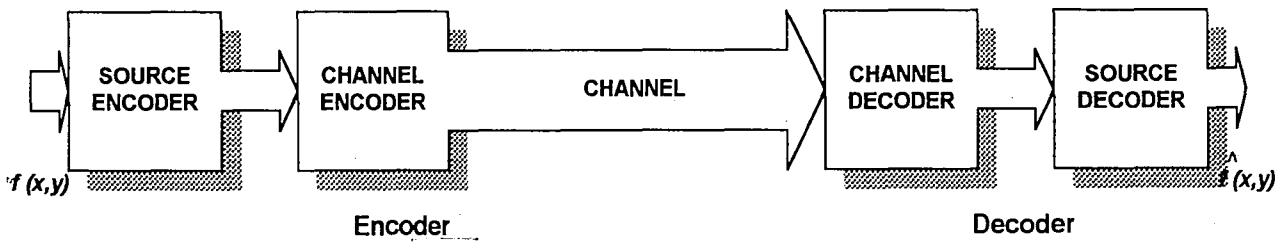

Figure. 1.1 : A general compression system model.

Both the encoder and decoder shown in Fig. 1.1 consist of two subblocks. The encoder is made up of a *source encoder*, which removes input redundancies, and *channel encoder,* which increase, the noise immunity of the source encoder's output. On the other side, the decoder also includes a *channel decoder* followed by a *source decoder*. If the channel is noise free (not prone to error), the channel encoder and decoder are omitted [13].

Channel Errors (noise):

According to Fig. 1.1 The channel encoder and decoder play an important role in the overall encoding-decoding process when the channel is noisy.

They are designed to reduced the impact of channel noise by inserting 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 " [13]. In practice, error correcting bits are appended to quantized samples before encoding to compensate for channel errors. Often, the error correcting codes used are designed to reduce the probability of bit errors, and for simplicity, equal protection is provided to all the samples.

To account for channel errors, one has to add redundancy to the input. On the other hand, the data compression techniques tend to remove the redundancy in the source data.

Thus a proper *tradeoff* between source coding (redundancy removal) and channel coding (redundancy injection) has to be achieved in the design of data compression system [9].

In this study; the effects of one type of channel noise distribution, which is the uniform distribution, on three coding technique namely: Vector Quantization, Discrete Cosine Transform, and Vector Transform Coding, will be investigated.

Chapter 2

Vector Quantization

Vector quantization was introduced in the late 1970's as a scheme for effectively mapping a sequence of vectors into a digital sequence of numbers. The technique employed uses a multidimensional extension of a simple algorithm by Lloyd due to Lide, Buzo, and Gray.

A vector quantizer (block quantizer, K-dimensional quantizer, block source code, or matrix quantizer) consists of a reproduction alphabet or codebook $Y = \{\hat{x}_i; i = 1,2,...N\}$ of N codewords together with a mapping $Q(.)$ that assigns to each K-dimensional real value input vector x a codeword $\hat{x}_i \epsilon$ Y.

For waveform coding, the codewords $\hat{x}_i$ should be similar to the source vectors x-that is, the codewords should be also K-dimensional real-value vectors. In the pattern recognition literature, the codewords are called the reference patterns or templates. The size N of the codebook is also called the number of levels and it determines the encoding rate.

A vector quantizer can also be seen as a combination of two functions:: an ***encoder*** , which observes a particular input vector x and generates the address i of the codeword $\hat{x}_i$ specified by $Q(x)$,and a matching ***decoder,*** which uses this address to generate the reproduction vector $\hat{x}_i$. When x is quantized as $\hat{x}_i$ a quantization error results with respect to a distortion measure $d(x, \hat{x}_i)$.

The distortion measure represents the penalty or cost associated with reproducing vector x by $\hat{x}_i$. Then the best mapping $Q(.)$ is the one that minimizes $d(x, \hat{x}_i)$,the square error distortion [1] .

2.1 Definition of Vector Quantization

A vector quantizer can be defined as a mapping Q of K-dimensional Euclidean space R^k into a finite subset Y of R^k . Thus,

$$Q: R^k \rightarrow Y \tag{2.1}$$

Where $Y = (\hat{x}_i ; i = 1,2,...,N)$ is the set of the reproduction vectors and N the number of vectors in Y. It can be also seen as a combination of two functions:: an encoder , which views the input vector x and generates the address of the reproduction vector specified by $Q(x)$, and a decoder , which uses this address to generate the reproduction vector $\hat{x}$ [19].

2.2 Memoryless Vector Quantization

Vector quantization is a method for the mapping of an input sequence of vectors into another sequence of discrete vectors (as stated in the previous section). The goal in such operation is data compression.

The simplest VQ is the *memoryless vector quantizer*. With this type, instead of quantizing scalars individually, as in PCM, they are combined in blocks and then jointly quantized. In this respect, memoryless VQ is a generalization of PCM to the multidimensional case [4].

Data compression is achieved because a short code is assigned to each discrete vector, and only the code needs to be transmitted or stored. As shown in Figure 2.1, each vector in the input sequence is mapped, using *nearest neighbor rule,* to a discrete vector, the index of disceret vector is coded and transmitted. At the decoding stage, the discrete vector is looked up in the table

using the transmitted or stored code [4].

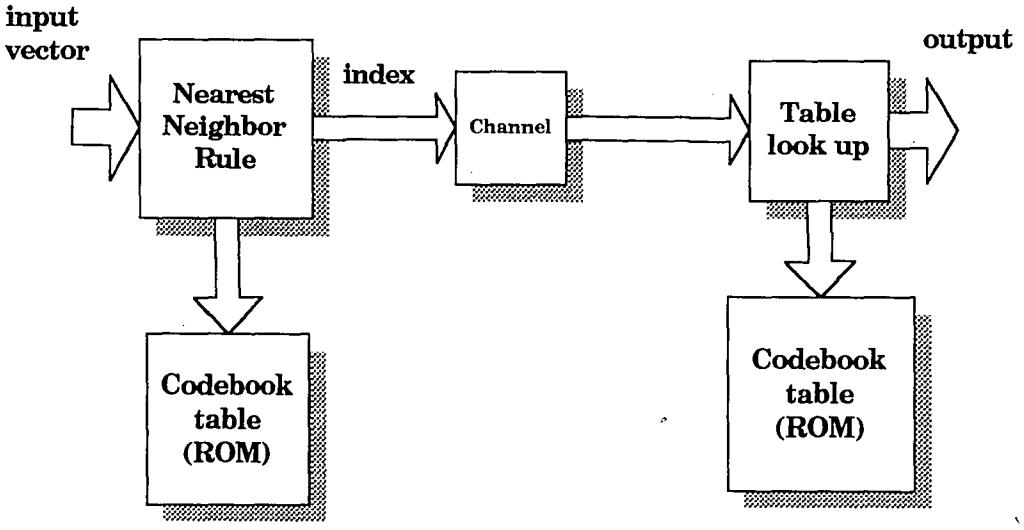

Figure 2.1: Block diagram of memoryless VQ.

Mathematically, memoryless vector quantization consists of two mapping:The encoder which assigns to each input vector $\mathbf{x} = (x_0, x_1, \ldots, x_{k-1})$ a channel symbol $\gamma(\mathbf{x})$ in some channel symbol set $\mathbf{M}$, and a decoder assigning to

each channel symbol in **M** a value in the reproduction alphabet **A.** The channel symbol set can be assumed as the space of all 2^R binary R-dimensional vectors. The reproduction alphabet **A** is a subset of the input vector space. With M elements in **M**, $R = \log_2 M$ is the rate of the quantizer in bits per vector, and $r = R/k$ is the rate in bits per sample, where k is the number of elements in the vector samples $(x_0,x_1,...,x_{k-1})$. In an image compression system the vectors can be consecutive subrasters and the samples typically are pixel values [4].

An important characteristic to note here is that in the case of PCM only integer number of bits per sample is possible, whereas VQ allows fractional values. This limits the bits per sample in PCM to 1 bit per sample or more, but VQ can have less than 1 bit per sample [4].

Since the goal of this quantization is data compression, the minimization needed is a "good" reproduction with the lowest bit rate R. To quantify this "good" reproduction, a measure of distortion is needed [4].

Distortion

A distortion measure d is an assignment of a cost $d(x,\hat{x})$ of reproducing any input vector x as a reproduction vector $\hat{x}$. Given such a distortion measure, we can quantify the performance of a system by an average distortion $Ed(x,\hat{x})$ between the input and the final reproduction: A system will be good if it yields a small average distortion [2].

In practice, the important average is the long term sample average or time average

$$\lim_{n\to\infty}\frac{1}{n}\sum_{i=0}^{n-1} d(x_i,\hat{x}_i) \qquad (2.2)$$

provided, of course, that the limit makes sense. Ideally a distortion measure should be tractable to permit analysis, computable so that it can be evaluated in real time and used in minimum distortion systems, and subjectively meaningful so that large or small quantitative distortion measures correlate with bad and good subjective quality [2].

The squared error distortion measure:

Here the input and reproduction spaces are K-dimensional Euclidean space

$$d(x,\hat{x}) = \|x-\hat{x}\| = \sum_{i=0}^{k-1}(x_i-\hat{x}_i)^2, \qquad (2.3)$$

the square of euclidean distance between vectors. This is the simplest distortion measure and the most common for waveform coding. It's a common practice to

measure the performance of a system by the signal - to - noise ratio (or signal - to - quantization - noise ratio)

$$SNR = 10\log_{10}\frac{E(\|x\|^2)}{E[d(x,\hat{x})]} \qquad (2.4)$$

This corresponds to normalizing the average distortion by the average energy and plotting it on a logarithmic scale: Large(small) SNR corresponds to small (large) average distortion [2].

The long term average distortion, as stated above, is what must be minimized. For a stationary and ergodic process, the limiting time average is the mathematical expectation. Although the mathematical expectation is useful for developing theoretical results for performance, it is not very useful for the design of the quantizer since the statistics of the vector sources are not generally known. In the case of images, there isn't any generally accepted accurate probability distribution. Therefore, a common approach is to use a long series of images as a training sequence. Given a way to calculate the average distortion in the images of the training set of a vector quantizer, design a code that minimizes this average distortion [4].

For a stationary and ergodic process, the training sequence approach average should be very close to the expected value. But real sources are not always stationary and ergodic. Nonetheless, if a sufficiently long training sequence is used, the resulting code should preform almost as well on new data

from the same source as it did for the training sequence. A sufficient condition for this to be true is that the source be asymptotically mean stationary, it is not required for it to be stationary and ergodic [4].

properties of optimal quantizers

Two necessary conditions for a VQ to be optimal follow easily using the same logic as in Lloyd's [7] classical development for optimal PCM with a mean square error distortion measure [4]:

1. The encoder that minimizes the average distortion is the one that selects the code that yields the minimum distortion at the output. This means that the best encoder doses nearest neighbor mapping.

2. The decoder that minimizes the average distortion is the one that assigns to each code the generalized centroid of all the input vectors that at the encoder result in that code.

The fact that the encoder can be optimized for the decoder and vice versa formed the basis of Lloyd's original optimal PCM design algorithm for a scalar random variable with a known probability density function and a squared error distortion. The general VQ design algorithms considered are bases on the simple observation that Lloyd's basic development is valid for vectors, for sample distributions, and for a verity of distortion measure. The only requirement on the distortion measure is the one that can compute the centroid. The basic

algorithm is the following [2]:

Step 0. Given: A training sequence and an initial decoder.

Step 1. Encode the training sequence using the given decoder minimum distortion rule. If the average distortion is small enough, quit.

Step 2. Replace the old reproduction vector of each codeword by the centroid of all training vectors which were mapped into that codeword in step 1. Go to 1.

Each step of the algorithm must either reduce average distortion or leave it unchanged. The algorithm is usually stopped when the relative distortion decrease falls below some small threshold. It should be emphasized that such iterative improvement algorithms may not in general yield truly optimum codes.

It is known that the algorithm will yield locally optimum quantizers, but in general there may be numerous such codes and many may yield poor performance. (see, e.g., [2]) It often useful therefore to enhance the .

potential by providing it with good initial codebooks, and by trying it on several different initial codebooks [2].

Initial codebooks

The basic design algorithm is an iterative improvement algorithm and requires an initial code to improve. Two basic approaches have been developed: One can start with a simple codebook of the correct size or one can start with small codebook and recursively construct larger ones [2].

Random codes

The simplest example of the first technique is: Use the first 2^R vectors in the training sequence as the initial codebook. An obvious modification more natural for highly correlated data is to select several widely spaced words from the training sequence [2].

Product codes

Another example of the first approaches is to use a scalar code such as a uniform quantizer k times in succession and then prune the resulting vector codebook down to the correct size. The mathematical model for such a code is a product code. Which can be explained as follows:

Say we have a collection of codebooks, C_i , $i = 0,1,...,m-1$, each consisting of M_i vectors of dimension k_i , and having rate $R_i = log_2 M_i$ bits per vector. Then the *product codebook* C is define as the collection of all $M = \prod_i M_i$ possible concatenations of m words drawn successively from the m codebooks C_i . The

dimension of the product codebook is, $k = \sum_{i=0}^{m-1} k_i$ the sum of the dimensions of the

component codebooks. The products code is denoted mathematically as a Cartesian products:

$$C = \underset{1=0}{\overset{m-1}{X}} C_i = \{all\ vectors\ of\ the\ form\ (\hat{x}_0,\hat{x}_1,\ldots,\hat{x}_{m-1}); \quad \hat{x}_i\ in\ C_i;\ i = 0,1,\cdots,m-1\} \tag{2.5}$$

Thus, for example, using a scalar quantizer with rate R/K, k times in succession yields a product k-dimensional vector quantizer of rate R bits per vector. This product code can be used as initial code for the design algorithm [2].

In the waveform coding applications where the reproduction and input alphabet are the same (k-dimensional Euclidean space) an alternative product code provides a means of growing better initial guesses from smaller dimensional codes [2]. Begin with a scalar quantizer C_0 and use a two - dimensional product code C_0 X C_0 as an initial guess for designing a two - dimensional VQ. On completion of the design we have a two-dimensional code, C^2. From an initial guess for a three-dimensional code as all possible pairs from C^2 and scalars from C_0, that is, use the product code C^2 X C_0 as an initial guess. Continuing in this way, given a good *k-1* dimensional VQ described as codebook C^{k-1} , an initial guess for a k-dimensional code design is the product code C^{k-1} X C_0. One can use such product code constructions with a different initial scalar

code C_0, such as those produced by the scalar version of the *splitting algorithm* [2].

Splitting

In this algorithm, instead of constructing long codes from smaller dimensional codes, we can construct a sequence of bigger codes having a fixed dimension using a " splitting " technique [7], [9]. This method can be used for any fixed dimension, including scalar codes.

In this technique one first finds the optimum 0 rate code - the centroid of the entire training sequence, as depicted on Fig. 2.2a. For a two-dimensional input alphabet. This single codeword is then split to form two codewords (Fig. 2.2b). For example, the energy can be perturbed slightly to form a second distance word or one might purposefully find a word distant from the first.

It is convenient to have the original codewords a member of the new pair to ensure that the distortion will not increase. The algorithm is then run to get a good rate 1 bit per vector code as indicated in Fig. 2.2c. The design continues in this way in stages as shown: The final code of one stage is split to form an initial code for the next [2].

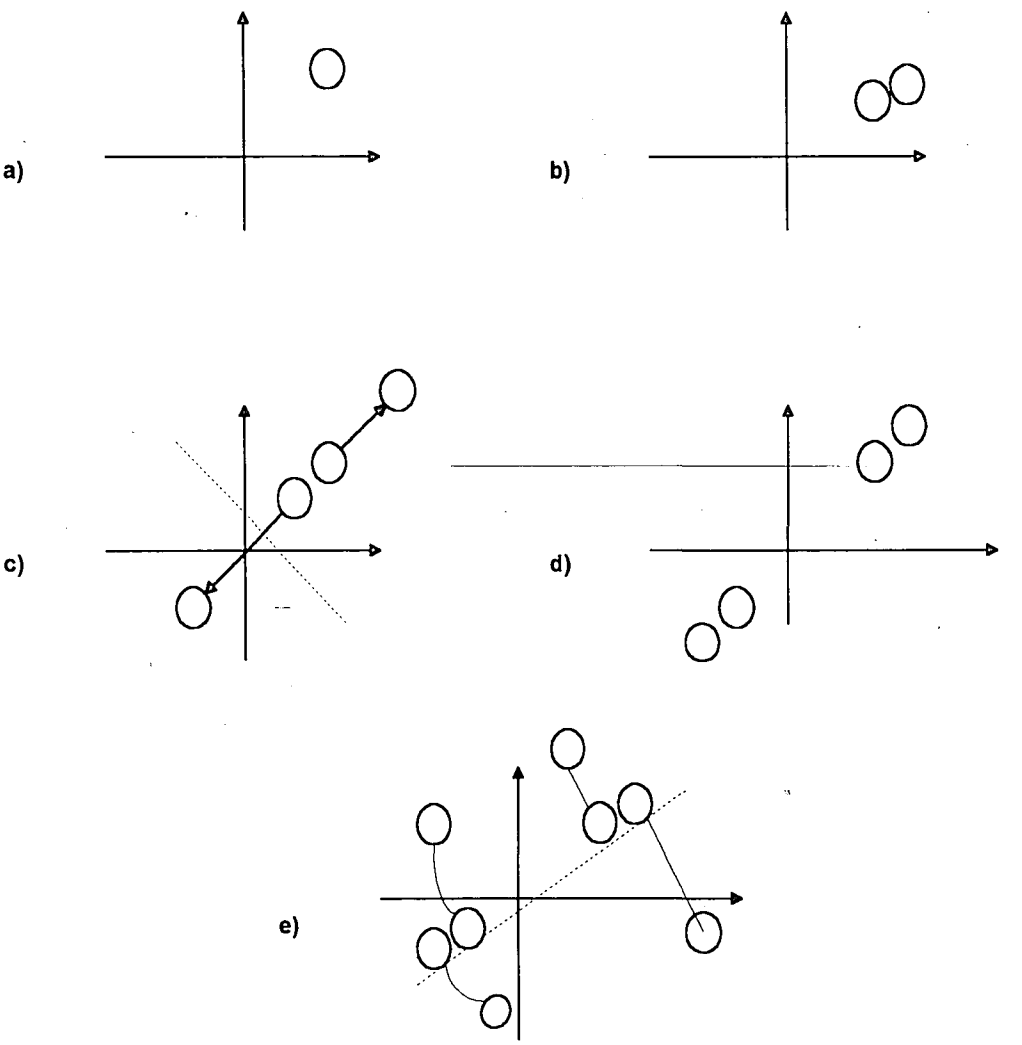

Figure 2.2: Splitting algorithm

The Figure 2.2, can be explained as follows:

A large code is defined in stages: at each stage each codeword of a small code is

split into two new codewords, giving an initial codebook of twice the size. The algorithm is run to get a new better codebook. In part (a) Rate 0: The centroid of the entire training sequence. Part (b) Initial Rate 1: The one codeword is split to form an initial guess for a two-word code. Part (c) Final Rate 1: The algorithm produces a good code with two words. The dotted lines indicate the Voroni regions. Part (d) Initial Rate 2: The two words are split to form an initial guess for a four-word code. Part (e) Final Rate 2: The algorithm is run to produce a final four word code [2].

2.3 Variation of Memoryless Vector Quantization

In this section some of the variations of memoryless vector quantization will be investigated, keeping in mind that these variations are aimed at reducing

the computation or memory requirement of a full search memoryless VQ [2].

Tree-searched VQ

Tree-searched VQ is a natural byproduct of the splitting algorithm for generating initial code guesses. We focus on the case of a binary tree, but more general tree will provide better performance while retaining a significant

reduction in complexity [2].

Suppose we have a good rate 1 code as in Figure 2.2c and we form a new rate two code by splitting the two codewords as in Figure 2.2d. Instead of running a full search VQ design on the resulting 4-word codebook, however, we divide the training sequence in two pieces, collecting together all those vectors encoded into a common word in the 1 bit codebook, that is, all of the training sequence vectors in a common cell of the Voronoi partition. For each of these subsequances of training vectors, we then find a good 1-bit code using the algorithm. The final codebook (so far) consists of the four codewords in the two 1-bit codebooks designed for the two subsequances. A tree-searched encoder selects one of the words not by an ordinary full search of this codebook, but instead it uses the first one bit codebook designed on the whole sequence to select a second code and it then picks the best word in the second code. This encoder can then be used to further subdivide the training sequence and construct even better codebooks for subsequances. The encoder operation is shown in Figure 2.3 [2].

The tree is designed one layer at a time: each new layer being designed so that the new codebook available from each node is good for the vectors encoded into the node. Observe that there are 2^R possible reproduction vectors as in the full search VQ, but now R binary searches are made instead of a single 2^R - ary search. In addition, the encoder storage requirement has doubled.

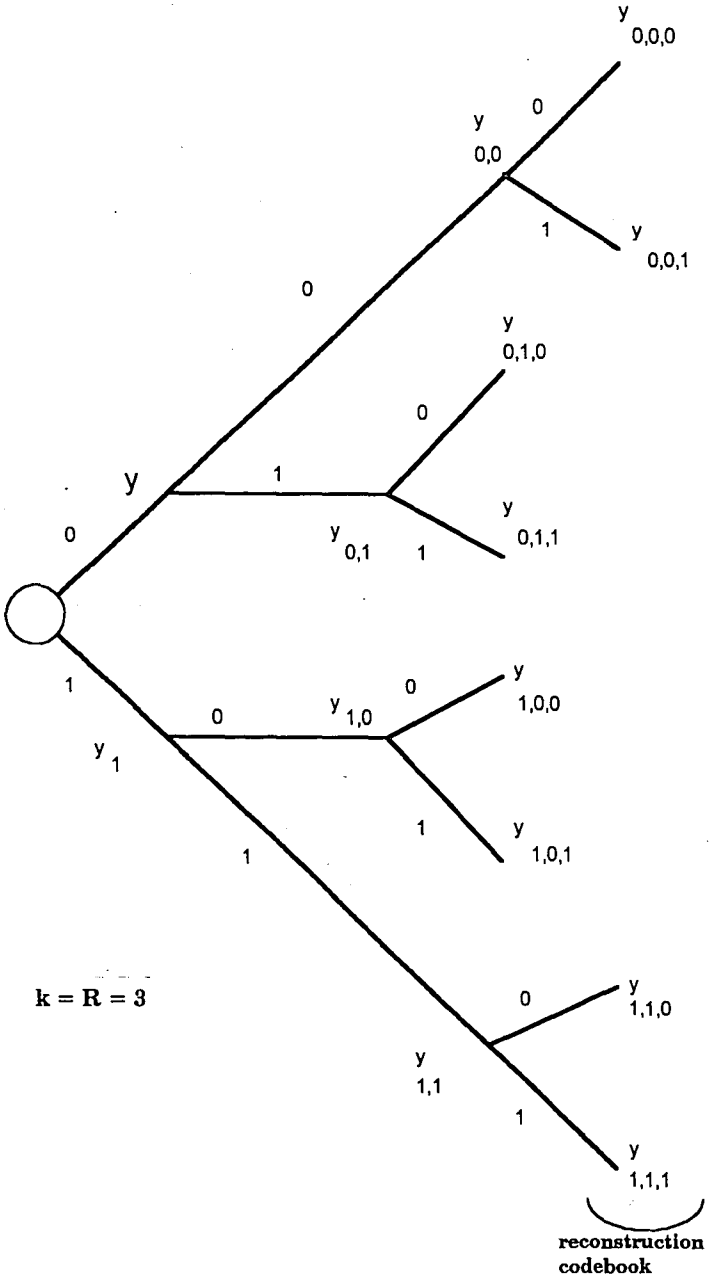

Figure 2.3: Tree-searched VQ.

The search, however, is much more efficient if done sequentially than is a full search. Thus, one may trade performance for efficiency of implementation.

Nonbinary trees can also be used where at the i^{th} layer codebooks of rate R_i are used and the overall rate is then $\Sigma_i R_i$. Other techniques can be used to design tree-searched codes. For example, Adoul *et al.* [3] use a separating hyper-plane approach. Another approach is to begin with a full search codebook and to design a tree-search into the codebook. One technique for accomplishing this is to first group the codewords into close disjoint pairs and then from the centroid of the pairs as the node label of the immediate ancestor of the pair. One then works backwards through the tree, always grouping close pairs. Ideally. one would like a general design technique for obtaining a tree search into an arbitrary VQ codebook with only a small loss of average distortion [2].

The Figure 2.3, can be explained as follows: A binary encoder tree is shown for a three-dimensional one bit per sample VQ. The encoder makes a succession of R minimum distortion choice from binary codebooks, where the available codebook at each level consists of labels of nodes in the next level. The labels of the nodes of the final layer are the actual reproduction codewords. At each node the encoder chooses the minimum distortion available label and, if the new index is a 0 (1), sends a channel symbol of 0 (1) and advances up (down) to the next node. After R binary selections the complete channel codeword has been sent and the reproduction codeword specified to the decoder [2].

Multistep VQ

A multistep VQ is a tree-searched VQ where only a single small codebook is stored for each layer of the tree instead of a different codebook for each node of each layer.

Such codes provide the computation reduction of tree-searched codes while reducing the storage requirements below that of even ordinary VQ [2].

The first example of such a code was the multistage codebook. For the simplicity we again confine interest to codes which make a sequence of binary decisions. The first layer code is designed as in the tree-searched case. This codebook is used to encode the training sequence and then a training sequence of error or residual vectors is formed. For waveform coding applications the error vectors are simply the difference of the input vectors and their codewords.

The algorithm is then run to design a binary VQ for this vector training sequence of coding errors. The reconstruction for these two bits is then formed by combining the two codewords. For waveform coding this is accomplished by adding the first codeword to the error codeword. This reproduction can then be used to form a "finer" error vector and a code designed for it. Thus an input vector is encoded in stages as with the tree-searched code, but now only R binary codebooks and hence $2R$ total codewords need to be stored. Observed that there are still 2^R possible final codewords, but we have not needed this much storage because the code can be reconstructed by adding different combinations of a smaller set of words. A multistage VQ is depecated in the figure 2.4.

In figure 2.4, the input vector is first encoded by one VQ and an error vector is formed. The second VQ then encodes the error vector. The two channel symbols from the two VQ's together form the complete channel symbols for the entire encoder. The decoder adds together the corresponding reproduction vectors.

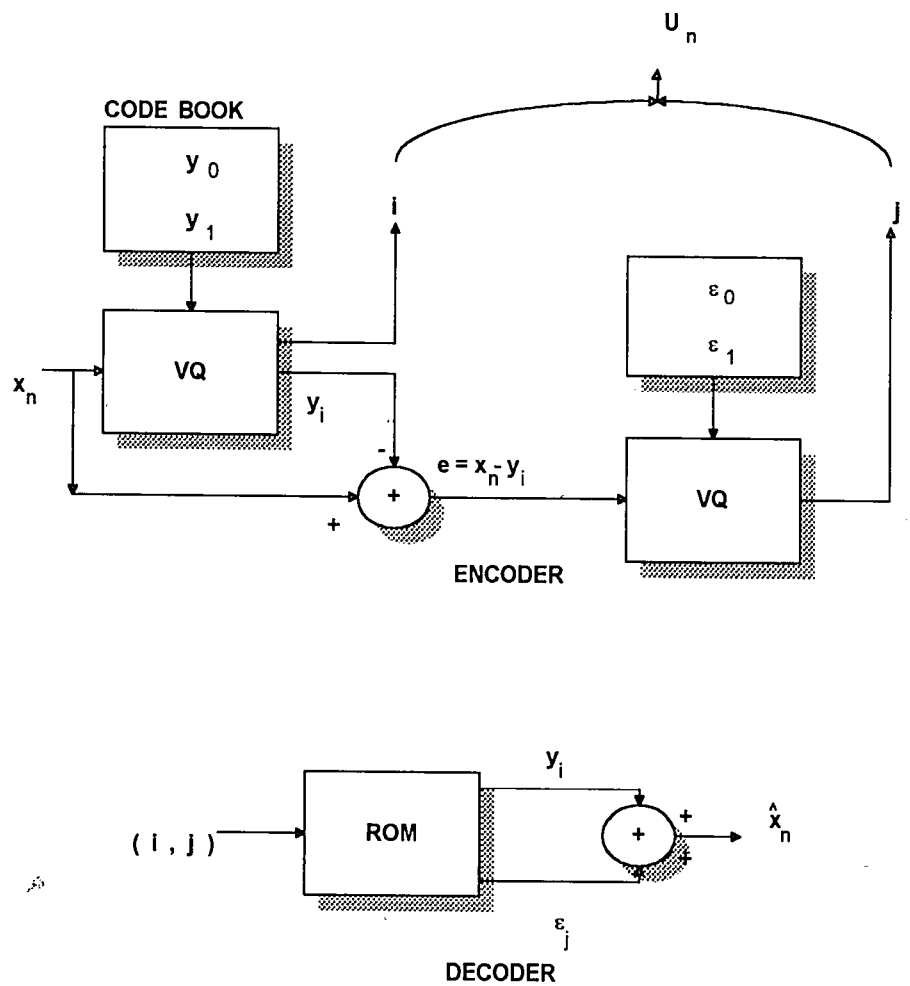

Figure 2.4: Multistage VQ with 2 Stages

Gain/shape VQ

In this type, different VQ's are used to code the "gain" and "shape" of a waveform. "Shape" is defined as the input vector normalized by a "gain" term. Another possibility-especially for images where the sample mean of pixel intensities in a small region is slowly varying-is to scalary quantize the sample mean of the vector, then subtract the coded sample mean from all samples in the vector, and finally VQ the vector.

A variation of this scheme is to convert each vector to another vector with zero mean and unit standard deviation, then they can be Memoryless VQ with LBG algorithm. The mean and standard deviation can be scalar quantized and sent with the vector codes. Also, a VQ can be used for that later statistical information.

Separating mean VQ

This is another example of product code, where a sample mean instead of an energy term is removed [7]. To understand the basic idea behind this type of VQ, consider this: define the sample mean $\langle \mathbf{x} \rangle$ of a k-dimensional vector by. $k^{-1}\sum_{i=0}^{k-1} x_i$ In a separated mean VQ one first uses a scalar quantizer to code the sample mean of a vector, then the coded sample mean is subtracted from all the components of the input vector to form a new vector with

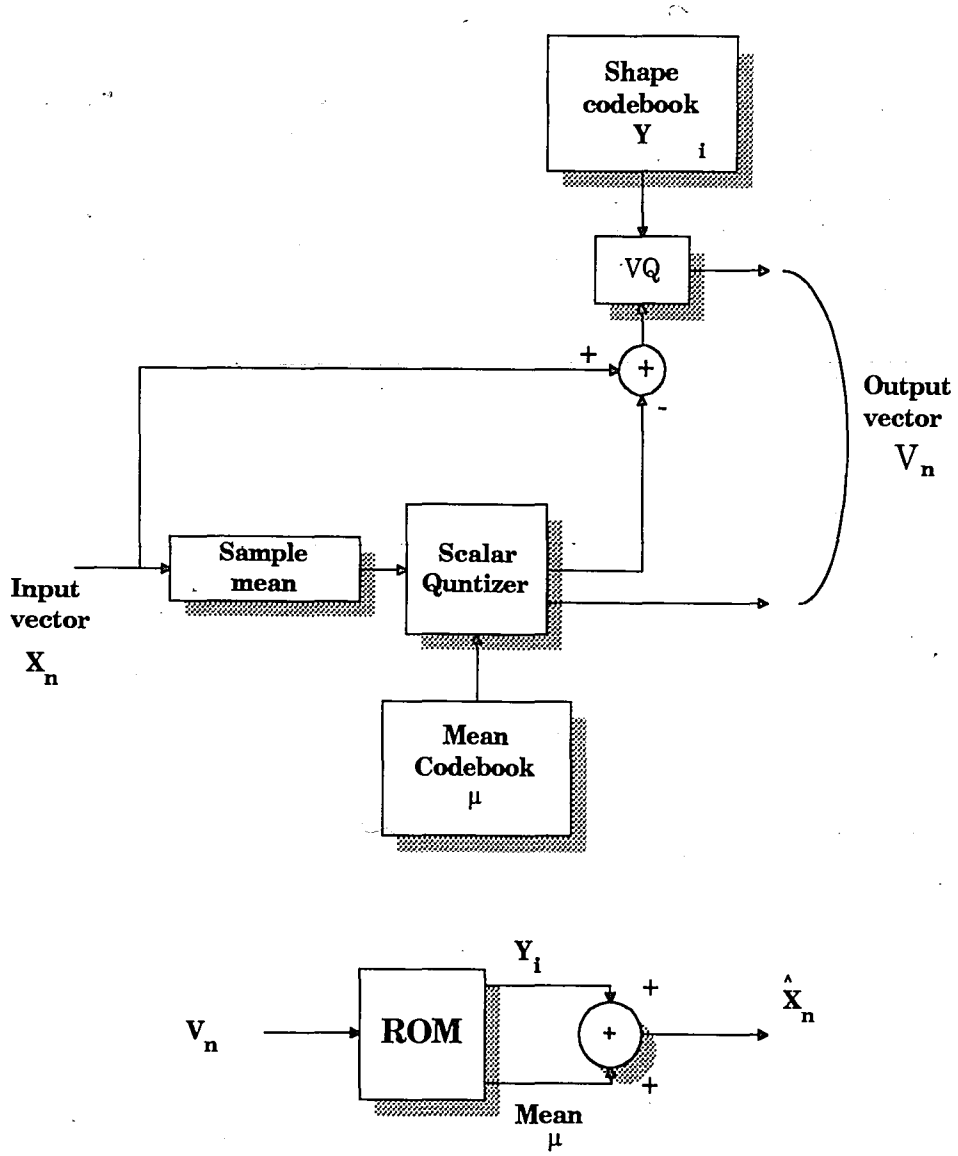

Figure 2.5: Separating Mean VQ.

approximately zero sample mean. This new vector is then quantized. Such a system is depicted in Figure 2.5. The basic motivation here is that in image coding the sample mean of pixel intensities in small rectangular block represents a relatively slowly varying average background value of pixel intensity around which there is a variation [2].

The figure explained how the separating mean VQ is working: The sample mean of the input vector is computed, scalar quantized, and the subtracted from each component of the input vector. The resulting vector with approximately zero sample mean is then vector quantized. The decoder adds the coded sample mean to all components of the shape vector.

Lattice VQ

This type of VQ can be defined as a *k*-dimensional generalization of the scalar uniform quantizer. A lattice quantizer is a quantizer whose codewords form a subset of a lattice. A lattice in *k*-dimensional space is a collection of all vectors of the form $y = \sum_{i=0}^{n-1} a_i e_i$, where n is less than or equal to k, where $\mathbf{e}_0,\ldots,\mathbf{e}_{n-1}$ are a set of linearly independent vectors in $\boldsymbol{R}^k$, and where the a_i are arbitrary integers [2].

2.4 Memory Vector Quantization

An improvement to the memoryless VQ can be incorporated by adding a memory. It' s important to note that information theory implies that the memoryless VQ can preform arbitrarily close to the optimal data compression.

Therefore, VQ's with memory can preform no better than memoryless VQ, but the complexity (space and time) of a memoryless VQ using larger vectors

can preclude it's use for a given distortion specification, where a less complex VQ using vectors with memory can meet the specified distortion level [4].

In this section three types of the memory VQ will be introduced: Feedback VQ, Finite-State VQ, and Vector Predictive Quantization.

Feedback Vector Quantizers

The simplest way to do this , is by choosing a different codebook for each vector, depending on the previous vectors. The decoder need to know which codebook to use for each vector it has to decode. This can be accomplished in several ways. If the encoder chooses the codebook based on its previous outputs, the decoder can track the same selections. This is called feedback vector quantization, while if the codebook selection is transmitted explicitly, then it is called adaptive vector quantization.

Feedback vector quantization can be considered as a generalization of the scalar adaptive quantizer with backward estimation.

The adaptive vector quantization can be considered as a vector generalization of the scalar quantizer with forward estimation.

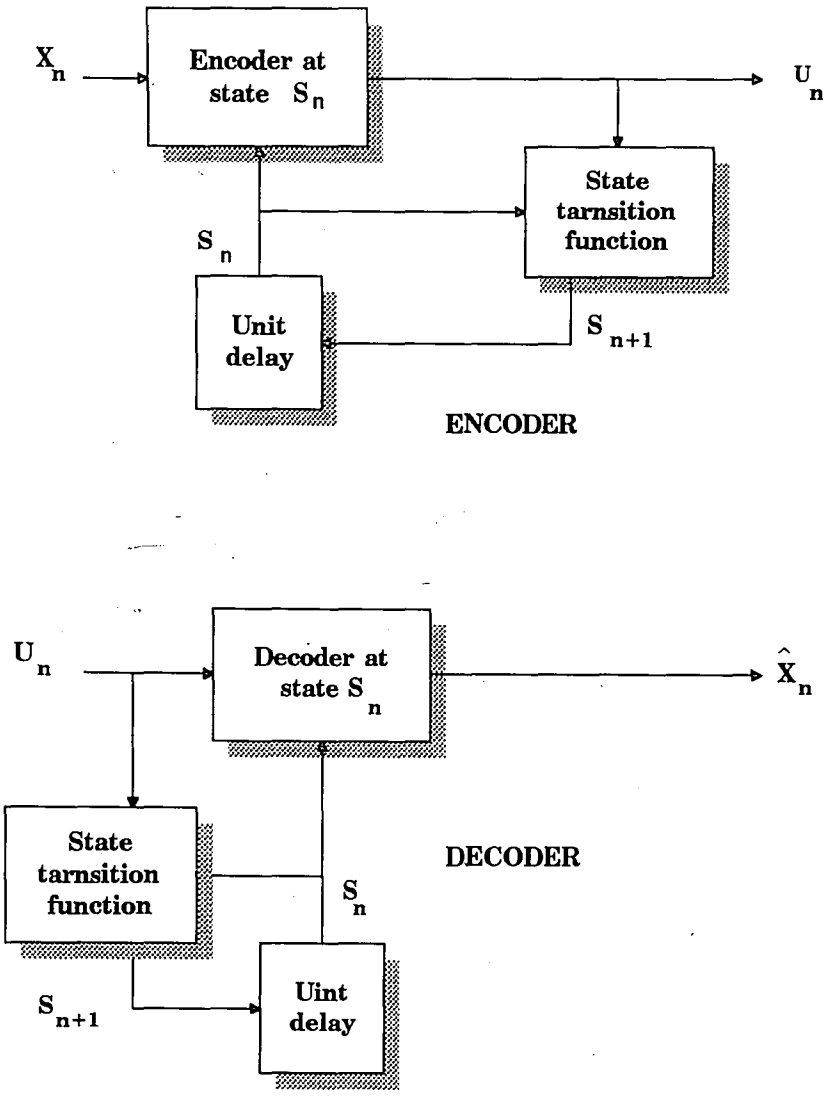

Figure 2.6: Feedback VQ.

The feedback VQ can be described as follows: The encoder and the decoder can be considered to have a state. For each distinct state there can be a different codebook associated with that state. Both the encoder and the decoder start from the same state. When the encoder outputs the first code, it uses it to determine the next state. When the decoder receives the code it correctly decodes it since it is in the same state that the encoder was to encode it, and it uses the received code to transition to the next state. As long as the

decoder can track the encoder states, the decoding is correctly preformed. But the big drawback of this approach is that if an error occurs and the decoder loses track of the state of the encoder, the results from the decoder are invalid from that point and on. A solution is to have a periodic reset on the states, or error control [4].

It has not been stated that the number of states has to be finite, but in the cases where it is, the system is called a Finite-State VQ, and it is suitable for VLSI implementation.

Finite-State Vector Quantization (FSVQ)

A general procedure for designing a FSVQ can be outlined as follows [14]:

1. Design an initial set of state codebooks and an arbitrary next-state function.
2. Given the next state function, use a variation of the LBG algorithm to attempt to improve the state codebooks.

The variation of the LBG algorithm is a slight extension to include the states of the system. When replacing all the reproduction vectors by the centroids of the vectors that mapped into them, take into account the state of system, that is, calculate all the centroids for each state separately and substitute them for the reproduction vectors separately for each state.

The procurement of the initial set of codebooks is a difficult matter. A possible approach is to use the LBG algorithm to design an initial codebook.

Then partition the training sequence by grouping all the training vectors that follows the ones that map together. For each group the LBG algorithm is now used to design it's codebook. A more detailed description can be found in [14].

For image coding several approaches to designing a FSVQ have been proposed, one mentioned in [19] uses a state transition function based on a classifier using intensity and geometric correlations between neighboring blocks [14].

Vector Predictive Quantization

The vector generalization of DPCM is the vector predictive quantizer (VPQ). A predicator is used to forecast the next vector, based on previous vectors, and only the error in this predication needs to be transmitted. The predicator can be simply the last vector, or an average of the last two vectors. The LBG algorithm can be used to design a VQ for prediction error sequence. For image coding a possible approach is making use of a classifier to separate the blocks into categories and use a separate predictor and error codebook for each category [4].

2.5 Image Coding using Vector Quantization

A fundamental goal of data compression is to reduce the bit rate for transmission or data storage while maintain an acceptable fidelity or image quality. Numerous bandwidth compression techniques have been devolved , such as DPCM, Transform Coding, hybrid coding, and adaptive versions of these techniques in response to the growth of image-processing methods. These techniques usually exploit the psychovisual as well as statistical redundancies in the image data to reduce the bit rate [5].

One deficiency with all of the conventional coding techniques is that quantization is preformed on individual real - valued samples of waveforms or pixels of images. These techniques are not optimal since the processed samples are still somehow correlated or dependent. Therefore, and according to Shannon's rate-distortion theory, a better performance is always achievable in theory by coding vectors instead of scalars.

The rest of this chapter will be devoted for the applications of VQ to images in the spatial domain, the predicative domain, the transform domain, and combinations of these known as hybrid domains.

2.5.1 Spatial Vector Quantization (SVQ)

In this domain adaptive and nonadaptive vector quantizers have been designed to code single fame images. The basic concept involves partitioning an image to two-dimensional vectors. Each vector is then compared to a set of standard vector templates stored in a ROM, and a codeword identifying the best match is then transmitted. The receiver reconstructs the image using the corresponding templates in place of the original vectors [19].

Classified VQ

At very low bit rates, the edges of the image can be severely distorted, because there aren't enough vectors to reproduce all possible orientations of an image. If the vectors are preclassified by their orientations and then vector quantized with an appropriate codebook, an improvement is achieved , in spite of the overhead incurred to indicate the vector class.

This approach can be extended to using a larger number of classes. Appropriate codebooks for each class can be constructed with the LBG algorithm. An example of the classes can be : horizontal edges, vertical edges, 45° edges, no edge, significant gradient but on edge, shades, textures, etc. An improvement is to use different coding depending on the character of the block ; Gain/Shape VQ for edge blocks, VQ on the DCT (Disceret Cosine Transform) coefficients for blocks with moderate gradient, etc.

One aspect that has not been considered until now is the fact that a fixed block size is not necessarily optimum throughout the picture, the classification can further subdivide complex blocks (containing several edges) and leave some big blocks (shade or textures) undivided. For a better performance , a split-merge segmentation technique can be used to define the subblocks [4].

Code Replenishment VQ

A simple approach to avoid having a fixed codebook for the whole image is to subdivide the image in blocks of several vectors, a codebook is created for each subimage, each created codebook is transmitted along with the codewords for the vectors in that subblock. But this approach suffers from several problems: redundant vectors among subimages are multiply transmitted, there is too much side information to transmit (all codebooks), the computation requirements are too great (create all new codebooks).

Another approach to replenishing the codebooks is not to change the whole codebook each time, instead only change a part of it. A measure of the distortion for each transmitted vector is taken, and if the distortion is too great, instead of transmitting the codeword, this vector is given another codeword and is added to the codeword.

If the coding is preformed on a series of images, then it is possible to

generate a codebook only for the first image and transmit it along with the codewords for this first image. For the following images, if a codeword for a vector does not change from the previous image, it is not necessary to transmit. Although the images in a sequence have a tendency to remain very similar, a codebook replenishment is necessary to handle the changing statistics of the image. But the image sequence has a dramatic change it is better to create a new codebook [4] .

Hierarchical VQ

An important point that has been ignored in adaptive SVQ is that the block size is constant throughout the encoding process. A better technique which could lower the bit rate significantly is to use variable block length. One such technique was introduced by Nasrabadi [42] called adaptive hierarchical VQ. In this coding technique, a quad-tree algorithm [19] is first used to partition the image into blocks of size 2 x 2, 4 x 4, 8 x 8, and 16 x 16. This information about partitioning the image is represented by a quad - tree and is counted as the side information to transmitted. In the coding process, the small blocks, 2 x 2 and 4 x 4, are coded by using a typical SVQ (shape vector quantizer). The larger blocks representing constant patterns are encoded by applying VQ in transform domain. Many of the high frequency coefficients can be discarded and thus the effective dimension of the blocks is reduced for computational purposes.

Intrafram VQ

In an image sequence, successive frames are usually very highly correlated. A simple intraframe coding system that only transmits the intensity of the pixels that change between successive frames is known as a frame replenishment coding system [19].

2.5.2 Predictive Vector Quantization

In the previous section, we considered zero memory vector quantization of scalar random variables . However, since the consecutive vectors are statistically dependent, better performance can be achieved if intervector correlation is incorporated in the encoder . The function of such technique is exactly the same as that of the feedback VQ and it's variation in FSVQ , outlined in in section 2.4.

2.5.4 Binary Vector Quantizer (BVQ)

Facsimile Coding Using BVQ

Digital images such as business letters and documents, weather maps , and engineering drawings, etc.,are nominally two-tone (black and white).

Efficient coding for the digital transmission or storage of two-tone images has been accomplished by a number of techniques such as run length coding and entropy coding of the original data. One such technique employs entropy coding is obtained by dividing the original images into blocks of N pixels, transmitting short codewords for more frequent vectors and long one for less ones. This can be implemented using Huffman coding scheme, but the design and implementation of that code will be very complicated since they required the evaluation of joint and conditional probabilities and a large lookup table for the storage of the codewords.

One possible way to increase the compression factor safely is to allow some image degradation in the coding process [19]. For a given vector size , only a small subset of all possible pixel patterns of the block is allowed. Any pattern which is not allowed is replaced by the pattern in the allowed subset that matches it most closely. This is known as a binary vector quantizer and codebook could be designed using the LBG algorithm.

Block Truncation Coding Using Binary Vector Quantization

Recently, a new technique called block truncation coding (BTC) has been developed [11], [16]. This technique uses a one-bit nonparametric quantizer, adaptive over local regions of the image. In general , the picture is divided into N x N blocks which are coded individually, each into a two-level signal. The

level, a and b , for each block are chosen such that the first two sample moments are preserved. These are given by :

$$a = \bar{X} - \bar{\sigma}\sqrt{\frac{q}{m-q}} \tag{2.6}$$

$$b = \bar{X} - \bar{\sigma}\sqrt{\frac{m-q}{q}} \tag{2.7}$$

where $\bar{x}$ and $\bar{\sigma}$ are the mean and the standard deviations of the block, respectively, q is the number of samples greater then $\bar{x}$, and $m = N^2$ is the number of samples in the block . Each block is then described by the values of $\bar{x}$, $\bar{\sigma}$, and an M x N bit plane consisting of 1's and 0's indicating whether the pixel has value above or below $\bar{x}$ [19].

2.5.5 Subband Vector Quantizers

Subband coding of images introduced by Woods and O'Neil [42] employed DPCM to encode each band. But VQ can be employed here too, where it exploits the redundancies between the spectral bands. The vectors would be taken as one

sample for each of the bands, that is , the dimension of the vectors would be the number of bands [4].

Chapter 3

Transform Coding

Disceret transform coding is a technique that takes blocks of pixels from the image and transforms them to another domain, which is different from the pixel intensity domain [4].

3.1 Definition of Transform Coding

In transform coding, a reversible, linear transform is used to map the image into a set of transform coefficients, which are then quantized and coded.

For most natural images, a significant number of coefficients had small magnitudes and can be coarsely quantized(or discarded entirely) with little image distortion. Figure 3.1 shows a typical transform coding system.

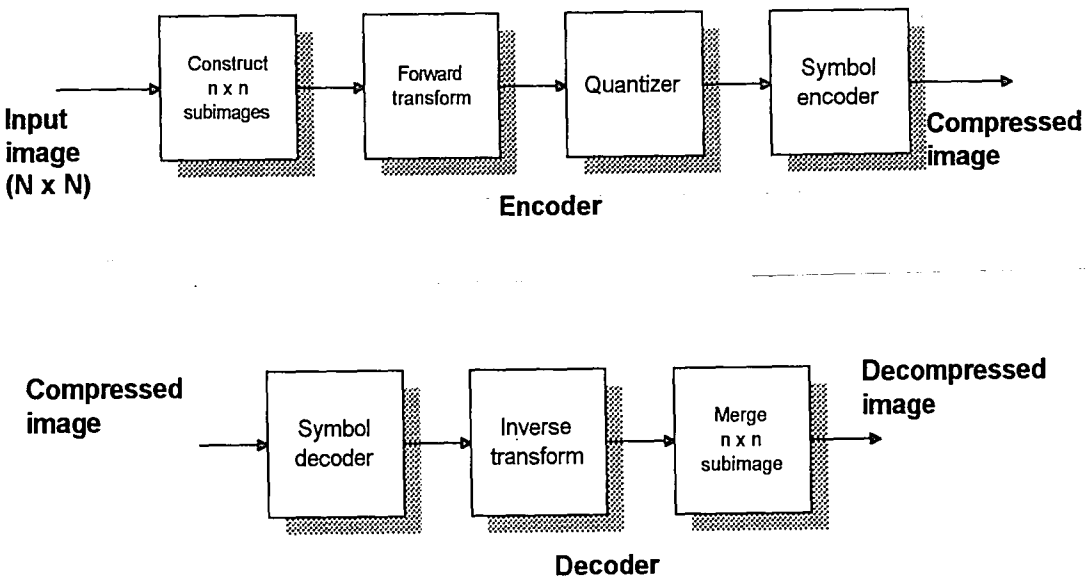

Figure 3.1 : A transform coding system

the decoder implements the inverse sequence of steps (with the exception of quantization function) of the encoder, which preforms four relatively straightforward operations : subimage decomposition, transformation, quantization, and coding. An N x N input image first is subdivided into subimages of size n x n. which are then transformed to generate $(N/n)^2$ n x n subimages transform arrays. The goal of the transformation process is to decorrelate the pixels of each subimage, or to pack as much information as possible into the smallest number of transform coefficients. The quantization stage then selectively eliminates or more coarsely quantizes the coefficients that carry the least information. These coefficients have the smallest impact on reconstructed subimage quality. The encoding process terminates by coding the quantized coefficients. Any or all of the transform encoding steps can be adapted to local image content, called *adaptive transform coding* , or fixed for all subimages, called *nonadaptive transform coding* [7].

The important characteristics that can make this approach useful for image coding are:

- *Correlation reduction property* : the transform decorrlates the coefficients in the given block.

- *Energy compaction property* : the energy is concentrated in fewer coefficients [21].

This allows the use of two mechanisms to achieve bit-rate reduction :

- Some of the transform coefficients can be discarded because they

contribute very little to the perceived image content.

- Some of the transform coefficients can be coarsely quantized and that does not degrade the picture quality dramatically.

Transform Coding techniques are generally expensive computationally, but for low bit/rate applications , they usually perform better than other simpler techniques, like waveform coding.

In a simple transform image coding system, a block of pixels from the original image $f(n_1, n_2)$ first undergoes the linear transform, which yield the transform coefficients $T_f(k_1, k_2)$, and it is then quantized to $\hat{T}_f(k_1, k_2)$ Before storage or transmission the quantized coefficients $\hat{T}_f(k_1, k_2)$ are assigned to binary codewords. This is repeated for all blocks that make up the image.

The decoding stage recovers the transform coefficients , $\hat{T}_f(k_1, k_2)$, from the codewords, and performs the inverse transform to obtain $\hat{f}(n_1, n_2)$, the recovered block. All the blocks together make the recovered image [4].

3.2 Image Coding Transforms

The choice of a particular transform in a given application depends on the amount of reconstruction error that can be tolerated and the computational resources available. Compression is achieved during the quantization of the

transformed coefficients (not during the transformation step).

For image coding the useful transforms are linear and can be expressed as,

$$T_f(k_1,k_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} f(n_1,n_2)\ a(n_1,n_2,k_1,k_2) \tag{3.1}$$

$$f(n_1,n_2) = \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} T_f(k_1,k_2)\ b(n_1,n_2,k_1,k_2) \tag{3.2}$$

where $a(n_1,n_2,k_1,k_2)$ are the orthonormal basis functions that define the transform, and $b(n_1,n_2,k_1,k_2)$ defines the inverse transform. It is possible to look at $f(n_1,n_2)$ as a linear combination of the basis functions $b(n_1,n_2,k_1,k_2)$, where $T_f(k_1,k_2)$ are amplitudes of these basis functions in the linear combination. Furthermore, it is possible to interpret the coefficients $T_f(k_1,k_2)$ as amplitudes of generalized spectral components when the basis functions have some form of sinusoidal behavior.

The general form presented above can be tremendously expensive computationally, therefore it is often simplified to the case when the transformation is separable. In the separable case, the transform can be expressed as,

$$T_f(k_1,k_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} f(n_1,n_2)\ a_R(n_1,k_1)\ a_C(n_2,k_2) \tag{3.3}$$

$$f(n_1,n_2) = \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} T_f(k_1,k_2)\ b_R(n_1,k_1)\ b_C(n_2,k_2) \tag{3.4}$$

where $a_R(n_1,k_1)$ and $b_R(n_1,k_1)$ are the row basis functions, $a_C(n_2,k_2)$ and $b_C(n_2,k_2)$ are the column basis functions. It is now possible to preform 1-D transforms on the columns, followed by another 1-D transform on the rows,

$$T_f(k_1,k_2) = \sum_{n_1=0}^{N_1-1} [\sum_{n_2=0}^{N_2-1} f(n_1,n_2)a_R(n_1,k_1)]a_C(n_2,k_2) \tag{3.5}$$

$$f(n_1,n_2) = \sum_{k_1=0}^{N_1-1} [\sum_{k_2=0}^{N_2-1} T_f(k_1,k_2)b_R(n_1,k_1)]b_C(n_2,k_2) \tag{3.6}$$

With separable basis functions, the computation can be reduced by orders of magnitude, compared to direct computation. For some of the popular transforms, additional computational savings are achieved due to the behavior of the basis functions.

An example of a separable discrete linear transform whose transform coefficients can be interpreted as spectral components is the Discrete Fourier Transform (DFT),

$$F(k_1,k_2) = \frac{1}{\sqrt{N_1 N_2}} \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} f(n_1,n_2)\, e^{-j(2\pi/N_1)k_1 n_1}\, e^{-j(2\pi/N_2)k_2 n_2} \quad (3.7)$$

$$f(n_1,n_2) = \frac{1}{\sqrt{N_1 N_2}} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} F(k_1,k_2)\, e^{-j(2\pi/N_1)k_1 n_1}\, e^{-j(2\pi/N_2)k_2 n_2} \quad (3.8)$$

An important characteristic of the transforms that have been used for image coding is the property that coefficients with small magnitude contribute only small amount of energy to the signal [21].

From the energy compaction point of view, the best transform is the Karhunen-Loève Transform (KLT) [32]. In the KLT the basis functions are real, orthonormalized eigenvectors of $K_f(n_1,n_2,l_1,l_2)$ [32], where

$$K_f(n_1,n_2,l_1,l_2) = E[(f(n_1,n_2) - E[f(n_1,n_2)])(f(l_1,l_2) - E[f(l_1,l_2)])] \quad (3.9)$$

The KLT transform is important more theoretically than practically, because of the difficulties involved. Th basis functions depend on the image characteristics, and must be calculated for each different image. In general there is no computationally efficient algorithm to calculate the transform coefficients.

For these and other difficulties the KLT is rarely used in image coding, and also because the DCT (Discrete Cosine Transform) is very close to the KTL performance in images which are highly correlated.

With images that exhibit "nearly stationary" statistics, it is found that their KLT basis function are very similar to the ones obtained for the KLT of highly correlated first order Markov processes [12]. And of the transforms

considered for image coding, the DCT also shows very similar basis functions. This seems to be part of the reason of the good performance observed for DCT [32] in image coding applications. Considering the performance and the computation cost, the DCT is considered the best choice [21].

3.3 Variations of Image Coding Transforms

3.3.1 Transform Vector Quantization (VTQ)

The purpose of transform coding is to convert statistically dependent or correlated picture elements into independent or uncorrected coefficients [6]-[29]. Because of the computational complexity an image is usually divided into subimages of reasonable size and then a one- or two-dimensional unitary transform is preformed on each subimage. The transformed coefficients are then nonuniformly quantized by a scalar quantizer. The quantization levels are given by a bit assignment matrix in which some of the high-frequency coefficients are discarded. The same bit assignment matrix is stored at the receiver or is transmitted for each image [16] .

Adaptive Transform Coding Using Vector Quantization

In adaptive transform coders several bit assignment matrices are

designed. Each subblock of the image is then classified into one of several classes according to the activity content of the block and coded by the appropriate bit assignment matrix. VQ in the transform domain in the above manner where each sub-block is considered as a vector has only one advantage, that is, the ac coefficients have a well-behaved Laplacian distribution which could be exploited in the design of the codebook.

Interframe Transform Vector Quantization (ITVQ)

When several frames are to be compressed, the complexity for three-dimensional coding is very large. One approach is to use two-dimensional transform coding in the two spatial directions, and use a vector quantizer in the temporal dimension. The dimension of the vector used in this temporal direction should be adaptive , since image sequences can have little change over some time, but they can have drastic changes at other times [4].

Other transforms that have been considered for image coding are:

Discrete Fourier Transform (DFT). The DFT has been mentioned above, and it have been used in some early image coding system [43]. But the DFT does not achieve an energy compaction as good as the DCT. The reason DFT has worse energy compaction then the DCT is due to the sharp discontinuity that occurs at the block boundary, which contributes energy to the high frequency

components [21]. The DCT does not present this artificial discontinuity [4].

Discrete Sine Transform (DST). The DST is defined by

$$S(k_1,k_2) = \frac{2}{\sqrt{(N_1+1)(N_2+1)}} \sum_{n_1=0}^{N_1} \sum_{n_2}^{N_2} f(n_1,n_2) \sin\left(\frac{k_1 n_1 \pi}{N_1+1}\right) \sin\left(\frac{k_2 n_2 \pi}{N_2+1}\right) \quad (3.10)$$

It has found use in the case of Recursive Block Coding [12], where the image information is separated into two components, and one component typically has it's KLT basis functions very similar to basis functions of DST [4].

Slant Transform. The slant transform is thought to better approximate the local behavior of the image, and therefore provide better energy compaction [32]. It is similar to the DCT and the performance is not much different in most useful cases. But the DCT is preferred, although there is an efficient algorithm to calculate the Slant Transform.

Walsh-Hadamard Transform (WHT). This transform can be constructed recursively, starting from $H_1 = 1$ then,

$$H_{2N} = \begin{bmatrix} H_N & H_N \\ H_N & -H_N \end{bmatrix} \quad (3.11)$$

Then the column vectors of the T matrix are the basis functions of the WHT, where

$$T = \frac{1}{\sqrt{N}} H_N \tag{3.12}$$

an example of N=4

$$T = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \tag{3.13}$$

In the WHT the concept of frequency is called *sequency* and is defined as the number of zero crossing divided by two. To have the coefficients ordered by decreasing energy content it is necessary to use sequancy ordered WHT [32]. The uses of only +1's and -1's in the WHT transform makes it very fast, but the energy compaction is not as good as the DFT or DCT [4].

Haar Transform. The Haar transform basis functions contain only +1's,-1's, and zeros. This allows very fast computation, since the operations are very simple, but the energy compaction performance is not very good [4].

Lapped Orthogonal Transform (LOT). This transform attempts to reduce the blocking effect that appears in the DCT when the bit rate is very low.

In this transform , the basis functions form adjacent blocks overlap, that is, for a 1-D block of N samples, N coefficients are calculated that map the block into a set of N basis functions. But these basis functions have more than N

samples each [41]. The LOT greatly reduces the block to block discontinuities compared to the DCT at low bit rates, and a fast algorithm (30 % slower than a similar DCT) has been implemented [25].

3.4 Discrete Cosine Transform (DCT)

Most practical transform coding systems are based on the DCT, which provides a good compromise between information packing ability and computational complexity. In fact, the properties of the DCT have proved to be of such practical value that it has become the international standard for transform coding system. Compared to other input independent transforms, it has the advantages of having been implemented in a single integrated circuit, packing the most information into the fewest coefficients[5] (for most natural images), and minimizing the blocklike appearance, called *blocking artifact*, that results when the boundaries between subimages become visible. This last property is particularly important in comparisons with the other sinusoidal transforms . Take for example,the DFT , the implicit n-point periodicity gives rise to boundary discontinuities that result in substantial high-frequency transform content. When the DFT transform coefficients are truncated or

[5] Ahmed et al. [1974] first noticed that KLT basis images (functions) of a first-order Markov image source closely resemble the DCT's basis images. As the correlation between adjacent pixels approaches one, the input dependent KLT basis images become identical to the input independent DCT basis images (Clark [1985]) [19].

quantized, Gibbs phenomenon[6] causes the boundary points to take on erroneous values, which appear in an image as blocking artifact. That is, the boundaries between adjacent subimages become visible because the boundary pixels of subimages assume the mean values of discontinuities formed at the boundary points. The DCT reduces this effect, because it's implicit $2n$-points periodicity dose not inherently produce boundary discontinuities.

An important point to mention here, is that the KLT, not the DCT, is the optimal transform in an information packing sense. That is, the KLT minimizes the mean -square error for any input image and any number of retained coefficients. However, because the KLT is data dependent, obtaining the KLT basis images for each subimage, in general, is a nontrivial computational task. For this reason, the KLT is seldom used in practice. Instead , a transform, such as the DCT, whose basis images are fixed (input independent), normally is selected.

The DCT is defined as,

$$C_f(k_1,k_2) = \frac{2}{\sqrt{N_1N_2}} \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} w_1(k_1)w_2(k_2)f(n_1,n_2)\cos(\frac{\pi}{2N_1}k_1(2n_1+1))\cos(\frac{\pi}{2N_2}k_2(2n_2+1)) \quad (3.14)$$

$$f(n_1,n_2) = \frac{1}{\sqrt{N_1N_2}} \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} w_1(k_1)w_2(k_2)C_f(k_1,k_2)\cos(\frac{\pi}{2N_1}k_1(2n_1+1))\cos(\frac{\pi}{2N_2}k_2(2n_2+1)) \quad (3.15)$$

where

---

[6] The phenomenon, occurs because the Fourier transform fails to converge uniformly at discontinuities. At discontinuities, Fourier expansions take the mean values.

$$w_1(k_1) = \begin{cases} \frac{1}{2}, & k_1 = 0 \\ 1, & \leq k_1 \leq N_1 - 1 \end{cases} \tag{3.16}$$

$$w_2(k_2) = \begin{cases} \frac{1}{2}, & k_2 = 0 \\ 1, & 1 \leq k_2 \leq N_2 - 1 \end{cases} \tag{3.17}$$

Subimage size selection

The usual approach is to divide the image into blocks. These blocks are then transformed independently. Since each block is processed independently, the coding can take advantage of this and be adaptive to the local image characteristics within each block. The blocking also reduces the intensive computations, and the memory requirements are also reduced, since only one block is needed in memory at a time. Another advantage of this is that to allows parallel implementations, where each processor operates on individual blocks [4].

Typical block sizes are 8 x 8 and 16 x 16 . The energy compaction improves as the block size is increased, but the average energy compaction does not improve much with block size above 8 x 8 [32].

Bit allocation

In most transform coding systems, the retained coefficients are selected on the basis of maximum variance, called ***zonal coding***, or on the basis of maximum magnitude, called ***threshold coding***, The over all process of truncating, quantizing, and coding the coefficients of a transformed subimage is commonly called ***bit allocation.*** Therefore, the available bits must be decided among all the blocks and coefficients as to provide the best image at that bit-rate. Typically in DCT the variance is much greater in low frequency coefficients, and they need more bits to be accurately described, while the high frequency coefficients can be coarsely quantized [4].

Zonal coding of coefficients

The energy compaction resulting from the transformation allows the discarding of many of the coefficients, which in part is responsible for the compression achieved. One approach for reducing the number of coefficients,is to consider zonal coding. Within the block of transformed coefficients, a zone with the important coefficients is selected, and all the other coefficients are discarded. Only the coefficients within the selected zone are coded, the discarded coefficients are assumed to be zero. The shape and the size of the zone is affected by many factors, including the transform used and the available number of bits in which to fit the code. The zones selected are those that have been shown to contribute most to the image perception, typically the low frequency

coefficients [4].

We can understand zonal coding based on information theory concept of viewing information as uncertainty. Therefore, the transform coefficients of maximum variance carry the most image information and should be retained in the coding process [19].

Threshold coding of coefficients

An adaptive method that is also used is threshold coding. It adapts to local block statistics. Now the coefficients resulting from the transform are compared to some defined threshold, and only those above the threshold are coded; the rest are discarded. Threshold has an advantage over zonal coding in that sometimes a coefficient outside of zone is large and important, and zonal coding would just discard it, just the same as if it were not important. Threshold coding would find it important and would code it. But in threshold coding the location of the coded coefficients is not known in advance, therefore the location of coefficients has to be coded too, which slightly increases overhead [4].

3.4.1 Distortions in DCT

DCT transform coding is a lossy technique, it is necessary to consider what form this loss takes.

The effect of quantizing the transform coefficients produces quantization noise. The character of quantization noise in transform coded images is different from the one in waveform coding. One characteristic is the loss in detail due to the high frequency components being small and getting discarded(by zonal or threshold coding); the images appear blurred compared to the original.

When the coefficients are quantized too coarsely, the image shows a noticeable graininess.

If within a block there is a smooth area, and a high detail busy area, the quantization error in the resulting high frequency coefficients(to account for the detail),will show as a distortion pattern on the smooth area [4].

3.5 JPEG standard

The JPEG standard is named after the Joint Photographic Experts Group, which developed it, joint refers to the collaboration between ISO and CCITT.

The standard addresses the needs for continuous-tone still -image applications such as desktop publishing, photovideotex,graphic art, or medical imaging.

The standard developed is really a family of image compression techniques rather than a single image compression algorithm. It provides several modes that allow *sequential encoding, progressive encoding, hierarchical*

encoding and lossless encoding.

Sequential encoding takes the image and encodes it in a single left-to-right, top-to-bottom scan.

Progressive encoding makes it possible for the decoder to show the image as it is being built up, adding detail as the decoding progresses.

In hierarchial encoding lower resolution versions of the image are accessible without needing to decompress the whole image. This is useful for browsing image collections, or for low resolution displays.

The lossless method is based on a DPCM scheme, followed by an entropy coder.

Figure 3.2 will illustrate the general idea of JPEG lossy compression.

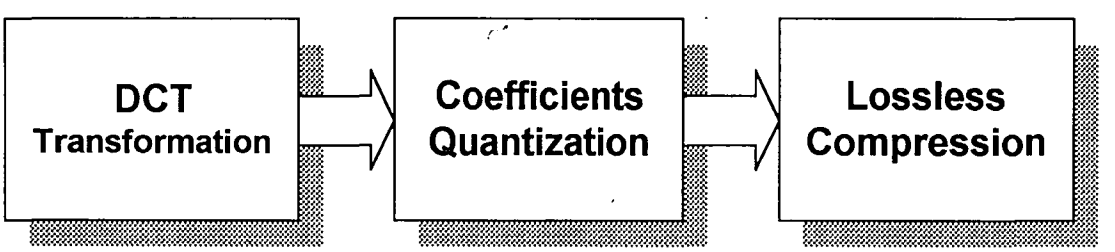

Figure 3.2 : JPEG lossy compression

Only the Baseline sequential coder-decoder (codec) will be discussed here, which has been the only JPEG codec implemented widely. The Baseline sequential algorithm specifies that the image has to be broken down into 8 x 8 blocks. A DCT on each block results in 64 coefficients, which are quantized through a quantization table. Finally, the quantized coefficients are run length and huffman, or arithmetic coded [4].

Preprocessing

Color images can be represented in different color systems where the image is made up of several components (RGB,CMYK,YUV), and grayscale images are considered as having a single component. Each color component is coded separately. For efficient compression it may desirable to convert the image to a different color system before encoding it. Gamma adjustments may also be performed here.

DCT transform

The different components of the image are then divided into 8 x 8 non-overlapping blocks. The sequence of 8 x 8 blocks goes then through a DCT transformation. The resulting 64 coefficients represent the spatial frequency contents of the block. The zero frequency component is called the DC coefficient, the rest are the AC coefficients.

Quantization

The coefficients are now uniformly quantized using a 64 elements quantization table. Quantization is defined as the division of each DCT coefficient by it's corresponding quantizer step size (in the quantization table). followed by rounding to the nearest integer. The quantization table can specify a different step size for each coefficient. This is the way the process discards information which is not visually significant.

Before the quantization step, no loss of information has taken place (assuming full precision in the operations). It is the quantization step that introduces the loss in the image. If the image has more than one component, each one can have its quantization table.

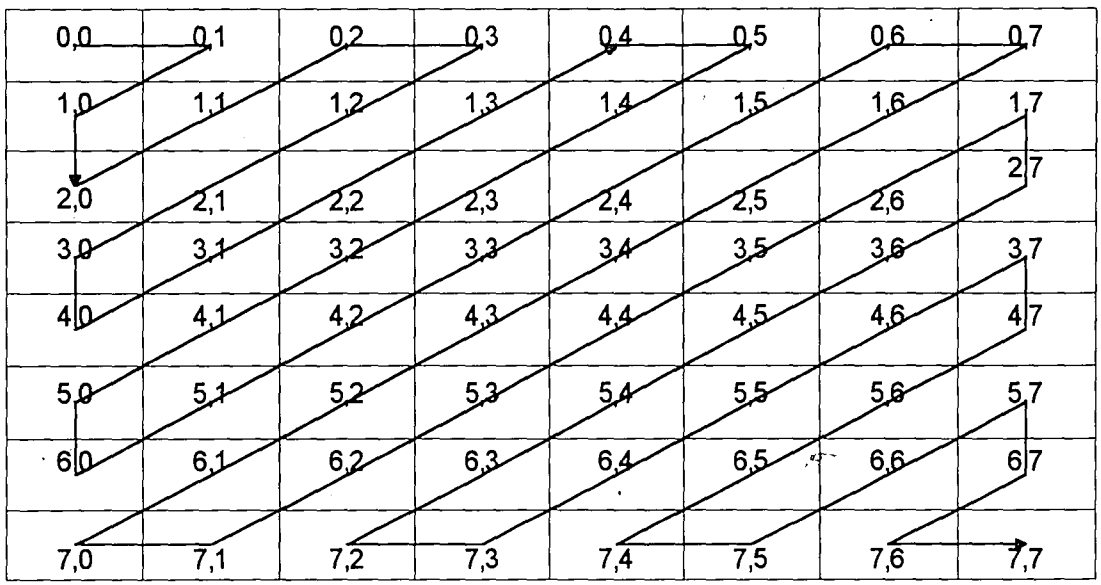

Figure 3.3 : The path of the zig-zag sequence

Coefficient reordering

After the coefficients have been quantized, the AC coefficients are reorganized as a linear array in ascending order of frequency. This is accomplished by scanning the AC coefficients in the block in a zigzag route starting from the lower frequency coefficients towards the highest frequency ones. Considering the 8 x 8 block of coefficients with the DC coefficient (0,0), and the highest frequency coefficient (7,7). Figure 3.3 illustrate the scanning process.

Coding of DC coefficients

The DC coefficients are treated differently. Because there is usually still a strong correlation between the average values (DC coefficients) of adjacent blocks, each DC coefficient is encoded as the difference from the DC coefficient of the previous block.

Entropy coding of AC coefficients

Finally, all quantized AC coefficients in the order specified and the DC coefficients differences are entropy coded to achieve further compression losslessly. Two alternatives can be used: Huffman or arithmetic encoding.

In the Baseline sequential codec each non-zero AC coefficient is represented by the a runlength of zero AC coefficients and its nonzero value. This sequence of runlength/nonzero-coefficient is then Huffman coded [4].

Decoding

Decompression is roughly the inverse from compression, but some additional steps may be taken to produce a better output image.

Briefly, these are the steps followed:

- Huffman or arithmetic decoding of the coefficient sequence.
- Quantization descaling and zigzag reordering of the elements in each 8 x 8 block.
- Assemble the image from the 8 x 8 blocks.
- Inverse DCT transformation of each 8 x 8 block.
- Interpolation of subsampled components (if any) to recreate the correct sized raster. At this point a pixel image of the original dimensions has been recreated.
- If color, space reconversion when needed (e.g., YCbCr to RGB). Gamma adjustment may also be performed here.

Chapter 4

Vector Transform Coding

Recently a new transform was introduced for image compression, the *Vector Transform.*

This transform operates on blocks of vectors, unlike the "scalar transforms" which operate on blocks of scalars.

To introduce the concept of the Vector Transform, it is useful to examine the analogy between some scalar and vector techniques.

PCM takes a sequence of discrete samples and quantizes them. If a sequence of vector is taken and jointly quantized, it is called memoryless VQ. Therefore, memoryless VQ can be seen as the vector generalization of PCM.

In DPCM a sequence of samples is taken, and only the error of the predicted value for the next sample, is coded. When compressing a sequence of

vectors, where the next vector is predicted from previous vectors, and the difference is coded, the technique is called Predictive VQ. Thus, Predictive VQ can be considered a vector generalization of DPCM.

In the transform coding, as discussed in the previous chapter, a sequence of samples is taken and transformed to another domain, where they quantized. It is possible now to see that Vector Transform is generalization of that to the vector case.

A sequence of vectors is transformed into another set of vectors of a different domain. Then this set of vectors can be vector quantized (VQ),resulting in a sequence of codewords.

To recover the image, the codewords are looked up in a table, and the resulting vectors are inverse-transformed into the recovered sequence of vectors that describes the image [20]. Figure 4.1 illustrate Image coding using the Vector Transform scheme [20].

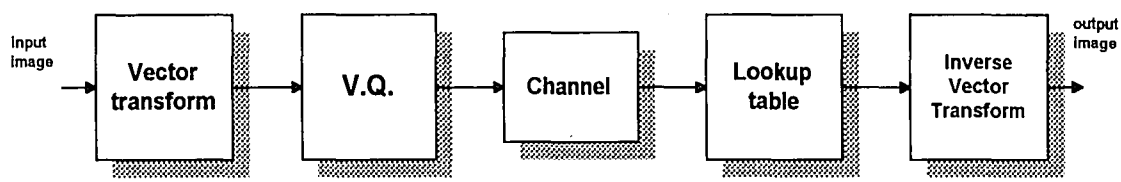

Figure 4.1 : Image Coding using the Vector Transform

4.1 Vector Transform

Consider a set of N vectors $\{ x_0, x_1, x_2, \ldots, x_{N-1} \}$ where each vector has N/2 elements, $x_n = [x_{0,n}, x_{1,n}, x_{2,n}, \ldots, x_{(N/2)-1,n}]^T$ with n=0,1,2,...,N-1. The vector transform is defined [20]:

$$X_k^T = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_n^T W^{nk} \tag{4.1}$$

$$x_n^T = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X_k^T W^{-nk} \tag{4.2}$$

where

$$W = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ -1 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix}_{\frac{N}{2} X \frac{N}{2}} \tag{4.3}$$

with this properties:

1.

$$W^{N/2} = -I \tag{4.4}$$

2.

$$W^N = I \tag{4.5}$$

3.

$$W^T = W^{-1} = W^{N-1} \tag{4.6}$$

4.

$$[x_0, x_1, x_2, \cdots, x_{(N/2)-2}, x_{(N/2)-1}]W = [-x_{(N/2)-1}, x_0, x_1, x_2, \cdots, x_{(N/2)-2}] \tag{4.7}$$

5. If N is a power of 2, i.e. $N = 2^t$, then

$$S = \sum_{k=0}^{N-1} W^{qk} = \begin{cases} NI & if\ q \bmod(N) = 0 \\ 0 & if\ q \bmod(N) \neq 0 \end{cases} \tag{4.8}$$

Properties 1 through 4 are easy to understand and a proof of property 5 can be found in [20].

With the properties mentioned above it can be shown that $\{x_n\}$ and $\{X_k\}$ form a transform pair. Recalling the definition of the vector transform,

$$X_k^T = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_n^T W^{nk} \tag{4.9}$$

If $\{\acute{x}_n\}$ is the inverse transform of $\{\mathbf{X_k}\}$ above, then it must be true that

$\acute{x}_n = x_n$ for n=0,1,2,...,N-1.

and using property 5

$$\acute{x}_k^T = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X_n^T W^{-nk} \tag{4.10}$$

$$\acute{x}_n^T = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \frac{1}{\sqrt{N}} \sum_{m=0}^{N-1} x_m^T W^{mk} W^{-nk} \tag{4.11}$$

$$\acute{x}_n^T = \frac{1}{N} \sum_{m=0}^{N-1} x_m^T \sum_{k=0}^{N-1} W^{(m-n)k} \tag{4.12}$$

$$S = \sum_{k=0}^{N-1} W^{(m-n)k} = \begin{cases} NI & if\ (m-n)\ \mathrm{mod}(N) = 0 \\ 0 & if\ \ (m-n)\ \mathrm{mod}(N) \neq 0 \end{cases} \tag{4.13}$$

only when m=n then **S = NI,** thus $\acute{x}_n^T = \frac{1}{N} x_m^T NI = x_n^T$ for n=0,1,2,...,N-1.

It is certain that (m-n) mod(N) = 0 only occurs when m = n because both m,n $\in$ {0,1,2,...,N-1} which implies -N < m-n <N [20].

4.2 Statistical properties of Vector Transform

With the Vector Transform now defined, it's properties can be discussed. Scalar transforms like the DCT has the energy packing property which allows compression by reducing the energy content on some coefficients and thus

making it possible to discard them, or coarsely quantize them. Also, the transform coefficients are less correlated which allows efficient coding even if performed separately.

The Vector Transform also shows similar properties. The set of transformed vectors are less correlated than the set of data vectors. The set of transform vectors have their energy concentrated in a few of the vectors, other vectors contribute very little to the total energy.

It has been shown [20] that for a simplified 2-D model of pixels correlations, 91% of the energy is concentrated in vectors $\mathbf{X}_0$, $\mathbf{X}_1$, and $\mathbf{X}_7$ for N=8. it is also shown that correlations between vectors in the transform domain is much less than that of correlations between the data vectors. The simplified 2-D model of pixel correlation is based on these assumptions:

- The correlation is separable into the product of horizontal and vertical correlations.
- Both —horizontal and vertical— processes as first order Markov process.
- The one-step correlation coefficient is ρ in both directions.

The vector set is taken from the image raster as N horizontally adjacent vectors. These vectors are N/2 pixels long and are taken as columns of the raster. For the results mentioned above $\rho = 0.91$.

4.3 VQ in the Vector Transform domain

As mentioned above, pixels from an image are grouped into vectors of length N/2, and sets of N vectors are then Vector Transformed into sets of N transform vectors. After the Vector Transform, the number of vectors is still the same as in the original image. But since now some of these vectors are much less important than others, it is possible to compress the image by Vector Quantizing these transform vectors with codebooks sized appropriately for their contribution to the energy content of the image.

Several algorithms have been considered for the codebook size allocation, and their design [20].

A simple extension of an algorithm proposed for bit allocation in "scalar" transform coding [20] results in this formula for the number of bits allocated to the k^{th} transform vector $\mathbf{X_k}$:

$$b_k = \frac{N}{2}R + \frac{1}{2}(\log_2 \prod_{i=0}^{N/2-1} \sigma_{i,k}^2 - \frac{1}{N}\sum_{k=0}^{N-1} \log_2 \prod_{i=0}^{N/2-1} \sigma_{i,k}^2) \qquad (4.14)$$

where $\sigma_{i,k}^2$ is the variance of the i^{th} element of the k^{th} transform vector. Then N codebooks can be created with the LBG algorithm [20] where the k^{th} codebook has 2^{b_k} entries.

Another approach is to allocate bit directly using the transform vectors. It is based on the relation between the VQ distortion and the entropy of the codebook [20]. The derived bit allocation formula is [20],

$$b_k = \frac{N}{2}R + \frac{N}{4}(\log_2 \sigma_k^2 - \frac{1}{N}\sum_{k=0}^{N-1} \log_2 \sigma_k^2) \tag{4.15}$$

where

$$\sigma_k^2 = \sum_{i=0}^{N/2-1} \sigma_{i,k}^2 \tag{4.16}$$

Again, once the codebook size is known, the LBG algorithm is used to create the codebooks.

But the LBG algorithm allows another approach. In the cases where it is difficult to obtain a relationship between distortion and the number of quantization levels in a close form, it is still possible to use the LBG algorithm.

The LBG algorithm can create a codebook C_k for the transform vector $\mathbf{X_k}$ given a distortion measure D_k for a specified number of bits b_k . The objective is minimize the average distortion,

$$D = \sum_{k=0}^{N-1} \frac{\sigma_k^2}{N} D_k \tag{4.17}$$

If the distortion measure D_k is complicated function of b_k , the minimization of D can be a very complex nonlinear problem [20]. But the number of codebooks is constant (N) and the total number of bits to allocate is

fixed

($\sum_{k=0}^{N-1} b_k = const$). This makes the search space finite; an exhaustive search is feasible.

Start with a bit allocation { b_k }, and create k codebooks, one for each transform vector. Compute the average distortion as defined above. Repeat this for all possible bi allocations { b_k }, and keep the one with the lowest average distortion D.

Finally, it is possible to incorporate the iteration for bit allocation into LBG algorithm. This modified LBG algorithm follows [20],

1. For k=0,1,2,...,N-1, set b_k=0, S_k=1, and C_k= { centroid of $\mathbf{X_k}$of the training data }.

2. Compute the weighted distortion $\frac{\sigma_k^2}{N} D_k$ for each k using a distortion measure of choice.

3. For each k, split the reproduction vector in C_k into two vectors and optimize C_k with S_k = 2(i.e., b_k=1).

4. Compute the weighted distortion $\frac{\sigma_k^2}{N} D_k$, again for each k.

5. Let D_{old} and D_{new} be the first and second sets of weighted distortions, respectively.

6. Let B_{old} and B_{new} be the first and the second sets of bits allocations,

respectively.

7. Let C_{old} and C_{new} be the first and the second sets of codebooks, respectively.

8. Take the difference of the two weighted distortions for each k and let it be Δ_k .

9. Identify the index $k = k_m$, which has the largest decrease in the weighted distortion, i.e., $\Delta_{km} = \max \{ \Delta_k \}$.

10. Let $b_{km} = b_{km} + 1$ in both B_{old} and B_{new} .

11. Replace the codebook C_{km} in C_{old} with that in C_{new} .

12. Replace the value of $\frac{\sigma_{km}^2}{N} D_{km}$ in D_{old} with the corresponding value in D_{new}.

13. Split each reproduction vector in C_{km} into two vectors, optimize C_{km} with double size, and replace C_{km} in C_{new} with the newly optimized C_{km} .

14. Compute the weighted distortion $\frac{\sigma_{km}^2}{N} D_{km}$ of the new codebook C_{km} and put it in D_{new} .

15. Take the difference of the two weighted distortions in D_{old} and D_{new} for the index k_m and replace the old value of Δ_{km} with this new value.

16. Repeat steps 9-15 until the total number of bits in B_{old} equals the predetermined value (N/2)R.

17. C_{old} now contains the codebooks to code the transform vectors.

This modified algorithm produces a set of codebooks of the appropriate sizes.

Recent research in this area has showed the existence of other Vector Transform, and a way to find them. The research for one with better energy packing properties is expected to yield a better Vector Transform than the one described here [20].

CHAPTER 5

Comparsion

In this chapter we present the results of testing three coding schemes under the influence of simulated channel noise with uniform distribution. These coding schemes are : VECTOR QUANTIZATION, VECTOR TRANSFORM CODING, and DISCRETE COSINE TRANSFORM.

5.1 Anatomy of the Test Process

As mentioned above we have tested these three schemes under the influence of channel noise with uniform distribution. The simulation of the channel noise was implemented by flipping one bit (randomly) every specific number

of bits , knowing as Bit Error Rate (BER).

The test consists of the different aspects , which can be explained as follows:

I) ***SCHEMES:***

1.VEGTOR QUANTIZATION with 0.25 and 0.50 bit per pixel (bpp).

2.VECTOR TRANSFORM CODING with 0.25 and 0.50 bit per pixel (bpp).

3.DISCRETE COSINE TRANSFORM with 0.50 bit per pixel (bpp).

II) ***CHANNEL NOISE:***

Different Bit Error Rates:

$$\left(\frac{1}{1000}, \frac{1}{2000}, \cdots, \frac{1}{10{,}000}, \frac{1}{50{,}000}, \frac{one\ bit}{whole\ compressed\ data} \right).$$

III) ***SIGNAL -TO-NOISE RATIO:***

Both types of S/N (rms and peak).

When calculating S/N we considered three different ways:

1.S/N for Original image *vs* Quantized image .

2.S/N for Original image *vs* Noisy image.

3.S/N for Quantized image *vs* Noisy image.

The following pages present the results of the test under above conditions.

Comparing VT,VQ abd DCT for Orignal vs Quantized images					
	rms SNR for Orignal vs Quantized				
VTbpp.50	31.4163				
VQbpp.50	25.9387				
VTbpp.25	21.9573				
DCTbpp.50	20.9353				
VQbpp.25	20.7503				
	peak SNR for Orignal vs Quantized				
VTbpp.50	35.9821				
VQbpp.50	29.3577				
VTbpp.25	27.6348				
DCTbpp.50	26.6251				
VQbpp.25	22.8071				

Table 5.1: Comparing Original image vs Quantized image.

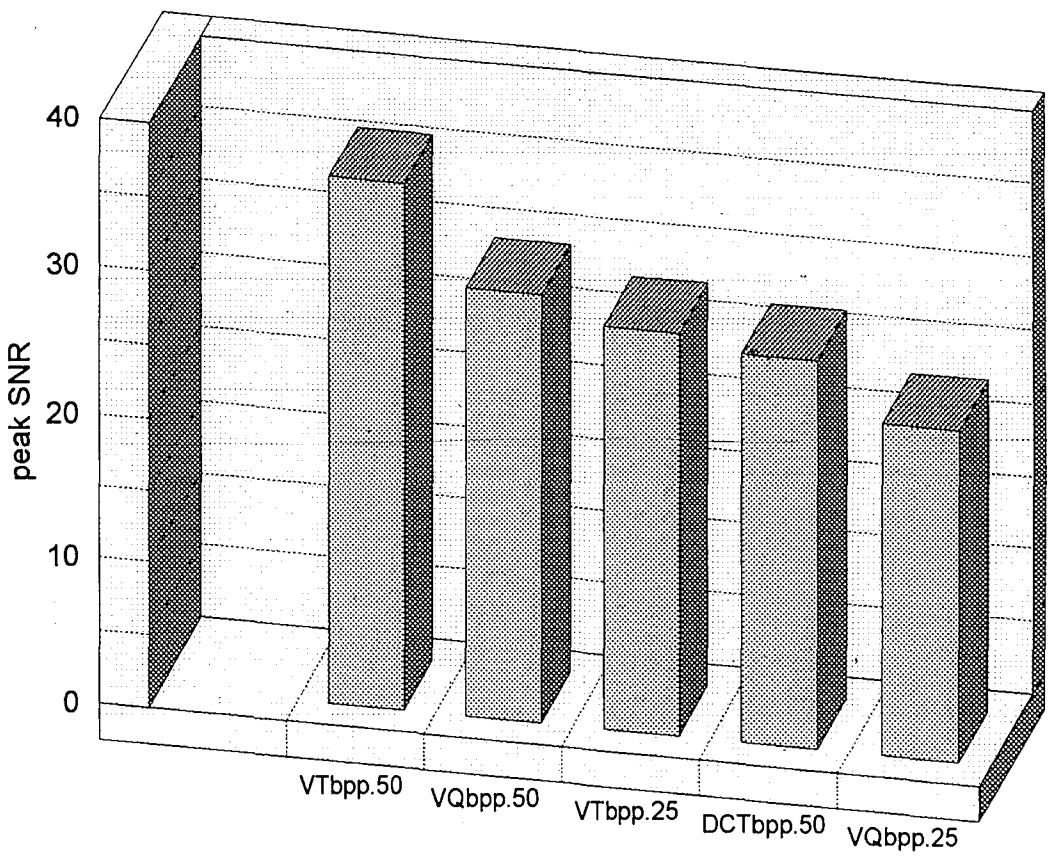

Figure 5.1 : Comparing Ordinal image vs Quantized image (peak SNR)

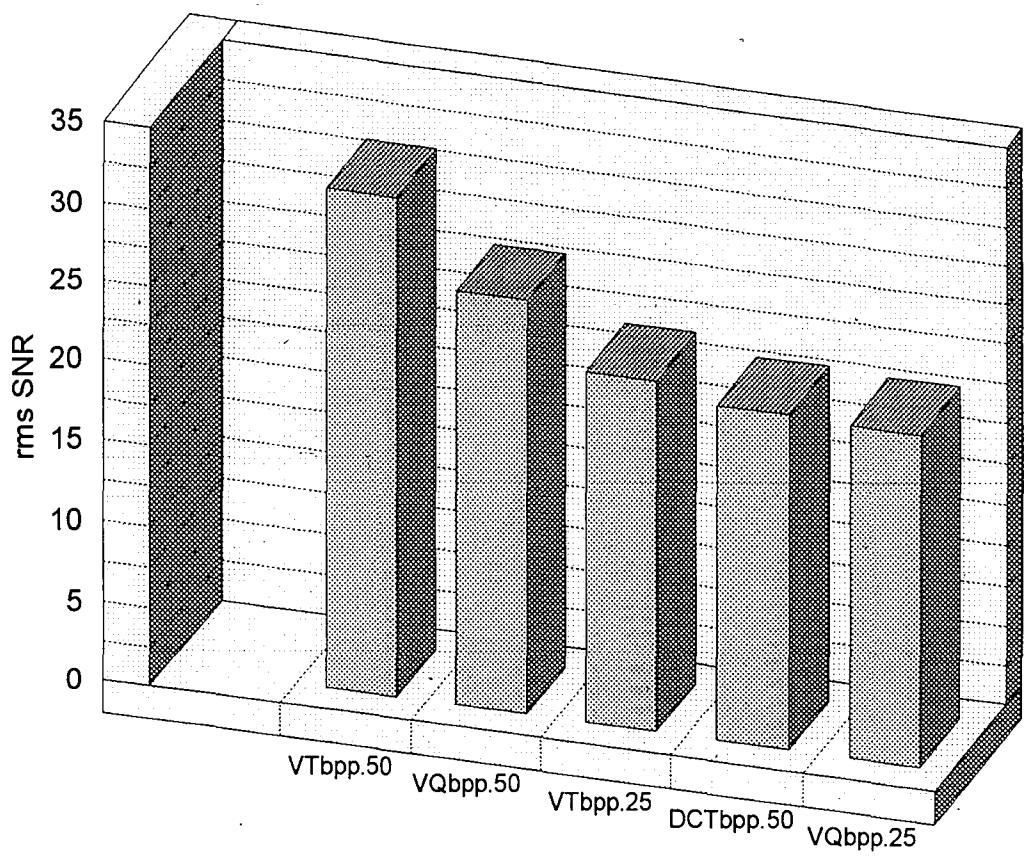

Figure 5.2 : Comparing Original image vs Quantized image (rms SNR)

Comparing VT,VQ abd DCT for Orignal vs Noisy images					
		rms SNR for Orignal vs Noisy			
	VTbpp.25	VTbbp.50	VQbpp.25	VQbpp.50	DCTbpp.50
BER=1/1000	21.3298	28.9548	20.4247	24.7967	19.3539
BER=1/2000	21.794	26.6001	20.6203	25.3373	19.9533
BER=1/3000	21.7729	29.6983	20.6695	25.4407	20.3618
BER=1/4000	21.812	30.987	20.686	25.6963	20.5143
BER=1/5000	21.8865	26.5357	20.7046	25.6936	20.1011
BER=1/6000	21.9172	30.2153	20.6988	25.7511	20.6421
BER=1/7000	21.8758	29.4352	20.7215	25.7578	20.7648
BER=1/8000	21.9403	31.3417	20.7175	25.6358	20.6142
BER=1/9000	21.8563	30.5361	20.7137	25.7241	20.8854
BER=1/10,000	21.9157	31.0439	20.7336	25.7234	20.8431
BER=1/50,000	21.9559	31.4113	20.7463	25.9364	20.9309
one bit only	21.9566	31.4144	20.7463	25.9385	20.9335
no noise	21.9573	31.4163	20.7503	25.9387	20.9353
		peak SNR for Orignal vs Noisy			
	VTbpp.25	VTbbp.50	VQbpp.25	VQbpp.50	DCTbpp.50
BER=1/1000	27.0042	33.7455	22.4823	28.2154	25.0483
BER=1/2000	27.4707	31.8831	22.6777	28.7553	25.6392
BER=1/3000	27.4495	34.6413	22.7263	28.8588	26.0487
BER=1/4000	27.4887	35.5516	22.7428	29.1155	26.238
BER=1/5000	27.5636	31.4208	22.7609	29.1129	25.7832
BER=1/6000	27.5945	34.7807	22.7566	29.1691	26.3307
BER=1/7000	27.5535	34.2763	22.778	29.1766	26.4538
BER=1/8000	27.6177	35.9075	22.7744	29.0548	26.3045
BER=1/9000	27.5332	35.9116	22.708	29.1432	26.5772
BER=1/10,000	27.5929	35.7244	22.79	29.1424	26.5324
BER=1/50,000	27.6334	35.9771	22.803	29.3555	26.621
one bit only	27.6341	35.9803	22.803	29.3574	26.6235
no noise	27.6348	35.9821	22.8071	29.3577	26.6251

Table 5.2 : Comparing Original image vs Noisy image

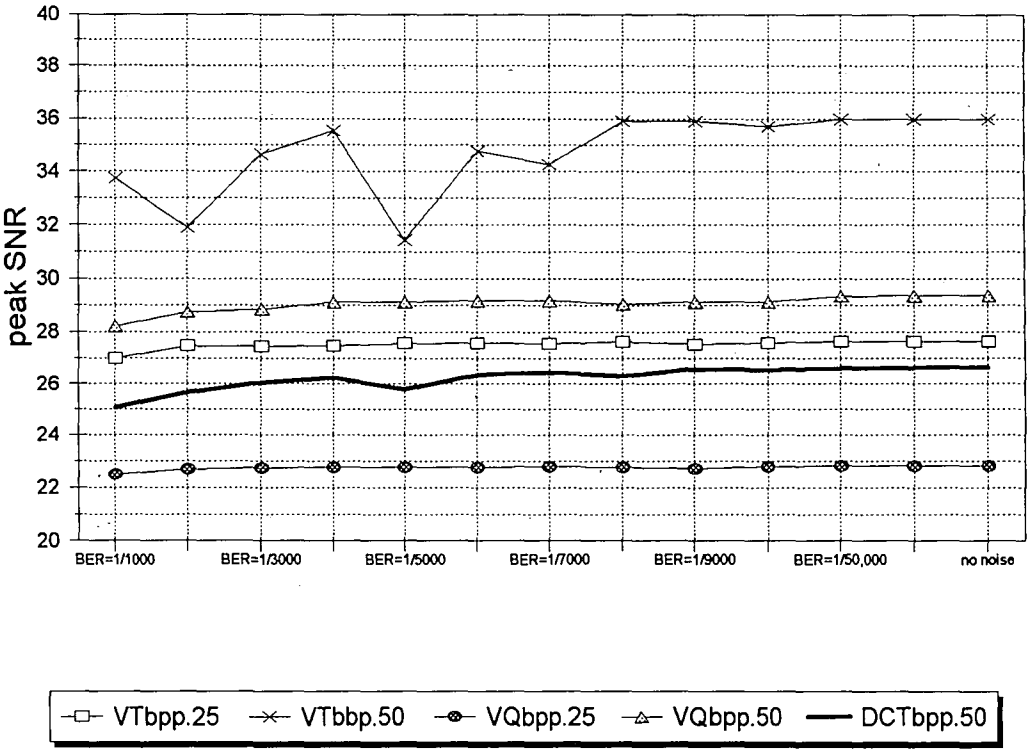

Figure 5.3 : Comparing Original image vs Noisy image (peak SNR)

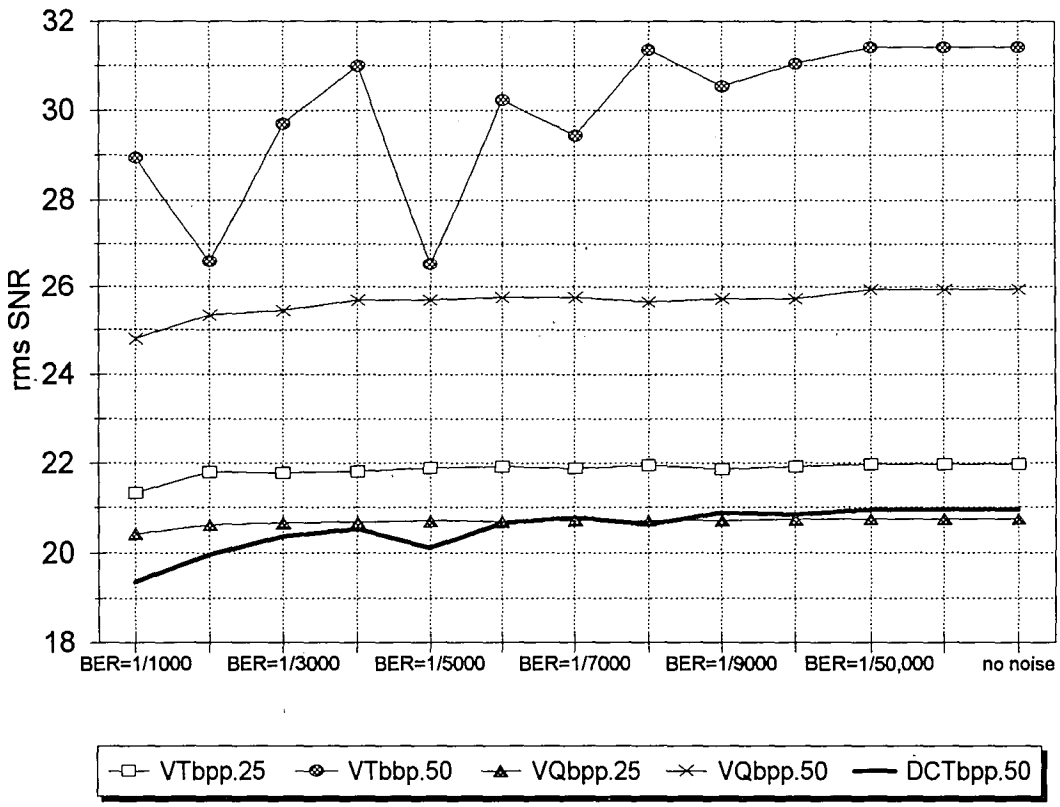

Figure 5.4 : Comparing Original image vs Noisy image (rms SNR)

Comparing VT,VQ abd DCT for Quantized vs Noisy images					
		rms SNR for Quantized vs Noisy			
	VTbpp.25	VTbbp.50	VQbpp.25	VQbpp.50	DCTbpp.50
BER=1/1000	29.9982	32.5948	31.88	31.1953	24.8707
BER=1/2000	35.9273	28.3292	36.2144	34.1313	27.1219
BER=1/3000	35.5081	34.5518	37.8998	35.2139	29.5835
BER=1/4000	36.3998	41.3386	39.0355	38.2099	31.0188
BER=1/5000	39.7184	28.2893	40.3013	38.2867	27.864
BER=1/6000	42.559	36.4798	39.8629	39.5542	32.6071
BER=1/7000	39.1793	33.739	42.477	39.452	34.9885
BER=1/8000	45.8926	48.8606	42.1422	37.5003	32.3657
BER=1/9000	38.0372	37.8885	41.4747	38.9761	41.1982
BER=1/10,000	41.7201	41.829	44.6811	38.9047	38.0965
BER=1/50,000	56.6771	60.0356	51.7384	59.3713	50.4616
one bit only	60.3324	64.8734	51.7384	70.9737	57.7024
		peak SNR for Quantized vs Noisy			
	VTbpp.25	VTbbp.50	VQbpp.25	VQbpp.50	DCTbpp.50
BER=1/1000	35.6726	37.3855	33.9377	34.6141	30.5651
BER=1/2000	41.6039	33.6123	38.2718	37.5493	32.8077
BER=1/3000	41.1847	39.4948	39.9566	38.632	35.2704
BER=1/4000	42.0765	45.9033	41.0923	41.629	36.7425
BER=1/5000	45.3956	33.1344	42.3575	41.706	33.5461
BER=1/6000	48.2364	41.0451	41.9207	42.9721	38.2957
BER=1/7000	44.857	38.5801	44.5335	42.8707	40.6775
BER=1/8000	51.57	53.4265	44.1991	40.9193	38.056
BER=1/9000	43.7141	43.2641	43.5318	42.3953	46.89
BER=1/10,000	47.3973	46.5096	46.7375	42.3237	43.7859
BER=1/50,000	62.3546	64.6015	53.7951	62.7904	56.1517
one bit only	66.0099	69.4393	53.7951	74.3927	63.3924

Table 5.3 : Comparing Quantized image vs Noisy image

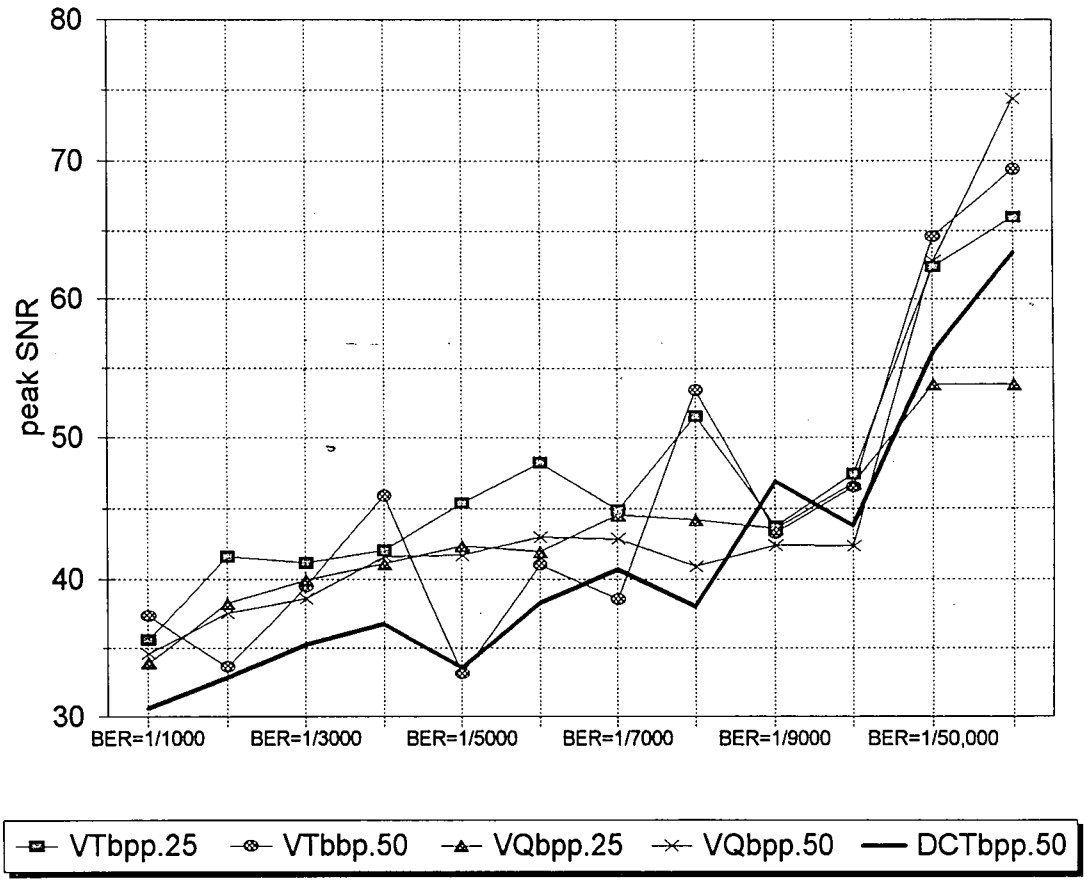

Figure 5.5 : Comparing Quantized image vs Noisy image

(peak SNR)

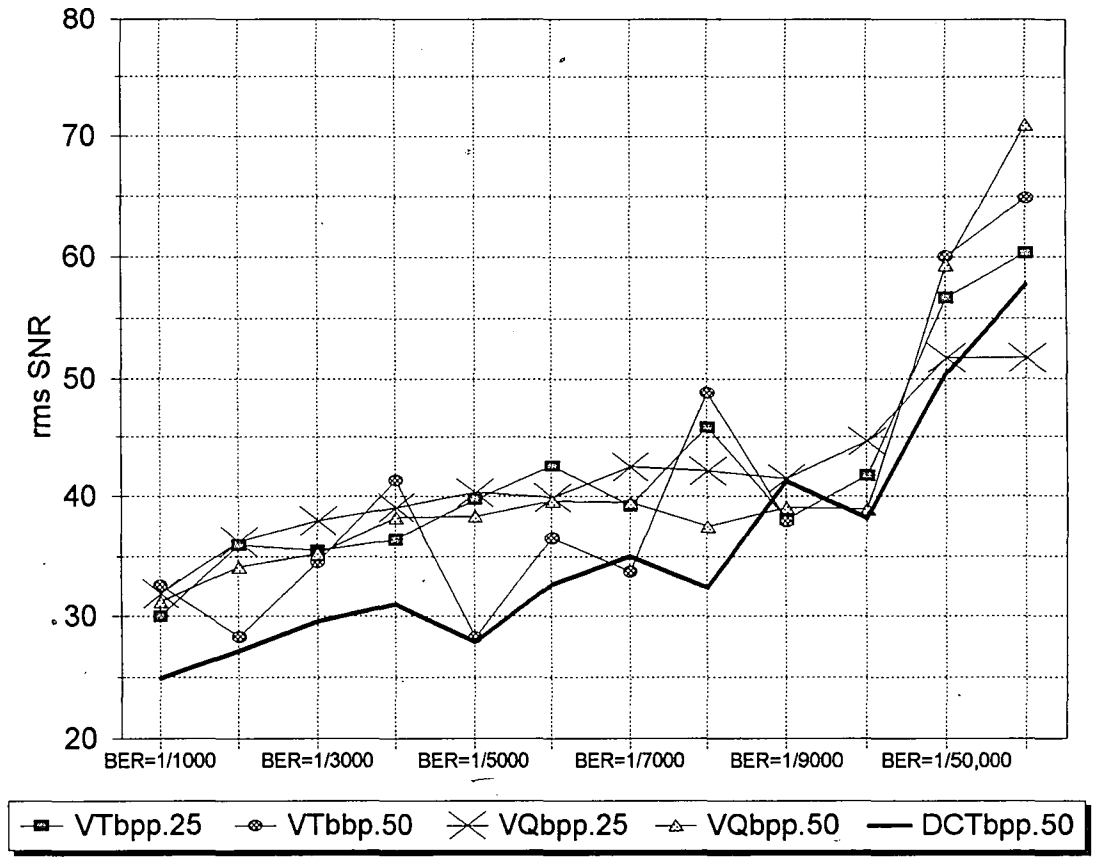

Figure 5.6 : Comparing Quantized image vs Noisy image

(rms SNR)

5.2 Conclusions

We can summarized the results of the above tests in the following table:

RANK	CODING SCHEMES
1 (best)	VT with (0.50 bpp)
2	VQ with (0.50 bpp)
3	VT with (0.25 bpp)
4	DCT with (0.50 bpp)
5 (worst)	VQ with (0.25 bpp)

Table 5.4 : Final conclusion of the comparison

According to the above table, the Vector Transform with (0.50 bpp) comes in the first place followed by the Vector Quantization with (0.50 bpp) in the second place. Vector Transform with (0.25 bpp) takes the third position, while Discrete Cosine Transform with (0.50 bpp) comes fourth. The worst performance under such channel noise was by Vector Quantization scheme with (0.25 bpp).

Important Points to mention :

- Channel Noise used here is one with uniform distribution, this open the doors for other types of channel noise to be investigated with their effects on the above coding schemes.
- The above schemes can be tested with different bit per pixel rates.
- Different BER (bit error rates) can be used to investigate the performance of these schemes in high BER cases .
- In the case of DCT scheme we introduced noise to the data part only, because any change in the headers or indices ,makes the decoding process impossible and we can not recover the data.

References

[1] Huseyin Abut.*Vector quantization*.IEEE Press,NewYork,NY.,1990.

[2] H. Abut,R.M. Gray, and G. Rebolledo.Vector quantization of speech and speech-like waveforms.*IEEE Transaction on Acoustics Speech and Signal processing* ASSP-30:423-435,June 1982.

[3] J.P. Adoul,J.L. Debray,and D.Dalle.Spectral DPCM coders with L predictors.*Proc. of the 1980 IEEE International Conference on Acoustics*

[4] Tirso Alonso.*Digital image compression.* Master thesis,Lehigh University,May 1992.

[5] G. B. Anderson and T.S. Huang. Piecewise Fourier transformation for picture bandwidth compression.*IEEE Trans.Commun.* COM-20: 388-491, June 1972.

[6] H. C. Anderws,J.Kane, and W. K. Pratt.Hadamard transform image coding.*Proc.IEEE.*57:58-68,Jan. 1989.

[7] R.L.Baker,and R.M.Gray.Differential vector quantization of achromatic imagery.*Proc.of International Picture Coding Symposium*.April 1984.

[8] R. L. Baker and R.M.Gray.Image compression using non-adaptive spatial vector quantization.Pro.*Conf.Rec. Sixteenth Asilmaor Conf.Circuits,Syst.Comput.*,page: 55-61,October 1982.

[9] A. Buzo,Gray,A.H.,jr., and R.M.Gray.and J.D.Markel.Speech coding based upon vector quantization.*IEEE Transactions on Acoustics Speech and*

Signal Processing ASSP-28:562-574,October 1980.

[10] W. H. Chen and C.H. Smith.Adaptive coding of monochrome and color images.*IEEE Trans. Commun.* COM-22: 1285-1292, November 1977.

[11] E.J. Delp and D.R. Mitchell. Image compression using block truncation coding.*IEEE Trans.Commun.* COM-27,Sept. 1979.

[12] Paul Michael Farrelle .*Recursive block coding for imag*e data *compression*.Springer-Verlag NewYork Inc., New York,NY 1990.

[13] Rafael G. Gonzales and Richard E. Woods. *Digital image processing*. Addison Wesley Publishing company, Inc.,NewYork,1992.

[14] Robert M. Gray. Vector quantization. *IEEE Acoustics,Speech, and Signal processing magazine*.1:4-29,April 1984.

[15] Robert M. Gray and Karinin E. Multiple local optima in vector quantizers. *IEEE Transaction on information theory*,IT-28:256-261,March 1992.

[16] D.J.Healy and D.R.Mitchell .Digital video bandwidth compression using block truncation coding.*IEEE Trans.Commun.* COM-29:1809-1817,Dec. 1981.

[17] Anil K. Jain. Image data compression : A review. *Proc. of IEEE.* 69(3):349-389,March 1981.

[18] Brain W. Kernighan and Dennis M. Ritchie.*The C programming language*.Prentice Hall software series,Prentice Hall, Inc.,Englewood Cliffs,NJ.,1988.

[19] D.R. Knudson.Digital encoding of newspaper graphics.*Electron.Syst.Lab.*

MIT.Rep. ESL-616, August 1975.

[20] Weiping Li. Vector transform and image coding. *IEEE Transaction on Circuits and Systems for Video Technology*, 1(4):297-307,December 1991.

[21] Jae S. Lim. *Two-Dimensional Signal and Image Processing*. Prentice Hall Signal processing Series.Prentice Hall,Inc.,Englewood Cliffs,NJ,1990.

[22] Yoseph Linde, Andrés Buzo, and Robert M. Gray. An algorithm for vector quantizer design. *IEEE Transaction on communications*,COM-28(1):84-95,January 1980.

[23] S.P.Lloyd. Least squares quantization in PCM.*Bell Laboratories Technical Note*,1957.

[24] Thomas J. Lynch. *Data compression techniques and applications*. Lifetime Learning Publications.,Belmont,CA.,1985.

[25] Henrique S. Malvar and David H. Staelin. The lot: Transform coding without blocking effects. *IEEE Transc. on Acoustics,Speech, and Signal Processing*,37(4):553-559, April 1989.

[26] Michael J. Miller. Multimedia. *Pc Magazine*,page 114,March 1992.

[27] K.N. Nagan. Adaptive transform coding of video signals.*Proc.IEEE*,129: Pt. F., Feb. 1982.

[28] Nasser M. Nasrabadi and Robert A.King. Image coding using vector quantization : A review. *IEEE Transactions on communications*,COM-36(8):957-971,August 1988.

[29] N. M. Nasrabadi abd R. A. King. Computationally efficient adaptive

block-transform coding.*Proc.EUSIPCO-83 and European Conf.*,Sept. 1983.

[30] N.M. Nasrabadi.Use of vector quantizers in image coding.*Proc.IEEE Int.Conf.Acoust.,Speech,Signal Processing*.page:125-128,March 1985.

[31] Mark Nelson. *The data compression book*. M&T Publishing Inc.,Redwood city,CA.,1992.

[32] Arun N. Netravali and Barry G. Haskell. *Digital Pictures:Representation and compression*. Applications Communications Theory.Plenum Publishing Corporation,New York,NY, 1988.

[33] W.k. Partt.*Image transmission techniques*. Acadmic ,NewYork,1979.

[34] W. R. Pratt, W.H.Chen, and L.R. Welch.Slant transform image coding.*IEEE Trans. Commun.* COM-25:1075-1093, August 1974.

[35] J.M.Sabin and R.M.Gray.Product code vector quantizers for speech waveforms coding.*Conference Record Globecom '82*.page:1087-1091,December 1982.

[36] J.M.Sabin and R.M.Gray.Product code vector quantizers for waveform and voice coding.*IEEE Transaction on Acoustics Speech and Signal processing*.ASSP,April 1984.

[37] H.Samet. The quad-tree and related hierarchical data structures.*ACME Comput. Surveys*.16(2):188-260,June 1984.

[38] David Stepian.*Key papers in the development of information theory*. IEEE Press,NewYork,NY.,1974.

[39] M. Tasto and P.A. Wintz.Image coding by adaptive block quantization. *IEEE Trans.Commun. Technol.* COM-19: 956-972, December 1971.

[40] P.A. Wintz.Transform picture coding.Proc.IEEE.60:809-823,July 1972.

[41] John W. Woods, editor. Subband image coding. The Kluwer International Series in Engineering and Computer Science. Kluwer Acadmic Publishers, Norwell, MA, 1991.

[42] John W. Woods and Sean D. O'Neil. Subband coding of images. *IEEE Trans.on Acoustics,Speech,and Signal Processing*,ASSP-34(5):1278-1288,October 1986.

[43] Johnson K. Yan and david J. Sakrion. Encoding of images based on a two-component source model. *IEEE Transactions on Communications*,COM-25(11):1315-1322,Nov.1977.

[44] R.E. Ziemer and W.H. *Princeples of communications,system,modulation,noise.* Houghton Mifflin Co.,Boston,MA.,1985.

Appendix A

Th following pages contains detail results of the tests on the specific three coding schemes.

The term **IBER** means Inverse of Bit Error Rate.

The term **bpp** means bit per pixel.

VECTOR QUANTIZATION

(0.25 bpp)

IBER = 1000
Original Image: lady512.l
Quantized Image: ladyN1.25
Difference Image: out
max error = 142, min error = -129, rms error = 12.6239
max signal = 212, min signal = 44, rms signal = 132.564
peak SNR = 22.4823 (db), rms SNR = 20.4247 (db)

Original Image: lady.VQ25
Quantized Image: ladyN1.25
Difference Image: out
max error = 88, min error = -91, rms error = 3.37617
max signal = 212, min signal = 44, rms signal = 132.564
peak SNR = 33.9377 (db), rms SNR = 31.88 (db)

IBER = 2000

Original Image: lady512.l
Quantized Image: ladyN2.25
Difference Image: out
max error = 142, min error = -129, rms error = 12.3431
max signal = 212, min signal = 44, rms signal = 132.568
peak SNR = 22.6777 (db), rms SNR = 20.6203 (db)

Original Image: lady.VQ25
Quantized Image: ladyN2.25
Difference Image: out
max error = 91, min error = -86, rms error = 2.04983
max signal = 212, min signal = 44, rms signal = 132.568
peak SNR = 38.2718 (db), rms SNR = 36.2144 (db)

IBER = 3000

Original Image: lady512.l
Quantized Image: ladyN3.25
Difference Image: out

max error = 142, min error = -129, rms error = 12.2742
max signal = 212, min signal = 44, rms signal = 132.577
peak SNR = 22.7263 (db), rms SNR = 20.6695 (db)

Original Image: lady.VQ25
Quantized Image: ladyN3.25
Difference Image: out
max error = 88, min error = -84, rms error = 1.68842
max signal = 212, min signal = 44, rms signal = 132.577
peak SNR = 39.9566 (db), rms SNR = 37.8998 (db)

IBER = 4000

Original Image: lady512.l
Quantized Image: ladyN4.25
Difference Image: out
max error = 142, min error = -129, rms error = 12.251
max signal = 212, min signal = 44, rms signal = 132.577
peak SNR = 22.7428 (db), rms SNR = 20.686 (db)

Original Image: lady.VQ25
Quantized Image: ladyN4.25
Difference Image: out
max error = 88, min error = -91, rms error = 1.48147
max signal = 212, min signal = 44, rms signal = 132.577
peak SNR = 41.0923 (db), rms SNR = 39.0355 (db)

IBER = 5000

Original Image: lady512.l
Quantized Image: ladyN5.25
Difference Image: out
max error = 142, min error = -129, rms error = 12.2255
max signal = 212, min signal = 44, rms signal = 132.586
peak SNR = 22.7609 (db), rms SNR = 20.7046 (db)

Original Image: lady.VQ25
Quantized Image: ladyN5.25
Difference Image: out
max error = 91, min error = -84, rms error = 1.28066
max signal = 212, min signal = 44, rms signal = 132.586
peak SNR = 42.3575 (db), rms SNR = 40.3013 (db)

IBER =6000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 142, min error = -129, rms error = 12.2315
max signal = 212, min signal = 44, rms signal = 132.562
peak SNR = 22.7566 (db), rms SNR = 20.6988 (db)

Original Image: lady.VQ25
Quantized Image: ladyN
Difference Image: out
max error = 88, min error = -53, rms error = 1.34671
max signal = 212, min signal = 44, rms signal = 132.562
peak SNR = 41.9207 (db), rms SNR = 39.8629 (db)

IBER= 7000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 142, min error = -129, rms error = 12.2014
max signal = 212, min signal = 44, rms signal = 132.581
peak SNR = 22.778 (db), rms SNR = 20.7215 (db)

Original Image: lady.VQ25
Quantized Image: ladyN
Difference Image: out
max error = 83, min error = -83, rms error = 0.996857
max signal = 212, min signal = 44, rms signal = 132.581
peak SNR = 44.5335 (db), rms SNR = 42.477 (db)

IBER= 8000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 142, min error = -129, rms error = 12.2065
max signal = 212, min signal = 44, rms signal = 132.576
peak SNR = 22.7744 (db), rms SNR = 20.7175 (db)

Original Image: lady.VQ25
Quantized Image: ladyN
Difference Image: out
max error = 85, min error = -88, rms error = 1.03599
max signal = 212, min signal = 44, rms signal = 132.576
peak SNR = 44.1991 (db), rms SNR = 42.1422 (db)

IBER= 9000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 142, min error = -129, rms error = 12.2115
max signal = 212, min signal = 44, rms signal = 132.573
peak SNR = 22.7708 (db), rms SNR = 20.7137 (db)

Original Image: lady.VQ25
Quantized Image: ladyN
Difference Image: out
max error = 86, min error = -88, rms error = 1.11871
max signal = 212, min signal = 44, rms signal = 132.573
peak SNR = 43.5318 (db), rms SNR = 41.4747 (db)

IBER = 10000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 142, min error = -129, rms error = 12.1845
max signal = 212, min signal = 44, rms signal = 132.583
peak SNR = 22.79 (db), rms SNR = 20.7336 (db)

snr lady.VQ25 ladyN 512 512 out
Original Image: lady.VQ25
Quantized Image: ladyN
Difference Image: out

max error = 49, min error = -53, rms error = 0.773452
max signal = 212, min signal = 44, rms signal = 132.583
peak SNR = 46.7375 (db), rms SNR = 44.6811 (db)

IBER= 50000 (ONE BIT ONLY)

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 142, min error = -129, rms error = 12.1663
max signal = 212, min signal = 44, rms signal = 132.579
peak SNR = 22.803 (db), rms SNR = 20.7463 (db)

Original Image: lady.VQ25
Quantized Image: ladyN
Difference Image: out
max error = 0, min error = -47, rms error = 0.343206
max signal = 212, min signal = 44, rms signal = 132.579
peak SNR = 53.7951 (db), rms SNR = 51.7384 (db)

VECTOR QUANTIZATION

(0.50 bpp)

IBER 1000

Original Image: lady512.1
Quantized Image: ladyN1.50
Difference Image: out
max error = 165, min error = -154, rms error = 7.65057
max signal = 228, min signal = 31, rms signal = 132.901
peak SNR = 28.2154 (db), rms SNR = 24.7967 (db)

Original Image: lady.VQ50
Quantized Image: ladyN1.50
Difference Image: out
max error = 166, min error = -138, rms error = 3.66237
max signal = 228, min signal = 31, rms signal = 132.901
peak SNR = 34.6141 (db), rms SNR = 31.1953 (db)

IBER = 2000

Original Image: lady512.1
Quantized Image: ladyN
Difference Image: out
max error = 167, min error = -162, rms error = 7.18952
max signal = 228, min signal = 31, rms signal = 132.913
peak SNR = 28.7553 (db), rms SNR = 25.3373 (db)

Original Image: lady.VQ50
Quantized Image: ladyN
Difference Image: out
max error = 162, min error = -158, rms error = 2.61217
max signal = 228, min signal = 31, rms signal = 132.913
peak SNR = 37.5493 (db), rms SNR = 34.1313 (db)

IBER = 3000

snr lady512.l ladyN 512 512 out
Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 113, min error = -140, rms error = 7.10441
max signal = 228, min signal = 31, rms signal = 132.911
peak SNR = 28.8588 (db), rms SNR = 25.4407 (db)

Original Image: lady.VQ50
Quantized Image: ladyN
Difference Image: out
max error = 107, min error = -139, rms error = 2.30603
max signal = 228, min signal = 31, rms signal = 132.911
peak SNR = 38.632 (db), rms SNR = 35.2139 (db)

IBER =4000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 103, min error = -143, rms error = 6.89752
max signal = 228, min signal = 31, rms signal = 132.895
peak SNR = 29.1155 (db), rms SNR = 25.6963 (db)

Original Image: lady.VQ50
Quantized Image: ladyN
Difference Image: out
max error = 101, min error = -143, rms error = 1.63312
max signal = 228, min signal = 31, rms signal = 132.895
peak SNR = 41.629 (db), rms SNR = 38.2099 (db)

IBER =5000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 136, min error = -132, rms error = 6.89958
max signal = 228, min signal = 31, rms signal = 132.893

peak SNR = 29.1129 (db), rms SNR = 25.6936 (db)
Original Image: lady.VQ50
Quantized Image: ladyN
Difference Image: out

max error = 135, min error = -134, rms error = 1.61869
max signal = 228, min signal = 31, rms signal = 132.893
peak SNR = 41.706 (db), rms SNR = 38.2867 (db)

IBER =6000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 104, min error = -153, rms error = 6.85508
max signal = 228, min signal = 31, rms signal = 132.913
peak SNR = 29.1691 (db), rms SNR = 25.7511 (db)

Original Image: lady.VQ50
Quantized Image: ladyN
Difference Image: out
max error = 103, min error = -141, rms error = 1.39913
max signal = 228, min signal = 31, rms signal = 132.913
peak SNR = 42.9721 (db), rms SNR = 39.5542 (db)

IBER =7000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 97, min error = -134, rms error = 6.84916
max signal = 228, min signal = 31, rms signal = 132.901
peak SNR = 29.1766 (db), rms SNR = 25.7578 (db)

Original Image: lady.VQ50
Quantized Image: ladyN
Difference Image: out
max error = 95, min error = -131, rms error = 1.41557
max signal = 228, min signal = 31, rms signal = 132.901
peak SNR = 42.8707 (db), rms SNR = 39.452 (db)

IBER = 8000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 121, min error = -153, rms error = 6.94588

max signal = 228, min signal = 31, rms signal = 132.897
peak SNR = 29.0548 (db), rms SNR = 25.6358 (db)

Original Image: lady.VQ50
Quantized Image: ladyN
Difference Image: out
max error = 119, min error = -148, rms error = 1.77215
max signal = 228, min signal = 31, rms signal = 132.897
peak SNR = 40.9193 (db), rms SNR = 37.5003 (db)

IBER =9000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 118, min error = -110, rms error = 6.87552
max signal = 228, min signal = 31, rms signal = 132.895
peak SNR = 29.1432 (db), rms SNR = 25.7241 (db)

Original Image: lady.VQ50
Quantized Image: ladyN
Difference Image: out
max error = 123, min error = -105, rms error = 1.49522
max signal = 228, min signal = 31, rms signal = 132.895
peak SNR = 42.3953 (db), rms SNR = 38.9761 (db)

IBER =10000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 151, min error = -143, rms error = 6.87616
max signal = 228, min signal = 31, rms signal = 132.898
peak SNR = 29.1424 (db), rms SNR = 25.7234 (db)

Original Image: lady.VQ50
Quantized Image: ladyN
Difference Image: out
max error = 146, min error = -147, rms error = 1.50758
max signal = 228, min signal = 31, rms signal = 132.898
peak SNR = 42.3237 (db), rms SNR = 38.9047 (db)

IBER =50000 (ONLY ONE BIT)

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 85, min error = -73, rms error = 6.70955
max signal = 228, min signal = 31, rms signal = 132.897
peak SNR = 29.3555 (db), rms SNR = 25.9364 (db)

Original Image: lady.VQ50
Quantized Image: ladyN
Difference Image: out
max error = 23, min error = -9, rms error = 0.142872
max signal = 228, min signal = 31, rms signal = 132.897
peak SNR = 62.7904 (db), rms SNR = 59.3713 (db)

Vector Transform

(0.25 bpp)

IBER = 1000

Quantized Image: ladyN
Difference Image: out
max error = 100, min error = -115, rms error = 11.3849
max signal = 255, min signal = 0, rms signal = 132.684
peak SNR = 27.0042 (db), rms SNR = 21.3298 (db)

Quantized Image: ladyN
Difference Image: out
max error = 38, min error = -38, rms error = 4.19671
max signal = 255, min signal = 0, rms signal = 132.684
peak SNR = 35.6726 (db), rms SNR = 29.9982 (db)

IBER = 2000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 100, min error = -115, rms error = 10.7896
max signal = 255, min signal = 0, rms signal = 132.65
peak SNR = 27.4707 (db), rms SNR = 21.794 (db)

Original Image: lady.vt25
Quantized Image: ladyN
Difference Image: out
max error = 22, min error = -22, rms error = 2.12004
max signal = 255, min signal = 0, rms signal = 132.65
peak SNR = 41.6039 (db), rms SNR = 35.9273 (db)

IBER = 3000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 100, min error = -115, rms error = 10.816
max signal = 255, min signal = 0, rms signal = 132.65

peak SNR = 27.4495 (db), rms SNR = 21.7729 (db)
Original Image: lady.vt25
Quantized Image: ladyN
Difference Image: out
max error = 36, min error = -36, rms error = 2.22487
max signal = 255, min signal = 0, rms signal = 132.65
peak SNR = 41.1847 (db), rms SNR = 35.5081 (db)

IBER = 4000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 100, min error = -115, rms error = 10.7672
max signal = 255, min signal = 0, rms signal = 132.649
peak SNR = 27.4887 (db), rms SNR = 21.812 (db)

Original Image: lady.vt25
Quantized Image: ladyN
Difference Image: out
max error = 22, min error = -22, rms error = 2.00779
max signal = 255, min signal = 0, rms signal = 132.649
peak SNR = 42.0765 (db), rms SNR = 36.3998 (db)

IBER = 5000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 100, min error = -115, rms error = 10.6748
max signal = 255, min signal = 0, rms signal = 132.643
peak SNR = 27.5636 (db), rms SNR = 21.8865 (db)

Original Image: lady.vt25
Quantized Image: ladyN
Difference Image: out
max error = 23, min error = -23, rms error = 1.37013
max signal = 255, min signal = 0, rms signal = 132.643
peak SNR = 45.3956 (db), rms SNR = 39.7184 (db)

IBER = 6000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 100, min error = -115, rms error = 10.6369
max signal = 255, min signal = 0, rms signal = 132.639
peak SNR = 27.5945 (db), rms SNR = 21.9172 (db)

Original Image: lady.vt25
Quantized Image: ladyN
Difference Image: out
max error = 19, min error = -19, rms error = 0.98792
max signal = 255, min signal = 0, rms signal = 132.639
peak SNR = 48.2364 (db), rms SNR = 42.559 (db)

IBER = 7000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 100, min error = -115, rms error = 10.6873
max signal = 255, min signal = 0, rms signal = 132.634
peak SNR = 27.5535 (db), rms SNR = 21.8758 (db)

Original Image: lady.vt25
Quantized Image: ladyN
Difference Image: out
max error = 32, min error = -32, rms error = 1.45778
max signal = 255, min signal = 0, rms signal = 132.634
peak SNR = 44.857 (db), rms SNR = 39.1793 (db)

IBER = 8000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 100, min error = -115, rms error = 10.6086
max signal = 255, min signal = 0, rms signal = 132.639
peak SNR = 27.6177 (db), rms SNR = 21.9403 (db)

Original Image: lady.vt25
Quantized Image: ladyN
Difference Image: out

max error = 11, min error = -11, rms error = 0.673038
max signal = 255, min signal = 0, rms signal = 132.639
peak SNR = 51.57 (db), rms SNR = 45.8926 (db)

IBER = 9000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 100, min error = -115, rms error = 10.7123
max signal = 255, min signal = 0, rms signal = 132.647
peak SNR = 27.5332 (db), rms SNR = 21.8563 (db)

Original Image: lady.vt25
Quantized Image: ladyN
Difference Image: out
max error = 28, min error = -28, rms error = 1.66279
max signal = 255, min signal = 0, rms signal = 132.647
peak SNR = 43.7141 (db), rms SNR = 38.0372 (db)

IBER = 10000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 100, min error = -115, rms error = 10.6389
max signal = 255, min signal = 0, rms signal = 132.641
peak SNR = 27.5929 (db), rms SNR = 21.9157 (db)

Original Image: lady.vt25
Quantized Image: ladyN
Difference Image: out
max error = 19, min error = -19, rms error = 1.08812
max signal = 255, min signal = 0, rms signal = 132.641
peak SNR = 47.3973 (db), rms SNR = 41.7201 (db)

IBER = 50000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 100, min error = -115, rms error = 10.5893
max signal = 255, min signal = 0, rms signal = 132.637
peak SNR = 27.6334 (db), rms SNR = 21.9559 (db)

Original Image: lady.vt25
Quantized Image: ladyN
Difference Image: out
max error = 8, min error = -8, rms error = 0.194451
max signal = 255, min signal = 0, rms signal = 132.637
peak SNR = 62.3546 (db), rms SNR = 56.6771 (db)

ONLY ONE BIT

Original Image: lady512.1
Quantized Image: ladyN
Difference Image: out
max error = 100, min error = -115, rms error = 10.5886
max signal = 255, min signal = 0, rms signal = 132.637
peak SNR = 27.6341 (db), rms SNR = 21.9566 (db)

Vector Transform

(0.5 bpp)

IBER = 1000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 47, min error = -40, rms error = 4.74611
max signal = 247, min signal = 16, rms signal = 133.069
peak SNR = 33.7455 (db), rms SNR = 28.9548 (db)

Original Image: lady.vt50
Quantized Image: ladyN
Difference Image: out
max error = 30, min error = -30, rms error = 3.12131
max signal = 247, min signal = 16, rms signal = 133.069
peak SNR = 37.3855 (db), rms SNR = 32.5948 (db)

IBER = 2000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 82, min error = -55, rms error = 6.21201
max signal = 247, min signal = 3, rms signal = 132.812
peak SNR = 31.8831 (db), rms SNR = 26.6001 (db)

Original Image: lady.vt50
Quantized Image: ladyN
Difference Image: out
max error = 75, min error = -45, rms error = 5.09068
max signal = 247, min signal = 3, rms signal = 132.812
peak SNR = 33.6123 (db), rms SNR = 28.3292 (db)

IBER = 3000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out

max error = 45, min error = -47, rms error = 4.35514
max signal = 247, min signal = 12, rms signal = 133.019
peak SNR = 34.6413 (db), rms SNR = 29.6983 (db)

Original Image: lady.vt50
Quantized Image: ladyN
Difference Image: out
max error = 37, min error = -37, rms error = 2.49072
max signal = 247, min signal = 12, rms signal = 133.019
peak SNR = 39.4948 (db), rms SNR = 34.5518 (db)

IBER = 4000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 45, min error = -38, rms error = 3.75492
max signal = 247, min signal = 22, rms signal = 133.03
peak SNR = 35.5516 (db), rms SNR = 30.987 (db)

Original Image: lady.vt50
Quantized Image: ladyN
Difference Image: out
max error = 13, min error = -13, rms error = 1.1403
max signal = 247, min signal = 22, rms signal = 133.03
peak SNR = 45.9033 (db), rms SNR = 41.3386 (db)

IBER = 5000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 89, min error = -58, rms error = 6.25629
max signal = 252, min signal = 19, rms signal = 132.771
peak SNR = 31.4208 (db), rms SNR = 26.5357 (db)

Original Image: lady.vt50
Quantized Image: ladyN
Difference Image: out
max error = 84, min error = -55, rms error = 5.13612
max signal = 252, min signal = 19, rms signal = 132.771
peak SNR = 33.1344 (db), rms SNR = 28.2493 (db)

IBER = 6000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 56, min error = -56, rms error = 4.10346
max signal = 247, min signal = 22, rms signal = 133.02
peak SNR = 34.7807 (db), rms SNR = 30.2153 (db)

Original Image: lady.vt50
Quantized Image: ladyN
Difference Image: out
max error = 45, min error = -45, rms error = 1.99493
max signal = 247, min signal = 22, rms signal = 133.02
peak SNR = 41.0451 (db), rms SNR = 36.4798 (db)

IBER = 7000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 69, min error = -38, rms error = 4.48405
max signal = 248, min signal = 16, rms signal = 132.871
peak SNR = 34.2763 (db), rms SNR = 29.4352 (db)

Original Image: lady.vt50
Quantized Image: ladyN
Difference Image: out
max error = 37, min error = -20, rms error = 2.73201
max signal = 248, min signal = 16, rms signal = 132.871
peak SNR = 38.5801 (db), rms SNR = 33.739 (db)

IBER = 8000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 45, min error = -38, rms error = 3.60418
max signal = 247, min signal = 22, rms signal = 133.012
peak SNR = 35.9075 (db), rms SNR = 31.3417 (db)

Original Image: lady.vt50
Quantized Image: ladyN
Difference Image: out

max error = 8, min error = -8, rms error = 0.479579
max signal = 247, min signal = 22, rms signal = 133.012
peak SNR = 53.4265 (db), rms SNR = 48.8606 (db)

IBER = 9000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 45, min error = -38, rms error = 3.95473
max signal = 247, min signal = 0, rms signal = 133.021
peak SNR = 35.9116 (db), rms SNR = 30.5361 (db)

Original Image: lady.vt50
Quantized Image: ladyN
Difference Image: out
max error = 30, min error = -30, rms error = 1.69626
max signal = 247, min signal = 0, rms signal = 133.021
peak SNR = 43.2641 (db), rms SNR = 37.8885 (db)

IBER = 10000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 45, min error = -38, rms error = 3.73004
max signal = 247, min signal = 19, rms signal = 133.017
peak SNR = 35.7244 (db), rms SNR = 31.0439 (db)

Original Image: lady.vt50
Quantized Image: ladyN
Difference Image: out
max error = 15, min error = -15, rms error = 1.07759
max signal = 247, min signal = 19, rms signal = 133.017
peak SNR = 46.5096 (db), rms SNR = 41.829 (db)

IBER = 50000

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 45, min error = -38, rms error = 3.57541
max signal = 247, min signal = 22, rms signal = 133.012

peak SNR = 35.9771 (db), rms SNR = 31.4113 (db)
Original Image: lady.vt50
Quantized Image: ladyN
Difference Image: out
max error = 4, min error = -4, rms error = 0.132467
max signal = 247, min signal = 22, rms signal = 133.012
peak SNR = 64.6015 (db), rms SNR = 60.0356 (db)

ONE BIT ONLY

Original Image: lady512.l
Quantized Image: ladyN
Difference Image: out
max error = 45, min error = -38, rms error = 3.57413
max signal = 247, min signal = 22, rms signal = 133.012
peak SNR = 35.9803 (db), rms SNR = 31.4144 (db)

Original Image: lady.vt50
Quantized Image: ladyN
Difference Image: out
max error = 3, min error = -3, rms error = 0.0758959
max signal = 247, min signal = 22, rms signal = 133.012
peak SNR = 69.4393 (db), rms SNR = 64.8734 (db)

DCT coding

(0.50 bpp)

IBER = 1000

Original Image: lady512.l
Quantized Image: lady43
Difference Image: out
max error = 172, min error = -181, rms error = 14.2043
max signal = 255, min signal = 1, rms signal = 131.861
peak SNR = 25.0483 (db), rms SNR = 19.3539 (db)

Original Image: lady43.dct
Quantized Image: lady43
Difference Image: out
max error = 166, min error = -166, rms error = 7.52629
max signal = 255, min signal = 1, rms signal = 131.861
peak SNR = 30.5651 (db), rms SNR = 24.8707 (db)

IBER = 2000

snr lady512.l lady43 512 512 out
Original Image: lady512.l
Quantized Image: lady43
Difference Image: out
max error = 142, min error = -176, rms error = 13.2701
max signal = 255, min signal = 1, rms signal = 131.99
peak SNR = 25.6392 (db), rms SNR = 19.9533 (db)

Original Image: lady43.dct
Quantized Image: lady43
Difference Image: out
max error = 127, min error = -127, rms error = 5.81362
max signal = 255, min signal = 1, rms signal = 131.99
peak SNR = 32.8077 (db), rms SNR = 27.1219 (db)

IBER = 3000

Original Image: lady512.l
Quantized Image: lady43
Difference Image: out
max error = 141, min error = -186, rms error = 12.6589
max signal = 255, min signal = 1, rms signal = 131.974
peak SNR = 26.0487 (db), rms SNR = 20.3618 (db)

Original Image: lady43.dct
Quantized Image: lady43
Difference Image: out
max error = 126, min error = -127, rms error = 4.37836
max signal = 255, min signal = 1, rms signal = 131.974
peak SNR = 35.2704 (db), rms SNR = 29.5835 (db)

IBER = 4000

Original Image: lady512.l
Quantized Image: lady43
Difference Image: out
max error = 141, min error = -172, rms error = 12.4348
max signal = 255, min signal = 0, rms signal = 131.933
peak SNR = 26.238 (db), rms SNR = 20.5143 (db)

Original Image: lady43.dct
Quantized Image: lady43
Difference Image: out
max error = 126, min error = -127, rms error = 3.71034
max signal = 255, min signal = 0, rms signal = 131.933
peak SNR = 36.7425 (db), rms SNR = 31.0188 (db)

IBER = 5000

Original Image: lady512.l
Quantized Image: lady43
Difference Image: out
max error = 141, min error = -172, rms error = 13.0519
max signal = 255, min signal = 1, rms signal = 132.047
peak SNR = 25.7832 (db), rms SNR = 20.1011 (db)

Original Image: lady43.dct
Quantized Image: lady43
Difference Image: out
max error = 126, min error = -127, rms error = 5.33984
max signal = 255, min signal = 1, rms signal = 132.047
peak SNR = 33.5461 (db), rms SNR = 27.864 (db)

IBER = 6000

Original Image: lady512.l
Quantized Image: lady43
Difference Image: out
max error = 141, min error = -153, rms error = 12.2546
max signal = 255, min signal = 1, rms signal = 131.949
peak SNR = 26.3307 (db), rms SNR = 20.6421 (db)

Original Image: lady43.dct
Quantized Image: lady43
Difference Image: out
max error = 127, min error = -127, rms error = 3.09066
max signal = 255, min signal = 1, rms signal = 131.949
peak SNR = 38.2957 (db), rms SNR = 32.6071 (db)

IBER = 7000

Original Image: lady512.l
Quantized Image: lady43
Difference Image: out
max error = 141, min error = -157, rms error = 12.0821
max signal = 255, min signal = 1, rms signal = 131.942
peak SNR = 26.4538 (db), rms SNR = 20.7648 (db)

Original Image: lady43.dct
Quantized Image: lady43
Difference Image: out
max error = 60, min error = -127, rms error = 2.34941
max signal = 255, min signal = 1, rms signal = 131.942
peak SNR = 40.6775 (db), rms SNR = 34.9885 (db)

IBER = 8000

Original Image: lady512.l
Quantized Image: lady43
Difference Image: out
max error = 141, min error = -151, rms error = 12.2916
max signal = 255, min signal = 1, rms signal = 131.923
peak SNR = 26.3045 (db), rms SNR = 20.6142 (db)

Original Image: lady43.dct
Quantized Image: lady43
Difference Image: out
max error = 127, min error = -127, rms error = 3.17712
max signal = 255, min signal = 1, rms signal = 131.923
peak SNR = 38.056 (db), rms SNR = 32.3657 (db)

IBER = 9000

Original Image: lady512.l
Quantized Image: lady43
Difference Image: out
max error = 141, min error = -133, rms error = 11.9117
max signal = 255, min signal = 1, rms signal = 131.901
peak SNR = 26.5772 (db), rms SNR = 20.8854 (db)

Original Image: lady43.dct
Quantized Image: lady43
Difference Image: out
max error = 61, min error = -15, rms error = 1.14904
max signal = 255, min signal = 1, rms signal = 131.901
peak SNR = 46.89 (db), rms SNR = 41.1982 (db)

IBER = 10000

Original Image: lady512.l
Quantized Image: lady43
Difference Image: out
max error = 177, min error = -133, rms error = 11.9733
max signal = 255, min signal = 1, rms signal = 131.937
peak SNR = 26.5324 (db), rms SNR = 20.8431 (db)

Original Image: lady43.dct
Quantized Image: lady43

Difference Image: out
max error = 107, min error = -107, rms error = 1.64263
max signal = 255, min signal = 1, rms signal = 131.937
peak SNR = 43.7859 (db), rms SNR = 38.0965 (db)

IBER = 50000

Original Image: lady512.l
Quantized Image: lady43
Difference Image: out
max error = 141, min error = -133, rms error = 11.8518
max signal = 255, min signal = 1, rms signal = 131.926
peak SNR = 26.621 (db), rms SNR = 20.9309 (db)

Original Image: lady43.dct
Quantized Image: lady43
Difference Image: out
max error = 22, min error = -6, rms error = 0.395593
max signal = 255, min signal = 1, rms signal = 131.926
peak SNR = 56.1517 (db), rms SNR = 50.4616 (db)

ONLY ONE BIT.

Original Image: lady512.l
Quantized Image: lady43
Difference Image: out
max error = 141, min error = -133, rms error = 11.8484
max signal = 255, min signal = 1, rms signal = 131.927
peak SNR = 26.6235 (db), rms SNR = 20.9335 (db)

Original Image: lady43.dct
Quantized Image: lady43
Difference Image: out
max error = 11, min error = 0, rms error = 0.171875
max signal = 255, min signal = 1, rms signal = 131.927
peak SNR = 63.3924 (db), rms SNR = 57.7024 (db)

Biography

Ahmed Yehia Banafa was born in Aden, Yemen on January 15 1965.

He received the B.S. degree from the King Abodulaziz University, Jeddah, Saudi Arabia, in 1986, and he is working towards the M.S. degree from Lehigh University, Bethlehem PA, both in Electrical Engineering.

During summer of 1984 ,he worked with Ericsson Telecom Company Jeddah,Saudi Arabia as an assistant engineer for installing ,testing , and maintenance of digital telephone exchanges. In summer 1985, a practical training on the communications network of Jeddah Sea Port , was preformed as a partial requirement for his B.S. degree. In 1989 to 1990 ,he worked for Northern Telcom Company , Jeddah, Saudi Arabia as operation engineer for installing, testing and maintenance of large digital Business communications systems , including computer networks,telephone networks, and data networks.

During his undergraduate study , he received two distinction awards in 1981, 1982. His senior Project for B.S. was on designing of small scale digital communication system .

His Thesis presents a study of channel noise effects on digital image compression schemes.

He is a student member of IEEE

He is a reviewer for Potential Magazine of IEEE.

END

OF

TITLE