65500

DIGITAL IMAGE COMPRESSION TECHNIQUES

A Thesis Submitted to the
Graduate School of Natural and Applied Science of
Dokuz Eylül University
In Partial Fulfillment of the Requirements for
the Degree of Master of Science in Electrical and Electronics Engineering

by

İlker KILIÇ

February, 1997

İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

[illegible]

Yrd. Doç. Dr. Reyat Yılmaz
(Advisor)

[illegible]

.....Dr. Yavuz.....ŞENOL

(Committee Member)

[illegible]

..Dr....Haldun....Sarnel

(Committee Member)

Approved by the
Graduate School of Natural and Applied Sciences

[illegible]

Prof. Dr. Macit TOKSOY
Director

ACKNOWLEDGMENTS

I want to thank to assistant professor Reyat YILMAZ for his supervising and to research assistants Ahmet ÖZKURT and Güleser KALAYCI for their helps and to assistant professor Bekir KARLIK for providing the biomedical images that I used in the thesis.

İlker KILIÇ

ABSTRACT

In some science branches it is required that images are processed and stored into memory. However, 65536 byte of memory is required for an image of 256x256 pixel size to be stored without any processing. This causes serious memory problem if we have lots of images to be stored. In order to overcome this problem different image compression techniques are used.

In this thesis, the most popular image compression techniques of nowadays were investigated. It is seen that Run-Length Coding Algorithm is successful at low compression ratios. If the compression ratio is increased the algorithm becomes unsuccessful, and horizontal lines, which are undesired covers the image. The Discrete Fourier Transform and the Discrete Cosine Transform are better than Run-Length algorithm at many different compression rates. Nowadays, the other mostly used compression technique is the Vector Quantization. In this technique, the image is divided into blocks of small images. All blocks are entered into a training algorithm. Using this training algorithm, a codebook which can represent the image is obtained. The compression is achieved by using this codebook. By using this codebook at the reconstruction process, the image becomes very close to the original image.

The Hierarchical Finite State Vector Quantization (HFSVQ) is improved version of Vector Quantization technique. In this technique, the image is divided into blocks of different sizes. Training algorithm is applied to the each block of group. By using the codebook the compression process is achieved. It is noticed that this algorithm has perfect results when the original image has large areas of constant gray level.

In this thesis HFSVQ algorithm was applied on different biomedical images (MRI and BT images). In this algorithm the sizes of the blocks change. Large blocks were used to represent the low contrast area of the image so high compression ratios were obtained. Small blocks were used to represent the high contrast area of the image so low compression ratios were obtained. Since the backgrounds of all the biomedical images have large areas of constant gray level, it is seen that HFSVQ algorithm is very suitable for biomedical images. Finally, HFSVQ algorithm was applied on different MRI and BT images and very low mean square errors were obtained at high compression ratios. It is seen that HFSVQ algorithm which is used to compress the biomedical images is better than mostly other techniques which have been used so far.

ÖZET

Bazı bilim dallarında görüntünün işlenip bilgisayarda saklanması gerekmektedir. Oysa 256x256 piksel çözünürlüğe sahip bir görüntünün hiç işlenmeden saklanması için hafizada 65536 byte' lık yere ihtiyaç vardır. Bu da özellikle çok resim saklanması gerektiren durumlarda ciddi bellek problemi oluşturur. Bu problemi ortadan kaldırmak için değişik görüntü sıkıştırma teknikleri geliştirilmiştir.

Bu tezde, günümüzde en çok kullanılan görüntü sıkıştırma teknikleri incelenmiştir. Bunlardan Run-Length coding ' in az bir sıkıştırma gerektiren uygulamalarda başarılı olduğu gözlenmiştir. Sıkıştırma oranını arttırdığımızda bu algoritmanın başarısız olup, sonuçta oluşan uzun yatay çizgilerin resmi anlaşılmaz hale getirdiği tespit edilmiştir. Ayrık Fourier ve Kosinüs Transform görüntü sıkıştırma teknikleri birçok sıkıştırma oranında daha başarılı sonuç veren yöntemlerdir. Son zamanlarda sıkça kullanılan bir diğer sıkıştırma tekniği ise vektör kuantalamadır. Bu teknikte resim sabit büyüklükte bloklara ayrılır. Tüm bloklar bir eğitime tabi tutulup sonuçta resmi temsil edebilecek bir vektör grubu oluşturulur. Bu vektör grubu yardımıyla sıkıştırma yapıldığında sonuçta elde edilecek resmin orjinal resme yakınlığı göze çarpmıştır.

Vektör kuantalamanın ileri bir versiyonu olan Hiyerarşik vektör kuantalama yönteminde, resim, boyutları birbirinden farklı bloklara ayrılıp herbir blok grubu için ayrı eğitme işlemi yapılır. Sonuçta elde edilen herbir vektör grubu yardımıyla resim kodlanır. Bu yöntemin özellikle resimde çok boş alan olan uygulamalarda mükemmel bir sonuç verdiği gözlenmiştir.

Bu çalışmada Hiyerarşik vektör kuantalama algoritması değişik biyomedikal görüntüler üzerinde denenmiştir (MRI ve BT görüntüleri). Bu algoritmada kullanılan blok büyüklükleri değişkendir. Renk tonu değişiminin az olduğu bölgelerde büyük bloklar kullanılıp yüksek sıkıştırma, renk tonu değişiminin fazla olduğu bölgelerde ise küçük bloklar kullanılıp düşük bir sıkıştırma elde edilmiştir. Tüm biyomedikal görüntülerde fon, geniş bir yer kaplayıp sabit bir renk tonuna sahiptir. Bu yüzden bu algoritmanın biyomedikal görüntüler için uygun olduğu görülür. Sonuçta Hiyerarşik Vektör Kuantalama algoritması MRI ve BT görüntülerine uygulanıp çok yüksek sıkıştırma oranlarında çok düşük ortalama karesel hatalar elde edildi. Bundan başka bu algoritmanın şu ana kadar biyomedikal görüntüler üzerinde uygulanan diğer algoritmalardan daha iyi olduğu gözlendi.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE
INTRODUCTION

The Image data compression technique is one of the major part in the image processing. Very often, in practical situations, one encounters images defined over 256x256 or 512x512 pixels. The intensity value (gray level) at each pixel of an image is typically represented by 1 byte. Since a typical image usually has at least 256x256 pixels, representing an image digitally requires 65536 bytes. Such digital representations not only require large amounts of memory but often cause some problems if the image data needs to be transmitted on a severely band-limited medium such a telephone line. It is desirable to represent the data of an image with considerably fewer number of bits and at the same time be able to reconstruct this image that is close to the original image. Image compression refers to transforming an image data to a different representation which requires less memory.

Image compression techniques are useful in brodcast television, remote sensing via satellite, aircraft, radar, sonar, teleconferencing, computer communications, facsimile transmission, etc.

There are two image compression techniques, namely the lossy and the lossless. In the lossless compression technique, very low compression ratios such as 2 bit/pixel, 3 bit/pixel are obtained. However the reconstruct image is as same as the original one [7]. In lossy compression technique very high compression ratios such as 0.2 bit/pixel, 0.1 bit/pixel are obtained and the reconstruct image is not the same but very close to the original one. Since the compression ratio and the bit rate are very important at transmitting any digital information from one point to an other then lossy image compression technique is more popular than the other technique.

Different image compression techniques have been improved by the help of discrete signal processing. Discrete Fourier Transform is a kind of compression technique but it is generally used as a first step of other compression techniques. The most popular and efficient method called Vector Quantization (VQ) was presented by Robert M. Gray [5] in 1984, and improved by Nasser M. Nasrabadi & Robert A. King [10] in 1988. In the following years the VQ algorithm was again improved and mentioned in a paper by Nasser M. Nasrabadi & Yushu Feng [9] in 1990. Four years later, hierarchical finite state vector quantization algorithm was found by Ping Yu & Anastasios N. Venetsanopoulos [18].

In the second Chapter of this thesis, some image enhancement techniques such as filtering and histogram equalization are given. In the Chapter three, five image compression techniques are presented in details. They are the run-length compression algorithm, the discrete fourier transform (DFT), the vector quantization (VQ), the discrete cosine transform (DCT) and hierarchical finite state vector quantization (HFSVQ). All these algorithms were written in "C" which is a high-level language. The performance results of the algorithms (reconstructed images, SNR and MSE) were obtained and compared to each other. The comparison ratios and MSE values of these algorithms were given in the tables. Then these image compression techniques were applied to the biomedical images (MRI and BT) in the Chapter four. It was seen that the most successful one is HFSVQ. For comparison reason, the performance results of the other biomedical image compression techniques are also given in this Chapter. In the Chapter five, the conclusion is given.

CHAPTER TWO

DIGITAL IMAGE ENHANCEMENT

2.1 DIGITAL IMAGE REPRESENTATION

The term gray level image or simply image, refers to a two dimensional light intensity function f(x,y), where x and y denote spatial coordinates and the value of f at any point (x,y) is proportional to the brightness (or gray level) of the image at that point. Figure 2.1 illustrates the axis convention used throughout this thesis. Sometimes viewing an image function in perspective with the third axis being brightness is useful. Figure 2.1 viewed in this wat and would appear as a series of active peaks in regions with numerous changes in brightness levels and smoother regions or plateaus where the brightness levels varied little or were constant. Assigning proportionately higher values to brighter areas would make the height of the components in the plot proportional to the corresponding brightness[2].

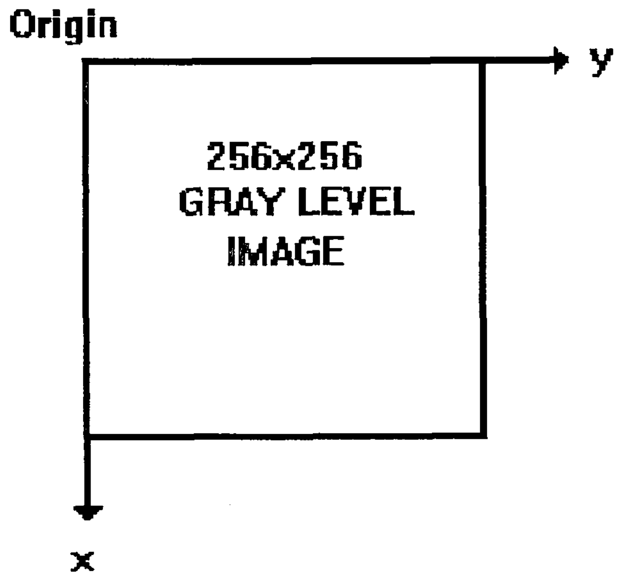


Figure 2.1 The Image Coordinate System Definition

A digital image is an image f(x,y) that is discretized both in spatial coordinates and brightness. A digital image can be considered as a matrix whose row and column indices identify a point in the image and the correspondind matrix element value represents the gray level at that point. The elements of such a digital array are called image elements, pixels.

2.2 HISTOGRAM PROCESSING

The histogram of a digital image with gray levels in the range [0, L-1] is a discrete function $p(r_k) = n_k / n$. Here r_k is the k th gray level, n_k is the number of pixels in the image with that gray level, n is the total number of pixels in the image, and k=0, 1, 2,..., L-1

Here $p(r_k)$ gives an estimate of the probability of occurrence of gray level r_k. A plot of this function for all values of k provides a global description of an image. For example, Figure 2.3 shows the histograms of four basic types of images. The histogram shown in Figure 2.3 (a) shows that the gray levels are concentrated toward the zero gray level. Thus this histogram corresponds to an image with overall dark characteristics. Just opposite is true in Figure 2.3 (b). The histogram shown in Figure 2.3 (c) has a narrow shape, which indicates little dynamic range and thus represents an image having low contrast. Since all gray levels occur toward the middle of the gray scale, the image would appear a murky gray. Figure 2.3 (d) shows a histogram with significant spread, represents an image with high contrast. Figure 2.3 (e) shows the histogram of the image in Figure 2.2.

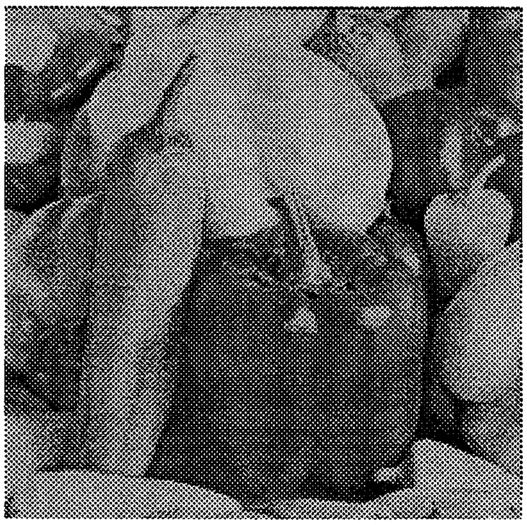
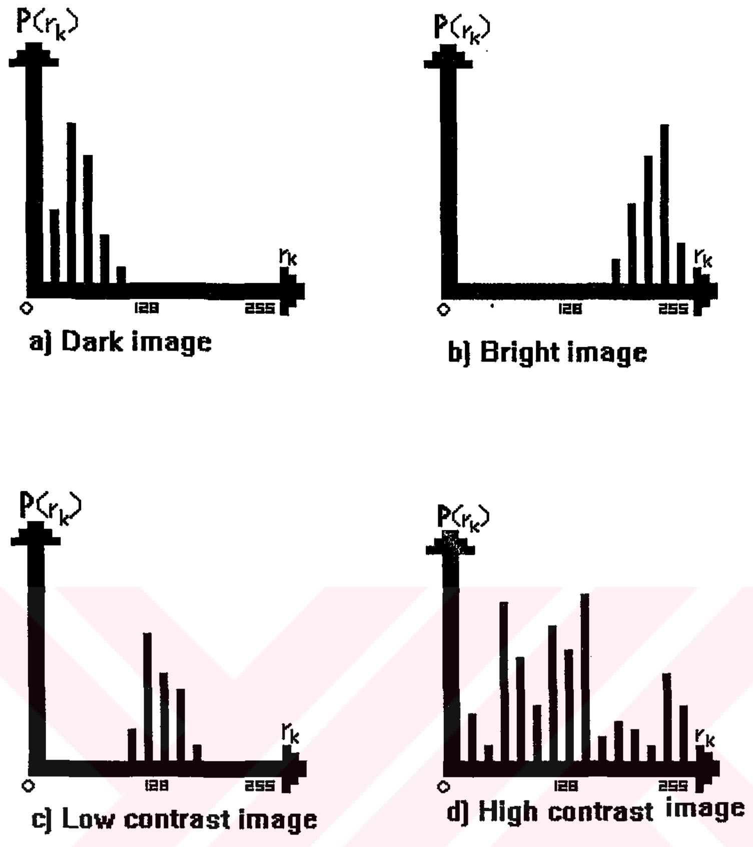
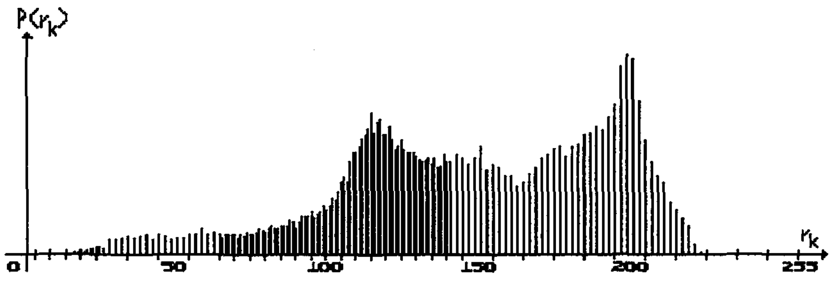


Figure 2.2 Original 256x256 Peppers Image





e) Histogram of The Image in Fig 2.2

Figure 2.3 Histograms Corresponding to Four Basic Image Types

2.3 HISTOGRAM EQUALIZATION

The purpose of histogram equalization technique is to process an image so that the result is more suitable than the original image for a specific application [6].

Let the variable r represent the gray levels in the image. In the initial part of our discussion, I assume that pixel values are continuous quantities that have been normalized so that they lie in the interval [0,1], with r=0 representing black and r=1 representing white. Later, I consider a discrete formulation and allow pixel values to be in the interval [0, L-1]. L is the number of gray levels.

For any r in the interval [0,1], the transformation function is given by

$$s = T(r) \qquad 2.1$$

which produce a level s for every pixel value r in the original image. It is assumed that the transformation function given in Eq. 2.1 satisfies the following conditions:

(a) T(r) is single-valued and monotonically increasing function in the interval $0 \le r \le 1$.

(b) $0 \le T(r) \le 1 \;\; for \;\; 0 \le r \le 1$.

Condition (a) preserves the order from black to white in the gray scale. Condition (b) guarantees a mapping that is consistent with the allowed range of pixel values.

In order to be useful for digital image processing, the transformation function must be used in the discrete form. We deal with probabilities of the gray levels that take on discrete values:

$$P(r_k) = \frac{n_k}{n} \qquad 0 \le r_k \le 1 \qquad k = 0,1,2,\ldots,L-1 \qquad 2.2$$

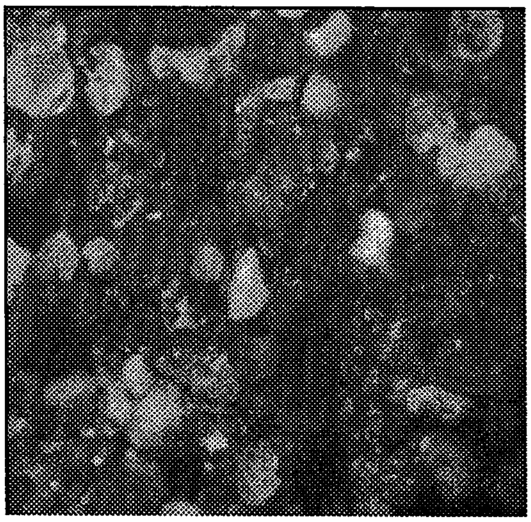
where $P(r_k)$ is the probability of the k th gray level, n_k is the number of times this level appears in the image, and n is the total number of pixels in the image. A plot of $P(r_k)$ versus r_k is called a histogram, and the technique used for obtaining a uniform histogram is known as histogram equalization. The discrete form of the transformation;

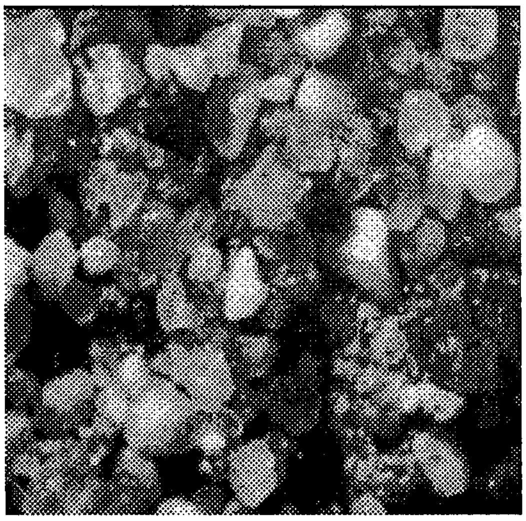
$$s_k = T(r_k) = \sum_{j=0}^{k} \frac{n_j}{n} = \sum_{j=0}^{k} P(r_k) \qquad 0 \le r_k \le 1 \quad k = 0,1,2,\ldots,L-1 \qquad 2.3$$

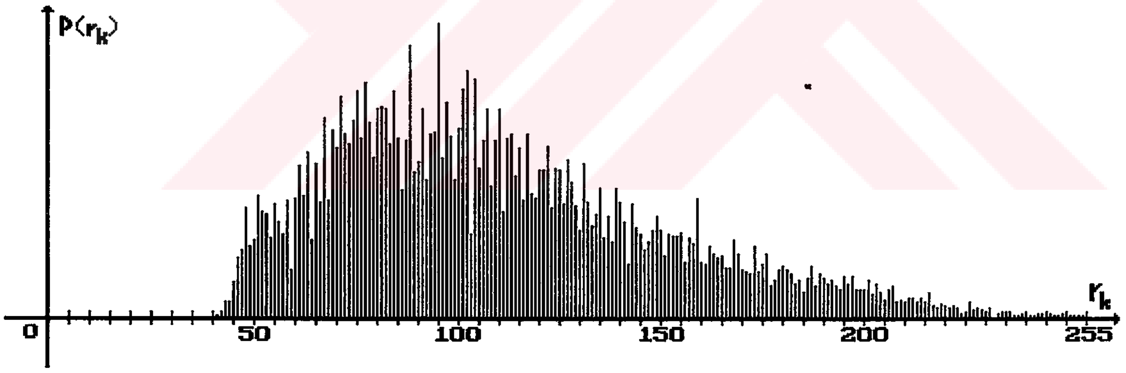
The inverse transformation is denoted;

$$r_k = T^{-1}(s_k) \quad 0 \le s_k \le 1 \qquad 2.4$$

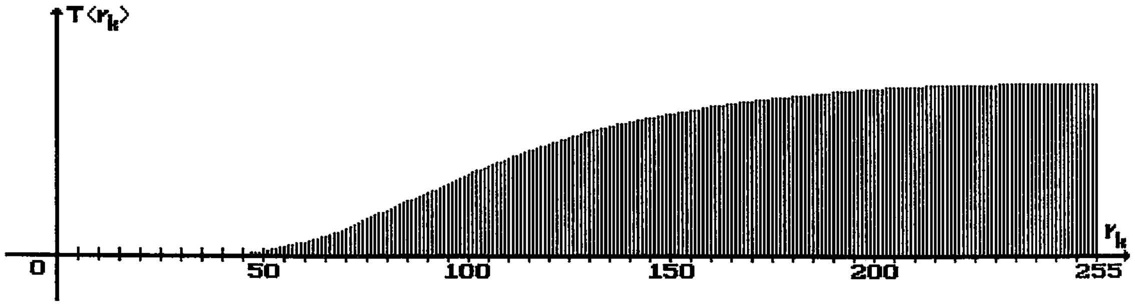
where both $T(r_k)$, that is called as the cumulative distribution function of r, and $T^{-1}(s_k)$ are assumed to satisfy conditions (a) and (b). $T^{-1}(s_k)$ represent the histogram of new image.


(a) Original Image 1 With The Size Of 256x256

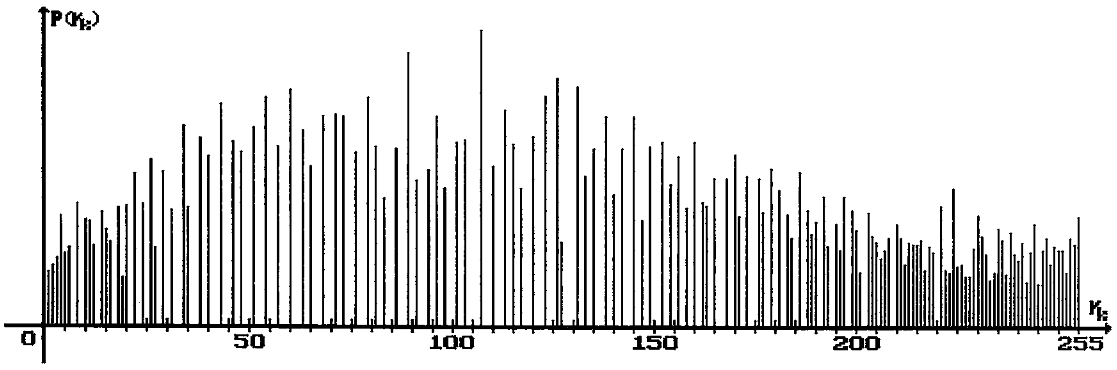

(b) The Image After Histogram Equalization



(c) The Histogram of the Original Image



(d) The Cumulative Distribution of the Original Image



(e) The Histogram After the Histogram Equalization Process

Figure 2.4 Histogram Equalization Process

2.4 FILTERING

2.4.1 Neighborhood

The principal approach to defining a neighborhood about (x,y) is to use a square or rectangular subimage area centered at (x,y) as in Figure 2.5. The center of subimage is moved from pixel to pixel starting at the top left corner and applying the operator at each location (x,y) to yield g at that location g(x,y)=T[f(x,y)]. Square and rectangular neighborhood shapes are common because their implementations are easy. A square shaped operator which has 3x3 coefficients to use for the filtering is called mask as you see in Figure 2.6.

Lowpass filters attenuate or eliminate high-frequency components in the Fourier domain while the filter passes low frequencies. High-frequecy components characterize edges and other sharp details in an image, so the effect of lowpass filtering is image blurring. Highpass filters attenuate or eliminate low-frequency components. Since these components are responsible for the slowly varying characteristics of an image, such as overall contrast and average intensity, the net result of highpass filtering is a reduction of these features and a correspondingly apparent sharpening of edges and other sharp details [4].

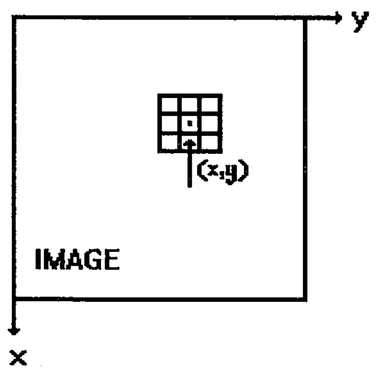


Figure 2.5 A 3x3 Neigborhood about a point (x,y) in an image

2.4.2 Lowpass (Smoothing) Filters

For a 3x3 filter, the simplest arrangement would be a mask in which all coefficients have a value of 1. However, the response would then be the sum of gray levels for nine pixels, which could cause response to be out of the valid gray-tone range. The solution is to scale the sum by dividing response by nine. Figure 2.6 (a) shows the resulting mask. In Figure 2.6 (b) a 5x5 mask. In all these cases, the response would be the average of all the pixels in the area of the mask.

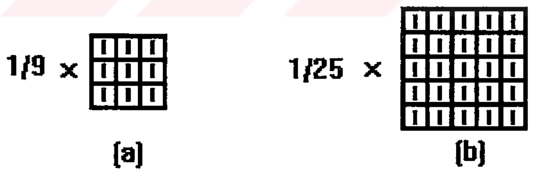


Figure 2.6 Lowpass filter masks of various sizes

2.4.3 Median filtering

One of the principal difficulties of the lowpass filtering is that it blurs edges and other sharp details. If the goal is to achieve noise reduction rather than blurring, an alternative approach is to use median filters. The gray level of each pixel is replaced by the median of the gray levels in a neighborhood of that pixel, instead of by the average. This method is effective when the noise pattern consists of strong, spikelike components and the characteristic to be preserved is edge sharpness. Median filters are nonlinear.

The median of a set of values (m) is such that half the values in the set are less than m and half are greater than m. In order to obtain median filtering in a neighborhood of a pixel, we first sort the values of the pixel and its neighbors, determine the median, and assign this value to the pixel. For example, in a 3x3 neighborhood the median is the 4th largest value, in a 5x5 neighborhood the 13th largest value, and so on. When several values in a neighborhood are the same, all equall values must be grouped. For example, 3x3 neighborhood has values (10, 20, 20, 20, 15, 20, 20, 25, 100). These values are separated as (10, 15, 20, 20, 20, 20, 20, 25, 100), which results in a median of 20. Thus the basic function of median filtering is to force points with distinct intensities to be more like their neighbors, actually eliminating intensity spikes that appear isolated in the area of the filter mask.

2.4.4 Highpass filtering

The shape of the impulse response needed to implement a highpass (sharpening) spatial filter indicates that the filter should have positive coefficients near its center, and negative coefficients in the outer part. For a 3x3 mask, choosing a positive value in the center location with negative coefficients in the rest of the mask.

Figure 2.7 shows the classic implementation of a 3x3 highpass filter. Note that the sum of the coefficients is zero. This means when the mask is over an area of constant or slowly varying gray level, the output of the mask is zero or very small. This result is reasonable with what is expected from the corresponding frequency domain filter. Note that this highpass filter mask eliminates the zero frequency term. Eliminating zero frequency term reduces the average gray-level value in the image to zero, reducing significantly the global contrast of the image.

$$1/9 \times \begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline -1 & 8 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

Figure 2.7 The highpass filter mask

Reducing the average value of an image to zero implies that the image must have some negative gray tones. As we deal only with positive levels, the results of highpass filtering involve some form of scaling so that the gray levels of the final result span the range [0,L-1] Taking the absolute value of the filtered image makes all values positive. This is not a good idea because large negative values would appear brightly in the image.

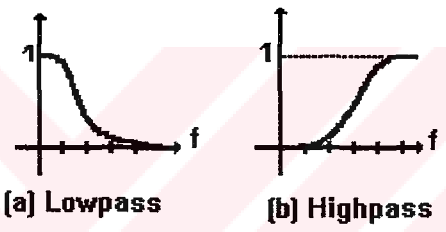
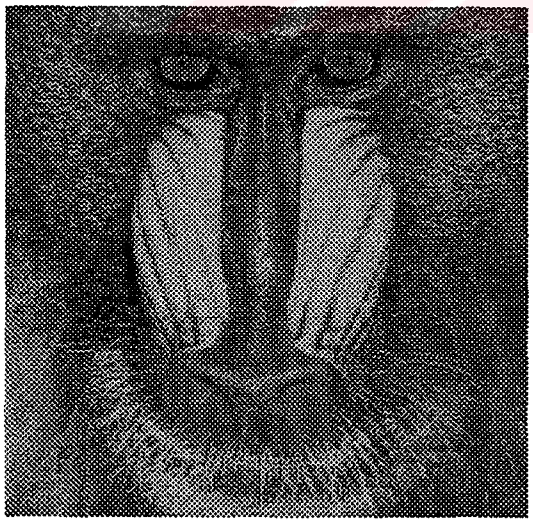
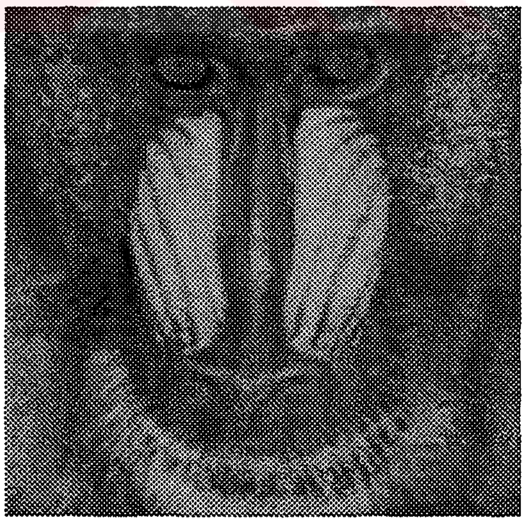
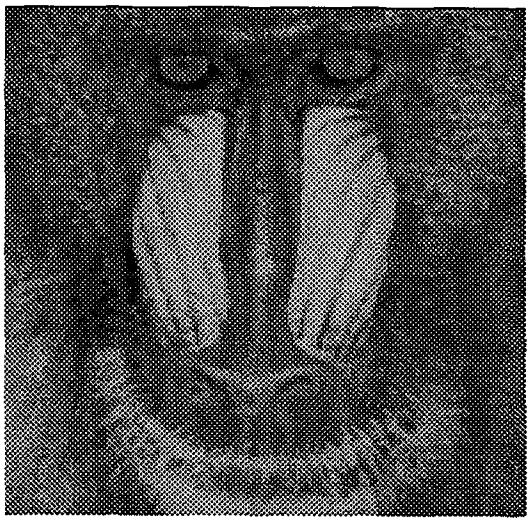


Figure 2.8 Basic shapes for circularly symmetric frequency domain filters


(a) Original 256x256 "Baboon" Image


(b) Smoothing Filtered Image


(c) Median Filtered Image

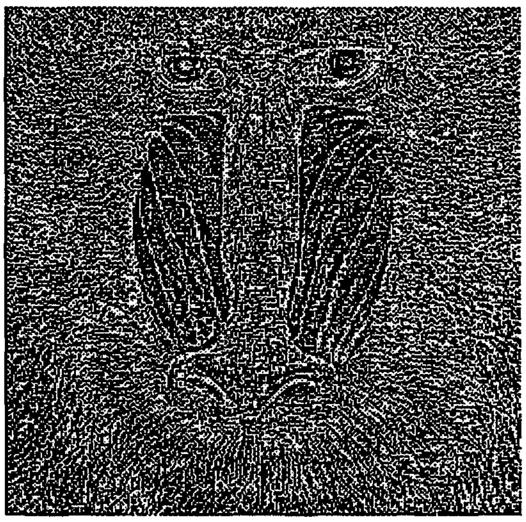

(d) Highpass Filtered Image

Figure 2.9 Filter Applications

CHAPTER THREE
IMAGE COMPRESSION TECHNIQUES

3.1 RUN-LENGTH ENCODING AND COMPRESSION

Run-length encoding and compression is a line based process where pixel intensities are not stored or transmitted individually, but rather on a given line [14].

Run-length encoding follows these steps below:

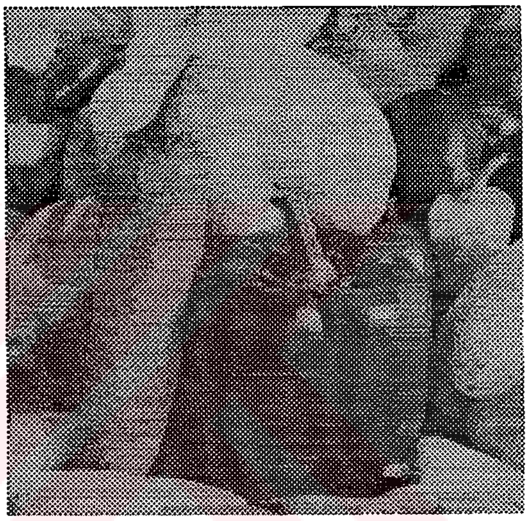
1. The gray level of the first pixel on a line is stored and denoted as reference pixel f(x, y). A run counter is set to 1, indicating that the current 'run' consists of 1 pixel.

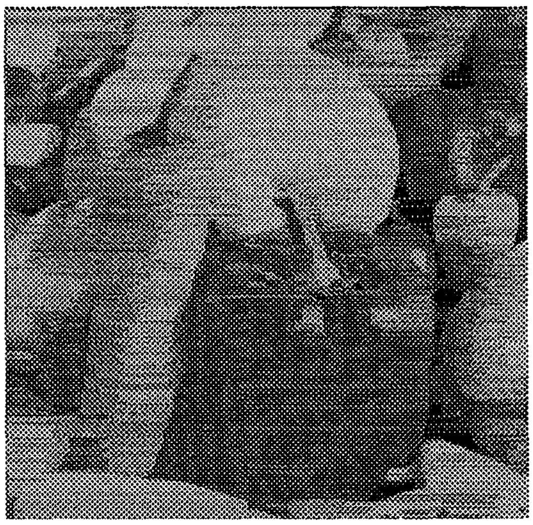
2. The gray levels of the pixels to the right of f(x, y) along the line (e. g, f(x+1, y), f(x+2, y), ..., etc) are then sequentially compared to the reference pixel intensity. If the intensity difference magnitude exceeds a threshold (T), a new reference value is stored and the run counter is reset to 1 (indicating a new run). Otherwise the run counter is incremented by 1 to extend the existing run. Candidate pixels are compared to the reference value, so that intensity functions with ramp characteristics will not be falsely encoded as constant intensity runs.

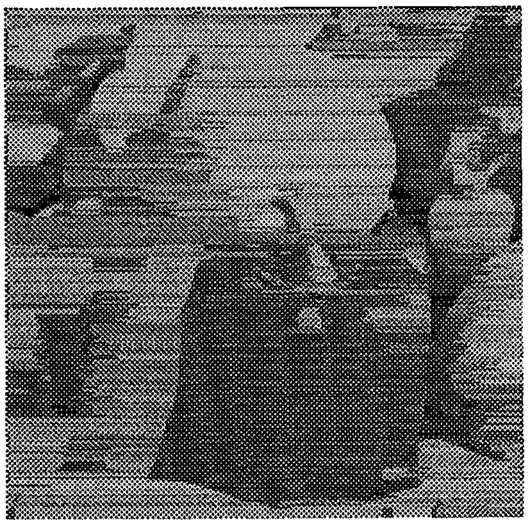
3. The process is repeated for all pixels of the image, thereby generating a data set consisting of an intensity and threshold-dependent set of pairs representing the encoded image of the form { f(x, y), rl} where rl is the length of the given run (in pixels) with intensity f(x,y).

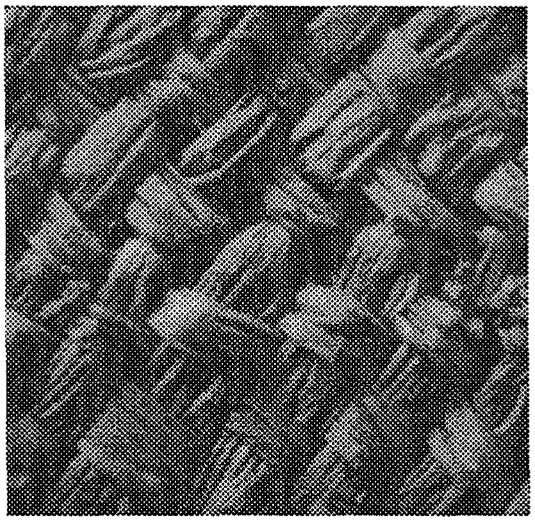
Since there are likely to be runs of length >> 1 in the compressed image representation, the resulting compressed data set will be smaller than the raw image data . These effects are highly dependent on the threshold T, which must be chosen to yield both a reasonable image representation (ie, the reconstructed image should appear subjectively 'close' to the original image) and a reasonable number of runs, each consisting of a reasonable number of pixels. Requirements on the threshold are conflicting; small threshold values tend to maximize the retained image representation while minimizing compression, whereas large thresholds produce significant compression while introducing significant distortion. Figure 3.1 shows an application of run-length compression at different compression ratios and also at different mean square errors.

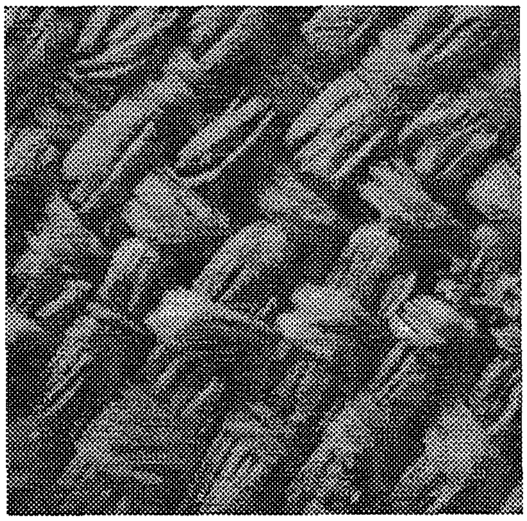
(a) Original 256x256 Peppers Image

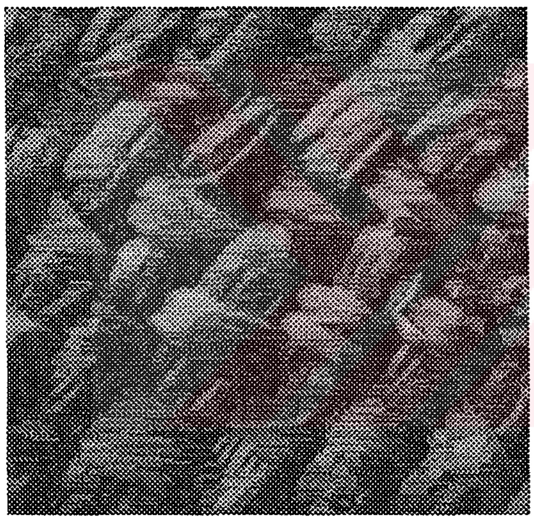

(b)Threshold= 25,MSE=128.63,SNR=27.03 dB, Comp= Ratio=1.49 bit/pixel.


(c) Threshold= 35,MSE=247.91, SNR= 24.18 dB, Comp Ratio=1.02 bit/pixel.


(d)Threshold=57,MSE=658.43, SNR=19.94 dB, Comp Ratio=0.50 bit/pixel.


(e) Original Image 2 With The Size Of 256x256


(f) Reconstructed Image with Threshold= 25, MSE=138.07, SNR=26.72 dB, Comp Ratio=2.69 bit/pixel.


(g) Reconstructed Image with Threshold= 40, MSE=352.42, SNR=22.66 dB, Comp Ratio=1.41 bit/pixel.

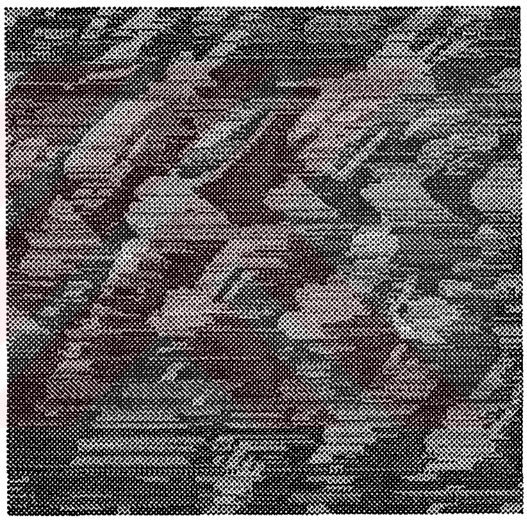

(h) Reconstructed Image with Threshold= 60, MSE=750.87, SNR=19.37 dB, Comp Ratio=0.70 bit/pixel.

Figure 3.1 An Application Of Run-Length Encoding And Compression

3.2 DISCRETE FOURIER TRANSFORM (DFT) AND COMPRESSION

Transform theory has played a major role in image processing for many years, and it continues to be a topic of interest in theoretical as well as applied work in this field. The transforms are usually used as the first step of an image compression technique especially the Discrete Fourier Transform. However, in my thesis I am going to show that DFT can be used as a compression technique [4].

Asume that f(x) be a continuous function of a real variable x. The Fourier transform of f(x) is defined by the equation

$$F\{f(x)\} = F(u) = \int_{-\infty}^{\infty} f(x)\exp[-j2\pi ux]dx \tag{3.1}$$

where j=$\sqrt{-1}$. By using inverse Fourier transform f(x) can be obtained;

$$F^{-1}\{F(u)\} = f(x) = \int_{-\infty}^{\infty} F(u)\exp[j2\pi ux]du \tag{3.2}$$

Equations 3.1 and 3.2 are called the Fourier transform pair. Here if f(x) is continuous, integrable and F(u) is integrable then these equations exist. These conditions are almost always exist in practice.

All f(x) values throughout my thesis are real, and f(x) represents the intensity function of an image. The Fourier transform of a real function is generally complex; that is,

$$F(u) = R(u) + jI(u) \tag{3.3}$$

where R(u) and I(u) are the real and imaginary components of F(u), respectively. It is convenient to express Eq 3.3 in exponential form, that is,

$$F(u) = |F(u)|.e^{j\varphi(u)} \tag{3.4}$$

where

$$|F(u)| = [R^2(u) + I^2(u)]^{1/2} \tag{3.5}$$

and

$$\varphi(u) = \tan^{-1}\left[\frac{I(u)}{R(u)}\right] \tag{3.6}$$

The magnitude function $|F(u)|$ is called the Fourier spectrum of f(x) and $\varphi(u)$ its phase. The square of the spectrum,

$$P(u) = |F(u)|^2 = R^2(u) + I^2(u) \qquad (3.7)$$

is commonly called power spectrum of f(x). The term spectral density is also used to represent the power spectrum.

The variable u which appears in the Fourier transform is often called the frequency variable. This name comes from expression of the exponential term, $\exp[-2\pi ujx]$ - using Euler's formula - in the form:

$$\exp[-j2\pi ux] = \cos(2\pi ux) - j\sin(2\pi ux) \qquad (3.8)$$

Interpreting the integral in Eq 3.1 as a limit summation of discrete terms makes evident that F(u) is composed of an infinite sum of sine and cosine terms. Each value of u determines the frequency of its corresponding sine-cosine pair.

Discrete form of Fourier transform is obtained by the following method;

Assume that a continuous function f(x) is discretized into a squence

$$\{f(x_0), f(x_0+\Delta x), f(x_0+2\Delta x), \ldots, f(x_0+[N-1]\Delta x)\} \qquad (3.9)$$

by taking N samples Δx units apart, as shown in Figure 3.2. It will be convenient to use x as either a discrete or continuous variable, depending on the contex of the discussion. To do so requires defining

$$f(x) = f(x_0 + x\Delta x) \qquad (3.10)$$

where x assumes the discrete values 0, 1, 2, 3, ..., N-1. In other words, the sequence $\{f(0), f(1), f(2), \ldots, f(N-1)\}$ represents any N uniformly spaced samples from a corresponding continuous function. With the above notation, the discrete Fourier transform pair that applies to sampled functions is given by

$$F(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)\exp[-j2\pi ux/N] \qquad (3.11)$$

for u=0,1,2,3, ..., N-1 and

$$f(x) = \sum_{u=0}^{N-1} F(u)\exp[j2\pi ux/N] \qquad (3.12)$$

for x=0, 1, 2, 3, ..., N-1.

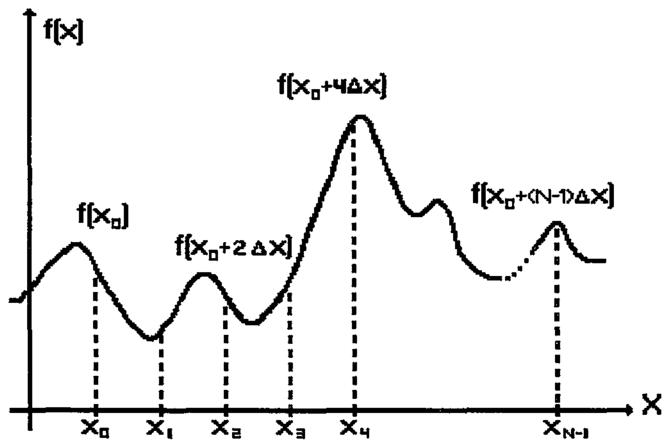


Figure 3.2 Sampling a Continuous Function

The values u=0, 1, 2, ..., N-1 in the discrete Fourier transform (Eq 3.11) correspond to samples of the continuous transform at values 0, Δu, 2Δu, ..., (N-1)Δu. In other words, F(u) represents F(uΔu).This definition is similar to that used for the discrete f(x), except that the samples of F(u) start at the origin of tyhe frequency axis. The terms Δu and Δx are given by the expression

$$\Delta u = \frac{1}{N\Delta x} \tag{3.13}$$

In the two dimensional case the discrete Fourier transform pair is

$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\exp\left[-j2\pi\left(ux/M + vy/N\right)\right] \tag{3.14}$$

for u=0, 1, 2, 3, ..., M-1, v=0, 1, 2, ..., N-1, and

$$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)\exp\left[j2\pi\left(ux/M + vy/N\right)\right] \tag{3.15}$$

for x=0, 1, 2, ..., M-1 and y=0, 1, 2, ..., N-1

Sampling of a continuous function is now in a two dimensional coordinate system with divisions of width Δx and Δy in the x and y axis, respectively. In the 1-D case the discrete function f(x,y) represents samples of the function $f(x_0 + x\Delta x, y_0 + y\Delta y)$ for x=0, 1, 2, ..., M-1 and y=0, 1, 2, ..., N-1. Similar technique is applied to F(u,v). The sampling increments in the spatial and frequency domains are given by

$$\Delta u = \frac{1}{M\Delta x} \tag{3.16}$$

and

$$\Delta v = \frac{1}{N\Delta y} \qquad (3.17)$$

When images are sampled in a square array, M=N and

$$F(u,v) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\exp[-j2\pi(ux+vy)/N] \qquad (3.18)$$

for u, v=0, 1, 2, ..., N-1 and

$$f(x,y) = \frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} F(u,v)\exp[j2\pi(ux+vy)/N] \qquad (3.19)$$

for x, y=0, 1, 2, ..., N-1. Note that 1/N term is in both Equation 3.18 and 3.19. Because F(u,v) and f(x,y) a Fourier transform pair, the grouping of these constant terms is arbitrary. In practice, images typically are digitized in square arrays, so I will be mostly use the Fourier transform pair in Equation 3.18 and 3.19.

3.2.1 Zig-zag order

In the encoding process of discrete Fourier transform first of all, image pixels are grouped into 8x8 blocks, and each block is transformed by the DFT formulation (Equation 3.18) into a set of 64 values referred to as DFT coefficients. The fist one of these values is referred to as the DC coefficient and the other 63 as the AC coefficients.Than these 64 coefficients are arranged in order by the zig-zag order rule which is shown in Figure 3.3. The DC coefficient is most important coefficient in a 8x8 block. Because it contains more image informations than the all other AC coefficients. The DC coefficient and following few AC coefficients are also contains the low frequency features of an image. The rest of the coefficients represent the high frequency features of an image. In the discrete Fourier transform compression technique some of the coefficients including the DC one are stored into memory. Since the rest of the coefficients are set to zero a compression ratio is obtained. The reconstructed image is obtained by using inverse discrete Fourier transform formulation (Eq 3.19). Since the some of the coefficients are set to zero in the copression algorithm the reconstructed image will be noisy. The encoder and the decoder algorithms are given by the block diagrams in Figure 3.4 and Figure 3.5 respectively.

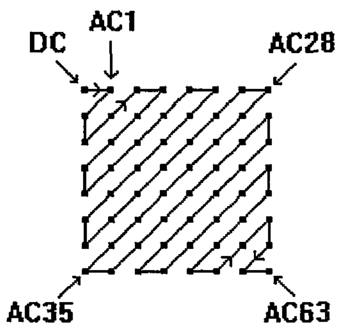


Figure 3.3 Zig-Zag Order

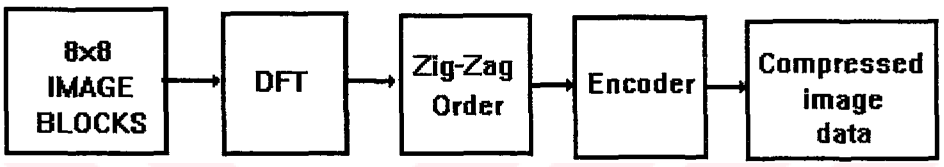


Figure 3.4 DFT - Based Compression Block Diagram

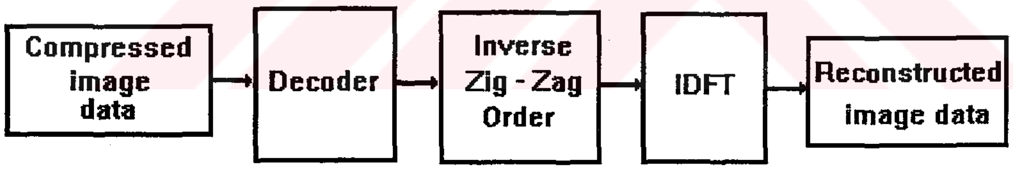
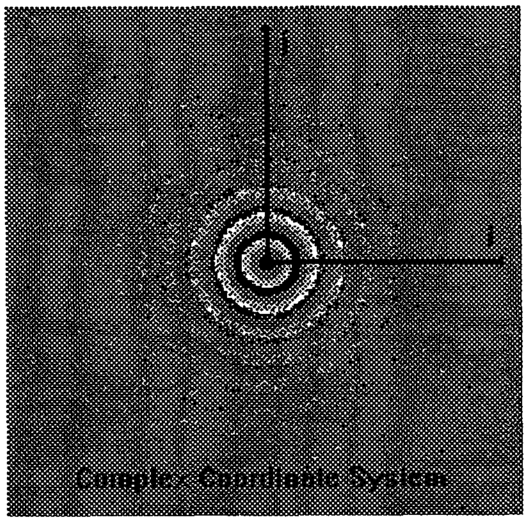


Figure 3.5 DFT - Based Decoder Block Diagram

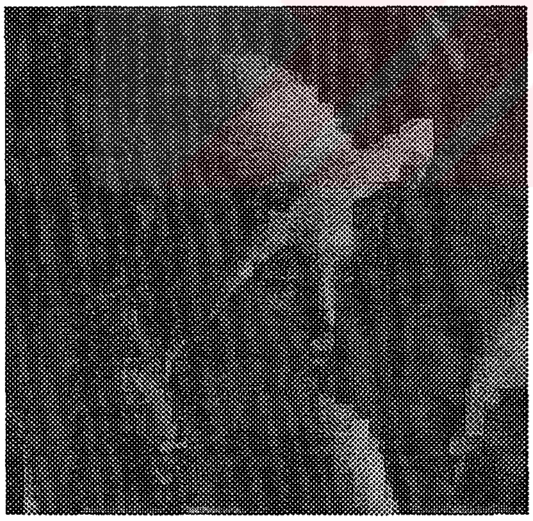
In the compression algorithm only the phase or the magnitude of the coefficients are stored into the memory. If the number of stored coefficient is n then the compression ratio= n/8 bit/pixel. Some examples of this technique are given in Figure 3.6.

3.2.2 Applications of DFT based compression technique

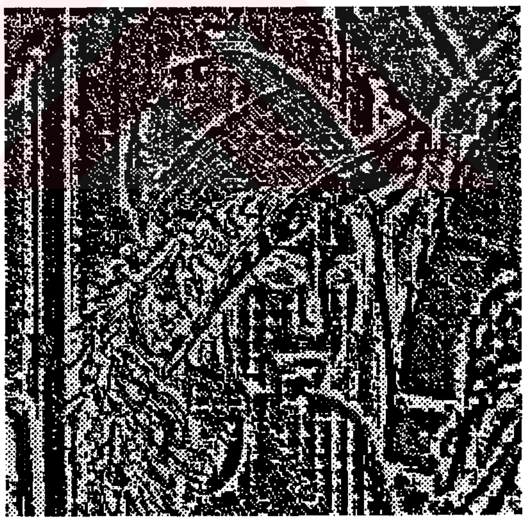
(a) 256x256 Original Lena Image



(b) Displaying $|F(u,v)|$. The brighhter the pixels the bigger the $|F(u,v)|$ values

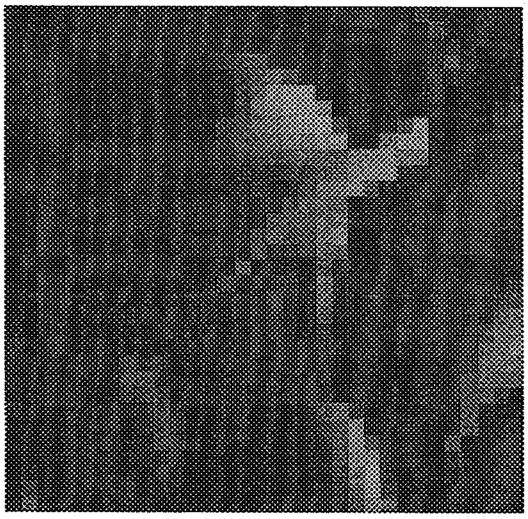


(c) Lowpass Filtered Image (IDFT) with Using the Coefficients Between 1and 11in the Zig-Zag Order

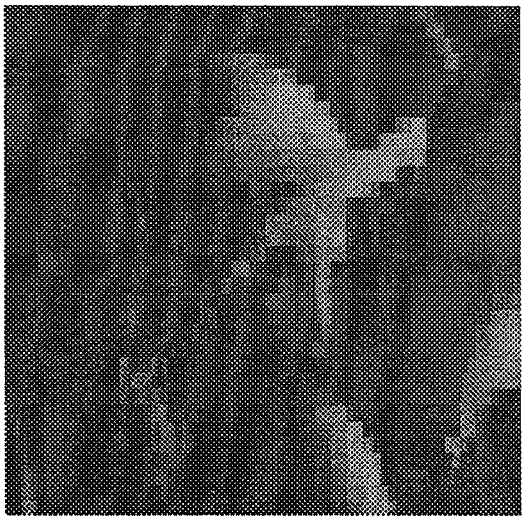


(d) Highpass Filtered Image (IDFT) with Using the Coefficients Between 20and 64 in the Zig-Zag Order

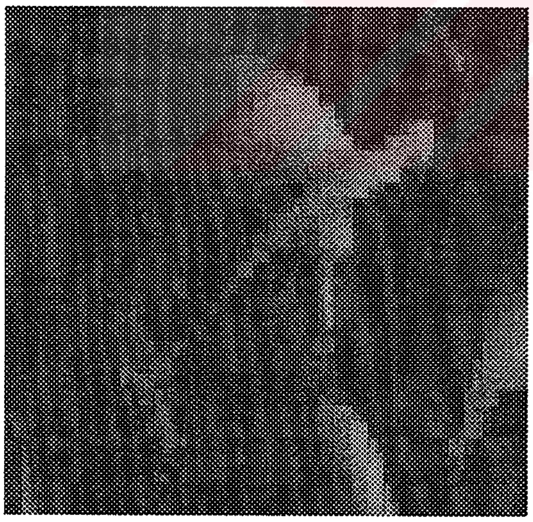
Figure 3.6 Displaying The Different Features Of DFT



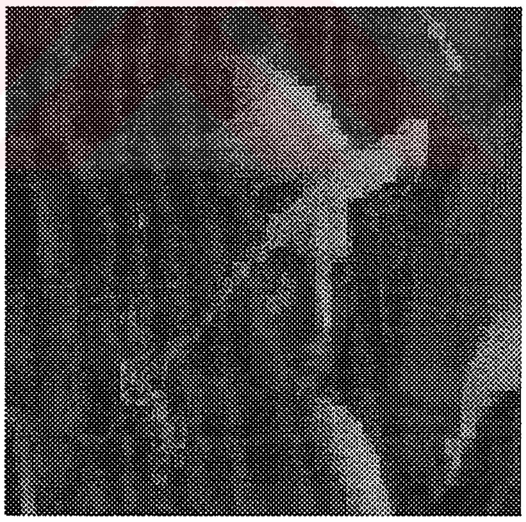
(a) MSE=281.68, SNR=23.63 dB,
Comp. Ratiio=0.125 bit/pixel



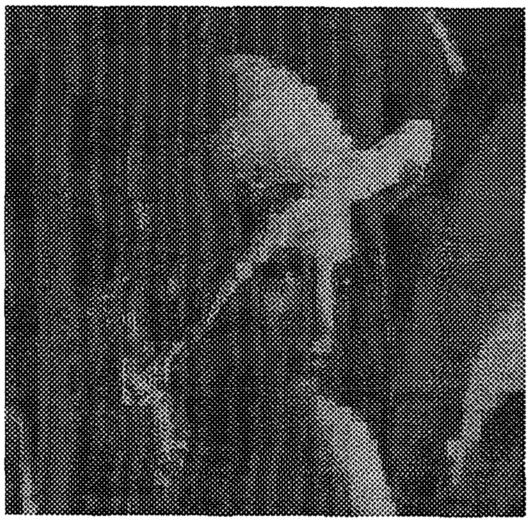
(b) MSE=205.59, SNR=25.00 dB,
Comp. Ratio=0.25 bit/pixel



(c) MSE=173.54, SNR=25.74 dB,
Comp. Ratio=0.5 bit/pixel



(d) MSE=127.04, SNR=27.09 dB,
Comp. Ratio=1.0 bit/pixel


(e) MSE=104.45, SNR=27.94 dB,
Comp. Ratio=2.0 bit/pixel.

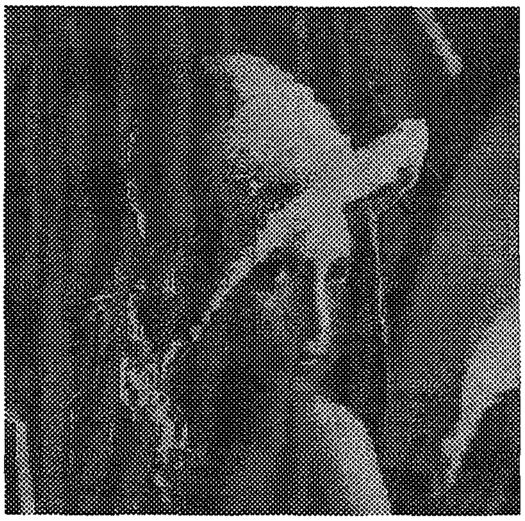

(f) MSE=54.65, SNR=30.75 dB,
Comp. Ratio=4.0 bit/pixel.

Figure 3.7 Reconstructed Images at Different Compression Ratios

3.3 DISCRETE COSINE TRANSFORM (DCT) AND COMPRESSION

Discrete cosine transform which is quite similar to DFT is used as a compression technique. This algorithm begins with blocking the image into 8x8 pixel size. After finding 64 coefficients by the help of DCT formulations the Zig-Zag order rule is applied. Then some of the coefficients are stored into memory while the rest of them are set to zero. As we use a few of coefficients in the reconstruction algorithm a compration ratio is achieved. DCT of a pixel is a real number, but DFT of a pixel is a complex number. Because of this reason DCT has less calculation complexity than the DFT. So DCT is more popular than DFT [4].

The 1-D discrete cosine transform (DCT) is defined as

$$F(u)=\sum_{k=0}^{N-1}2f(k)\cos\left[\frac{u\pi(2k+1)}{2N}\right] \tag{3.20}$$

for u=0, 1, 2, ..., N-1. Similarly, the inverse DCT is defined as

$$f(k)=\frac{1}{N}\sum_{u=0}^{N-1}w(u)F(u)\cos\left[\frac{u\pi(2k+1)}{2N}\right] \tag{3.21}$$

for k=0, 1, 2, ..., N-1. In both Equation 3.20 and 3.21, w(u) is

$$w(u) = \begin{cases} 1/2 & \textit{for} \quad u = 0 \\ 1 & \textit{for} \quad u = 1, 2, \ldots, N-1 \end{cases} \tag{3.22}$$

The corresponding 2-D DCT pair is

$$F(k,l) = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} 4 f(m,n) \cos\left[\frac{\pi k(2m+1)}{2N}\right] \cos\left[\frac{\pi l(2n+1)}{2N}\right] \tag{3.23}$$

for k, l=0, 1, 2, 3, ..., N-1, and

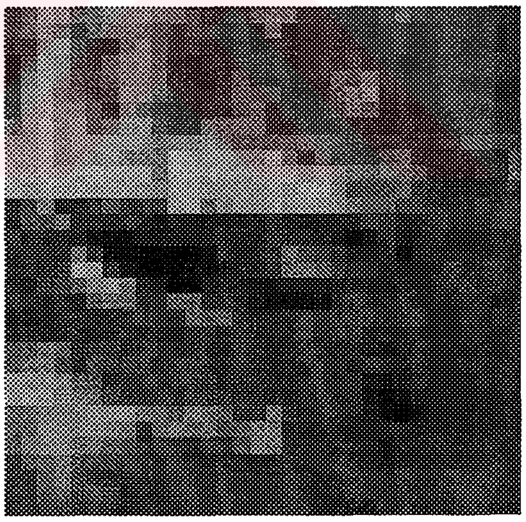
$$f(m,n) = \frac{1}{N^2}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} w_1(k) w_2(l) F(k,l) \cos\left[\frac{\pi k(2m+1)}{2N}\right] \cos\left[\frac{\pi l(2n+1)}{2N}\right] \tag{3.24}$$

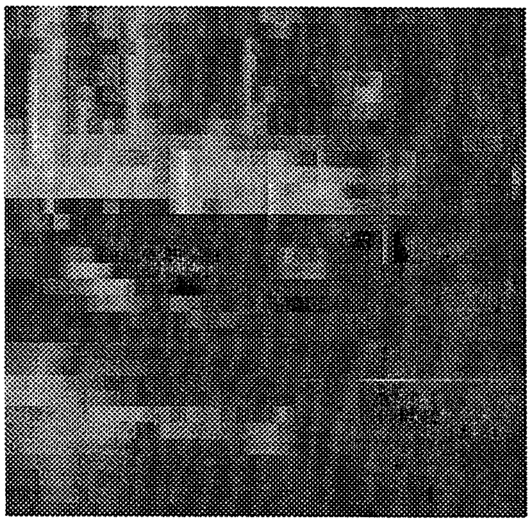
for m, n=0, 1, 2, 3, ..., N-1.

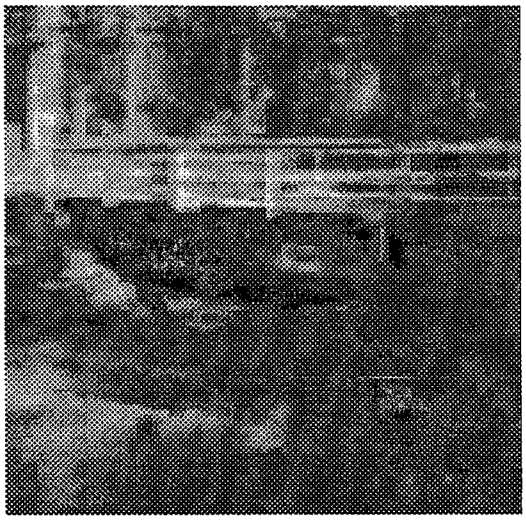
$$w_1(k) = \begin{cases} 1/2 & \textit{for} \quad k = 0 \\ 1 & \textit{for} \quad k = 1, 2, \ldots, N-1 \end{cases} \qquad w_2(l) = \begin{cases} 1/2 & \textit{for} \quad l = 0 \\ 1 & \textit{for} \quad l = 1, 2, \ldots, N-1 \end{cases} \tag{3.25}$$

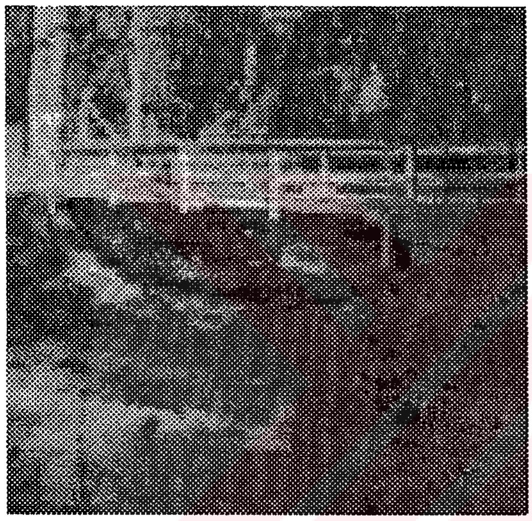
In recent years the discrete cosine transform has become the method of choice for image data compression. Figure 3.8 shows some compression results which were done by DCT technique.

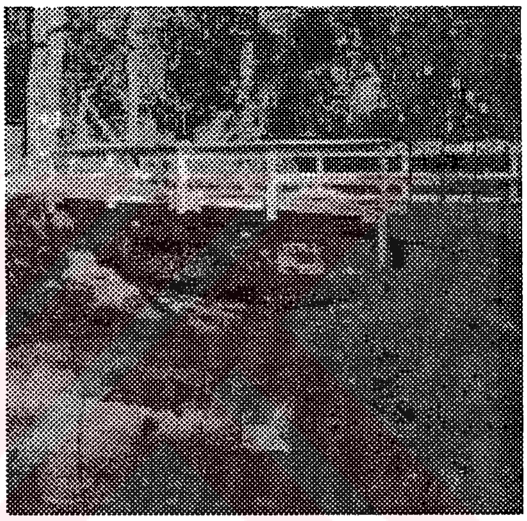
(a) Original 256x256 "View" Image

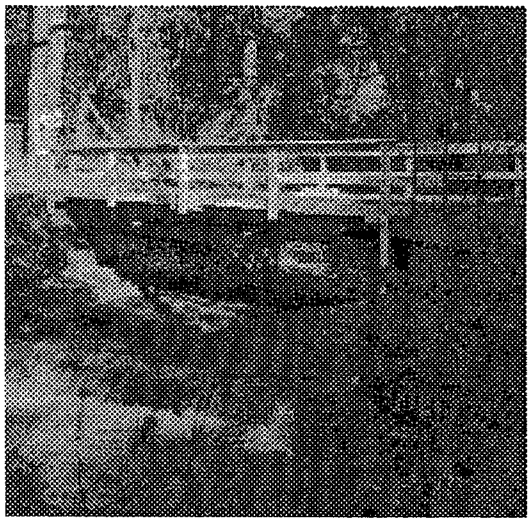

(b) MSE=895.50, SNR=18.61 dB
Comp. Ratio=0.125 bit/pixel.


(c) MSE=778,SNR=19.2dB,Comp=0.25b/p


(d) MSE=507,SNR=21.07 dB,Comp=0.5b/p


(e) MSE=395.2,SNR=22.2 dB,Comp=1b/p


(f) MSE=254.1,SNR=24.0 dB,Comp=2b/p


(g) MSE=129.9, SNR=27.0dB,Comp=4 b/p

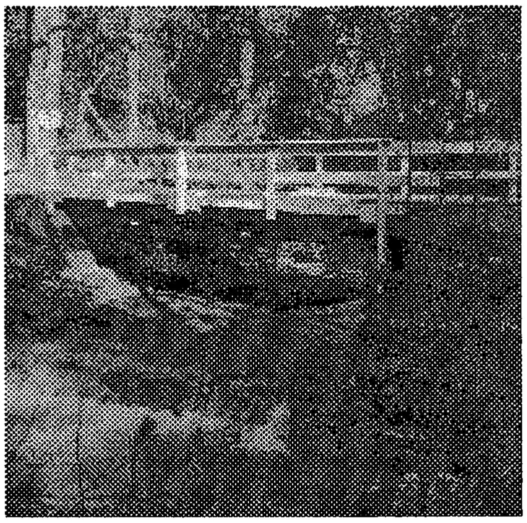

(h) MSE=57.7,SNR=30.5 dB,Comp=6 b/p

Figure 3.8 Reconstructed Images at Different Compression Ratios

3.4 VECTOR QUANTIZATION (VQ)

Image data compression using vector quantization (VQ) has received a lot of attention in the last ten years because of its simplicity and adaptability. VQ requires the input image to be processed as vectors or blocks of image pixels. The encoder takes a vector and finds the best or closest match, based on some distortion criterion, from its stored codebook. The address of the best match is then transmitted to the decoder. The decoder takes this address number and finds the corresponding vector from its codebook. The reconstructed image is obtained. Data compression is achieved in this process because the transmission of the address requires fewer bits than transmitting the vector itself [9],[5].

The performance of encoding and decoding by VQ is dependent on the available codebook and the distribution of the image data relative to it. Hence, the design of an efficient and robust codebook is very important in VQ. Linde, Buzo and Gray first suggested a practical suboptimal clustering analysis algorithm, known as the LBG algorithm to generate a codebook based on some training set. It is said that the algorithm only guarantees a locally optimum codebook relative to the source data used (the training set). The simulated annealing (SA) method of generating a codebook tries to obtain a global optimum by a stochastic relaxation technique. Another algorithm, called deterministic annealing uses a fuzzy clustering and leads to a global optimum.

The size of codebook and the vector dimension also play a major role in determining the overall performance. From Shannon's rate distortion theory we know that the larger the vector dimension, the better the potential performance. However, with increased vector dimension the required codebook size also increases, and the result is an exponential increase in encoding complexity. So for practical limitations one is forced to work with low-dimensionality VQ with lesser quality, despite the fact that better VQ performance is theoretically possible. The increase in codebook size also introduces the problem of empty cells in the generated codebook. Empty cells makes the codeword zero.

3.4.1 Codebook design by LBG algorithm

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,

$$Q : R^k \to Y \tag{3.26}$$

where $Y = (\hat{x}_i \; ; i = 1, 2, 3, \ldots, N)$ is the set of reproduction vectors and N the number of vectors in Y. It can also be seen as a combination of two functions : an encoder, which views the input vector x and produces the address of the reproduction vector specified by Q(x), and a decoder, which uses this address to produce the reproduction vector $\hat{x}$. If a distortion measure $d(x, \hat{x})$ which represents the penalty or cost associated with reproducing vectors x by $\hat{x}$ is defined, then the best mapping Q is the one which minimizes the distortion. The LBG algorithm and other variations of this algorithm are based upon this minimization, using a training data set as the signal [15],[10].

One simple distortion measure for waveform coding is the square error distortion given by

$$d(x, \hat{x}) = \|x - \hat{x}\|^2 = \sum_{j=0}^{K-1} (x_j - \hat{x}_j)^2 \tag{3.27}$$

K is the number of x vectors.

The goal in designing a vector quantizer is to obtain a quantizer consisting of N reproduction vectors, such that it minimizes the expected distortion. If there is no other quantizer that can achieve the minimum expected distortion this condition can be provided. Lloyd proposed an iterative nonvariational method known as his “Method 1” for the design of scalar quantizers. Recently, Linde, Buzo and Gray extended Lloyds’ basic approach to the general case of vector quantizers.

Let the expected distortion be approximated by the expression

$$D(x, q(x)) = \frac{1}{N} \sum_{i=0}^{N-1} d(x_i, \hat{x}_i) \tag{3.28}$$

The LBG algorithm for a known distribution training sequence follows these rules :

1) Let N=number of levels ; distortion threshold ε ≥ 0. Assume an initial N level reproduction alphabet $\hat{A}_0$, and a training sequence $\left(x_j \, ; j = 0, 1, 2, \ldots, n-1\right)$, and m= number of iterations, set to zero.

2) Given $\hat{A}_m = \left(y_i \, ; i = 1, 2, \ldots, N\right)$, find the minimum distortion partition $P\left(\hat{A}_m\right) = \left(S_i \, ; i = 1, \ldots, N\right)$ of the training sequence : $x_j \in S_i$ if $d\left(x_j, y_i\right) \le d\left(x_j, y_l\right)$ for all l. Compute the average distortion.

$$D_m = D\left[\left(\hat{A}_m, P\left(\hat{A}_m\right)\right)\right] = n - 1\sum_{j=0}^{n-1} \min{}_{y \in A_m} d\left(x_j, y\right) \tag{3.29}$$

3) If $\left(D_{m-1} - D_m\right) / D_m \le \varepsilon$, stop the iteration with $\hat{A}_m$ as the final reproduction alphabet; otherwise continue.

4) Find the optimal reproduction alphabet $\hat{x}\left(P\left(\hat{A}_m\right)\right) = \left(\hat{x}\left(S_i\right) \, ; i = 1, 2, \ldots, N\right)$ for $P\left(\hat{A}_m\right)$ where

$$\hat{x}\left(S_i\right) = \frac{1}{\left\| S_i \right\|} \sum_{j : x_j \in S_i}^{m} x_j \tag{3.30}$$

5) Set $\hat{A}_{m+1} = \hat{x}\left(S_i\right)$, increment m to m+1, and go to (2).

In the above iterative algorithm an initial reproduction alphabet $\hat{A}_0$ was assumed in order to begin the algorithm. There are several techniques to obtain the initial codebook. The simplest technique is to use the first widely spaced words from the training data. Linde, Buzo and Gray used a splitting technique where the centroid for the training sequence was calculated and split into two close vectors. The centroids or the reproduction vectors for the two partitions were then found. Each resulting vector was then split into two vectors and the above procedure was repeated until an N level initial reproduction vector was obtained.

Splitting was performed by adding a fixed perturbation vector ε to each vector y_i generating two vectors $y_i + \varepsilon, y_i - \varepsilon$.

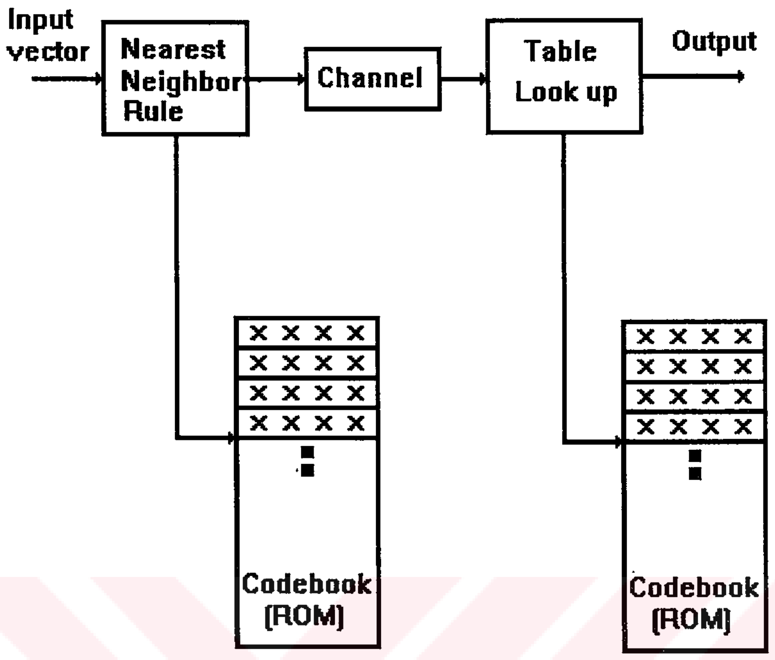


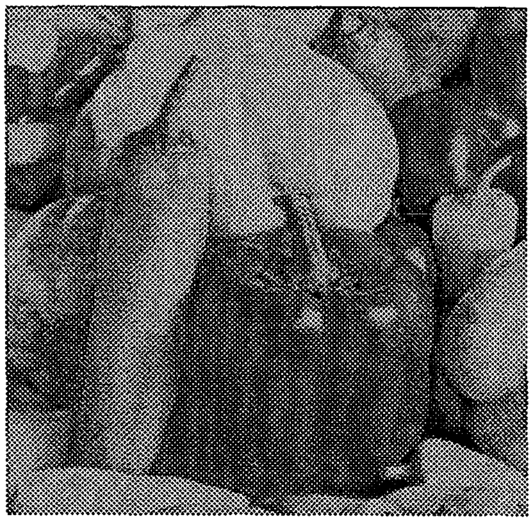
Figure 3.9 Block Diagram Of Simple VQ

3.4.2 Appication

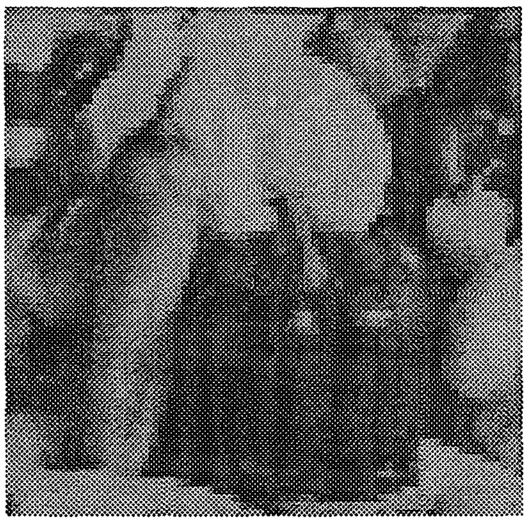
The vectors that represent an image can be selected with different sizes. However, increasing the vector sizes also adds computational complexity to our work. So the most common vector sizes are selected 1x2, 2x2, 4x4 and 16x16 pixel dimensions respectively. Below, I gave some original images and their vector quantized images with MSE (mean square error), bit rates and SNR (signal to noise ratio) at different vector sizes. The MSE and SNR formulations are given in Eq. 3.31 and Eq 3.32. Here, N is the number of pixels in the image.

$$MSE = \frac{1}{N}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left(x_{ij} - \hat{x}_{ij}\right)^2 \qquad (3.31)$$

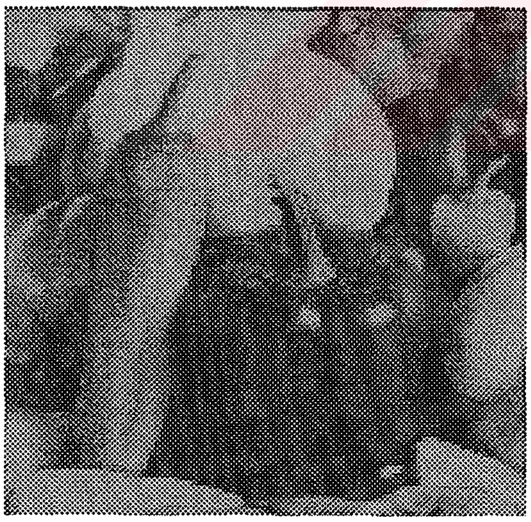
$$SNR = -10\log\frac{MSE}{255^2}\ dB \qquad (3.32)$$



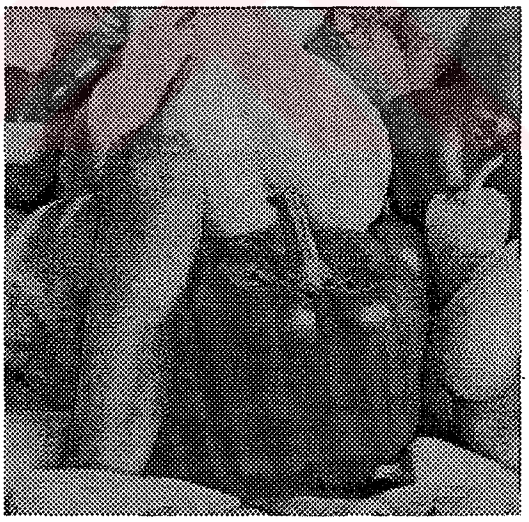
(a) Original 256x256 Peppers Image



(b) Block Size= 4x4 pixels, Codebook Size=128, MSE=326.3, SNR=23.0 dB, Compression=0.437 bit/pixel

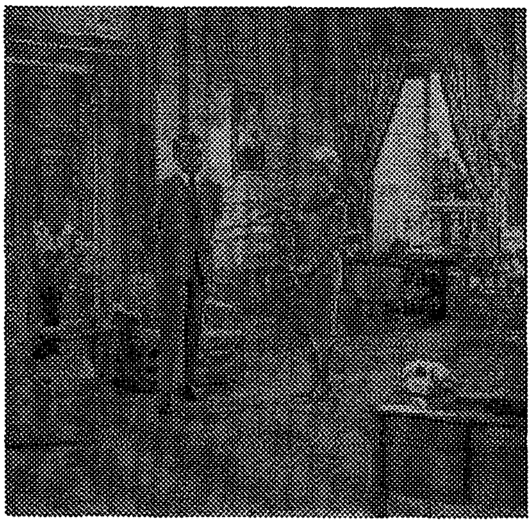


(c) Block Size= 2x2 pixels, Codebook Size=128, MSE=278.4, SNR=23.68 dB, Compression=1.75 bit/pixel

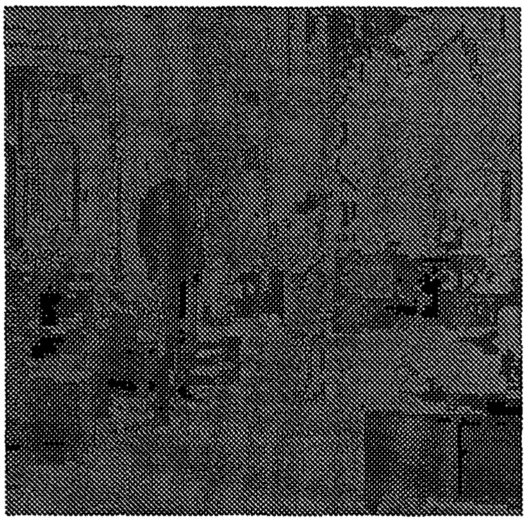


(d) Block Size= 1x2 pixels, Codebook Size=128, MSE=7.21, SNR=39.54 dB, Compression=3.5 bit/pixel

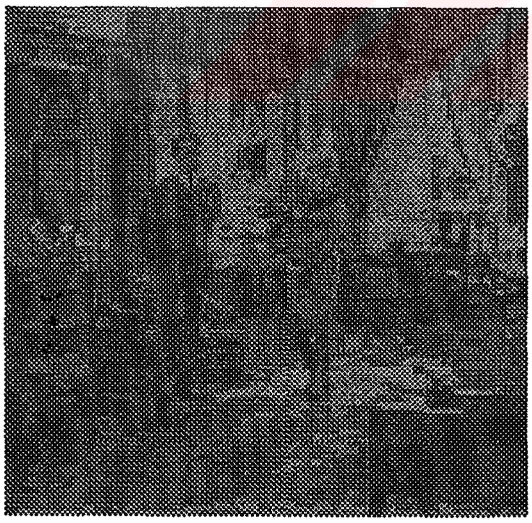
Figure 3.10 VQ Results at Different Block Sizes



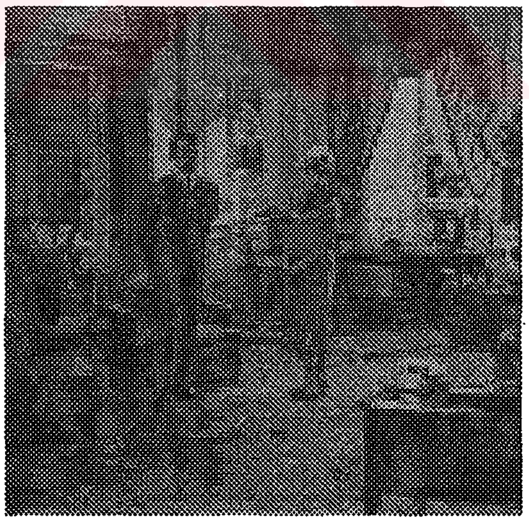
(a) Original 256x256 Room Image



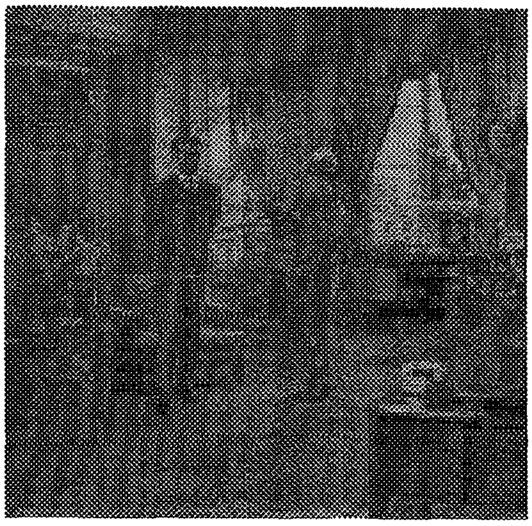
(b) Block Size= 2x2 pixels, Codebook Size=2, MSE=448.38, SNR=21.61 dB, Compression=0.250 bit/pixel



(c) Block Size= 2x2 pixels, Codebook Size=4, MSE=196.38, SNR=25.120 dB, Compression=0.5 bit/pixel



(d) Block Size= 2x2 pixels, Codebook Size=8, MSE=109.790, SNR=27.72 dB, Compression=0.75 bit/pixel


(e) Block Size= 2x2 pixels, Codebook Size=16, MSE=82.04, SNR=29.0 dB, Compression=1.0 bit/pixel

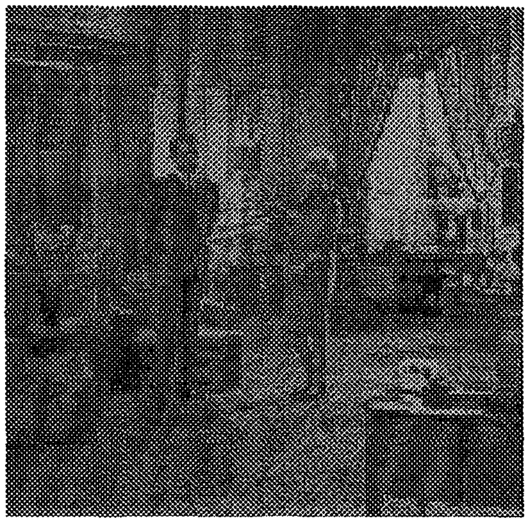

(f) Block Size= 2x2 pixels, Codebook Size=64, MSE=69.92, SNR=29.68 dB, Compression=1.5 bit/pixel

Figure 3.11 VQ Results at Different Codebook Size

Table 3.1 Codebook of the Image in Figure 3.11 (e).

Number	Codeword (x_1, x_2, x_3, x_4)				Number	Codeword (x_1, x_2, x_3, x_4)			
[1]	44	45	43	43	[9]	58	55	64	61
[2]	158	165	157	164	[10]	154	150	156	152
[3]	94	91	97	93	[11]	100	119	82	102
[4]	126	129	127	130	[12]	139	123	131	114
[5]	72	76	65	69	[13]	80	76	83	79
[6]	179	173	176	175	[14]	194	194	193	193
[7]	108	103	111	107	[15]	116	119	116	119
[8]	147	145	144	143	[16]	135	137	137	139

Table 3.2 Codebook of the Image in Figure 3.10 (d).

N	Codeword		N	Codeword		N	Codeword		N	Codeword	
[1]	166	179	[33]	183	191	[65]	109	131	[97]	87	93
[2]	174	176	[34]	185	186	[66]	110	146	[98]	82	88
[3]	172	171	[35]	183	182	[67]	122	129	[99]	91	76
[4]	166	170	[36]	188	184	[68]	126	126	[100]	93	85
[5]	175	161	[37]	180	178	[69]	117	117	[101]	70	91
[6]	188	162	[38]	180	173	[70]	115	122	[102]	60	86
[7]	191	150	[39]	175	183	[71]	125	120	[103]	80	104
[8]	195	129	[40]	180	183	[72]	121	121	[104]	64	111
[9]	157	159	[41]	190	190	[73]	149	96	[105]	108	75
[10]	158	167	[42]	188	189	[74]	150	115	[106]	128	79
[11]	167	164	[43]	205	179	[75]	175	96	[107]	138	44
[12]	163	160	[44]	193	178	[76]	180	64	[108]	109	47
[13]	124	192	[45]	195	197	[77]	132	127	[109]	72	73
[14]	103	177	[46]	196	194	[78]	136	123	[110]	78	80
[15]	157	193	[47]	192	195	[79]	129	132	[111]	87	51
[16]	145	178	[48]	193	192	[80]	134	136	[112]	83	65
[17]	161	134	[49]	211	209	[81]	127	104	[113]	38	45
[18]	173	146	[50]	209	212	[82]	126	114	[114]	42	51
[19]	160	151	[51]	208	205	[83]	114	104	[115]	41	37
[20]	155	153	[52]	207	208	[84]	117	110	[116]	49	35
[21]	150	153	[53]	219	220	[85]	109	114	[117]	27	21
[22]	151	159	[54]	218	216	[86]	114	113	[118]	22	28
[23]	151	146	[55]	213	216	[87]	107	108	[119]	39	32
[24]	146	149	[56]	215	212	[88]	100	108	[120]	32	30
[25]	143	131	[57]	200	200	[89]	68	153	[121]	41	91
[26]	141	131	[58]	200	197	[90]	50	132	[122]	44	71
[27]	141	141	[59]	196	200	[91]	87	128	[123]	64	68
[28]	143	145	[60]	198	199	[92]	99	117	[124]	59	61
[29]	135	143	[61]	204	201	[93]	103	99	[125]	69	54
[30]	125	145	[62]	201	202	[94]	109	98	[126]	67	39
[31]	132	163	[63]	202	206	[95]	95	102	[127]	56	51
[32]	140	155	[64]	205	204	[96]	95	94	[128]	48	53

Table 3.3 Codebook of the Image in Figure 3.11 (b).

Number	Codeword (x_1, x_2, x_3, x_4)			
[1]	84	83	84	83
[2]	139	139	139	139

3.5 HIERARCHICAL FINITE STATE VECTOR QUANTIZATION (HFSVQ)

The hierarchical finite state vector quantization is a quite new technique which is developed on the basis of different VQ techniques. In the HFSVQ, an original image is divided to blocks of different sizes, which are then assigned into different layers, according to their gray-scale contrast. The blocks of low contrast which are located in a smooth region of the image will have large sizes and be assigned into higher layers. Fewer codewords can be used in the higher layers since there are strong correlations between the pixels in the smooth regions. The blocks located in edge regions where the gray scales vary dramatically from pixel to pixel need large number of codewords to represent all of the different block types [18].

There are two parts in HFSVQ. One part consist of the structure codes which provide the data of layer assignment of the image blocks. The other part consists of the local address codes of the codewords in the codebook. It is proved that this coding scheme is more efficient than VQ and by using HFSVQ we achieve a further bit rate reduction especially on the biomedical images.

3.5.1 Structure map construction

Any gray level image can be divided into several regions according to its gray scale contrast. The regions with low contrast correspond to the smooth area of the image and the regions which have edges with different sharpness demonstrate relatively high gray scale contrast. In this work, I used four types of layers. The higher layers have the blocks in smooth regions and the lower ones have the blocks in fluctuating regions.

Fist, we decompose the whole image into a group of blocks of size 16x16. If the gray scale contrast in a block is lower than a given threshold, the block is assigned into layer 1 (L_1), which collects the blocks located in the smoothest regions of the image. Then we decompose the rest of the blocks into subblocks of size 8x8. The same threshold is used and blocks located in a fairly smooth regions are assigned into layer 2 (L_2). Finally, we furter decompose the remainder into blocks of size 4x4 and assign them to the edge layer, layer 3 (L_3). In order to obtain more perceptual result we need to provide a more accurate edge reconstruction. Therefore a special layer for edges, layer 4 (L_4) is constructed from layer 3. Layer 4 collects the blocks with large gray scale transition by using a threshold with higher value. Here, the gray scale contrast is calculated by using horizontal gradyent and perpendicular gradyent. The horizontal gradyent gives us the average difference between the horizontal pixels in a block. The perpendicular gradyent gives us the average difference between the perpendicular pixels in a block. If the both gradyents in a block are smaller than the threshold, then that block is used in the smooth layer.

If our image size is 512x512, then the procedure above is used. Otherwise, for example, if the image size is 256x256 then the corresponding block sizes must be 8x8 for (L_1), 4x4 for (L_2) and 2x2 for (L_3) and (L_4).

After each layer has its own blocksi, then the LBG rule is applied to the layers to find the codebook for each layer. 4x4 codeword size is used in all LBG trainings. There are some advantages for this subcodebook generation strategy :

1) Only layer members are involved in training for generation of the codebook. Both the iteration times and the number of comparisons in each iteration are considerably reduced. As a result, the total training time can be shortened.

2) By applying layer assignment, we can adjust the size of each codebook according to the accuracy requirement for reconstruction of the different regions of an image by either choosing the threshold values or changing the sizes of the codebook for specific layers.

In the application 8 codewords for layer1 and layer 2, 32 codewords for layer 3, 128 codewords for layer 4 are found by the LBG algorithm. This means, we need only 3 bits to represent the layer 1 and layer 2, 5 bits for the layer 3 and 7 bits for the layer 4. So a compression is achieved. Compression result are stored into the computer memory by the help of a structure tree shown in Figure 3.12. The assignment result can be demonstrated as a structure map shown in Figure 3.13. In Figure 3.13 there are zeros and ones which are called structure codes. Structure codes are needed to record the structure map which is based on the branch distribution of the structure tree. The symbol "*" in Figure 3.13 indicates that there is a codeword address code inserted between the structure codes. In the reconstruction algorithm the structure codes are decoded again by the help of the structure tree.

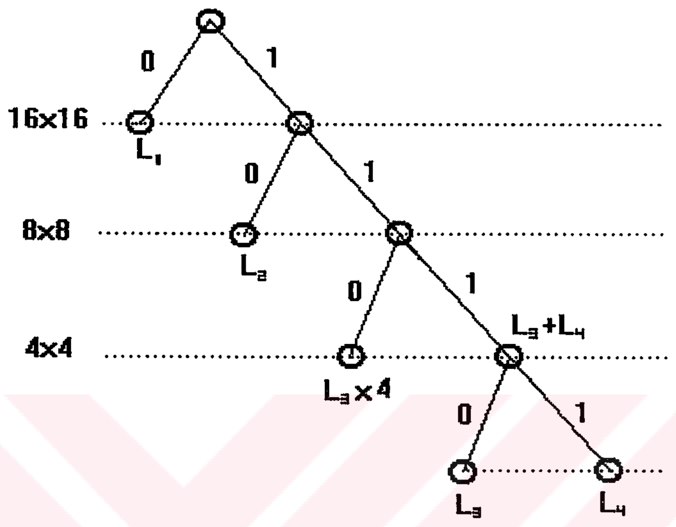


Figure 3.12 Structure Tree

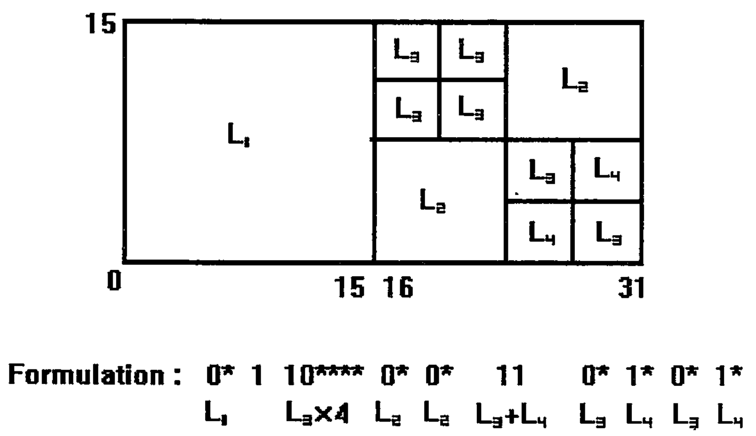
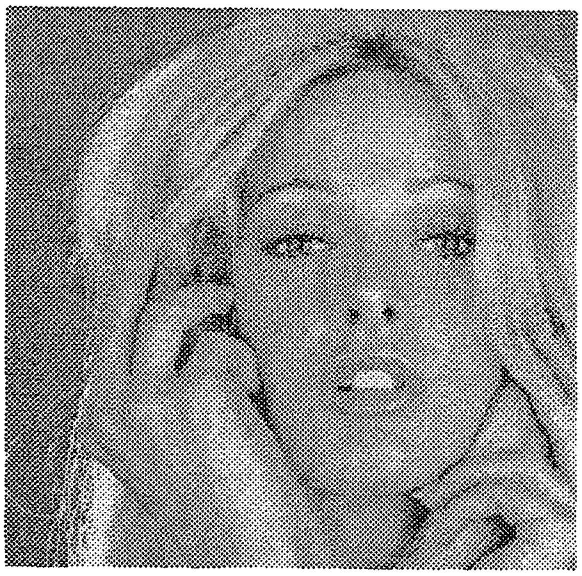
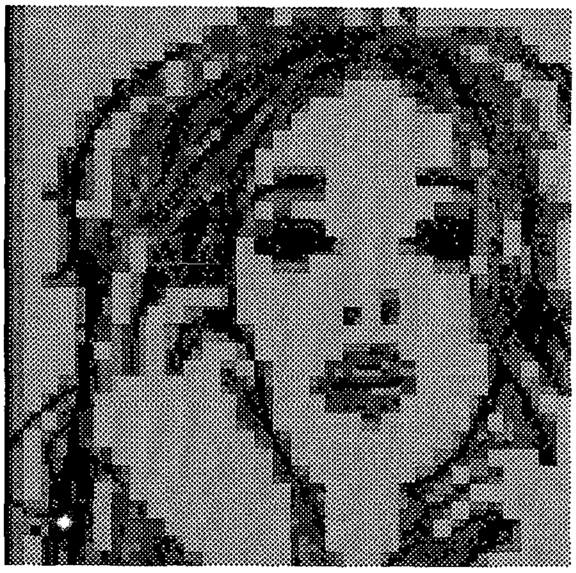
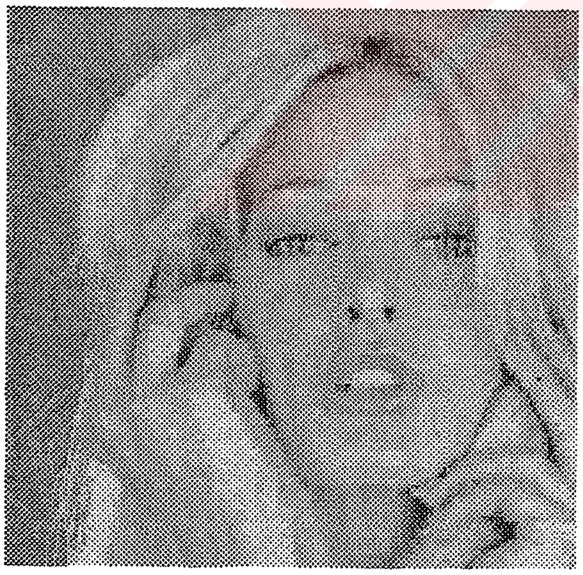

15
0
15 16
31
L₁
L₂
L₃
L₄
Formulation : 0* 1 10**** 0* 0* 11 0* 1* 0* 1*
L₁ L₃x4 L₂ L₂ L₃+L₄ L₃ L₄ L₃ L₄

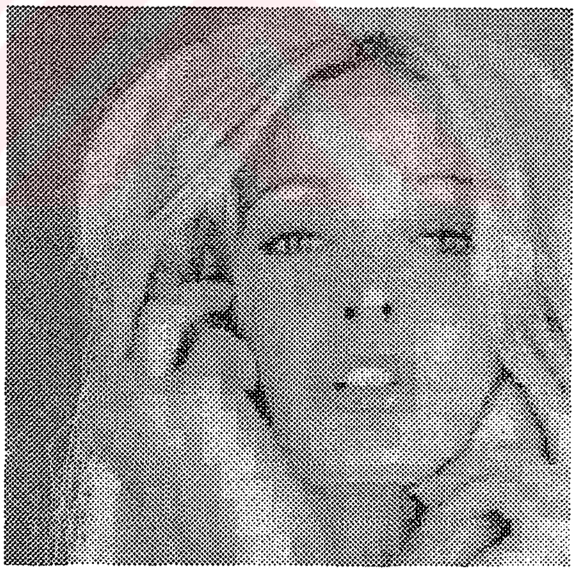
Figure 3.13 Structure Map

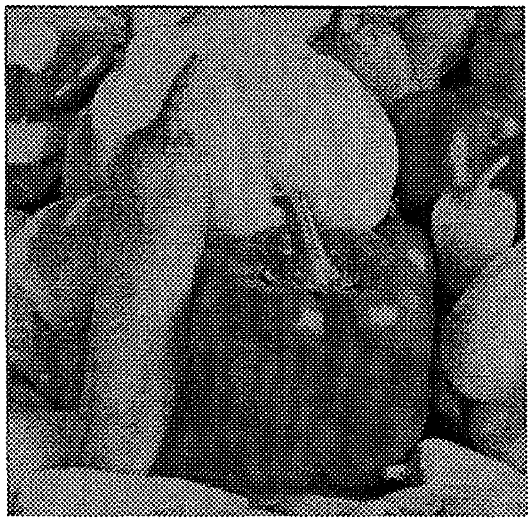
3.5.2 Application


(a) Original 512x512 "Tiffany" Image

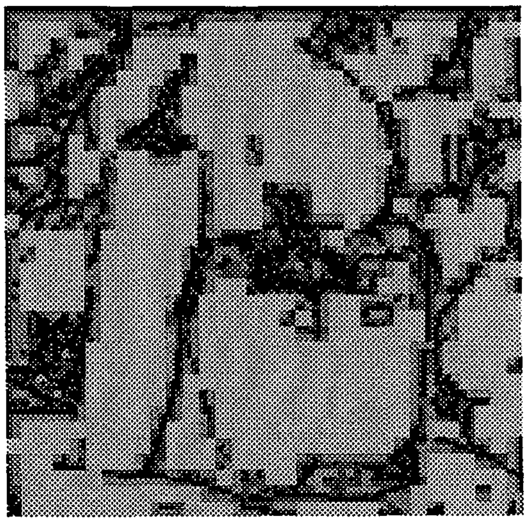

(b) Structure Map At Threshold 1=4.5, Threshold 2=10.0


(c) Threshold 1 = 2.5, Threshold 2 = 20.0, MSE= 61.15, SNR=30.26 dB,Compression Ratio=0.270 bit/pixel

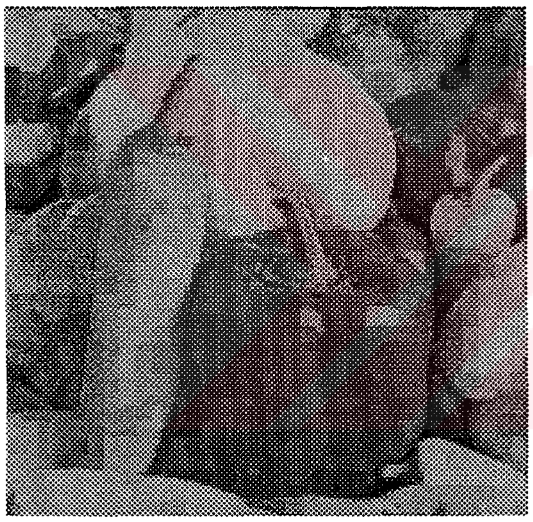

(d) Threshold 1 = 5.0, Threshold 2 = 40.0, MSE= 81.91, SNR=29.00 dB,Compression Ratio=0.143 bit/pixel



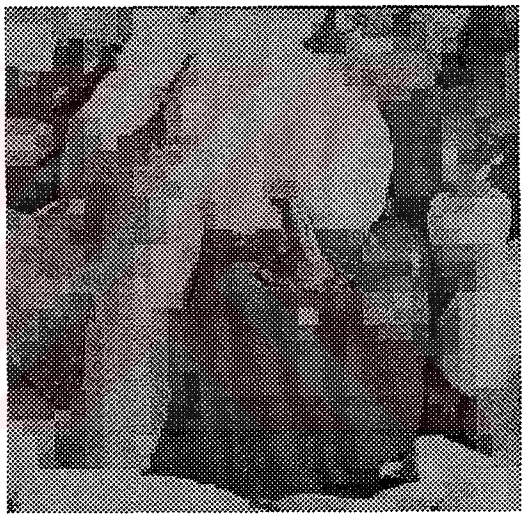
(e) Original 256x256 Peppers Image



(f) Structure Map At Threshold 1=6.0, Threshold 2=11.0



(g) Threshold 1 = 7.0, Threshold 2 = 30.0, MSE= 95.69, SNR=28.32 dB,Compression Ratio=0.5 bit/pixel



(h) Threshold 1 = 13.0, Threshold 2 = 55.0, MSE= 175.84, SNR=25.68 dB,Compression Ratio=0.250 bit/pixel

Figure 3.14 Reconstructed Images of HFSVQ Algorithm

3.6 CONCLUSION

A comparison of compression techniques according to MSE and bit rate was given in Table 3.5, Table 3.6, Table 3.7, Table 3.8.

Table 3.4 Histogram informations of some original images

NAME	SIZE	HISTOGRAM INFORMATION
PEPPERS	256x256 pixel	High Contrast Image
BABOON	256x256 pixel	High Contrast Image
LENA	256x256 pixel	Low Contrast Image
TIFFANY	512x512 pixel	Bright Image

Table 3.5 MSE results of the reconstructed images at 2 b/p compression.

Comp. Ratio 2 bit/pixel	RUN LENGTH	DFT	DCT	VQ	HFSVQ
PEPPERS	74.98	127.84	36.13	255.60	145.63
BABOON	304.14	235.55	187.99	183.47	134.45
LENA	85.94	104.45	43.02	79.59	108.04
TIFFANY	39.64	48.87	23.31	74.32	74.32

Table 3.6 MSE results of the reconstructed images at 0.5 b/p compression.

Comp. Ratio 0.5 bit/pixel	RUN LENGTH	DFT	DCT	VQ	HFSVQ
PEPPERS	658.43	220.48	151.07	304.80	95.69
BABOON	925.64	356.98	339.39	336.95	293.57
LENA	668.00	173.54	137.26	139.41	76.03
TIFFANY	319. 21	87.27	72.34	107.49	100.18

Table 3.7 MSE results of the reconstructed images at 0.333 b/p compression.

Comp. Ratio 0.333 b/p	RUN LENGTH	DFT	DCT	VQ	HFSVQ
PEPPERS	962.42	242.83	180.47	120.30	117.98
BABOON	1130.97	380.71	364.94	308.71	289.26
LENA	1048.60	179.11	145.89	99.04	100.78
TIFFANY	574.02	91.46	76.58	64.83	62.32

Table 3.8 MSE results of the reconstructed images at 0.250 b/p compression.

Comp. Ratio 0.250 b/p	RUN LENGTH	DFT	DCT	VQ	HFSVQ
PEPPERS	1212.73	319.75	286.57	175.84	153.57
BABOON	1197.55	434.97	424.34	371.41	322.75
LENA	1223.73	205.12	180.70	141.56	117.11
TIFFANY	926.14	109.05	102.39	69.56	61.15

We can say that the Run Length coding is efficient at low compression ratios. MSE of the reconstructed image increases exponentially when the bit rate is decreased. So, at the high compression ratios Run Length Algorithm becomes useless.

The MSE values of the reconstructed images of DCT and DFT are quite reasonable at all bit rates. We can also see that MSE value of DCT is always smaller than DFT's. As a result it can be said that, in high compression rates DCT and DFT are better than Run Length Coding.

At high compression rates such as 0.5 bit/pixel, 0.33 bit/pixel, 0.25 bit/pixel the MSE of HFSVQ algorithm decreases to an optimum value. Especially the HFSVQ technique has the lowest MSE value at high compression rates. This is because of the hierarchical blocking structure of the HFSVQ.

CHAPTER FOUR
BIOMEDICAL IMAGE COMPRESSION

4.1 INTRODUCTION

There are two types of biomedical imaging; one is CT (Computer Tomography) and the other one is MRI (Magnetic Resonance Imaging).

CT and MRI systems are quite similar to each other. The MRI data is obtained from a patient by using following method:

The patient is placed in a magnetic field. Usually the strength of the field is between 0.5 and 1.5 Tesla. This field aligns the atoms with a magnetic moment, such as hydrogen atoms. The hydrogen atoms are then perturbed by an RF signal. As the atoms return to their equilibrium position, they emit their own RF signal, with a frequency that depends on the local magnetic field strength. By introducing variable gradients in the applied fields, and taking phase shifts in the signal into account, the hydrogen density within a particular volume can be reconstructed from the RF signals. The information obtained in this way is a measure of the amount of magnetisation of the hydrogen in each volume element. This value differs fo different types of tissues due to differences in the hydrogen density and its chemical bonds [12].

Both MRI and CT are commonly used in hospitals. In an average sized hospital, many tera-bytes of digital imaging data are generated every year, almost all of which has to be kept and archived. Archieving this large amount of image data in the computer memory is very difficut without any compression.

There are many biomedical image compression algorithms. These techniques were improved in the last ten years. The most common technique is Vector Quantization [2]. The other techniques are Interframe coding [11], Discrete Hartley Transform (DHT) which is the alternative to the Discrete Cosine Transform [16], Mixed Transform [13] and Entropy-Coded DPCM [1]. JPEG algorithm can also be used to compress the biomedical images.

4.2 APPLICATION OF HFSVQ ALGORITHM ON BIOMEDICAL IMAGES

In this chapter HFSVQ algorithm is going to be applied to different biomedical images. These images are called BT (Brain Tomography) and MRI (Magnetic Resonance Image). The image explanations are ;

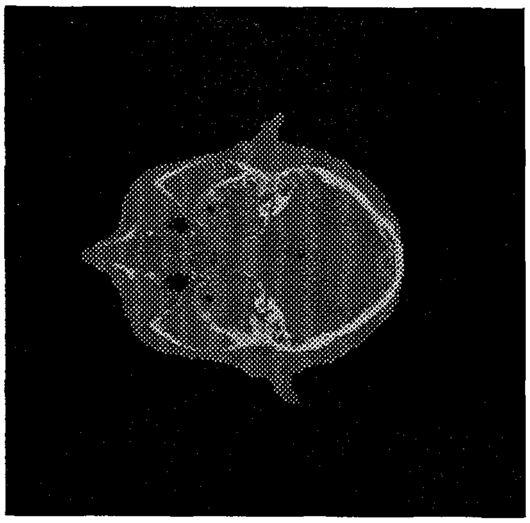
MRI-1 : The axial cranium section passing through sifenoidal sinus.

MRI-2 : The T_1 weighted axial section passing through PONS surface.

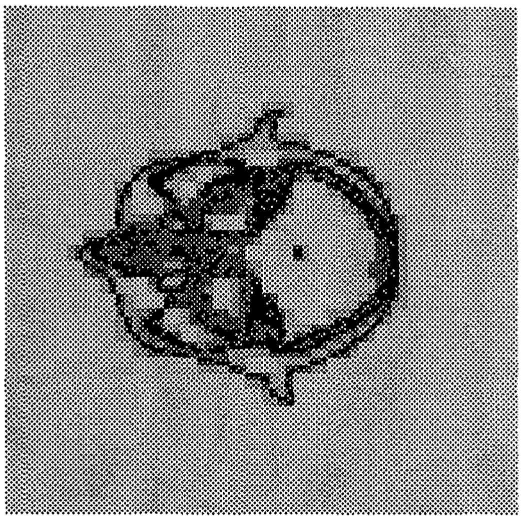
MRI-3 : A Knee image.

BT : The axial cranium section passing through the ventricular plate. In the soft tissue algorithm, intraventricular mass and gray matter are observed.

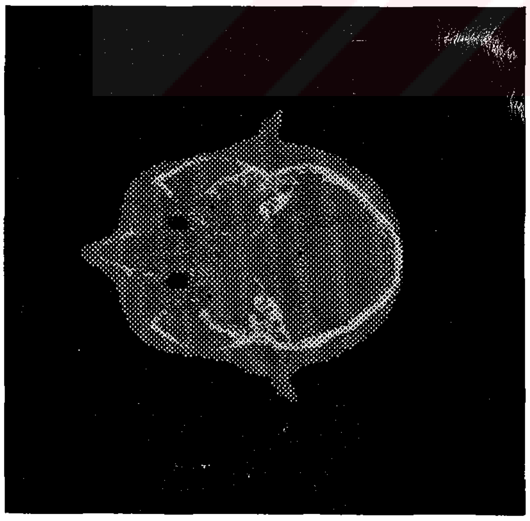
When the HFSVQ algorithm was applied on MRI-1, the compression ratios 0.235 bit/pixel and 0.200 bit/pixel were obtained at the MSE of 17.39 and 28.9 respectively. Since the background area of MRI-2 is smaller than the MRI-1, lower compression ratios are obtained on MRI-2 ; 0.64 bit/pixel and 0.496 bit/pixel at the MSE of 52.17 and 57.47 respectively. The compression ratios of 0.58 bit/pixel at the MSE of 35.59 was obtained for the BT image. When the algorithm was applied on the 512x512 pixel biomedical image (MRI-3) , high compression ratio of 0.184 bit/pixel at the MSE of 61.5 was obtained.



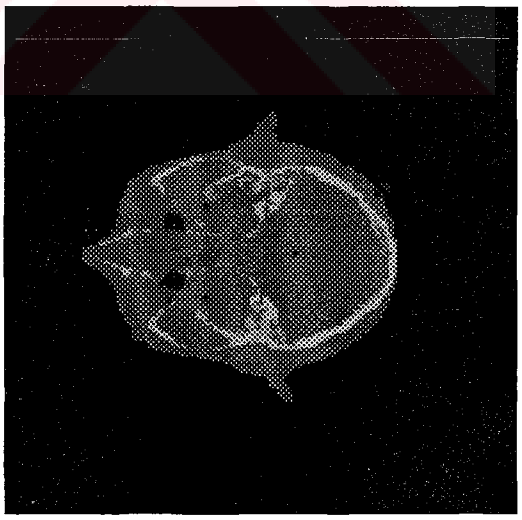
(a) Original 256x256 MRI-1 Image "The axial cranium section passing through sifenoidal sinus"



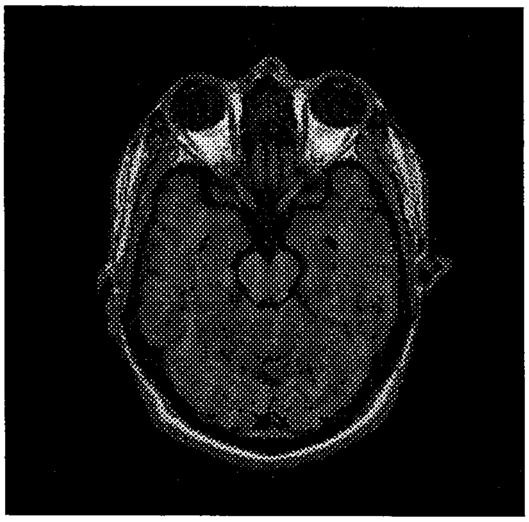
(b) Structure map

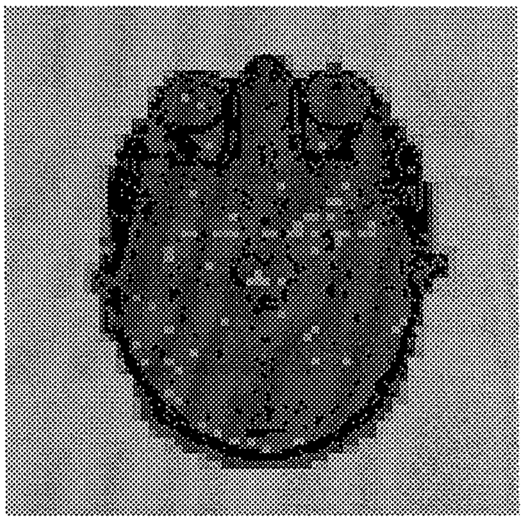


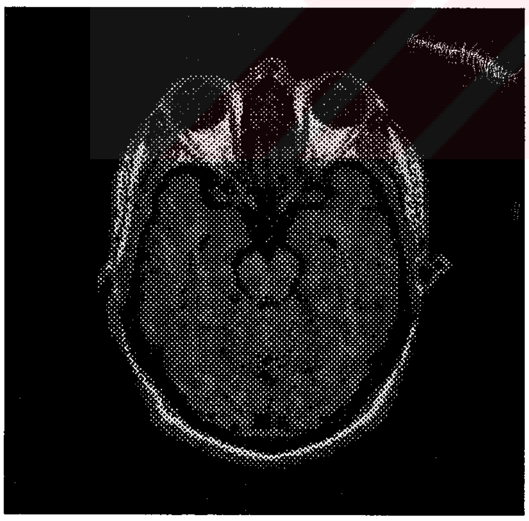
(c) Threshold 1=5, Threshold 2= 60, MSE=17.39, SNR=35.72 dB, Compression Ratio=0.235 bit/pixel.

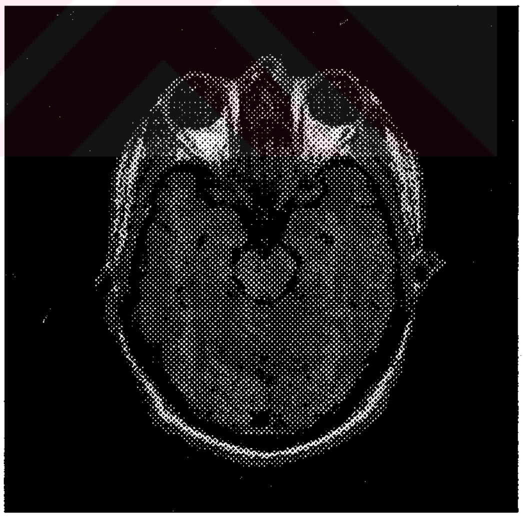


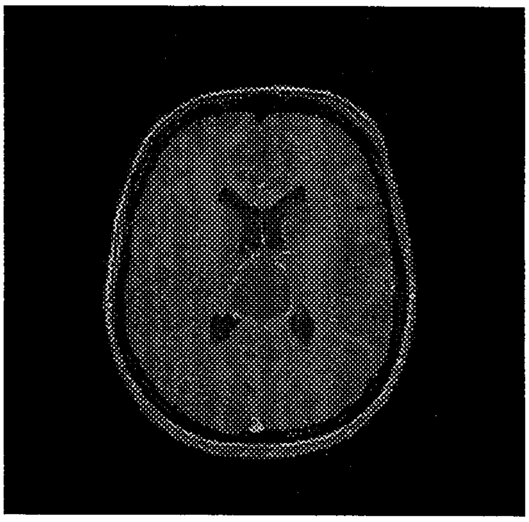
(d) Threshold 1=8,. Threshold 2=70, MSE=28.90, SNR=33.52 dB, Compression Ratio=0.200 bit/pixel


(e) Original 256x256 MRI-2 Image "The T_1 weighted axial section passing through PONS surface"

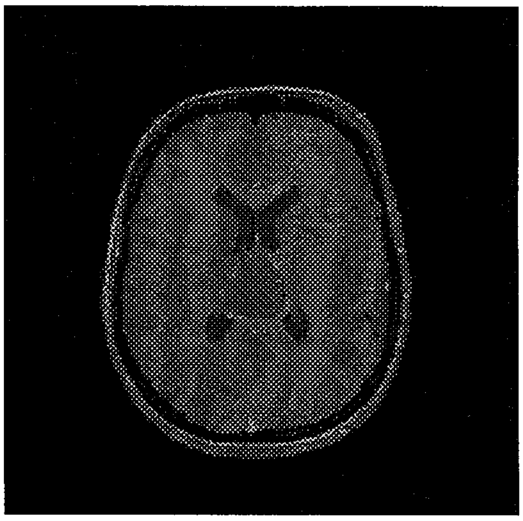

(f) Structure Map


(g) Threshold 1= 0.5, Threshold 2=100, MSE=52.17, SNR=30.95 dB, Compression Ratio=0.64 bit/pixel.

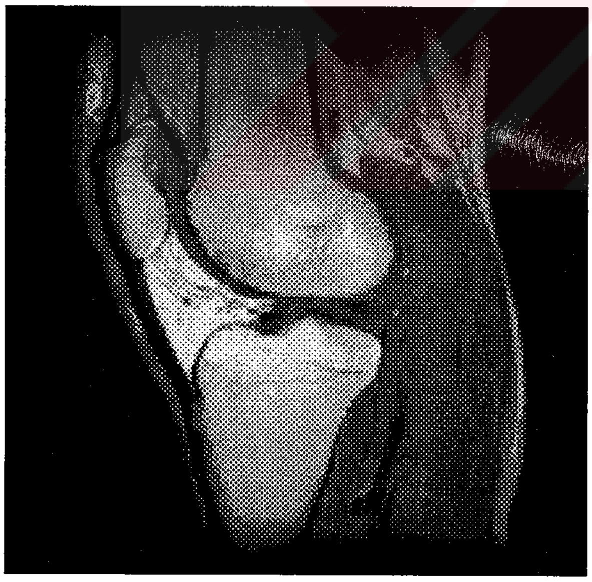

(h) Threshold 1=6.5, Threshold 2=100, MSE=57.47, SNR=30.54 dB, Compression Ratio=0.496 bit/pixel.



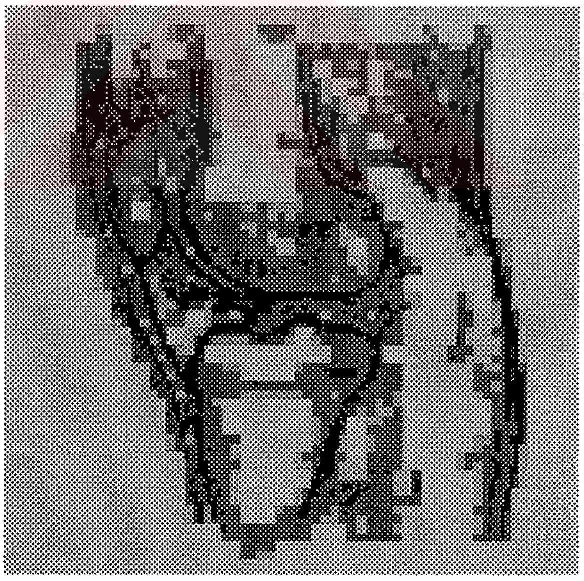
(k) Original 256x256 BT Image "The axial cranium section passing through the ventricular plate. In the soft tissue algorithm, intraventricular mass and gray matter are observed"



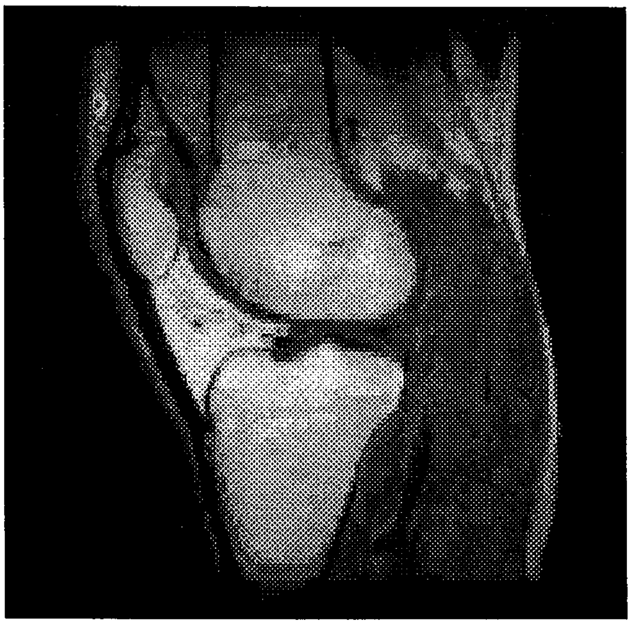
(l) Threshold 1= 1.0, Threshold 2=100, MSE=35.59, SNR=32.62 dB, Compression Ratio=0.58 bit/pixel.



(m) Original 512x512 MRI-3 "Knee" Image



(n) Structure map



(p) Threshold 1= 2.0, Threshold 2=50,
MSE=61.5, SNR=30.24 dB, Compression
Ratio=0.184 bit/pixel.

Figure 4.1 HFSVQ Applications On Biomedical Images

It is seen that bit rate and SNR of HFSVQ algorithm were better than all other methods that was mentioned above. As mentioned in chapter 3, since the block sizes change acording to the gray levels of the images, HFSVQ saves the detailes of the images. The algorithm also has very high SNR. So we can use this algorithm in biomedical image compression safely.

4.3 THE PERFORMANCE RESULTS

In the Interframe Coding [11], the inherent noise of MRI is dealt with by using a median filter within the estimation loop. The residue frames are quantized with a zero-tree wavelet coder, which includes aritmetic entropy coding. The results of this coding technique are approximately 1.2 bit/pixel, 1.4 bit/pixel, 1.6 bit/pixel at the SNR of 24 dB, 25 dB, 26 dB respectively on a 256X256 MRI brain image. The other compression technique is the Tree-Structure Vector Quantization [2]. By using this technique the compression ratios of 0.5 bit/pixel, 1 bit/pixel, 2 bit/pixel at the SNR of 25 dB, 32 dB, 40 dB can be obtained respectively. Discrete Hartley Transform (DHT) [16] is the biomedical image compression technique which is the alternative to the Discrete Cosine Transform. DHT equations are quite similar to DCT equations. By using the algorithm the compression ratio of 1.6 bit/pixel at the SNR of 37.7 dB can be obtained on a biomedical image. The other technique is the Mixed Transform [13]. This technique consists of two or more nonorthogonal transforms. Both lossy and lossless compression implementations are considered. By using this technique the compression ratio of 1.6 bit/pixel at the SNR of 28.42 dB can be obtained on a medical image. An other technique is Entropy-Coded DPCM [1]. This algorithm is better then the others. In this technique, 0.84 bit/pixel at the SNR of 38.1 dB can be obtained. Almost all other lossy compression of medical images performed by using variations on the standard discrete cosine transform (DCT) coding algorithm combined with scalar quantization and lossless coding. In the JPEG compression algorithm DCT is combined with aritmetic coding [1],[6]. In this algorithm 0.86 bit/pixel can be obtained at the SNR of 37.4 dB.

All methods that is mentioned above had tried to obtain very low MSE values. So their compression ratios were very low. However, in chapter 4.2 the Hierarchical Finite State Vector Quantization algorithm was used to compress the biomedical images. By using this method 0.235 bit/pixel and 0.184 bit/pixel were obtained at the SNR of 35.72 dB and 30.24 dB respectively.

The results of HFSVQ shows that the goal of this algorithm is to obtain very high compression ratios at the MSE of optimum values which was achieved in this thesis.

CHAPTER FIVE
CONCLUSION

This thesis presents different types of image compression techniques that are commonly used in the last twenty years. These compression techniques are Run-lenght coding, Discrete Fourier Transform, Discrete Cosine Transform, Vector Quantization and Hierarchical Finite State Vector Quantization which is an improved version of VQ. A comparison of these techniques according to MSE and bit rate was given in Table 3.5, Table 3.6, Table 3.7 and Table 3.8.

When the compression ratio is very low such as 1 bit/pixel, 2bit/pixel, 4 bit/pixel etc, Run Length coding is efficient. MSE of the reconstructed image increases exponentially when the compression ratio is increased. So, at the high compression ratios Run Length Algorithm becomes useless.

DFT and DCT are the standart compression techniques. They are also used to filter the images. The compression formulations of DFT and DCT are quite simple to implement. The MSE values of the reconstructed images of DCT and DFT are quite reasonable at all bit rates. We also see that MSE value of DCT is always smaller than DFT's. Because, although we obtain a magnitude value and a phase value for each pixel after DFT process, we only use one of them in the compression process. So, this causes extra error on this system. Since DCT result value is only magnitude, we do not need to ignore any value in the compression process. As a result we can say that, in high compression ratios, DCT and DFT are better than the Run Length Coding.

At low compression rates such as 1 bit/pixel, 2 bit/pixel, 4 bit/pixel VQ and HFSVQ are not very good according to the other compression techniques, this is bacause of the limited codebook size. At high compression rates such as 0.5 bit/pixel, 0.33 bit/pixel, 0.25 bit/pixel the MSE of these algorithms decrease to an optimum value. Especially the HFSVQ technique has the lowest MSE value at high compression rates. This is because of the hierarchical blocking structure of the HFSVQ.

In this thesis the HFSVQ compression technique was applied to the biomedical images. Since the black area (background) in the used image is quite large, HFSVQ algorithm is very suitable. So, optimum MSE values and high compression rates were obtained, which is generally required in a compression algorithm.

Nowadays, both MRI and CT are commonly used in hospitals. In fact, archieving this large amount of image data in the computer memory is very difficult without high compression. The final goal in the compression techniques is to obtain very low MSE values. So the compression ratio is also very low. However, using the HFSVQ algorithm optimum MSE values are obtained at very high compression ratios in this thesis.

REFERANCES

[1] Chen, Keshi & Ramabadran, Tenkasi V. (1994). Near-Lossless Compression of Medical Images Through Entropy-Coded DPCM. IEEE Transactions on Medical Imaging, 13, 538-548.

[2] Cosman, P. C. & Tseng, C. & Gray, R.M. & Olshen, R.A. & Moses, L.E. & Davidson, H.C. & Bergin, C.J. & Riskin E.A. (1993). Tree-Structured Vector Quantization of CT Chest Scans: Image Quality and Diagnostic Accuracy. IEEE Transactions on Medical Imaging, 12, 727-739.

[3] Cosman, Pamela C. & Gray, Robert M. & Olshen, Richard A. (1994). Evaluating Quality of Compressed Medical Images : SNR , Subjective Rating and Diagnosting Accurarcy. Proceedings Of The IEEE, 82, 919-932.

[4] Gonzales, Rafael C. & Woods, Richard E. (1992). Digital Image Processing. Addison-Wesley Publishing Company, New York.

[5] Gray, Robert M. (1984, April). Vector Quantization. IEEE ASSP Magazine, pp 4-23.

[6] Lim, Jae S.(1990). Two-Dimensional Signal and Image Processing.Prentice Hall, United States of America.

[7] M.I.T. Media Laboratory Perceptual Computing Section Technical Report. (1994). Exaggerated Consensus in Lossless Image Compression. Texas: Popat, Kris & Picard, Rosalin W.

[8] M.I.T. Media Laboratory Perceptual Computing Section Technical Report. (1994). On Modal Modeling for Medical Images: Underconstrained Shape Description and Data Compresssion. Cambridge: Sclaroff, Stan. & Pentland, Alex P.

[9] Nasrabadi, Nasser M. & Feng, Yushu. (1990). Image Compession Using Address - Vector Quantization. IEEE Transactions on Communications, 38, 2166-2173.

[10] Nasrabadi, Nasser M. & King, Robert A. (1988). Image Coding Using Vector Quantization: A Review. IEEE Transactions On Communications, 36, 957-971.

[11] Nasratinia, Aria & Mohserian, Nader & Orchard, Michael T. & Liu, Bede. (1996). Interframe Coding of Magnetic Resonance Images. IEEE Transactions on Medical Imaging, 15, 639-647.

[12] Nosratinia, Aria & Mohsenian, Nader & Orchard, Michael T. & Liu, Bede. (1996) Interframe Coding of Magnetic Resonance Images. IEEE Transactions on Medical Imaging, 15, 639-647.

[13] Ramaswamy, A. & Mikhael, W. B.(1996). A Mixed Transform Approach for Efficient Compression of Medical Images. IEEE Transactions on Medical Imaging, 15, 343-352

[14] Schalkoff, Robert J. (1988). Digital Image Processing and Computer Vision. Addison-Wesley Publishing Company, New York.

[15] Sitaram, Vijay S. & Huang, Chien - Min & Israelsen, Paul D. (1994) . Efficient Codebooks for Vector Quantization Image Compression with an Adaptive Tree Search Algorithm. IEEE Transactions on Communications, 42, 3027-3032.

[16] Villasenor , John D . (1993) . Alternatives to the Discrete Cosine Transform for Irreversible Tomographic Image Compression. IEEE Transactions on Medical Imaging 12, 803-811.

[17] Wieringa, H. J. & Peters, M.J. (1993). Medical & Biological Engineering & Computig, 31, 600-606.

[18] Yu, Ping. & Venetsanopoulos, Anastasias N. (1994). Hierarchical Finite-State Vector Quantization for Image Coding. IEEE Transactions on Communications, 42, 3030-3036.