IMPLEMENTATION AND COMPARISON OF IMAGE COMPRESSION TECHNIQUES

A MAJOR THESIS SUBMITTED
TOWARDS THE PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE

MASTER OF ENGINEERING
IN
ELECTRONICS & COMMUNICATION

Submitted By:-
RASHMI GUPTA
ENROLL NO. : 7/E&C/2003
UNIVERSITY ROLL NO. : 3109

Under the guidance of
Mrs. RAJESHWARI PANDEY

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
DELHI COLLEGE OF ENGINEERING
UNIVESITY OF DELHI
DELHI-110042
2005

CERTIFICATE

This is to certify that the thesis entitled **"Implementation and Comparison of Image Compression Techniques"** being submitted by Rashmi Gupta for the degree of Master of Engineering in Electronics & Communication in the Department of Electronics & Communication, Delhi College of Engineering, University of Delhi , is a record of bonafide work carried out by her under my supervision & guidance. The matter contained in this thesis has not been submitted elsewhere for award of any other degree .

(Mrs. Rajeshwari Pandey)
Lecturer
Deptt. of Electronics & Communication communication
Delhi College of Engineering
Delhi-110042

(Prof. A. Bhattacharya)
Head of Department
Deptt. of Electronics &
Delhi College of Engineering
Delhi-110042

ACKNOWLEDGEMENT

I express my sincere gratitude towards Mrs. Rajeshwari Pandey, my project guide , for giving me an opportunity to work under her, for being a constant source of guidance and encouragement, and for her immense patience.

I am also thankful to Head, Prof. A. Bhattacharya, for his encouragement and kind co-operation throughout the thesis period.

I express my deep sense of gratitude to my friends for their sincere co-operation and continued encouragement.

Last but not least, I am grateful to my mother in law, husband Dr. Anil Gupta and my dear son Master Ashutosh Gupta for whose love and sacrifice for me I have no words.

(Rashmi Gupta)

07/E&C/2003
Roll No.3108

ABSTRACT

In this thesis, a comparative study between transforms used for the compression of still images i.e. Discrete cosine transform (DCT), Karhunen Loeve Transform (KLT) and Discrete wavelet transform (DWT) is presented. MATLAB software is used to generate image compression results for algorithms using DCT, KLT and DWT.

The Karhunen Loeve Transform (KLT) is optimal transform , as it packs most of the energy into a fewer number of frequency domain elements and it completely de-correlates its elements but it is theoretical concept & practically very difficult to implement as its basis functions are data dependent.

In terms of complexity involved, DCT achieves the fastest computational performance due to fast algorithms.

Wavelet Transform has superior performance than other transforms in terms of peak signal to noise ratio & image quality. DWT provides higher compression ratio due to the wavelet transform multi-resolution capability which allows it to analyze signal at various scale and resolutions.

CONTENTS

CHAPTER 1

INTRODUCTION

Digital images are used in many diverse applications, including multimedia technology, digital photography, Internet viewing, image archiving and medical imaging. With the advance development in Internet and multimedia technologies, the amount of information handled by computers has grown exponentially over the past decades. This information require large amount of storage space and transmission bandwidth.

One of possible solution to this problem is to compress the information so that the storage space and transmission time can be reduced. Hence in order to transmit and store digital images, the images must be compressed, otherwise each image would require a huge amount of memory. Different Transform methods of image compression are Discrete Wavelet Transform (DWT), Discrete Cosine Transform (DCT), Karhunen Loeve Transform (KLT) , Discrete Fourier Transform (DFT), Discrete Sine Transform (DST) , Hadamard Transform, Harr Transform, Slant Transform & Singular value decomposition transform.

The energy compaction property of the DCT, KLT and DWT is well suited for image compression since, as in most images, the energy is concentrated in the low to middle frequencies, and the human eye is more sensitive to the low frequencies.

The optimal transform is the KLT as it packs most of the energy into a fewer no. of frequency domain elements but it is theoretical concept & practically very difficult to implement because its basis functions are image dependent so this complicates the digital implementation.

The Discrete Cosine Transform (DCT) was introduced by Ahmed in 1974. DCT is a approximation of the optimal Karhunen Love transform (KLT) with large correlation coefficient. It has satisfactory performance in terms of energy compaction capability, and many fast DCT algorithms with efficient hardware and software while also having image independent basis functions.

DWT based coders have outperformed DCT coders both in terms of image quality and higher PSNR. The most important reason why wavelet transformation is so powerful is its Multi-Resolution Analysis (MRA) capability, which allows it to analyze signal at various scale and resolution.

In terms of complexities involved in implementing DCT and DWT, DCT achieves the fastest computational performance due to fast algorithms.

1.1 GRAYSCALE AND COLOR IMAGE COMPRESSION

1.1.1. GRAYSCALE IMAGE COMPRESSION

A digital grayscale image is typically represented by 8 bits per pixel (bpp) in its uncompressed form. Each pixel has a value ranging from 0 (black) to 255 (white). Transform methods are applied directly to a two dimensional image by first operating on the rows, and then on the columns. Transforms that can be implemented in this way are called separable.

1.1.2 COLOR IMAGE COMPRESSION

A digital color image is stored as a three-dimensional array and uses 24 bits to represent each pixel in its uncompressed form. Each pixel contains a value representing a red ®, green (G), and blue (B) component scaled between 0 and 255– this format is known as the RGB format. The PSNR is measured for each compressed component just as for grayscale images. The three output components are reassembled to form a reconstructed 24-bit color image (image out).

1.2 APPROACH TO THESIS :

1. Study & Evaluation of Image Compression Techniques using Discrete Wavelet Transform, Discrete Cosine Tranform and Karhunen- Loeve Transform.
2. Implementation of Image Compression techniques for gray images using MATLAB software.
3. Implementation of Image Compression techniques for color images using MATLAB software.
4. Comparison of obtained results in terms of compression ratio, peak signal to noise ratio, human visual system & complexity involved using MATLAB software.

1.3 OUTLINE OF THE THESIS

The remaining chapters of this work are organized as follows:

Chapter 2 briefly discusses fundamentals of image compression & summary of JPEG standard. Chapter 3 discusses summary of image compression technique using Karhunen Loeve Transform and its algorithm. Chapter 4 discusses summary of image compression technique using Discrete Cosine Transform and its algorithm. Chapter 5 discusses summary of image compression technique using Discrete Wavelet Transform and its algorithm.

Chapter 6 presents the simulation results of above mentioned compression techniques & comparison based on compression ratio, human visual system, peak signal to noise ratio & complexity involved.

Chapter 7 presents the conclusions of the thesis along with suggestions for future work in this area.

CHAPTER 2

OVERVIEW OF IMAGE COMPRESSION & JPEG STANDARD

2.1 INTRODUCTION OF IMAGE COMPRESSION

Reducing the amount of data to reproduce images or video is called compression and it saves

- Storage space
- Increase access speed
- A way to achieve digital motion video on personal computer.

Compression is concerned with minimizing the no. of bits required to represent an image. Perhaps the simplest and most dramatic form of data compression is the sampling of band limited images where an infinite number of pixels per unit area is reduced to one sample without any loss of information.

A common characteristic of most images is that the neighbouring pixels are correlated and therefore contain redundant information. The foremost task then is to find less correlated representation of the image. Two fundamental components of compression are redundancy and irrelevancy reduction.

Redundancy reduction aims at removing duplication from the signal source (image/video). **Irrelevancy reduction** omits parts of the signal that will not be noticed by the signal receiver, namely the Human Visual System (HVS). In general, three types of redundancy can be identified.

- Spatial redundancy which is due correlation between neighbouring pixel values.
- Spectral Redundancy which is due to correlation between different color planes or spectral bands.
- Temporal Redundancy which is due to correlation between adjacent frames in a sequence of images (in video applications).

Image compression researches aims to reduce the number of bits required to represent an image by removing these redundancies. Only spatial redundancy and spectral redundancy are removed in still images.

2.1.1 NEED OF COMPRESSION

The example given below clearly illustrate the need for very high storage space, large transmission bandwidth and long transmission time for uncompressed image, audio and video data.

Multimedia data	Size/ Duration	Bits/Pixel	Uncompressed Size	Transmission B.W	Transmission Time
A page of text	11"x8.5"	Varying resolution	4-8KB	32-64 Kb/page	1.1-2.2 sec
Telephone quality	10sec.	8bps	80KB	64Kb/sec	22.2 sec
Grayscale image	512 x 512	8bpp	262KB	2.1 Mb/image	1 min 13 sec
Color image	512 x 512	24bpp	786KB	6.29Mb/image	3min 39 sec
Medical image	2048 x 2048	12bpp	5.16MB	4.13Mb/image	23 min 54 sec
SHD Image	2048 x 2048	24bpp	12.58MB	1000Mb/image	58 min 15 sec
Full-motion Video	640x480,1min (30frames/sec)	24bpp	1.66GB	221 Mb/sec	5 days 8hrs.

Table 2.1: Multimedia data types with uncompressed storage space, transmission bandwidth and transmission time required. The prefix kilo-denotes a factor of 1000 rather than 1024.

At the present state of technology, the only solution is to compress multimedia data before its storage and transmission, and decompress it at the receiver for playback. For example, with a compression ratio of 32:1, the space, bandwidth and the transmission time requirements can be reduced by a factor of 32, with acceptable quality.

2.1.2 CLASSIFICATION OF COMPRESSION TECHNIQUES

2.1.2.1 LOSSLESS VS. LOSSY COMPRESSION:

LOSSLESS: In lossless compression schemes, the reconstructed image after compression is numerically identical to the original image. However lossless compression can only achieve a modest amount of compression.
Common lossless compressors include pkzip, winzip, bzip1 and bzip2. These are able to reconstruct an exact duplicate of the original input file after it has been compressed.

LOSSY: An image reconstructed following lossy compression contains degradation relative to the original image. Often this is because the compression scheme completely discards the redundant information. However, lossy schemes are capable of achieving much higher compression. Under normal viewing conditions, no visible loss is perceived (visually lossless).
Common lossy compressors include JPEG, MPEG, MP3 etc. These formats generally work by reproducing a file that can be quite different to the original at bit level, while being indistinguishable to the human ear or eye for most particular purposes.

2.1.2.1 PREDICTIVE VS.TRANSFORM CODING :

PREDICTIVE CODING: In predictive coding, information already sent or available is used to predict future values, and the difference is coded. Since this is done in the image or spatial domain, it is relatively simple to implement and is readily adapted to local image characteristics. Differential Pulse Code Modulation is one particular example of predictive coding.

TRANSFORM CODING: Transform coding first transforms the image from its spatial domain representation to a different type of representation using the some well known transform and then codes the transformed value (coefficients). This method provides greater data compression compared to predictive methods, although at the expense of greater computation.
In transform coding, a reversible, linear transform (such as F.T.) is used to map the image into a set of transform coefficients.
The goal of transformation process is to de-correlate the pixels of each sub image, or to pack as much information as possible into the smallest number of coefficients.

2.1.3 IMAGE COMPRESSION CHARACTERISTICS

There are three main characteristics by which image-compression algorithms can be judged: compression ratio, compression speed, and image quality. These characteristics can be used to determine the suitability of a given compression algorithm to our application.

2.1.3.1 COMPRESSION RATIO

Compression performance is often specified by giving the ratio of input data to output data for compression process. This basic measure for the performance of a compression algorithm is compression ratio.

$$C.R. = \frac{size\ of\ original\ \ image\ \ - size\ of\ compressed\ \ \ image}{size\ of\ original\ \ image}$$

This ratio gives an indication of how much compression is achieved for a particular image.

The compression ratio achieved usually indicates the picture quality. Higher the compression ratio, the poorer the quality of the resulting image. The trade off between compression ratio and picture quality is an important one to consider when compressing images.

Furthermore, some compression schemes produce compression ratios that are highly dependent on the image content. This aspect of compression is called data dependency. Using an algorithm with a high degree of data dependency, an image of crowd at a football game (which contains a lot of detail) may produce a very small compression ratio, whereas an image of a blue sky (which consists mostly of constant colors and intensities) may produce a very high compression ratio.

A much better way to specify the amount of compression is to determine the number of bits per displayed pixel needed in the compressed bit stream.

2.1.3.2 IMAGE QUALITY

Image quality describes the fidelity with which an image-compression scheme recreates the source image data. Compression scheme can be characterized as being either lossy or lossless. Lossless schemes preserve all of the original data. Lossy compression does not preserve the data precisely. Image data is lost, and it cannot be recovered after compression. Most lossy scheme try to compress the data as much as

possible, without decreasing the image quality in a noticeable way. Some scheme may be either lossy or lossless, depending upon the quality level desired by the user.
Higher compression ratio may produce lower picture quality. Quality and compression can also vary according to source image characteristics and scene content. One measure for the quality of the picture is number of bits per pixel in a compressed image (BPP) which is defined as the total number of bits in the compressed image divided by the number of pixels.

$$bpp = \frac{number \ \ of \ bits \ in \, compressed \ \ image}{no\,.of \ \ pixels}$$

According to this measure, four different picture qualities are defined.

(bits / pixel)	Picture quality
0.12-0.5	Moderate to good quality
0.5-0.75	Good to very good quality
0.75-1.0	Excellent quality
1.5-2.0	Undistinguishable from the original

2.1.3.3 COMPRESSION SPEED

Compression time and decompression time are defined as the amount of the time required to compress and decompress a picture, respectively. Their value depends on the following considerations:

- The complexity of the compression algorithm
- The efficiency of the software or hardware implementation of the algorithm
- The speed of the utilized processor or auxiliary hardware

2.1.3.4 PSNR

Peak signal-to-noise ratio (PSNR) is the standard method for comparing a compressed image with the original image. It is not a direct measure of the perceptual visual quality, i.e. the way the image looks to human eye. However, PSNR can be used as an indicator of image quality.

$$PSNR = 10 \log \frac{MAX^2}{\frac{1}{MXN}\sum_{m=1}^{M}\sum_{n=1}^{N}(X(m,n) - Y(m,n))^2}$$

where M and N are width and height of image. X is the original image data and Y is the compressed image data. MAX is the max. value that a pixel can have, 255.
PSNR is measured in decibels (dB).

2.2 INTRODUCTION OF JPEG STANDARD

JPEG stands for Joint Photographic Experts Groups is a standard image compression mechanism. There was no adequate standard for compressing 24-bit per pixel color data, committee came up with algorithm for compressing color or grayscale images depicting real world scenes (like photographs). JPEG handles only still images but there is a related standard called MPEG for motion pictures.

JPEG is "lossy," meaning that the decompressed image is not exactly same as the one started with. (There are lossless image compression algorithms, but JPEG achieves much greater compression than is possible with lossless methods). JPEG is designed to exploit known limitations of the human eye, notably the fact that small color changes are perceived less accurately than small changes in brightness. Thus, JPEG is intended for compressing images that will be looked at by humans. If images are planed to machine-analyze, the small errors introduced by JPEG may be problem, even if they are invisible to our eyes.

A useful property of JPEG is that adjusting compression parameters can vary the degree of lossiness. This means that the image-maker can trade off file size against output image quality. Extremely small files can make if little poor quality is acceptable.

JPEG images have become a default standard for a variety of mediums, mainly the internet and other applications where it is necessary to have images of high quality, but low data size. The advantages of this include low transmission times for Internet web browsing and the development of useful digital cameras that can store a number of images on a relatively low amount of on-board memory.

2.2.1 JPEG ARCHITECHURE

In the JPEG image compression the input image is divided into 8-by-8 or 16-by-16 blocks, and the two-dimensional DCT is computed for each block. The DCT coefficients are then quantized, coded, and transmitted. The JPEG receiver (or JPEG file reader) decodes the quantized DCT coefficients, computes the inverse two-dimensional DCT of each block, and then puts the blocks back together into a single image. For typical images, many of the DCT coefficients have values close to zero; these coefficients can be discarded without seriously affecting the quality of the reconstructed image.

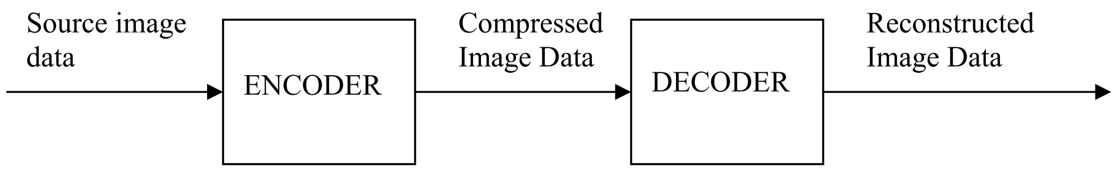


Figure 2.1 : Block Diagram of Image Compression

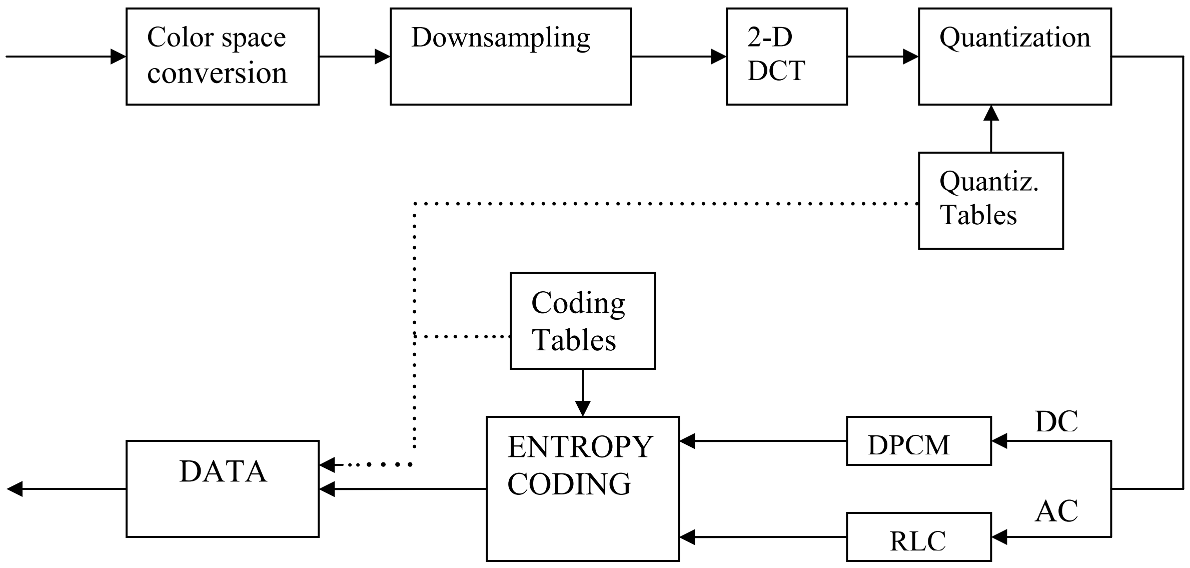


Figure 2.2 : Block Diagram of JPEG Encoder

2.2.2 MAJOR PARTS OF JPEG ENCODER

- Color space Conversion and Down-sampling
- DCT (Discrete Cosine Transformation)
- Quantization
- Zigzag Scan
- DPCM on DC component & RLE on AC Components
- Entropy Encoding (Huffman Encoding & Arithmetic Encoding)

CHAPTER 3

THE KARHUNEN-LOEVE TRANSFORM (KLT)

Originated from the series expansions for random processes developed by Karhunen and Loeve in 1947 and 1949 based on the work of Hoteling in 1933 (the discrete version of the KL transform). Also known as Hoteling transform or method of principal component. It packs the maximum energy in first few samples. It minimizes the mean square error for any truncated series expansion. Error vanishes in case there is no truncation. The idea is to transform a signal into a set of uncorrelated coefficients.

Karhunen Loeve Transform provides the orthogonal basis along which the coefficients are uncorrelated.

3.1 K.L TRANSFORM OF IMAGES

An N x N image is represented by a two dimensional random sequence *v (m,n).* It can be represented by matrix of order N x N. Alternatively, a given N x N image can be viewed as an N^2 x 1 column vector v. Now just as one dimensional signal can be represented by an orthogonal series of basis function, an image can also be generated by unitary matrices.

A general orthogonal series expansion for an N x N image v (m, n) is given as,

General form:

$$v(m,n) = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} u(k,l)\psi(k,l,m,n)$$

where m, n = 0,1,2………., N-1

And the kernel $\psi(k, l, m, n)$ is given by the orthonormalized eigenvectors of the correlation matrix, i.e. it satisfies

$$\lambda i \psi i = R \psi i \qquad i= 0, \dots , N^2 - 1$$

R is the (N^2 x N^2) autocorrelation matrix of image and ψ_i are (N^2 x 1) eigen vectors. So R matrix of the image mapped into an (N^2 x 1*)* vector and ψi is the ith column of ψ

If *R* is separable, i.e.,

$R = R1 \otimes R2$

then the KL kernel is also separable, i.e.,

$\psi(k, l; m, n) = \psi 1(m, k)\, \psi 2(n, l)$

or

$\psi = \psi 1 \otimes \psi 2$

ψ is called the eigen matrix of R.

The elements *u (k ,l)* are called the transform coefficients.

$$u(k,l) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} v(m,n)\psi(k,l,m,n)$$

where k ,l =0, 1 ,2,….,N-1

For images, the eigen matrix of auto-correlation matrix R can be obtained using the separable property of auto correlation matrix R. In which we separate the N ^{3} x N ^{3} matrix into three N x N matrix and then find the eigen matrix of each. After that by taking the kronecker product of these eigen matrix we get the eigen matrix of auto correlation matrix R.

Advantages of separability

Reduce the computational complexity from $O(N^6)$ to $O(N^3)$

Recall that an N x N eigen value problem requires $O(N^3)$ computations

3.2 PROPERTIES OF THE KL TRANSFORM

1. Decorrelation:

   The KL transform coefficients are uncorrelated and have zero mean, i.e.,

   $E[v(k,l)] = 0$ for all k,l
2. It minimizes the mean square error for any truncated series expansion. Error vanishes in case there is no truncation.
3. Among all unitary transformations, KL packs the maximum average energy in the first few samples of v.

3.3 DRAWBACKS

1. Unlike other transforms, the KL is image dependent.
2. It is practically very difficult to implement.
3. It is computationally very intensive.

3.4 ALGORITHM FOR K-L TRANSFORM

```
1)  Input Image
2)  Resize the image of size 256*256
3)  Image into column matrix (b)
4)  Find mean (m) of b
5)  Find zero mean image i.e.
        c = b-m
6)  Find autocorrelation of zero mean image.
7)  Find eigen values and eigen vectors
8)  Multiplication of zero mean image with eigen vectors (g)
9)  Input the value of threshold
10) If eigen value =< threshold value
        g(i) = 0
    otherwise
        g(i) = original value
11) Displaying the no. of compressed coefficients
```

12) Recovery of the compressed image

13) Computing the compression ratio

14) Computing the PSNR of image

15) Computing the simulation time of image

16) Plotting of original image,

17) Plotting of KLT & Reconstructed image

18) Plotting of SNR with respect to compression ratio

19) Plotting of Simulation time with respect to compression ratio

CHAPTER 4

DISCRETE COSINE TRANSFORM (DCT)

Discrete cosine transform (DCT) is a technique for converting signal into elementary frequency components. It is widely used in image compression.

The DCT has the property that most of the visually significant information about the image is concentrated in just a few coefficients of the DCT. For this reason, the DCT is often used in image compression applications. The DCT is the heart of the international standard lossy image compression algorithm known as JPEG. (The name comes from the working group that developed the standard: the Joint Photographic Experts Group).

The DCT is a lossless and reversible mathematical transformation that converts a spatial amplitude representation of data into a spatial frequency representation. One of the advantages of the DCT is its energy compaction property i.e. the signal energy is concentrated on a few components while most other components are zero or negligible small. The DCT was first introduced in 1974 and since it has been used in many applications such as filtering, transmultiplexers, speech coding, image coding (still frame, video and image storage), pattern recognition, image enhancement. The DCT is widely used in image compression applications, especially in lossy image compression, MPEG moving image compression, and the H.261 and H.263 video-telephony coding schemes. The energy compaction property of the DCT is well suited for image compression as in the most images, the energy is concentrated in the low frequencies, and the human eye is more sensitive to the low frequencies.

In the JPEG image compression algorithm, the input image is divided into 8-by-8 or 16-by-16 blocks, and the two-dimensional DCT is computed for each block. The DCT coefficients are then quantized, coded, and transmitted. The JPEG receiver (or JPEG file reader) decodes the quantized DCT coefficients, computes the inverse two-dimensional DCT of each block, and then puts the blocks back together into a single image. For typical images, many of the DCT coefficients have values close to zero; these coefficients can be discarded without seriously affecting the quality of the reconstructed image.

4.1 THE 2-D DISCRETE COSINE TRANSFORM (DCT)

The discrete cosine transform (DCT) is closely related to the discrete Fourier transform. It is a separable, linear transformation; that is, the two-dimensional transform is equivalent to a one-dimensional DCT performed along a single dimension followed by a one-dimensional DCT in the other dimension. The definition of the two-dimensional DCT for an input image A and output image B is

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} \cos \frac{\pi(2m+1)p}{2M} \cos \frac{\pi(2n+1)q}{2N}$$

where $0 \leq p \leq M\text{-}1$
$0 \leq q \leq N\text{-}1$

$$\alpha_p = \begin{cases} 1/\sqrt{M}, & p = 0 \\ \sqrt{2/M}, & 1 \leq p \leq M-1 \end{cases}$$

$$\alpha_q = \begin{cases} 1/\sqrt{N}, & q = 0 \\ \sqrt{2/N}, & 1 \leq q \leq M-1 \end{cases}$$

M = Number of rows in the input data set

N = Number of columns in the input data set

m = Row index in the time domain $0 \leq m \leq M\text{-}1$

n = Column index in the time domain $0 \leq n \leq N\text{-}1$

Amn = Time domain data

p = Row index in the frequency domain

q = Column index in the frequency domain

Bpq = Frequency domain coefficients

The values *Bpq* are called the DCT coefficients of A. (Matrix indices in MATLAB always start at 1 rather than 0; therefore, the MATLAB matrix elements *A(1,1)* and *B(1,1*) correspond to the mathematical quantities *A(0,0)* and *B(0,0)* respectively.

4.2 THE 2-D INVERSE DISCRETE COSINE TRANSFORM (IDCT)

IDCT2 computes the two dimensional inverse discrete cosine transform using

$$A_{mn} = \alpha_p \alpha_q \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} B_{pq} \cos \frac{\pi(2m+1)p}{2M} \cos \frac{\pi(2n+1)q}{2N}$$

$$\alpha_p = \begin{cases} 1/\sqrt{M}, & p = 0 \\ \sqrt{2/M}, & 1 \le p \le M-1 \end{cases}$$

$$\alpha_q = \begin{cases} 1/\sqrt{N}, & q = 0 \\ \sqrt{2/N}, & 1 \le q \le M-1 \end{cases}$$

M = Number of rows in the input data set

N = Number of columns in the input data set

m = Row index in the time domain $0 \le m \le M\text{-}1$

n = Column index in the time domain $0 \le n \le N\text{-}1$

Amn = Time domain data

p = Row index in the frequency domain

q = Column index in the frequency domain

Bpq = Frequency domain coefficients

The inverse DCT equation can be interpreted as that any M x N matrix, A can be written as a sum of *MN* functions of the form

$$\alpha_p \alpha_q \cos \frac{\pi(2m+1)p}{2M} \cos \frac{\pi(2n+1)q}{2N}$$

These functions are called the basis functions of the DCT. The DCT coefficients *Bpq*, then, can be regarded as the weights applied to each basis function.
Horizontal frequencies increase from left to right, and vertical frequencies increase from top to bottom. The constant-valued basis function at the upper left is often called the DC basis function, and the corresponding DCT coefficient *B (0,0)* is often called the DC coefficient.

4.3 THE 64 (8 X 8) DCT BASIS FUNCTIONS

As in the one-dimensional case, each element B(p,q) of the transform is the inner product of the input and a basis function, but in this case, the basis functions are nxm matrices. Each two-dimensional basis matrix is the outer product of two of the one-dimensional basis vectors. Each basis matrix can be thought of as an image. For *n = m = 8,* the *64* basis images in the array are shown in figure 4.1.

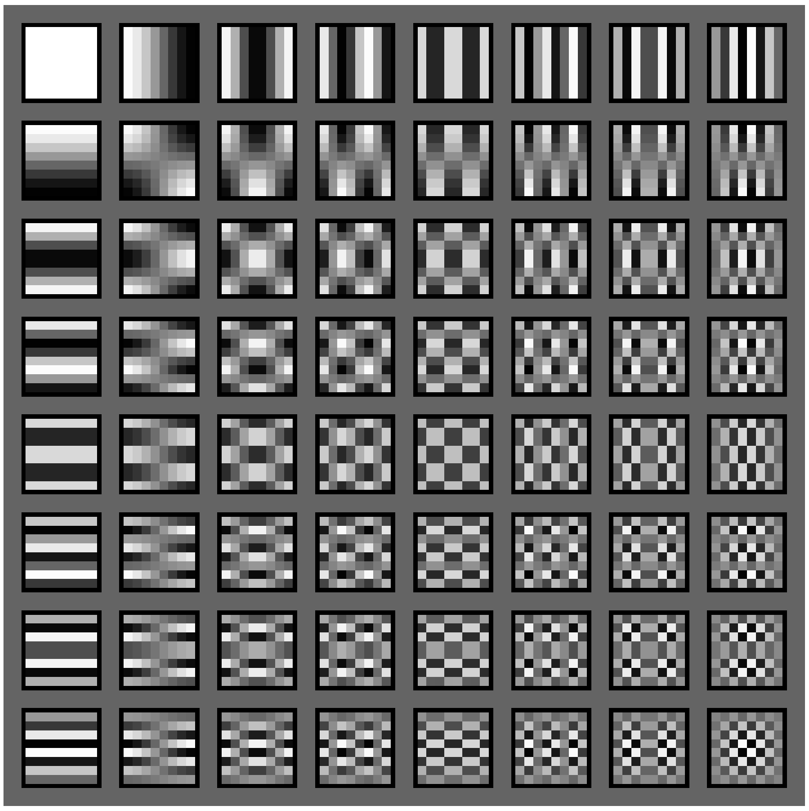
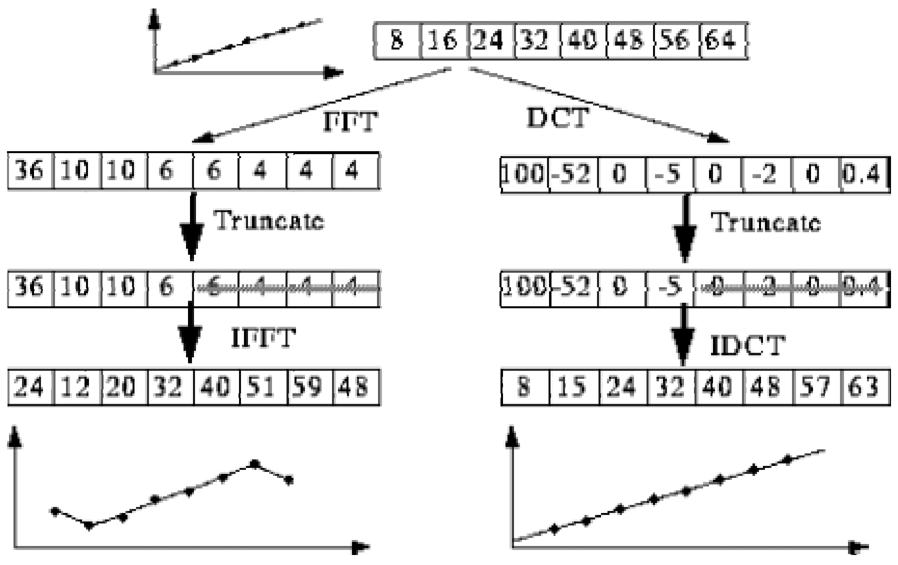


Figure 4.1: The *8X8* Array of Basis Images For The 2D Discrete Cosine Transform

Each basis matrix is characterized by a horizontal and a vertical spatial frequency. The matrices shown here are arranged left to right and bottom to top in order of increasing frequencies.

Why DCT not FFT? -- DCT is like FFT, but can approximate linear signals well with few coefficients.



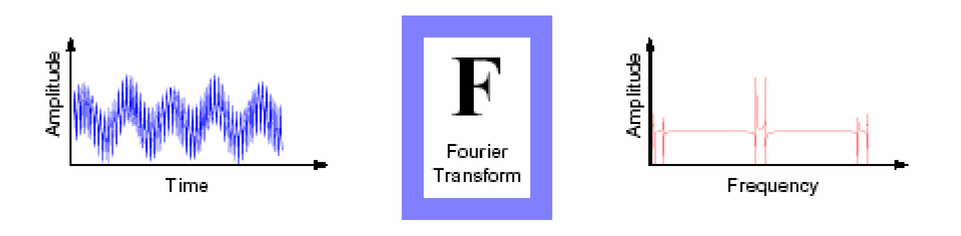
4.4 PROPERTIES OF DCT

1. The cosine transform is real.
2. It is a fast transform.
3. It is very close to the KL transform.
4. It has excellent energy compaction for highly correlated data.

CHAPTER 5

WAVELET TRANSFORM

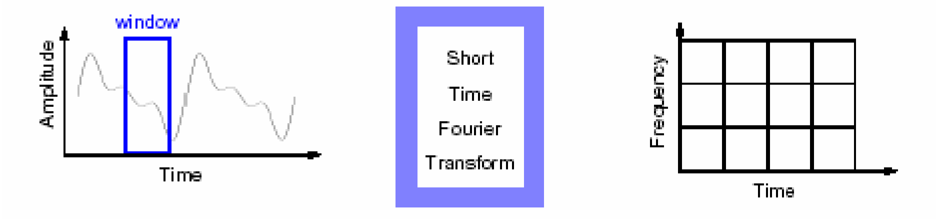
Fourier analysis breaks down a signal into constituent sinusoids of different frequencies. Fourier analysis is as a mathematical technique for transforming our view of the signal from time-based to frequency-based.



For many signals, fourier analysis is extremely useful because the signal's frequency content is of great importance. Fourier analysis has a serious drawback. In transforming to the frequency domain, time information is lost. When looking at a fourier transform of a signal, it is impossible to tell when a particular event took place.

If the signal properties do not change much over time is called a stationary signal. However, most interesting signals contain numerous non-stationary or transitory characteristics: drift, trends, abrupt changes, and beginnings and ends of events. These characteristics are often the most important part of the signal, and Fourier analysis is not suited to detect them.

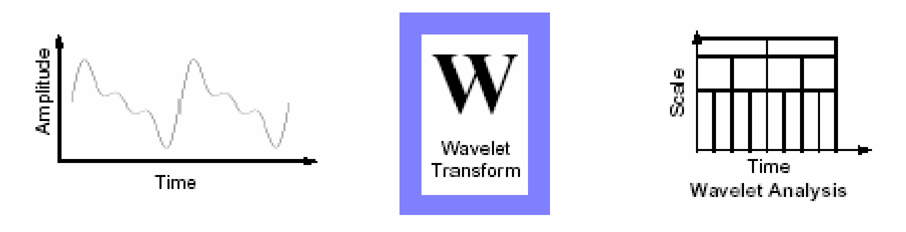
In an effort to correct this deficiency, Dennis Gabor (1946) adapted the Fourier transform to analyze only a small section of the signal at a time—a technique called windowing the signal, called the **Short-Time Fourier Transform** (STFT), maps a signal into a two-dimensional function of time and frequency.



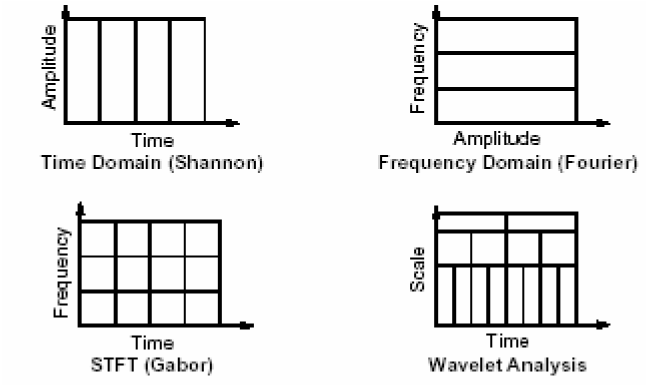
The STFT represents a sort of compromise between the time and frequency based views of a signal. It provides some information about both when and at what frequencies a signal event occurs. However, this information can be obtained with limited precision, and that precision is determined by the size of the window.

While the STFT compromise between time and frequency information can be useful, the drawback is that once a particular size for the time window is choosed, that window is the same for all frequencies. Many signals require a more flexible approach—one where the window size can vary to determine more accurately either time or frequency.

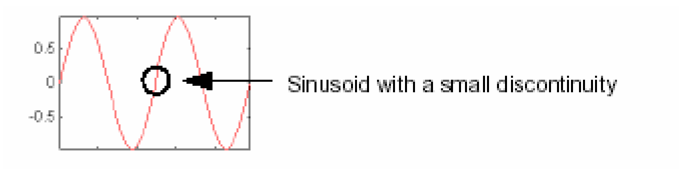
Wavelet analysis represents the next logical step: a windowing technique with variable-sized regions. Wavelet analysis allows the use of long time intervals where we want more precise low-frequency information, and shorter regions where we want high-frequency information.



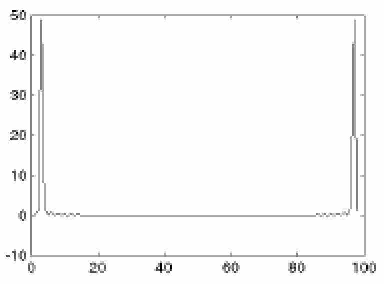
The time-based, frequency-based, and STFT views of a signal:

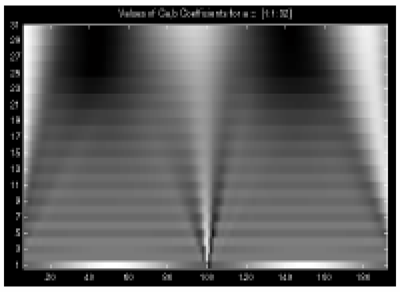


Wavelet analysis does not use a time-frequency region, but rather a time-scale region. One major advantage of wavelets is the ability to perform local analysis that is, to analyze a localized area of a larger signal. Consider a sinusoidal signal with a small discontinuity one so tiny as to be barely visible. Such a signal easily could be generated in the real world, perhaps by a power fluctuation or a noisy switch.

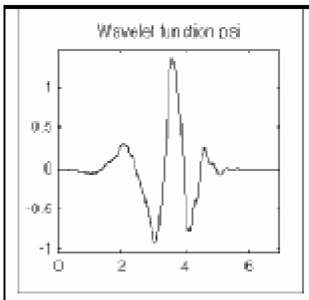
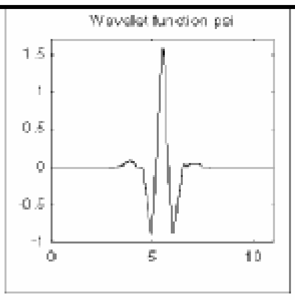
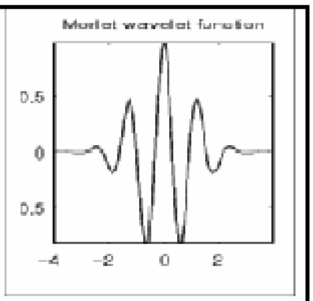


A plot of the Fourier coefficients (as provided by the fft command) of this signal shows a flat spectrum with two peaks representing a single frequency. However, a plot of wavelet coefficients clearly shows the exact location in time of the discontinuity. Wavelet analysis can often compress or de-noise a signal without appreciable degradation.

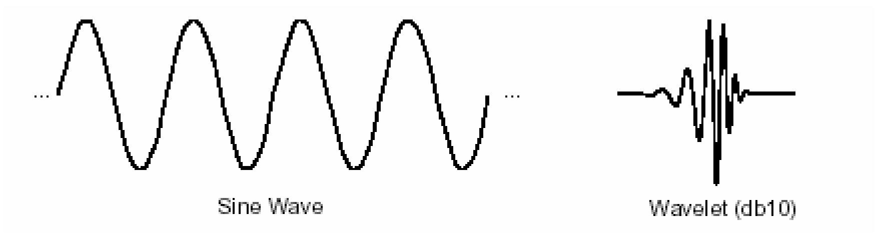

Fourier Coefficients


Wavelet Coefficients

A wavelet is a waveform of effectively limited duration that has an average value of zero. The basis functions of the wavelet transform are known as wavelets. There are a variety of different wavelet functions to suit the needs of different applications. In general, a wavelet is a small wave that has finite energy concentrated in time. It is this characteristic about a wavelet that gives it the ability to analyze any time-varying signals.

		
dB4	**coif2**	**Morlet**

Comparing wavelets with sine waves, which are the basis of Fourier analysis, sinusoids do not have limited duration. They extend from minus to plus infinity. And where sinusoids are smooth and predictable, wavelets tend to be irregular and asymmetric.



Fourier analysis consists of breaking up a signal into sine waves of various frequencies. Similarly, wavelet analysis is the breaking up of a signal into shifted and scaled versions of the original (or mother) wavelet.

From pictures of wavelets it is observed that signals with sharp changes can be better analyzed with an irregular wavelet than with a smooth sinusoid, just as some foods are better handled with a fork than a spoon.

There are two types of wavelet transform. They are the continuous wavelet transform (CWT) and discrete wavelet transform (DWT). The main idea about the wavelet transform is the same in both of these transforms. However, they differ in the way the transformation is being carried out. In CWT, an analyzing window is shifted along the time domain to pick up the information about the signal. This process is difficult to implement and the information that has been picked up may overlap and result in redundancy. In still Image Compression using Wavelet Transform DWT, signals are analyzed in discrete steps through a series of filters. This method is realizable in a computer and has the advantage of extracting non-overlapping information about the signal.

To take a wavelet transform, a wavelet base function is first selected and then each possible scaling and translation of that wavelet is correlated with the function to be transformed. The correlation values thus obtained are the coefficients of the wavelet transform. The equation for scaling and translation of the wavelet function is

$$\psi_{a,b} = \frac{1}{\sqrt{a}} \psi \left(\tfrac{t-b}{a} \right)$$

where $\psi(t)$ is the original, or “mother” wavelet, a is the scale factor, and b is the transform factor.

5.1 THE CONTINOUS WAVELET TRANSFORM

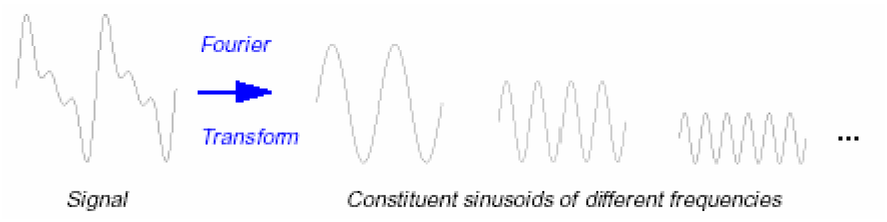
Mathematically, the process of fourier analysis is represented by the Fourier transform:

$$F(w) = \int_{\infty}^{-\infty} f(t)e^{-jat}\,dt$$

which is the sum over all time of the signal *f(t)* multiplied by a complex exponential.

(Complex exponential can be broken down into real and imaginary sinusoidal components).

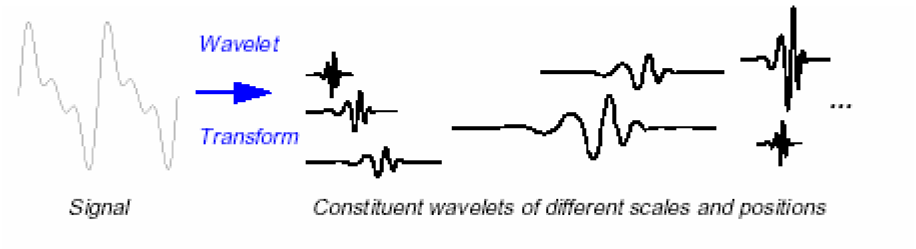
The results of the transform are the fourier coefficients, which when multiplied by a sinusoid of frequency, yield the constituent sinusoidal components of the original signal. Graphically, the process looks like:



Similarly, the continuous wavelet transform (CWT) is defined as the sum over all time of the signal multiplied by scaled, shifted versions of the wavelet function :

$C(scale, position) = \int_{-\infty}^{\infty} f(t)\psi(scale, position, t)dt$ The result of the CWT are many wavelet coefficients C, which are a function of scale and position.

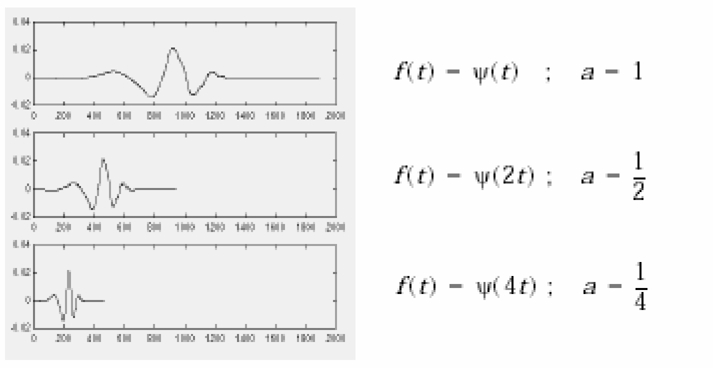
Multiplying each coefficient by the appropriately scaled and shifted wavelet yields the constituent wavelets of the original signal:



Scaling

Wavelet analysis produces a time-scale view of a signal. Scaling a wavelet simply means stretching (or compressing) it.

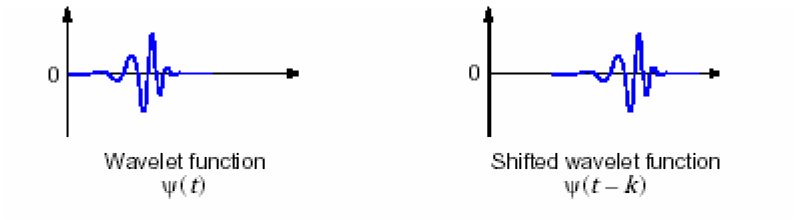
The smaller the scale factor, the more "compressed" the wavelet.



For a sinusoid the scale factor is related (inversely) to the radian frequency. Similarly, with wavelet analysis, the scale is related to the frequency of the signal.

Shifting

Shifting a wavelet simply means delaying its onset. Mathematically, delaying a function by k is represented by *f(t-k)* :



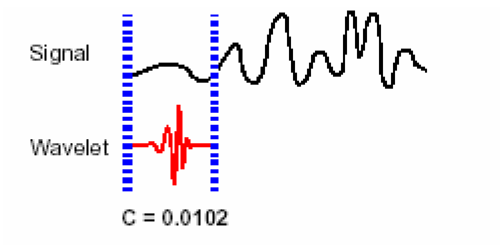
Five Easy Steps to a Continuous Wavelet Transform

The continuous wavelet transform is the sum over all time of the signal multiplied by scaled, shifted versions of the wavelet. This process produces wavelet coefficients that are a function of scale and position. In fact, there are the five steps for creating a CWT:

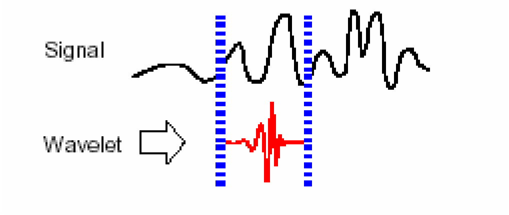
1 A wavelet is taken and compared it to a section at the start of the original signal.

2 A number C is calculated that represents how closely the wavelet is correlated with this section of the signal. The higher C is, the more the similarity. If the signal energy and the wavelet energy are equal to one, C may be interpreted as a correlation coefficient.

The results will depend on the shape of the wavelet choose.



3 The wavelet is shifted to the right and steps 1 and 2 are repeated until the whole signal is covered.

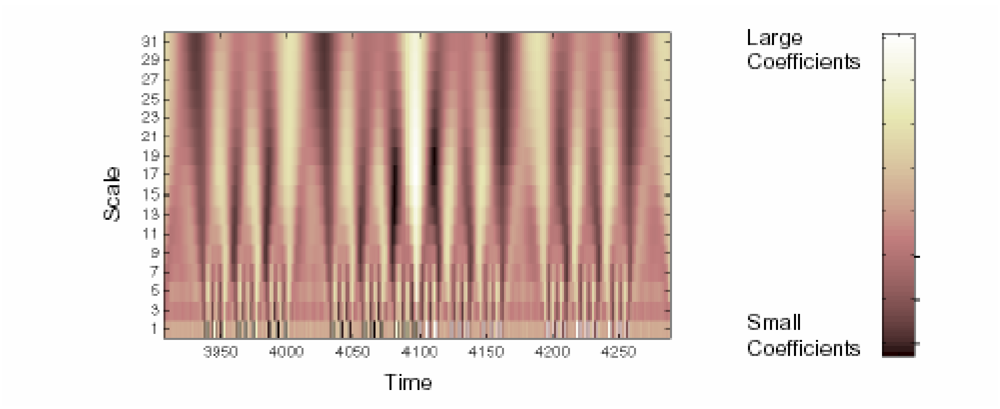


4 The wavelet is scaled (stretched) and steps 1 through 3 are repeated.

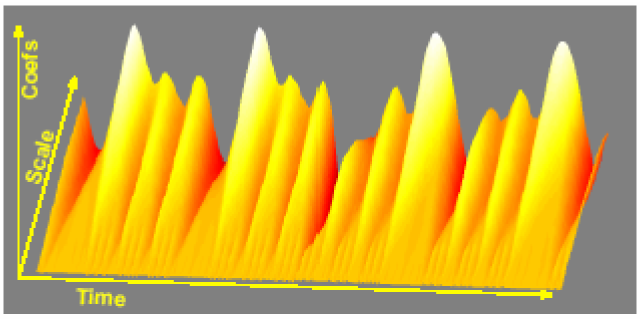
5 Steps 1 through 4 are repeated for all scales.

Now there are coefficients produced at different scales by different sections of the signal. The coefficients constitute the results of a regression of the original signal performed on the wavelets.

The plot of coefficients is made where the x-axis represents position along the signal (time), the y-axis represents scale, and the color at each x-y point represents the magnitude of the wavelet coefficient C.



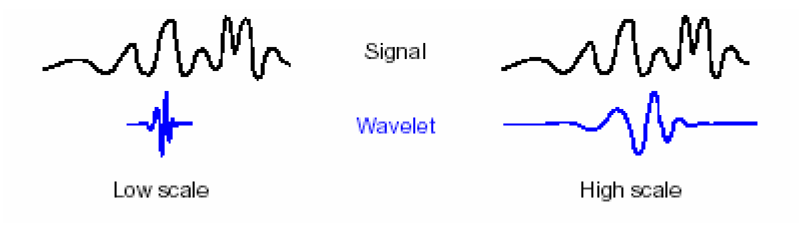
These coefficient plots resemble a bumpy surface.



The continuous wavelet transform coefficient plots are the time-scale view of the signal. It is a different view of signal data than the time-frequency fourier view, but it is not unrelated.

Scale and Frequency

The scales in the coefficients plot (shown as *y*-axis labels) run from 1 to 31. The higher scales correspond to the most "stretched" wavelets. The more stretched the wavelet, the longer the portion of the signal with which it is being compared, and thus the coarser the signal features being measured by the wavelet coefficients.



Thus, there is a relation between wavelet scales and frequency as

•low scale ⇒compressed wavelet ⇒ rapidly changing details ⇒high frequency .

•high scale⇒stretched wavelet⇒slowly changing coarse features⇒low frequency .

5.2 THE DISCRETE WAVELET TRANSFORM

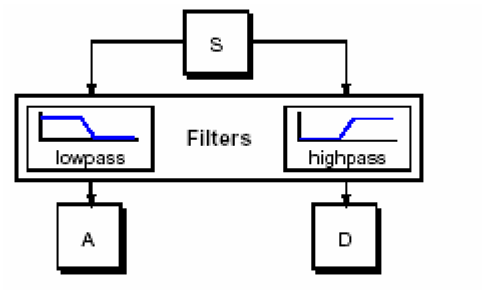
Calculating wavelet coefficients at every possible scale is a very large amount of work. Scales and positions are selected based on powers of two called dyadic scales and positions. This analysis will be much more efficient and accurate. Such analysis is obtained from the discrete wavelet transform (DWT). An efficient way to implement this scheme using filters was developed in 1988 by Mallat.

In the discrete wavelet transform, an image signal can be analyzed by passing it through an analysis filter bank followed by a decimation operation. This analysis filter bank, which consists of a low pass and a high pass filter at each decomposition stage, is commonly used in image compression. When a signal passes through these filters, it is split into two bands. The low pass filter, which corresponds to an averaging operation, extracts the coarse information of the signal. The high pass filter, which corresponds to a differencing operation, extracts the detail information of the signal. The output of the filtering operations is then decimated by two.

5.2.1 ONE-STAGE FILTERING: APPROXIMATIONS AND DETAILS

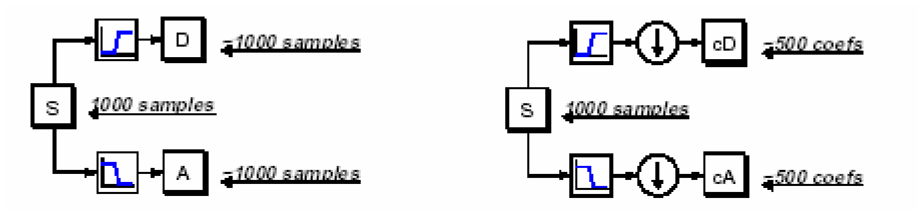
For many signals, the low-frequency content is the most important part. The high-frequency content, on the other hand, imparts flavor or nuance. Consider the human voice. If the high- frequency components are removed, the voice sounds different, but still it can tell what's being said. However, if the low-frequency components are removed, only gibberish is heard.

Wavelet analysis consists of approximations and details. The approximations are the high-scale, low- frequency components of the signal. The filtering process, at its most basic level, looks like this:



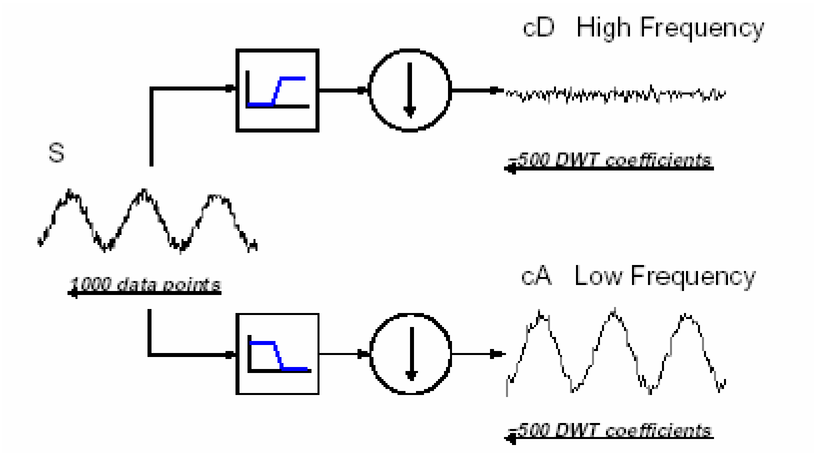
The original signal, S, passes through two complementary filters and emerges as two signals. If the original signal S consists of 1000 samples of data then the resulting signals will each have 1000 samples, for a total of 2000.

Hence down sampling is performed to maintain the same length a original one which will produce two sequences called cA and cD.



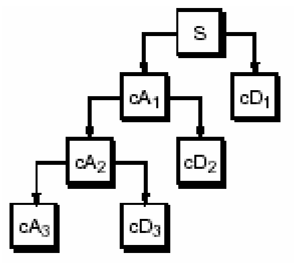
The process on the right, which includes down sampling, produces DWT coefficients. To gain a better appreciation of this process, a one-stage discrete wavelet transform of a signal is performed. Our signal is a pure sinusoid with high-frequency noise added to it.

Schematic diagram is shown with real signals inserted into it:

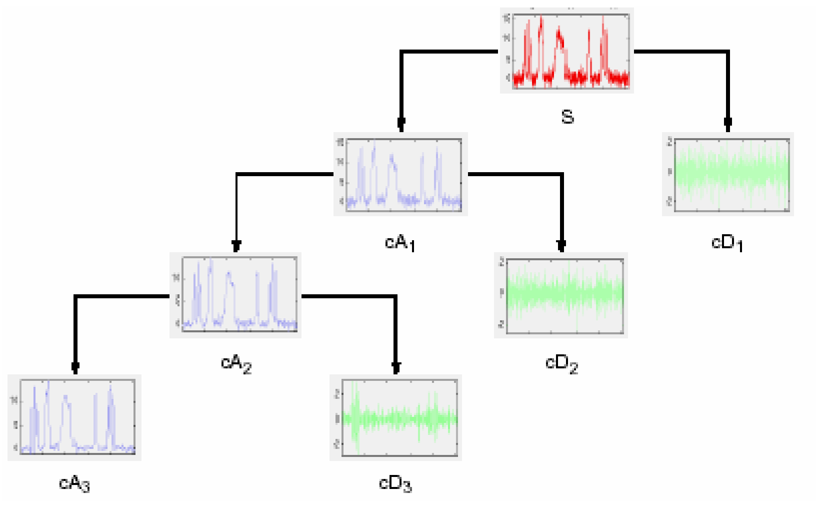


5.2.2 MULTIPLE - LEVEL DECOMPOSITION

The decomposition process can be iterated, with successive approximations being decomposed in turn, so that one signal is broken down into many lower resolution components. This is called the wavelet decomposition tree.

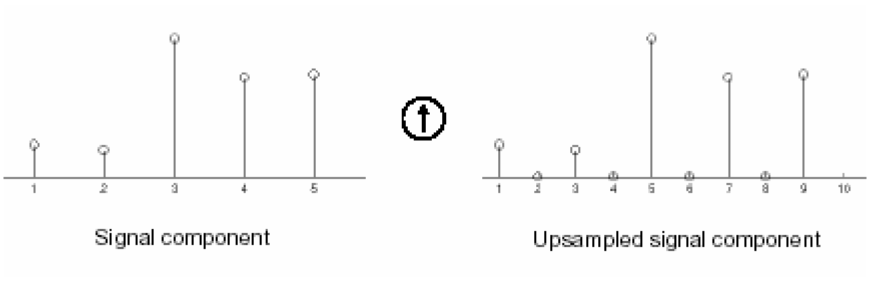


Signal’s wavelet decomposition tree can yield valuable information.



5.2.3 WAVELET RECONSTRUCTION

Where wavelet analysis involves filtering and down sampling, the wavelet reconstruction process consists of up-sampling and filtering. Up-sampling is the process of lengthening a signal component by inserting zeros between samples:



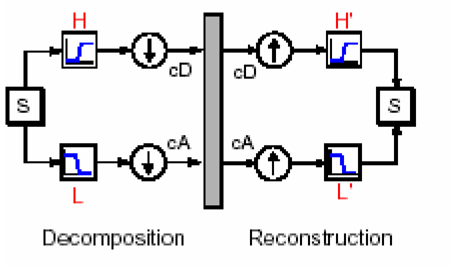
The Wavelet toolbox includes commands, like idwt and waverec, that perform single-level or multilevel reconstruction, respectively, on the components of one-dimensional signals.

5.2.4 RECONSTRUCTION FILTERS

The filtering part of the reconstruction process is important because it is the choice of filters that is crucial in achieving perfect reconstruction of the original signal.

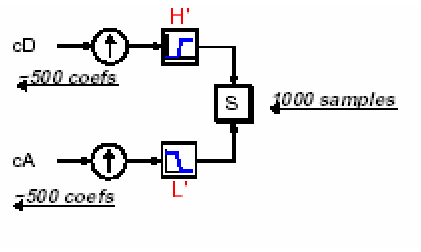
The down sampling of the signal components performed during the decomposition phase introduces a distortion called aliasing. By carefully choosing filters for the decomposition and reconstruction phases that are closely related (but not identical), the effects of aliasing can be cancelled out.

The low and high pass decomposition filters (L and H), together with their associated reconstruction filters (L' and H'), form a system of what is called quadrature mirror filters:

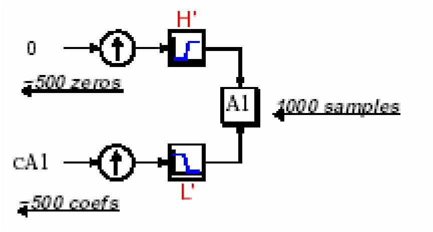


5.2.5 RECONSTRUCTING APPROXIMATIONS AND DETAILS

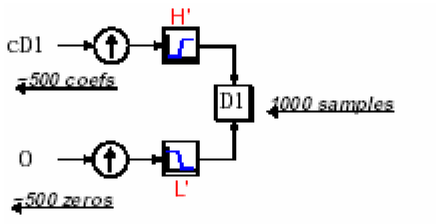
It is observed that it is possible to reconstruct our original signal from the coefficients of the approximations and details.



It is also possible to reconstruct the approximations and details themselves from their coefficient vectors. As an example, the first-level approximation A1 can be reconstructed from the coefficient vector cA1 by passing the coefficient vector cA1 through the same process used to reconstruct the original signal. However, instead of combining it with the level-one detail cD1, a vector of zeros is feeded in place of the detail coefficients vector:



The process yields a reconstructed approximation A1, which has the same length as the original signal S and which is a real approximation of it.
Similarly, the first-level detail D1 is reconstructed using the analogous process:

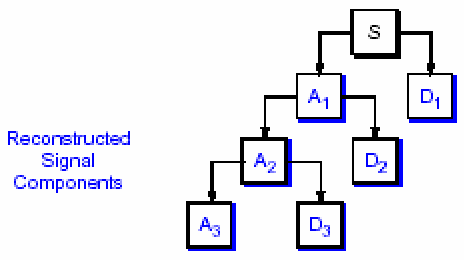


The reconstructed details and approximations are true constituents of the original signal. In fact, we find when we combine them that:

A1+D1 = S

The coefficient vectors cA1 and cD1 were produced by down sampling and are only half the length of the original signal cannot directly be combined to reproduce the signal. It is necessary to reconstruct the approximations and details before combining them.

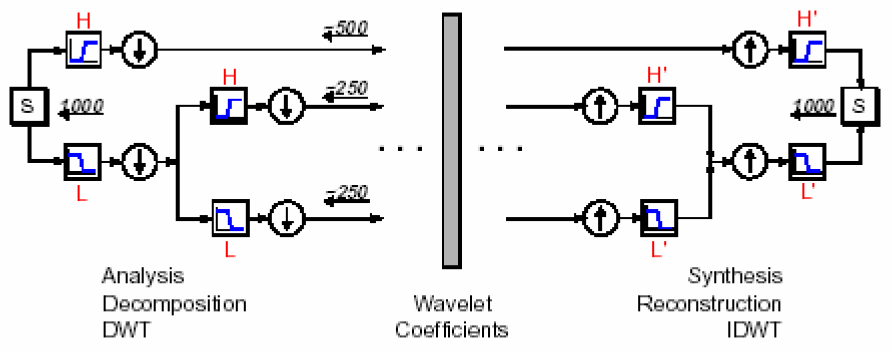
Extending this technique to the components of a multilevel analysis, we find that similar relationships hold for all the reconstructed signal constituents. That is, there are several ways to reassemble the original signal:



$$
\begin{aligned}
S &= A_1 + D_1 \\
&= A_2 + D_2 + D_1 \\
&= A_3 + D_3 + D_2 + D_1
\end{aligned}
$$

5.2.6 MULTISTEP DECOMPOSITION AND RECONSTRUCTION

A multi-step analysis-synthesis process can be represented as:

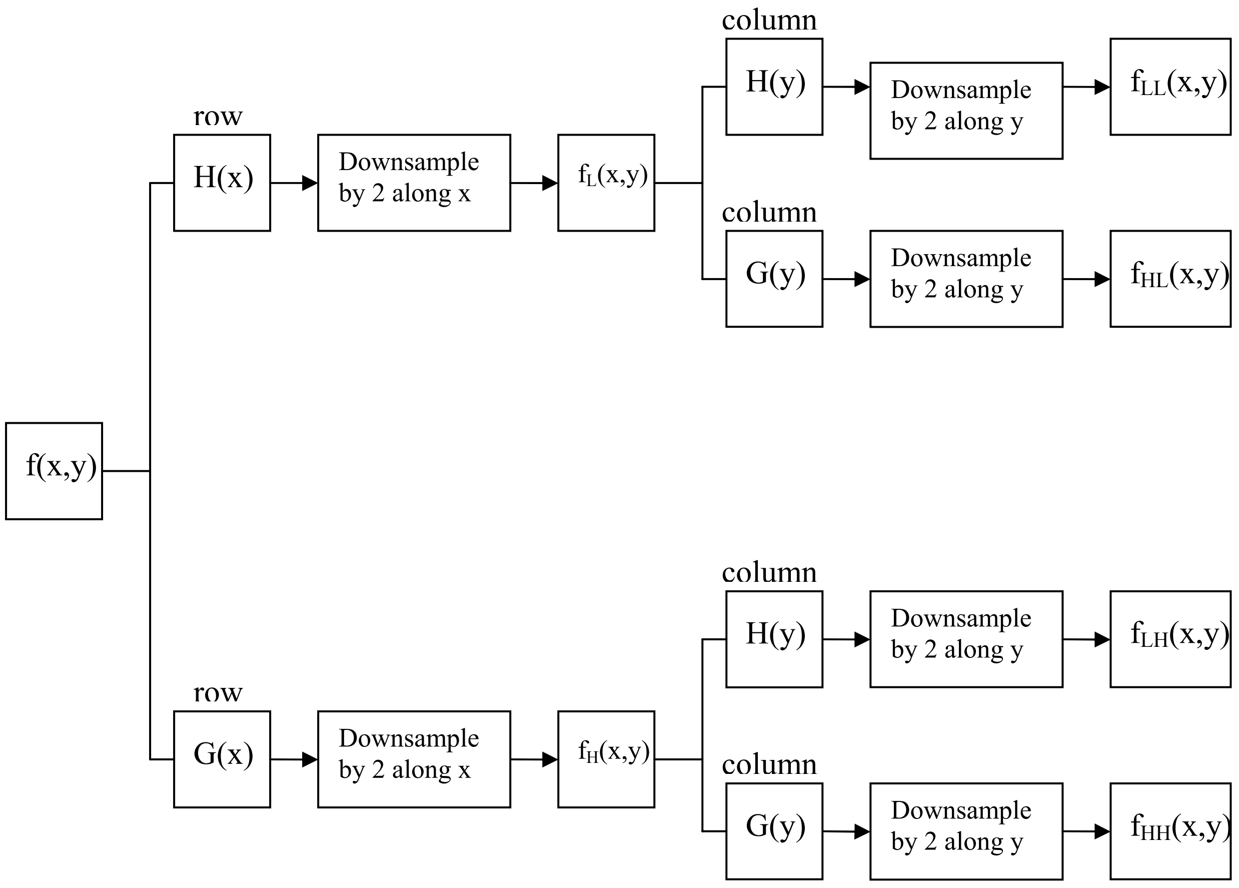


This process involves two aspects: breaking up a signal to obtain the wavelet coefficients, and reassembling the signal from the coefficients.

Decomposition and reconstruction at same length are discussed. There is no use of breaking up a signal to have the satisfaction of immediately reconstructing it. The wavelet coefficients can modify before performing the reconstruction step. Wavelet analysis is performed because the coefficients thus obtained have many known uses, de-noising and compression being most important among them.

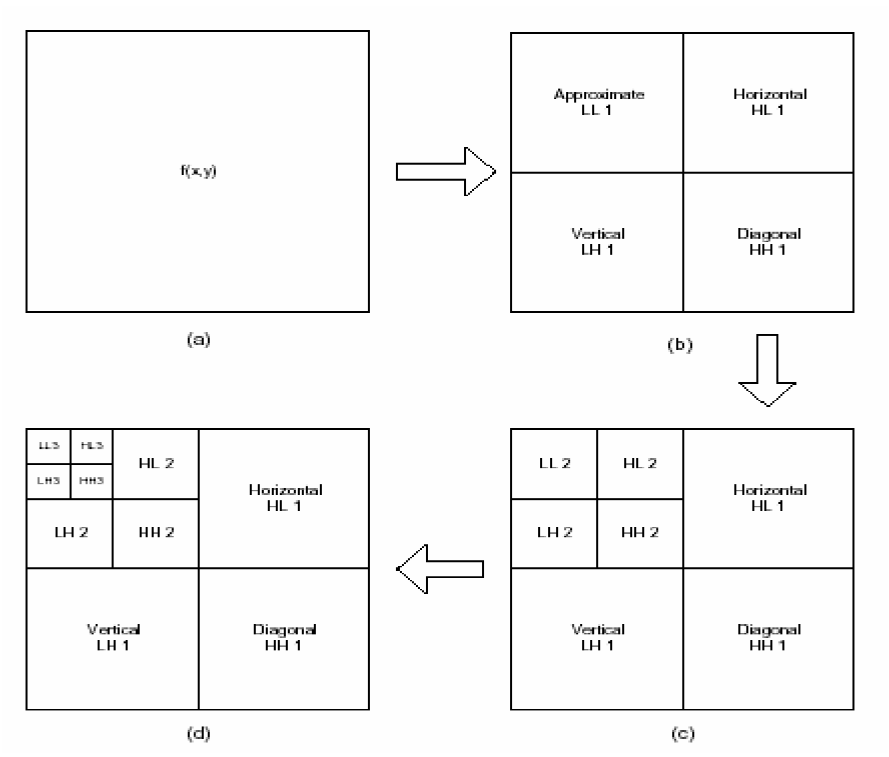
5.2.7 TWO DIMENTIONAL DISCRETE WAVELET TRANSFORM

A two-dimensional transform can be accomplished by performing two separate one-dimensional transforms. First, the image is filtered along the x-dimension and decimated by two. Then, it is followed by filtering the sub-image along the y-dimension and decimated by two. Finally, we have split the image into four bands denoted by LL, HL, LH and HH after one-level de-composition .



Further decompositions can be achieved by acting upon the LL sub-band successively and the resultant image is split into multiple bands.

The figure shows two Dimensional Discrete Wavelet Transform with original image, one level de-composition, two level decompositions and three level decompositions.



In mathematical terms, the averaging operation or low pass filtering is the inner product between the signal and the scaling function Φ whereas the differencing operation or high pass filtering is the inner product between the signal and the wavelet function Ψ .

Average coefficients,

$$c_j(k) = < f(t), \phi_{j,k}(t) > = \int f(t), \phi_{j,k}(t) dt$$

Detail coefficients,

$$d_j(k) = < f(t), \psi_{j,k}(t) > = \int f(t), \psi_{j,k}(t) dt$$

The scaling function or the low pass filter is defined as

$$\phi_{j,k}(t) = 2^{j/2} \phi(2^j t - k)$$

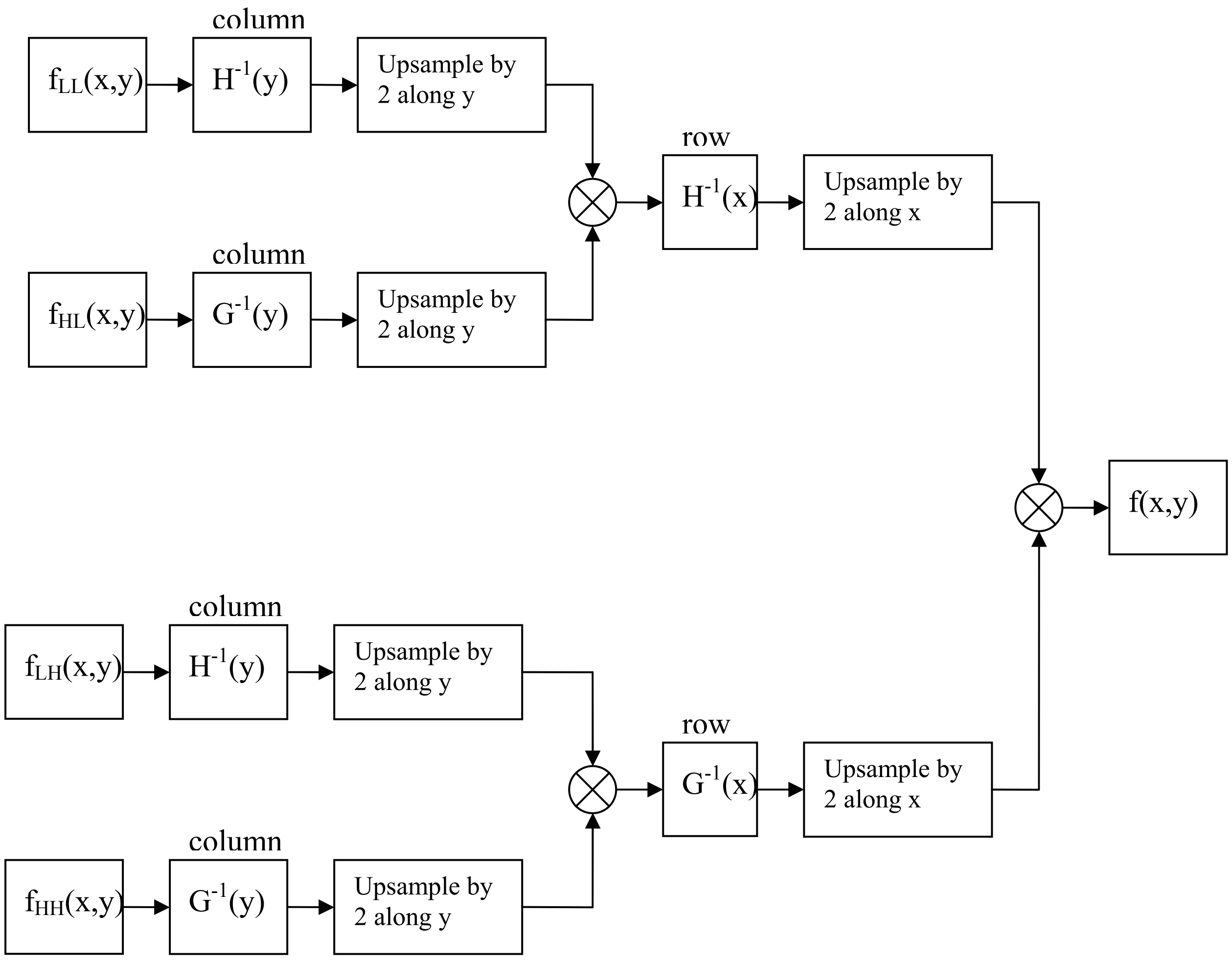
The wavelet function or the high pass filter is defined as

$$\psi_{j,k}(t) = 2^{j/2} \psi(2^j t - k)$$

where j denotes the discrete scaling index
k denotes discrete translation index

5.2.8 TWO DIMENSIONAL INVERSE DISCRETE WAVELET TRANSFORM

The reconstruction of the image can be carried out by the following procedure. First, upsampling is done by a factor of two on all the four sub-bands at the coarsest scale, and filter the sub-bands in each dimension. Then the four filtered sub bands are added to reach the low-low sub-band at the next finer scale. This process is repeated until the image is fully reconstructed.



CHAPTER 6

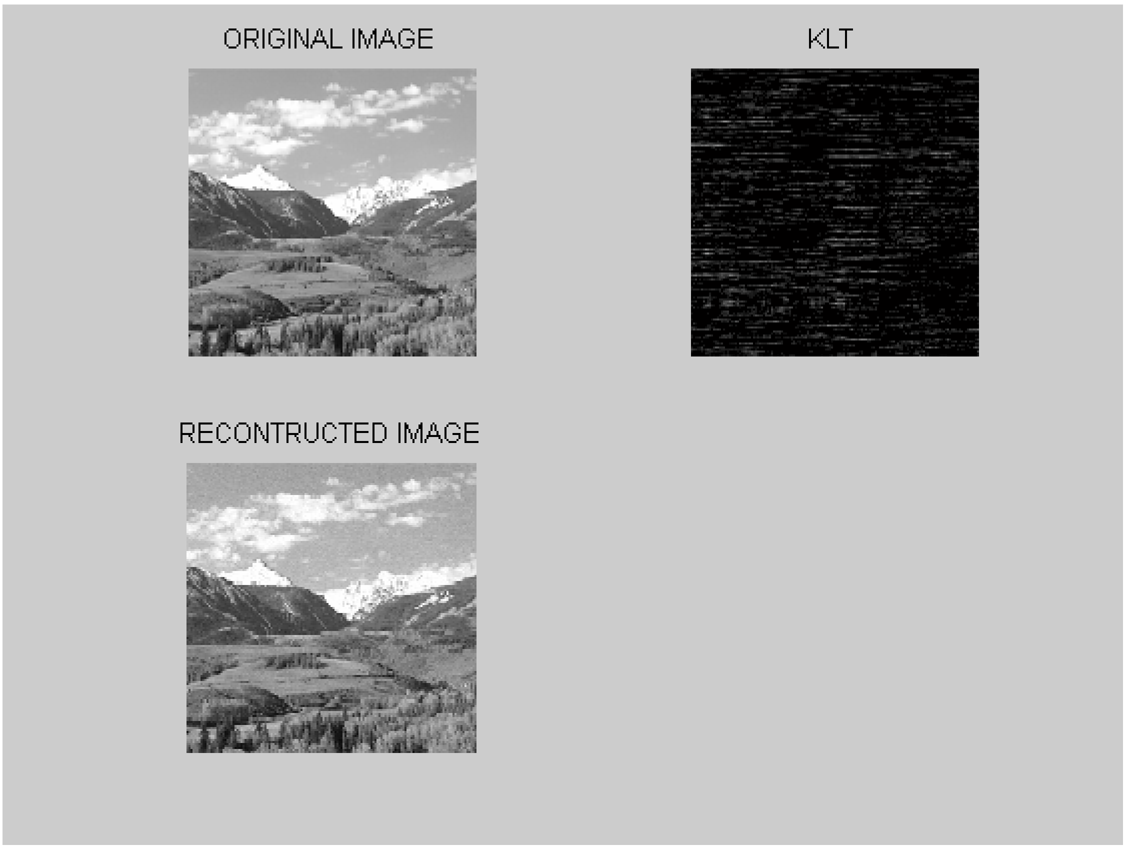
SIMULATION RESULTS

The results presented in this section are obtained using gray & color images using MATLAB software & a comprehensive comparison is made between data compression transforms i.e. the Karhunen-Loeve Transform (KLT), the Discrete Cosine Transform (DCT) and the Discrete Wavelet Transform (DWT). The simulation results are based on PSNR, Computational complexity & Image quality of the reconstructed image.

6.1 VISUAL RESULTS FOR GRAY IMAGE

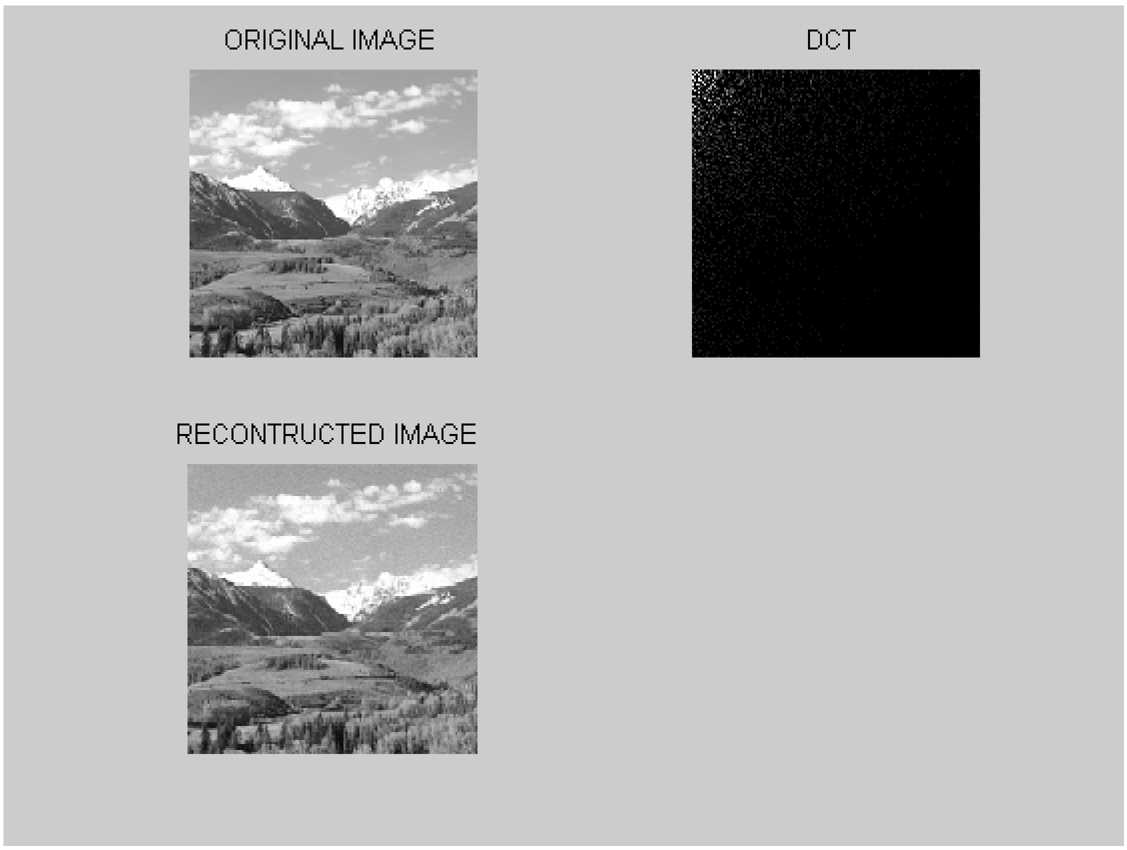
OUTPUT FILE OF GRAY IMAGE USING KLT

Original image size (bytes) = 65536
Compressed image size (bytes) = 16291
Decompressed image size (bytes) = 65536
Compression Ratio = 75%
Signal to Noise Ratio (db) =30.92
Simulation Time (Seconds) = 3.68



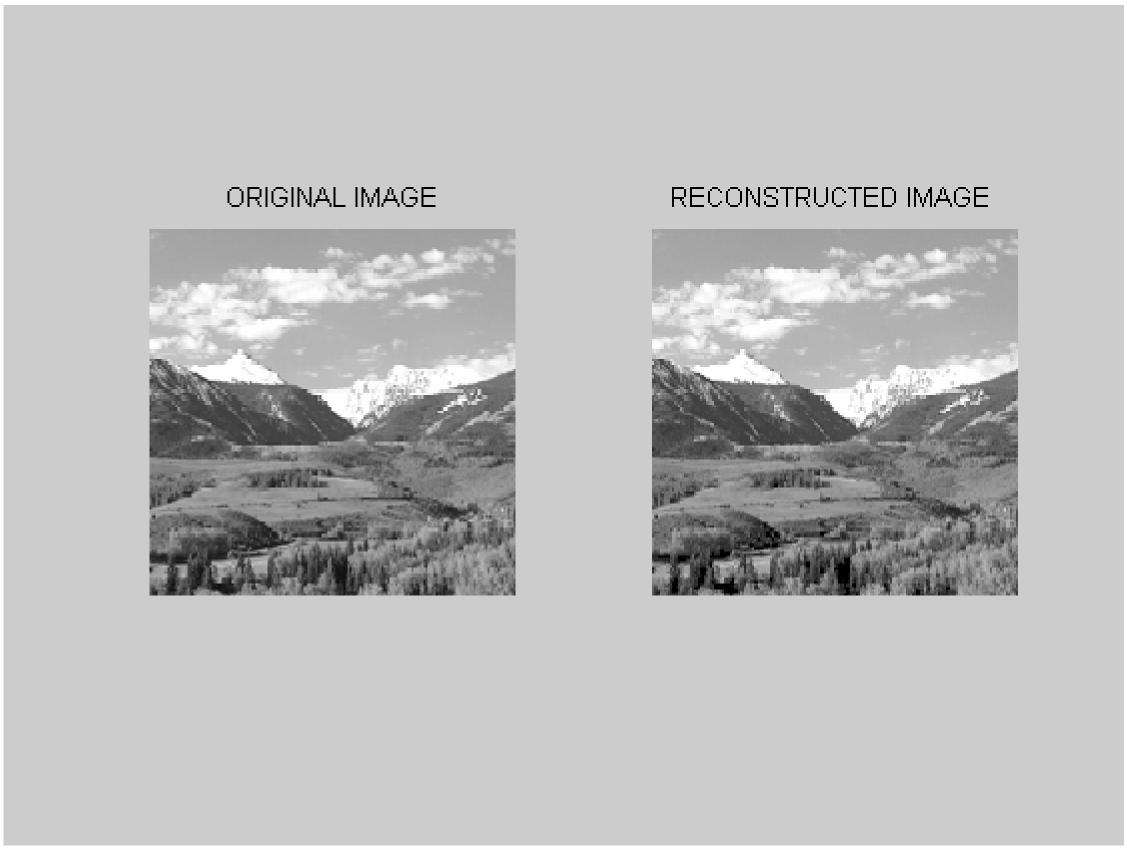
OUTPUT FILE OF GRAY IMAGE USING DCT

Original image size (bytes) = 65536
Compressed image size (bytes) = 16264
Decompressed image size (bytes) = 65536
Compression Ratio = 75%
Signal to Noise Ratio(db) =29.26
Simulation Time (Seconds) = 1.10



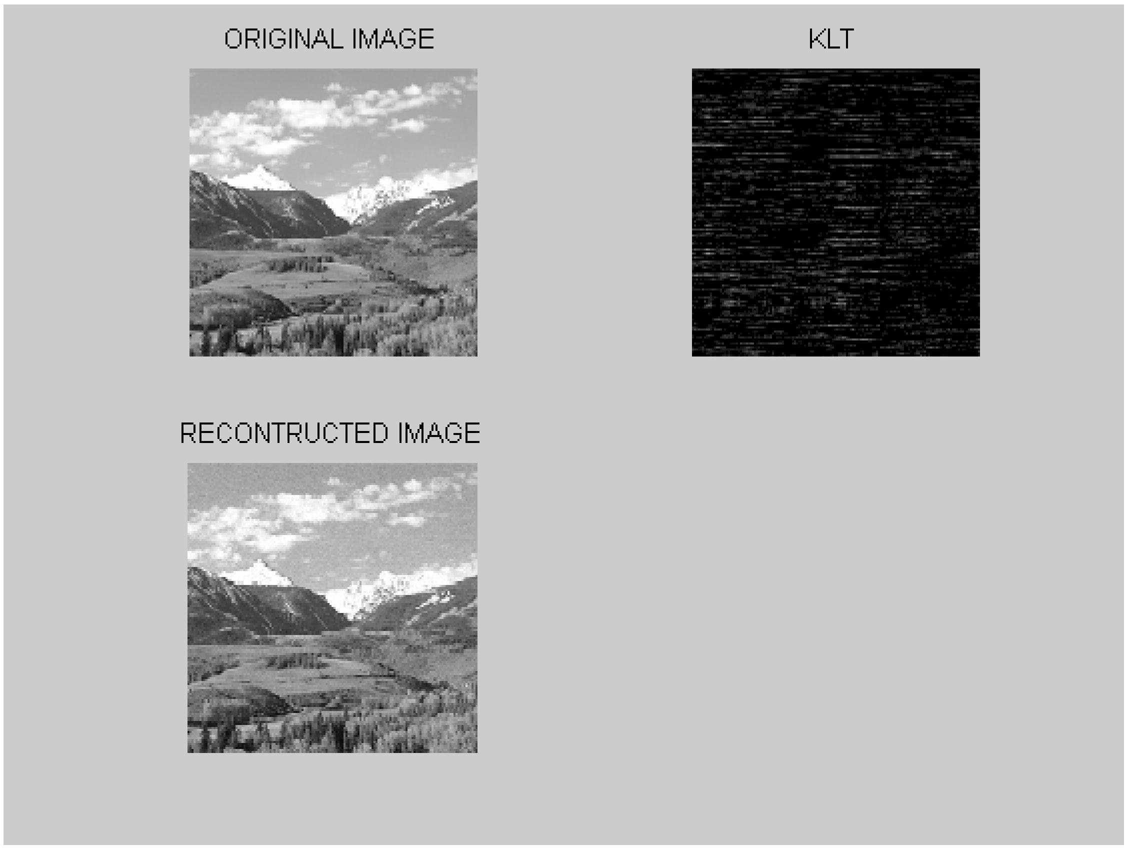
OUTPUT FILE OF GRAY IMAGE USING DWT

```
Original image size (bytes) =  65536
Compressed image size (bytes) =  16291
Decompressed image size (bytes) =  65536
Compression Ratio = 75%
Signal to Noise Ratio (db)  =31.24
Simulation Time (Seconds) = 1.31
```



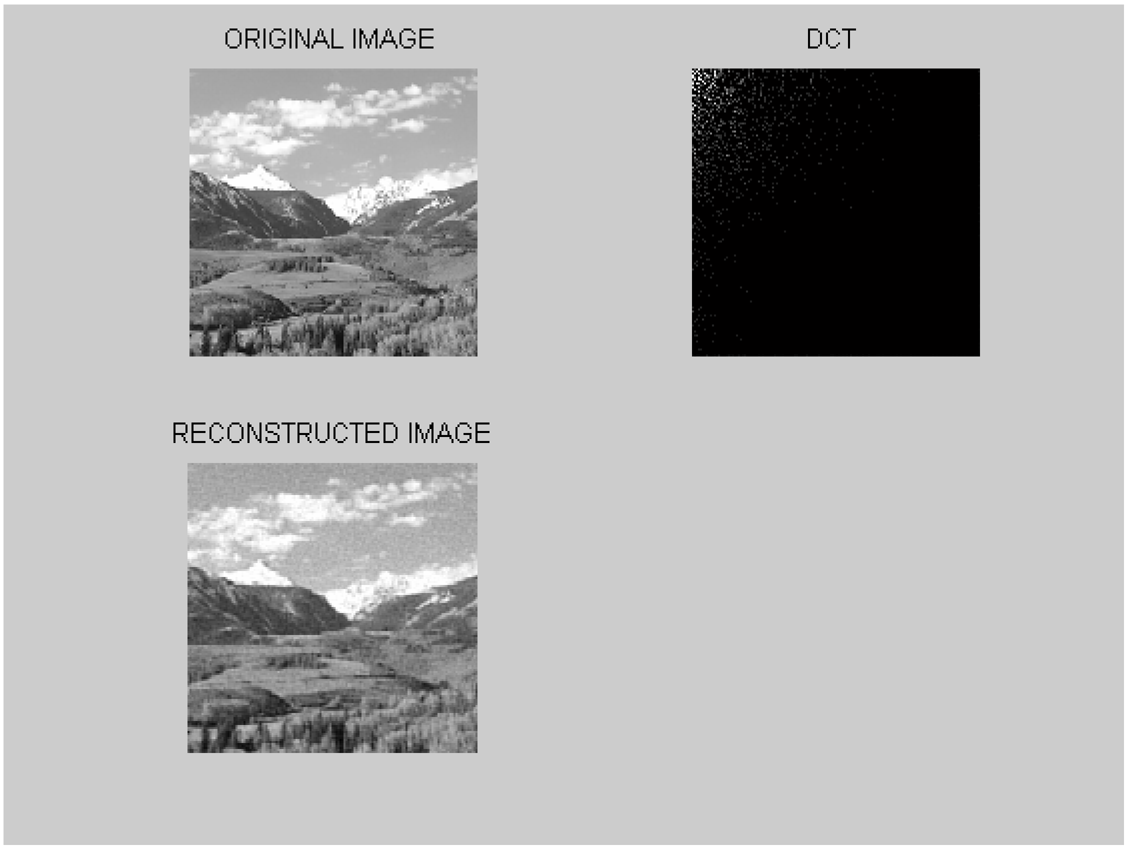
OUTPUT FILE OF GRAY IMAGE USING KLT

Original image size (bytes) = 65536
Compressed image size (bytes) = 9682
Decompressed image size (bytes) = 65536
Compression Ratio = 85%
Signal to Noise Ratio (db) =29.23
Simulation Time (Seconds) = 3.79



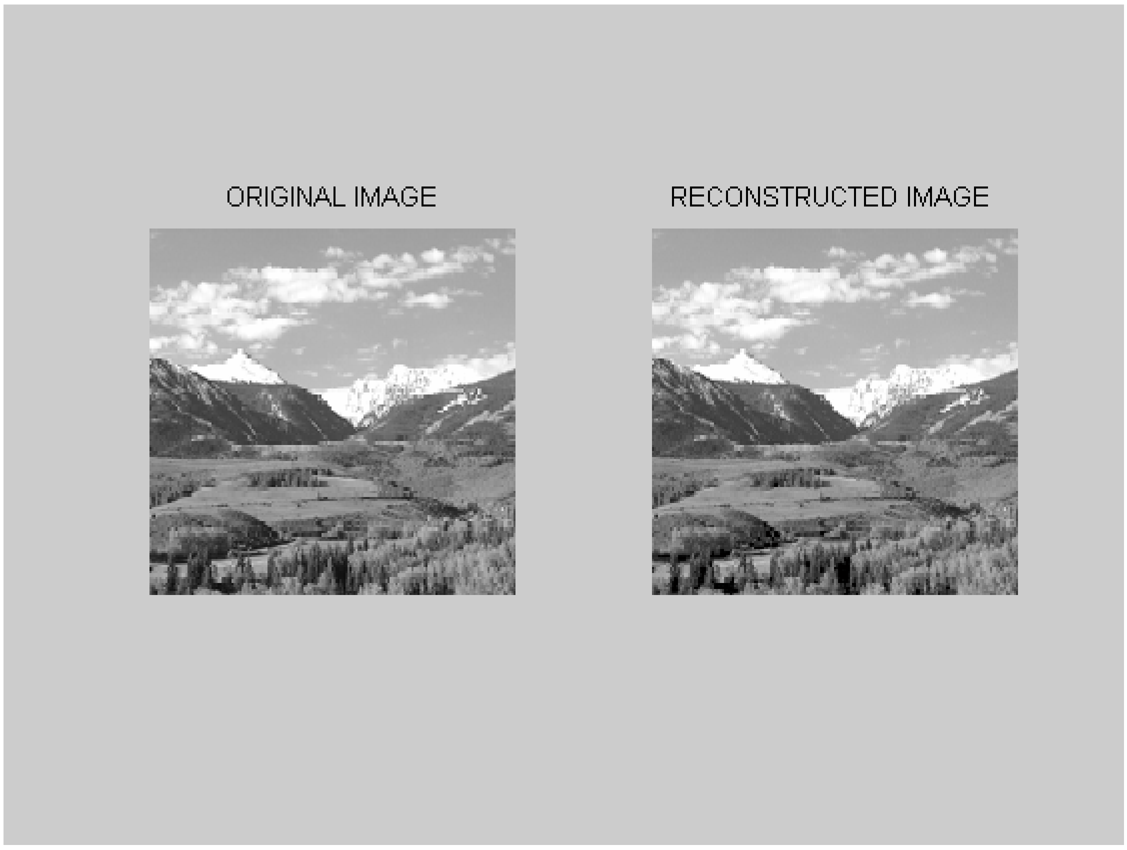
OUTPUT FILE OF GRAY IMAGE USING DCT

Original image size (bytes) = 65536
Compressed image size (bytes) = 9765
Decompressed image size (bytes) = 65536
Compression Ratio = 85%
Signal to Noise Ratio (db) =26.32
Simulation Time (Seconds) = 1.15



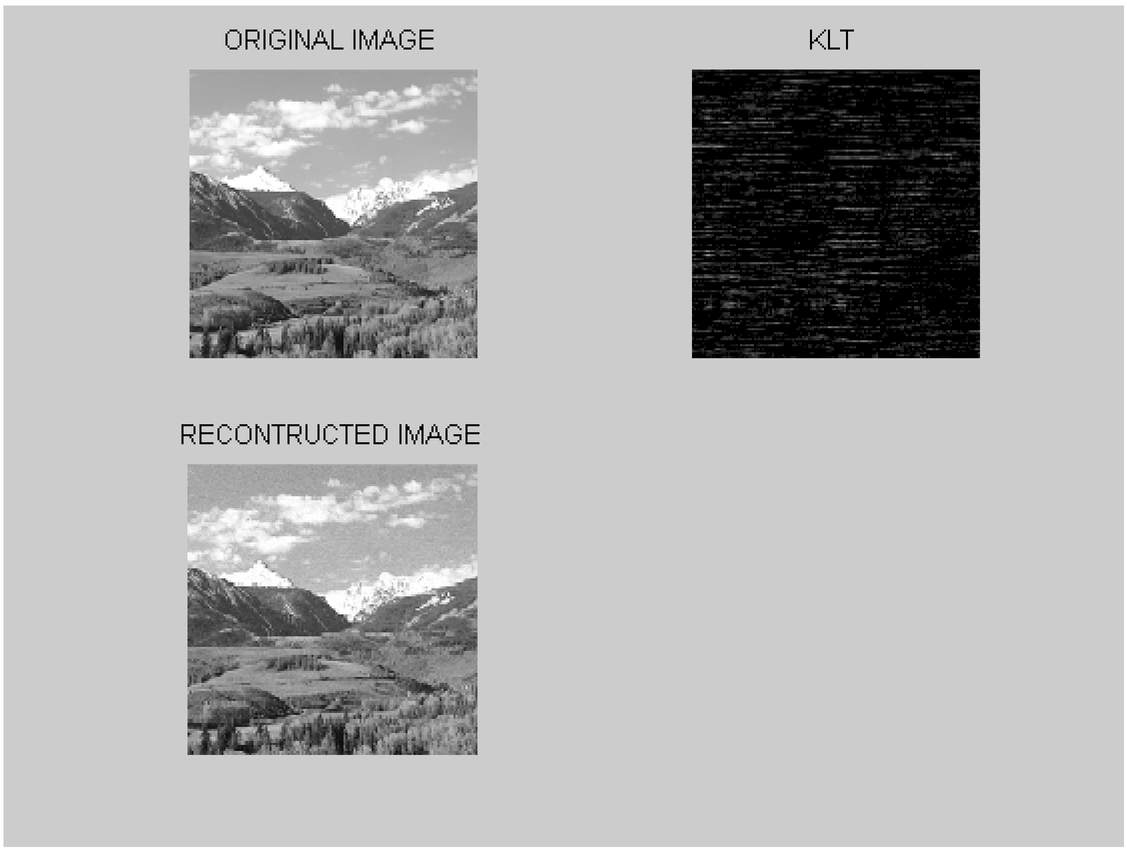
OUTPUT FILE OF GRAY IMAGE USING DWT

```
Original image size (bytes) =  65536
Compressed image size (bytes) =  9682
Decompressed image size (bytes) =  65536
Compression Ratio = 85%
Signal to Noise Ratio (db)  =30.29
Simulation Time (Seconds) = 1.33
```



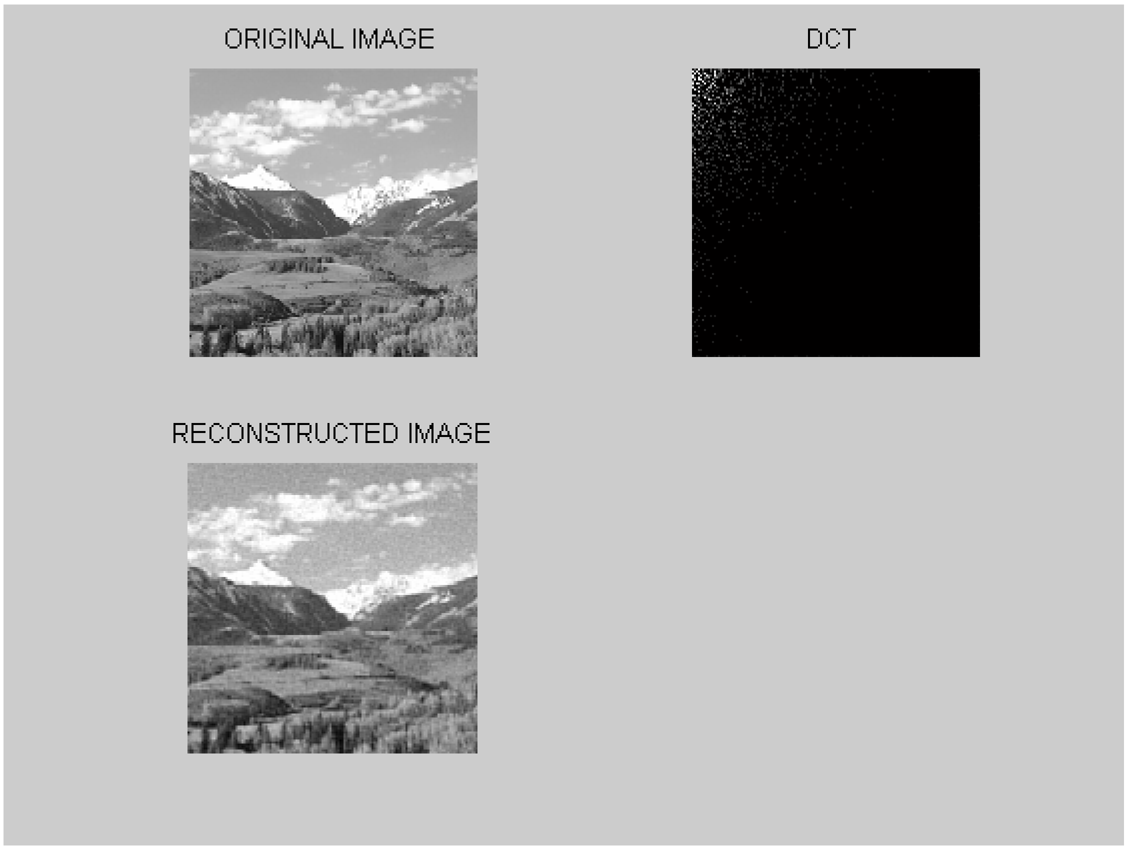
OUTPUT FILE OF GRAY IMAGE USING KLT

Original image size (bytes) = 65536
Compressed image size (bytes) = 6471
Decompressed image size (bytes) = 65536
Compression Ratio = 90%
Signal to Noise Ratio (db) =28.47
Simulation Time (Seconds) = 3.95



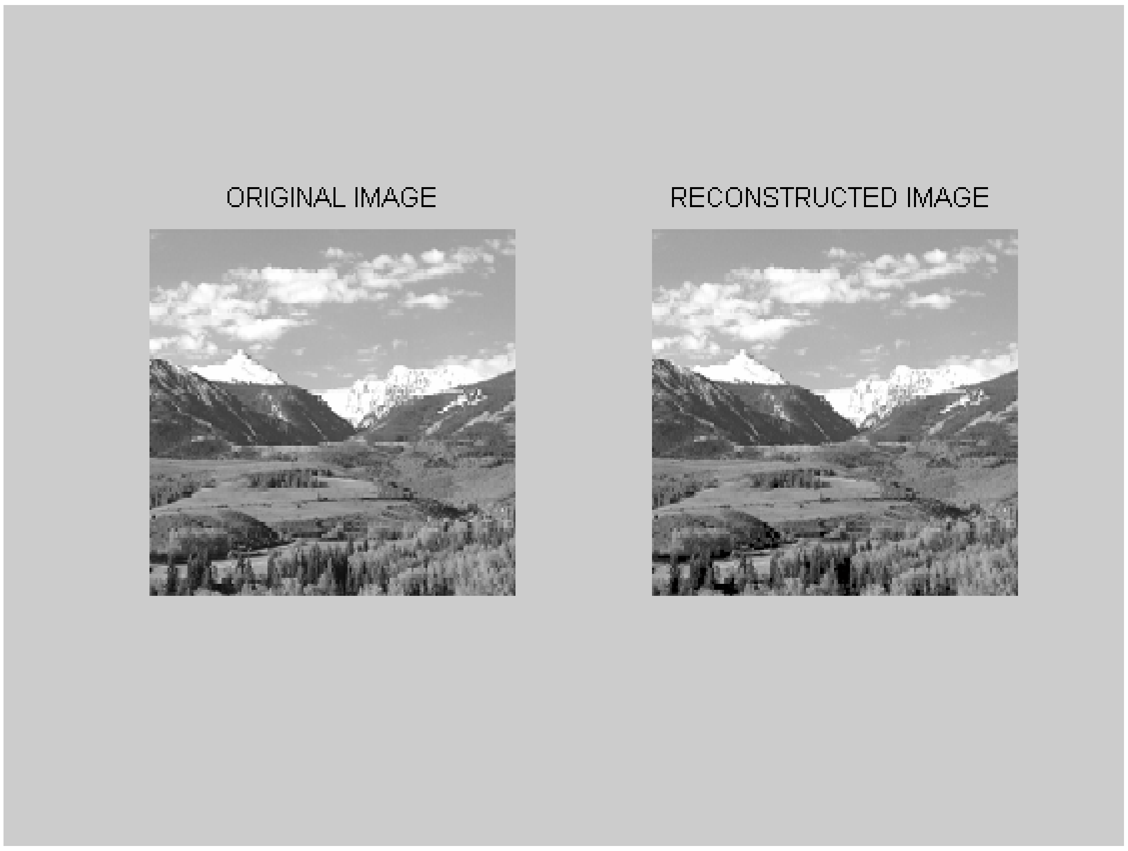
OUTPUT FILE OF GRAY IMAGE USING DCT

```
Original image size (bytes) =  65536
Compressed image size (bytes) =  6382
Decompressed image size (bytes) =  65536
Compression Ratio = 90%
Signal to Noise Ratio (db)  =24.49
Simulation Time (Seconds) = 1.21
```

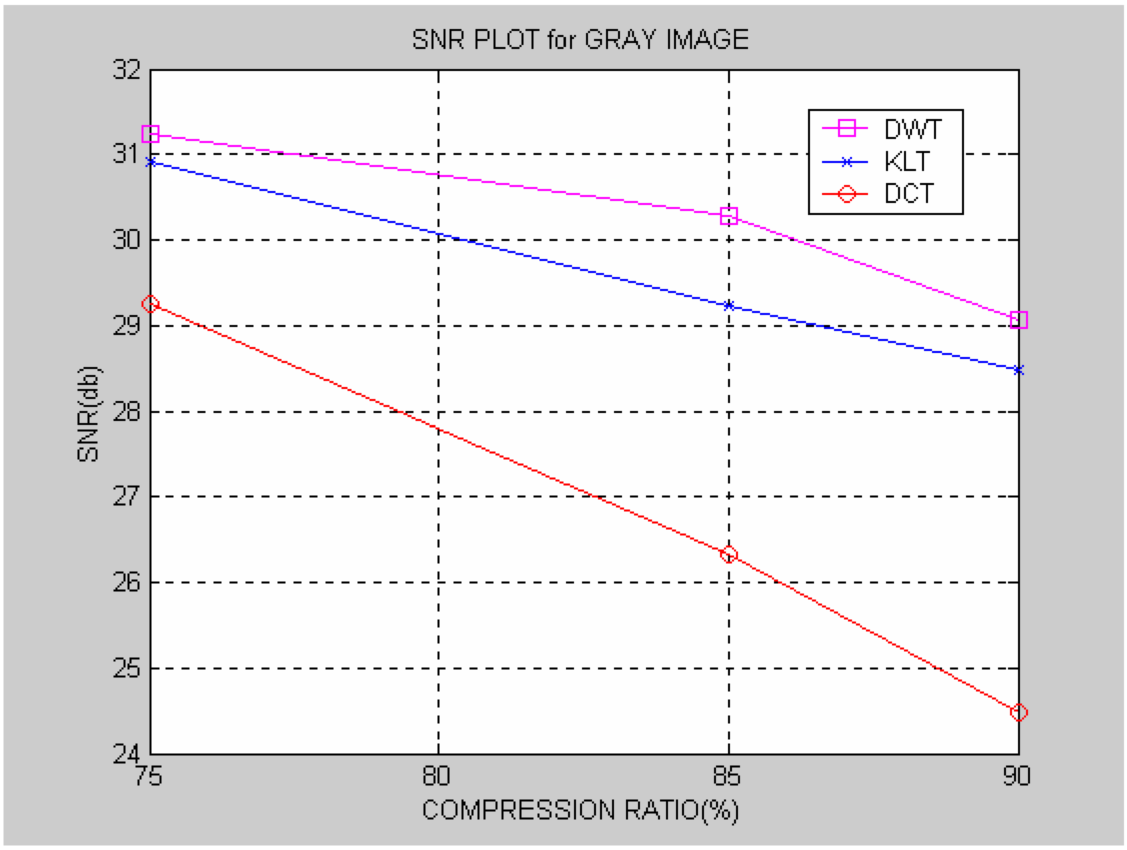


OUTPUT FILE OF GRAY IMAGE USING DWT

```
Original image size (bytes) =  65536
Compressed image size (bytes) =  6479
Decompressed image size (bytes) =  65536
Compression Ratio = 90%
Signal to Noise Ratio (db)  =29.06
Simulation Time (Seconds) = 1.34
```



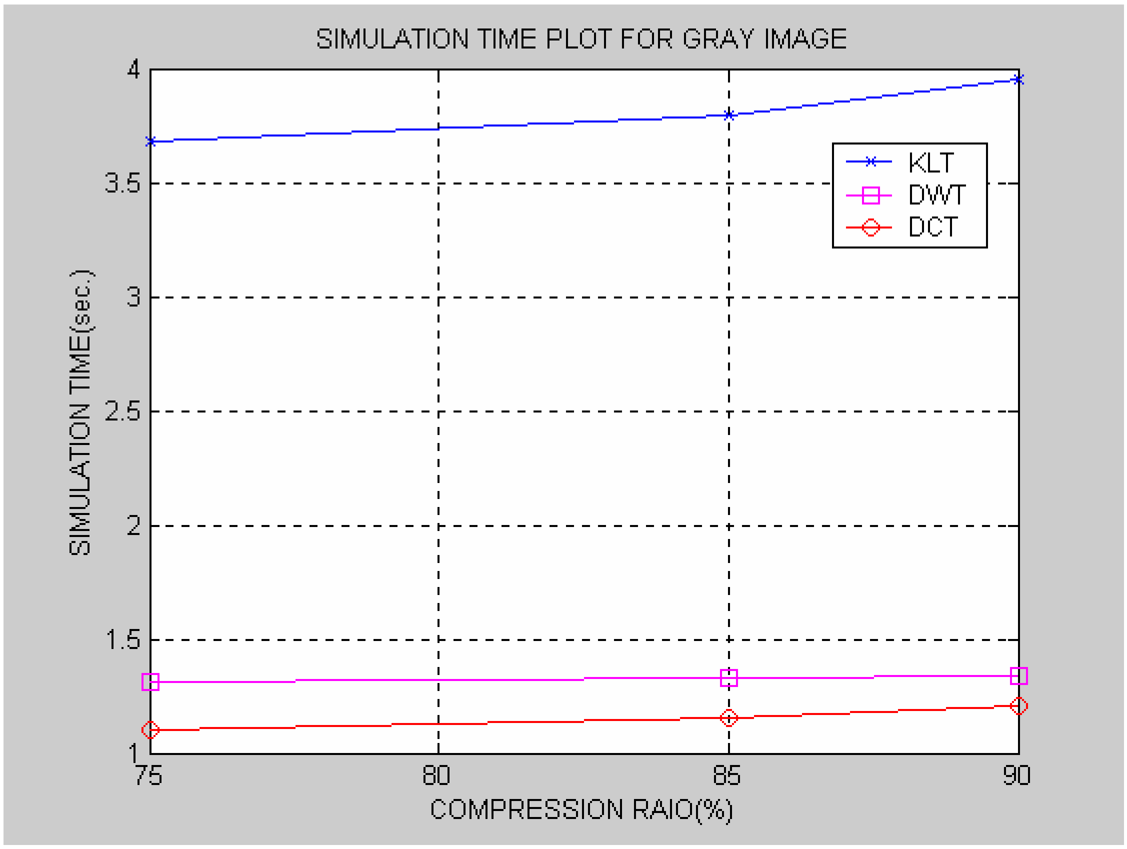
6.1.1 SNR PERFORMANCE FOR GRAY IMAGE



	CR = 75%	CR = 85%	CR = 90%
DWT	31.24	30.29	29.06
KLT	30.92	29.23	28.47
DCT	29.26	26.32	24.49

Table 6.1 : SNR Values For Gray Image (All values in db)

6.1.2 SIMULATION TIME PERFORMANCE FOR GRAY IMAGE



	CR = 75%	CR = 85%	CR = 90%
KLT	3.68	3.79	3.95
DWT	1.31	1.33	1.34
DCT	1.1	1.15	1.21

Table 6.2 : Simulation Time For Gray Image (All values in seconds)

6.2 VISUAL RESULTS FOR COLOR IMAGE

OUTPUT FILE OF COLOR IMAGE USING KLT

Original image size (bytes) = 196608

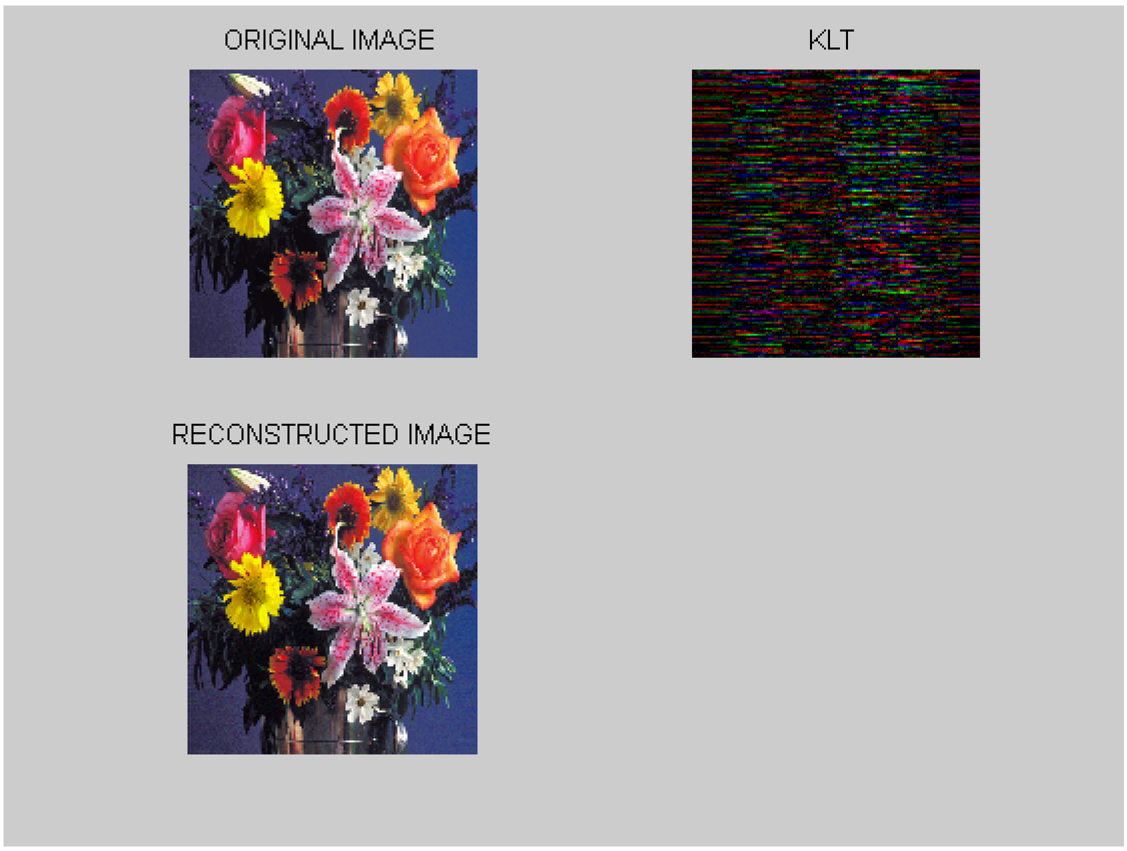
Compressed image size (bytes) = 58761

Decompressed image size (bytes) = 196608

Compression Ratio = 70%

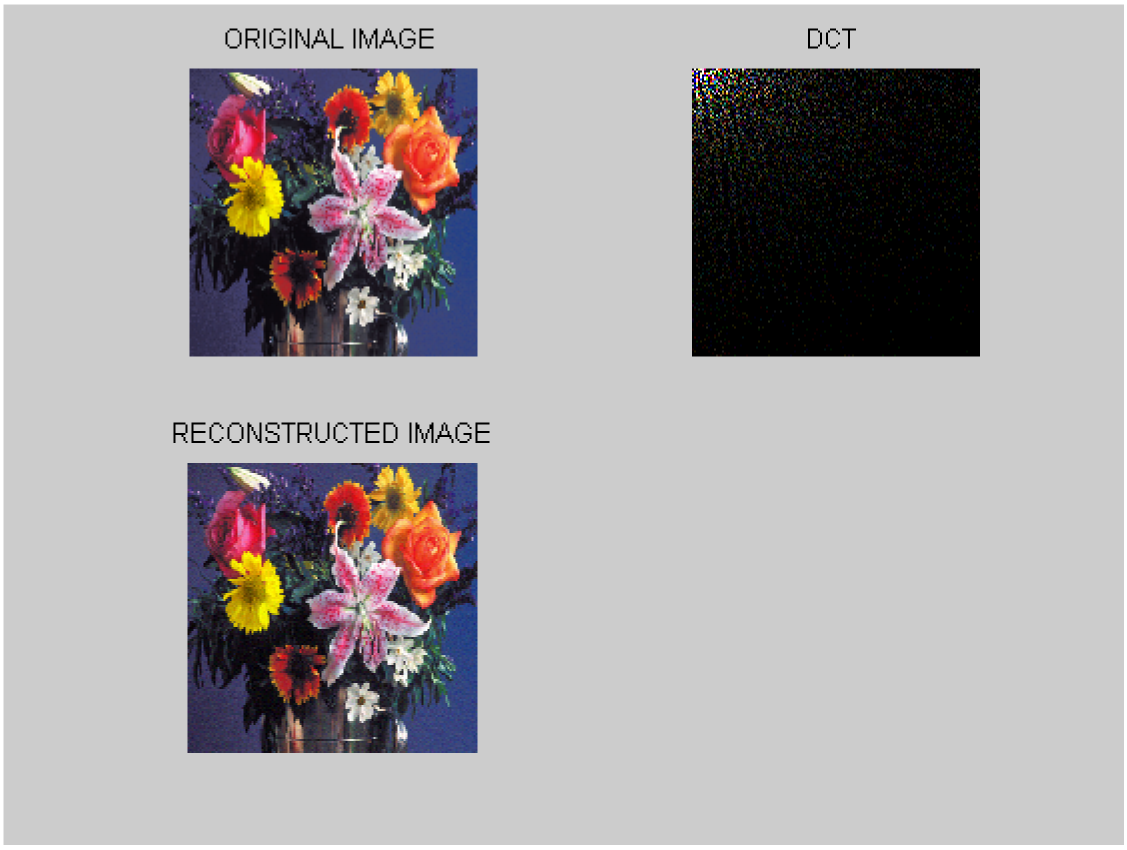
Signal to Noise Ratio (db) =27.98

Simulation Time (Seconds) = 10.50



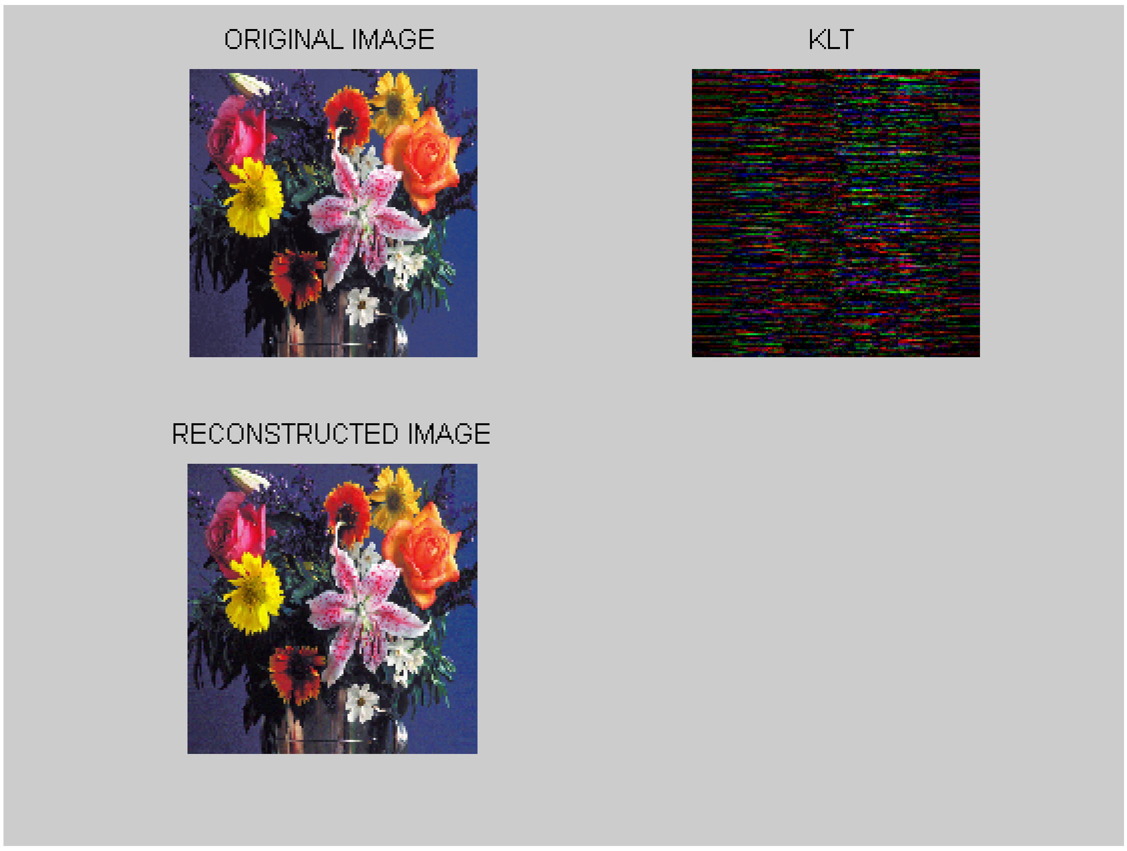
OUTPUT FILE OF COLOR IMAGE USING DCT

```
Original image size (bytes) = 196608
Compressed image size (bytes) =  58896
Decompressed image size (bytes) =  196608
Compression Ratio = 70%
Signal to Noise Ratio (db)  =23.05
Simulation Time (Seconds) = 3.57
```



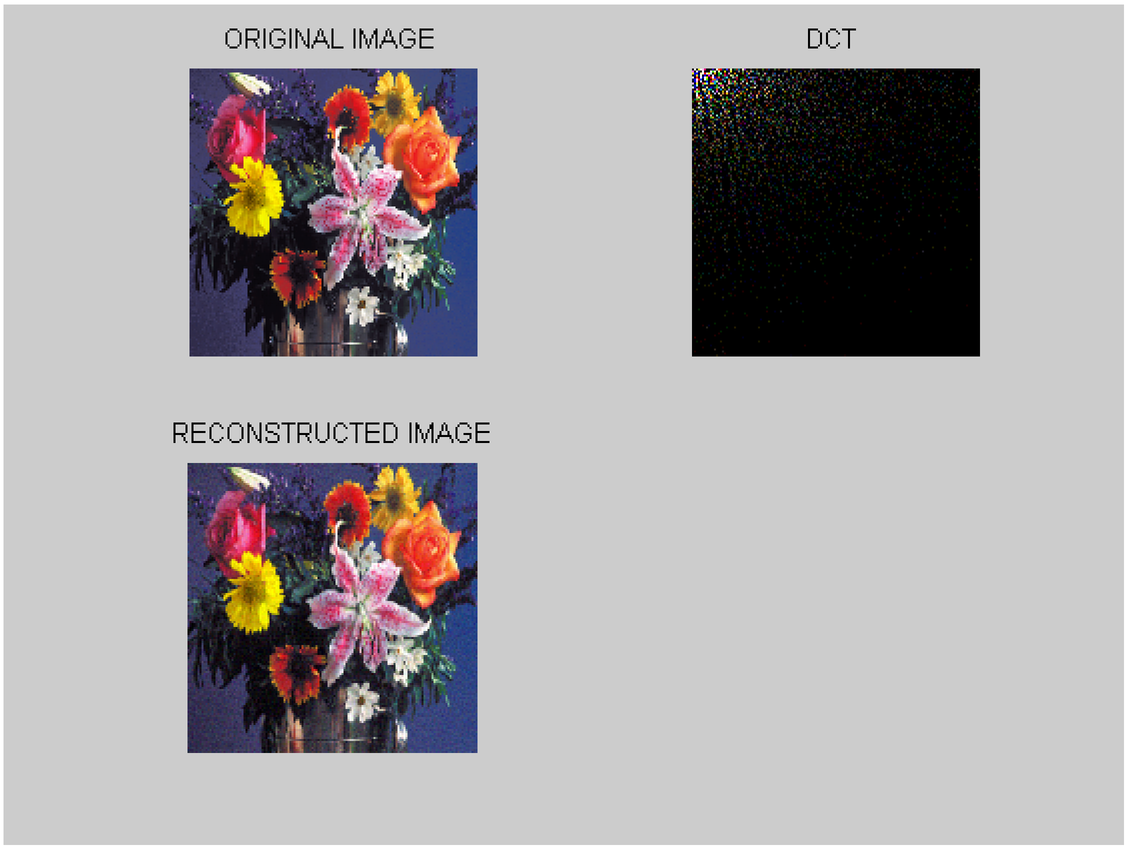
OUTPUT FILE OF COLOR IMAGE USING KLT

Original image size (bytes) = 196608
Compressed image size (bytes) = 38838
Decompressed image size (bytes) = 196608
Compression Ratio = 80%
Signal to Noise Ratio (db) = 24.29
Simulation Time (Seconds) = 10.68



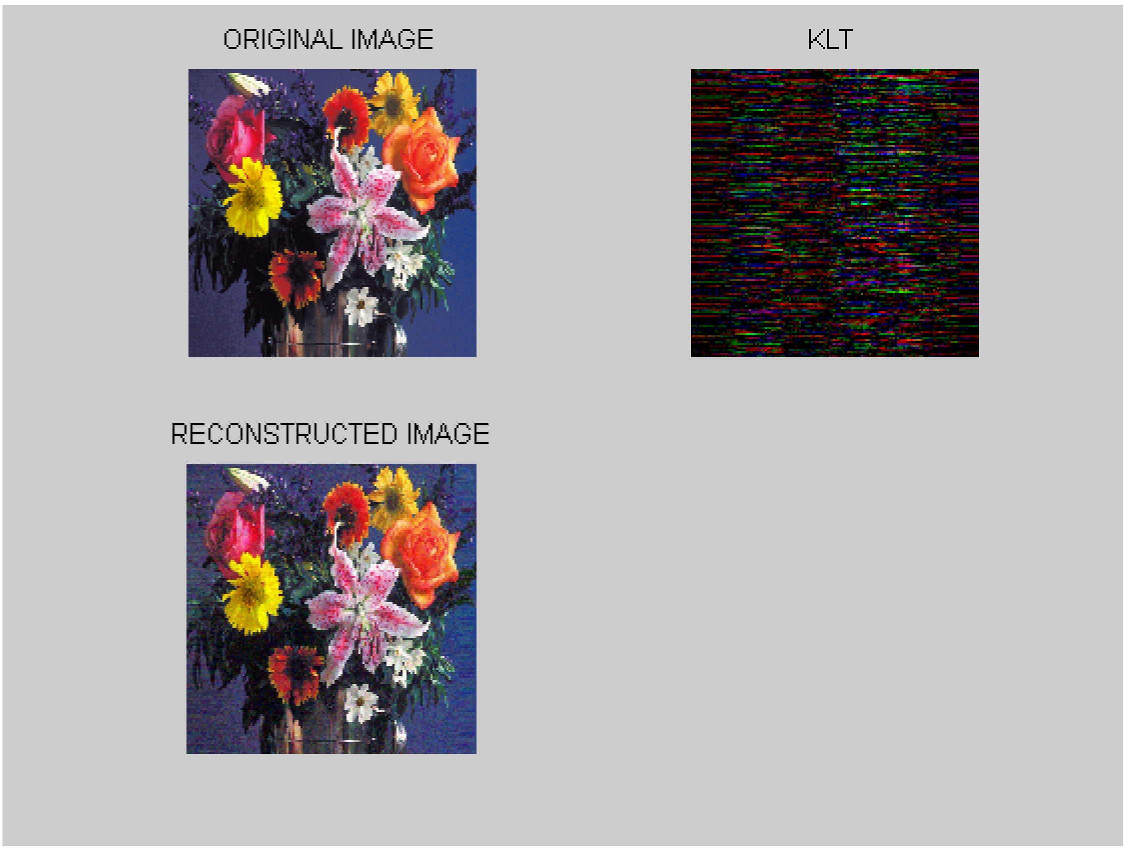
OUTPUT FILE OF COLOR IMAGE USING DCT

```
Original image size (bytes) = 196608
Compressed image size (bytes) =  38864
Decompressed image size (bytes) =  196608
Compression Ratio = 80%
Signal to Noise Ratio (db)  =20.47
Simulation Time (Seconds) = 3.60
```



OUTPUT FILE OF COLOR IMAGE USING KLT

```
Original image size (bytes) = 196608
Compressed image size (bytes) =  18986
Decompressed image size (bytes) =  196608
Compression Ratio = 90%
Signal to Noise Ratio (db) =19.59
Simulation Time (Seconds) = 10.80
```



OUTPUT FILE OF COLOR IMAGE USING DCT

Original image size (bytes) = 196608

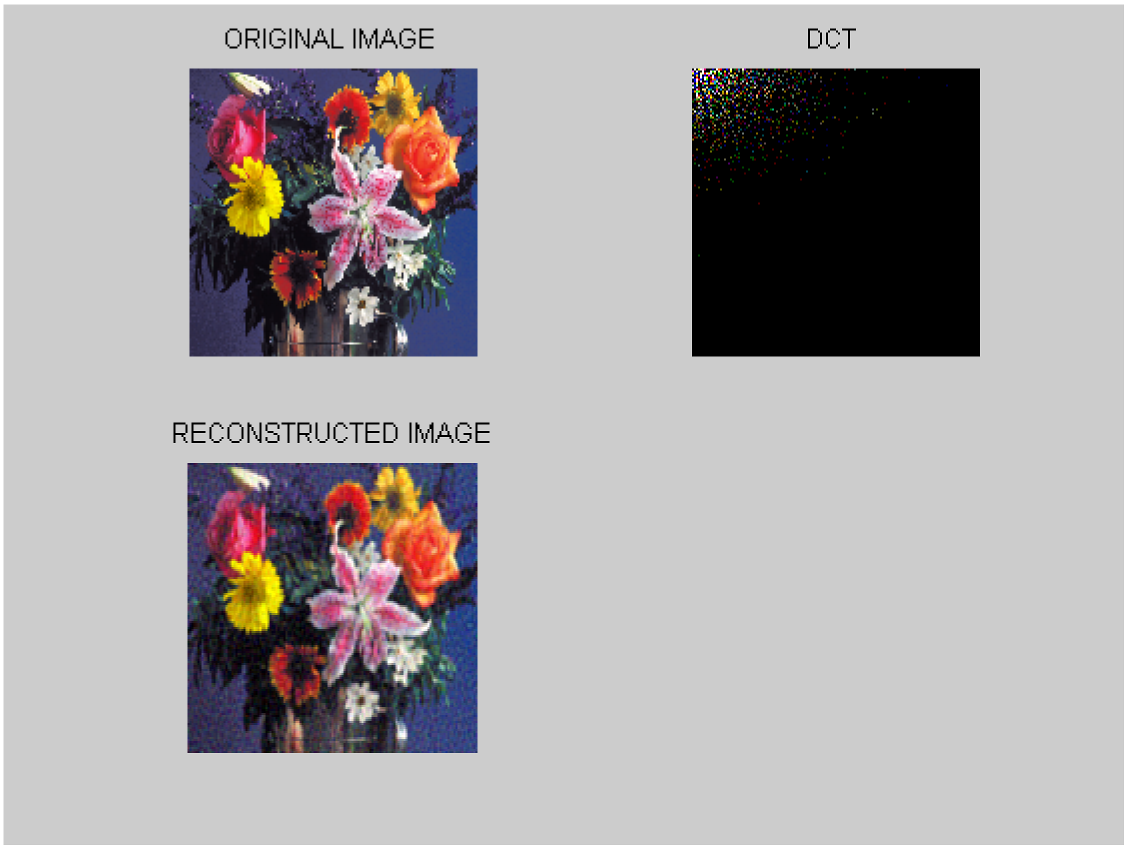
Compressed image size (bytes) = 18978

Decompressed image size (bytes) = 196608

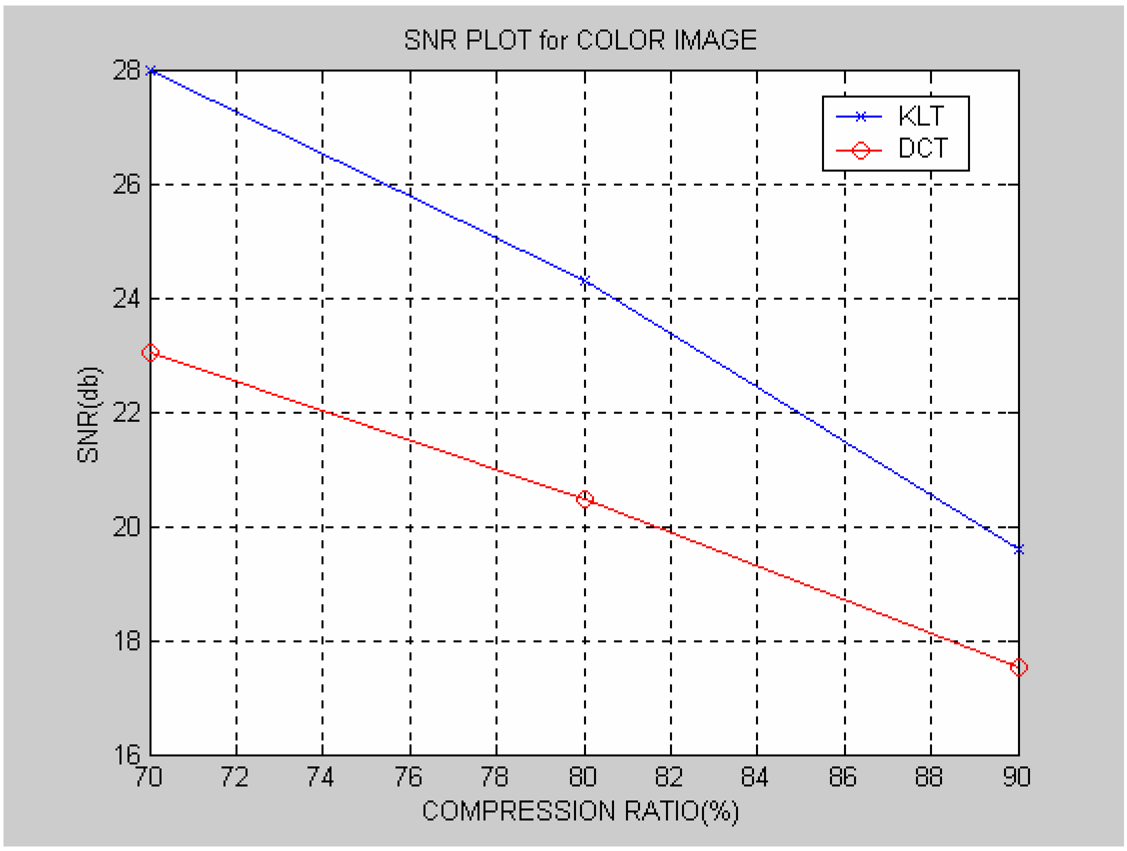
Compression Ratio = 90%

Signal to Noise Ratio (db) = 17.51

Simulation Time (Seconds) = 3.68



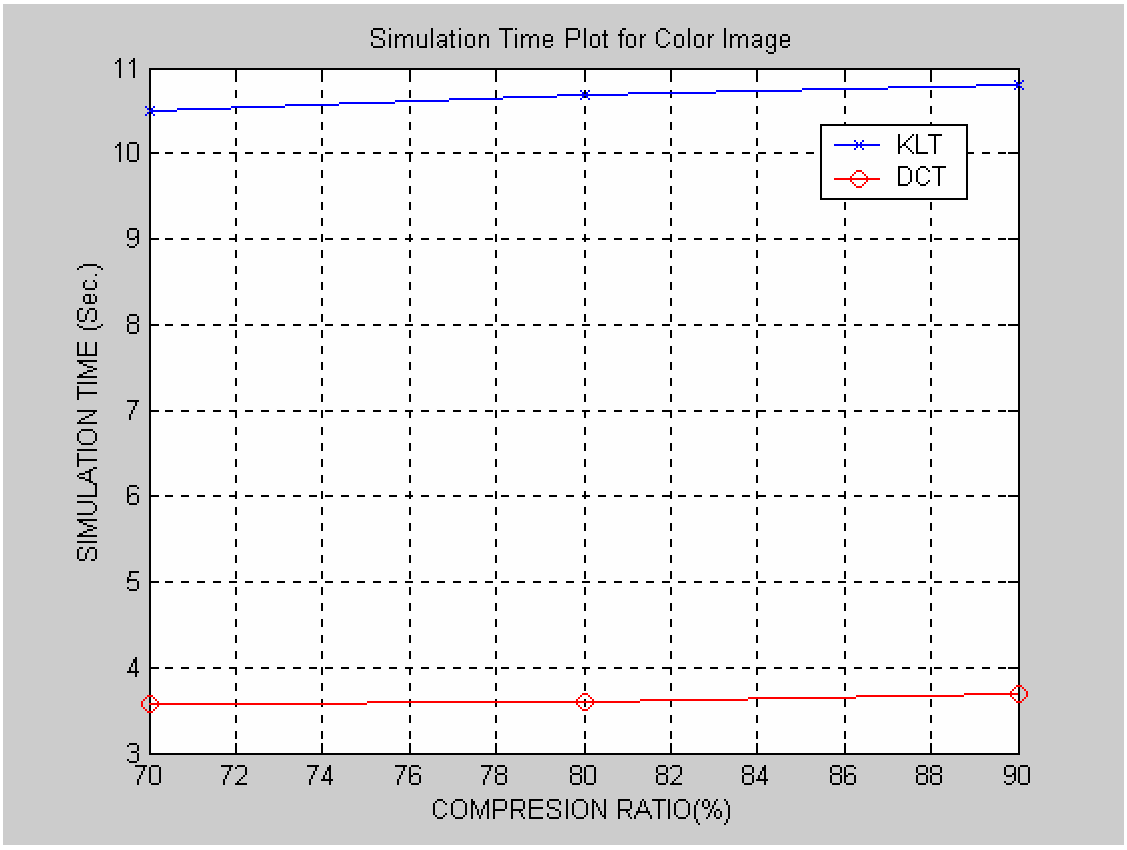
6.2.1 SNR PERFORMANCE FOR COLOR IMAGE



	CR = 70%	CR = 80%	CR = 90%
KLT	27.98	24.29	19.59
DCT	23.05	20.47	17.51

Table 6.3 : SNR Values For Color Image (All values in db)

6.2.2 SIMULATION TIME PERFORMANCE FOR COLOR IMAGE



	CR = 70%	CR = 80%	CR = 90%
KLT	10.50	10.68	10.80
DCT	3.57	3.60	3.68

Table 6.4 : Simulation Time For Color Image (All values in seconds)

CHAPTER 7

CONCLUSIONS AND SCOPE FOR FUTURE WORK

The performance of KLT, DCT and DWT was compared for images in response to human visual system (HVS), PSNR and the Complexities involved in implementation. In terms of complexities involved, KLT is practically very difficult to implement as its basis functions are data dependent while in terms of image quality and PSNR, it has excellent results.

By observing the simulations it can be seen that the compression of an image by DCT is inferior to compression by using wavelet transform. For the same amount of compression, wavelet has produced a better result than DCT in terms of image quality & PSNR. Wavelet transformation is powerful because of its multi-resolution decomposition technique. This technique allows wavelets to de-correlate an image and concentrate the energy in a few coefficients.

DCT achieves the fastest computational performance due to fast algorithms.

DCT achieves the fastest computational performance due to fast algorithms & DWT has out performed in terms of image quality & higher PSNR so there is further scope for evaluation of code which uses combination of DCT & DWT in order to achieve the fastest computational performance, image quality & higher PSNR and these techniques can be implemented on DSP processors for real time applications.

REFERENCES

1. Rafael C.Gonzalez and Richard E. Woods, Digital Image Processing, Addison Wesley publishing company, 1993, ISBN 0-201-50803-B
2. Rafael C.Gonzalez and Richard E. Woods, Digital Image Processing, using MATLAB Addison Wesley publishing company,1997, ISBN0-201-50803-B
3. Anil K. Jain, Fundamentals of Digital Image Processing, Pearson Education, 1989, ISBN 81-297-0083-2
4. Azriel Rosenfeld and Avinash C.Kak, Digital Picture Processing, Academic Press, INC, 1982, San Diego, California 92101
5. N. Ahmed, T. Natararajan and K.R. Rao, "Discrete Cosine Transform", IEEE Trans. Computers, Vol. C-23, pp.90-94, 1974
6. Rao K.R. and Yip, P. Discrete Cosine Transform-Algorithms, Advantages, Applications. Academic Press, Inc. London, 1990
7. W H Chen, and S C.Fralick "A Fast Computational Algorithm for the Discrete Cosine Transform" IEEE Trans Commun COM-25 (September 1977) 1004-1009
8. M J Narasimha and A. M Peterson "On the computation of the Discrete Cosine Transform." IEEE Trans. Commun.COM-26, (June 1978): 934-936
9. W.D. Ray and R M Driver " Further Decomposition of the Karhunen-Loeve Series Representation of a Stationary Random Process' IEEE Trans Info Theory IT-11 (November 1970); 663-668
10. A. Graps, "An Introduction to Wavelets", IEEE Computational Sciences and Engineering, Vol. 2, No. 2, Summer 1995, pp 50-61
11. M.L. Hilton, B.D. Jawerth, A. Sengupta, " Compressing Still and Moving Images with Wavelets, Multimedia Systems, Vol. 2 and No. 3, Apr. 18, 1994
12. C. Mulcahy, 'Image Compression Using The Haar Wavelet Transform' Spelman College Science & Mathematics Journal, Vol. 1, No 1, Apr. 1997, pp. 22-31
13. Amir Averbuch, Danny Lazar, Moshe Israeli, "Image Compression Using Wavelet Transform and Multiresolution Decomposition", IEEE transactions on image processing, vol. 5, No. 1, January 1996.
14. S. Saha, " Image Compression ' from DCT to Wavelets: A Review,,, http://www.acm.org/crossroads/xrds6-3/sahaimgcoding.html (current May 1, 2001)

15. R.Polikar, 'The WaveletTutorial' - http://engineering.rowan.edu/~polikar/WAVELETS/WTtutorial.html, (current March 20, 2001)

16. C. Valens, " A Really Friendly guide to Wavelets„ http://perso.wanadoo.fr/polyvalens/clemens/wavelets/wavelets.html, (current March 20, 2001)

17 JPEG vs. JPEG 2000- Comparison, Available: http://www.dspworx.com/primer_jpeg_vs__jpeg2000.htm

18 Wavelet Image Compression Construction Kit http://www.cs.dartmouth.edu/~gdavis/wavelet/wavelet.html

19 Comparative study of DCT- and wavelet-based image coding http://www.debut.cis.nctu.edu.tw/~sasami/ work/presentations/19991126.pdf

20 Discrete Cosine Transform. http://www.dynamo.ecn.purdue.edu/~ace/jpeg-tut/jpgdct1.html

21 The Wavelet Transform http://www.eso.org/projects/esomidas/ doc/user/98NOV/volb/node308.html

22 Mallat, S. G. A Theory for Multiresolution Signal Decomposition: The Wavelet Representation, IEEE Trans. PAMI, vol. 11, no. 7, July 1989, pp. 674

23 Froment, J. and Mallat, S. Second Generation Compact Image Coding with Wavelets, in C.K. Chui, editor, Wavelets: A Tutorial in Theory and Applications, vol. 2, Academic Press, NY, 1992

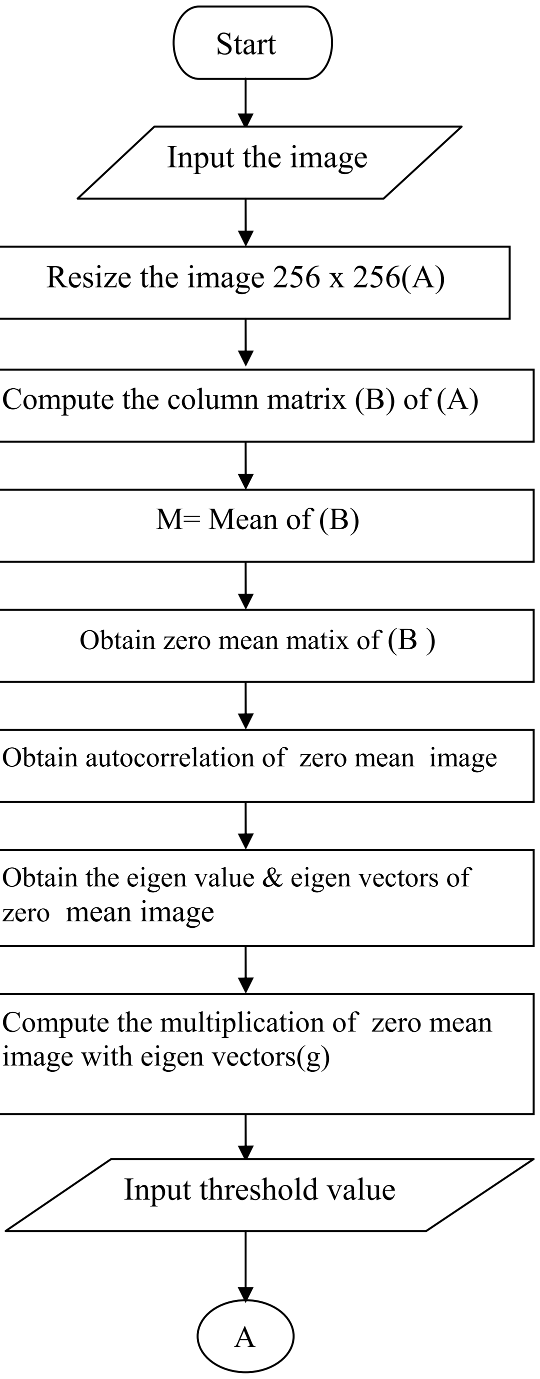
24 Lewis, A. S. and Knowles, G. Image Compression Using the 2-D Wavelet Transform, *IEEE Trans. IP,* vol. 1, no. 2, April 1992, pp. 244-250.

25 Strang, G. and Nguyen, T. Wavelets and Filter Banks, Wellesley-Cambridge Press, Wellesley, MA, 1996, http://www-math.mit.edu/~gs/books/wfb.html

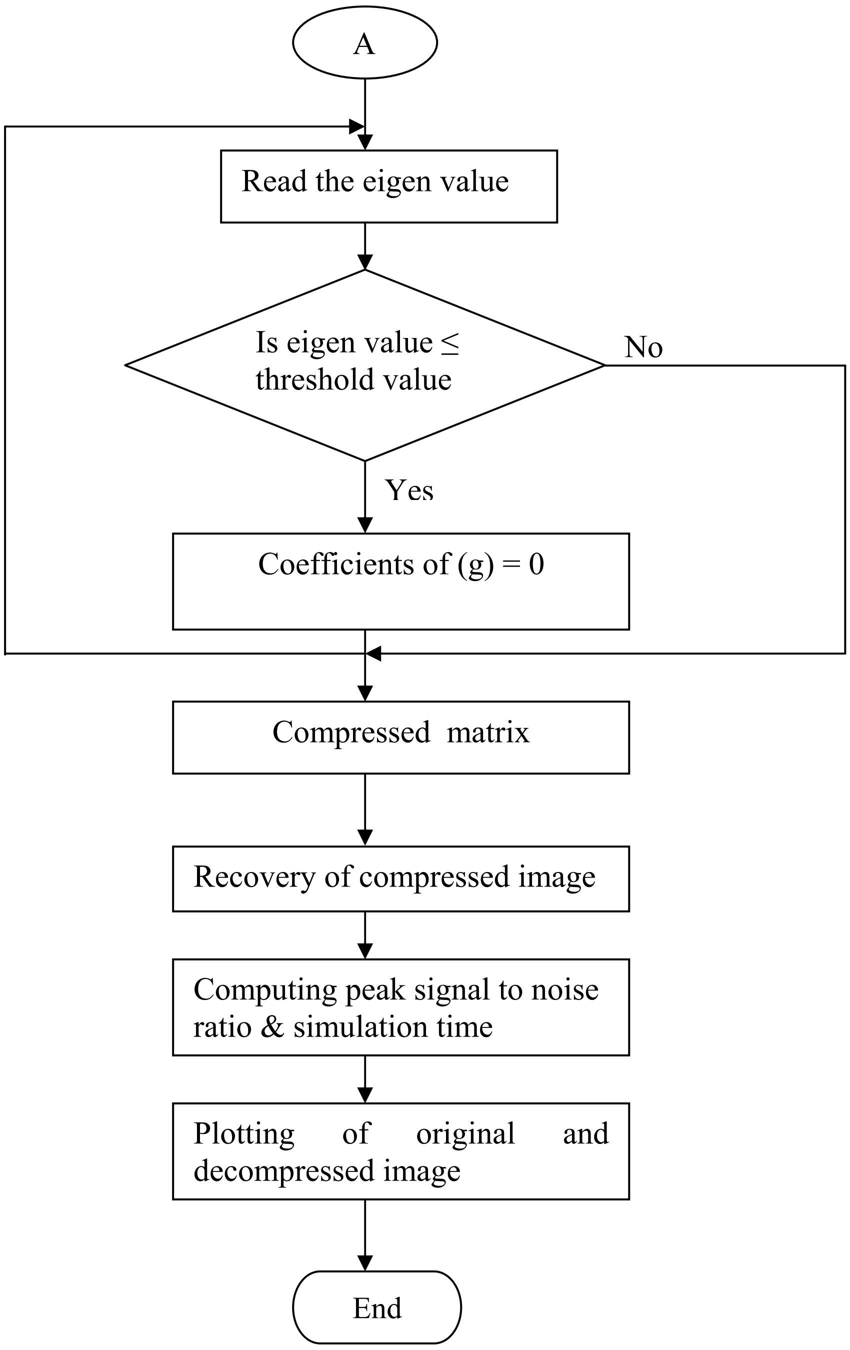
26 Taubman, D. High Performance Scalable Image Compression with EBCOT, submitted to *IEEE Tran. IP,* Mar. 1999, http://maestro.ee.unsw.edu.au/~taubman/activities/preprints/ebcot.zip

APPENDIX

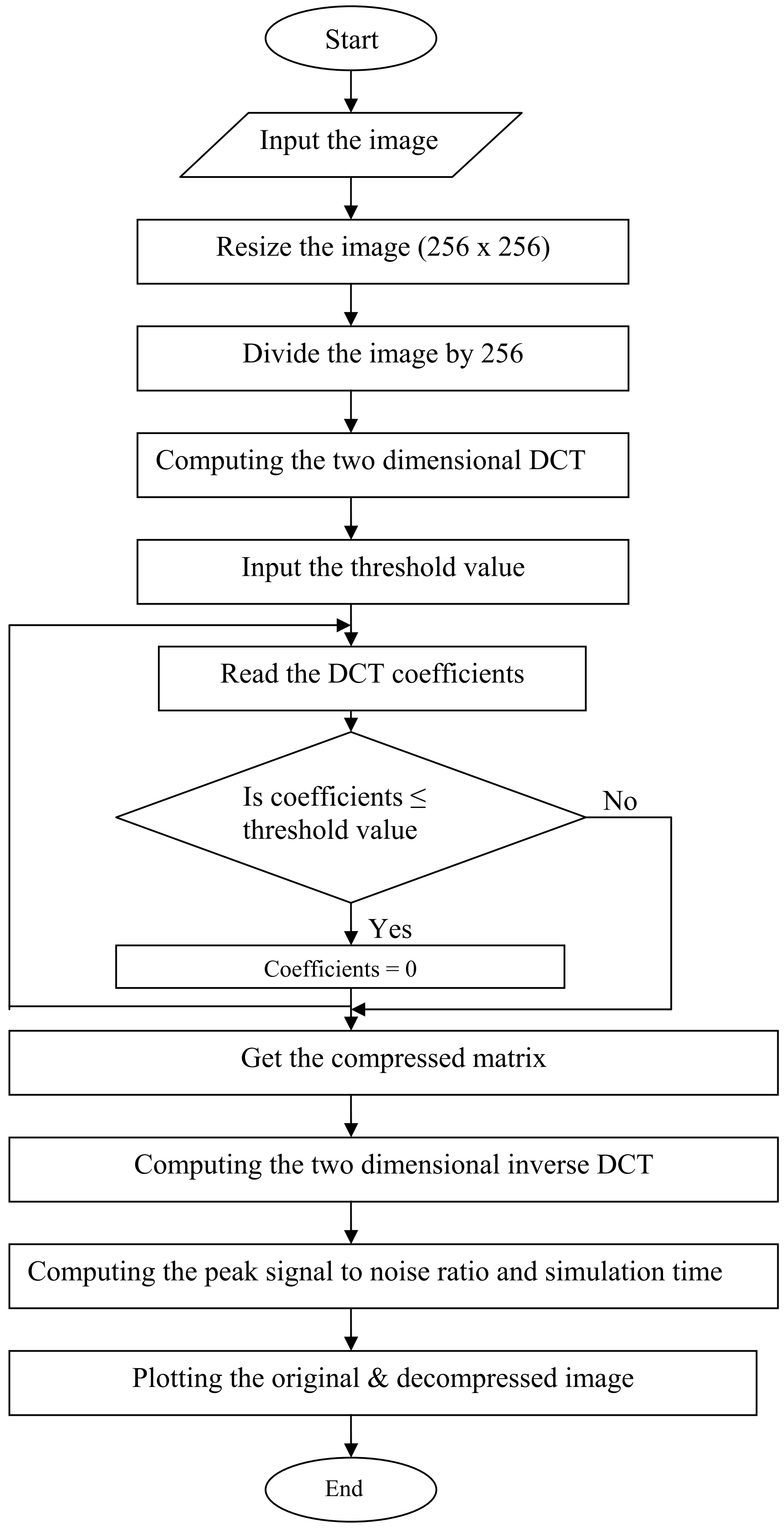
FLOW CHART FOR KARHUNEN LOEVE TRANSFORM



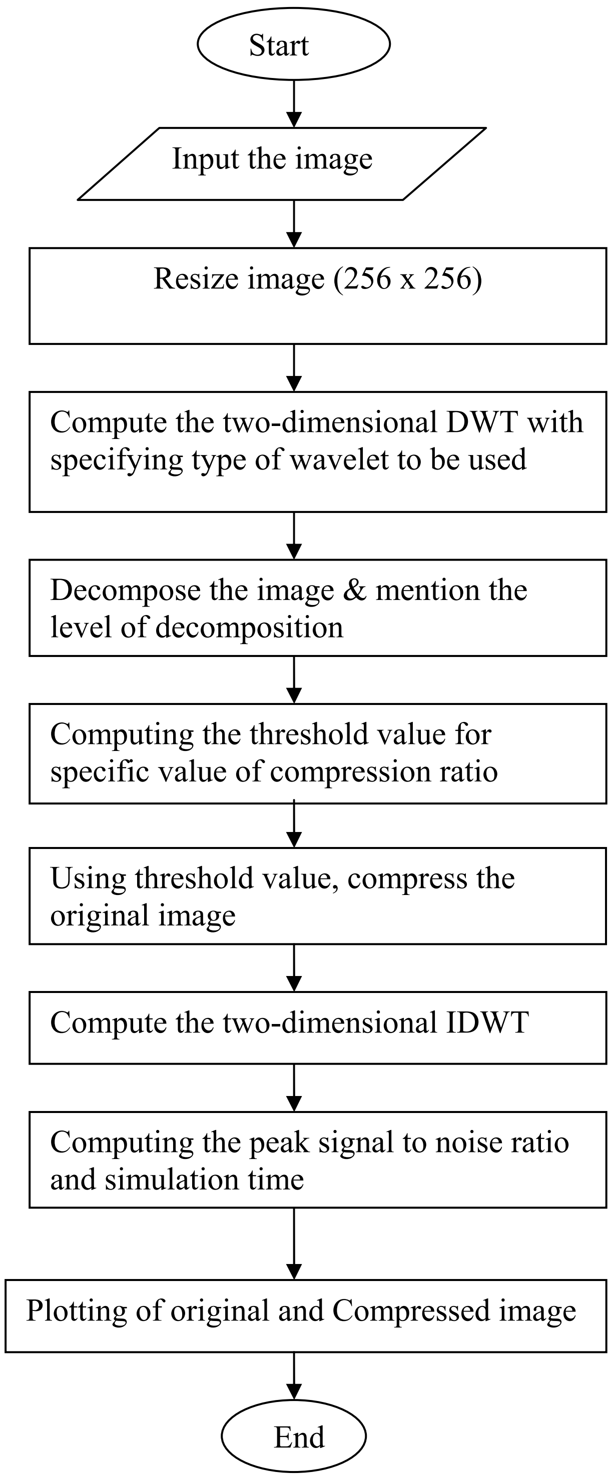
Start
Input the image
Resize the image 256 x 256(A)
Compute the column matrix (B) of (A)
M= Mean of (B)
Obtain zero mean matix of (B)
Obtain autocorrelation of zero mean image
Obtain the eigen value & eigen vectors of zero mean image
Compute the multiplication of zero mean image with eigen vectors(g)
Input threshold value
A


A
Read the eigen value
Is eigen value ≤ threshold value
No
Yes
Coefficients of (g) = 0
Compressed matrix
Recovery of compressed image
Computing peak signal to noise ratio & simulation time
Plotting of original and decompressed image
End

FLOW CHART FOR IMAGE COMPRESSION USING DCT

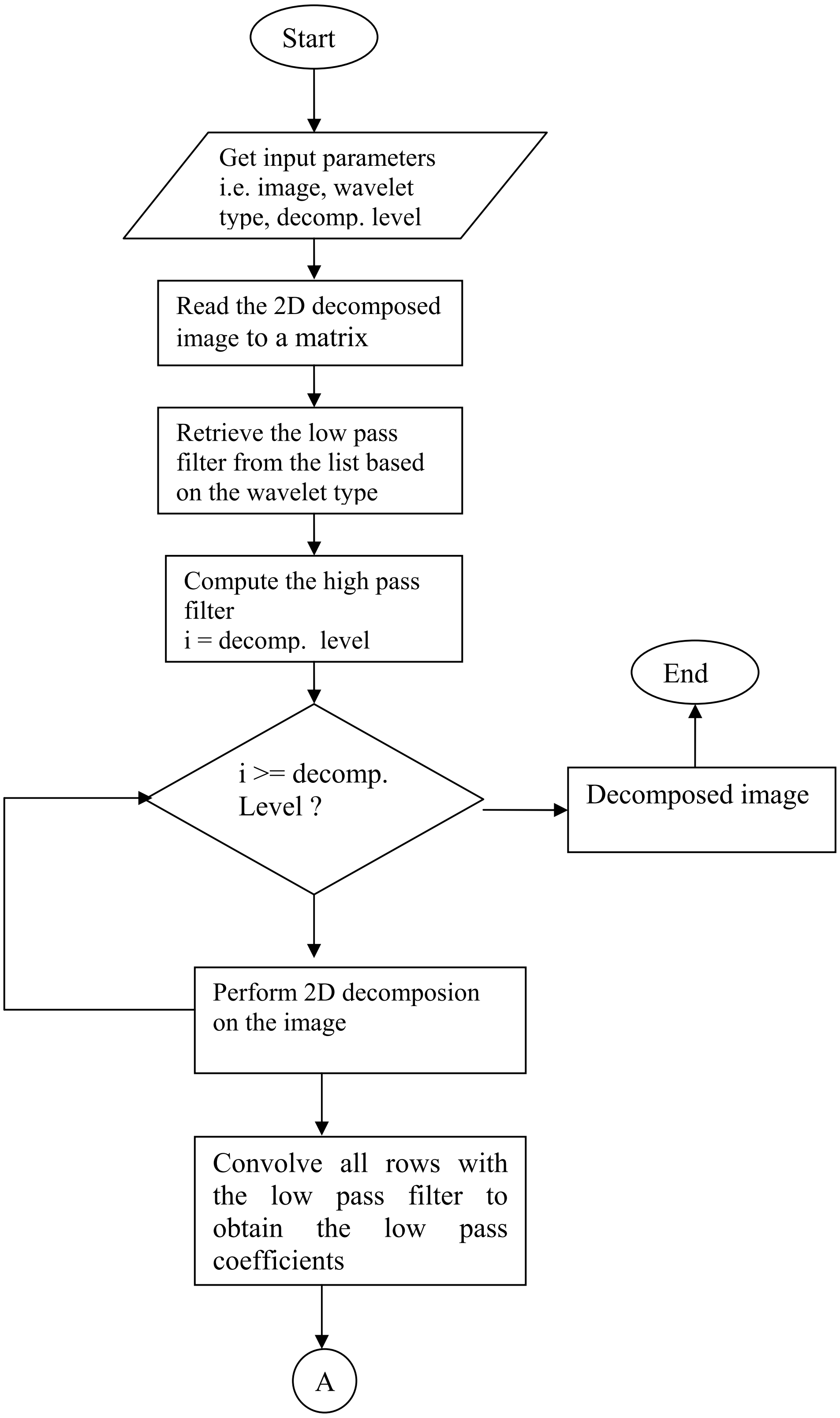

Start
Input the image
Resize the image (256 x 256)
Divide the image by 256
Computing the two dimensional DCT
Input the threshold value
Read the DCT coefficients
Is coefficients ≤ threshold value
No
Yes
Coefficients = 0
Get the compressed matrix
Computing the two dimensional inverse DCT
Computing the peak signal to noise ratio and simulation time
Plotting the original & decompressed image
End

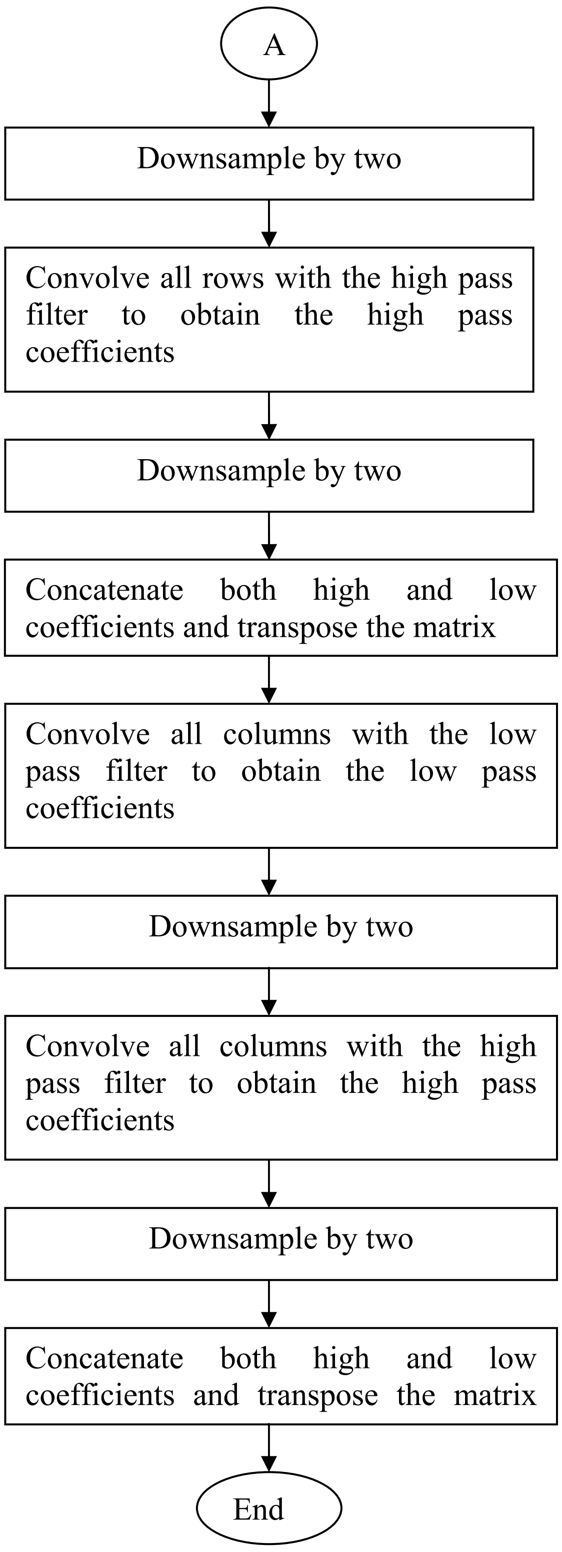
FLOW CHART FOR IMAGE COMPRESSION USING DWT



Start
Input the image
Resize image (256 x 256)
Compute the two-dimensional DWT with specifying type of wavelet to be used
Decompose the image & mention the level of decomposition
Computing the threshold value for specific value of compression ratio
Using threshold value, compress the original image
Compute the two-dimensional IDWT
Computing the peak signal to noise ratio and simulation time
Plotting of original and Compressed image
End

FLOW CHART FOR TWO DIMENSIONAL DWT


Start
Get input parameters i.e. image, wavelet type, decomp. level
Read the 2D decomposed image to a matrix
Retrieve the low pass filter from the list based on the wavelet type
Compute the high pass filter
i = decomp. level
i >= decomp. Level ?
Decomposed image
End
Perform 2D decomposion on the image
Convolve all rows with the low pass filter to obtain the low pass coefficients
A


A
Downsample by two
Convolve all rows with the high pass filter to obtain the high pass coefficients
Downsample by two
Concatenate both high and low coefficients and transpose the matrix
Convolve all columns with the low pass filter to obtain the low pass coefficients
Downsample by two
Convolve all columns with the high pass filter to obtain the high pass coefficients
Downsample by two
Concatenate both high and low coefficients and transpose the matrix
End

PROGRAM CODE

```
%GDCT75
%To find the Two Dimensional Lossy Cosine Transform of the GRAY  Images
function [] = LYdct2d
 clc;
clear all;
close all;
ImageName = input('Enter the filename(tif) ','s');
ImageExt = 'tif';
Input = input('Enter the name of the resized file ','s');
Transformed = input('Enter the name of the transformed image ','s');
Reconstructed = input('Enter the name of the reconstructed image ','s');

t=cputime;
I=imread(ImageName,ImageExt);
img=imresize(I,[256 256]);
img=double(img)/255;
imwrite(img,Input,ImageExt);

Transform=dct2(img);
count=0;
for i=1:255

   for j=1:255

       if abs(Transform(i,j))<=0.054

           Transform(i,j)=0;
           count=count+1;

       end
         end
   end
count;
imwrite(Transform,Transformed,ImageExt);

R=idct2(Transform);

imwrite(R,Reconstructed,ImageExt);

CR = (count/65536)*100
size(img)
size(R)
figure (1);
subplot(221);imshow(img);title('original image');
subplot(222);imshow(Transform);title('dct');
subplot(223);imshow(R);title('compressed image')
```

```
% SNR(img,R) returns the signal to noise ratio between img and R (dB).
% img is the original image and R is reconstructed image.
% The SNR value is useful to calculate the distortions on an image.

[m n] = size(img);
total_img = 0;
total_Dif = 0;

for u = 1:m
    for v = 1:n
        total_img = total_img + img(u, v)^2;
        total_Dif = total_Dif + (img(u, v) - R(u, v))^2;
    end
end

SNR = (total_img) / (total_Dif);
SNR = (10 * log10(SNR))
simulation_time=cputime-t

%GKLT75
%To find the Lossless Karhunen- Loeve Transform of the GRAY Images

function [] = LYklt
ImageName = input('Enter the filename(tif) ','s');
ImageExt = 'tif';
Input = input('Enter the name of the resized file ','s');
Transformed = input('Enter the name of the transformed image ','s');
Reconstructed = input('Enter the name of the reconstructed image ','s');
t=cputime;
I=imread(ImageName,ImageExt);
img=imresize(I,[256 256]);
img=double(img)/255;
imwrite(img,Input,ImageExt);

[a,b]= size(img);

m = sum(img');

Mean = (1/b)*(m');

newmean=(Mean)*(Mean');

X=0;

for i=1:b

   X = X + (img(:,i)*img(:,i)');
```

```
end

newx = (1/b)* X;

covmatrix = newx - newmean;

[V,D] = eig(covmatrix);

for i=1:b

   output(:,i) = V*(img(:,i) - Mean);

end

count1=0;
for i=1:255

  for j=1:255
        if abs(output(i,j))<=.054

        output(i,j)=0;
        count1=count1+1;
     end
        end
   end

count1;
imwrite(output,Transformed,ImageExt);

for i=1:b

   R(:,i) = V'*output(:,i) + Mean;

end
imwrite(R,Reconstructed,ImageExt);

size(img)
size(R)
figure ;
subplot(221);imshow(img);title('original image');
subplot(222);imshow(output);title('klt');
subplot(223);imshow(R);title('compressed image')
[m n] = size(img);
total_img = 0;
total_Dif = 0;
for u = 1:m
    for v = 1:n
```

```
total_img = total_img + img(u, v)^2;
        total_Dif = total_Dif + (img(u, v) - R(u, v))^2;
    end
end

SNR = (total_img) / (total_Dif);
SNR = (10 * log10(SNR))
simulation_time=cputime-t

%GDWT
%To find the discrete wavelet transform of GRAY Image

function [] = LYdwt
ImageName = input('Enter the filename(tif) ','s');
ImageExt = 'tif';
Input = input('Enter the name of the resized file ','s');

Transformed = input('Enter the name of the transformed image ','s');
Reconstructed = input('Enter the name of the reconstructed image ','s');
t=cputime;

I=imread(ImageName,ImageExt);

im_input = imresize(I,[256 256]);

input=double(im_input);

[cA,cH,cV,cD] = dwt2(im_input,'bior');
[C,S] = wavedec2(im_input,2,'bior3.7');

[m,n] = size(cA);
temp_cA = reshape(cA,1,m*n);
temp_cA = sort(temp_cA);
temp_cH = reshape(cH,1,m*n);
temp_cH = sort(temp_cH);
temp_cV = reshape(cV,1,m*n);
temp_cV = sort(temp_cV);
temp_cD = reshape(cD,1,m*n);
temp_cD = sort(temp_cD);

index = round(0.112*(length(temp_cA)/4));
cA25_cutoff = temp_cA(index);
cA25 = cA.*(cA(:,:)>= cA25_cutoff);
cH25_cutoff = temp_cH(index);
cH25 = cH.*(cH(:,:)>= cH25_cutoff);
cV25_cutoff = temp_cV(index);
cV25 = cV.*(cV(:,:)>= cV25_cutoff);
cD25_cutoff = temp_cD(index);
```

```
cD25 = cD.*(cD(:,:)>= cD25_cutoff);
% compression
im_rec = idwt2(cA25,cH25,cV25,cD25,'bior3.7');
imwrite(im_rec,'CompressedImage.tif');

% display results
figure; subplot(1,2,1); imshow(im_input); title('Original Image');
subplot(1,2,2); imshow(uint8(im_rec75)); title('Compressed Image ');

%SNR
[m n] = size(im_input);
total_img = 0;
total_Dif = 0;

for u = 1:m
    for v = 1:n
        total_img = total_img + input(u, v)^2;
        total_Dif = total_Dif + (input(u, v) - im_rec75(u, v))^2;
    end
end

SNR = (total_img) / (total_Dif);
SNR_75 = (10 * log10(SNR))
simulation_time=cputime-t

%CDCT90
%To find the Lossy  Direct Cosine Transform of the color  Images

function [] = LYcolordct
ImageName = input('Enter the filename(tif) of color image ','s');
ImageExt = 'tif';
Input = input('Enter the name of the resized file ','s');
Transformed = input('Enter the name of the transformed image ','s');
Reconstructed = input('Enter the name of the reconstructed image ','s');
time1=cputime;
[X,map1]=imread(ImageName,ImageExt);
img=imresize([X,map1],[256 256]);
img=double(img)/256;
size(img)
R=img(:,:,1);

G=img(:,:,2);

B=img(:,:,3);

n1=dct2(R);

n2=dct2(G);
```

```
n3=dct2(B);
count1=0;
for i=1:256

   for j=1:256

      if abs(n1(i,j))<=0.158

         n1(i,j)=0;
         count1=count1+1;

      end
        end
   end
count1;
count2=0;
for i=1:256

   for j=1:256

      if abs(n2(i,j))<=0.158

         n2(i,j)=0;
         count2=count2+1;

      end

   end

end
count2;
count3=0;
for i=1:256

   for j=1:256

      if abs(n3(i,j))<=0.158

         n3(i,j)=0;
         count3=count3+1;
      end

   end

end
count3;
count=count1+count2+count3;
```

```
n = n1;
n(:,:,2) = n2;
n(:,:,3) = n3;
imwrite(n,Transformed,ImageExt);
transform = imread(Transformed,ImageExt);

size(transform)
%compression ratio=(size of original image-size of compressed image)/(size of
original image)

CR=(count/196608)*100
m1=idct2(n1);

m2=idct2(n2);

m3=idct2(n3);

m= m1;

m(:,:,2) = m2;

m(:,:,3) = m3;

imwrite(m,Reconstructed,ImageExt);

[Y,map2]= imread(Reconstructed,ImageExt);

R=imresize([Y,map2],[256 256]);

R=double(R)/256;

figure (1);
subplot(221);imshow(img);title('original image');
subplot(222);imshow(transform);title('dct');
subplot(223);imshow(R);title('compressed image');


% SNR(img,R) returns the signal to noise ratio between I and R (dB).
% I is the original image and R is a modified version of I.
% The SNR value is useful to calculate the distortions on an image.

[m n] = size(img);
total_img = 0;
total_Dif = 0;

for u = 1:m
    for v = 1:n
```

```
     total_img = total_img + img(u, v)^2;
       total_Dif = total_Dif + (img(u, v) - R(u, v))^2;
    end
end
% if (total_Dif == 0)
%     total_Dif = 1;
% end
S = (total_img) / (total_Dif);
S = (10 * log10(S))

time=cputime-time1

%CKLT90
%To find the Lossless Karhunen- Loeve Transform of the color Images

function [] = LYklt
ImageName = input('Enter the filename(tif) of color image ','s');
ImageExt = 'tif';
Input = input('Enter the name of the resized file ','s');
Transformed = input('Enter the name of the transformed image ','s');
Reconstructed = input('Enter the name of the reconstructed image ','s');
time1=cputime;
[X,map1]=imread(ImageName,ImageExt);
img=imresize([X,map1],[256 256]);

img=double(img)/256;

R=img(:,:,1);
G=img(:,:,2);
B=img(:,:,3);
[a,b]= size(R);

t = sum(R');

Mean1= (1/b)*(t');

newmean1=(Mean1)*(Mean1');

X=0;

for i=1:b

   X = X + (R(:,i)*R(:,i)');

end

newx = (1/b)* X;
```

```
covmatrix1 = newx - newmean1;

[V1,D1] = eig(covmatrix1);

for i=1:b

  n1(:,i) = V1*(R(:,i) - Mean1);

end
q = sum(G');

Mean2 = (1/b)*(q');

newmean2=(Mean2)*(Mean2');

Y=0;

for i=1:b

   Y = Y + (G(:,i)*G(:,i)');

end

newy = (1/b)* Y;

covmatrix2 = newy - newmean2;

[V2,D2] = eig(covmatrix2);

for i=1:b

  n2(:,i) = V2*(G(:,i) - Mean2);

end

r = sum(B');

Mean3 = (1/b)*(r');

newmean3=(Mean3)*(Mean3');

Z=0;

for i=1:b

   Z= Z + (B(:,i)*B(:,i)');

end
```

```
newz = (1/b)* Z;

covmatrix3 = newz - newmean3;

[V3,D3] = eig(covmatrix3);

for i=1:b

  n3(:,i) = V3*(B(:,i) - Mean3);

end
count1=0;

for i=1:255

  for j=1:255

      if abs(n1(i,j))<=.12

     n1(i,j)=0;
      count1=count1+1;

      end

   end

end
count1;
count2=0;
for i=1:255

  for j=1:255

      if abs(n2(i,j))<=.12

        n2(i,j)=0;
      count2=count2+1;
      end

   end

end
count2;
count3=0;

for i=1:255
```

```
 for j=1:255

       if abs(n3(i,j))<=.12

           n3(i,j)=0;
          count3=count3+1;
       end

   end

end

count3;
 count4=count1+count2+count3;
n=n1;
n(:,:,2) = n2;

n(:,:,3) = n3;

imwrite(n,Transformed,ImageExt);

D = imread(Transformed,ImageExt);


for i=1:b

   m1(:,i) = V1'*n1(:,i) + Mean1;

end
for i=1:b

   m2(:,i) = V2'*n2(:,i) + Mean2;

end
for i=1:b

   m3(:,i) = V3'*n3(:,i) + Mean3;

end
m=m1;
m(:,:,2)=m2;
m(:,:,3)=m3;

imwrite(m,Reconstructed,ImageExt);
[Y,map2]= imread(Reconstructed,ImageExt);

R=imresize([Y,map2],[256 256]);
```

```
R=double(R)/256;
%compression ratio=(size of original image-size of compressed image)/(size of
original image)

size(img)
size(R)

figure (1);
subplot(221);imshow(img);title('original image');
subplot(222);imshow(D);title('klt');
subplot(223);imshow(R);title('compressed image');

% SNR(img,R) returns the signal to noise ratio between I and R (dB).
% I is the original image and R is a modified version of I.
% The SNR value is useful to calculate the distortions on an image.

[m n] = size(img);
total_img = 0;
total_Dif = 0;

for u = 1:m
    for v = 1:n
        total_img = total_img + img(u, v)^2;
        total_Dif = total_Dif + (img(u, v) - R(u, v))^2;
    end
end
% if (total_Dif == 0)
%     total_Dif = 1;
% end
S = (total_img) / (total_Dif);
S = (10 * log10(S))

time=cputime-time1
```

DWT

```
t1=cputime;
load Image
image(X);
X1=imresize(X,[256 256]);
size(X1);
colormap(map);
% colorbar;
% Perform a single-level decomposition of the image
% using the bior3.7 wavelet. Type:
[cA1,cH1,cV1,cD1] = dwt2(X1,'bior3.7');
[C,S] = wavedec2(X1,2,'bior3.7');

% To construct the level-one approximation and details
% (A1, H1, V1, and D1) from the coefficients cA1, cH1,
% cV1, and cD1, type:
A1 = upcoef2('a',cA1,'bior3.7',1);
H1 = upcoef2('h',cH1,'bior3.7',1);
V1 = upcoef2('v',cV1,'bior3.7',1);
D1 = upcoef2('d',cD1,'bior3.7',1);
sx = size(X1);
A1 = idwt2(cA1,[],[],[],'bior3.7',sx);
H1 = idwt2([],cH1,[],[],'bior3.7',sx);
V1 = idwt2([],[],cV1,[],'bior3.7',sx);
D1 = idwt2([],[],[],cD1,'bior3.7',sx);

% To display the results of the level 1 decomposition type:
figure;
colormap(map);
subplot(2,2,1); image(wcodemat(A1,192));
title('Approximation A1')
axis square
subplot(2,2,2); image(wcodemat(H1,192));
title('Horizontal Detail H1')
axis square
subplot(2,2,3); image(wcodemat(V1,192));
axis square
title('Vertical Detail V1')
subplot(2,2,4); image(wcodemat(D1,192));
title('Diagonal Detail D1')
axis square
Xsyn = idwt2(cA1,cH1,cV1,cD1,'bior3.7');
% To perform a level 2 decomposition of the image
% (again using the bior3.7 wavelet), type:
[C,S] = wavedec2(X1,2,'bior3.7');

% To extract the level 2 approximation coefficients from C, type:
cA2 = appcoef2(C,S,'bior3.7',2);
% To extract the first- and second-level detail coefficients from C, type:
cH2 = detcoef2('h',C,S,2);
cV2 = detcoef2('v',C,S,2);
cD2 = detcoef2('d',C,S,2);
cH1 = detcoef2('h',C,S,1);
cV1 = detcoef2('v',C,S,1);
cD1 = detcoef2('d',C,S,1);
% or
[cH2,cV2,cD2] = detcoef2('all',C,S,2);
[cH1,cV1,cD1] = detcoef2('all',C,S,1);

% To reconstruct the level 2 approximation from C, type:
A2 = wrcoef2('a',C,S,'bior3.7',2);
```

```
% To reconstruct the level 1 and 2 details from C, type:
H1 = wrcoef2('h',C,S,'bior3.7',1);
V1 = wrcoef2('v',C,S,'bior3.7',1);
D1 = wrcoef2('d',C,S,'bior3.7',1);
H2 = wrcoef2('h',C,S,'bior3.7',2);
V2 = wrcoef2('v',C,S,'bior3.7',2);
D2 = wrcoef2('d',C,S,'bior3.7',2);
% To display the results of the level 2 decomposition, type:
figure;
colormap(map);
subplot(2,4,1);image(wcodemat(A1,192));
title('Approximation A1')
axis square
subplot(2,4,2);image(wcodemat(H1,192));
title('Horizontal Detail H1')
axis square
subplot(2,4,3);image(wcodemat(V1,192));
title('Vertical Detail V1')
axis square
subplot(2,4,4);image(wcodemat(D1,192));
title('Diagonal Detail D1')
axis square
subplot(2,4,5);image(wcodemat(A2,192));
title('Approximation A2')
axis square
subplot(2,4,6);image(wcodemat(H2,192));
title('Horizontal Detail H2')
axis square
subplot(2,4,7);image(wcodemat(V2,192));
title('Vertical Detail V2')
axis square
subplot(2,4,8);image(wcodemat(D2,192));
title('Diagonal Detail D2')
axis square
% To reconstruct the original image from the wavelet decomposition structure,
% type:
X0 = waverec2(C,S,'bior3.7');
sorh='h';

x=abs(C);
x=sort(x);
dropindex = length(x) * (90)/100;
   dropindex = round(dropindex);
   threshold = x(dropindex);

if (threshold == 0)
end
threshold;
```

```
% Compressing an Image.
% To compress the original image X, use the ddencmp command to calculate
% the default parameters and the wdencmp command to perform the actual
% compression. Type:
[thr,sorh,keepapp] = ddencmp('cmp','wv',X);
[Xcomp,CXC,LXC,PERF0,PERFL2]=
wdencmp('gbl',C,S,'bior3.7',2,threshold,sorh,1);

% Displaying the Compressed Image.
% To view the compressed image side by side with the original, type:
figure
colormap(map);
subplot(121); image(X1); title('Original Image');
 axis square
subplot(122); image(Xcomp); title('Compressed Image');
axis square

% SNR(X,R) returns the signal to noise ratio between img and R (dB).
% img is the original image and R is a modified version of img.
% The SNR value is useful to calculate the distortions on an image.

[m n] = size(X1);
total_X = 0;
total_Dif = 0;

for u = 1:m
    for v = 1:n
       total_X = total_X + X1(u, v)^2;
       total_Dif = total_Dif + (X1(u, v) - Xcomp(u, v))^2;
    end
end

SNR = (total_X)/ (total_Dif);
SNR = (10 * log10(SNR))
simulation_time=cputime-t1

% SNR Plots for Gray Image
x=[75 85 90];
w=[31.24,30.29,29.06];
k=[30.92,29.23,28.47];
d=[29.26 26.32,24.49];
plot(x,w,'-ms',x,k,'-bx',x,d,'-ro')
h = legend('DWT','KLT','DCT',3);
Xlabel('COMPRESSION RATIO(%)')
Ylabel('SNR(db)')
Title('SNR PLOT for GRAY IMAGE');
grid on;
% SIMULATION TIME PLOT FOR GRAY IMAGE
x=[75 85 90];
```

```
k=[3.68,3.79,3.95];
w=[1.31,1.33,1.34];
d=[1.1,1.15,1.21];
plot(x,k,'-bx',x,w,'-ms',x,d,'-ro')
h = legend('KLT','DWT','DCT',3);
Xlabel('COMPRESSION RAIO(%)')
Ylabel('SIMULATION TIME(sec.)')
Title('SIMULATION TIME PLOT FOR GRAY IMAGE');
grid on;
```