IMAGE COMPRESSION USING FRACTIONAL HARTLEY TRANSFORM

a thesis submitted in partial fulfillment of the requirements for the award of the degree of

MASTER OF ENGINEERING

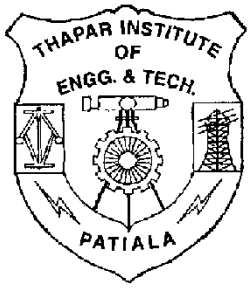
IN

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted By:
Pradeep Kumar
Roll No-8034111

Under the guidance of:

Kulbir Singh



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
THAPAR INSTITUTE OF ENGINEERING & TECHNOLOGY
(DEEMED UNIVERSITY)
PATIALA – 147004
INDIA

CERTIFICATE

I, Pradeep Kumar hereby certified that the work which is being presented in this thesis entitled **"Image compression using fractional Hartley transform"** by me in partial fulfillment of requirements for the award of degree of Master of Engineering in Electronics and Communication from Thapar Institute of Engineering and Technology (Deemed University), Patiala, is an authentic record of my own work carried under the supervision of Mr. Kulbir Singh.

The matter presented in this thesis has not been submitted in any other University / Institute for the award of any degree.

(Pradeep Kumar)
Signature of the Student

This is certified that the above statement made by the candidate is correct to the best of my knowledge

(Kulbir Singh)
Supervisor

(Dr R. S. Kaler)
HOD,ECED,
T.I.E.T, Patiala

(Dr D.S Bawa)
Dean of Academic Affairs
T.I.E.T, Patiala

ACKNOWLEDGEMENT

Apart from person efforts and steadfastness to work, constant inspiration and encouragement given by a number of individuals served as the driving force in earning this day in my life. To quote them all may be a difficult task but direct and indirect assistance and guidance received is gratefully acknowledged. I would like to express my feelings of gratefulness and submit my acknowledgements for them further in the following lines.

First of all, I take this opportunity to express my deep sense of gratitude to **Kulbir Singh**, Electronics and Communication Engineering Department, Thapar Institute Engineering & Technology, Patiala, under whom this research work has been completed. The invaluable guidance, constant encouragement and moral support throughout course of the work is gratefully acknowledged. Also I am thankful to him for providing excellent computational facilities and aid in preparation of the thesis. I consider my self fortunate to have had the opportunity to work with such as caring academician.

My thanks are due to **Dr. R.S. Kaler**, Head of the Electronics and Communication Engineering Department of this institute and all faculty members of the department for their help and moral support.

It escapes my understanding as to how I should express my appreciation and gratitude for my friends for their help and guidance.

Last but not the least many thank to my parents, sisters, for their efficiency, readiness and heartfelt appreciation to works all those who helped me directly or indirectly to accomplish my work in the end.

I am thankful and grateful to the almighty for bringing this day in my life.

PLACE: TIET, PATIALA **(PRADEEP KUMAR)**

ABSTRACT

As use and reliance on computers continues to grow, so does the need for efficient ways of storing large amount of data. For example, someone with a web page or online catalog that uses dozens or hundreds of images will more than likely need to use some form of image compression to store those images. The purpose of image compression is to achieve a very low bit rate representation, while preserving a high visual quality of decompressed images.

Basically there are two compression techniques, lossless and lossy. In lossy compression techniques various transforms are used. There are many transforms like Fourier transform, K-L transform, Haar transform, Hartley transform etc. In general the Fourier transform is difficult to apply because it depends on complex numbers. This problem can be avoided by using a similar transform known as the Hartley transform The Hartley transform maps real-valued sequences into real-valued frequency domain sequences.

This work aims to implement the image compression using fractional Hartley transform (FRHT). The fractional Hartley transform is the generalization of Hartley transform that depends on a parameter ‘a’, which provides the additional degree of freedom. With variation of its parameter ‘a’. It is found that by using fractional Hartley transform, high visual quality decompressed image can be achieved for same amount of compression as compared to Hartley transform. By varying ‘a’ to different values, an optimum value of ‘a’ can be achieved with low root mean square error (RMSE), better peak signal to noise ratio (PSNR) i.e. better quality of decompressed image.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE NO TITLE PAGE NO

LIST OF TABLES

TABLE NO TITLE PAGE NO

LIST OF ABBREVIATIONS

CR	Compression Ratio
DCT	Discrete Cosine Transform
DFRHT	Discrete Fractional Hartley Transform
DHT	Discrete Hartley Transform
FRFT	Fractional Fourier Transform
FRHT	Fractional Hartley Transform
FT	Fourier Transform
HT	Hartley Transform
PSNR	Peak Signal to Noise Ratio
PCM	Pulse Code Modulation
RLE	Run Length Encoding
RMSE	Root Mean Square Error

CHAPTER 1
INTRODUCTION

Efficient digital representation of image and video signals has been the subject of considerable research over the past 20 years. Modern image and video coding techniques offer the possibility to store or transmit the vast amount of data necessary to represent digital images and video in an efficient and robust way. New audio-video applications in the field of communications, multimedia and broadcasting became possible based on these digital image and video coding techniques [8]. Increasingly, medical images are acquired or stored digitally. This is especially true of grayscale images that are used in radiology applications. These images may be very large in size and number and compression offers a means to reduce the cost of storage and increase the speed of transmission [3].

Although the cost of storage is falling precipitously as the capacity per device increases and the cost of transmission bandwidth is also falling; there remains a strong demand for image compression. Since the speed of computing is also increasing dramatically, the sophistication and complexity of compression schemes that is practical for use is increasing. For transfer over networks with high bandwidth, or for storage on electromechanical devices (disk or tape), considerable time can be spent on compression before it becomes a factor in the total transfer time.

A digital image is composed of pixels, which can be thought of as small dots on the screen and it becomes more complex when the pixels are colored. An enormous amount of data is produced when a two dimensional light intensity function is sampled and quantized to create digital image. In fact, the amount of data generated may be so great that it results in impractical storage, processing and communication requirements [15].

1.1 Image Compression

Image compression means minimizing the size in bytes of a graphics file without degrading the quality of the image to an unacceptable level. The reduction in file size allows more images to be stored in a given amount of disk or memory space. It also reduces the time required for images to be sent over the internet or downloaded from web pages.

Uncompressed multimedia (graphics, audio and video) data requires considerable storage capacity and transmission bandwidth. Despite rapid progress in mass-storage density, processor speeds, and digital communication system performance, demand for data storage capacity and data-transmission bandwidth continues to outstrip the capabilities of available technologies. The recent growth of data intensive multimedia-based web applications have not only sustained the need for more efficient ways to encode signals and images but have made compression of such signals central to storage and communication technology [3, 11].

1.2 Need of Compression

An image, 1024 pixel*1024 pixel*24 bit, without compression, would require 3 MB of storage and 7 minutes of transmission, utilizing a high speed, 64 kbps, ISDN line. If the image is compressed at a 10:1 compression ratio, the storage requirement is reduced to 300 KB and the transmission time reduces to less than 6 seconds.

Image files, in an uncompressed form, are very large. And the internet, especially for people using a 56kbps dialup modem, can be pretty slow. This combination could seriously limit one of the web's most appreciated aspects - its ability to present images easily [8].

The figures in Table 1.1 show the qualitative transition from simple text to full-motion video data and the disk space, transmission bandwidth, and transmission time needed to store and transmit such uncompressed data.

The examples above clearly illustrate the need for sufficient storage space, large transmission bandwidth, and long transmission time for image,

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

Table 1.1: Multimedia data types and uncompressed storage space, transmission bandwidth, and transmission time required

Multimedia Data	**Size/Duration**	**Bits/Pixel or Bits/Sample**	**Uncompressed Size (B for bytes)**	**Transmission Bandwidth (b for bits)**	**Transmission Time (using a 28.8K Modem)**
A page of text	11'' x 8.5''	Varying resolution	4-8 KB	32-64 Kb/page	1.1 - 2.2 sec
Telephone quality speech	10 sec	8 bps	80 KB	64 Kb/sec	22.2 sec
Grayscale Image	512 x 512	8 bpp	262 KB	2.1 Mb/image	1 min 13 sec
Color Image	512 x 512	24 bpp	786 KB	6.29 Mb/image	3 min 39 sec
Medical Image	2048 x 1680	12 bpp	5.16 MB	41.3 Mb/image	23 min 54 sec
SHD Image	2048 x 2048	24 bpp	12.58 MB	100 Mb/image	58 min 15 sec
Full-motion Video	640 x 480, 1 min	24 bpp	1.66 GB	221 Mb/sec	5 days 8 hrs

	(30 frames/sec)				

1.3 Types of Compression

In the case of video, compression ratio causes some information to be lost; some information at a detailed level is considered not essential for a reasonable reproduction of scene. This type of compression is called lossy compression. Audio compression on the other hand, is not lossy. It is called lossless compression.

1.3.1 Lossless Compression

In lossless compression schemes, the reconstructed image, after compression, is numerically identical to the original image. However lossless compression can only a achieve a modest amount of compression. Lossless coding guaranties that the decompressed image is absolutely identical to the image before compression. This is an important requirement for some application domains, e.g. medial imaging, where not only high quality is in demand, but unaltered archiving is a legal requirement. Lossless techniques can also used for the compression of other data types where loss of information is not acceptable, e.g. text documents and program executables[3, 8].

1.3.2 Lossy Compression

Lossy is a term applied to data compression techniques in which some amount of the original data is lost during the compression process. Lossy image compression applications attempt to eliminate redundant or unnecessary information in terms of what the human eye can perceive. An image reconstructed following lossy compression contains degradation relative to the original. Often this is because the compression scheme

completely discards redundant information. However, lossy schemes are capable of achieving much higher compression. Under normal viewing conditions, no visible loss is perceived (visually lossless).

Lossy image data compression is useful for application to world wide web images for quicker transmission across the internet. An image reconstructed following lossy compression contains degradation relative to the original. Often this is because the compression scheme completely discards the redundant information [15].

1.4 Applications

Applications of data compression are primarily in transmission and storage of information. Image transmission application are in broadcast television, remote sensing via satellite, military applications via aircraft, radar and sonar, teleconferencing, computer communications, facsimile transmission etc. image storage is required for educational business documents, medical images that arise in computer tomography, magnetic resonance imaging and digital radiology, motion pictures, satellite images, weather maps etc. Application of data compression is also possible in the development of fast algorithms where the number of operations required to implement an algorithm is reduced by working with compressed data [15].

Over the years, the need for image compression has grown steadily. Currently it is recognized as an "Enabling Technology". It plays a crucical role in many important and diverse applications such as:

I. Business documents, where lossy compression is prohibited for legal reasons.
II. Satellite images, where the data loss is undesirable because of image collect cost.
III. Medical images, where difference in original image and uncompresse4d one can compromise diagnostic accuracy.
IV. Televideo conferencing.
V. Remote sensing.

VI. Space and hazardous waste control applications.

VII. Control of remotely piloted vehicles in military.

VIII. Facsimile transmission (FAX).

1.4 Fractional Hartley Transform

Before discussing the fractional Hartley transform fractional operations are discussed as follows:

1.4.1 Fractional Operations

Going from the whole of an entity to fractions of it represents a relatively major conceptual leap. The fourth power of 3 may be defined as $3^4 = 3\times3\times3\times3$, but it is not obvious from this definition that how $3^{3.5}$ might be defined. It must have taken sometime before the common definition $3^{3.5} = 3^{7/2} = \sqrt{3^7}$ emerged. The first and second derivatives of the function $f(x)$ is commonly denoted by : $\frac{df(x)}{dx}$ and $\frac{d^2 f(x)}{dx} = \frac{d}{dx}\left[\frac{df(x)}{dx}\right] = \frac{d[df(x)/dx]}{dx} = \left(\frac{d}{dx}\right)^2 f(x)$ respectively.

Similarly higher order derivatives are defined. Now what is meant by the 2.5th derivative of a function? It might not be clear from the above definition. Let F(μ) denote the FT of $f(x)$. The FT of the nth derivative of $f(x)$ [i.e. $\frac{d^n f(x)}{dx^n}$]is known to be given by $(i2\pi\mu)^n$F(μ), for any positive integer n. Now let us generalize this property by replacing n with the real order '**a**' and take it as the ath derivative of $f(x)$. Thus to find $\frac{d^a f(x)}{dx^a}$, the ath derivative of $f(x)$, find the inverse Fourier transform of $(i2\pi\mu)^a$F(μ). Both of these examples are dealing with the fractions of an operation performed on an entity, rather than fractions of the entity itself. $4^{0.5}$ is the square root of the integer 4. The function $[f(x)]^{0.5}$ is the square root of the function $f(x)$. But

$\frac{d^{0.5} f(x)}{dx^{0.5}}$ is the 0.5th derivative of $f(x)$, with $\left(\frac{df(x)}{dx}\right)^{0.5}$ being the square root of the derivative operator $\frac{d}{dx}$. The process of going from the whole of an entity to fractions of it underlies several of the more important conceptual developments. e.g. fuzzy logic, where the binary 1 & 0 are replaced by continuous values representing certainty or uncertainty of a proposition.[4, 12, 14].

1.4.2 Hartley and Fractional Hartley Transform

Conventionally, the unitary transforms have been widely used in signal compression, interpolation, and adaptive filtering. Some typical ones are the discrete Hadamard transform, the discrete Hartley transform (DHT) discrete Fourier transform (DFT), the discrete cosine transform (DCT), and the discrete sine transform (DST), among others.

The discrete Hartley transform (DHT) is an invertible linear transform closely related to the DFT. In the DFT, one multiplies each input by cos - i * sin (a complex exponential), whereas in the DHT each input is multiplied by simply cos + sin. Thus, the DHT transforms n real numbers to n real numbers, and has the convenient property of being its own inverse.

In recent years the concept of fractional operator and measure has been investigated extensively in many engineering and science applications. Four typical examples are described as follows. The first is fractional derivative and integral are defined by many mathematicians and applied to solve some physical problems. The second is the fractional Fourier transform has been studied in the optic community and signal processing area. The third is fractional dimension is used to measure some real word data such as coastline, dust in the air, clouds and neurons in the body. The fractional dimensional has being applied widely to pattern recognition and classification. The last is fractional lower-order moment has been used to analyze non-Gaussian signal which is more realistic than the Gaussian model in signal processing applications. On the other hand, various unitary

transforms have been widely used in image compression and adaptive filtering except Fourier transform. Some typical ones are sine, cosine and Hartley transform etc. So far, the fractional versions of these versions are generalized. The generalized form of Hartley transform is called as fractional Hartley transform [18].

1.6 Objective of Thesis

The objectives of thesis are:

- To achieve image compression using a new technique i.e. fractional Hartley Transform.
- To compare the performance of this new transform with existing transform i.e. Hartley transform.
- To implement fractional Hartley transform on many natural grayscale images.
- To study the effects of variations of CR on Root Mean Square Error, Peak signal to Noise Ratio.
- To compare the RMSE and PSNR for various value of fractional order 'a' for same compression ratio.

1.7 Methodology

PC Configuration:	AMD 1.33 GHz 256 Mb of RAM
Operating System:	Windows XP Professional, Version 2002(Service Pack 2)
Software:	MATLAB 6.1

1.8 Organization of Thesis

Chapter 1 discusses the introduction to compression, need of compression, types of compression, applications of image compression, introduction to fractional Hartley transform and objective of thesis. In chapter 2 the principles behind compression are discussed. Various types of data

redundancy i.e. coding redundancy, interpixel redundancy, psychovisual redundancy present in image are discussed. In chapter 3 image compression techniques i.e. lossless compression techniques and lossy compression techniques are discussed.

In chapter 4 the linear fractional transform and fractional Hartley transform are discussed. In chapter 5 the image compression model for fractional Hartley transform and the image compression characteristics are discussed. Chapter 6 provides the simulation results of five images i.e. Lena, Cameraman, Flower, Rice, Barbara with the variation of parameter 'a' using fractional Hartley transform. Chapter 7 describes the conclusions and prospects of the future work.

CHAPTER 2

PRINCIPLES BEHIND COMPRESSION

2.1 Introduction

There is a difference between information and data. Information is the message that is conveyed; data is the words used to convey it. Information is, simply put, unexpected information. There is significant data content between, "It's 10:00 and all's well," and "Fire!" The longer message is expected, so we don't really think much about it, hence little information. The shorter message is unexpected, and thus has very high information

content. One should basically judge the information content of a message by the difference in the way one would act if they heard the message and if they didn't.

Consider another example: if a letter is dropped from a word so that you only got "hel?o", you could probably guess that the letter "l" was missing. Hence, the letter "l" in this context has little information. However, if I gave you with word "?ook", you might equally guess "l", "b", "h", "c", and so on. In this case, the letter "l" would have significant information content. The very same symbol can have different information content depending on the context [8].

Redundancy is the difference between the shortest way one could convey a piece of information (i.e., the information content itself) and the data used to represent it. In the previous example, the first "l" is highly redundant while the second "l" is not.

There are three basic types of redundancy in images:

I. Coding Redundancy
II. Interpixel Redundancy
III. Visual Redundancy

All image compression algorithms attack one or more of these kinds of redundancy.

2.2 Coding Redundancy

Pixels have to be encoded in images. Like all things in a computer, we have to assign some arbitrary set of bits for different possible values. Sometimes, some values are more probable that others. As such, they are more expected and thus have less information content. Others are less probable (less expected) and thus have more information content. The key to reducing

coding redundancy is to assign bits according to the information content. The more information, the more bits; the less information, the fewer bits.

If we try to eliminate coding redundancy in textual information, we can measure the frequency with which each letter occurs. If it occurs more frequently, encode it with as few bits as possible. If it doesn't occur much at all, use more bits. Thus, we might encode "E" with a few bits and "X" or "Q" with lots of them. Who cares if "X" takes 12, 20, or even 50 bits--it doesn't occur that often anyway.

Likewise, we can use such *variable-length coding* for pixels or other information for images. For example, if an image is 50 and 50 black pixels? Why not use one bit to signal if the pixel is black or not, and then follow the non-black ones by the actual grey-level value (8 bits). This would use one bit for half the image and nine bits for the other half. This would use 4.5 bits per pixel instead of eight.

The best known and most efficient method for variable-length encoding is *Huffman coding*. Pure Huffman coding is computationally expensive and requires measuring the full statistics of the data. Typically, a modified form is used that is based on presumed frequency distributions and not-quite-perfect encoding [8].

2.3 Interpixel Redundancy

If you were to scan across a row of a binary image and read off the values, and if the values were "white", "white", "white", "white", "white", what would you guess the next one to be? While either "black" or "white" are possible, you'd probably guess that the next one is white. This makes sense for images because they usually have regions of constant or similar values. This is called *interpixel redundancy*.

In order to reduce the interpixel redundancies in an image, the 2-D pixel array normally used for human viewing and interpretation must be

transformed in to a more efficient format. For example, the differences between adjacent pixels can be used to represent an image. Transformations of this type are referred to as mapping. They are called reversible if the original image elements can be reconstructed from the transformed data set [8].

A common characteristic of most images is that the neighboring 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* or correlation between neighboring pixel values.
- *Spectral Redundancy* or correlation between different color planes or spectral bands.
- *Temporal Redundancy* or correlation between adjacent frames in a sequence of images (in video applications).

Image compression research aims at reducing the number of bits needed to represent an image by removing the spatial and spectral redundancies as much as possible. Since we will focus only on still image compression, we will not worry about temporal redundancy.

2.4 Visual Redundancy

The third way of reducing redundancy is to realize that our senses aren't equally sensitive to all things. In this sense, information content relates to the ability of our visual systems to tell the difference.

Examples of this include

- Our greater sensitivity to changes in intensity than to changes in hue.
- Our greater sensitivity to low-frequency changes than to high-frequency ones.

By transforming our image into spaces that correspond to these properties, we can then better reduce the number of bits we devote to them.

For example, by transforming to YIQ color spaces, we separate intensity from color and can correctly apportion the bits. Likewise, by transforming to frequency-based spaces (such the frequency domains of the Fourier Transform or DCT), we can distribute our precious bits according to spatial frequency.

Psychological redundancy is fundamentally different from the redundancies discussed earlier. Unlike coding and interpixel redundancy, psychologically redundancy is associated with real or quantifiable visual information. Its elimination is possible only because the information itself is not essential for normal visual processing. Since the elimination of psycho visually redundant data results in a loss of quantitative information, it is commonly referred to as quantization. This terminology is consistent with normal usage of the word, which generally means the mapping of a broad range of input values to a limited number of output values. As it is an irreversible operation (visual information is lost), quantization results in lossy data compression [3, 8].

CHAPTER 3

IMAGE COMPRESSION TECHNIQUES

3.1 Introduction

Basic image compression techniques are of two types, lossless compression techniques and lossy compression techniques. In lossless compression scheme, the reconstructed image after compression, is numerically identical to the original image, i.e. original image can be reconstructed without any errors. On the other hand lossy compression schemes involve some loss of information, and data can not be recovered exactly. Often this is because the compression completely discards redundant information [15].

3.2 Lossless Compression Techniques

In lossless compression scheme, the reconstructed image after compression, is numerically identical to the original image, i.e. original image can be reconstructed without any errors. However lossless compression can only achieve modest amount of compression. This is important for applications like compression of text. It is very important that the reconstruction is identical to the original text, as very small differences can result in statements with very different meanings. Consider the sentences, "do now send money" and "do not send money". A similar argument holds for computer files and for certain types of data such as bank records. Various techniques for lossless compression are below:

3.2.1 Pixel Coding

In these techniques each pixel is processed independently; ignoring the inter pixel dependencies. Some pixel coding are discussed as follows:

3.2.1.1 Optimal Entropy Pixel Coding

By using an appropriate code, the bit rate of the encoded data can come close to the entropy of the data itself. This is achieved by using a code in which the length of each codeword is proportional to the frequency that it is used, such as a Huffman code or arithmetic coding. On the test images this resulted in a rate of 7.39 bits per pixel -- a 7% compression.

3.2.1.2 Adjacent Pixel Pair Coding

Neighboring pixels are generally highly correlated. In order to exploit this fact, the code may be chosen so that pixels are coded in sequential pairs rather than alone. Since there are more possibilities, this will increase the code rate; on the test images a rate of 11.98 bits per symbol was achieved. Since the symbols each represent 2 pixels, this translates to a normalized rate of 5.99 bits per pixel, or a 25% compression.

3.2.1.3 Predictive Pixel Coding

Another way to exploit this correlation is to predict the pixel value based on its neighbors, and then to code only the error. The values of the majority of these errors should be a small number of discrete integers, and hence these frequent errors should compress well.

3.2.1.4 Prediction using Previous Pixel

Prediction of a pixel as the value of the adjacent left pixel resulted in a rate of 5.31 bits per pixel for the error, a compression of 33%. This is also known as differential pulse code modulation (DPCM).

3.2.1.5 Prediction using Neighborhood Pixels

It is logical that with more information, prediction could be improved. Prediction using all adjacent received pixels was then implemented using first the minimum variance predictor, and then the minimum entropy predictor. The minimum variance predictor yielded 4.85 bits per pixel rate for 39% compression. The minimum entropy predictor

resulted in a rate of 4.90 bits per pixel, for 38% compression. The 2 predictors are so close that the difference between them in this case is more likely due to image content than general superiority of design [11].

3.2.2 PCM Coding

In PCM the incoming video signal is sampled, quantized and coded by a suitable code word (before feeding it to a digital modulator for transmission). The quantizer output is generally coded by a fixed-length binary code word having *B* bits. Commonly, 8 bits are sufficient for monochrome broadcast or vide4o conferencing quality images, whereas medical images or color video signals may require 10 to 12 bits per pixels.

The number of quantizing bits needed for visual display of images can be reduced to 4 to 8 bits per pixel by using companding, contrast quantization, or diethering techniques. Halftone techniques reduce the quantizer output to 1 bit per pixel, but usually the input sampling rate must be increased by a factor of 2 to 16. The compression achieved by these techniques is generally less than 2:1 [3].

In terms of a mean square distortion, the minimum achievable rate by PCM is given by the rate distortion formula:

$$R_{PCM} = \frac{1}{2}\log_2 \frac{\sigma_u^2}{\sigma_q^2}, \qquad \sigma_q^2 \langle \sigma_u^2$$

(3.1)

where σ_u^2 is the variance of the quantizer input and σ_q^2 is the quantizer mean square distortion.

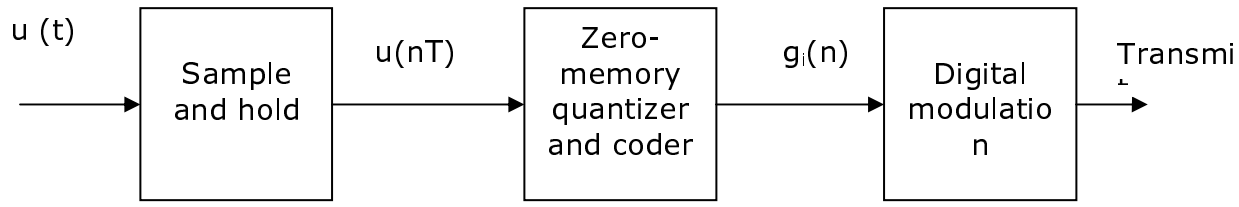
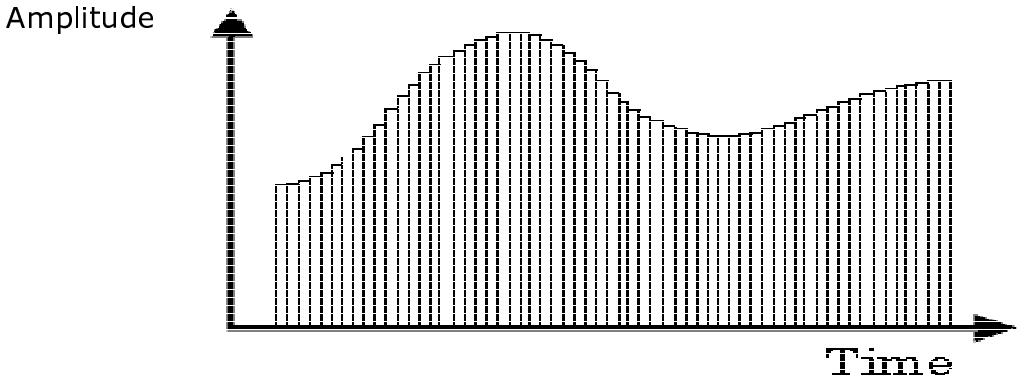
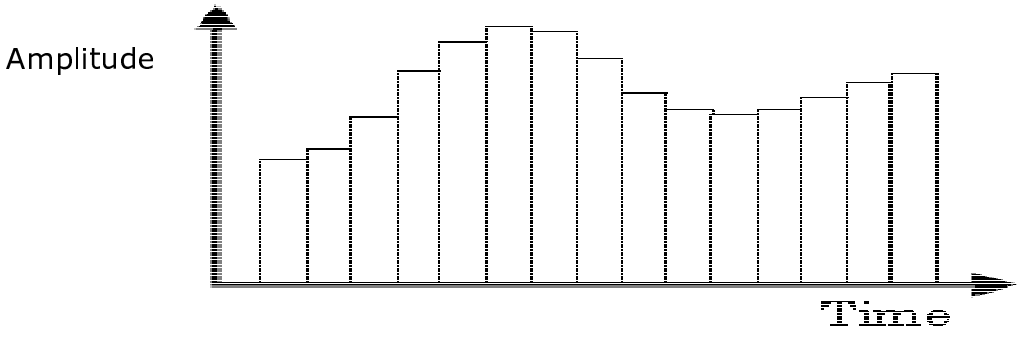


Figure3.1: Block diagram of PCM

An important term in the recording of sound waves is sampling taking a representation of the waveform at a given moment. The frequency between each sample is one key aspect in determining the accuracy of the waveform representation. In the two diagrams below, each vertical bar represents a time-slice the top of the bar representing the waveform. It can be seen quite clearly that the more frequent the *sample* the more accurate the waveform representation.



(a)



(b)

Figure3.2: Pulse code modulation with varying step size

The next influential factor is that of resolution, that is, the smallest level of change that can be detected in the waveform referred to as Quantization levels [13]. The following diagram attempts to illustrate this. The top-most peak of the wave doesn't reach the threshold of the next quantization level and is therefore truncated to the lower level. The result is that further detail is lost, therefore reducing the quality of the sound.

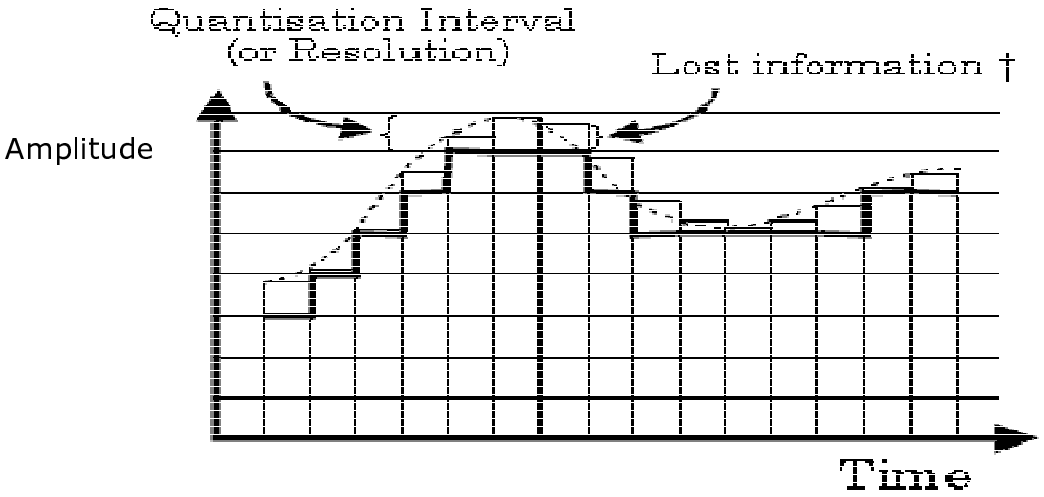


Figure3.3: Error in quantization

Knowing this and the information mentioned earlier, it can be used to effectively compress audio data such that the data requirements are reduced but the information is still intact, referred to as downsampling. Waveform coding and downsampling can be applied to visual as well as audio signals (or data) as light is also a waveform consisting of both colour and intensity values. In bitmap graphics, downsampling refers to reducing the resolution, that is, the number of dots per inch (dpi) or reduction in the color depth, for example, from a 16 million to 256 color palette.

Application of PCM

The above describes the basis of Pulse Code Modulation (PCM) which is used significantly in computers for digitizing analogue signals. It can be implemented in both hardware and software. There are various implementations and variations of PCM, for example, Differential Pulse Code Modulation (DPCM), which is also used by the JPEG

graphics compression, Adaptive DPCM (ADPCM) and Interactive Multimedia Association ADPCM (IMA ADPCM) [20].

3.2.3 Entropy Coding

If the quantized pixels are not uniformly distributed, then their entropy will be less than *B*, and there exist a code that uses less than *B* bits per pixels. In entropy coding the goal is to encode a block of *M* pixels containing *MB* bits with probabilities p_i, i=0, 1, 2, ……, *L-1*, *L*=2^{MB}, by $-\log_2 p_i$ bits, so that the average bit rate is

$$\sum_i p_i(-\log_2 p_i) = H \qquad (3.2)$$

This gives a variable-length code for each block, where highly probable blocks (or symbols) are represented by small-length codes, and vice versa. If $-\log_2 p_i$ is not an integer, the achieved rate exceeds *H* but approaches it asymptotically with increasing block size. For a given block size, a technique called *Huffman coding* is the most efficient fixed to variable length encoding method.

3.2.3.1 The Huffman Coding Algorithm

As there are many interpretations and enhancements to this method, we will take a brief look at the principle behind it. This method first determines the frequency with which symbols appear within the data. A binary-tree structure is then generated by sequentially combining the two least-frequent symbols via a node with the most frequent of the two on the left. The value of the node then becomes the sum of the two symbols' frequencies, and is subsequently treated as a symbol and included with the other symbols in the completion of the tree. Once the tree has been generated, the new binary representation of a symbol is ascertained by traversing the tree (in one direction) from the root node to the symbol recording a 0 whilst traversing a left-path and 1 for a right-path. For example, consider the symbols that appear in the sentence: "It then encodes each symbol with a variable-length code - the more frequent the symbol the shorter the code."

Below is a table showing the frequency of appearance of each symbol along with the number of *bits* required to represent them all using a fixed 8-*bit* representation. Along side is a possible representation of the bit-pattern generated by this compression technique, and the total number of *bits* required [7, 16].

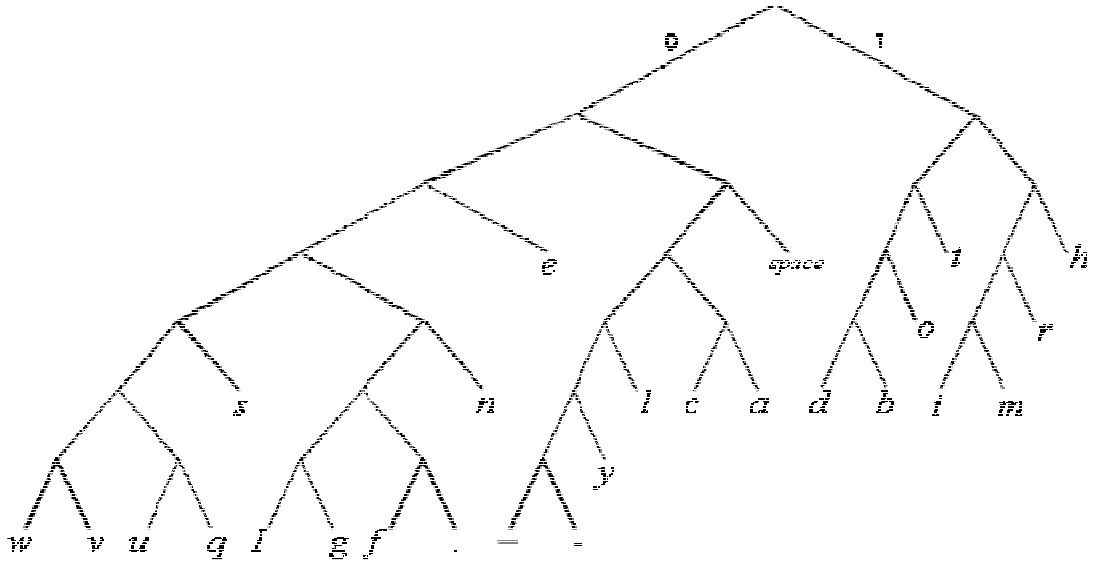


Figure3.4: Huffman code tree

Table3.1: Comparison of uncompressed and compressed by Huffman coding

Symbol	***f***	***Number of Bits required for symbols (uncompressed)***	***New Binary Representation***	***Number of Bits required for symbols (compressed)***
space	18	144	011	54
e	16	128	001	48
t	10	80	101	30
h	9	72	111	27
o	7	56	1001	28
r	5	24	1101	20

a	4	32	01011	20
c	4	32	01010	20
l	4	32	01001	20
n	4	32	00011	20
s	4	32	00001	20
b	3	24	10001	15
d	3	24	10000	15
m	3	24	11001	15
i	2	16	11000	10
y	2	16	010001	12
-	1	8	0100001	7
—	1	8	0100000	7
.	1	8	0001011	7
f	1	8	0001010	7
g	1	8	0001001	7
I	1	8	0001000	7
q	1	8	0000011	7
u	1	8	0000010	7
v	1	8	0000001	7
w	1	8	0000000	7
Total:	Σ 108	Σ 848		Σ 444

Advantages:

1. Algorithm is easy to implement

2. Produce a lossless compression of images

Disadvantage:

1. Efficiency depends on the accuracy of the statistical model used and type of image. Kay and Levine state that "The algorithm varies with different formats, but few get any better than 8:1 compression."

2. Compression of image files that contain long runs of identical pixels by Huffman is not as efficient when compared to RLE.

3. The Huffman encoding process is usually done in two passes. During the first pass, a statistical model is built, and then in the second pass the image data is encoded based on the generated model. From here we can see that Huffman encoding is a relatively slow process as time is required to build the statistical model in order to archive an efficient compression rate.

4. Another disadvantage of Huffman is that, all codes of the encoded data are of different sizes (not of fixed length). Therefore it is very difficult for the decoder to know that it has reached the last bit of a code, and the only way for it to know is by following the paths of the up-side down tree and coming to an end of it (one of the branch). Thus, if the encoded data is corrupted with additional bits added or bits missing, then whatever that is decoded will be wrong values, and the final image displayed will be garbage

3.2.4 Run-Length Encoding (RLE)

Instead of using individual codes for black and white pixels in binary images, we can encode runs of black or white pixels. The sequence 111111000000000011111 thus becomes 6 1s, 10 0s, and 5 1s. This is called *run-length encoding* or RLE.

An RLE image is thus encoded as a sequence of (count, value) pairs. Typically, there is fixed size for the count value, such as one byte. In this case, a run of 500 1s would be (256, 1), (244, 1). The size of the count value is a tradeoff: if too small, you end up encoding long runs with several short pieces instead of one, if too large, there's wasted space used for short runs.

There are some perverse cases where RLE can actually cause *negative compression*--cases where the compressed data, because of overhead, takes more space than the original. A pattern of alternating black and white pixels, for example, would be

encoded as (1,1),(1,0),(1,1),(1,0), etc. Instead of one bit per pixel, we're using one bit plus the length of the count field!

RLE is usually used for binary images, and is commonly used for FAX transmission encoding or other scanned paper documents. These use not just runs across one row, but also use redundancy across the rows.

RLE can also be used for grey-level images, though. Typically regular RLE won't work for noisy grey-level images because the value changes from pixel to pixel. However, if we look at the bit planes of the grey-level values, we see that there are large runs where the most-significant bit is 1 (lighter regions) and others where it is 0 (darker regions). It doesn't make sense to use this all the way down to the lease-significant bit, but is useful for the most-significant bits when combined with regular encoding of the least significant bits.

Consider the following example:

Here we have a series of blue x 6, magenta x 7, red x 3, yellow x 3 and green x 4, that is:

which is clearly a representation using fewer *bits* (only 2/3 the original).

This would not be a feasible method of compression if the raw data did not contain repeating symbols. In such a circumstance the compressed data would probably be larger. Consider the following:

This would give:

which is twice the size!

One advantage of this method is that it is sequential — once a particular series has been counted it could be transmitted. Consequently the principles of this method are also employed by the CCITT codec for fax communication in conjunction with the Huffman method [7].

Advantage:

- Algorithm is simple to implement
- Fast to execute
- Produce lossless compression of images

Disadvantage:

- Compression Ratio not as high compared to other more advanced compression algorithms.

3.2.5 Bit-Plane Coding

A 256 gray level image can be considered as a set of eight 1-bit planes, each of which can be run-length encoded. For 8-bit monochrome images, compression ratio of 1.5 to 2 can be achieved. This method becomes very sensitive to channel errors unless the significant bit planes are carefully protected.

Break image up into bit planes and apply run length coding to each plane.To get the bit planes of the above 8-bit grayscale image, AND the picture with the binary value corresponding to the desired plane; *i.e.*, ANDing the image with 10000000 should give the 7^{th} bit plane, ANDing the image with 00100000 should give the 5^{th} bit plane, etc. After the AND operation, use the thresholding operation to make a binary image. It may also be necessary to invert the result [3].

3.2.6 Lossless Predictive Coding

Let us turn to now an error free compression approach that does not require decomposition of an image into a collection of bit planes. The approach, commonly

referred to as lossless predictive coding, is based on eliminating the interpixel redundancies of closely spaced pixels by extracting and coding only the new information in each pixel. The new information of a pixel is defined as the difference between the actual and predictive value of that pixel.

Figure 3.8 shows the basic components of a lossless predictive coding system. The system consists of an encoder and a decoder, each containing an identical predictor. As each successive pixels of the input image, denoted f_n is introduced to the encoder, the predictor generates the anticipated value of that pixel based on some number of past inputs. The output of the predictor is then rounded to the nearest integer, denoted as $\hat{f}_n$, and used to form the difference or prediction error

$$e_n = f_n - \hat{f}_n$$

which is using a variable length code (by the symbol encoder) to generate the next element of the compressed data stream. The decoder reconstructs e_n from the received variable length code words and performs the inverse operations.

$$f_n = e_n + \hat{f}_n$$

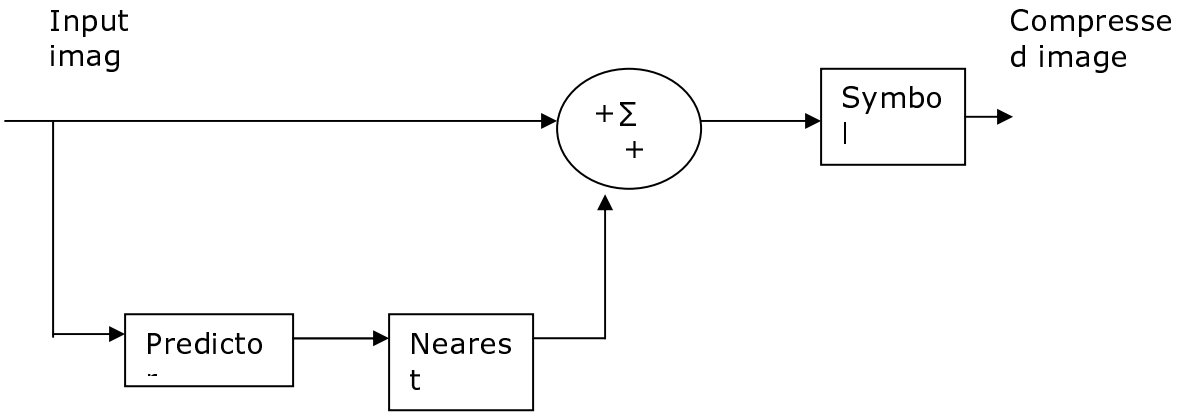
Variable local, global and adaptive methods can be used to generate $\hat{f}_n$. In the most cases, however, the prediction is formed by a linear combination of m previous pixels. That is,

$$\hat{f}_n = round\left[\sum_{i=1}^{m} \alpha_i f_{n-1}\right]$$

Where m is the order of the linear predictor, round is a function used to denote the rounding or nearest integer operation, and the α_i for i=1, 2, 3....,m are prediction coefficient. In raster scan application, the subsequent n indexes the predictor outputs in accordance with their time of occurrence. That is, f_n, $\hat{f}_n$ and e_n. In other cases, n is used as an index on the spatial coordinates and/or frame number (in a time sequences of

images) of an images. In 1-D linear predictive coding. For an example above equation can be written as

$$\hat{f}_n(x,y) = round\left[\sum_{i=1}^{m} \alpha_i f_{n-1}(x, y-i)\right]$$



(a)

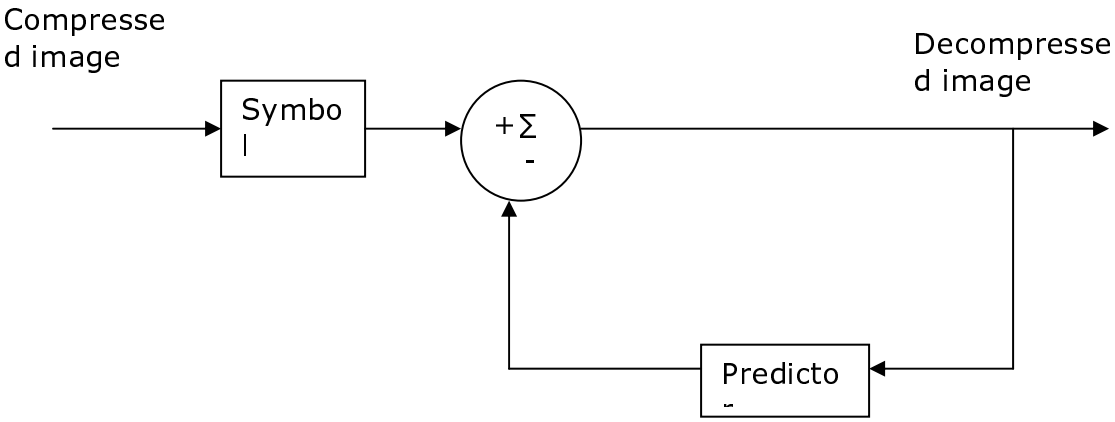


Figure 3.5: A lossless predictive coding model: (a) Encoder; (b) Decoder

where each subscribed variable is now expressed explicity as a function of spatial coordinates x and y. Note that the 1-D linear prediction $\hat{f}_n(x,y)$ is a function of the previous pixels on the current line alone. In 2-D predictive coding, the prediction

function is a function of the previous pixels in a left to right, top to bottom scan of an image. In the 3-D case, it is based on these pixels and the previous pixels of preceding frames [15].

3.3 Lossy Compression Techniques

Lossy compression techniques involve some loss of information and data that have been compressed using lossy techniques generally can not be recovered or reconstructed exactly. Often this is because the compression completely discards redundant information. However, lossy schemes are capable of achieving much higher compression. This is important for application like TV signals, Teleconferencing. Here is tradeoff between compression and accuracy [15, 19]. Various techniques for lossy compression are below:

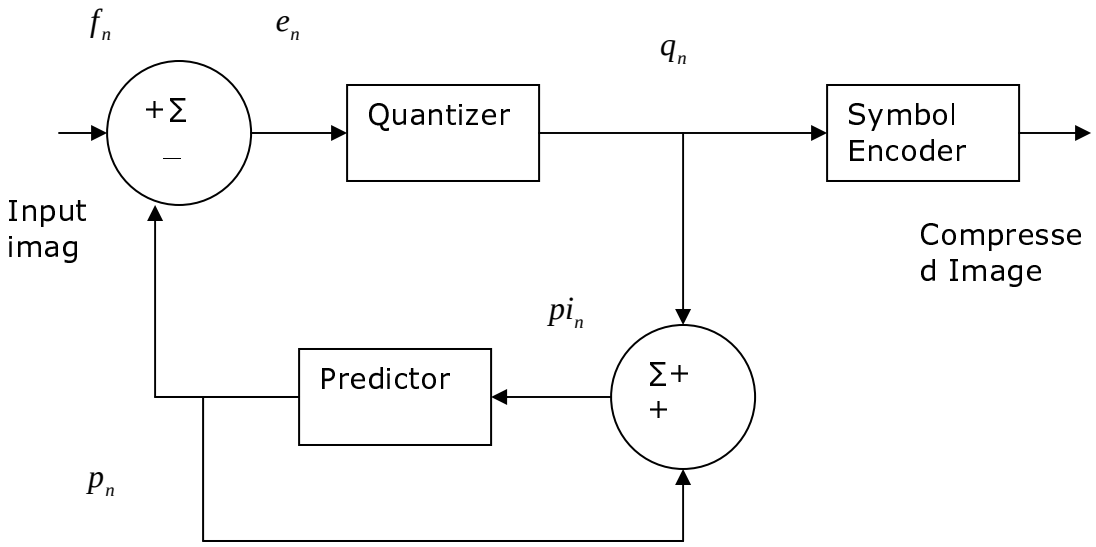
3.3.1 Lossy Predictive Coding

A quanytizer, that also executes rounding, is now added between the calculation of the prediction error e_n and the symbol encoder. It maps e_n to a limited range of values q_n and determines both the amount of extra compression and the deviation of the error free compression. This happens in a closed circuit with the predictor to restrict an increase in errors. The predictor does not use e_n but rather q_n, because both the encoder and decoder know it [3].

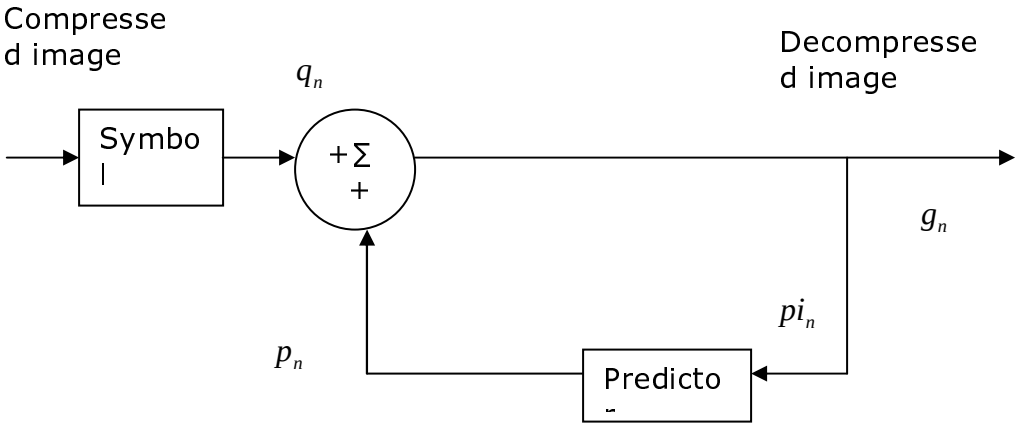
3.3.2 Transform Coding

By transforming an image from one space to another, we can not only separate different visual properties, but we can also change the predictive (and thus information containing) properties of the data. If we can transform it so that there is more coding or interpixel redundancy, we can compress the image better.

Energy compaction is a term often used to describe the ability of a particular transform to separate the information content and compact it into one part of the space. As we discussed earlier, the Hotelling or Karthunen-Loeve transform has optimal energy compaction [8].



(a)



(b)

Figure3.6: A lossy predictive coding model: (a) Encoder, (b) Decoder

In this section, we consider compression techniques that are based on modifying the transform of an image. In transform coding, a reversible, linear transform is used to map

the image in to a set of transform coefficients, which are then quantized and coded. For most natural images, a significant number of coefficients have small magnitudes and can be coarsely quantized (or discarded entirely) with little image distortion. Any of the transforms can be used to transform the image data.

3.3.2.1 General Model

Figure1 shows a typical transform coding system. The decoder implements the inverse sequence of steps (with the exception of the quantized function) of the decoder, which performs four relatively straight forward operations: sub image decomposition, transformations, quantization and coding**.**

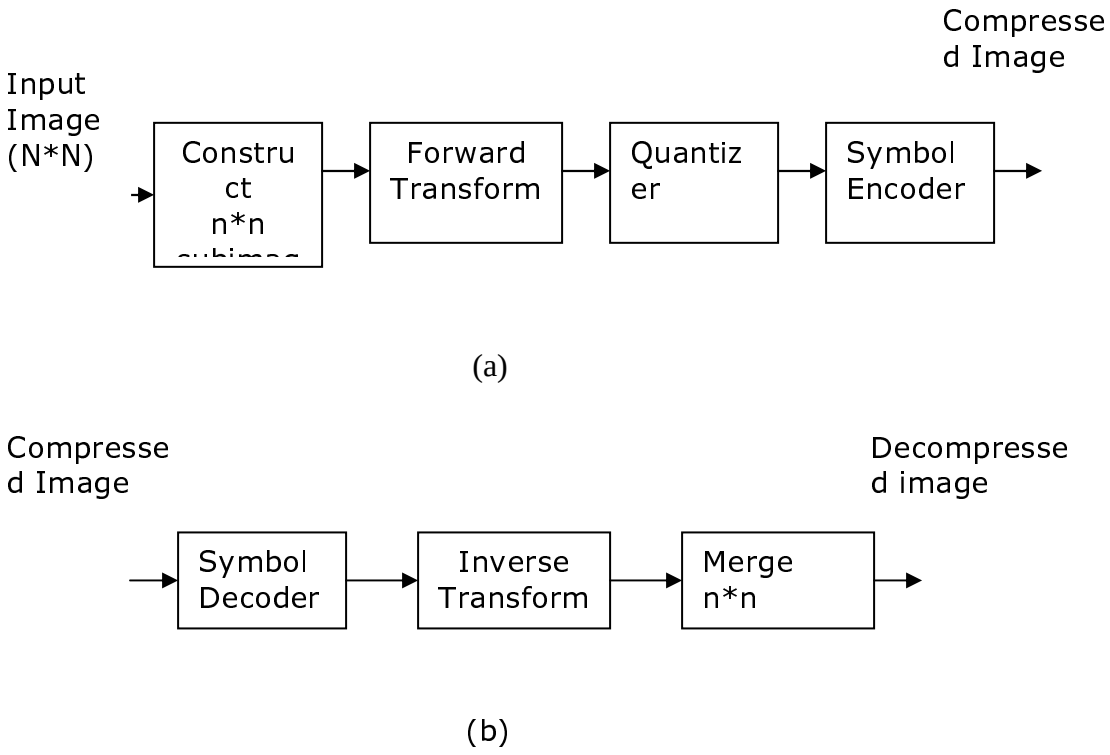


Figure 3.7: A transform coding system: (a) Encoder; (b) Decoder

An N*N input image first is subdivided in to sub images of size n*n, which are then transformed to generate $(N/n)^2$n*n sub image transform arrays. The goal of the transformation process is to decorrelate the pixels of

sub image, or to pack as much information as possible into the smallest number of transform coefficients. The quantization stage then selectively eliminates or more coarsely quantizes the coefficients that carry the least information. These coefficients have the smallest impact on reconstructed sub image quality. The encoding process terminates by coding (normally using a variable length code) image content, called adaptive transform coding, or fixed for all images, called no adaptive transform coding [11].

3.3.2.2 Transform Selection

Transform coding systems based on the Karhuen-Loeve transform(KLT), discrete Fourier transform(DFT), discrete Hartley transform(DHT), Walsh-Hadamard transform (WHT) and various other transforms have been constructed and/or studied extensively. The choice of a particular transform in a given application depends upon the amount of reconstruction error that can be tolerated and the computational resources available. Compression is achieved during the quantization of the transformed coefficients not during the transformation step [1].

3.3.2.3 Sub image Size Selection

A significant factor affecting transform coding, error and computational complexity is function of sub image size. In most application images are sub divided so that the correlation (redundancy) between adjacent sub images is reduced to some acceptable level so that n can be expressed as an integral power of 2, where n is the sub image dimension. The latter condition simplifies the computational of the sub image transformation. In general, both the levels of compression and computational complexity increase as the sub image size increases. The most popular sub image sizes are 8*8 and 16*16 [5, 10].

3.3.2.4 Bit Allocation

The reconstruction error associated with quantization is a function of number and relative importance of the transformed coefficient of the discarded, as

well as precision of the retained coefficient. In transformed, coding system, the retained coefficient can be selected on the basis of maximum variance, called zonal coding, or on the basis of maximum magnitude, called threshold coding. The overall process of truncating, quantizing and coding the coefficient of the transformed sub image is commonly called bit allocation [21].

Zonal coding is based on the information theory. The zonal sampling process can be viewed, as multiplying each transformed coefficient by corresponding element in a zonal mask, which is constructed by placing 1 in the location of maximum variance and 0 in all other locations. The coefficient retained during the zonal sampling process must be quantized and coded. Zonal coding is usually implemented by using usually a single fixed mask for all sub images.

Threshold coding, however, is inherently adaptive in the sense that the location of the transform coefficient retained for each sub image varies from one image to another. There are three basic ways to threshold a transformed sub image:

1. A single global threshold can be applied to all sub images.
2. A single global threshold can be a applied to all sub images.
3. The threshold can be varied as a function of each coefficient with in the sub image.

In the first approach, the level of compression differs from image to image, depending on the number of coefficient that exceeds the global threshold. In the second, called N-largest coding, the same number of coefficient is discarded for each sub image. As a result, the code rate is constant and is known in advance. The third technique, like the first, results in a variable code rate, but offers the advantage the thresholding and quantization can be combined.

3.3.2.5 Quantization Matrix

Here a particular quantization matrix have not been chosen, instead it can be defined at runtime when compression takes place. Selecting quantization matrix at ran-time is just like to dial in a picture quality value when compressing graphic.

3.3.2.6 Coding

The final step in the compression process is coding the quantized image. First the coefficients of the image are arranged in the zig-zag sequence. Then they have been encoded using run length coding of zeros values and variable length integer coding.

3.3.2.6.1 The Zig-Zag Sequence

A simple code has been developed that gives the count of consecutive zero values in the quantized image, this give a measure for outstanding compression. Because quantum bands of some values are arranged along the diagonals of the pixel matrix, so as to increase the length of run levels of quantized coefficient arranged in the zig-zag sequence.

3.3.2.6.2 Run Length and Variable Length Integer Coding

After reordering the quantized block in the zig-zag sequence. Basically, the output of the run-length and variable length integer encoding consists of a sequence of three tokens, repeated until the block is complete. The three tokens are like this:

Run length: The numbers of consecutive zeros precede the current element in the output matrix.

Bit Count: The number of bits to follow in the amplitude number.

Amplitude: The amplitude of the transform coefficient, the variable length integer coding schemes takes advantage of the fact that the transform

output should consist of mostly smaller numbers, which are required to encode with smaller number of bits [15].

CHAPTER 4

FRACTIONAL HARTLEY TRANSFORM

4.1 Introduction

In recent years the concept of fractional operator and measure has been investigated extensively in many engineering and science applications. Four typical examples are described as follows. The first is fractional derivative and integral are defined by many mathematicians and applied to solve some physical problems. The second is the fractional Fourier transform has been studied in the optic community and signal processing area. The third is fractional dimension is used to measure some real word data such as coastline, dust in the air, clouds and neurons in the body. The fractional dimensional has being applied widely to pattern recognition and classification. The last is fractional lower-order moment has been used to analyze non-Gaussian signal which is more realistic than the Gaussian model in signal processing applications [12].

On the other hand, various unitary transforms have been widely used in image compression and adaptive filtering except Fourier transform. Some typical ones are sine, cosine and Hartley transform etc. A generalized linear fractional transform is discussed in the proceeding section.

4.2 Linear Fractional Transform

Let T be a linear transform that maps the function $f(u)$ into the function $g(u')$, i.e.

$$T(f(u)) = g(u') \tag{4.1}$$

where variables u, u' are different because the domain would be change after transformation. Then the corresponding fractional transform of T is defined as:

$$T^{\alpha}(f(u)) = g^{\alpha}(u') \tag{4.2}$$

which must satisfy the boundary conditions

$$T^{0}(f(u)) = f(u) \tag{4.3}$$

$$T^{1}(f(u)) = g(u') \tag{4.4}$$

and the additive property

$$T^{\beta}(T^{\alpha}(f(u))) = T^{\alpha+\beta}(f(u)) \tag{4.5}$$

Given the transform T, the problem is how to define T^{α} such as boundary condition and additive property are satisfied. A systematic procedure will be described as follows. Let $e_n(u)$ be an eigenfunction of linear transform T with eigenvalue λ_n, then we have

$$T(e_n(u)) = \lambda_n e_n(u') \tag{4.6}$$

If $e_0(u), e_1(u), \ldots\ldots, e_\infty(u)$ is an orthonormal basis in the x domain, the function $f(u)$ can be rewritten as

$$f(u) = \sum_{n=0}^{\infty} a_n e_n(u)$$

(4.7)

Where the coefficient a_n is given by

$$a_n = \int_{-\infty}^{\infty} f(u) e_n(u) dx$$

(4.8)

Then the transform of $f(u)$ is given by

$$T(f(u)) = T(\sum_{n=0}^{\infty} a_n e_n(u)) = \sum_{n=0}^{\infty} a_n \lambda_n e_n(u')$$

(4.9)

Based on this equation the fractional transform T^α is defined as

$$T^\alpha(f(u)) = \sum_{n=0}^{\infty} a_n \lambda_n^\alpha e_n(u')$$

(4.10)

From equation (4.8) and (4.10) it can be obtained

$$T^\alpha(f(u)) = \sum_{n=0}^{\infty} a_n \lambda_n^\alpha e_n(u')$$

$$= \sum_{n-0}^{\infty} \left[\int_{-\infty}^{\infty} f(u) e_n(u) dx \right] \lambda_n^\alpha e_n(u')$$

$$= \int_{-\infty}^{\infty} K^\alpha(u, u') f(u) du$$

(4.11)

Where transform kernel is defined by

$$K^\alpha(u, u') = \sum_{n=0}^{\infty} \lambda_n^\alpha e_n(u) e_n(u')$$

(4.12)

From equation (4.11) it is clear that T^{α} is a linear transform and from equation (4.7), (4.9), (4.10) and it can be shown the boundary conditions are satisfied [4, 18]. Since the expression

$$T^{\alpha}\left(e_n(u)\right) = \lambda_n^{\alpha} e_n(u) \tag{4.13}$$

holds and T^{α} is linear, we have

$$\begin{aligned} T^{\alpha}\left(T^{\beta}\left(f(u)\right)\right) &= T^{\alpha}\left(\sum_{n=0}^{\infty} a_n \lambda_n^{\beta} e_n(u')\right) \\ &= \sum_{n=0}^{\infty} a_n \lambda_n^{\beta}\left[T^{\alpha}(e_n(u'))\right] \\ &= \sum_{n=0}^{\infty} a_n \lambda_n^{\beta}\left[\lambda_n^{\alpha} e_n(u')\right] \\ &= \sum_{n=0}^{\infty} a_n \lambda_n^{\alpha+\beta} e_n(u') \end{aligned} \tag{4.14}$$

This tells us the additive property is also satisfied. Given linear transform T, the procedure to find its fractional is now summarized as follows:

Find the eigenfunctions $e_n(u)$ and eigenvalues λ_n of transform T for n=0, 1, 2……., ∞.

Use equation (4.12) to compute the transform kernel $K_{\alpha}(u,u')$. The closed form formula may be obtained.

The fractional transform of $f(u)$ is then given by

$$T^{\alpha}\left(f(u)\right) = \int_{-\infty}^{\infty} K_{\alpha}(u,u')\, f(u)\, du \tag{4.15}$$

4.2 Fractional Hartley Transform

If linear transform T is Hartley transform, then it is defined as

$$T\left(f\left(u\right)\right)=\ g_{H}\left(u'\right)=\ \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}f\left(u\right)cas\ \left(uu'\right)du$$

(4.16)

where $cas(uu')=\cos(uu')+\sin(uu')$

The relation between HT and FT is given by

$$g_{H}\left(u'\right)\ =\ \frac{1+\ j}{2}g_{F}\left(u'\right)+\frac{1-\ j}{2}g_{F}\left(-u'\right)$$

(4.17)

Where $g_F(u')$ is the Fourier transform of $f(u)$.

From the equation (4.17) and the symmetric property of Hermite polynomial

$$H_n(-u)=\begin{cases}H_n(u).....if\ ..n..is..even\\ H_n(-u).....if\ ..n..is..odd\end{cases}$$

The eigenvectors of the Hartley transform are same those of Fourier transform, but the eigenvalues λ_n of eigenvector $e_n(u)$ are different and given by

$$\lambda_n=\begin{cases}e^{-jn\pi/2}.....if\ ....n....is..even\\ e^{-j(n-1)\pi/2}...if\ ....n....is..odd\end{cases}$$

Thus the transform kernel $K_H^{\alpha}(u,u')$ of fractional Hartley transform is given by

$$K_{H}^{\alpha}\left(u,u'\right)\ =\ \sum_{n=0}^{\infty}\lambda_{n}^{\alpha}e_{n}\left(u\right)e_{n}\left(u'\right)$$

$$=\ \left[E\left(u,u'\right)+\ O\left(u,u'\right)\right]e^{-\left(u^{2}+u'^{2}\right)/2}$$

(4.18)

where $E(u,u')$ and $O(u,u')$ are defined as

$$E(u,u') = \sum_{n=0}^{\infty} \frac{e^{-jn\alpha\pi}}{2^{2n}(2n)!\sqrt{\pi}} H_{2n}(u) H_{2n}(u')$$

$$O(u,u') = \sum_{n=0}^{\infty} \frac{e^{-jn\alpha\pi}}{2^{2n+1}(2n+1)!\sqrt{\pi}} H_{2n+1}(u) H_{2n+1}(u')$$

The fractional Fourier transform kernel $K_F^{\alpha}(u,u')$ is given by

$$K_F^{\alpha} = \left[E(u,u') + e^{-j\alpha\pi/2} O(u,u')\right] e^{-\left(u^2+u'^2\right)/2}$$

(4.19)

Using the symmetry property of Hermite polynomial it can be shown that

$$E(u,-u') = E(u,u')$$

$$O(u,-u') = -o(u,u')$$

From the equation (4.16), (4.18), (4.19) the relation between FRFT kernel and FRHT kernel can be given as

$$K_H^{\alpha} = \frac{1+e^{j\alpha\pi/2}}{2} K_F^{\alpha}(u,u') + \frac{1-e^{j\alpha\pi/2}}{2} K_F^{\alpha}(u,-u')$$

(4.20)

From above equation the formula FRHT kernel is given by

$$K_H^{\alpha}(u,u') = \sqrt{\frac{1-j\cot\phi}{2\pi}} e^{j\frac{u^2+u'^2}{2}\cot\phi} (\cos(uu'\csc\phi)) + e^{j(\phi-\frac{\pi}{2})} (\sin(uu'\csc\phi)$$

(4.21)

Based on this kernel, the FRHT of a function $f(u)$ is defined as

$$g_H^{\alpha}(u') = \int_{-\infty}^{\infty} K_H^{\alpha}(u,u')\, f(u)\, du$$

(4.22)

It is clear that the FRHT of real function $f(x)$ is complex except $\phi = k\pi/2$. Where k is odd integer [6, 9, 17].

Thus the period of fractional Hartley transform is 2. From equation (4.20) it can be shown that

$$g_H^{\alpha}(u') = \frac{1+e^{j\frac{\alpha\pi}{2}}}{2} g_F^{\alpha}(u') + \frac{1-e^{j\frac{\alpha\pi}{2}}}{2} g_F^{\alpha}(-u')$$

(4.23)

Above equation is the relation between fractional Hartley transform and the fractional Fourier transform. When $\alpha = 1$, the above equation becomes the relation between Hartley and Fourier transform [18].

4.3 Two Dimensional Fractional Hartley Transform

Two dimensional fractional Hartley transform of function $f(x,u)$ is given by the following equation:

$$g_H^{\alpha}(x',u') = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} K_H^{\alpha}(x,x',u,u')\, f(x,u)\, dx\, du \qquad (4.24)$$

where

$$K_H^{\alpha}(x,x',u,u') = \sqrt{\frac{1-j\cot\phi}{2\pi}} e^{j\frac{x^2+x'^2+u^2+u'^2}{2}\cot\phi} (\cos(xx'\csc\phi + uu'\csc\phi)) + e^{j(\phi-\frac{\pi}{2})} (\sin(xx'\csc\phi + uu'\csc\phi) \qquad (4.25)$$

Since the two dimensional FRHT can be computed by applying one dimensional transforms separately in both directions x and u, so two

dimensional FRHT is separable in two dimensions and can be for rows and columns of an image.

CHAPTER 5
IMAGE COMPRESSION USING FRACTIONAL HARTLEY TRANSFORM

5.1 Introduction

In image processing an important part is the compression. This means the reducing the dimensions of the images, to a level that can be easily used or processed. Image compression using transform coding yields extremely good compression, with controllable degradation of image quality. In the present implementation fractional Hartley transform, generalization of Hartley Transform has been chosen. One of the reasons for this is, the FRHT provides additional degree of freedom to the problem as parameter 'a' gives multidirectional application. With the extra degree of freedom provided by the FRHT, its fractional order 'a' high visual quality decompressed image can be achieved for same amount of compression as that of Hartley transform.

5.2 FRHT Compression Model

In image compression using FRHT, a compression model encoder performs three relatively straightforward operations i.e. Subimage decompression, Transformation and Quantization. The decoder implements the inverse sequence of steps of encoder. Because quantization results in irreversible

information loss, so inverse quantization block is not included in the decoder as shown in figure 5.1. Hence it is lossy compression technique[3, 15].

5.2.1 Subimage Decomposition

An image is first divided into non overlapped as shown in figure5.1 (a). The most popular subimage sizes are 8*8, 16*16. As subimage size increases, error decreases but computational complexity increases. Compression that operate on block of pixels i.e. subimages are often described as block coding techniques. The transform of a block of pixels may suffer from discontinuity effects resulting in the presence of blocking artifacts in the image after it has been decompressed.

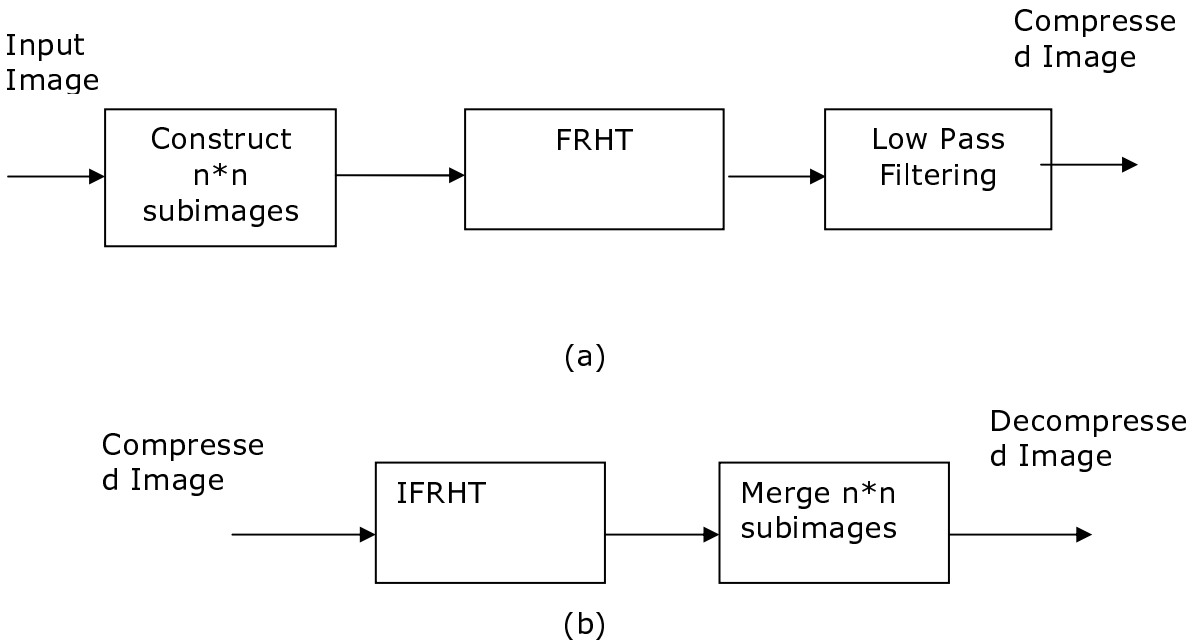


Figure5.1: FRHT image compression model, (a): Encoder, (b): Decoder

5.2.2 Transformation

In this step a 2-D FRHT is applied to each block to convert the gray levels of pixels in the spatial domain into coefficients in the frequency domain. By using FRHT a large amount of information is pack into smallest no. of transform coefficients, hence small amount of compression is achieved at

this step. At decoder inverse FRHT is applied. By changing the value of parameter 'a' to '-a' we get inverse FRHT.

The 2-D DHT for image is given by

$$g_H(x', u') = \frac{1}{2\pi} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} pixel \quad (x, u)\, cas \quad (xx')\, cas \quad (uu') \qquad (5.1)$$

The 2-D DFRHT for image is given by

$$g_H^{\alpha}(x', u') = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} K_H^{\alpha}(x.x', u, u')\, pixel\ (x, u)$$

(5.2)

where

$$K_H^{\alpha}(x, x', u, u') = \sqrt{\frac{1 - j\cot\phi}{2\pi}} e^{j\frac{x^2 + x'^2 + u^2 + u'^2}{2}\cot\phi} (\cos(xx'\csc\phi + uu'\csc\phi)) +$$

$$e^{j(\phi - \frac{\pi}{2})} (\sin(xx'\csc\phi + uu'\csc\phi)$$

(5.3)

5.2.3 Low Pass Filtering

The final step in compression process is low pass filtering. In this the higher frequency components of the transform coefficients are removed. The principle behind this process is based upon the information theory of viewing information as uncertainty that is human's eyes are less sensitive to the higher frequencies. In this the different masks, called zonal mask and multiplied to the sub image for making the higher frequency coefficients to zero [2].

5.3 Image Compression Characteristics

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

to the various applications. The following paragraphs discuss each of these attributes in more detail [11].

5.3.1 Compression Ratio

The compression ratio is equal to the size of the original image divided by the size of the compressed 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. Generally, the 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 a 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.

5.3.2 Compression Speed

Compression time and decompression time are defined as the amount of 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

Generally, the faster that both operations can be performed, the better. Fast compression time increases the speed with which material can be created. Fast

decompression time increases the speed with which the user can display and interact with images.

5.3.3 Image Quality

Image quality describes the fidelity with which an image-compression scheme recreates the source image data. Compression schemes 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 schemes try to compress the data as much as possible, without decreasing the image quality in a noticeable way. Some schemes may be either lossy or lossless, depending upon the quality level desired by the user [3, 11].

CHAPTER 6

RESULTS AND DISCUSSION

6.1 Introduction

The quality of compressed image can be measured by many parameters, which compare to the different compression techniques. The most commonly used parameters are Root Mean Square error (RMSE) and Peak Signal to Noise Ratio (PSNR). The PSNR value used to measure the difference between a decoded image $\hat{f}$ and its original image f is defined as follows. In general, the larger PSNR value, the better will be the decoded image quality.

$$RMSE = \left[\frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \left[\hat{f}(i, j) - f(i, j) \right]^2 \right]^{1/2}$$

$$PSNR = 10 \log_{10} \left[\frac{M \times N}{RMSE^2} \right]$$

where M*N is the size of the images, $\hat{f}(i,j)$ and $f(i,j)$ are the matrix elements of the decompressed and the original images at (i,j) pixel. In order to evaluate the performance of image compression systems, compression ratio matrix is often employed. In our results, compression ratio (CR) is computed as the ratio of nonzero entries in the original image to the non zero entries in the decompressed image.

CR=original image size/compressed image size

CR%=(1-(1/CR))*100

Image compression experiments using fractional Hartley transform are conducted on many images. The first image is the standard Lena image, whose face has graced the pages of many image processing books, papers and projects. The second and third images are camera man and flower in water used frequently in the image processing literature.

6.2 Lena Image

6.2.1 Effect of variation of Compression Ratio on Root Mean Square Error

Figure 6.1 (a) shows the original image. Figure 6.1 (b), (d), (f) shows the decompressed Lena image for different compression ratio for a=1. Figure 6.1(c), (e), (f) shows the compressed image for figure6.1 (b), (d), (f) respectively. As compression ratio is increased Root Mean Square Error is also increased. At the highest CR%, RMSE is highest and at lowest CR%, RMSE is highest. Figure 6.2 to figure 6.6 shows the different compressed and decompressed Lena image for a=0.97, 0.946, 0.90, 0.50, 0.1 respectively.

6.2.2 Effect of variation of Compression Ratio on Peak Signal to Noise Ratio

Figure 6.1 (b), (d), (f) shows the decompressed Lena image for different compression ratio for a=1. Figure 6.1(c), (e), (f) shows the compressed image for figure6.1 (b), (d), (f) respectively. As compression ratio is

increased Root Mean Square Error is also increased. At the highest CR%, PSNR is lowest and at lowest CR%, PSNR is highest. Figure 6.2 to figure 6.6 shows the different compressed and decompressed Lena image for a=0.97, 0.946, 0.90, 0.50, 0.1 respectively.

6.2.3 Effect of variation of 'a' on Root Mean Square Error

Figure 6.1 to 6.6 shows the different compressed and decompressed Lena images for different CR%. From the figures it is clear best that for a=0.946 lowest RMSE is achieved for various values of 'a'. At a=0.1, highest RMSE i.e. poor quality of image is obtained.

6.2.4 Effect of variation of 'a' on Peak signal to Noise Ratio

Figure 6.1 to 6.6 shows the different compressed and decompressed Lena images for different CR%. From the figures it is clear best that for a=0.946 highest PSNR is achieved for various values of 'a'. At a=0.1, lowest PSNR i.e. poor quality of image is obtained.

Fig6.1(a) Original Lena Image

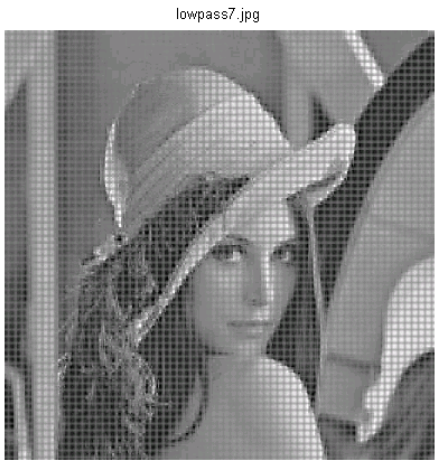
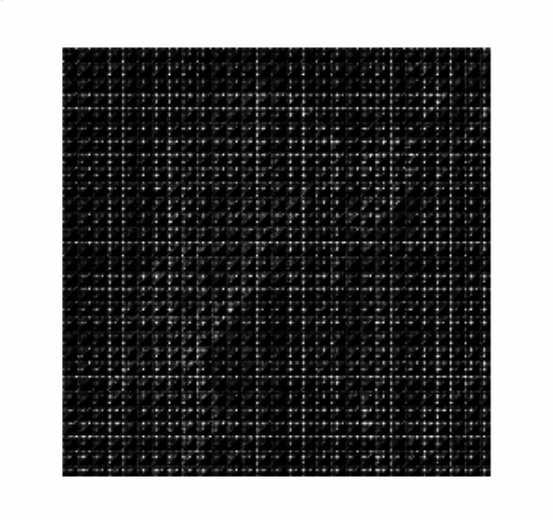


Fig6.1(b)Decompressed Lena image (a=1, CR%=67.1875, RMSE=34.5193)



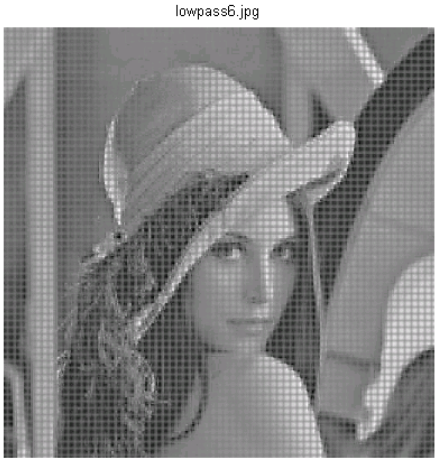
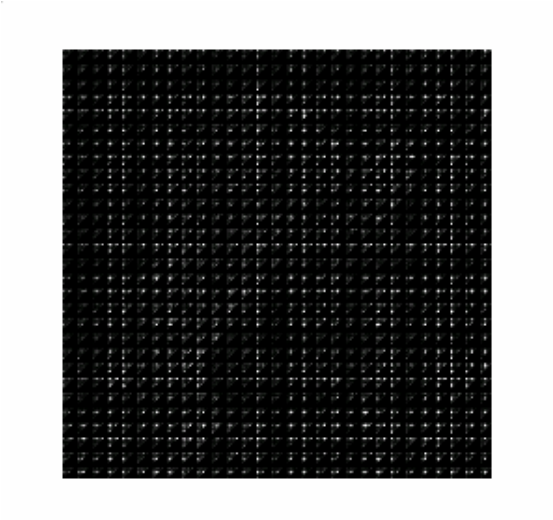


Fig6.1(c) Compressed Lena image (a=1, CR%=67.1875, RMSE=34.5193)

Fig6.1(d)Decompressed Lena image (a=1, CR%=76.5625, RMSE=37.9134)



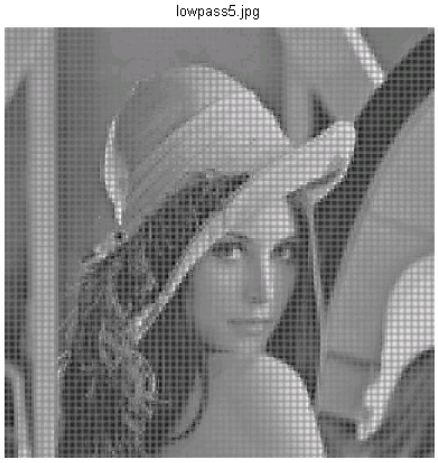


Fig6.1(e)Compressed Lena image (a=1, CR%=76.5625,RMSE=37.9134)

Fig6.1(f) Decompressed Lena image (a=1, CR%=84.3750,RMSE=39.5597)

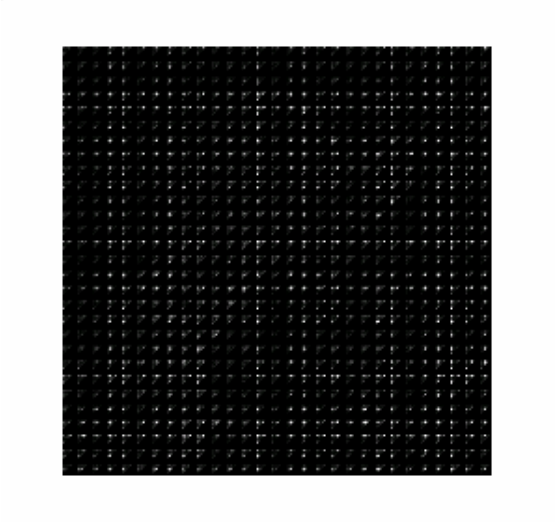


Fig6.1 (g) Compressed Lena image
*(a=1, CR%=*84.3750*, RMSE=*39.5597*)*

Fig6.1: Simulation results for Lena image (a=1)

Table6.1: Values of RMSE, PSNR for different CR% for Lena image (a=1)

Compression Ratio (%)	RMSE	PSNR
56.2500	4.7122	34.6663
67.1875	34.5193	17.3696
76.5625	37.9134	16.5549
84.3750	39.5597	16.1857
90.6250	40.4524	15.9919
95.3125	40.9451	15.8868
98.4375	41.2092	15.8309

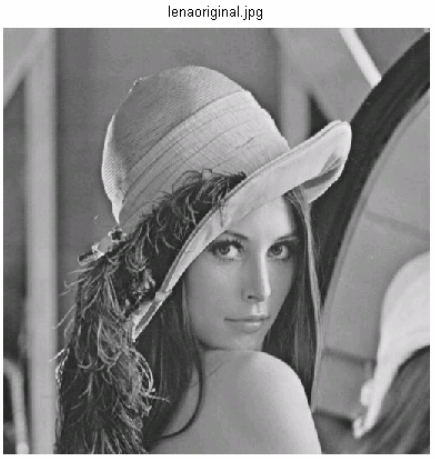


Fig6.2(a) Original Lena Image

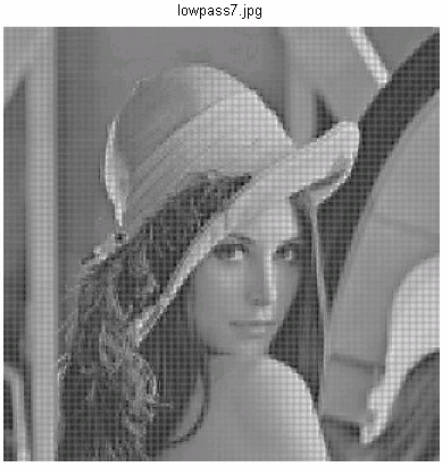
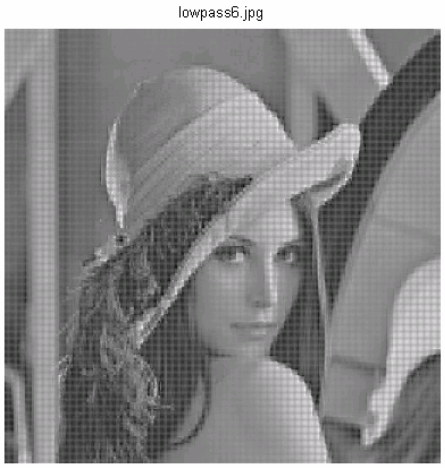


Fig6.2(b) Decompressed Lena image

(a=.97, CR%=67.1875*, RMSE*=30.5212)



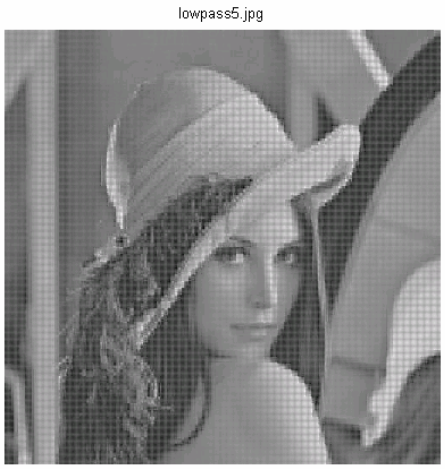


Fig6.2(c) Decompressed Lena image (a=.97,CR%=76.5625,RMSE=32.5556)

Fig6.2(d)Decompressed Lena image (a=.97,CR%=84.3750,RMSE=33.5855)

Fig6.2: Simulation results for Lena image (a=0.97)

Table6.2: Values of RMSE, PSNR for different CR% for Lena image (a=0.97)

Compression Ratio (%)	RMSE	PSNR
56.2500	4.6432	34.7945
67.1875	30.5212	18.4388
76.5625	32.5556	17.8783
84.3750	33.5855	17.6078
90.6250	34.0518	17.4880
95.3125	34.2743	17.4314
98.4375	34.3965	17.4005

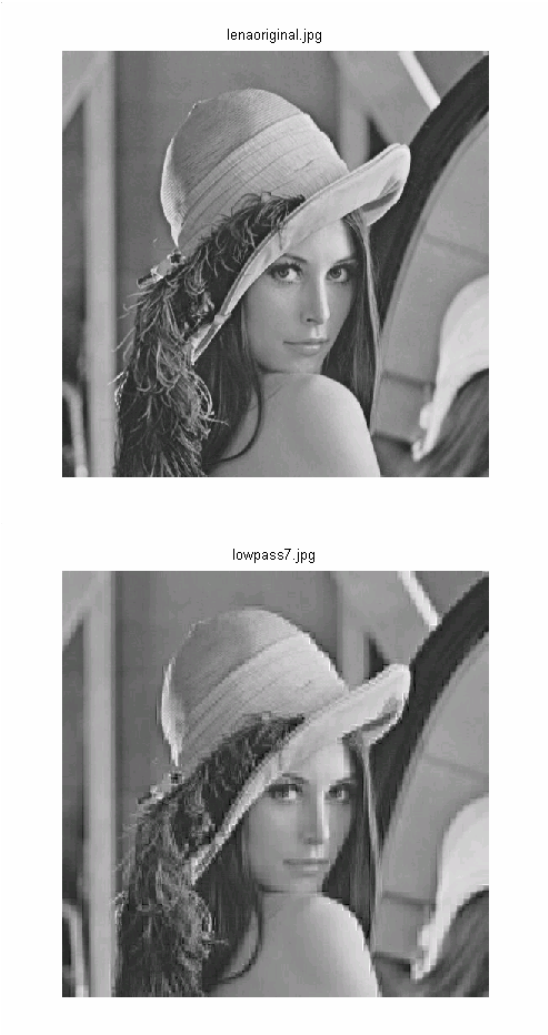


Fig6.3(a)Original Lena Image *Fig6.3(b)Decompressed Lena image*

(a=.946,CR%=67.1875,RMSE=24.8719)

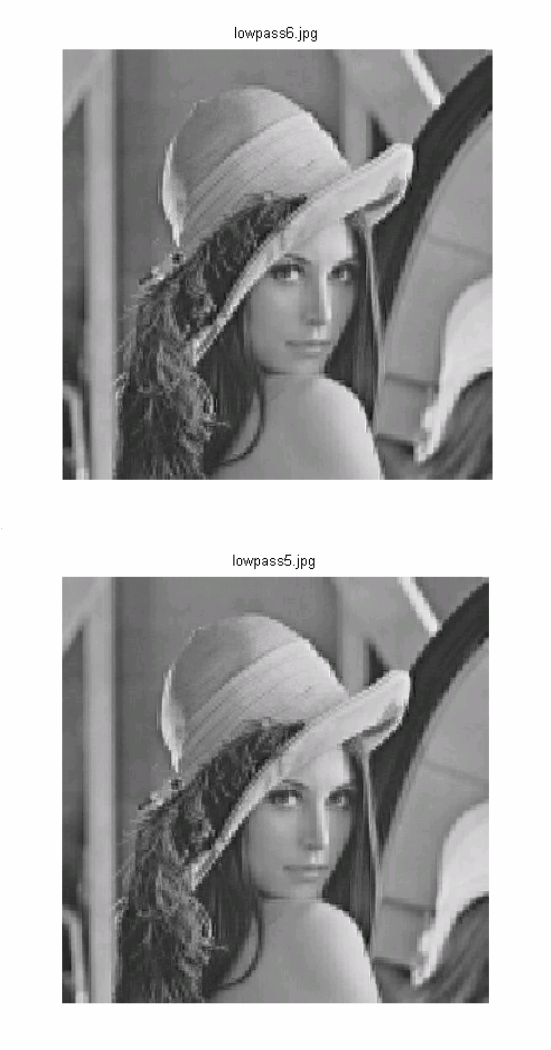


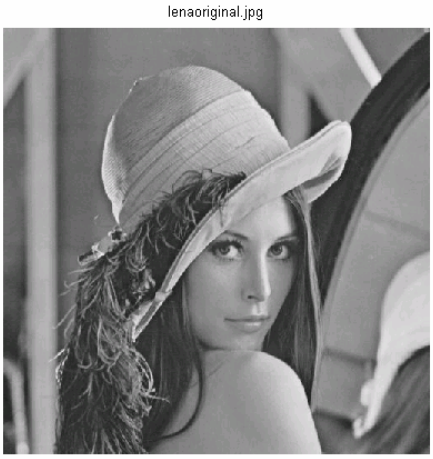
*Fig6.3(c)Decompressed Lena image (a=.946,CR%=*76.5625,*RMSE*=26.0112)

*Fig6.3(d)Decompressed Lena image (a=.946, CR%=*84.3750,*RMSE*=26.6809*)*

Fig6.3: Simulation results for Lena image (a=0.946)

Table6.3: Values of RMSE, PSNR for different CR% for Lena image (a=0.946)

Compression Ratio (%)	RMSE	PSNR
56.2500	4.5961	34.8830
67.1875	24.8719	20.2166
76.5625	26.0112	19.8276
84.3750	26.6809	19.6068
90.6250	26.9449	19.5213
95.3125	27.0498	19.4875
98.4375	27.0973	19.4723



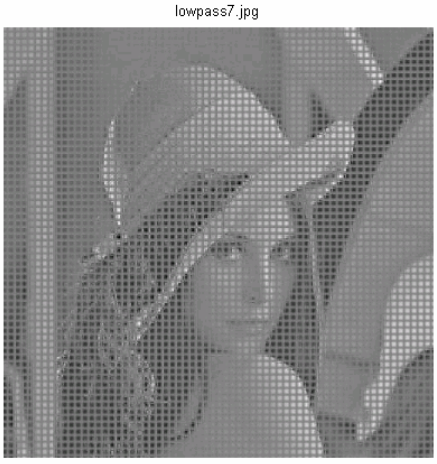


Fig6.4(a)Original Lena Image

Fig6.4(b)Decompressed Lena image

(a=.9, *CR%*=67.1875,*RMSE*=33.2266)

Fig6.4(c)Decompressed Lena image (a=.9, CR%=76.5625, RMSE=33.2510)

Fig6.4(d)Decompressed Lena image (a=.9, CR%=84.3750,RMSE=33.2890)

Fig6.4: Simulation results for Lena image (a=0.90)

Table6.4: Values of RMSE, PSNR for different CR% for Lena image (a=0.90)

Compression Ratio (%)	RMSE	PSNR
56.2500	4.7135	34.6639
67.1875	33.2266	17.7011
76.5625	33.2510	17.6947
84.3750	33.2890	17.6848
90.6250	33.4219	17.6502
95.3125	33.7218	17.5726
98.4375	34.8880	17.2773

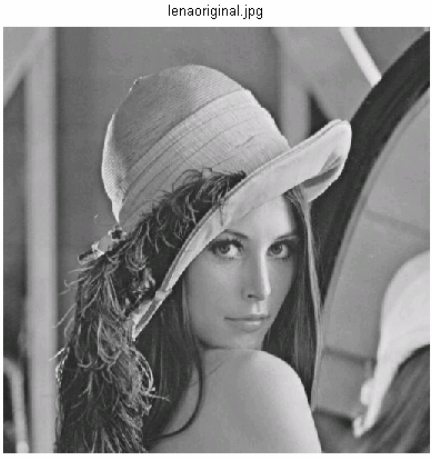


Fig6.5(a)Original Lena Image

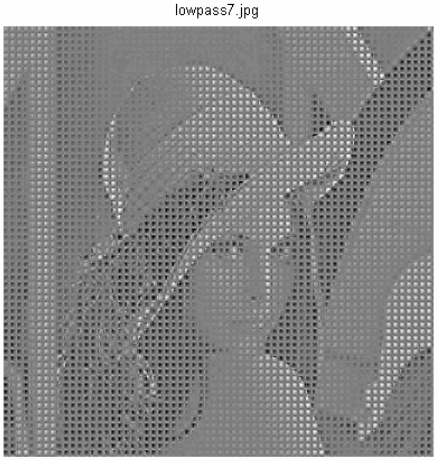


Fig6.5(b)Decompressed Lena image

(a=.5, CR%=67.1875,*RMSE*=41.0184)

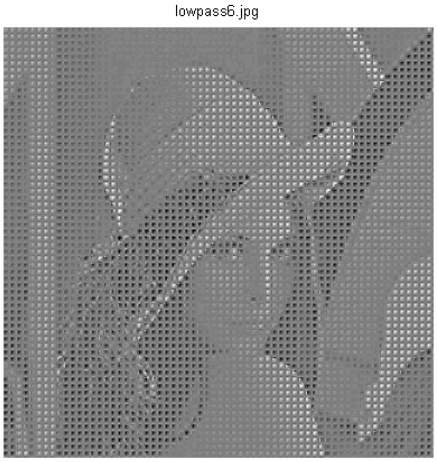


Fig6.5(c)Decompressed Lena image

(a=.5, CR%=76.5625, *MSE*=41.3164)

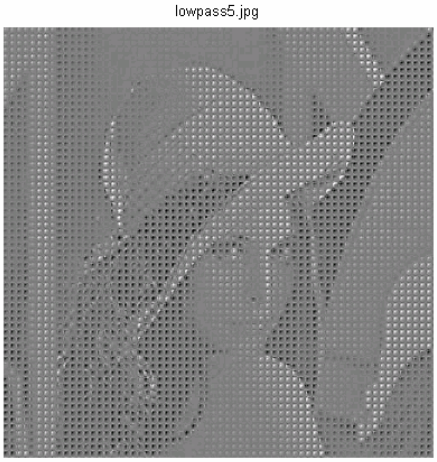


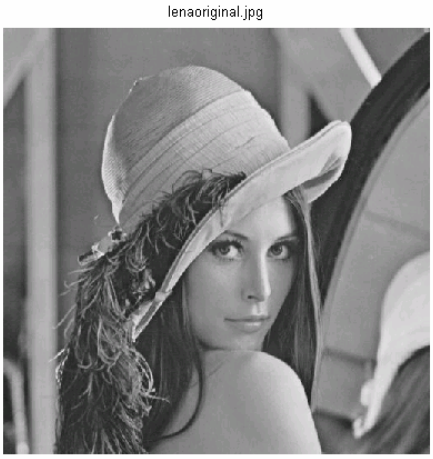
Fig6.5(d)Decompressed Lena image

(a=.5, CR%=84.3750, *MSE*=41.4904)

Fig6.5: Simulation results for Lena image (a=0.5)

Table6.5: Values of RMSE, PSNR for different CR% for Lena image (a=0.5)

Compression Ratio (%)	RMSE	PSNR
56.2500	4.7534	34.5907
67.1875	41.0184	15.8712
76.5625	41.3164	15.8084
84.3750	41.4904	15.7719
90.6250	41.9008	15.6864
95.3125	43.8804	15.2854
98.4375	45.1236	15.0427



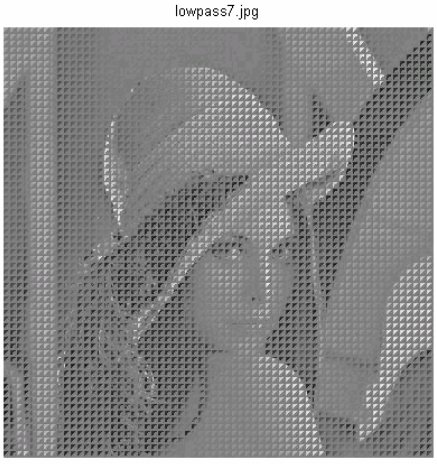
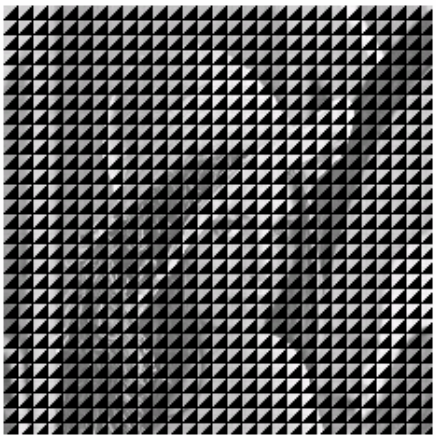


Fig6.6(a) Original Lena Image

Fig6.6(b) Decompressed Lena image

*(a=.1, CR%=*67.1875*, RMSE=*37.3692)



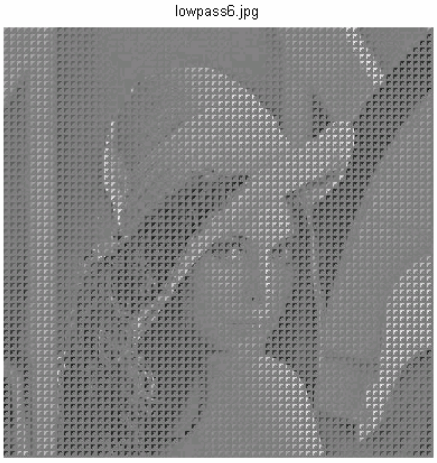


*Fig6.6(c) Compressed Lena image (a=.1, CR%=*67.1875*, RMSE=*37.3692)

*Fig6.6(d)Decompressed Lena image (a=.1, CR%=*76.5625*, RMSE=*39.7337)

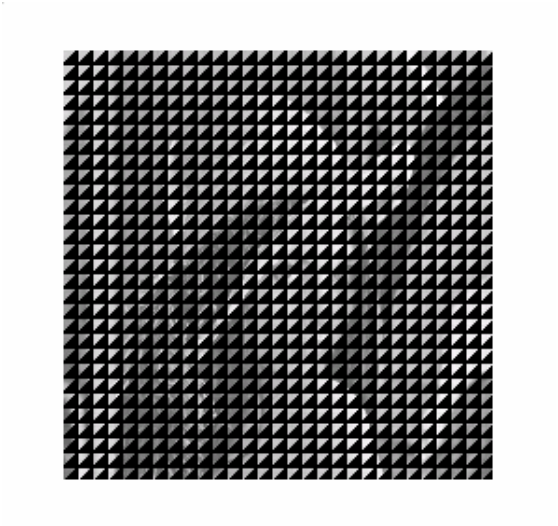


Fig6.6(e)Compressed Lena image (a=.1, CR%=76.5625, RMSE=39.7337)

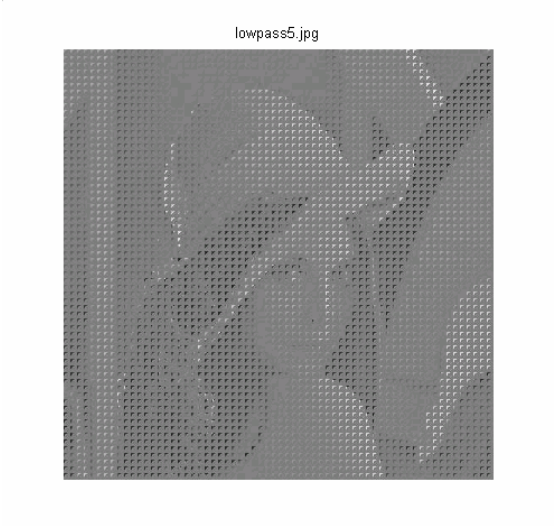


Fig6.6(f)Decompressed Lena image (a=.1, CR%=84.3750, RMSE=41.6262)

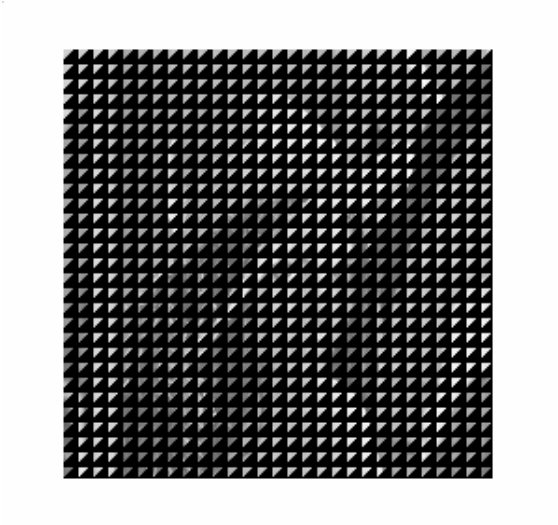


Fig6.6(g) Compressed Lena image (a=.1, CR%=84.3750, RMSE=41.6262)

Fig6.6: Simulation results for Lena image (a=0.1)

Table6.6: Values of RMSE, PSNR for different CR% for Lena image (a=0.1)

Compression Ratio (%)	RMSE	PSNR
56.2500	4.8578	34.4020
67.1875	37.3692	16.6805
76.5625	39.7337	16.1476
84.3750	41.6262	15.7435
90.6250	43.2166	15.4178
95.3125	44.2886	15.2050
98.4375	44.9284	15.0804

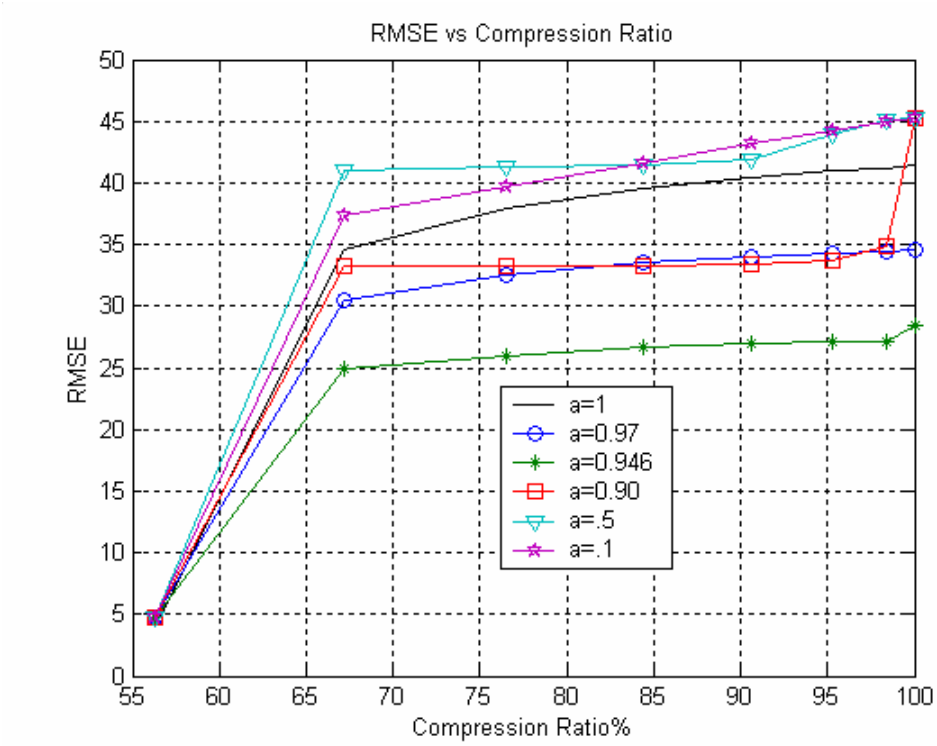


Figure6.7: RMSE versus CR% for different values of 'a' for Lena image

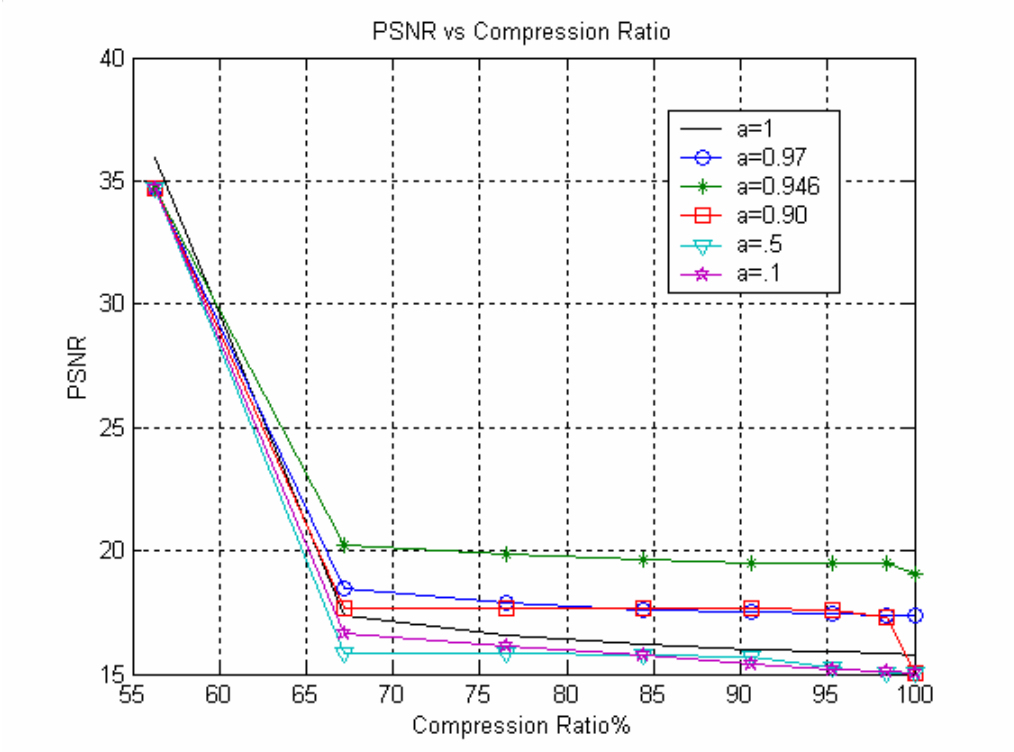


Figure6.8: PSNR versus CR% for different values of 'a' for Lena image

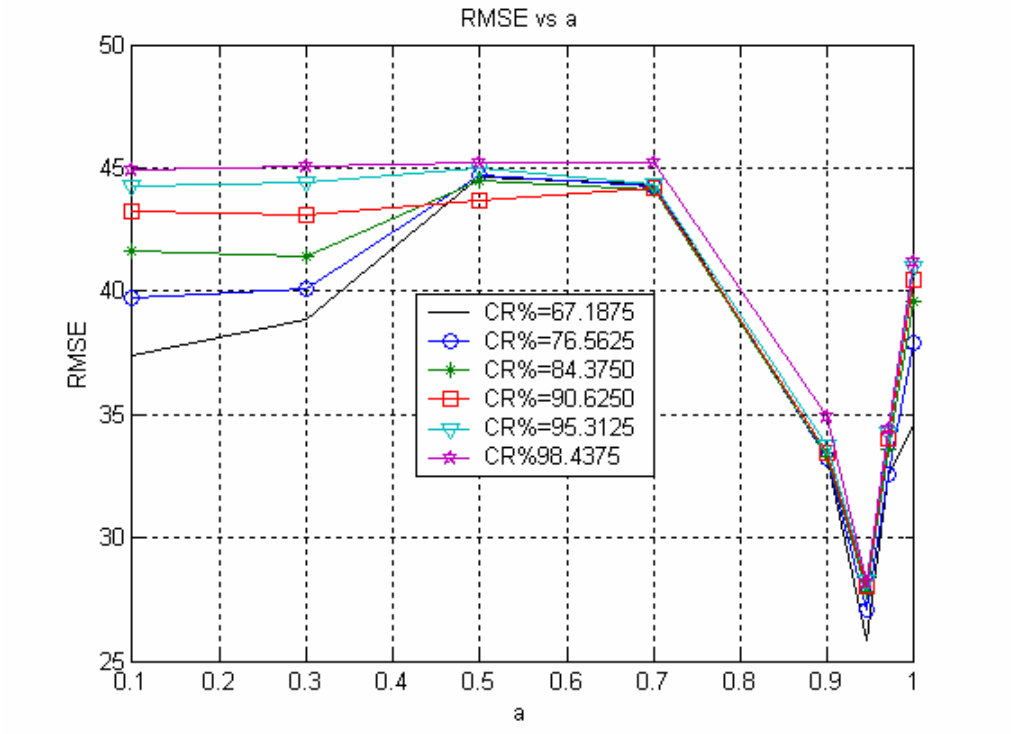


Figure6.9: RMSE versus 'a' for different values of CR% for Lena image

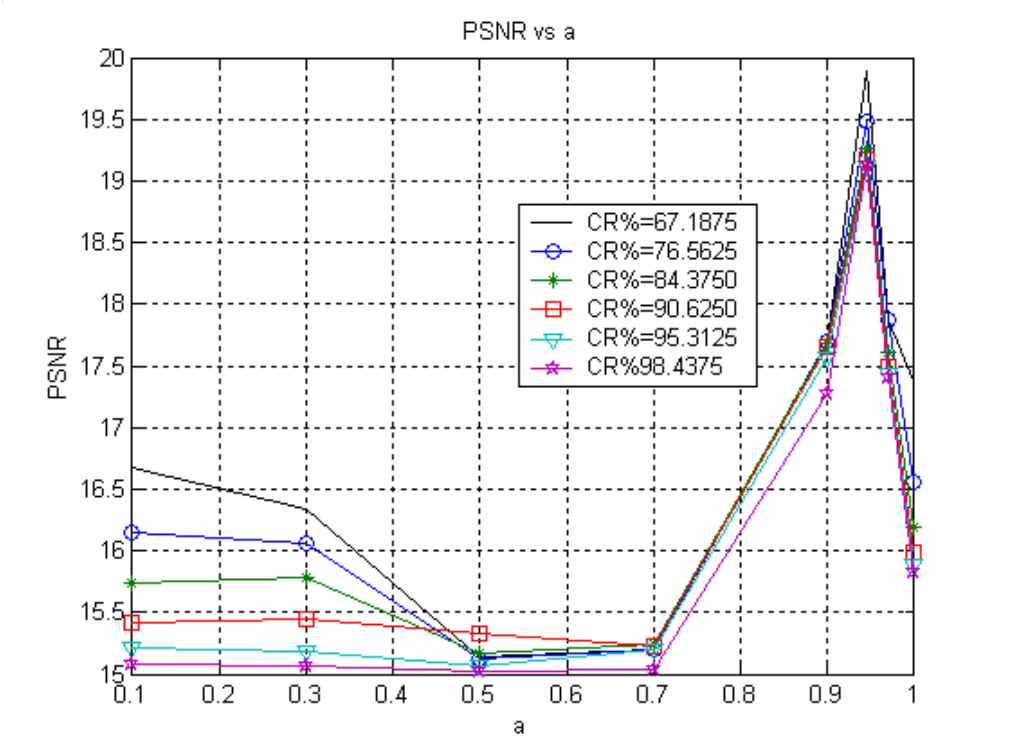


Figure6.10: PSNR versus 'a' for different values of CR% for Lena image

6.3 Cameraman Image

6.3.1 Effect of variation of Compression Ratio on Root Mean Square Error

Figure 6.11 (a) shows the original image. Figure 6.11 (b), (d), (f) shows the decompressed Cameraman image for different compression ratio for a=1. Figure 6.11(c), (e), (f) shows the compressed image for figure6.11 (b), (d), (f) respectively. As compression ratio is increased Root Mean Square Error is also increased. At the highest CR%, RMSE is highest and at lowest CR%, RMSE is highest. Figure 6.12 to figure 6.16 shows the different compressed and decompressed Cameraman image for a=0.97, 0.946, 0.90, 0.50, 0.1 respectively.

6.3.2 Effect of variation of Compression Ratio on Peak Signal to Noise Ratio

Figure 6.11 (b), (d), (f) shows the decompressed Cameraman image for different compression ratio for a=1. Figure 6.11(c), (e), (f) shows the

compressed image for figure6.11 (b), (d), (f) respectively. As compression ratio (CR) is increased Peak Signal to Noise Ratio is decreased. At the highest CR%, PSNR is lowest and at lowest CR%, PSNR is highest. Figure 6.2 to figure 6.6 shows the different compressed and decompressed Cameraman image for a=0.97, 0.946, 0.90, 0.50, 0.1 respectively.

6.3.3 Effect of variation of 'a' on Root Mean Square Error

Figure 6.11 to 6.16 shows the different compressed and decompressed Cameraman images for different CR%. From the figures it is clear best that for a=0.946 lowest RMSE is achieved for various values of 'a'. At a=0.1, highest RMSE i.e. poor quality of image is obtained.

6.3.4 Effect of variation of 'a' on Peak signal to Noise Ratio

Figure 6.11 to 6.16 shows the different compressed and decompressed Cameraman images for different CR%. From the figures it is clear best that for a=0.946 highest PSNR is achieved for various values of 'a'. At a=0.1, lowest PSNR i.e. poor quality of image is obtained.

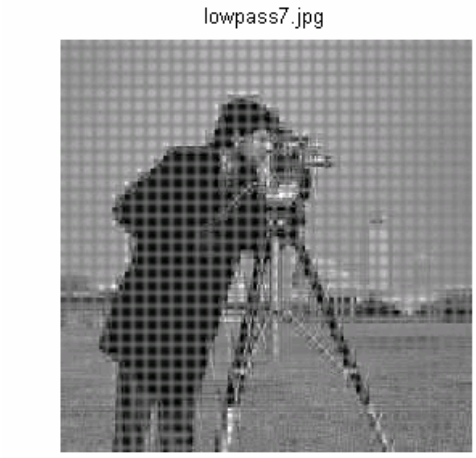


Fig6.11(a) Original Cameraman Image *Fig6.11(b)Decompressed Cameraman image*

(a=1, CR%=67.1875, *RMSE*=22.9160*)*

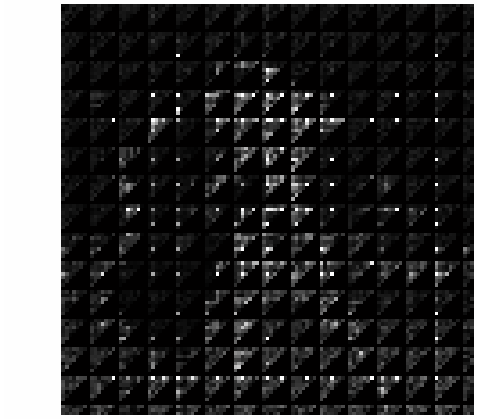


Fig6.11(c) Compressed image Fig6.11(d)Decompressed image

*(a=1, CR%=*67.1875*, RMSE=*22.9160*)* *(a=1, CR%=*76.5625*, RMSE=*27.4753*)*

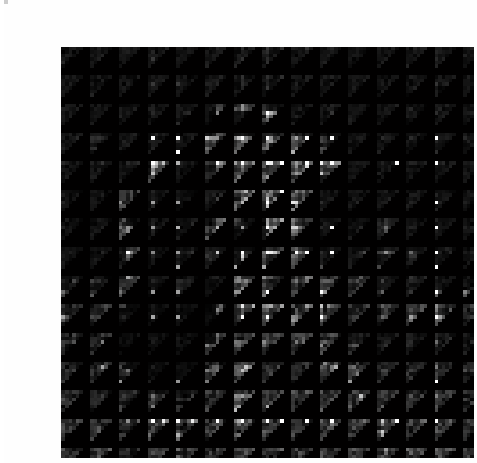


Fig6.11(e)Compressed image *Fig6.11(f) Decompressed image*

*(a=1, CR%=*76.5625*,RMSE=*27.4753*)* *(a=1, CR%=*84.3750*,RMSE=*29.8466*)*

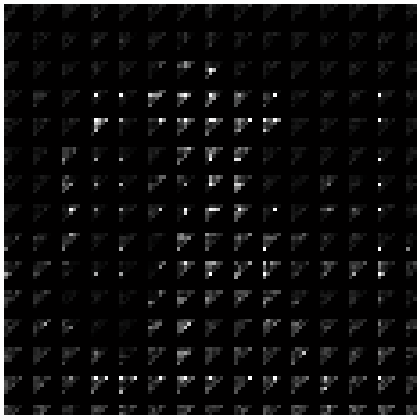


Fig6.11(g) Compressed image
*(a=1, CR%=*84.3750*, RMSE=*29.8466)

Fig6.11: Simulation results for Cameraman image (a=1)

Table6.7: Values of RMSE, PSNR for different CR% for Cameraman image (a=1)

Compression Ratio (%)	RMSE	PSNR
56.2500	13.0397	25.8255
67.1875	22.9160	20.9280
76.5625	27.4753	19.3520
84.3750	29.8466	18.6329
90.6250	31.2028	18.2469
95.3125	32.1849	17.9778
98.4375	32.8326	17.8047

Fig6.12(a) Original Cameraman Image *Fig6.12(b) Decompressed Cameraman image*

(a=.97,CR%=67.1875,*RMSE*=18.6169)

Fig6.12(c)Decompressed image
Fig6.12(d)Decompressed image

*(a=.97,CR%=*76.5625*,RMSE=*21.8548*)*
*(a=.97,CR%=*84.3750*,RMSE=*23.5954*)*

Fig6.12: Simulation results for Cameraman image (a=0.97)

Table6.8: Values of RMSE, PSNR for different CR% for Cameraman image (a=0.97)

Compression Ratio (%)	RMSE	PSNR
56.2500	13.0303	25.8317
67.1875	18.6169	22.7327
76.5625	21.8548	21.3399
84.3750	23.5954	20.6743
90.6250	24.5387	20.3338
95.3125	25.0845	20.1427
98.4375	25.4201	20.0273

Fig6.13(a)Original Cameraman Image *Fig6.13(b)Decompressed Cameraman image*

(a=.946,CR%=67.1875,RMSE=14.1097)

Fig6.13(c)Decompressed image (a=.946,CR%=76.5625,RMSE=16.2125)

Fig6.13(d)Decompressed image (a=.946, CR%=84.3750,RMSE=17.4355)

Fig6.13: Simulation results for Cameraman image (a=0.946)

Table6.9: Values of RMSE, PSNR for different CR% for Cameraman image (a=0.946)

Compression Ratio (%)	RMSE	PSNR
56.2500	13.0275	25.8336
67.1875	14.1097	25.1404
76.5625	16.2125	23.9338
84.3750	17.4355	23.3021
90.6250	18.0714	22.9910
95.3125	18.3670	22.8500
98.4375	18.5153	22.7802

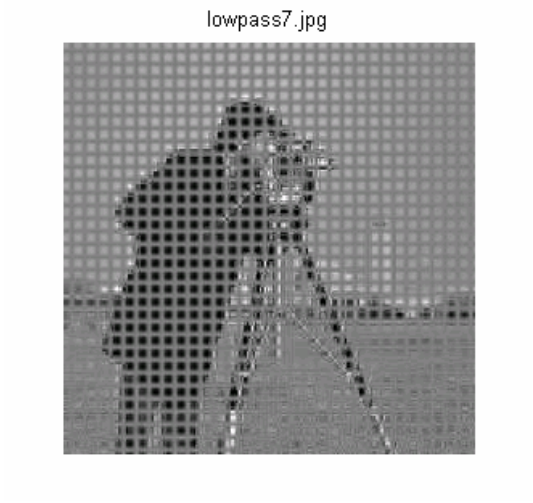


Fig6.14(a)Original Cameraman Image *Fig6.14(b)Decompressed Cameraman image*

(a=.9, CR%=67.1875,*RMSE*=46.3478)

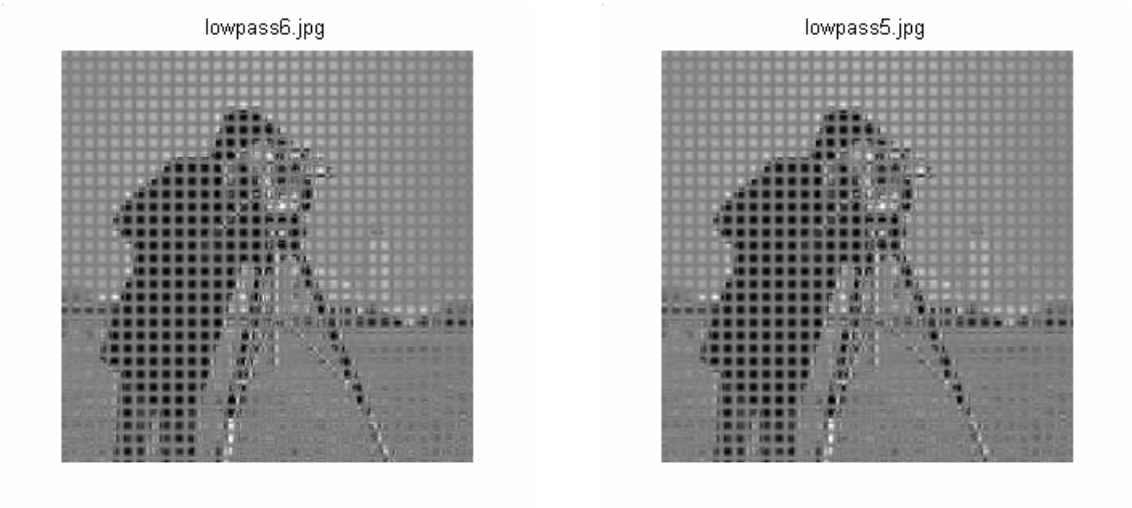


Fig6.14(c)Decompressed image (a=.9, CR%=76.5625, RMSE=46.4480)

Fig6.14(d)Decompressed image (a=.9, CR%=84.3750,RMSE=46.5947)

Fig6.14: Simulation results for Cameraman image (a=0.90)

Table6.10: Values of RMSE, PSNR for different CR% for Cameraman image (a=0.90)

Compression Ratio (%)	RMSE	PSNR
56.2500	13.0492	25.8191
67.1875	46.3478	14.8102
76.5625	46.4480	14.7915
84.3750	46.5947	14.7641
90.6250	46.9834	14.6919
95.3125	47.6116	14.5765
98.4375	49.5767	14.2253

Fig6.15(a)Original Cameraman Image

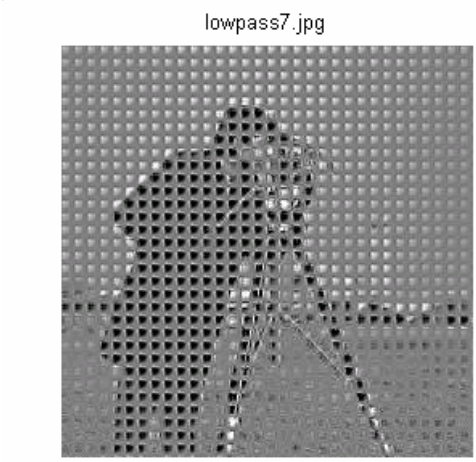


Fig6.15(b)Decompressed Cameraman image (a=.5, CR%=67.1875,*RMSE*=55.9851)

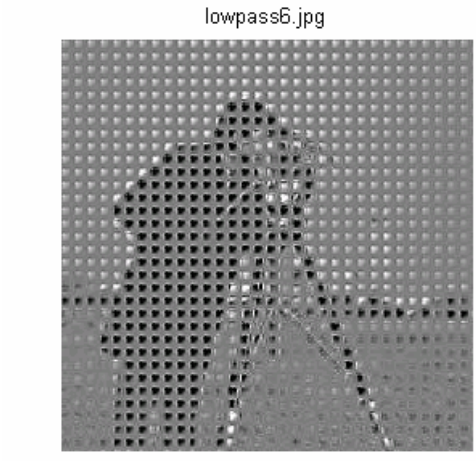


Fig6.15(c)Decompressed image (a=.5, CR%=76.5625, *RMSE*=56.4944)

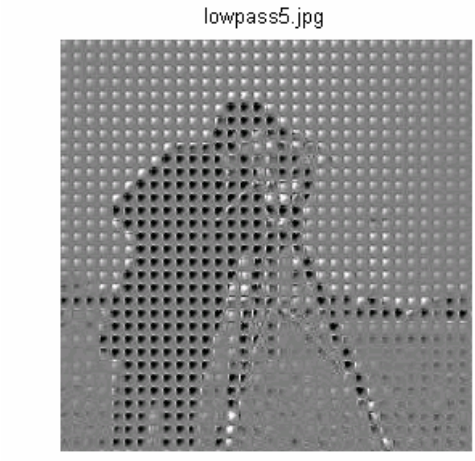


Fig6.15(d)Decompressed image (a=.5, CR%=84.3750, *RMSE*=56.9051)

Fig6.15: Simulation results for Cameraman image (a=0.5)

Table6.11: Values of RMSE, PSNR for different CR% for Cameraman image (a=0.5)

Compression Ratio (%)	RMSE	PSNR
56.2500	13.1074	25.7805
67.1875	55.9851	13.1694
76.5625	56.4944	13.0907
84.3750	56.9051	13.0278
90.6250	58.1128	12.8454
95.3125	60.9996	12.4243
98.4375	62.7128	12.1837

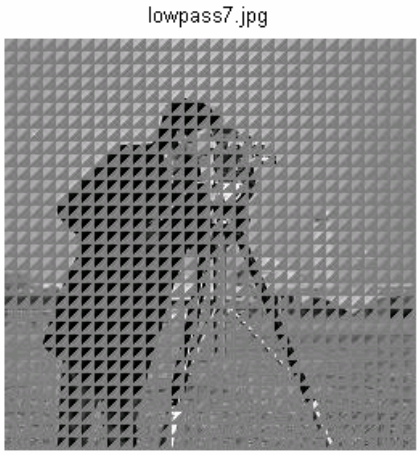
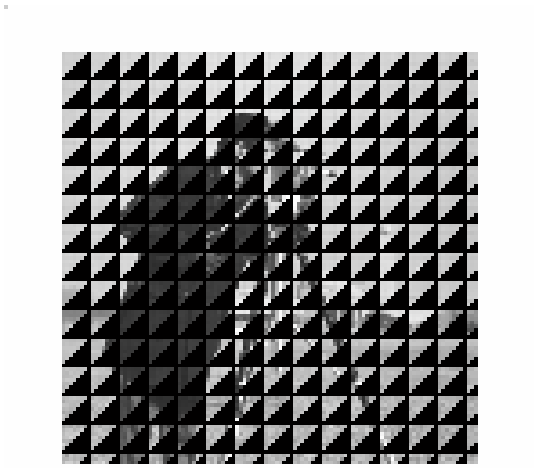


Fig6.16(a) Original Cameraman Image

Fig6.16(b) Decompressed Cameraman image

(a=.1, CR%=67.1875, *RMSE*=52.0143)



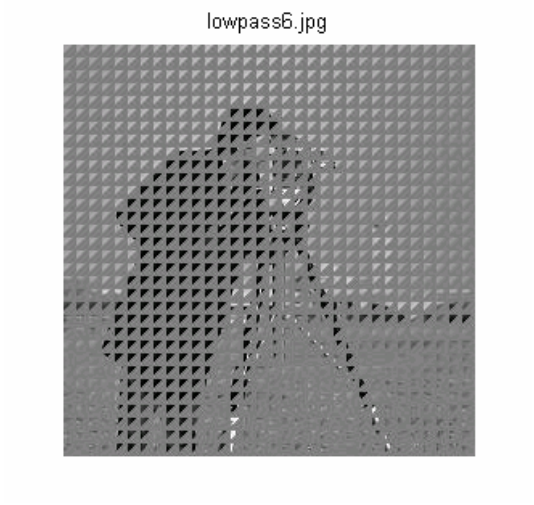
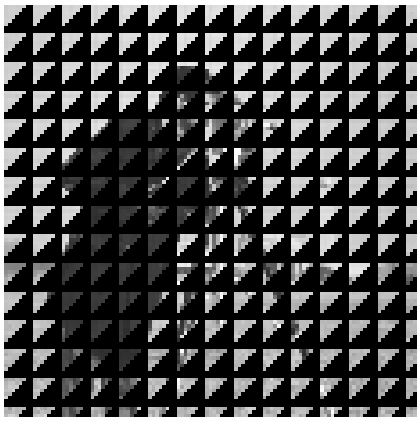


*Fig6.16(c)Compressed Cameraman image (a=.1, CR%=*67.1875*, RMSE=*52.0143*)*

*Fig6.16(d)Decompressed Cameraman image (a=.1, CR%=*76.5625*, RMSE=*55.3228*)*



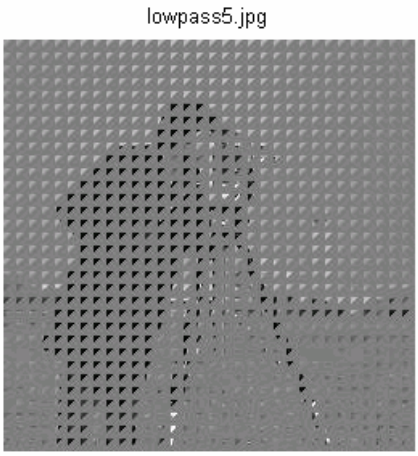


*Fig6.16(e)Compressed Cameraman image (a=.1, CR%=*76.5625*, RMSE=*55.3228*)*

*Fig6.16(f)Decompressed Cameraman image (a=.1, CR%=*84.3750*, RMSE=*57.9365*)*

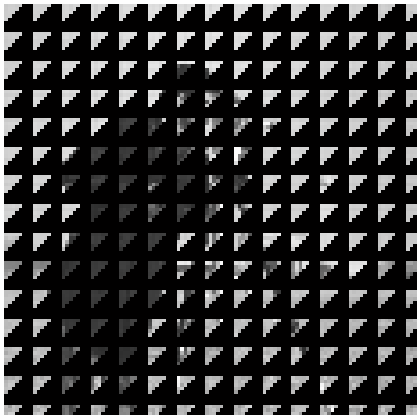


*Fig6.16(g) Decompressed Cameraman image (a=.1, CR%=*84.3750*, RMSE=*57.9365*)*

Fig6.16: Simulation results for Cameraman image (a=0.1)

Table6.12: Values of RMSE,PSNR for different CR% for Cameraman image(a=0.1)

Compression Ratio (%)	RMSE	PSNR
56.2500	13.1142	25.7760
67.1875	52.0143	13.8083
76.5625	55.3228	13.2727
84.3750	57.9365	12.8718
90.6250	60.0929	12.5543
95.3125	61.5915	12.3404
98.4375	62.4887	12.2148

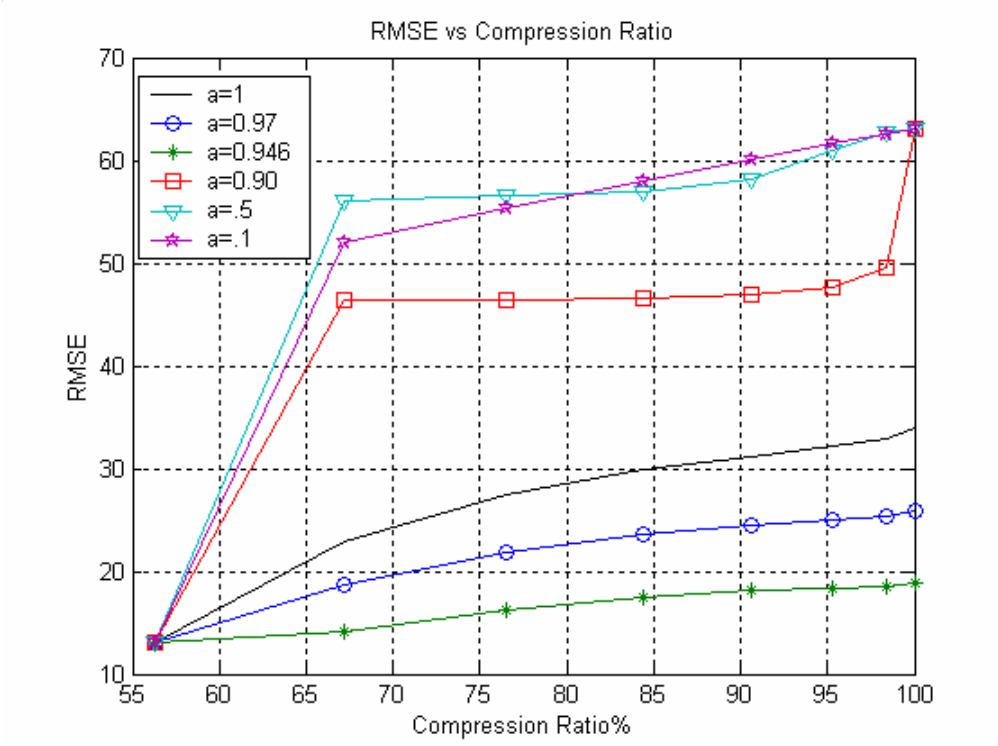


Figure6.17: RMSE versus CR% for different values of 'a' for Cameraman image

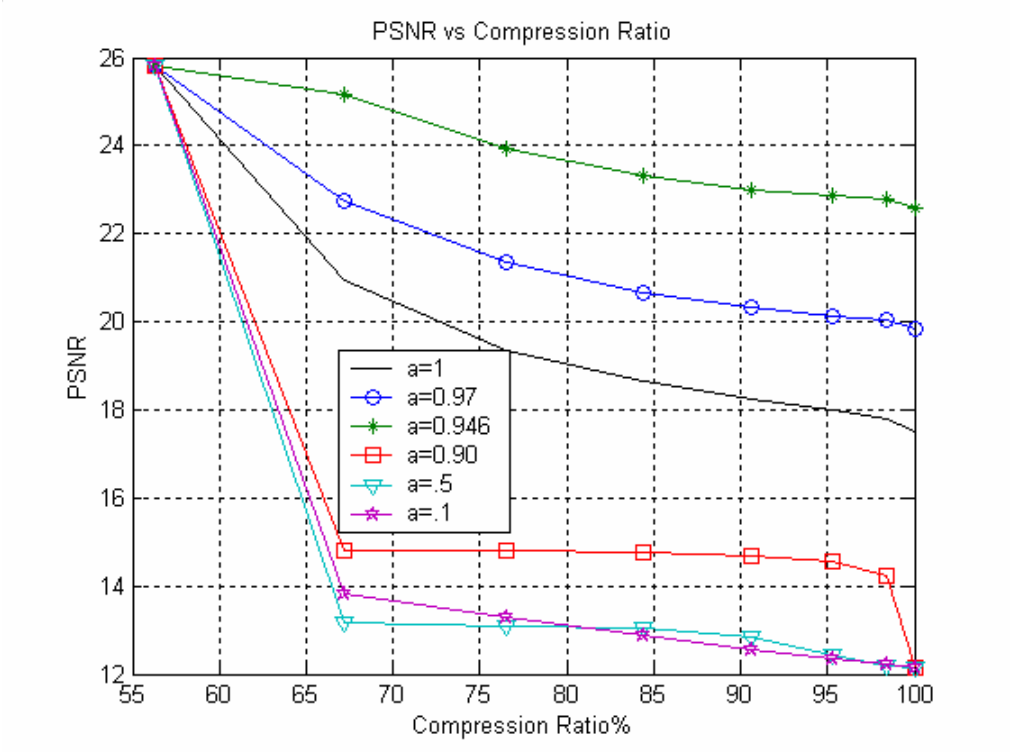


Figure6.18: PSNR versus CR% for different values of 'a' for Cameraman image

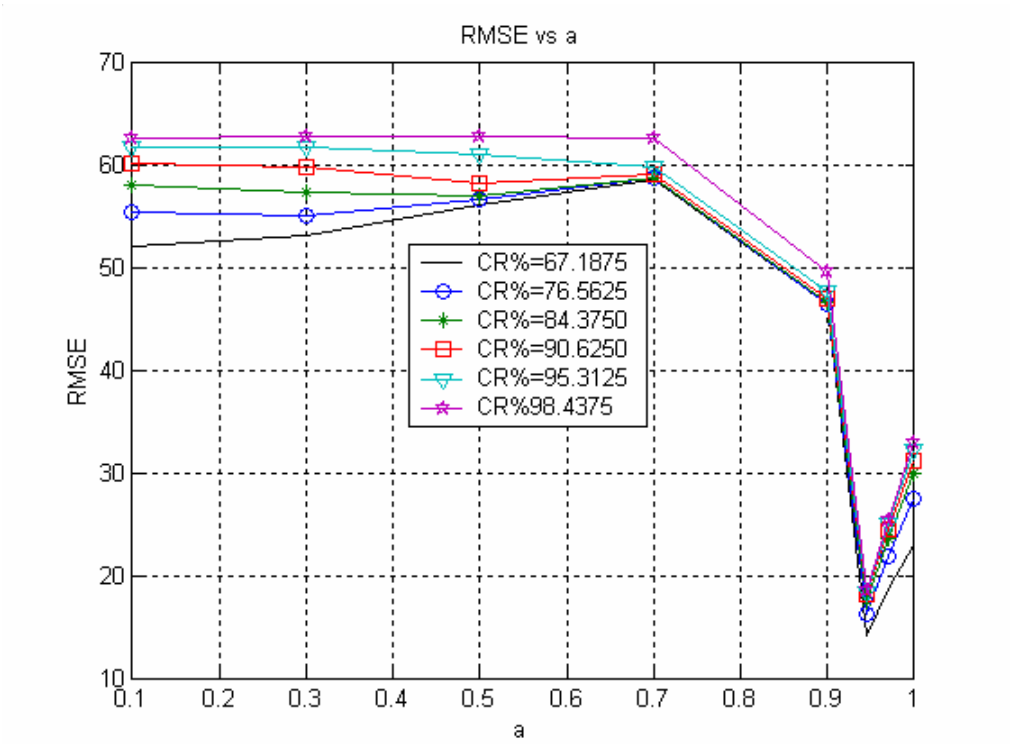


Figure6.19: RMSE versus 'a' for different values of CR% for Cameraman image

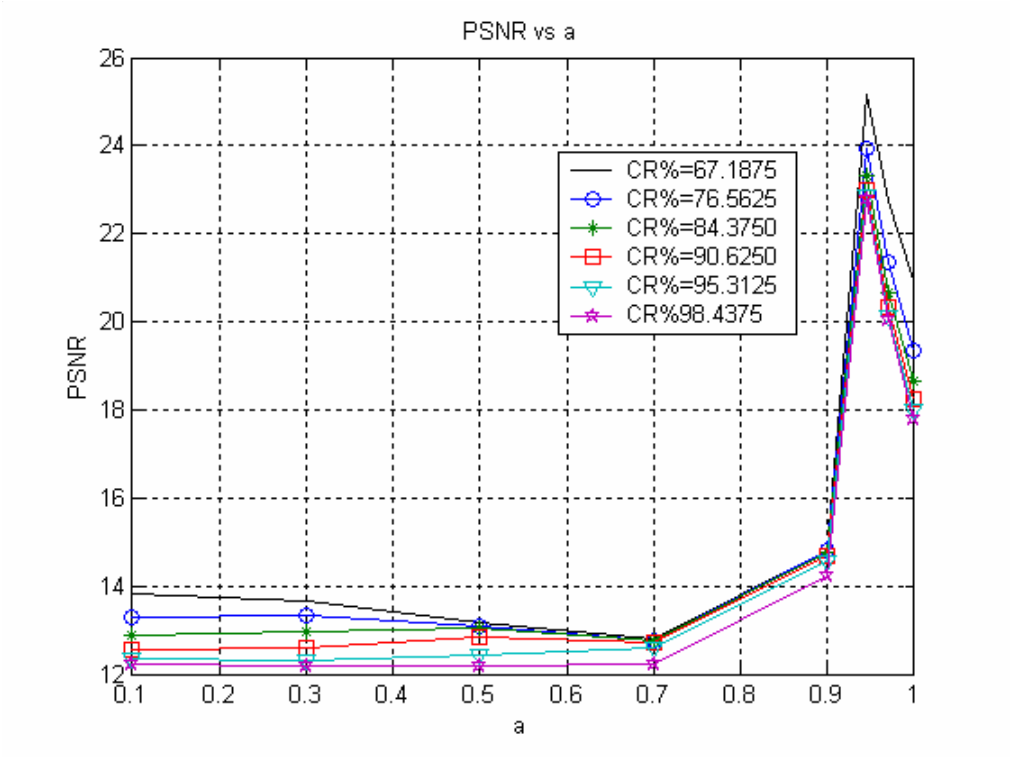


Figure6.20: PSNR versus 'a' for different values of CR% for Cameraman image

6.4 Flower image

6.4.1 Effect of variation of Compression Ratio on Root Mean Square Error

Figure 6.21 (a) shows the original flower image. Figure 6.21 (b), (d), (f) shows the decompressed Flower image for different compression ratio for a=1. Figure 6.21(c), (e), (f) shows the compressed image for figure6.21 (b), (d), (f) respectively. As compression ratio is increased Root Mean Square Error is also increased. At the highest CR%, RMSE is highest and at lowest CR%, RMSE is highest. Figure 6.22 to figure 6.26 shows the different compressed and decompressed Flower image for a=0.97, 0.946, 0.90, 0.50, 0.1 respectively.

6.4.2 Effect of variation of Compression Ratio on Peak Signal to Noise Ratio

Figure 6.21 (b), (d), (f) shows the decompressed Flower image for different compression ratio for a=1. Figure 6.21(c), (e), (f) shows the compressed image for figure6.21 (b), (d), (f) respectively. As compression ratio is increased PSNR is decreased. At the highest CR%, PSNR is lowest and at lowest CR%, PSNR is highest. Figure 6.22 to figure 6.26 shows the different compressed and decompressed Flower image for a=0.97, 0.946, 0.90, 0.50, 0.1 respectively.

6.4.3 Effect of variation of 'a' on Root Mean Square Error

Figure 6.21 to 6.26 shows the different compressed and decompressed Flower images for different CR%. From the figures it is clear best that for a=0.946 lowest RMSE is achieved for various values of 'a'. At a=0.1, highest RMSE i.e. poor quality of image is obtained.

6.4.4 Effect of variation of 'a' on Peak signal to Noise Ratio

Figure 6.21 to 6.26 shows the different compressed and decompressed Flower images for different CR%. From the figures it is clear best that for a=0.946 highest PSNR is achieved for various values of 'a'. At a=0.1, lowest PSNR i.e. poor quality of image is obtained.

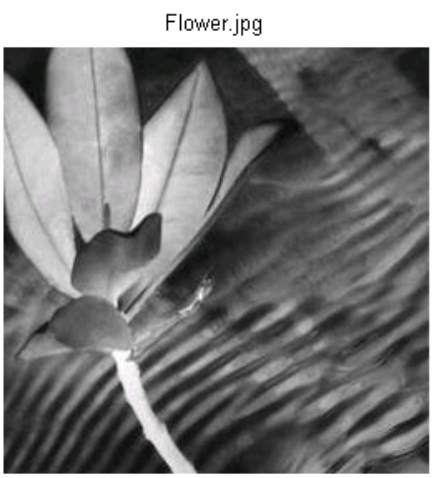


Fig6.21(a) Original Flower Image

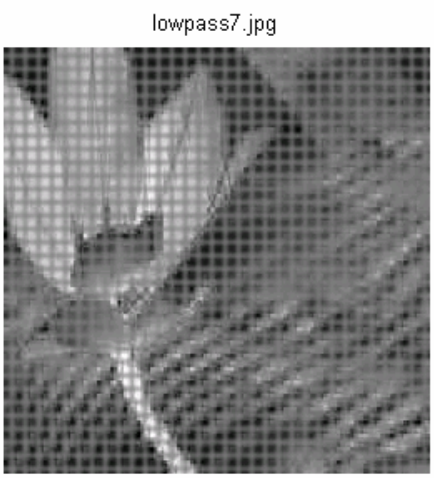


Fig6.21(b)Decompressed Flower image (a=1, CR%=67.1875, RMSE=27.2623)

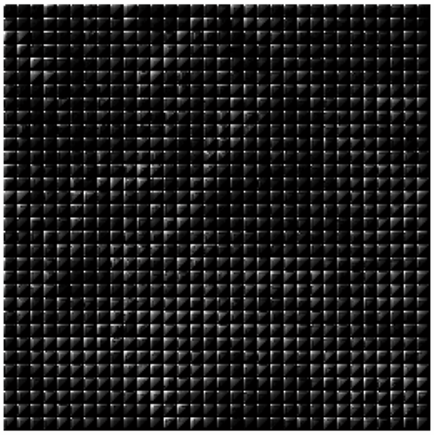


Fig6.21(c) Compressed Flower image (a=1, CR%=67.1875, RMSE=27.2623)

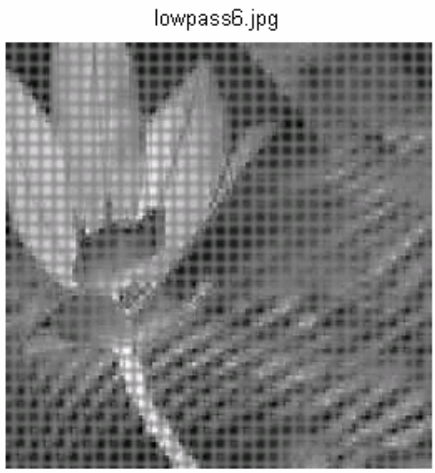
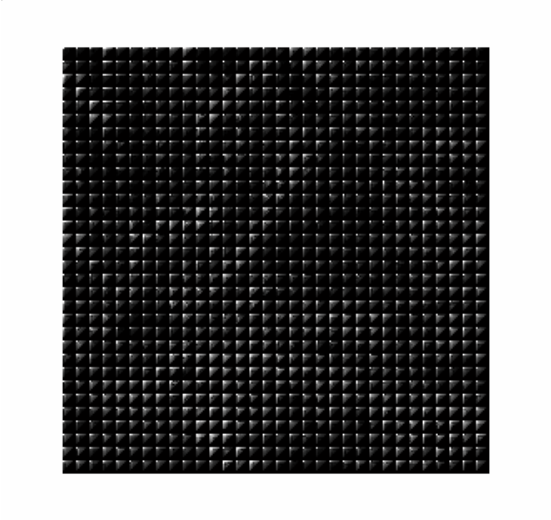


Fig6.21(d)Decompressed Flower image (a=1, CR%=76.5625, RMSE=32.6620)



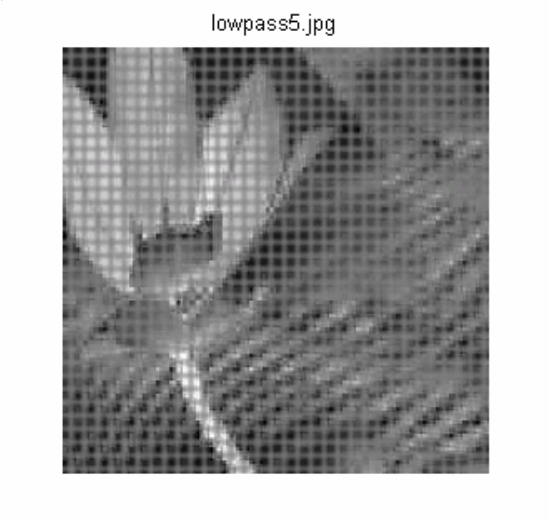


Fig6.21(e)Compressed Flower image (a=1, CR%=76.5625,RMSE=32.6620)

Fig6.21(f) Decompressed Flower image (a=1, CR%=84.3750,RMSE=35.2593)

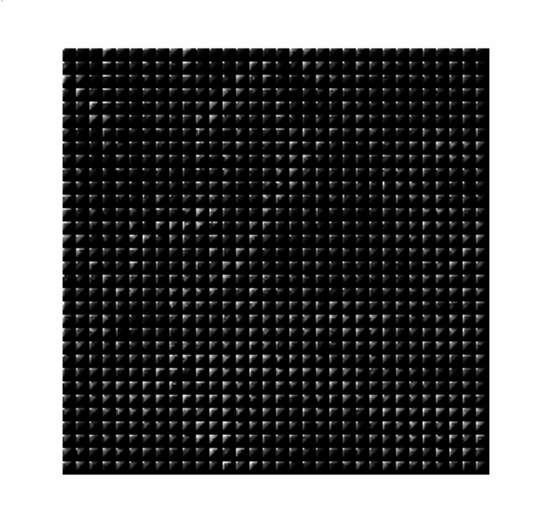


Fig6.21(g) Compressed Flower image (a=1, CR%=84.3750, RMSE=35.2593)

Fig6.1: Simulation results for Flower image (a=1)

Table6.13: Values of RMSE, PSNR for different CR% for Flower image (a=1)

Compression Ratio (%)	RMSE	PSNR
56.2500	7.3751	30.7754
67.1875	27.2623	19.4196
76.5625	32.6620	17.8499
84.3750	35.2593	17.1853
90.6250	36.7516	16.8253
95.3125	37.5513	16.6383
98.4375	37.9541	16.5456

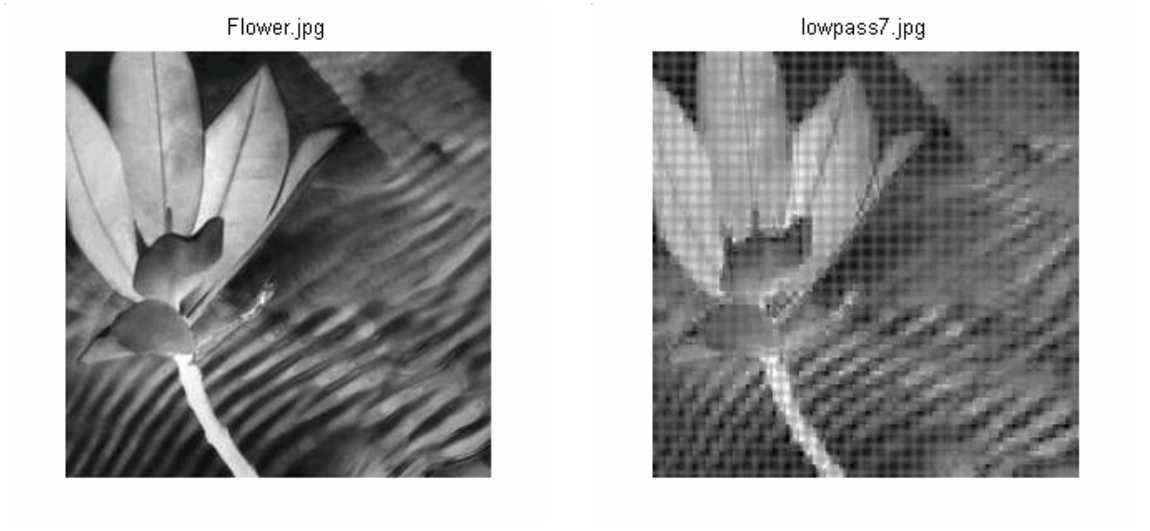


Fig6.22(a) Original Flower Image *Fig6.22(b) Decompressed Flower image*

(a=.97,CR%=67.1875,*RMSE*=23.0056)

*Fig6.22(c) Decompressed Flower image (a=.97,CR%=*76.5625,*RMSE=*26.7435*)*

*Fig6.22(d)Decompressed Flower image (a=.97,CR%=*84.3750,*RMSE=*28.5438*)*

Fig6.2: Simulation results for Flower image (a=0.97)

Table6.14: Values of RMSE, PSNR for different CR% for Flower image (a=0.97)

Compression Ratio (%)	RMSE	PSNR
56.2500	7.3700	30.7815
67.1875	23.0056	20.8941
76.5625	26.7435	19.5864
84.3750	28.5438	19.0206
90.6250	29.3947	18.7654
95.3125	29.7820	18.6517
98.4375	37.9541	18.5925

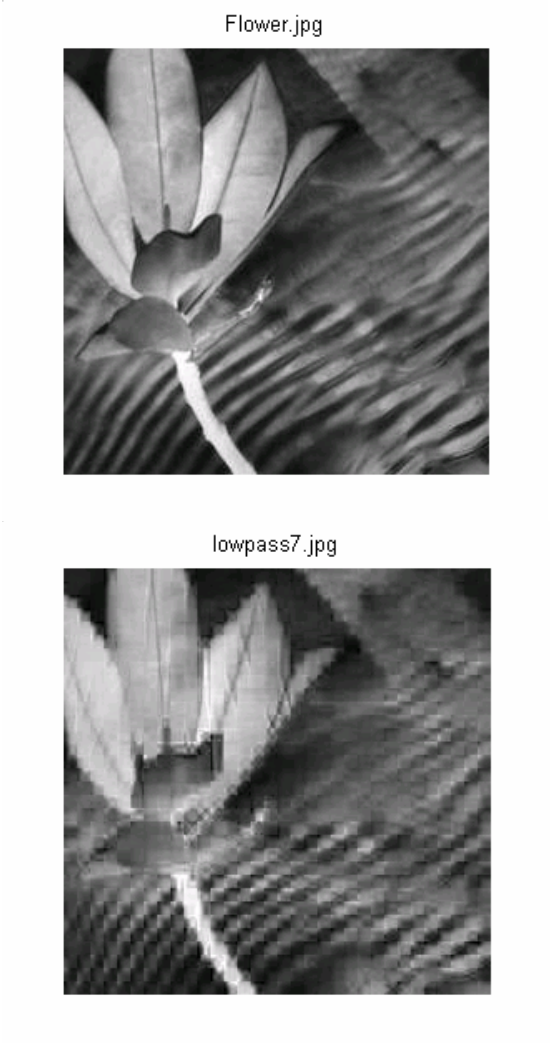


Fig6.23(a)Original Flower image *Fig6.23(b)Decompressed Flower image*

*(a=.946,CR%=67.1875,RMSE=*16.6448)

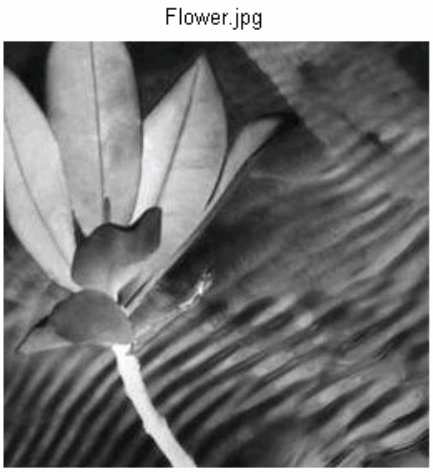
*Fig6.23(c)Decompressed Flower image (a=.946,CR%=*76.5625,*RMSE*=19.0061)

*Fig6.23(d)Decompressed Flower image (a=.946, CR%=*84.3750,*RMSE*=20.2380*)*

Fig6.23: Simulation results for Flower image (a=0.946)

Table6.23: Values of RMSE, PSNR for different CR% for Flower image (a=0.946)

Compression Ratio (%)	RMSE	PSNR
56.2500	7.3631	30.7896
67.1875	16.6448	23.7052
76.5625	19.0061	22.5529
84.3750	20.2380	22.0075
90.6250	20.7414	21.7940
95.3125	20.9295	21.7156
98.4375	21.0089	21.6827



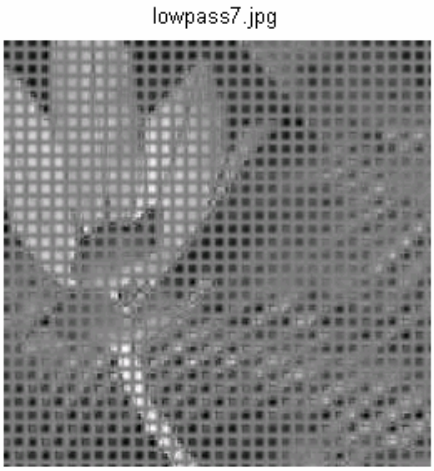


Fig6.24(a)Original Flower Image

Fig6.24(b)Decompressed Flower image

(a=.9, *CR%*=67.1875,*RMSE*=41.5683)

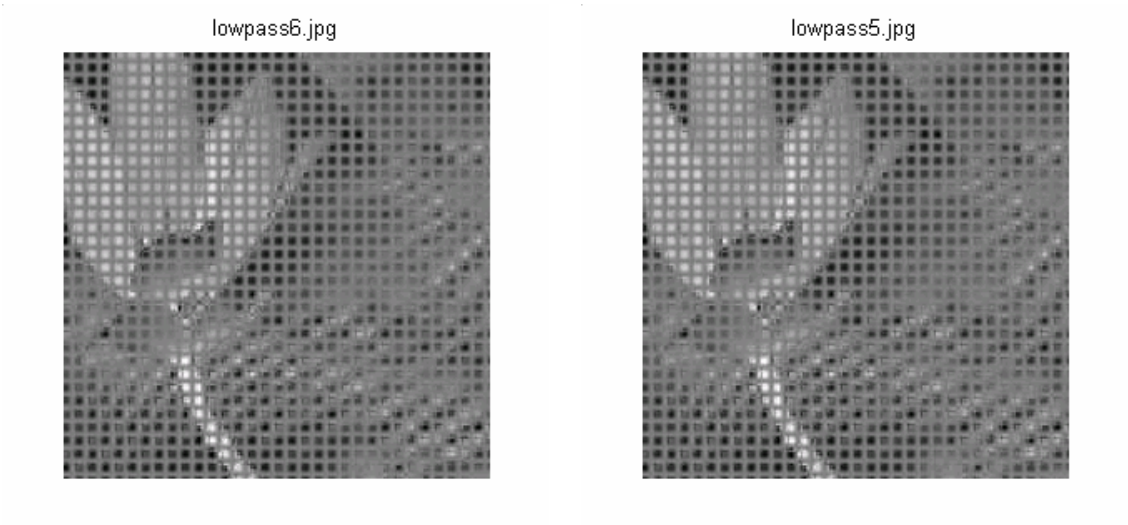


Fig6.24(c)Decompressed Flower image (a=.9, CR%=76.5625, RMSE=41.5968)

Fig6.24(d)Decompressed Flower image (a=.9, CR%=84.3750,RMSE=41.6649)

Fig6.24: Simulation results for Flower image (a=0.90)

Table6.16: Values of RMSE, PSNR for different CR% for Flower image (a=0.90)

Compression Ratio (%)	RMSE	PSNR
56.2500	7.3743	30.7764
67.1875	41.5683	15.7556
76.5625	41.5968	15.7496
84.3750	41.6649	15.7354
90.6250	41.9416	15.6779
95.3125	42.5896	15.5447
98.4375	44.5511	15.1536

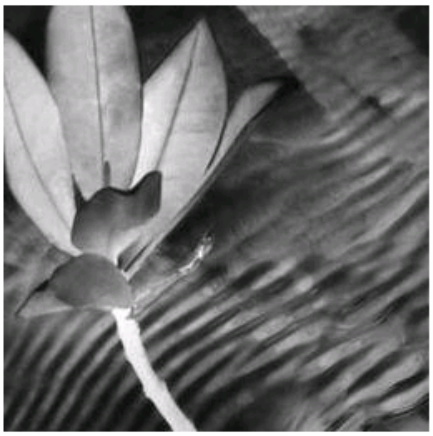


Fig6.25(a)Original Flower Image

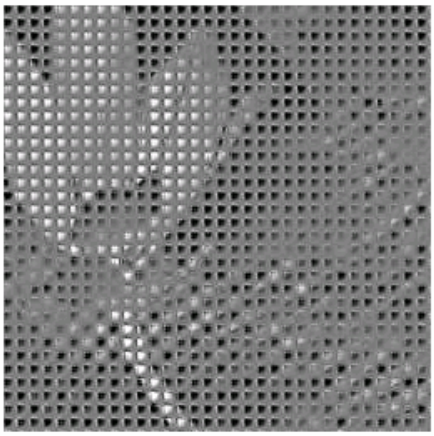


Fig6.25(b)Decompressed Flower image (a=.5, CR%=67.1875,*RMSE*=51.4248)

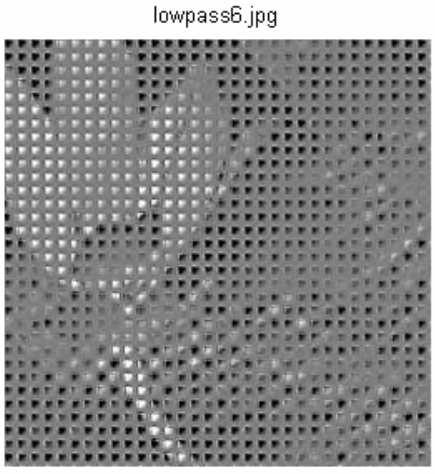


Fig6.25(c)Decompressed Flower image (a=.5, CR%=76.5625, *RMSE*=51.7793)

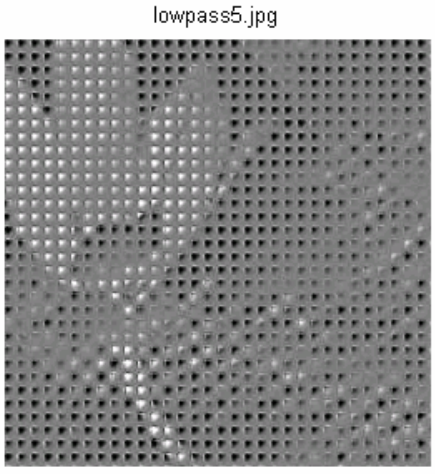
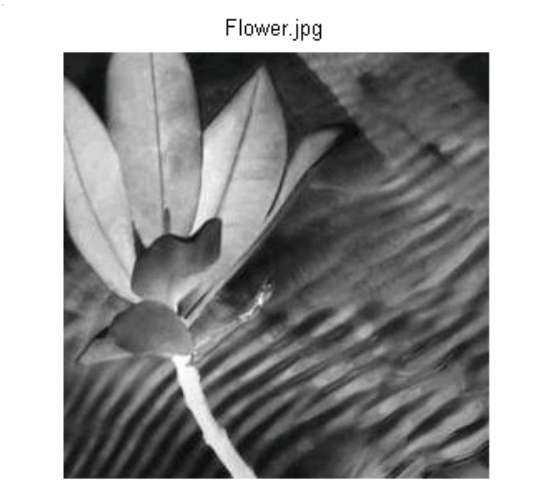


Fig6.25(d)Decompressed Flower image (a=.5, CR%=84.3750, *RMSE*=51.9984)

Fig6.25: Simulation results for Flower image (a=0.5)

Table6.17: Values of RMSE, PSNR for different CR% for Flower image (a=0.5)

Compression Ratio (%)	RMSE	PSNR
56.2500	7.3798	30.7699
67.1875	51.4248	13.9074
76.5625	51.7793	13.8477
84.3750	51.9984	13.8110
90.6250	52.7414	13.6878
95.3125	55.3644	13.2662
98.4375	56.9783	13.0166



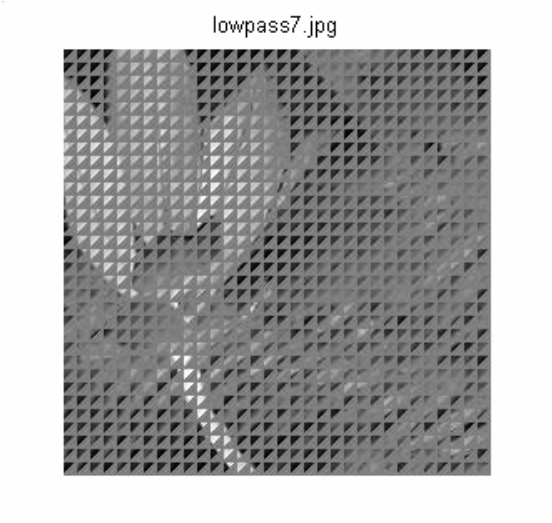
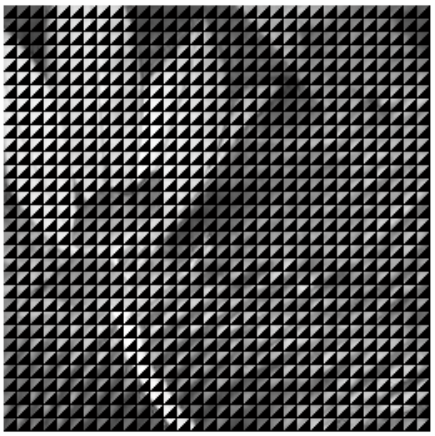


Fig6.26(a) Original Flower Image

*Fig6.26(b) Decompressed Flower image (a=.1, CR%=*67.1875*, RMSE=*47.2372)



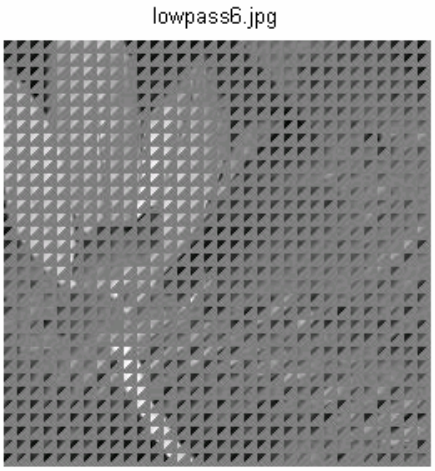
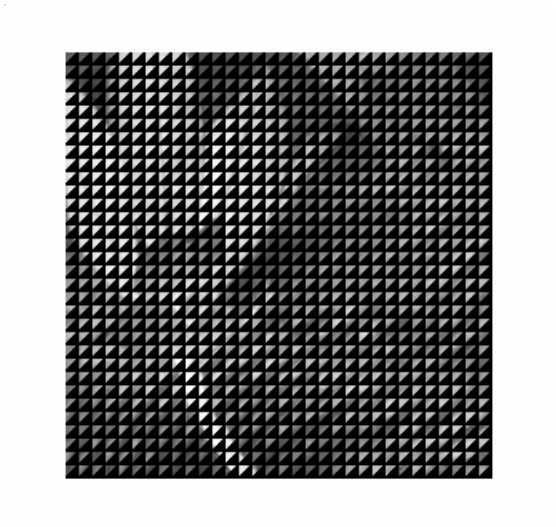


Fig6.26(c) Compressed Flower image (a=.1, CR%=67.1875, RMSE=47.2372)

Fig6.26(d)Decompressed Flower image (a=.1, CR%=76.5625, RMSE=50.2091)



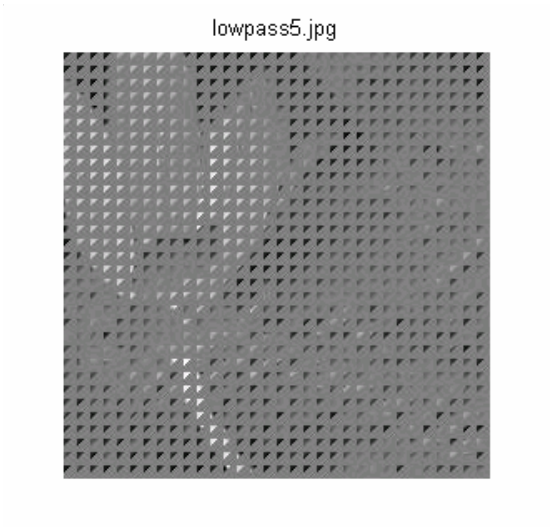


*Fig6.26(e)Compressed Flower image (a=.1, CR%=*76.5625*, RMSE=*50.2091*)*

*Fig6.26(f)Decompressed Flower image (a=.1, CR%=*84.3750*, RMSE=*52.5716*)*

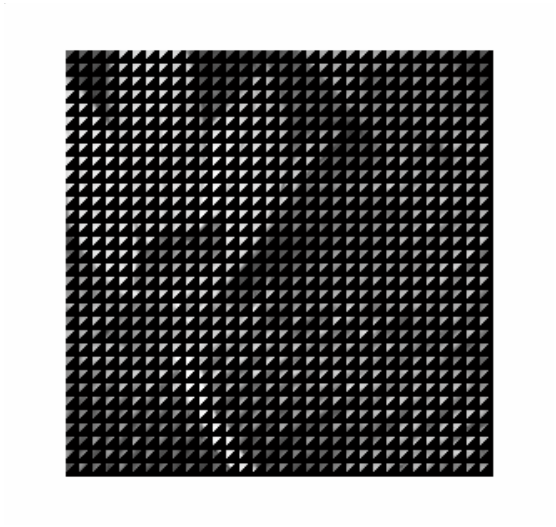


*Fig6.26(g) Compressed Flower image (a=.1, CR%=*84.3750*, RMSE=*52.5716*)*

Fig6.26: Simulation results for Flower image (a=0.1)

Table6.18: Values of RMSE, PSNR for different CR% for Flower image (a=0.1)

Compression Ratio (%)	RMSE	PSNR
56.2500	7.3874	30.7610
67.1875	47.2372	14.6451
76.5625	50.2091	14.1152
84.3750	52.5716	13.7158
90.6250	54.5326	13.3977
95.3125	55.8893	13.1842
98.4375	56.7327	13.0541

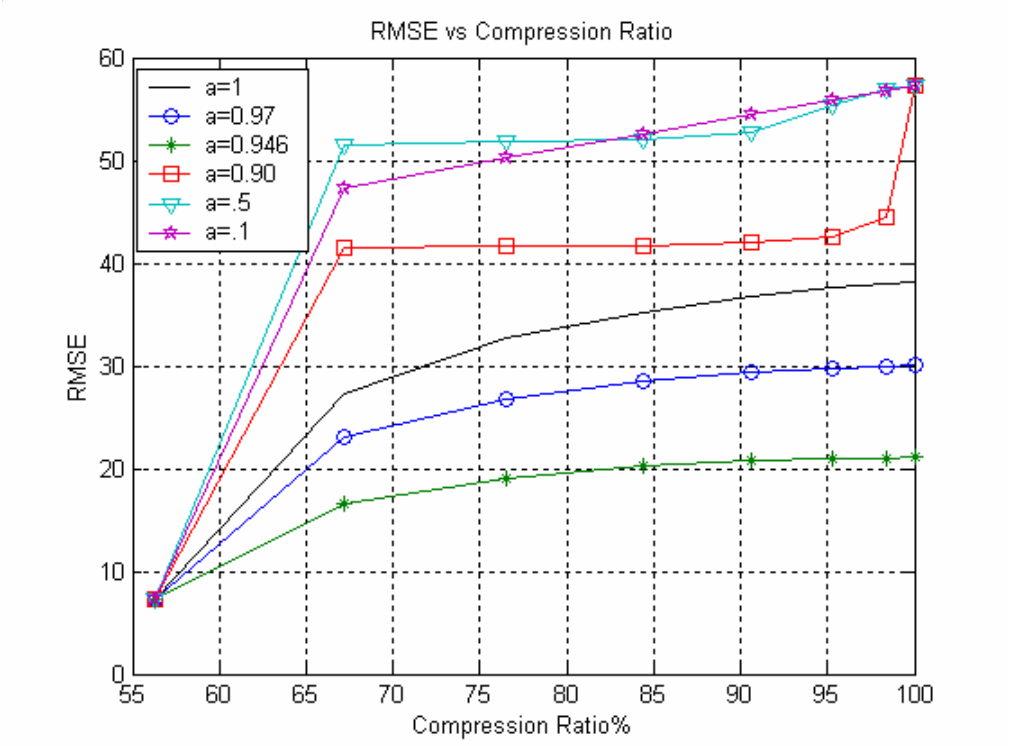


Figure6.27: RMSE versus CR% for different values of 'a' for Flower image

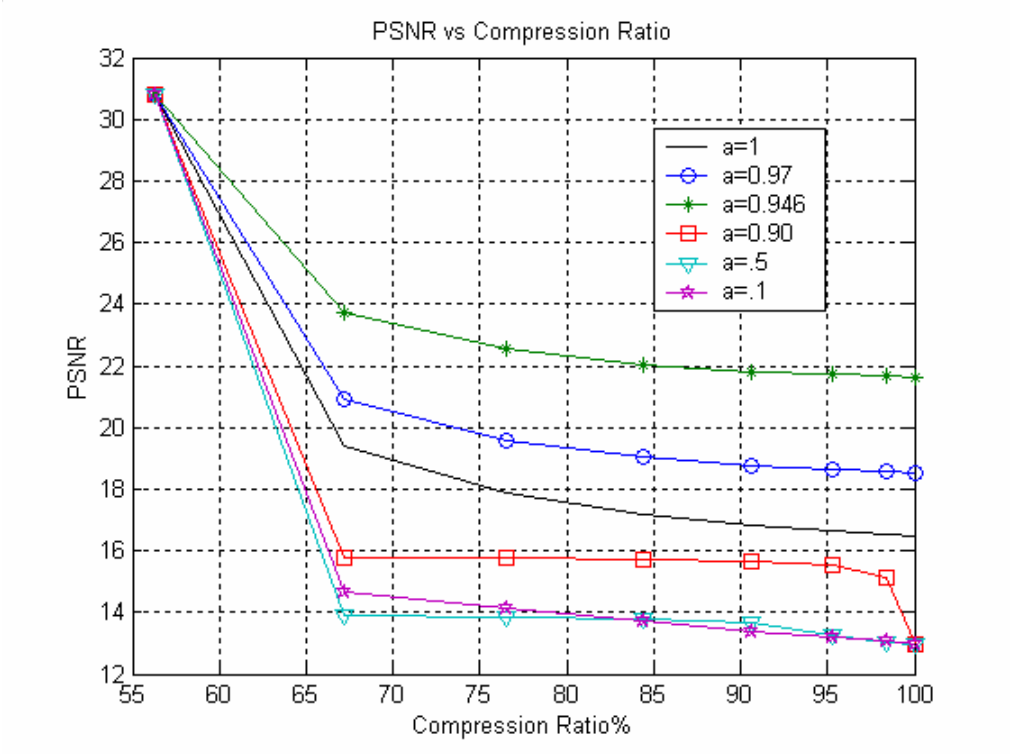


Figure6.28: PSNR versus CR% for different values of 'a' for Flower image

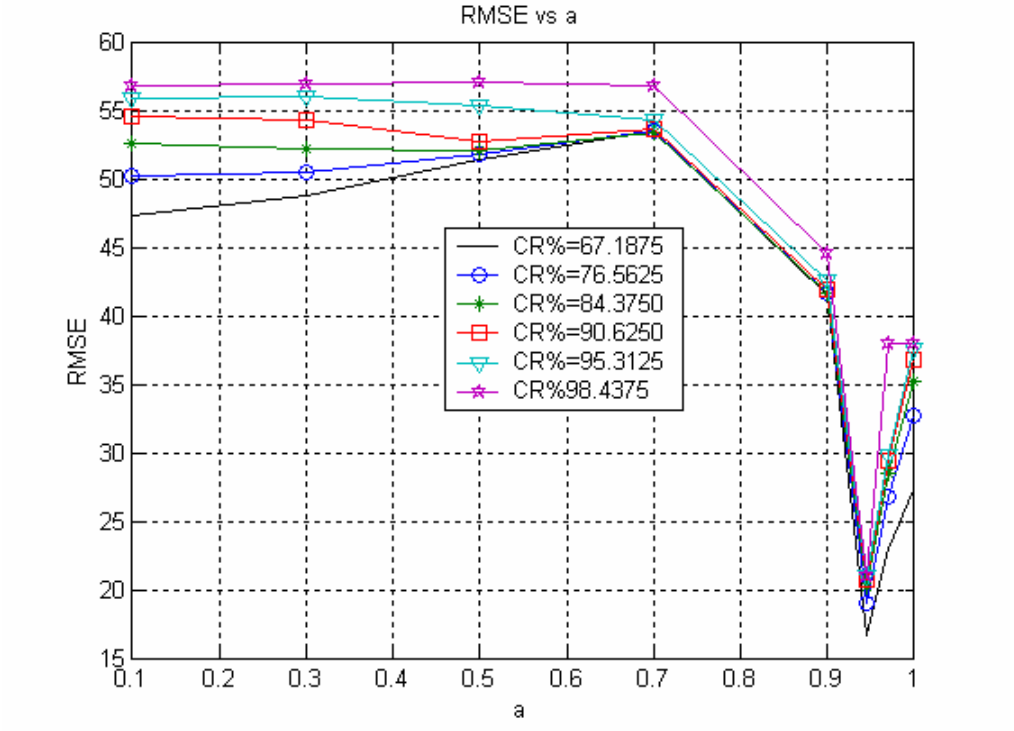


Figure6.29: RMSE versus 'a' for different values of CR% for Flower image

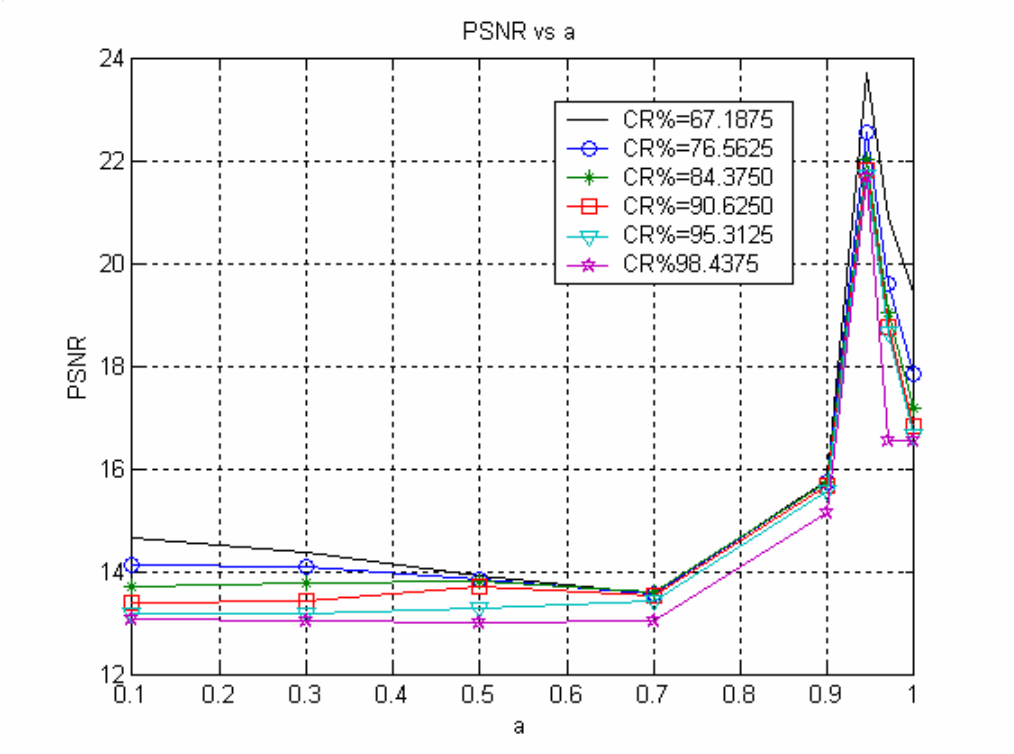


Figure6.30: PSNR versus 'a' for different values of CR% for Flower image

6.5 Rice image

6.5.1 Effect of variation of Compression Ratio on Root Mean Square Error

Figure 6.31 (a) shows the original Rice image. Figure 6.31 (b), (d), (f) shows the decompressed Rice image for different compression ratio for a=1. Figure 6.31(c), (e), (f) shows the compressed image for figure6.31 (b), (d), (f) respectively. As compression ratio is increased Root Mean Square Error is also increased. At the highest CR%, RMSE is highest and at lowest CR%, RMSE is highest. Figure 6.32 to figure 6.36 shows the different compressed and decompressed Rice images for a=0.97, 0.946, 0.90, 0.50, 0.1 respectively.

6.5.2 Effect of variation of Compression Ratio on Peak Signal to Noise Ratio

Figure 6.31 (b), (d), (f) shows the decompressed Rice images for different compression ratio for a=1. Figure 6.31(c), (e), (f) shows the compressed

image for figure6.31 (b), (d), (f) respectively. As compression ratio is increased PSNR is decreased. At the highest CR%, PSNR is lowest and at lowest CR%, PSNR is highest. Figure 6.32 to figure 6.36 shows the different compressed and decompressed Rice image for a=0.97, 0.946, 0.90, 0.50, 0.1 respectively.

6.5.3 Effect of variation of 'a' on Root Mean Square Error

Figure 6.31 to 6.36 shows the different compressed and decompressed Rice images for different CR%. From the figures it is clear best that for a=0.946 lowest RMSE is achieved for various values of 'a'. At a=0.1, highest RMSE i.e. poor quality of image is obtained.

6.5.4 Effect of variation of 'a' on Peak signal to Noise Ratio

Figure 6.31 to 6.36 shows the different compressed and decompressed Rice images for different CR%. From the figures it is clear best that for a=0.946 highest PSNR is achieved for various values of 'a'. At a=0.1, lowest PSNR i.e. poor quality of image is obtained.

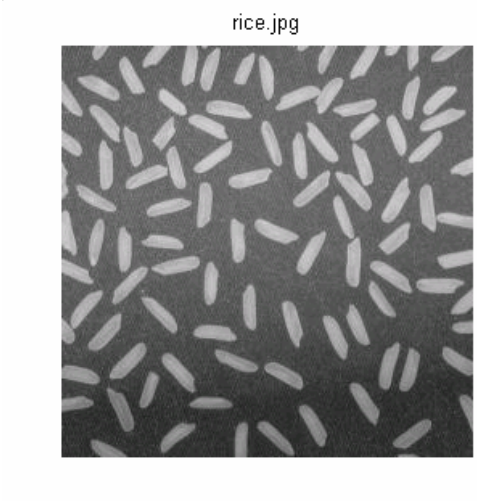


Fig6.31(a) Original Rice Image

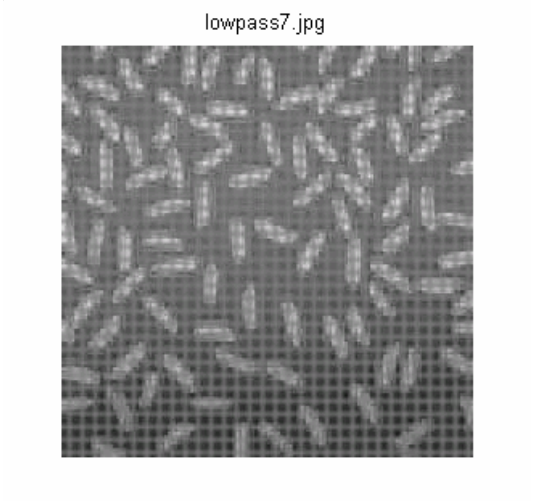
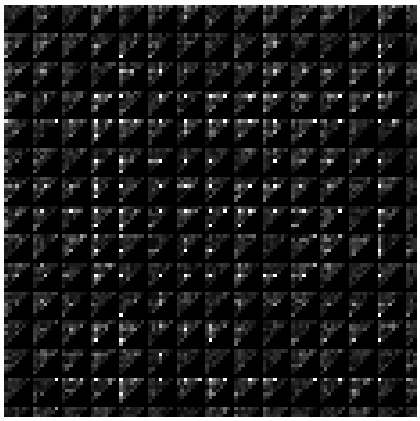


Fig6.31(b)Decompressed Rice image

(a=1, CR%=67.1875, RMSE=25.3267)



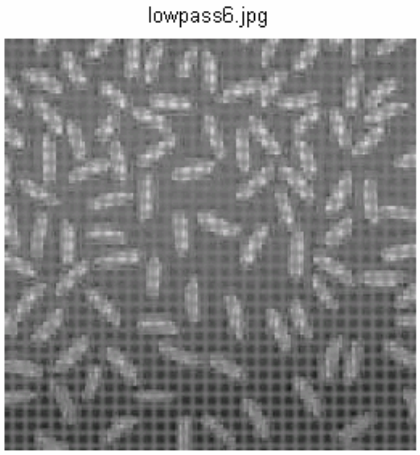
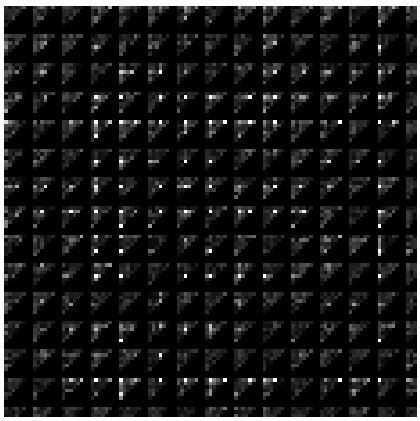


Fig6.31(c) Compressed Rice image (a=1, CR%=67.1875, RMSE=25.3267)

Fig6.31(d)Decompressed Rice image (a=1, CR%=76.5625, RMSE=30.8138)



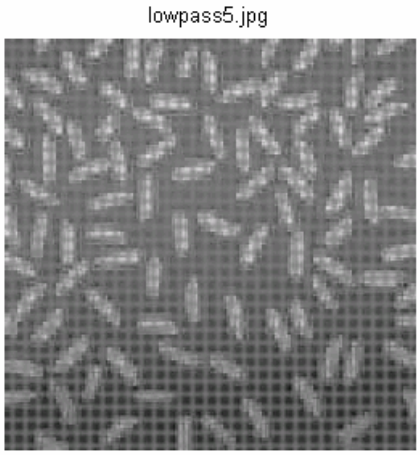


Fig6.31(e)Compressed Rice image

Fig6.31(f) Decompressed Rice image

*(a=1, CR%=*76.5625,*RMSE=*30.8138) *(a=1, CR%=*84.3750,*RMSE=*33.3319*)*

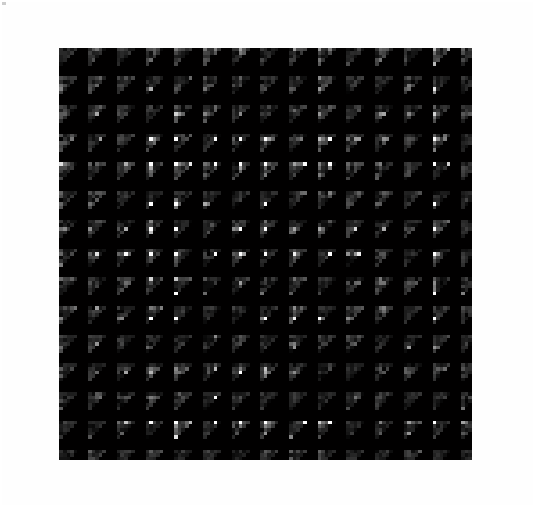
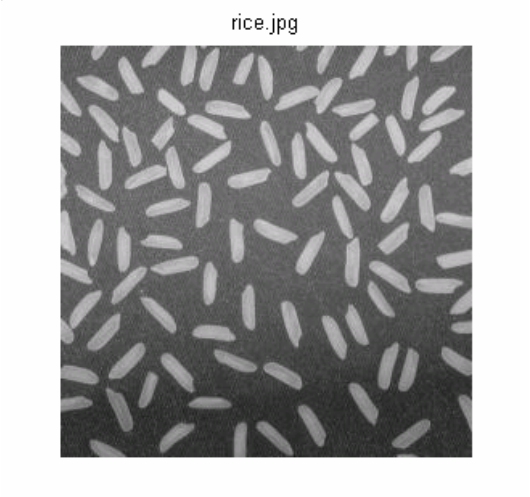


*Fig6.31(g) Compressed Rice image (a=1, CR%=*84.3750*, RMSE=*33.3319*)*

Fig6.31: Simulation results for Rice image (a=1)

Table6.19: Values of RMSE, PSNR for different CR% for Rice image (a=1)

Compression Ratio (%)	RMSE	PSNR
56.2500	4.3948	35.2720
67.1875	25.3267	20.0592
76.5625	30.8138	18.3559
84.3750	33.3319	17.6736
90.6250	34.7329	17.3160
95.3125	35.5627	17.1109
98.4375	36.0938	16.9822



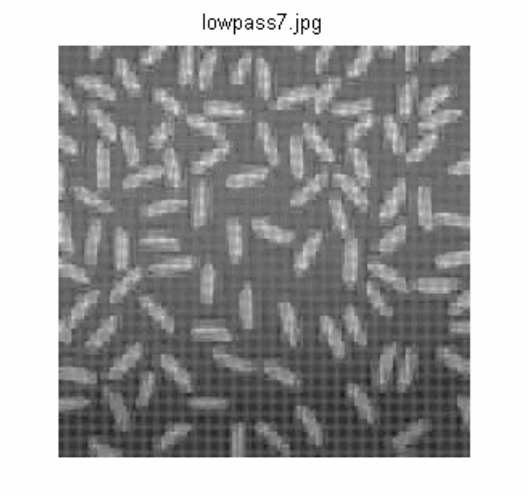


Fig6.32(a) Original Rice Image *Fig6.32(b) Decompressed Rice image*

(*a*=*.97,CR%*=67.1875,*RMSE*=22.9409)

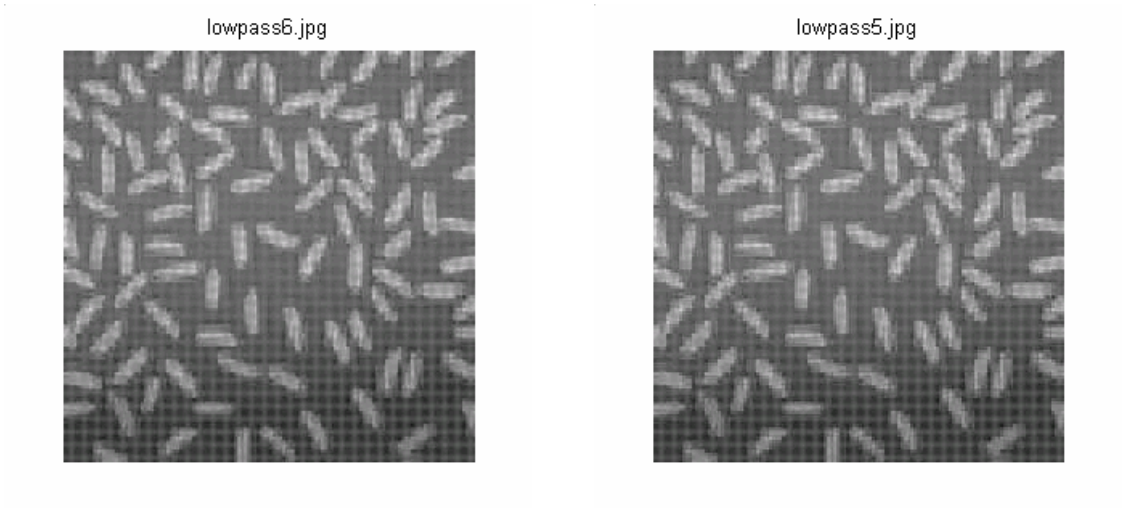


Fig6.32(c) Decompressed Rice image (a=.97,CR%=76.5625,RMSE=27.5277)

Fig6.32(d)Decompressed Rice image (a=.97,CR%=84.3750,RMSE=29.5074)

Fig6.32: Simulation results for Rice image (a=0.97)

Table6.20: Values of RMSE, PSNR for different CR% for Rice image (a=0.97)

Compression Ratio (%)	RMSE	PSNR
56.2500	4.3873	35.2869
67.1875	22.9409	20.9186
76.5625	27.5277	19.3354
84.3750	29.5074	18.7322
90.6250	30.4706	18.4532
95.3125	30.9753	18.3105
98.4375	31.2944	18.2215

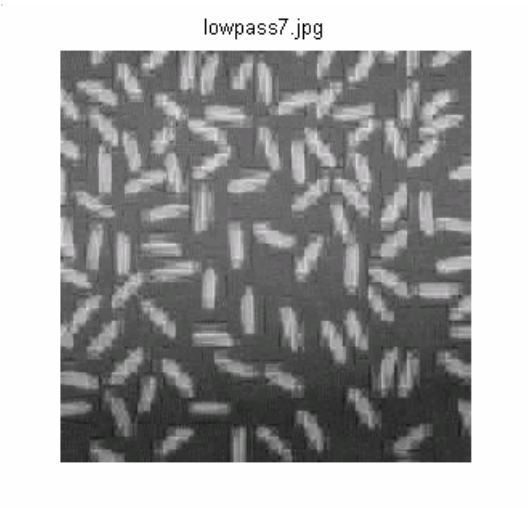


Fig6.33(a)Original Rice Image *Fig6.33(b)Decompressed Rice image*

(a=.946,CR%=67.1875,RMSE=19.2811)

Fig6.33(c)Decompressed Rice image *Fig6.33(d)Decompressed Rice image*

*(a=.946,CR%=*76.5625,*RMSE*=22.7199) *(a=.946, CR%=*84.3750,*RMSE*=24.1937*)*

Fig6.33: Simulation results for Rice image (a=0.946)

Table6.21: Values of RMSE, PSNR for different CR% for Rice image (a=0.946)

Compression Ratio (%)	RMSE	PSNR
56.2500	4.3235	35.4141
67.1875	19.2811	22.4282
76.5625	22.7199	21.0027
84.3750	24.1937	20.4568
90.6250	24.8307	20.2310
95.3125	25.1081	20.1345
98.4375	25.2531	20.0845

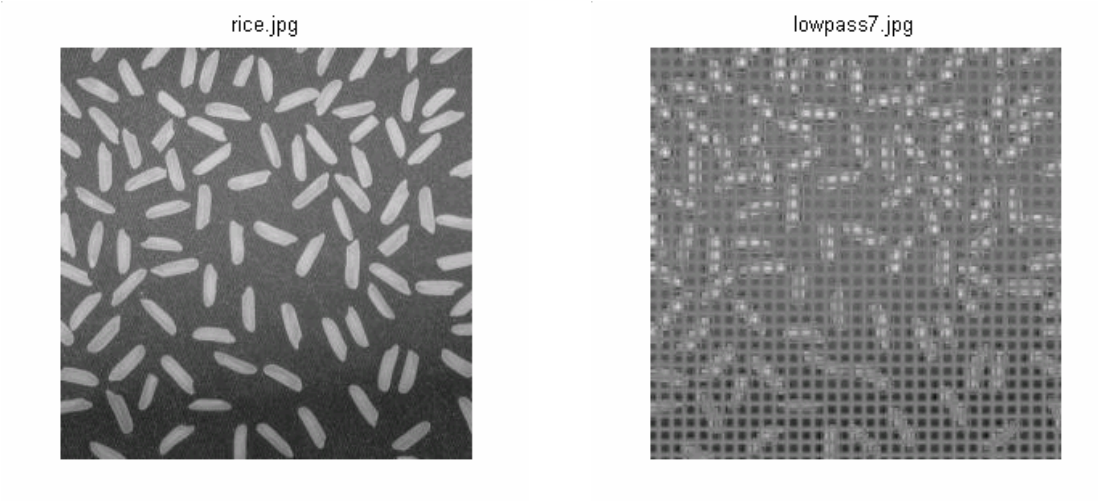
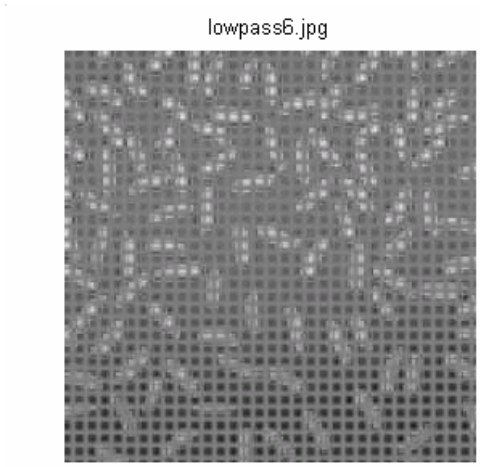


Fig6.34(a)Original Rice Image *Fig6.34(b)Decompressed Rice image (a=.9, CR%=67.1875,RMSE=33.6213)*



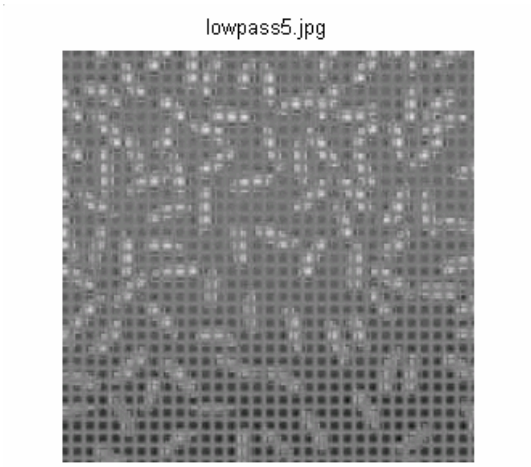


Fig6.34(c)Decompressed Rice image (a=.9, CR%=76.5625, RMSE=33.6936) *Fig6.34(d)Decompressed Rice image (a=.9, CR%=84.3750,RMSE=33.8161)*

Fig6.34: Simulation results for Rice image (a=0.90)

Table6.22: Values of RMSE, PSNR for different CR% for Rice image (a=0.90)

Compression Ratio (%)	RMSE	PSNR
56.2500	4.4005	35.2608
67.1875	33.6213	17.5985
76.5625	33.6936	17.5799
84.3750	33.8161	17.5483
90.6250	34.1936	17.4519
95.3125	35.0727	17.2314
98.4375	37.5694	16.6341

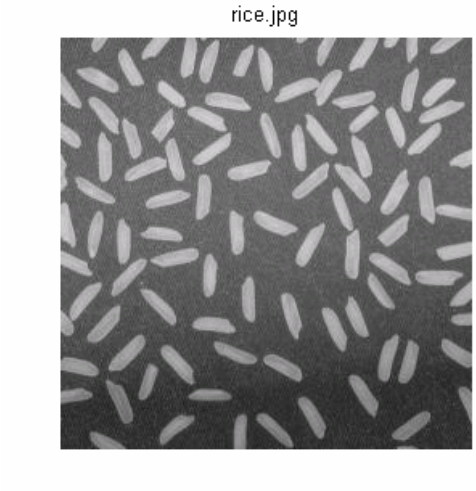


Fig6.35(a)Original Rice Image

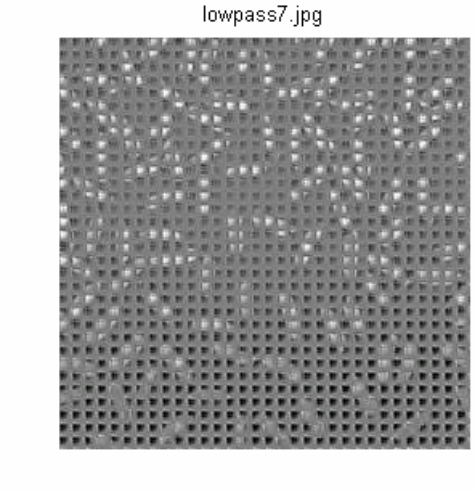


Fig6.35(b)Decompressed Rice image

(a=.5, *CR%*=67.1875,*RMSE*=40.7534)

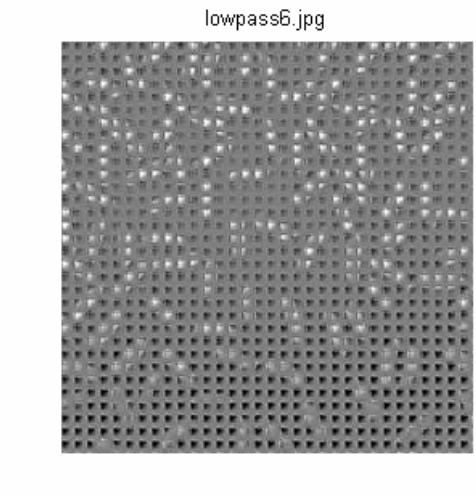


Fig6.35(c)Decompressed Rice image

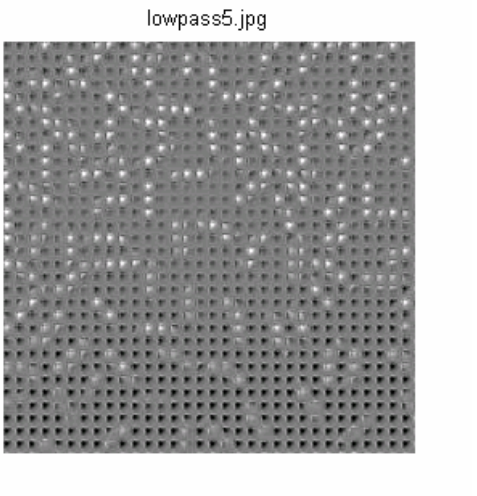


Fig6.35(d)Decompressed Rice image

(a=.5, CR%=76.5625, *RMSE*=41.1401*)* *(a=.5, CR%*=84.3750, *RMSE*=41.3816)

Fig6.35: Simulation results for Rice image (a=0.5)

Table6.23: Values of RMSE, PSNR for different CR% for Rice image (a=0.5)

Compression Ratio (%)	RMSE	PSNR
56.2500	4.4199	35.2226
67.1875	40.7534	15.9275
76.5625	41.1401	15.8455
84.3750	41.3816	15.7947
90.6250	41.7443	15.7189
95.3125	43.8936	15.2828
98.4375	45.2855	15.0116

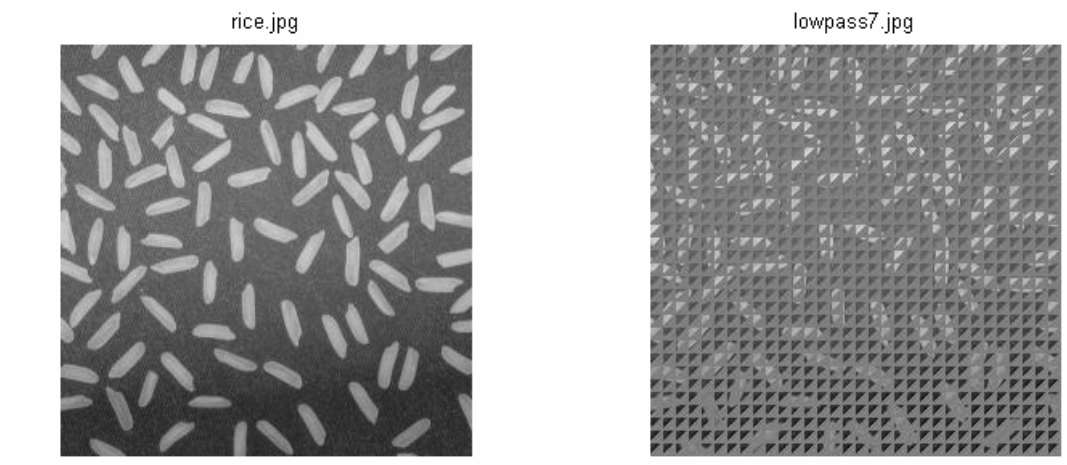


Fig6.36(a) Original Rice Image

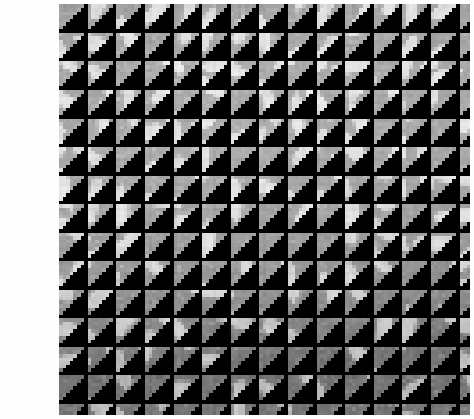


Fig6.36(b) Decompressed Rice image

(a=.1, CR%=67.1875, *RMSE*=37.6006)

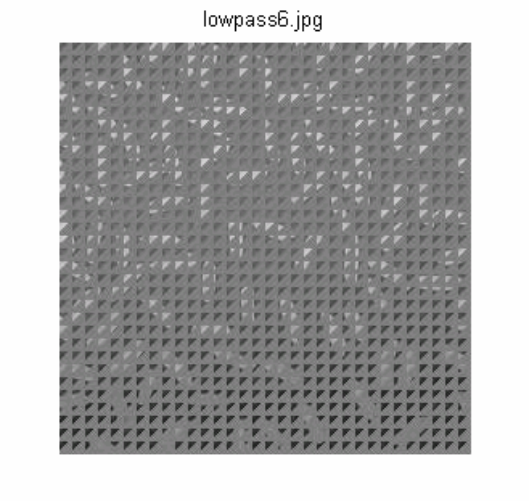


Fig6.36(c) Compressed Rice image

(a=.1, CR%=67.1875, *RMSE*=37.6006)

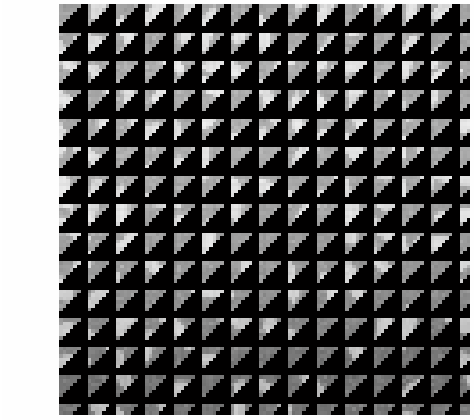


Fig6.36(d)Decompressed Rice image

(a=.1, CR%=76.5625, *RMSE*=39.9813)

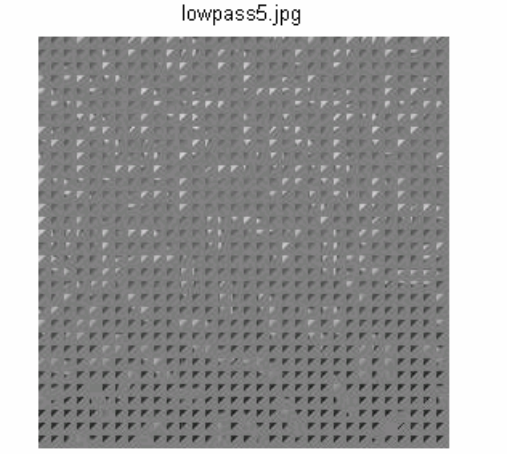


Fig6.36(e)Compressed Rice image (a=.1, CR%=76.5625, RMSE=39.9813)

Fig6.36(f)Decompressed Rice image (a=.1, CR%=84.3750, RMSE=41.8687)

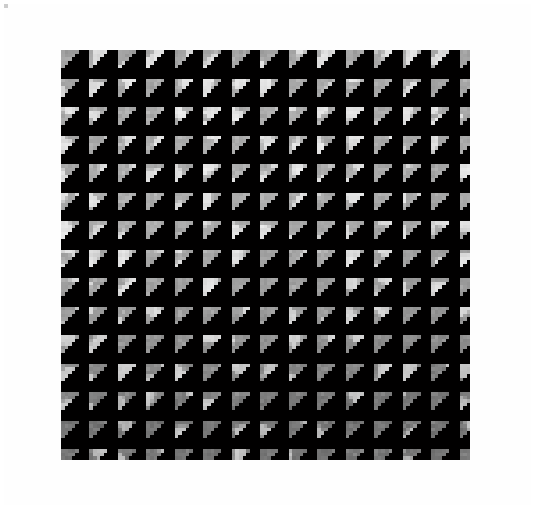


Fig6.36(g) Decompressed Rice image (a=.1, CR%=84.3750, RMSE=41.8687)

Fig6.36: Simulation results for Rice image (a=0.1)

Table6.24 Values of RMSE, PSNR for different CR% for Rice image (a=0.1)

Compression Ratio (%)	RMSE	PSNR
56.2500	4.4395	35.1841
67.1875	37.6006	16.6269
76.5625	39.9813	16.0937
84.3750	41.8687	15.6930
90.6250	43.4167	15.3777
95.3125	44.4906	15.1654

98.4375	45.1607	15.0356

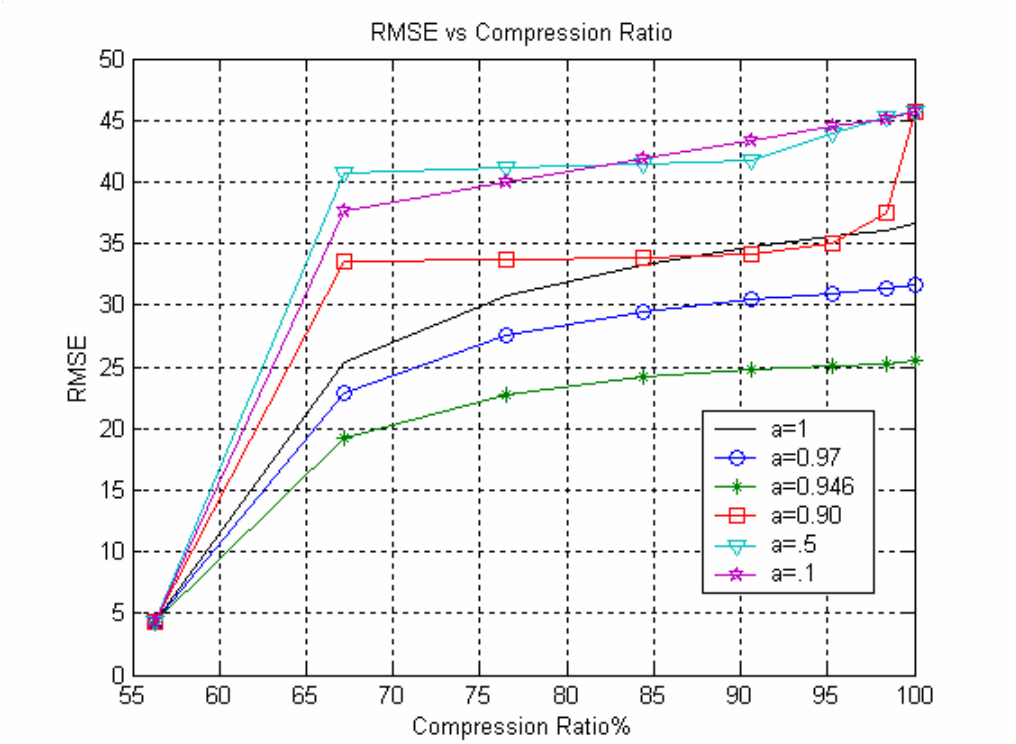


Figure6.37: RMSE versus CR% for different values of 'a' for Rice image

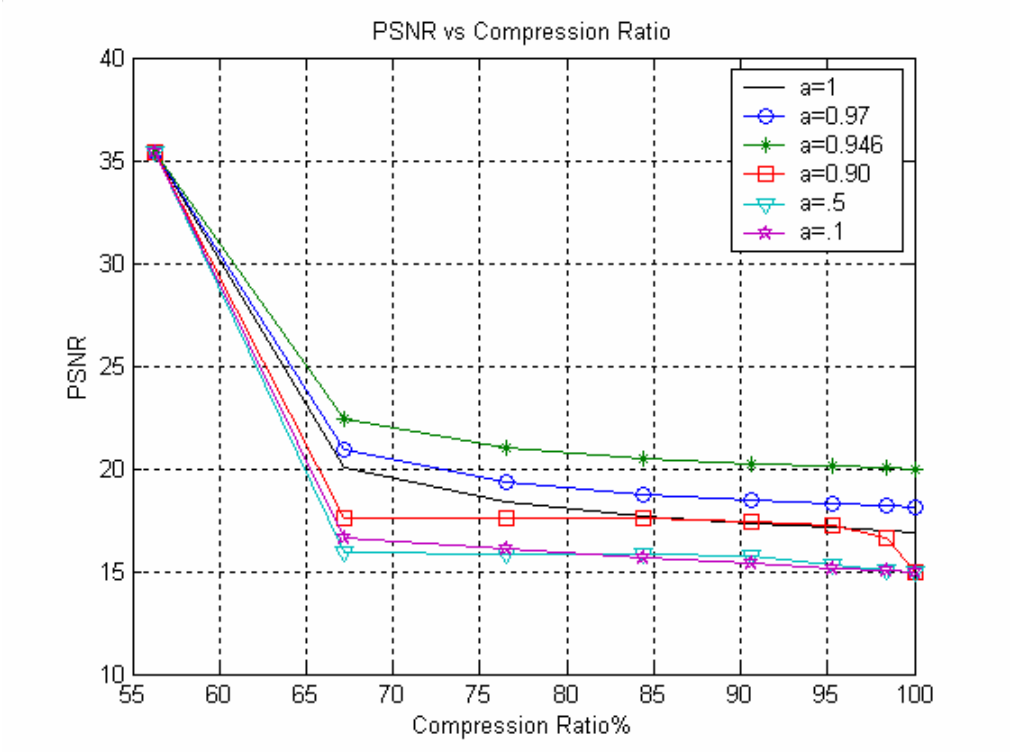


Figure6.38: PSNR versus CR% for different values of 'a' for Rice image

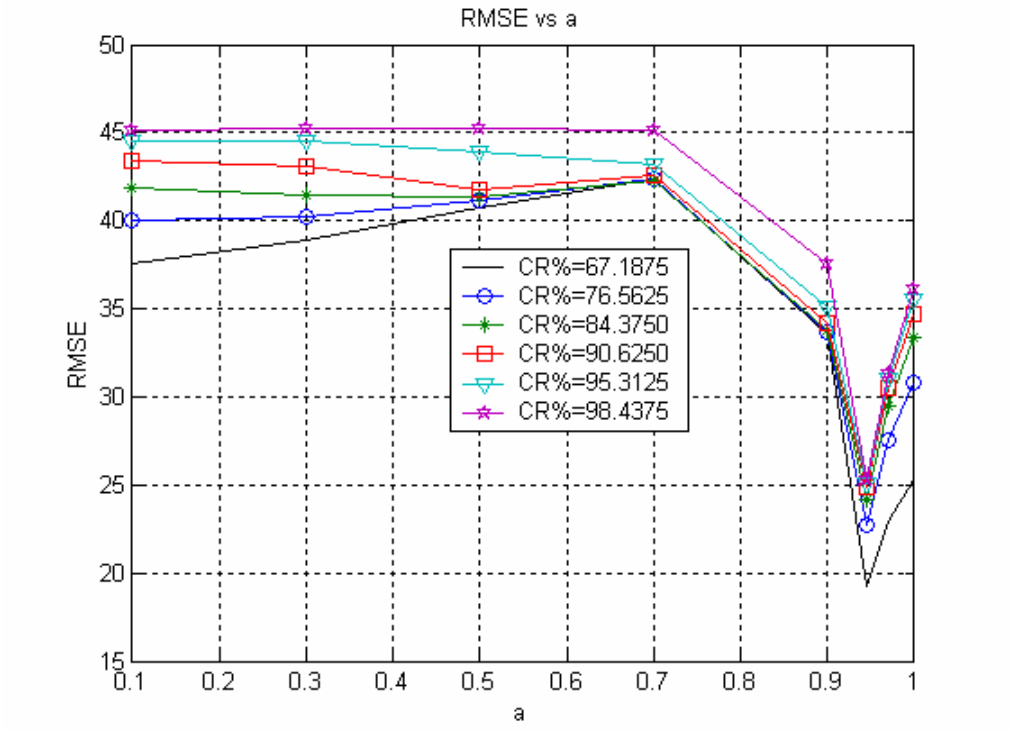


Figure6.39: RMSE versus 'a' for different values of CR% for Rice image

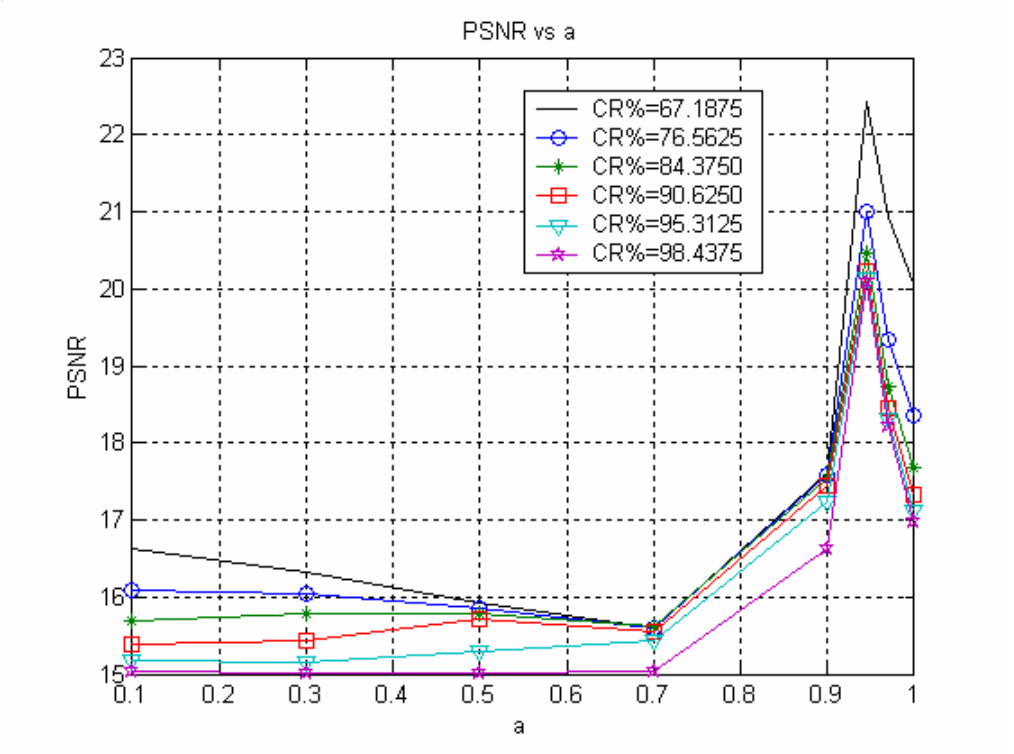


Figure6.40: PSNR versus 'a' for different values of CR% for Rice image

6.6 Barbara Image

6.6.1 Effect of variation of Compression Ratio on Root Mean Square Error

Figure 6.41 (a) shows the original Barbara image. Figure 6.41 (b), (d), (f) shows the decompressed Barbara image for different compression ratio for a=1. Figure 6.41(c), (e), (f) shows the compressed image for figure6.41 (b), (d), (f) respectively. As compression ratio is increased Root Mean Square Error is also increased. At the highest CR%, RMSE is highest and at lowest CR%, RMSE is highest. Figure 6.42 to figure 6.46 shows the different compressed and decompressed Barbara image for a=0.97, 0.946, 0.90, 0.50, 0.1 respectively.

6.6.2 Effect of variation of Compression Ratio on Peak Signal to Noise Ratio

Figure 6.41 (b), (d), (f) shows the decompressed Barbara image for different compression ratio for a=1. Figure 6.41(c), (e), (f) shows the compressed

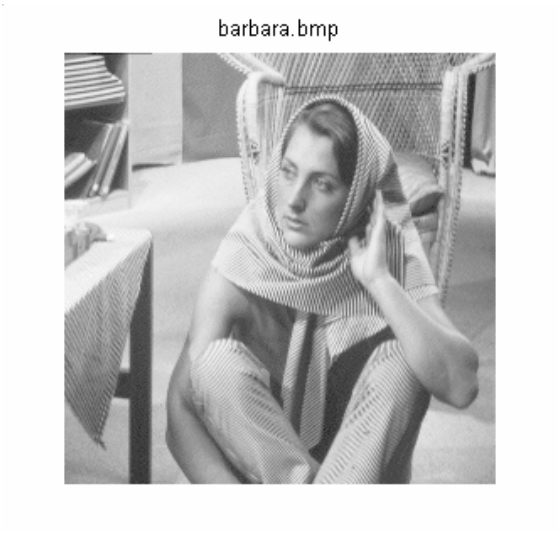
image for figure6.41 (b), (d), (f) respectively. As compression ratio is increased PSNR is decreased. At the highest CR%, PSNR is lowest and at lowest CR%, PSNR is highest. Figure 6.42 to figure 6.46 shows the different compressed and decompressed Barbara image for a=0.97, 0.946, 0.90, 0.50, 0.1 respectively.

6.6.3 Effect of variation of 'a' on Root Mean Square Error

Figure 6.41 to 6.46 shows the different compressed and decompressed Barbara images for different CR%. From the figures it is clear best that for a=0.946 lowest RMSE is achieved for various values of 'a'. At a=0.1, highest RMSE i.e. poor quality of image is obtained.

6.6.4 Effect of variation of 'a' on Peak signal to Noise Ratio

Figure 6.41 to 6.46 shows the different compressed and decompressed Barbara images for different CR%. From the figures it is clear best that for a=0.946 highest PSNR is achieved for various values of 'a'. At a=0.1, lowest PSNR i.e. poor quality of image is obtained.



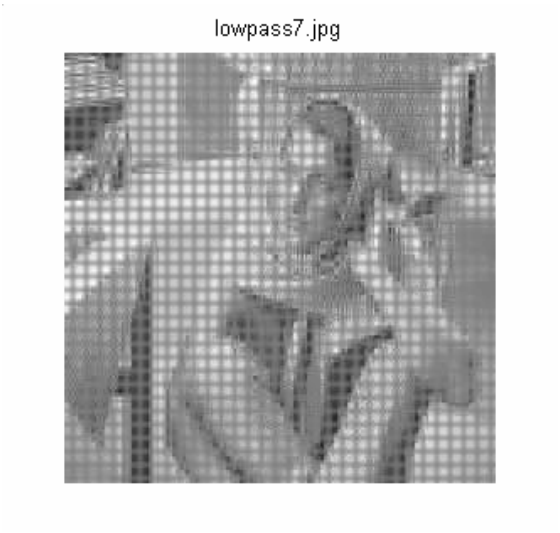


Fig6.41(a) Original Barbara image

Fig6.41(b)Decompressed Barbara image

*(a=1, CR%=*67.1875*, RMSE=*25.2491*)*

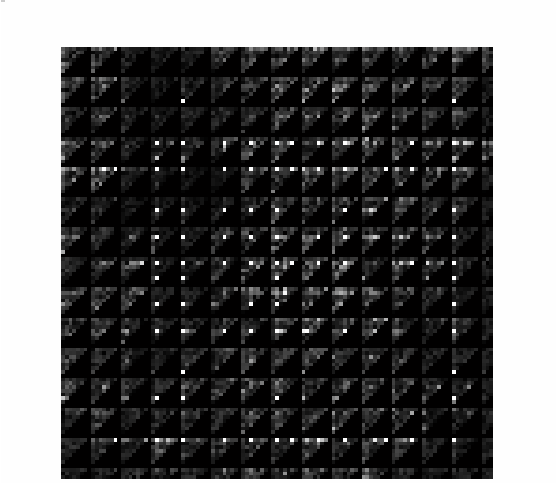


Fig6.41(c) Compressed Barbara image (a=1, CR%=67.1875, RMSE=25.2491)

Fig6.41(d)Decompressed Barbara image (a=1, CR%=76.5625, RMSE=29.8545)

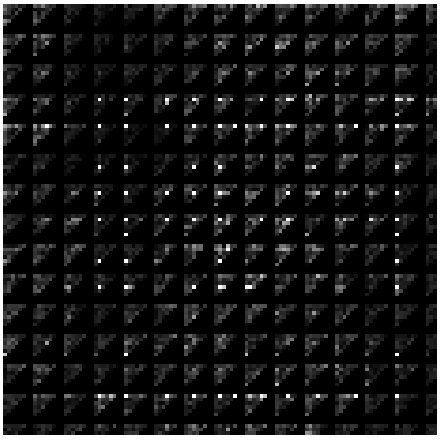


*Fig6.41(e)Compressed Barbara image (a=1, CR%=*76.5625,*RMSE=*29.8545)

*Fig6.41(f) Decompressed Barbara image (a=1, CR%=*84.3750,*RMSE=*31.8621*)*

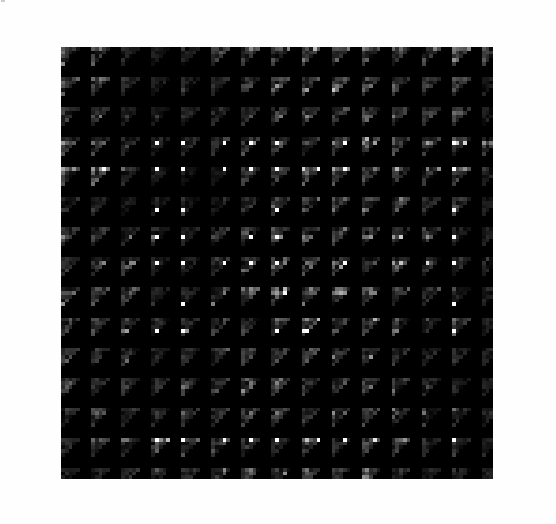


Fig6.41 (g) Compressed Barbara image (a=1, CR%=84.3750, RMSE=31.8621)

Fig6.41: Simulation results for Barbara image (a=1)

Table6.25: Values of RMSE, PSNR for different CR% for Barbara image (a=1)

Compression Ratio (%)	RMSE	PSNR
56.2500	9.7690	28.3338
67.1875	25.2491	20.0859
76.5625	29.8545	18.6306
84.3750	31.8621	18.0653
90.6250	33.0463	17.7483
95.3125	33.8986	17.5272
98.4375	34.6869	17.3275

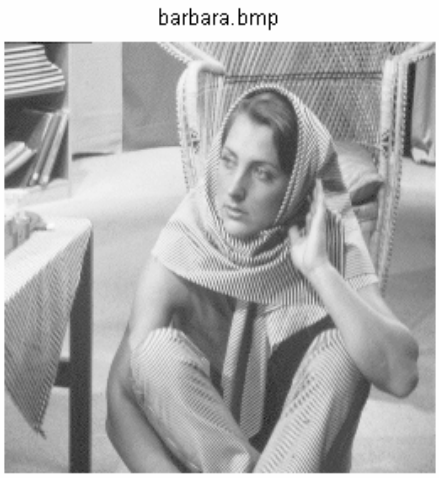


Fig6.42(a) Original Barbara Image *Fig6.42(b) Decompressed Barbara image*

(a=.97, CR%=67.1875, *RMSE*=20.9088)

Fig6.42(c) Decompressed Barbara image (a=.97,CR%=76.5625,RMSE=24.1508)

Fig6.42(d)Decompressed Barbara image (a=.97,CR%=84.3750,RMSE=25.5707)

Fig6.42: Simulation results for Barbara image (a=0.97)

Table6.26: Values of RMSE, PSNR for different CR% for Barbara image (a=0.97)

Compression Ratio (%)	RMSE	PSNR
56.2500	9.6555	28.4353
67.1875	20.9088	21.7242
76.5625	24.1508	20.4722
84.3750	25.5707	19.9759
90.6250	26.3109	19.7281
95.3125	26.7806	19.5744
98.4375	27.1860	19.4439

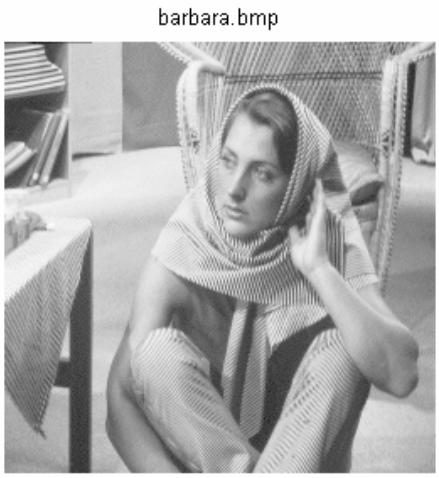


Fig6.43(a)Original Barbara Image *Fig6.43(b)Decompressed Barbara image*

(a=.946,CR%=67.1875,RMSE=14.5224)

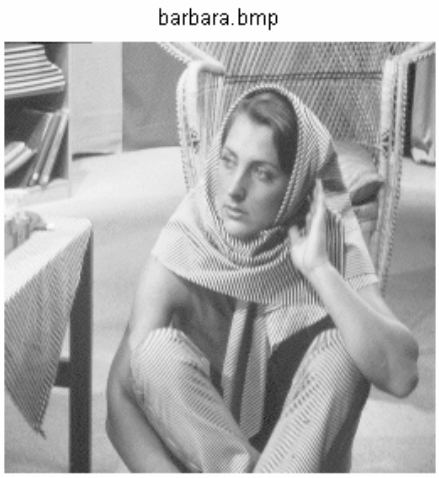
*Fig6.43(c)Decompressed Barbara image (a=.946,CR%=*76.5625,*RMSE*=16.6556)

*Fig6.43(d)Decompressed Barbara image (a=.946, CR%=*84.3750,*RMSE*=17.6115*)*

Fig6.43: Simulation results for Barbara image (a=0.946)

Table6.27: Values of RMSE, PSNR for different CR% for Barbara image (a=0.946)

Compression Ratio (%)	RMSE	PSNR
56.2500	9.4990	28.5772
67.1875	14.5224	24.8900
76.5625	16.6556	23.6996
84.3750	17.6115	23.2149
90.6250	18.0718	22.9908
95.3125	18.3119	22.8761
98.4375	18.4753	22.7990



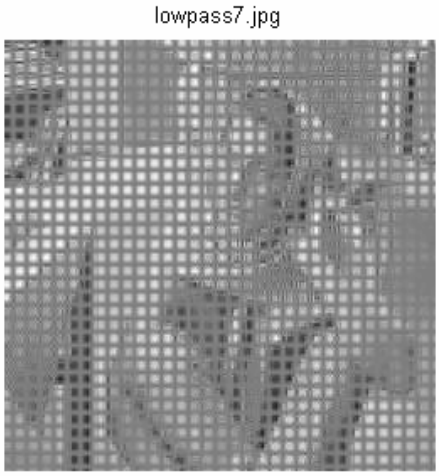
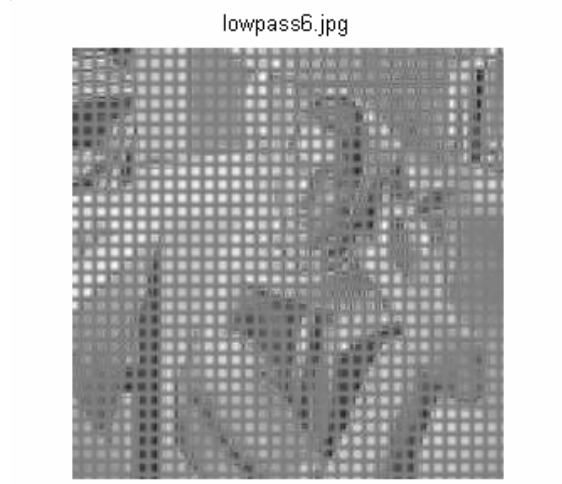


Fig6.44(a)Original Barbara Image *Fig6.44(b)Decompressed Barbara image*

(a=.9, *CR%*=67.1875,*RMSE*=43.8946)



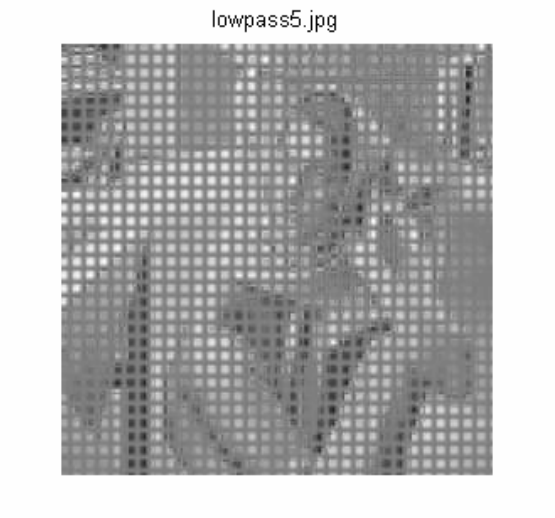


Fig6.44(c)Decompressed Barbara image (a=.9, CR%=76.5625, RMSE=44.0113)

Fig6.44(d)Decompressed Barbara image (a=.9, CR%=84.3750,RMSE=44.1327)

Fig6.44: Simulation results for Barbara image (a=0.90)

Table6.28: Values of RMSE, PSNR for different CR% for Barbara image (a=0.90)

Compression Ratio (%)	RMSE	PSNR
56.2500	9.6631	28.4285
67.1875	43.8946	15.2826
76.5625	44.0113	15.2595
84.3750	44.1327	15.2356
90.6250	44.3943	15.1843
95.3125	44.8234	15.1007
98.4375	46.8208	14.7220

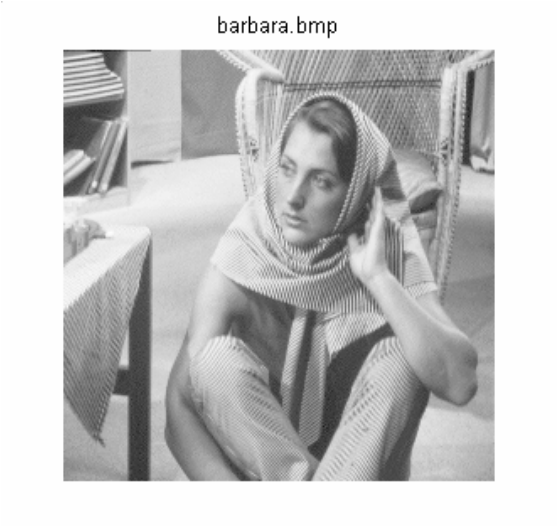


Fig6.45(a)Original Barbara Image

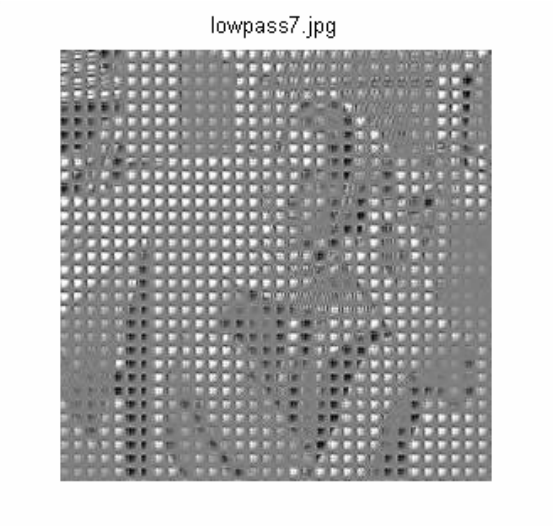
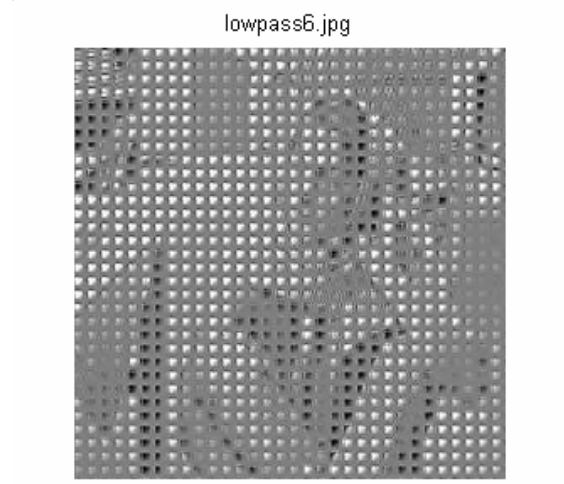


Fig6.45(b)Decompressed Barbara image

(a=.5, CR%=67.1875,*RMSE*=53.5410)



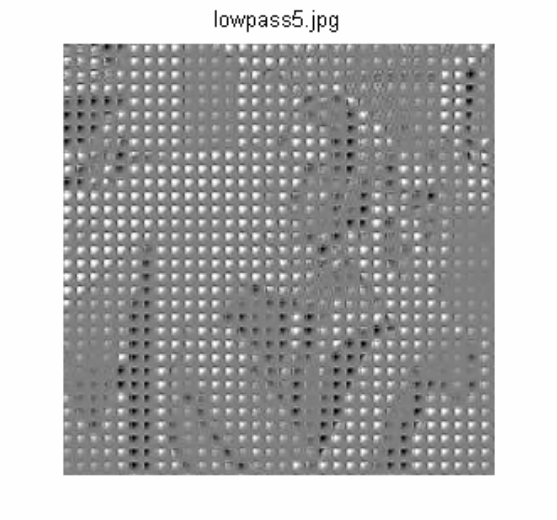


Fig6.45(c)Decompressed Barbara image (a=.5, CR%=76.5625, RMSE=54.0028)

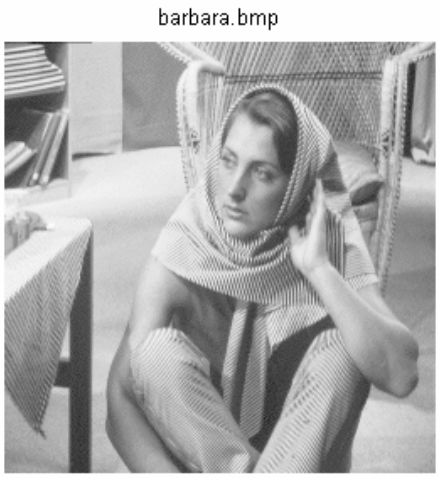
Fig6.45(d)Decompressed Barbara image (a=.5, CR%=84.3750, RMSE=54.3581)

Fig6.45: Simulation results for Barbara image (a=0.5)

Table6.29: Values of RMSE, PSNR for different CR% for Barbara image (a=0.5)

Compression Ratio (%)	RMSE	PSNR
56.2500	9.6696	28.4226

67.1875	53.5410	13.5571
76.5625	54.0028	13.4825
84.3750	54.3581	13.4255
90.6250	55.2581	13.2829
95.3125	57.9348	12.8720
98.4375	59.6022	12.6256



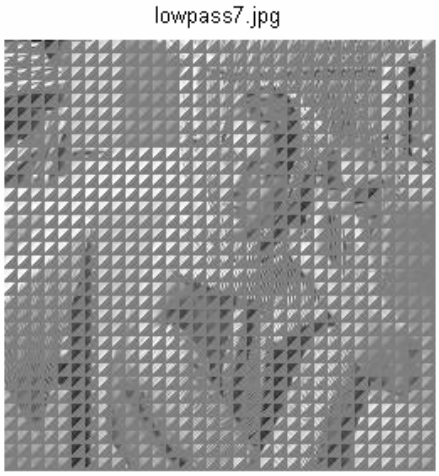


Fig6.46(a) Original Barbara Image

Fig6.46(b) Decompressed Barbara image

(a=.1, CR%=67.1875, *RMSE*=49.2215)

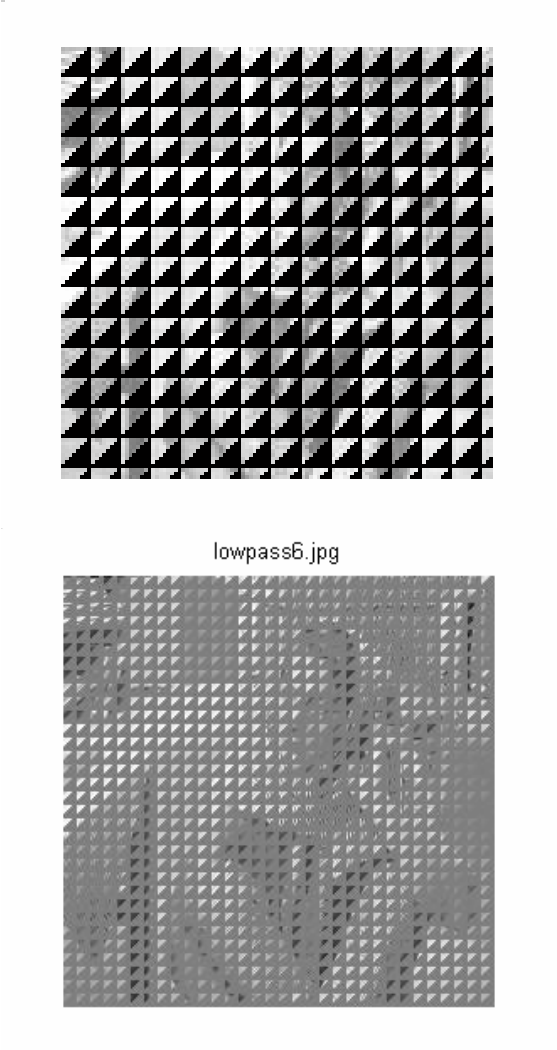


Fig6.46(c) Compressed Barbara image (a=.1, CR%=67.1875, RMSE=49.2215)

Fig6.46(d)Decompressed Barbara image (a=.1, CR%=76.5625, RMSE=52.3622)

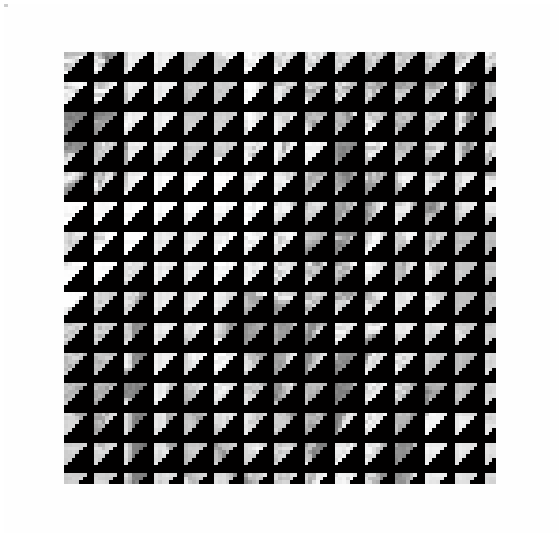


Fig6.46(e)Compressed Barbara image (a=.1, CR%=76.5625, RMSE=52.3622)

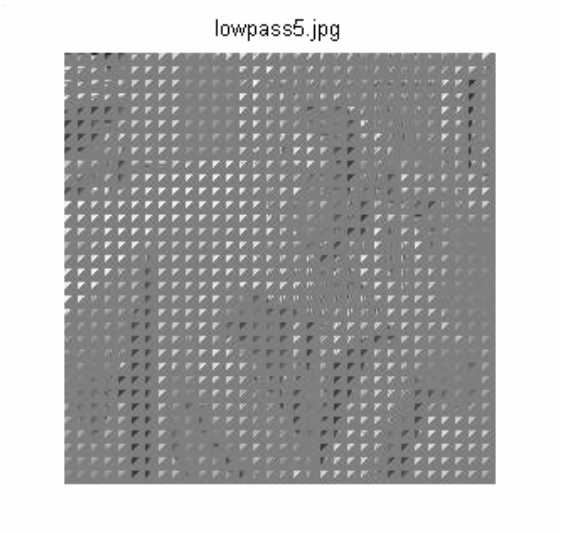


Fig6.46(f)Decompressed Barbara image (a=.1, CR%=84.3750, RMSE=54.8738)

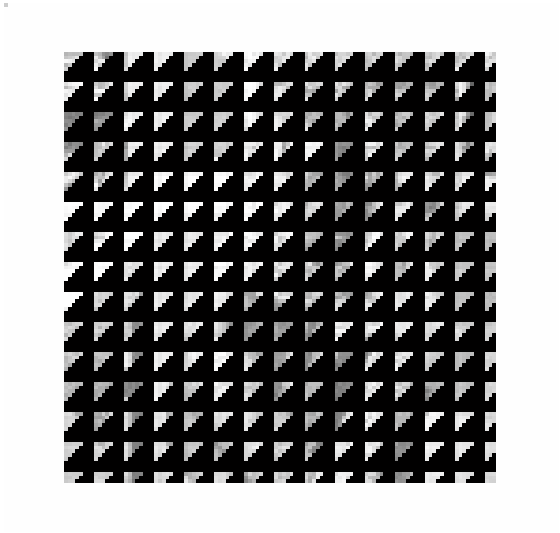


Fig6.46(g) Compressed Barbara image (a=.1, CR%=84.3750, RMSE=54.8738)

Fig6.46: Simulation results for Barbara image (a=0.1)

Table6.30: Values of RMSE, PSNR for different CR% for Barbara image (a=0.1)

Compression Ratio (%)	RMSE	PSNR
56.2500	9.7193	28.3781
67.1875	49.2215	14.2877
76.5625	52.3622	13.7504
84.3750	54.8738	13.3435
90.6250	56.9625	13.0190
95.3125	58.4187	12.7998
98.4375	59.3384	12.6641

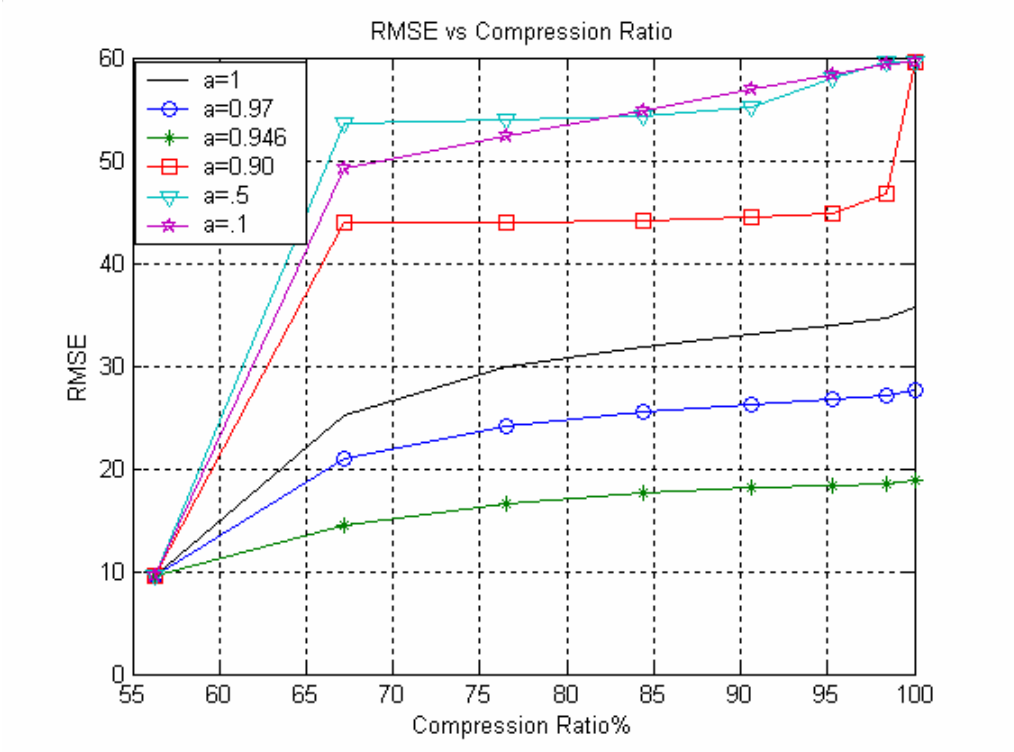


Figure6.47: RMSE versus CR% for different values of 'a' for Barbara image

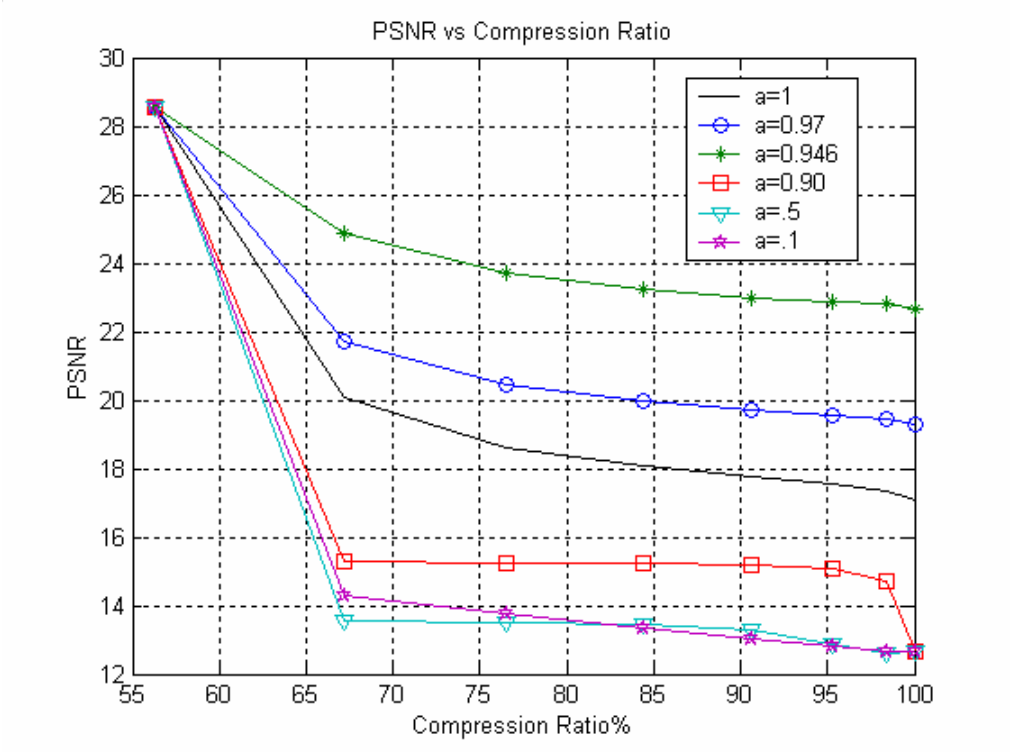


Figure6.48: PSNR versus CR% for different values of 'a' for Barbara image

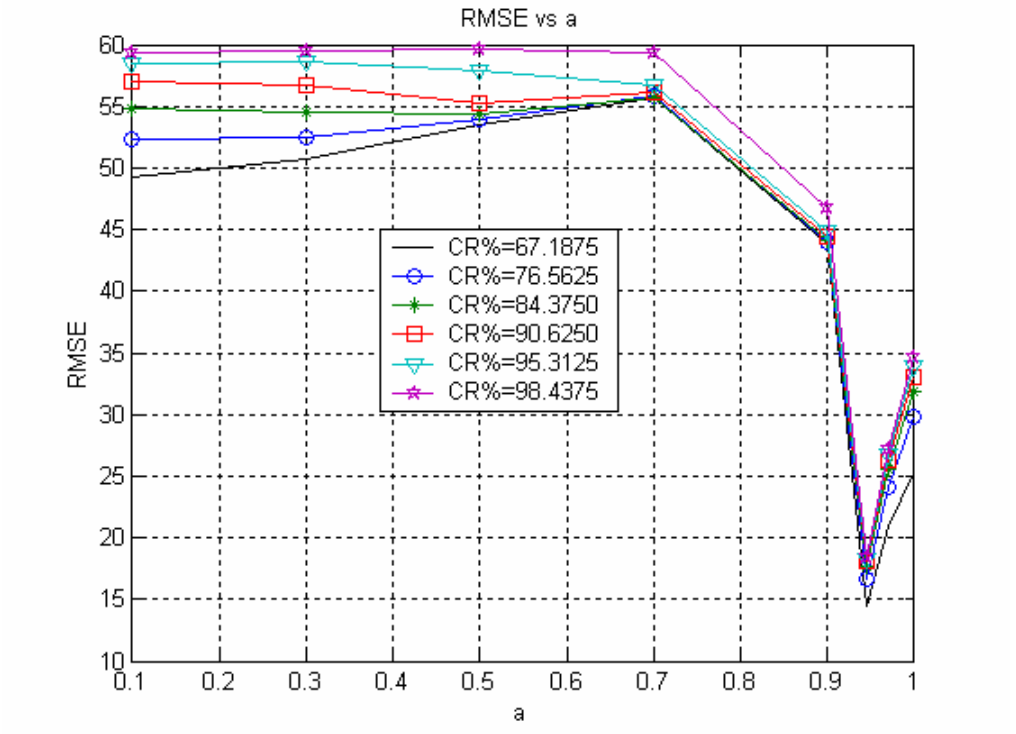


Figure6.49: RMSE versus 'a' for different values of CR% for Barbara image

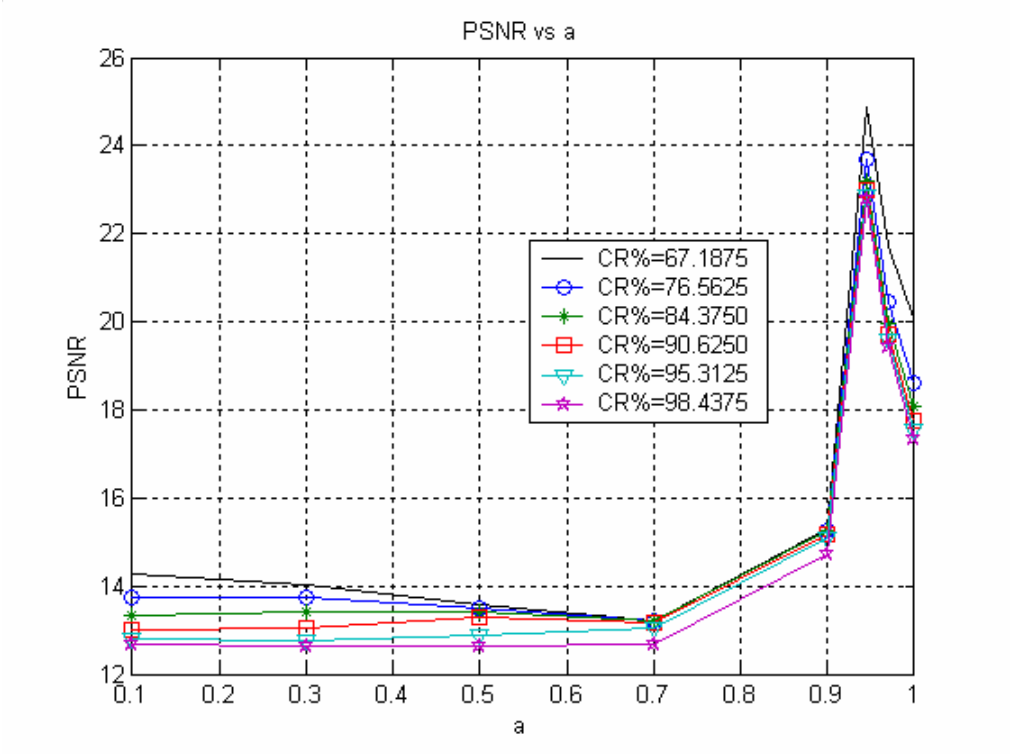


Figure6.50: PSNR versus 'a' for different values of CR% for Barbara image

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

In image processing an important part is compression of image, it reduces the amount of data required to represent the image. Transform compression technique is one of the important and commonly used technique for image compression. With the advent of fractional calculus the idea to use fractional transform for image compression is gaining popularity and researchers are trying to use various fractional transforms for image compression. This is because of the additional degree of freedom provided by its fractional order 'a' to achieve an optimal domain for which lesser RMSE for same amount of compression and better decompressed image is retained. By varying the parameter 'a' we can achieve same amount of compression as that of HT, but FRHT gives lesser RMSE and better PSNR i.e. better image quality as

compared to HT. It is clear that for all images a=0.946 is an optimum domain for which better decompressed is retained.

The suitability of various other fractional integral transforms like fractional Haar transform, fractional Wavelet transform etc. may be examined. The results can be compared with the existing integral transforms. For further compression the quantized coefficients can be entropy encoded. Fractional transforms can also be employed for new standards for image, can be formulated on similar lines to JPEG standard.

REFRENCES

[1] A. B. Watson, "Image Compression using the DCT", *Mathematica journal*, pp. 81-88, 1995.

[2] A. Gersho and R. M. Gray, "Vector Quantization and Signal Compression", *Kluwer Academic Publishers*, 1991.

[3] A. K. Jain, "Fundamentals of Digital Image Processing", *Prentice-Hall, Inc*, Englewood cliffs, 1989.

[4] A. W. Lohmann, D. Mendlovic, Z. Zalewsky and R. G. Dorsch, "Some Important Fractional Transformations for Signal Processing", *Optics Communications*, pp.18-20, April,1996

[5] C. C. Chen, "On The Selection Of Image Compression Algorithms", *Proc. International Conference on Pattern Recognition,* Brisbane, Australia, 1998.

[6] C. C. Tseng, "Eighenvalues and Eigenvectors of Generalized DFT, Generalized DHT, DCT-IV and DST-IV Matrices", *IEEE Transactions On Signal Processing*, vol. 50, No. 4, pp. 866-872, 2002.

[7] D. A. Huffman, "A Method for Construction of Minimum-redundancy Codes", *Proc. IRE*, vol 40,no. 10, pp. 1098-1101, 1952.

[8] G. K. Wallance, "The JPEG still Picture Compression Standard", *Comm. ACM*, vol. 34, no. 4, pp. 30-44, 1991.

[9] H.M. Ozaktas, Z.Zalevsky and M.A. Kutay,"The Fractional Fourier Transform with Applications in Optics and Signal Processing",*John Wiley & Sons Ltd*., New York,2000.

[10] J.W. Woods and S.W. O'Neil, "Subband Coding of Images", *IEEE Trans. ASSP*, vol. 34, no. 5, pp. 1278-1288, 1986.

[11] K. R. Persons, P.M. Pallison, A. Manduca, "Ultrasound Grayscale Image Compression with JPEG and Wavelet Techniques", *J Digit Imaging*, vol. 13, pp. 25-32, 2000.

[12] K.S. Miller and B. Ross, "An Introduction to the Fractional Calculus and Fractional Differential Equations", *John Wiely*, 1993.

[13] L. H. Zetterberg, S. Ericsson, C. Couturier, "DPCM Picture Coding with Two Dimensional Control of Adaptive Quantization", *IEEE Trans. Comm.* COM-32, No. 4,pp. 457-462, 1984.

[14] M. Shao and C.L. Nikias, "Signal Processing with Fractional Lower Order Moment: Stable Processes and Their Applications", *Proc. IEEE*, vol. 81, pp. 986-1010, 1993.

[15] R. C. Gonzalez, R. E. Woods, "Digital Image Processing", *Addison-Wesley Co*., 1993.

[16] R.R. Coifman and M.V. Wickerhauser, "Entropy Basede Algorithms for Best Basis Selection", *IEEE Trans Information Theory*, vol. 38, no. 2, pp. 713-718, 1992.

[17] S.C. Pei, C. Tseng, M. H. Yeh, D. Jian-jiun, J. Shyu, "Discrete Fractional Hartley and Fourier Transforms" *IEEE Transactions On Circuits And Systems—Ii: Analog And Digital Signal Processing*, Vol.45, No.6, Pp.665-675,June1998.

[18] S.C. Pei, C. Tseng, M. H. Yeh, D. Jian-jiun, "A New Definition of Continuous Fractional Hartley Transform",*IEEE*, pp 1485-1488, 1998.

[19] T. Luczak, "A Suboptimal Lossy Data Compression Based on Approximate Pattern Matching", *IEEE Trans Information Theory*, vol. 43, pp. 1439-1451, 2001.

[20] V.R. Algazi and J.T. De Witte, "Theoretical Performance of Entropy Coded DPCM", *IEEE Trans. Comm*., vol. 30, no. 5, pp. 1088-1095, 1982.

[21] Z. Jik Tanaka and S. Kitamura, "Block Permutation Coding of Image Using Cosine Transform", *IEEE Trans. Comm*., vol. 43, pp. 2223-2246, 1991.

LIST OF PUBLICATIONS

[1] Performance of Fractional Hartley Transform in Image Compression, accepted for presentation at *International Radar Symposium India (IRSI-05)*, IISc Banglore-India, 2005.

[2] Smart Antennas in Mobile Communication, presented at *national conference on Electronics Circuits and Communication Systems*, pp. 20-23, TIET Patiala-India, 2004.

[3] Wavelet Transform-A Review, presented at *national conference Power Engineering Practices and Energy Management*, pp. 474-476, TIET Patiala-India, 2005.