BETTER THROUGHPUT IMAGE COMPRESSION USING SET PARTITIONING IN HIERARCHICAL TREES WITH ENTROPY CODING

By

Asim Ashfaq

SUBMITTED TO THE FACULTY OF ELECTRICAL ENGINEERING

MILITARY COLLEGE OF SIGNALS,

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY,

RAWALPINDI, PAKISTAN

AUGUST 2017

In The name of Allah the most Beneficent and the most Merciful

DEDICATION

Dedicated to my family who has always been a source of motivation for me and also to my supervisor Col Dr Imran Touqir for being forthcoming and helping in fulfillment of this research work.

ABSTRACT

The increased reliance on images makes the significance of image compression more pronounced than ever. There is wide range of practical values for today that are dealing with the transmission of a large amount of image data therefore, it warrants image compression as a necessity of time for better handling. For last two decades, image compression is being performed more often than not by using most popular "Time Frequency" wavelet transformation. The Discrete Wavelet Transform (DWT) is a type of wavelet transform in which signal is transformed into discretely sampled wavelets. A number of wavelet based image compression techniques like Embedded Zero-tree Wavelet (EZW) transform and Set Partitioning in Hierarchical Trees (SPIHT) are being used to attain better standards of PSNR and compression ratios.

EZW, a computationally simple and very effective technique, is an embedded compression algorithm of its time that works on DWT to predict the absence of significant information by exploiting self-similarities across the scale. However, it lacked the insight about coefficient position, didn't cater for intra-band correlation and its performance with single embedded file was not much pronounced. The improvements in EWZ were brought in with the introduction of SPIHT, which is again a fully embedded codec algorithm. It uses principal of partial ordering by magnitude, set partitioning by importance of magnitude of the coefficients, self-similarity across the scale and ordered bit plan transmission.

SPIHT encodes the transformed coefficients with respect to their importance as compared to a given threshold. Statistical analysis have exhibited that the output bit-stream of SPIHT comprises of long series of zeroes which can be further compressed, therefore SPIHT is not recommended to be used as sole mean of compression. To this end, additional compression is being done by making use of different kinds of entropy encoding schemes. One of the entropy

encoding scheme which is concatenated with SPIHT for further compression is Huffman encoding.

This research is motivated by the requirement of a viable solution for fast transmission and less storage space. This research concentrates on saving comparatively more number of bits without compromising the quality of the image by combining two encodings “Set Partitioning in Hierarchical Tree and Huffman coding. This is done by making deft use Huffman encoding where it yields the optimized results and saves more numbers of bits thereby reducing the storage space and increasing the transmission time.

CERTIFICATE

It is to certify that final copy of thesis written by **Mr Asim Ashfaq** of Military College of Signals has been evaluated by me, found complete in all respect as per the specified format of NUST Statutes / Regulations, is free of plagiarism, errors and mistakes.

Dated: _____ _____

Col Dr Imran Tauqir

DECLARATION

No content of work presented in this thesis has been submitted in support of another award of qualification or degree either in this institution or anywhere else.

ACKNOWLEDGEMENTS

With profound humility, I pay my gratitude to Allah Almighty for enabling me to achieve another astounding milestone in my literary career. I would like to extend my special thanks to the faculty and administration of Military College of Signals and NUST, for proffering a commendable research environment at the institute. This arduous work would have not possible without the support of my supervisor Col. Dr. Imran Tauqir who not only provided timely guidance, profound encouragement and positive criticism but also ensured that I complete the assigned tasks in stipulated time. His affectionate and kind consideration towards my research helped me to carry on with my project in odd circumstances. I m also very obliged to my committee members, Lt. Col. Dr. Adil Masood and Lt. Col. Usman Malik for their intimate help in fulfillment of this research work.

I am deeply indebted to my great parents, siblings and colleagues for their encouragement and affectionate selfless prayers. Most importantly I would like to thank my beloved wife for understanding the significance of this assignment and sharing the burden of domestic liabilities. She always infused motivation to my efforts and encouraged me whenever I was fatigued and tended to relax. Last but not the least, her invariable appreciation was a constant source of motivation for me.

LIST OF ACRONYMS

1.	Peak Signal to Noise Ratio	**PSNR**
2.	Zero Tree Root	**ZTR**
3.	Isolated Zero	**IZ**
4.	Compression Ratio	**CR**
5.	Discrete Fourier Transform	**DFT**
6.	Discrete Wavelet Transform	**DWT**
7.	Dominant Pass	**D**
8.	Embedded Zero-tree Wavelet	**EZW**
9.	Mean Square Error	**MSE**
10.	Set Partitioning In Hierarchical Trees	**SPIHT**
11.	One Dimensional	**1 D**
12.	Subordinate Pass	**S**
13.	Two Dimensional	**2 D**
14.	Successive Approximation Quantization	**SAQ**
15.	Fast Fourier Transform	**FFT**
16.	Inverse Fourier Transform	**IFT**
17.	Inverse Discrete Fourier Transform	**IDWT**
18.	Positive	**Pos**
19.	Negative	**Neg**

TABLE OF CONTENTS

OF FIGURES

LIST OF TABLES

Chapter-1

Introduction

1.1 Background

Nowadays use of images has increased many folds. This merits efficient compression so as to reduce storage and bandwidth requirements. During the last two to three decades, many wavelets based efficient image compression methods with progressive bit stream output have come up as an efficient solution [1]. With the emergence of Embedded Zero Tree Wavelet (EZW) Encoding algorithm, the wavelet based image compression has witnessed massive development. Search of improvements in EZW led to another better encoding technique "Set Partitioning in Hierarchical Tree" (SPIHT).

SPIHT encoding has yielded efficient compression performance than EZW. This has been done by making use of property of self similarity of the coefficients. The output of SPIHT once further fed to entropy coding, it paves the way towards better results. To this end, several entropy encoding schemes like Huffman, Arithmetic and Lempel Ziv Welch have remained prime considerations, both for the lossless and lossy compression, for the researcher.

Nevertheless, the researchers have proposed a variety of combinations to obtain efficient compression results yet the room for improvements still exists. There is a possibility that entropy coding, once used by utilizing its best possible way, may produce more efficient compression by saving extra number of bits than before without compromising on the quality of original image and its basic facet like *Peak Signal to Noise Ratio (PSNR).* The purpose of this research is directed to

work on image compression that uses set partitioning in hierarchical (SPIHT) along with entropy encoding with a view to bring some improvements in image storage capacity and transmission time by saving more number of bits.

1.2 Research motivation

Since versatile information expressed graphically through images is considered much easier to collate, disseminate, articulate and assimilate therefore, utilization of digital Images has become part and parcel of every walk of life. Albeit, invent of smart devices have made use of images more frequent than ever yet this has put some extra constraint on bandwidth and storage space because extra bits are needed to sustain the image quality. To this end, there is a dire need to tackle the above highlighted issues of bandwidth and storage space. Therefore, this research is motivated by the requirement of a viable solution for fast transmission and less storage space. This research concentrates on saving comparatively more number of bits without compromising the quality of the image by combining two encodings "Set Partitioning in Hierarchical Tree and Entropy coding".

1.3 Research Objectives The research work is expected to have following objectives:-

1) Study embedded codec image compression techniques based on discreet wavelet transformation (DWT).
2) Analysis of "Set Partitioning in Hierarchical Tree" and its variants in detail in order to sift out its various facets for apt utilization.

3) In depth analysis of entropy coding technique in order to make its better use to bring further improvements in image compression.

4) Make use of both encodings, SPIHT and Entropy, by combining together for efficient image compression by saving more number of bits.

5) Propose a better image compression model for enhanced storage and fast transmission.

1.4 Image compression

In present era, huge amount of information is stored, processed and transmitted digitally thereby necessitates that deft methods be adopted or devised to meet the requirements of storage space and address the limitations of bandwidth. Image compression is the reduction of size of digital image without compromising on the quality and is achieved by minimizing the number of bytes of an image file with compromising on the quality of an image to a bare minimum level.

1.5 Principle of Image Compression

An image is comprised of lot of pixels which are correlated with each other, which is why the neighboring pixels are very similar. Due to this correlation, only a small amount of redundant information can be get rid of because if the information from an image is removed without de-correlating it, there are chances that some of the important information is also lost as result thereby affecting the image quality. Therefore, there is a need that image be first de-correlated before subjecting it to the compression. Above in view, the digital images are first converted into statistically uncorrelated dataset before transmission and storage.

Original or approximated image can be regenerated after the decompression process. Image compression deals with the redundancy. It eradicates or reduces the redundant and irrelevant data which is duplicated in the image there by preserving the image quality to a level which is acceptable by human eye. Two of the commonly known types of redundancies are Statistical Redundancy and Psycho Visual Redundancy:-

1.5.1 Statistical Redundancy

It is further classified into two types

1.5.1.1 Inter-Pixel Redundancy

It has been observed that neighboring pixels have similar values. Inter-Pixel redundancy is further divided in to Spatial, Spectral and Temporal redundancies. Correlation or redundancy of neighboring pixels is dealt by spatial redundancy. Spectral is related to different bands and color plans where as Temporal deals with adjacent frames in a sequence of image. While compressing an image, much of reliance is made on removal of maximum of spatial and spectral redundancies.

1.5.1.2 Coding Redundancy

It is based on the principle that some pixel values are more common than others. It can also be related to the representation of information which has been expressed in the form of codes. Gray levels of an image are allocated more than the required number of code symbols which causes coding redundancy.

1.5.2 Psycho Visual Redundancy It has been observed after innumerable experiments that all visual information is not necessarily picked up completely by human eye with the same sensitivity rather it differentiates some information as more important than the other. Therefore, extraction of only such information which is considered important to the human eye of a particular user by eliminating the unimportant information, which is termed as psycho visual redundant, comes under Psycho Visual Redundancy ambit.

1.6 Image Compression Process

Image compression process works over three tires. Starting from de-correlation that is done by many ways few are transformation, predictive coding and sub-band coding. This is followed by quantization which is to reduce the precision and achieve the better compression ratios. Last but not the least is the entropy encoding for optimizing the compression results. Compression processes are lossy and losses depending upon the techniques being used. Hence in general it can be said that there are three components in compression concatenated closely as shown in the figure 1.1.

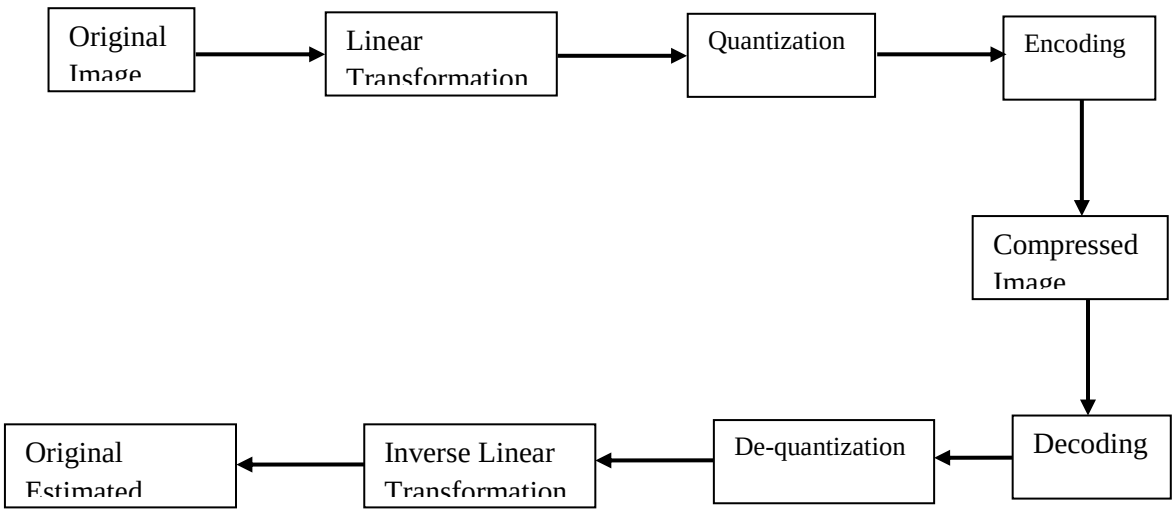


Figure 1.1: Compression and De-compression process

1.7 The Outline

This search work is comprised of seven chapters. Chapter 1 gives a prelude to the research incorporating background, motivation and contributions. Chapter 2 underlines need for image compression, its principles and processes. Chapter 3 describes about wavelet transform in general and DWT in particular. Chapter 4 highlights the EZW transform along with its advantages and shortcomings. Chapter 5 explains the concept as to how SPIHT has outperformed the EZW. Chapter 6 enunciates about entropy encoding and its apt utilization for optimal results that also includes the proposed methodology in detail and finally in chapter 7 the results along with future work have been presented.

Chapter-2

Wavelet Analysis and Transform

2.1 Introduction

This chapter will dilate upon wavelet, its analysis and wavelet transformation in general and Discrete Wavelet Transform, which forms the basis of present embedded encoding techniques, in particular. The facts as to how the wavelet has outperformed its predecessors and why DWT is being preferred over DCT or CWT as the basis of embedded codec will also come under discussion to establish the a coherent relationship with the upcoming chapters.

2.2 Wavelets

Wavelets are small wavelike mathematical functions of varying frequency and limited duration [2 & 3]. A single function f(x) that generates all these function is called the mother wavelet. Mother wavelet function is represented by the equation:-

$$f(x) = \sum_{k} \propto_k \varphi_{j,k(x)}$$

Where

$$\varphi_{j,k(x)=2^{\frac{j}{2}}\varphi(2^j x-k)}$$

Position of $\varphi_{j,k}$ (x) along the x-axis is defined by K whereas j tells about width of the function [4]. Higher frequency wavelet corresponds to narrow width and lower frequency corresponds to wider width[5].

2.3 Background

The signals nowadays are mostly time-domain signals in their raw format which once plotted, gives “Time Amplitude” representation. But then it is not the best method for presenting a signal as most of the distinct information is hidden in frequency contents. Since frequency is something that deals with the rate of change of that thing, therefore, if changes are rapid we say frequency is high and if these are slow then we say that frequency is low. Fourier Transform (FT) helps in measuring the frequency contents of a signal. If the Fourier Transform of a signal is taken in time domain, then representation of frequency amplitude of a signal is the outcome. It can be concluded that frequency information of a signal can be obtained by using Fourier Transform but it provides data about quantity of frequency in each signal and no information is rendered about at what time it existed.

Same time there is requirement when a signal needs information both in frequency and time domain for its apt utilization. Albeit, FT is reversible transform but only one representation i.e either frequency or time is present at a time therefore, could not meet desired requirements. However, with the introduction of wavelet transform representation of signal both in time and frequency became a reality.

2.4 Wavelet Transform

Data is converted in various frequency components with the help of wavelets. Out of these frequency components each one is representing to a specific resolution corresponding to the concerned scale [6]. The essential idea of this transform is to signify that any random function can be represented as a superposition of a basis function. Baby functions are obtained when mother wavelet like the one in figure 2.1 is scaled and shifted.

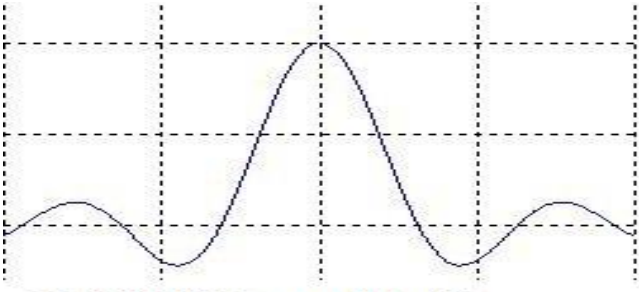
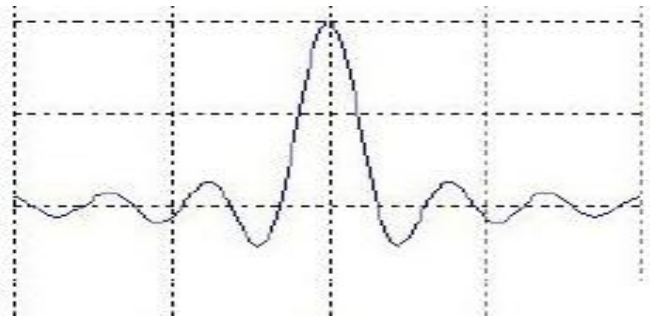
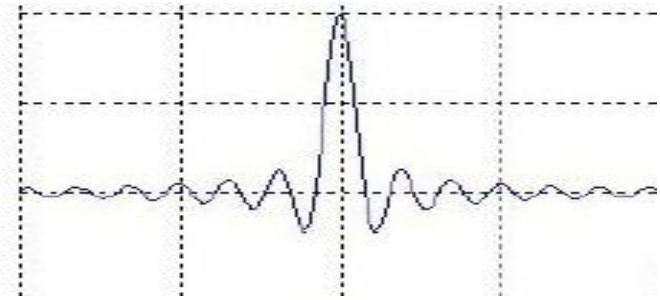


Figure-2.1: Mother wavelet w(t)

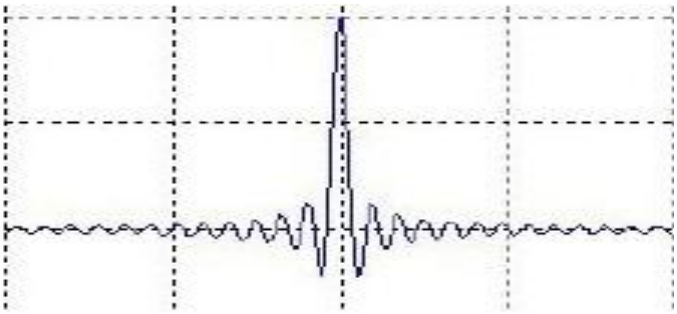
During the last two decades wavelet transform is being utilized in several fields of life like compression of an image, prediction of earthquake, turbulence and human vision. The analysis of wavelet creates a scale versus time plot of signal amicably. All functions of a wavelet w(2kt-m) can be obtained by utilizing a single mother wavelet w(t). Whereas the mother wavelet w(t) is a pulse or short interval wave that exists from time t=0 to t-T. Similarly a shifted wavelet that starts at t=m and terminates at t m+T is w (t-m) and w (2kT) is a scaled version obtained by starting mother wavelet at T=0 and terminating at t=T/2k as shown in figure 2.2. Wavelet gets narrower with the increase in the scaling factor. A wavelet which is wide in shape and size is equivalent to low frequency sinusoid of Fourier transform whereas narrow wavelets are the high frequency sinusoid of Fourier transform. Moreover wavelets with zero inner product are called orthogonal to each other.



(a)



(b)



(c)

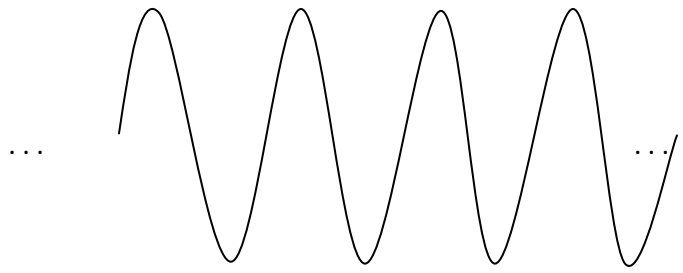
Figure-2.2: Scaling wavelet (a) k=1, (b) k=2 and (c) k=3

2.5 Wavelet Transform and Its Importance

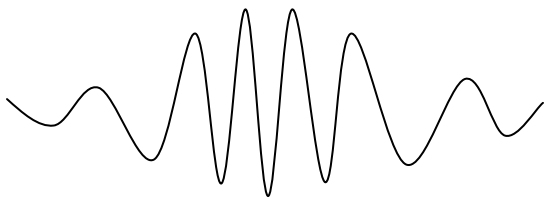
Nowadays wavelet transform has become very popular and is being preferred over other techniques in many areas. One of the main areas out of them is Data compression. In data compression, wavelet transform is given priority due to the reason of its ability to compress data and image at various resolutions levels [7] [8] [9]. Local analysis of larger signal can also be performed with the help of wavelets [3][8]. This characteristic of wavelets distinguishes it from the others. Moreover, wavelet coefficients also help in plotting the exact position of the discontinuity in time domain.

2.6 Preference over Fourier Transform

As discussed earlier Fourier transform can provided representation either in time or frequency domain at a time whereas wavelets transform can express the properties of a signal both in time and frequency domain simultaneously. The basis functions in Fourier transform are the sine waves that extends from –ve infinity to +ve infinity means there is no existence in a defined interval. As we know that sine waves are relatively predictable for being smooth as compared to the wavelets which are symmetric and irregular therefore, to analyze signals with sharp changes an irregular wavelet is a better option than a smooth sine wave. Same is evident from the figure 2.3.



(a)



(b)

Figure-2.3: Compression of Sine Wave and Wavelet (a) Sine Wave (b) Wavelet

2.7 Multi Resolution Analyses (MRA)

In multi resolution analysis, a series of approximations of a signal are produced with the help of a scaling function [10] and alteration in data between neighboring approximations are encoded with by using wavelet function. In equation 2.1 g(x) is a signal that can be studied as expansion function's linear combination

$$g(x) = \sum_{k} \alpha_k \phi_k(x) \qquad 2.1$$

Here α_k are called as real valued expansion-coefficients and $\phi_k(x)$ are called real valued expansion-functions. $\phi_k(x)$ are called basis functions if the expansion is unique. Here V as given under is the function space of expansion set $\{\phi_k(x)\}$

$$\mathrm{V} = span_k\{\phi_k(x)\} \qquad 2.2$$

And $g(x) \in V$ means that $g(x)$ is in the span of $\{\phi_k(x)\}$ and can be written in the form of Eq. 2.3. The coefficients α_k are computed by taking the inner products of the dual $\hat{\phi}_k(x)$'s and function $g(x)$. That is

$$\alpha_k = \langle \hat{\phi}_k(x), f(x) \rangle = \int \hat{\phi}_k^*(x)\, g(x)\, dx \qquad 2.3$$

If $\{\phi_k(x)\}$ is an orthonormal basis for *V*, then $\phi_k(x) = \hat{\phi}_k(x)$. If $\{\phi_k(x)\}$ are not orthonormal but an orthogonal basis for *V*, then the basis functions and their duals are called bi-orthogonal.

$$\langle \phi_j(x), \tilde{\phi}_k(x) \rangle = \delta_{jk} = \begin{cases} 0 & , j \neq k \\ 1 & , j = k \end{cases} \qquad 2.4$$

Now consider the set of expansion functions $\{\boldsymbol{\phi}_{\mathbf{j,k}}(\mathbf{x})\}$composed of integer translations and binary scalings of the real, square-integrable function $\boldsymbol{\phi}(\boldsymbol{x})$which is called a scaling function where

$$\phi_{j,k}(x) = 2^{j/2}\,\phi\,(2^j x - k) \tag{2.5}$$

for $k \in Z$ and $\phi(x)\epsilon\, L^2(R)$. Because the shape of $\phi_{j,k}(x)$ changes with *j*. We denote the subspace spanned over *k* for any *j* as

$$V_j = span_k\{\phi_{j,k}(x)\} \tag{2.6}$$

The scaling function has four fundamental requirements of multi-resolution analysis:-

1) The scaling function is orthogonal to its integer translates.
2) The subspaces spanned by the scaling function at low scales are nested inside those spanned at higher scales. That is

$$V_{-\infty} \subset \cdots \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \cdots \subset V_{\infty}$$

3) The only function that is common to all V_j is$f(x) = 0$. That is

$$V_{-\infty} = \{0\}$$

4) Any function can be represented with random precision. That is,

$$V_{\infty} = \{L^2(R)\}$$

The expansion functions of any subspace can be assembled from double-resolution copies of themselves. That is,

$$\phi(x) = \sum_k h_\phi(n)\sqrt{2}\,\phi(2x - n) \tag{2.7}$$

Where $h_{\emptyset}(n)$ coefficients are called as scaling function coefficients.

If we have a scaling function that qualifies the multi-resolution requirements, we can define a wavelet function $\psi(x)$that covers the difference between any two adjacent scaling subspaces, $V_j \; and \; V_{j+1}$. We can define the set$\{\psi_{j,k}(x)\}$of wavelets

$$\psi_{j,k}(x) = 2^{j/2}\,\psi\,(2^j x - k) \qquad 2.8$$

for all $k \in \mathbf{Z}$ that spans the space W_j where

$$W_j = span_k\,\psi_{j,k}\,(x) \qquad 2.9$$

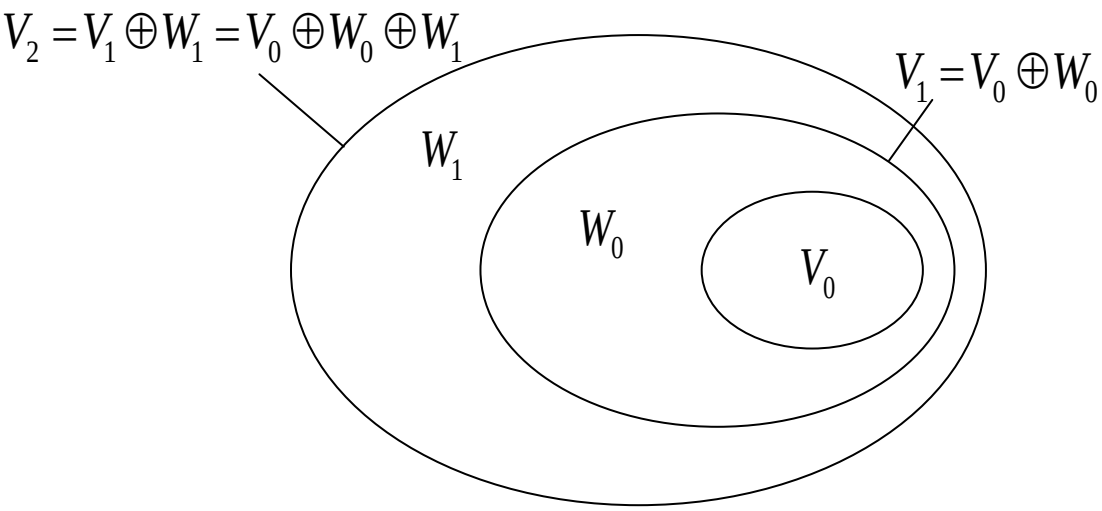


Figure-2.4: The relationship of scaling and wavelet function spaces

Scaling & wavelet functions subspaces as shown in figure 2.4 are linked by

$$V_{j+1} = V_j \oplus W_j \quad 2.10$$

Hence, space of all measurable, square-integrable function can be represented as

$$L^2(R) = V_0 \oplus W_0 \oplus W_1 \oplus W_2 \oplus \quad 2.11$$

Similar to the scaling function, the wavelet function can be stated as a weighted sum of shifted, double-resolution scaling functions. That is,

$$\psi(x) = \sum_n h_\psi(n)\sqrt{2}\,\phi(2x-n) \quad 2.12$$

Where $h_\psi(n)$ are called the wavelet function coefficients. It can be shown that $h_\psi(n)$ is related to $h_\phi(n)$by

$$h_\psi(n) = (-1)^n\, h_\phi(1-n) \quad 2.13$$

2.8 Types of Wavelet Transform

One of the most renowned time – frequency transform of the time is wavelet transform. For the analysis of frequency components in time domain wavelet functions are used on the same lines like sine and cosine waves are utilized in Fourier transform to carry out the analysis of a signal. Wavelet transforms can be discussed under following :-

2.8.1 Continuous Wavelet Transform (CWT)

The continuous wavelet transform is the natural extension of discrete transform. It transforms a continuous function in to a much redundant function of

two continuous variables which are scale and translation. The CWT offers the description of a signal in time and frequency domain which is kind of redundant but finely detailed. Problems related to signal identification and detection of concealed transients (difficult to detect the short lived elements of signal) are specifically treated with the help of CWT[11]. Fourier analysis in Fourier transform is mathematically expressed as:-

$$F(w) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t}\, dt \tag{2.13}$$

The exponential $''-j\omega t''$ is the superposition of real and imaginary components of the sinusoids and here the CWT is mathematically defined as under

$$C\,(\text{Scale}\,,\text{position}) = \int_{-\infty}^{\infty} f(t)\psi\,(\text{Scale}\,,\text{position}\,,t)\,dt \tag{2.14}$$

The result obtained will be CWT when a signal is multiplied with by a wavelet. Wavelet is used to define the basis functions of the wavelet transform.

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left[\frac{t-b}{a}\right];\ a, b \epsilon R^1 \ \text{ and } \ a > 0 \tag{2.15}$$

In the above equation 'a' is scaling factor which is used to select the width of a basis function and 'b' is used for the translation of the wavelet in the time domain. So the continuous wavelet transform can be defined as:

$$W_f(a,b) = \int_{-\infty}^{\infty} x(t)\,\psi_{a,b}(t)dt \tag{2.16}$$

2.8.1.1 The inverse Wavelet Transform

Inverse wavelet transform can be mathematically defined as:

$$x(t) = \frac{1}{C}\int_0^{\infty}\int_{-\infty}^{\infty} W_f(a,b)\,\psi_{a,b}(t)db\frac{da}{a^2} \tag{2.17}$$

$$\text{Here } C = \int_{-\infty}^{\infty}\frac{|\psi|^2}{\omega}d\omega < \infty \tag{2.18}$$

The Fourier transform of the mother wavelet ψ(t) results in the form of X(t) and it must fulfill two conditions. Number one, to stay away from singularity condition in the integral, C should be finite with X(t) to have zero mean. This is called the admissibility condition and can be expressed mathematically as follows

$$\int_{-\infty}^{\infty} \psi(t)dt = 0 \tag{2.19}$$

Number two, the mother wavelet must have finite energy.

$$\int_{-\infty}^{\infty} |\psi(t)|^2 dt = \infty \tag{2.20}$$

2.8.2 DWT Decomposition

In Fourier analysis, sinusoidal basis functions of different frequencies are obtained once Discrete Fourier Transform (DFT) decomposes a signal. This is a lossless transformation in which original signal can be completely recovered from its DFT representation. However, in case of wavelet analysis, the DWT decomposes a signal into a set of mutually orthogonal wavelet basis functions. These functions don't match with the sinusoidal basis functions. Moreover, wavelet functions are translated, scaled and dilated versions of mother wavelet φ. DWT is invertible like that of Fourier analysis, so that the original signal can be completely recovered. Haar wavelets and Daubechies set of wavelets are two of the most popular wavelets. The common properties of the two describe that wavelet functions are spatially localized, scaled, dilated and translated version of the mother wavelet. Moreover, each set of wavelet function makes an orthogonal set of basis functions.

2.8.2.1 DWT in One Dimensional

One-dimensional DWT is a multi-resolution frequency decomposition and localization of a one-dimensional, discrete-time signal [11].

Analysis equation of orthogonal DWT for a signal that belongs to $L^2(\mathcal{R})$ is defined as:-

$$a_{j,k} = \int x(t) 2^{j/2} \phi(2^j t - k) dt b_{j,k} = \int x(t) 2^{j/2} \psi(2^j t - k) dt \quad (2.21)$$

Similarly for any signal that belongs to $L^2(\mathcal{R})$ the synthesis equation of the orthogonal inverse can be mathematically defined as:

$$x(t) = 2^{N/2} \sum_k a_{N,k} \phi\,(2^N t - k) + \sum_{j=N}^{M-1} 2^{j/2} \sum_k b_{j,k} \psi\,(2^j t - k) \quad (2.22)$$

Here $\phi(t)$ is the scaling function which is orthogonal and $a_{j,k}$is to express the scaling. Orthogonal wavelet function has been expressed as $\psi(t)$ and $b_{j,k}$ is used to express wavelet coefficients. The analysis equation of bi-orthogonal DWT for a signal that belongs to $L^2(\mathcal{R})$ is written as

$$\tilde{a}_{j,k} = \int x(t) 2^{j/2} \tilde{\phi}(2^j t - k) dt \tilde{b}_{j,k} = \int x(t) 2^{j/2} \tilde{\psi}(2^j t - k) dt \quad (2.23)$$

For a signal that belongs to $L^2(\mathcal{R})$ the the synthesis equation of bi-orthogonal IDWT is defined as :

$$x(t) = 2^{N/2} \sum_k \tilde{a}_{N,k} \phi\,(2^N t - k) + \sum_{j=N}^{M-1} 2^{j/2} \sum_k \tilde{b}_{j,k} \psi\,(2^j t - k) \quad (2.24)$$

2.8.2.2 Two Dimensional DWT

It is of a great utilization for the processing of images and applications related to computer vision. One can say that it is a kind of straight forward extension of the one dimensional discrete wavelet transform.. It can be implemented by utilizing down-samplers and digital filters.[12]. It goes without saying that 2-D separable wavelet transform is made when two of 1-D wavelet transforms are connected in series. The data is passed through the rows and then through the columns of the 1-D wavelet transform. A perfect 2-D DWT followed by IDWT reconstruction filter bank has been expressed in figure 2.5.

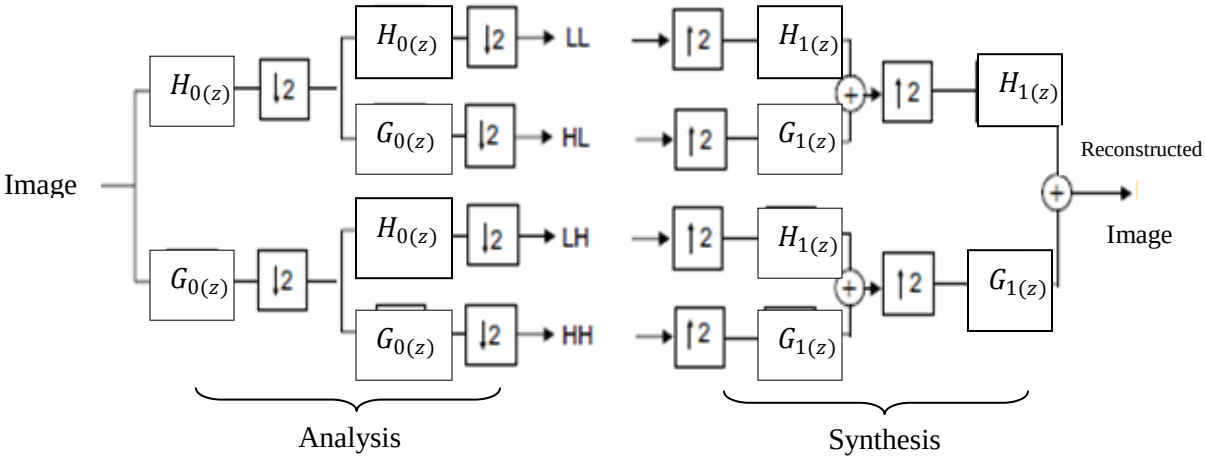


Figure-2.5: One level filter bank for computation of 2-D DWT

Projection of the image into basis of 2-D will result into transform coefficients. When two 1-D basis are multiplied the consequence is a 2-D separable basis function. Therefore for the images, we have 4xbasis functions which have been represented in the equations from 2.25 to 2.28.

$$\phi(u,v) = \phi(u)\,\phi(v) \quad (2.25)$$

$$\psi_1(u,v) = \psi(u)\,\phi(v) \quad (2.26)$$

$$\psi_2(u,v) = \phi(u)\,\psi(v) \quad (2.27)$$

$$\psi_3(u,v) = \psi(u)\,\psi(v) \quad (2.28)$$

$\phi(u,v)$ is the scaling function of the images whereas $\psi_1(u,v), \psi_2(u,v)$ and $\psi_3(u,v)$ represents the wavelet functions. When an image is projected into these basis functions, the coefficients are achieved after transformation and original image is decomposed into four sub-bands as under (Figure 2.6):-

- LL Sub-bands (Approximations)
- LH Sub-bands (Vertical Details)
- HL Sub-bands (Horizontal Details)
- HH Sub-bands (Diagonal Details)

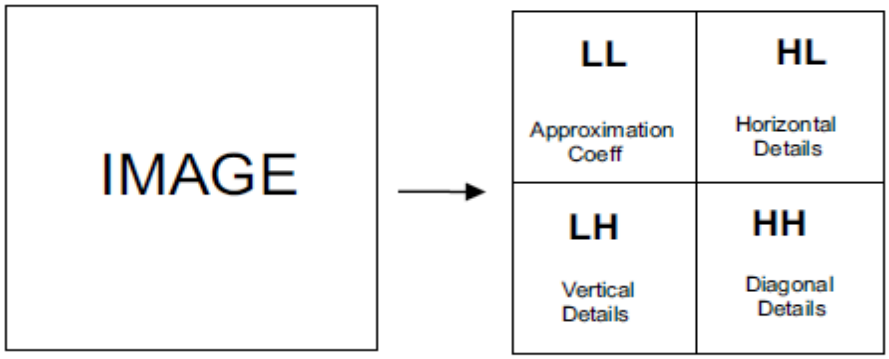


Figure-2.6: Output of 2-D Decomposition up to one level

Quantization
By
Progressive Encoding

Chapter-3

Embedded Zero-Tree Wavelet (EZW) Transform

3.1 Introduction

EZW is a coding algorithm which performs compression to a good extent with many types of images. The E of EZW is for embedded thereby depicting that it is a progressive coding. Z stands to represent data structure of Zero-trees which encodes the data and W for wavelet transform on which EZW encoders works on. In this chapter, EZW encoding has been discussed in detail.

3.2 Embedded Encoding

The embedded coding is defined by the fact that the order of the coded bits is set in accordance with their significance and lower code rates are adjusted at the start of the bit stream. To achieve the intended bit rate specified by channel, progressive encoding is capable of terminating encoding process at any stage. This is done by maintaining the bit calculation and truncating stream of bit by encoder, whenever the set bit rate is attained. Albeit, EZW uses more simple and state of the art progressive coding, yet we can compare it with one, where most significant bit plan is the starting point for the coding and gradually carries on with the most significant bit plan coming next and so forth. Reconstruction error at receiver will occur, if before addition of the less significant bit plan to bit stream we meet the target, reconstruction error is reduced at given target bit rate with the help of " significant ordering" of the embedded bit stream.

Accordingly, compression algorithm which generates embedded code, first thing that should be sent on the transmission network is the coarser version of the

image followed by refinement details within the framework of progressive broadcast.

The diagram of an embedded image coding system is as under:

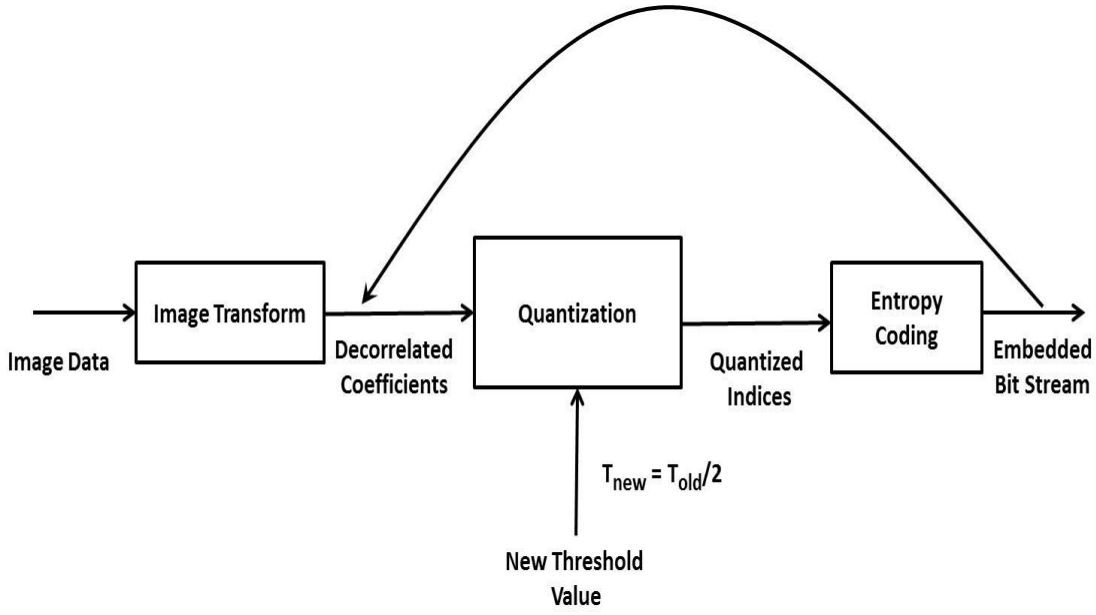


Figure-3.1: Embedded Coding Scheme

3.3 Zero-Tree Structure

Now let us see what is a zero tree, a quad-tree is termed as Zero-tree when a root node in a tree structure is greater than or equal to the other nodes but smaller than the given threshold to compare the wavelets against it.

A quad-tree with all the nodes smaller or equal to the root is termed as zero- tree. A single symbol is used to code it and the decoder reconstructs it as quad-tree which is filled with the zeros. A root smaller than the threshold, for which the coefficients are presently being measure, is required to clutter this definition. [13].

The encoder for EZW works on two basic observations, number one is that all natural images are low pass spectrum images. If passed through the wavelet transform energy in sub-bands decrease with the decrease in the scale (low scale means high resolution). It means that progressive encoding appear to be the best choice as higher bands add the details only. Number two is that the larger wavelet coefficients are given more importance over those of smaller ones..

3.4 EZW Encoding Process

The EZW algorithm results in a fully progressive bit streams for image coding [14] and the compression enactment of this technique is comparatively better than previously known methods. The EZW process is based on following major theories:

1) Hierarchical sub-band breakdown.
2) Zero-tree coding.
3) Entropy coded successive-approximation quantization.
4) A prioritization technique to define importance of coefficients basing on various characteristics [15].
5) Lossless data compression through entropy coding schemes.

In this algorithm encoding becomes the most important part. The test image undergoes the filters for DWT which yields the transform coefficients. This transforms results into de-correlated coefficients with as fewer dependencies among the samples as possible. Then the symbols are produced by quantizing these transformed coefficients for compression process. Here the embedded coding is done by using successive approximation quantization. It has been seen that it is the quantization phase where most of the information is lost [16]. Resultantly, to find the significance in quantization stage the threshold value is

fixed. In the final stage of encoding, the bit stream of symbols is sent for compression. At the decoder end, reverse process as enunciated at the encoder is applied. EZW algorithm has a privilege [17] that user can select bit rate as per his desire and encode the image according to that. Figure 3.2 explains the details as that how EZW coding algorithm is applied.

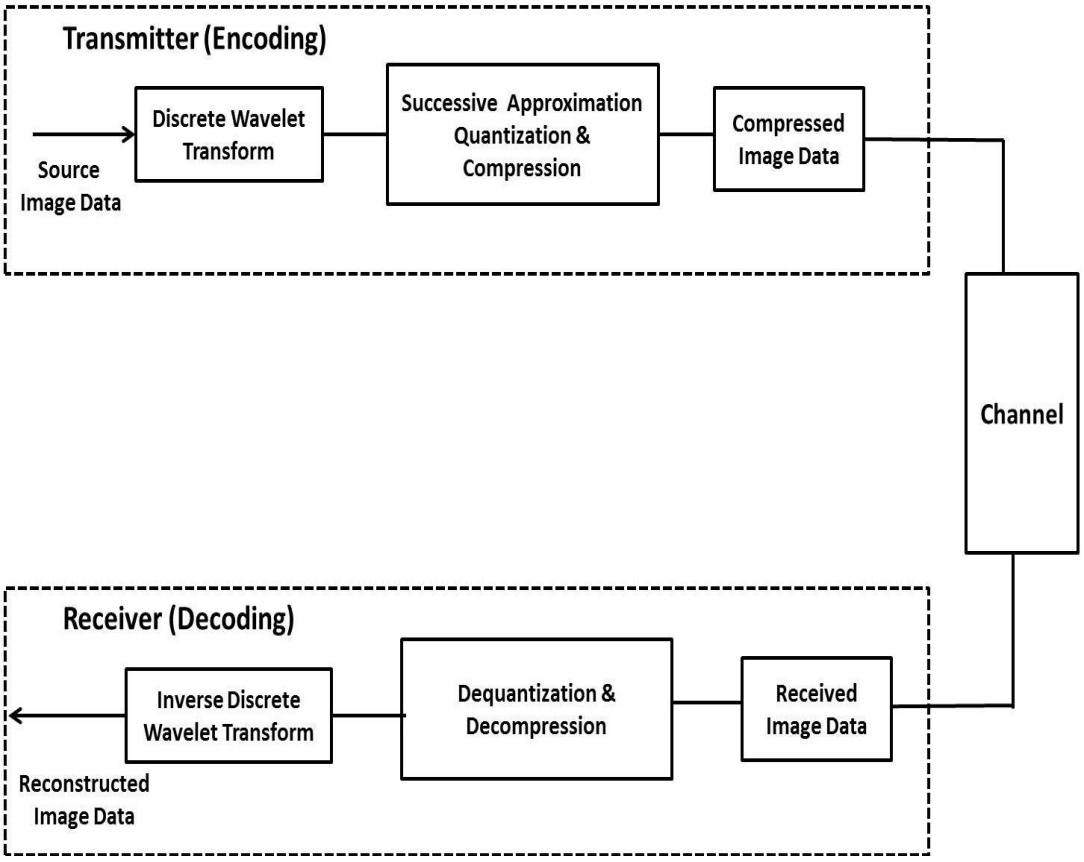


Figure 3.2: EZW Compression Diagram

In EZW image compression algorithm, some of the information is lost due to residual matrix left at transmitter end. It is because that the real images are made up of mostly low frequency information which is highly correlated. Moreover, the importance of high frequency information (such as edges) cannot be overlooked at the same time as it is of significance due to human perception of the image quality. Therefore, in high quality coding scheme it is necessary to precisely

signify the high frequency components. At root node the transformed coefficients can be measured as the tree or the trees with the lowest frequency components and with the children of every tree node being the spatially related coefficients in the next higher frequency band. It has been observed with great probability that one or more sub-trees will have zero or nearly zero coefficients [18].

In EZW, the statistical properties of the trees are used to proficiently code the locations of the substantial coefficients [19]. Since most of the coefficients will either be zero or approximately zero, the spatial locations of the significant coefficients consist of a large portion of the entire size of a compressed image. The significance of a coefficient is determined by comparing its modulus value or value of a node and its children in the case of a tree above with a specific threshold. Initially, the threshold is taken by considering the magnitude of maximum coefficient and then by iteratively lowering the threshold as per algorithm. Four kinds of different symbols [14] are used by the EZW to represent the wavelet coefficients:-

- Zero-Tree Root (ZTR)
- Isolated Zero (IZ)
- Significant Positive (Pos)
- Significant Negative (Neg)

Two binary bits are used by EZW to represent the above mentioned symbols. After every dominant pass, the existing threshold is updated by a factor of two. By scanning the trees and emitting one of the four symbols, the dominant pass encodes the significant coefficients which have not yet been found significant in previous iterations. Conditions for scanning of the children of a coefficient is

either should be significant or an isolated zero. One bit (MSB) is produced by the subordinate pass for each coefficient found significant in the previous significance passes.

3.5 Encoding Concept of EZW

3.5.1 Progressive Coding

In EZW algorithm, bits emerge as per their importance order in a bit stream. It is due to this feature that the beginning of the bit streams contains all the low rate codes [20]. This progressive code signifies a structure of binary conclusions that differentiate an image from the null or all gray image. The encoding stops when the user set target rate or distortion metric is reached [15]. This implies that the embedded coder can terminate coding on user set parameter [21] and offers the best representation of the image.

Stream which is binary coded, can comprehend progressive broadcast by utilizing multi- threshold EZW coding, as a result, coding rate / distortion metric can be restricted accurately. The coding process can be terminated either when bit budget is consumed [22] or compression ratio is reached. So at any given rate of coding, the coefficients required to represent an image will always contain the required information that was required at much lesser rates. Therefore, this may be done by choosing a target bit rate which fixed and decoder retains the option of terminating the decoding process at any point of time. So it is concluded that the decoder has the capability to interfere [23] the process of decoding at any time in the bit stream and still has the ability to reconstruct the image. For that reason, the compression technique which is progressive in nature sends the low frequency

information at the start. This is followed by the high frequency components i.e details within the framework of progressive broadcast.

3.5.2 Significance Map Encoding

It is has been observed in EZW scheme that a reasonable amount of the overall bits strength is needed for the coding of the position information [24]. Therefore, a significance map can be defined as a binary function the value of which tells whether a coefficient is significant or insignificant. A coefficient is quantized to zero if it not significant. So it is known to a decoder that no further information is needed by the significance map about that coefficient. However, non-zero value is assigned to a significant coefficient. It is achieved by encoding the location of the zeros [14]. After lot of statistical analysis it has been experimentally proved that in the wavelet transform, across different scales zeros can be forecasted accurately. Assumption on which EZW is based stats if at coarse scale, a wavelet coefficient is insignificant with respect to a given scale then at same spatial location at finer scales all the coefficients of same orientation are also likely to be insignificant [15][18]. In the significance map, the location of significant and insignificant wavelet coefficients signified specifically for every threshold T. Zeros are used to specify the positions of insignificant coefficients and locations of the significant coefficients are represented by values of "one" [18]. A Zero-tree Representation is used to code the significance map. It permits that insignificant coefficients are identified and predicted accurately across the scales. With this technique, total cost of encoding the significance maps is reduced by grouping the insignificant coefficients in exponentially growing trees across the

scales, and by encoding these coefficients with zero-tree symbols [15]. Samples of a significance map and quantized coefficients are presented in the table 3.1:

Quantized Coefficients							
64	56	48	32	24	16	0	0
56	28	40	24	16	23	0	0
40	40	30	24	16	8	0	8
32	32	32	24	24	16	0	0
24	24	16	8	0	0	8	0
16	16	8	0	0	8	0	0
0	0	0	8	0	0	0	0
0	0	0	0	0	0	0	0

Significance Map							
1	1	1	1	1	1	0	0
1	1	1	1	1	1	0	0
1	1	1	1	1	1	0	1
1	1	1	1	1	1	0	0
1	1	1	1	0	0	1	0
1	1	1	0	0	1	0	0
0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0

Table 3.1: Quantized Coefficients & Significance Map

The significance map is about location of location of non zero valued coefficient which are going to be transmitted and the coding of the significance is one of the important views of low bit rate image coding. After quantization followed by entropy coding, the zero symbol which occurs with the highest probability, should be extremely high in order to attain very low bit rates. Therefore, in this way a large portion of the bit budget is utilized in encoding the significance map. Due to this not only efficient encoding the significance map is achieved but also it offers a higher efficiency in compression.

To understand the significance of map coding in a better way, let us consider a encoding system with a typical transform method. There are three basic parts of a typical low bit-rate image coder which are shown in the figure 3.3 below [25].

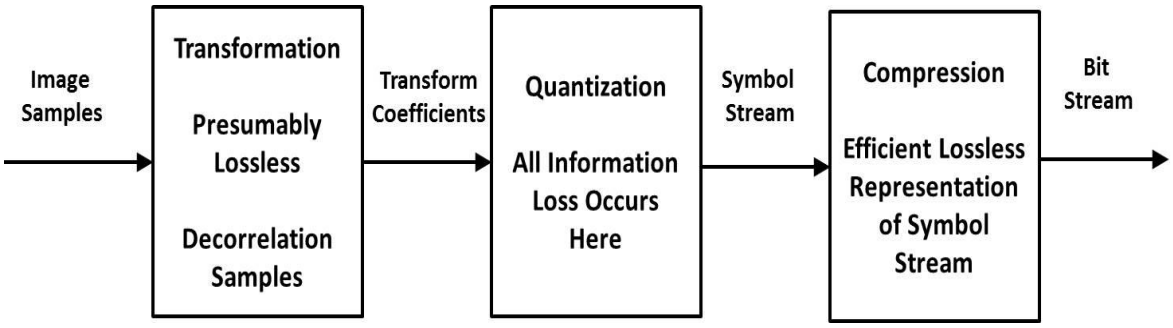


Figure 3.3: Low bit rate image coder

3.6 The Successive Approximation Quantization (SAQ)

The SAQ [26] provides a vital facet to embedded compression algorithms. It goes without saying that symbol stream produced by these algorithms consists of the bit streams for all possible lower rates. An embedded code is characterized by J.M Shapiro by defining two properties [15]:

1) While same data is encoded at different rates, for the level of smaller one the two resulting images must match exactly. This represents that for a given data rate a coded symbol consists of all the symbols for smaller data rates. As we add more symbols to them, the representations get more precise.

2) For a given data rate there should be a good representation.

SAQ is applied to perform the progressive / embedded encoding which is linked to the bit-plan encoding of the magnitudes [27]. The SAQ applies a series of thresholds sequentially to check the significance of given data. The initial threshold T_o is selected in such a way that $|Xj| < 2T_o$ for all transform coefficients. After this the subsequent thresholds are selected as Ti = Ti-1/2. Two separate lists of wavelet coefficients are present in the process of encoding and decoding. The coordinates of the coefficients, not found significant so far in the same order

during the process of initial scan, are part of a list known as “Dominant list”. The sub-bands are ordered in this scan, and with each sub-band, the set of coefficients are well-arranged. Absolute value of significant coefficients is stored in the subordinate list. For each threshold the list is scanned once.

To govern their significance, the coefficients are compared to the threshold T_o during the dominant pass. In the next step, if a coefficient is proved to be significant then its sign is ascertained as positive or negative. This significance map is coded as zero-tree later. A coefficient is coded as Pos if it is found to be significant then each time its absolute value is attached to a list called subordinate list. The coefficient in the wavelet transform range is set to zero to prevent the occurrence of significant coefficient as a zero-tree on future dominant passes at smaller thresholds. All those values which were previously found significant will now be subjected to the subordinate pass. Binary ‘1’ is used to code the refinement for every absolute value in the subordinate list. This points out that the old uncertainty interval contains the true value in the upper half. However, ‘0’ symbol shows that the value is in the lower half of the old uncertainty interval. The entropy encoding is done to the string of the symbols that is generated from this from this binary alphabet during a subordinate pass. The magnitudes on the subordinate list are sorted in decreasing magnitude after the completion of a subordinate pass. The process keeps alternating between subordinate passes and dominant passes and the threshold is halved before each dominant pass.

3.7 The EZW Algorithm

The coefficients’ magnitude is compared by the encoder with the initially selected threshold. Encoder sends the signal to the decoder whether the magnitude

is smaller or greater than the given threshold. For nearly precise results at the decoder end, encoder must also send the information about threshold value. The process is repeated till we find the smallest coefficient (desired to be sent) gets larger than the last computed threshold to obtain perfect reconstruction.

If the both encoder and decoder use a predefined criterion for the threshold instead of transmitting the threshold in each pass, then bandwidth can be saved as there will be no requirement of sending the threshold to the decoder. The threshold represents the number of bits to explain the binary value of the coefficients magnitude if the predefined criterion for the threshold is a serious of the powers of two [15]. As per Shapiro it is known as " bit plane coding".

The decoder needs information about the position of the coefficients to reconstruct the transmitted signal. Efficient encoders are differentiated from the inefficient ones with the help of coding of the positions.

EZW encoder uses a predetermined sequence of scanning for encoding the spatial position of the coefficients as shown in the figure 3.4. Using zero-trees a lot of positions are coded perfectly. Many orders scan can be used by the encoder [28], till it scans the coefficients of the lower sub-bands before scanning the coefficients of higher sub-bands.

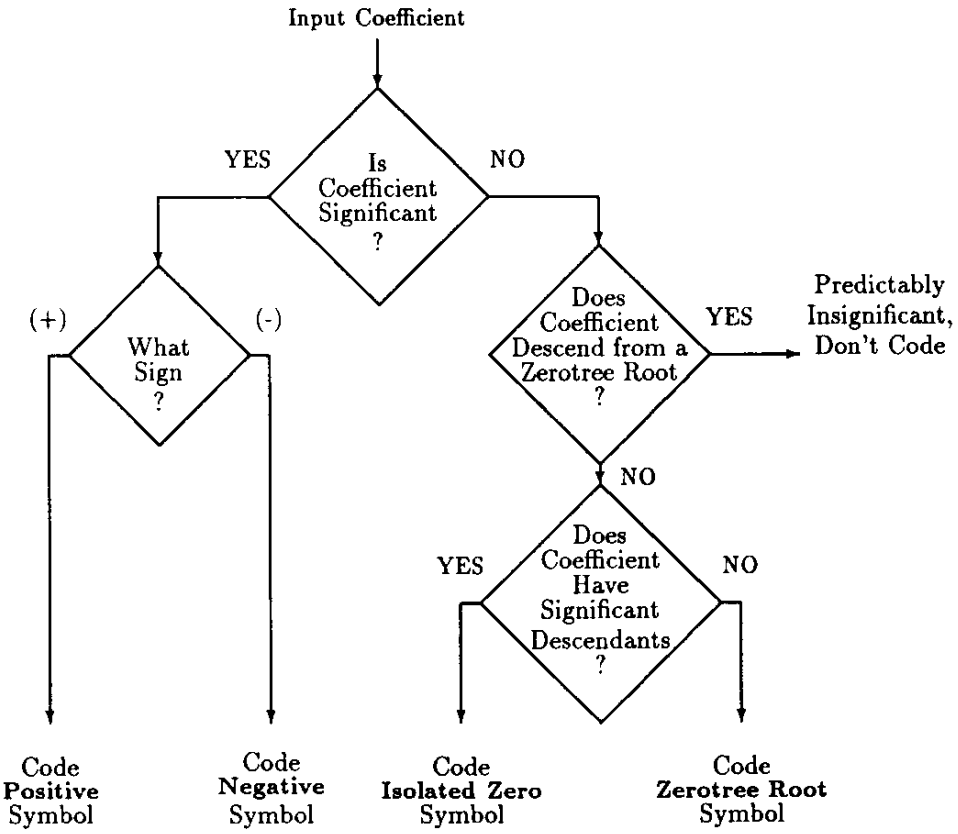


Figure 3.4: Flow chart to encode a significance map coefficient

The final results of compression are also affected by the order of scan. Initial threshold t_0 be calculated by adopting bit plane coding by using the relation as under

$$t_0 = 2^{\lfloor \log_2 (\mathrm{MAX}\ (|\gamma\ (x,y)|)) \rfloor} \tag{3.1}$$

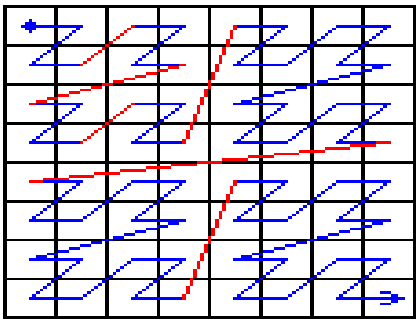
Where MAX (.) indicates the highest value of the wavelet coefficient $\gamma(x, y)$

3.8 Example

To illustrate the above stated algorithm we take an example shown in Table 3.2 and figure 3.5.

63	-34	49	10	7	13	-12	7
-31	23	14	-13	3	4	6	-1
15	14	3	-12	5	-7	3	9
-9	-7	-14	8	4	-2	3	2
-5	9	-1	47	4	6	-2	2
3	0	-3	2	3	-2	0	4
2	-3	6	-4	3	6	3	6
5	11	5	6	0	3	-4	4

(a)



(b)

Table 3.2: (a) data set **Figure 3.5:(b) scanning order (Morton scan)**

Here the example is being explained after taking the data from table 3.2 and running the Morton scan in figure 3.5 [15]. With initial threshold calculated as t_0=32 the EZW algorithm generates following bit stream after one pass.

D1: pnztpttttztttttttptt

S1: 1010

D2: ztnpttttttttt

S2: 100110

D3: zzzzzppnppnttnnptpttnttttttttptttptttttttttptttttttttttt

S3: 10011101111011011000

D4: zzzzzzztztznzzzzpttptpptpnptntttttptpnpppptttttptptttpnp

S4: 11011111011001000001110110100010010101100

D5: zzzzztzzzzztpzzzttptttttnptpptttptttnppntttttpnnpttptttpptttt

S5: 10111100110100010111110101101100100000000110110110011000111

D6: zzzttztttztttttnnttt

With the consideration in mind that EZW algorithm will require initially (at least) two bits for coding of the symbols in the alphabet { Pos, Neg, ZT, IZ} and another bit to code the symbol Z, therefore, a total of 33=26+7 bits were utilized after the first pass.

Set Partitioning in Hierarchical Trees (SPHT)

Chapter-4

SPIHT (Set Partitioning in Hierarchical Trees)

4.1 Introduction

Albeit, EZW was an efficient and computationally simple technique yet it had room for improvements which were done by Amir Said and A. Pearlman in 1996 [32]. Their technique works the concept of ordering which is partial by the magnitude having an algorithm of partitioning sorting, bit planes have ordered transmission and self similarity is being exploited across different scales of an image transform. SPIHT is a modified EZW algorithm proposed by Amir Said and Pearlman. The transmission of the ordered coefficients, and sub-bands of equivalent orientation having self-similarity, which were considered as the best features of EZW algorithm, were also incorporated in SPIHT. SPIHT yields high PSNR than EZW because of a special symbol that indicates the significance of child nodes of significance parent, and separation of child nodes from second generation descendants [29][30][31].

SPIHT algorithm outperforms the performance of its predecessor. SPIHT bit stream is in possession of a distinct property of compactness. It is due to this characteristic that only marginal gain is obtained once the output bit stream of SPIHT is further passed through the entropy encoding schemes.

Unlike EZW no ordering information is clearly transmitted to the decoder and this becomes another signature of SPIHT algorithm. Moreover, the execution path of the encoder is reproduced by the decoder and the ordering information is recovered. But then for smooth execution or recovery of the actual information the there is need that both the encoder and decoder posses the same execution time.

4.2 Progressive Transmission Scheme

To use partial ordering, a comparison of the magnitudes of the coefficients is drawn with a set of octave decreasing threshold [32]. The hierarchical sub-band transformation can be expressed like wavelet as under

$$c = \Omega(s) \tag{4.1}$$

Here c is the output array of the transformed coefficient, which are produced once Ω sub-band transformation is applied on the original image array s. Both the output and the original image have same dimensions of the coefficient array. Encoder and decoder process the coefficients as per the defined SPIHT algorithm. To reform an estimated image $\hat{s}$ we need to take the inverse transformation from the estimated array of coefficients $\hat{c}$ as under

$$\hat{s} = \Omega\,(\hat{c}) \tag{4.2}$$

For reconstruction of the estimated image at the decoder end, the mean-squared error is calculated with the help of following

$$D_{mse}\,(s - \hat{s}) = \frac{\|s-\hat{s}\|^2}{N} \tag{4.3}$$

$$\frac{\|s-\hat{s}\|^2}{N} = \frac{1}{N}\sum_{n_1}\sum_{n_2}(s_{n_1,n_2} - \hat{s}_{n_1,n_2})^2 \tag{4.4}$$

Where s_{n_1,n_2} is the intensity value of the pixel at location n_1, n_2 of image having N number of pixels. Mean square error (MSE) is independent because the sub-band transformation is lossless.

$$D_{mse}\,(s - \hat{s}) = D_{mse}\,(c - \hat{c}) \tag{4.5}$$

$$D_{mse}\,(c - \hat{c}) = \frac{1}{N}\sum_{n_1}\sum_{n_2}(c_{n_1,n_2} - \hat{c}_{n_1,n_2})^2 \tag{4.6}$$

In the above equation, c_{n_1,n_2} represents the coefficient of transformation at the position n_1, n_2. For every coefficient, $\hat{c}_{n_1,n_2}$ is set to zero by the decoder. If c_{n_1,n_2} is the coefficient value sent by the encoder, then mean square error is decreased by $\frac{(c_{n_1,n_2})^2}{N}$. This indicates that the coefficients with the larger magnitude have high significance in embedded bit stream encoding than those of smaller ones. It is because they play an important role in reducing the mean square error and thereby producing better reconstruction as compared to the smaller valued coefficients. Therefore, the coefficients are arranged with respect to their magnitudes in the embedded stream coding. Whereas in EZW algorithm we arrange significant coefficients in the subordinate pass. It is possible to broaden the concept of arranging the coefficients to the bit-planes once we order the coefficients in accordance with their binary representation. This can be expressed in general form as

$$\left|\log_2|c_n(k)|\right| \geq \left|\log_2|c_n(k+1)|\right| \quad k = 1,2,\ldots\ldots,N \tag{4.7}$$

In equation 4.20 $c_n(k)$ represents the coefficients which have been ordered according to their magnitude values.

Following example will enable us to assimilate ordering concept in a better way. Here we have array of coefficients: -3, -9, 16, 5, -57, 8, 38, 2, -12, 14, -17, -6, 25, and -7. The array of coefficients can be arranged by using above ordering equation as follows (Table 4.1):

Coefficient magnitude	57	38	25	16	17	14	12	9	8	7	6	5	3	2
Sign bit	1	0	0	0	1	0	1	1	0	1	1	0	1	0
Bit-5 (msb)	1	1	0	0	0	0	0	0	0	0	0	0	0	0
Bit-4	1	0	1	1	1	0	0	0	0	0	0	0	0	0
Bit-3	1	0	1	0	0	1	1	1	1	0	0	0	0	0
Bit-2	0	1	0	0	0	1	1	0	0	1	1	1	0	0
Bit-1	0	1	0	0	0	1	0	0	0	1	1	0	1	1
Bit-0 (lsb)	1	0	1	0	1	0	0	1	0	1	0	1	1	0

Table 4.1: Order of coefficients in binary representation

4.3 Set Partitioning Sorting Technique

To send the coefficients explicitly there is no need of ordering information for the Set partitioning technique. However, both decoder and encoder follow the same implementation path. If the encoder transmits the magnitude comparison results then the execution path helps the decoder to recover the sorting information.

So it can be said that the set partitioning is devoid of explicit ordering of the coefficients rather the coefficient values are observed for a given n, if they follow the inequality $2^n \leq |c_{n_1,n_2}| < 2^{n+1}$. A significant coefficient must follow the inequality $|c_{n_1,n_2}| > 2^n$ and if it does not follow this inequality then it is an insignificant coefficient. The coefficient subset T_n is examined if

$$\max_{n_1,n_2 \in T_m} |c_{n_1,n_2}| \geq 2^n \qquad (4.8)$$

Same holds well for the subset T_n, it stands significant or insignificant if it satisfies or does not satisfy the inequality respectively. We classify significant and insignificant subsets by portioning T_n if it is significant. So until we come across a single significant coefficient we continue with the partitioning of significant subsets into significant and insignificant coefficients. A sub-band hierarchical framework is followed in set partitioning technique.

4.4 Spatial Orientation Tree

It has been observed that low frequency components of an image contain most of the energy of an image. Therefore, as we move from highest to lowest levels of sub-band pyramid there is a decrease in the value of the variance [32]. Moreover, spatial similarity between the sub-bands can also be observed as sub-bands are linked spatially with each other. In the spatial orientation tree, the connection of the sub-bands is presented in figure 4.1 in the form recursively split four bands. The coordinates of a pixel are made use of to represent the associated node. The offspring, which are four in number for every node, are represented with likewise pixel location in the orientation pyramid of next lower level as explained with the help of arrows in the diagram below. The LL sub-band which resides at the highest level of the pyramid is exempted and does not hold any such relationship. It is the pixel in sub-band which forms the root and composes adjacent 2x2 pixels' group. Out of the four pixels of LL band three has offspring in HH, LH and HL sub-bands which are exiting in the same scale since only three sub-bands which decides the descendants. Whereas one pixels in LL band which is marked with '*', as displayed in figure 4.1 does not determines any descendants.

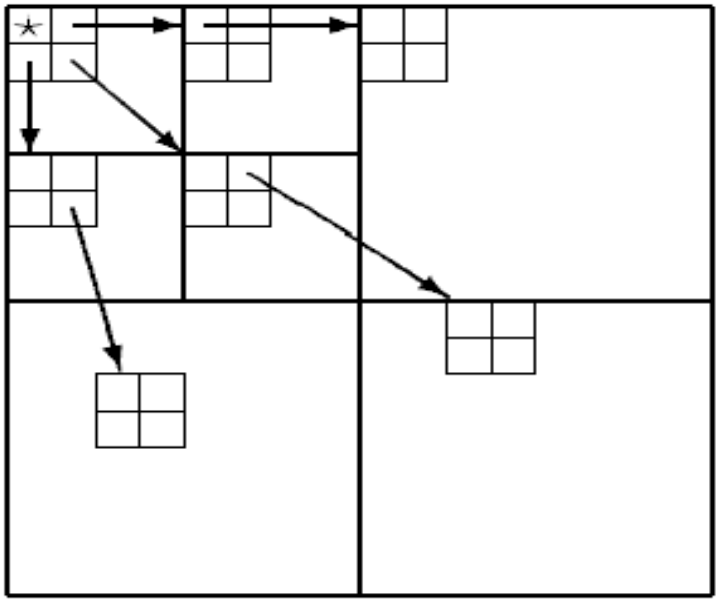


Figure 4.1: Spatial orientation tree

4.5 Set Partitioning Rules and Algorithm

For better assimilation of the concept, few important sets of notation which are utilized in SPIHT algorithm need to be understood well. It is also underlined that some of the SPIHT algorithm rules are also derived from these notations.

1) $\mathbf{O(n_1, n_2)}$**:** O stands for the offspring and $\mathbf{n_1}$&$\mathbf{n_2}$are the pixel coordinates of the offspring or it can be said that these represents the set of offspring of the node(n_1, n_2). The size can be four or zero depending upon the number of offspring. For example, the $O(0,1)$ in the figure 4.2 has coordinates of the pixels b_1, b_2, b_3 and b_4.

2) $\mathbf{D(n_1, n_2)}$ **:** Notation D is for the descendants whereas $\mathbf{n_1}$&$\mathbf{n_2}$ represents the positions of the pixels. So this represents the set of descendants. Descendants here include the offspring, offspring's offspring and so on depending upon the number of sub-bands. For

example the descendants set D(0,1) consists of the coordinates of the pixels $b_1, \ldots\ldots, b_4, b_{11}, \ldots\ldots, b_{14}, \ldots\ldots b_{41}, \ldots\ldots, b_{44}$. Since by now we know that every node may have four or no offspring therefore the size of this node may be either four or zero.

3) **$L(n_1, n_2)$:** This notation is used to represent the set which has the coordinates of a descendants at position at (n_1, n_2) less than the offspring or we can say that **$L(n_1, n_2)$**is the difference between $D(n_1, n_2)$ and $O(n_1, n_2)$.It may be represented as .

$$L(n_1, n_2) = D(n_1, n_2) - O(n_1, n_2) \quad (4.9)$$

4) **H:** It is comprised of all the roots of special orientation tree which belongs to highest level pyramid (i.e LL Sub-band).

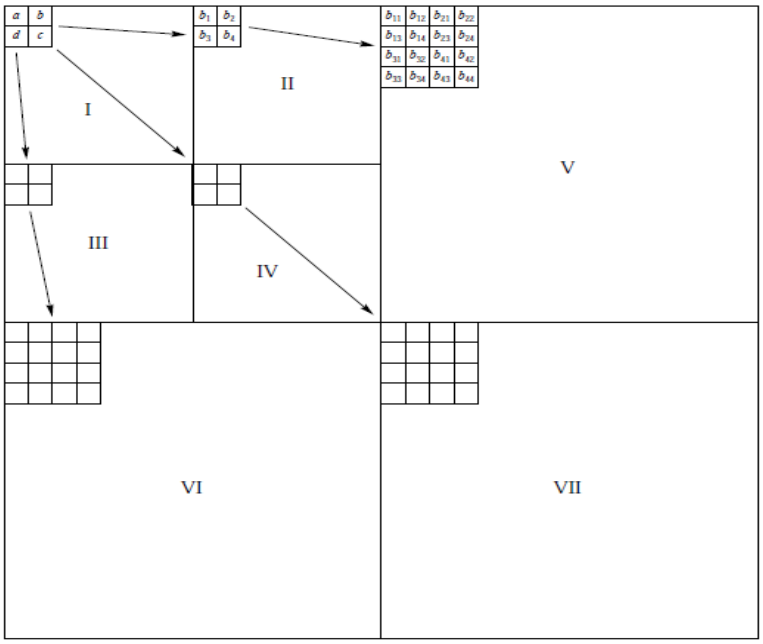


Figure 4.2: Data structure used in the SPIHT algorithm

4.6 SPIHT Encoding and Decoding

As name states rule of set partitioning technique is observed by SPIHT. The good thing is that at both encoder and decoder ends similar algorithm is being run. Moreover, ordering information is not sent explicitly like other algorithm of embedded transmission which makes SPIHT algorithm more efficient than the others as similar algorithm for both encoder and decoder. The set of lists which are continuously maintained / updated during the process are as under:-

1) LIP- List of insignificant pixels.
2) LSP- List of significant pixels.
3) LIS- List of insignificant sets.

The position of coordinates is the mean of identification for every element. LIP and LIS contains the elements which are individual pixels whereas, LIS has the sets of either as $L(n_1, n_2)$ orset $D(n_1, n_2)$.

From the outset every pixel and set is considered insignificant. 'n' is found with the help of the coefficients' ceiling magnitude. There are three fundamental passes the sorting pass, the refinement pass and the upgrading of quantization step pass.

Until the encoder transmits the least significant bit, the recursive process of these passes continues and keeps repeating itself iteratively. It is during the sorting pass that insignificant pixels of LIP are ascertained whether they are significant or otherwise. If found significant, then they are passed to the LSP. Similarly, significance of the sets of LIS is ascertained and those found significant are partitioned. These are also taken out from the LIS. Subsets having entries more than one are placed in the LIS. On the same lines single pixels are adjusted in the

LSP or LIP keeping in view their status. With this preview encoding algorithm may be summarized in four steps as under:-

4.6.1 Step-1: Initialization

- Output $n = \lfloor \log_2 (\max_{(x_1,x)} \{|c_{x_1,x_2}|\}) \rfloor$
- Set the LSP = $\{\phi\}$
- Set the LIP = $\{(x_1, x_2) \in H\}$ and LIS= $\{D(x_1, x_2), (x_1, x_2) \in H\}$

4.6.2 Step-2: The Sorting Pass

1) Each element of the LIP is checked for the significance. Give the output '1' or '0' to each entry depending upon whether it is found as significant or insignificant respectively. If significant then remove it from the LIP and place it in the LSP.

2) Now the significance of each set of LIS should be checked. Output its sign as significance if it is found to be significant. Following rule 2 or rule 3, partition this element depending upon the set if it is $L(x_1, x_2)$ or $D(x_1, x_2)$. All the three lists i.e LIP, LSP and LIS be upgraded throughout the process depending upon the significance.

4.6.3 Step-3: The Refinement Pass

In refinement pass, other than the elements which were having the same 'x' in the process of sorting pass and have been moved to LSP, the most significant bit from all the elements of the LSP be placed at the x^{th} position.

4.6.4 Step-4: Renewing Quantization Step Pass

In this step keep decreasing 'x' by 1 and keep repeating the steps of sorting pass and refinement pass until x=0.

The decoder follows the exactly the same steps as those of encoder. The output generated by the encoder becomes the input to the decoder.

4.7 Example 4.1

In this example I have applied the SPIHT algorithm, for one pass only, on the same example as was taken in case of EZW so that the comparison between the two techniques can be drawn. For better assimilation DWT matrix of an 8x8 image has been shown in table 4.2 at the cost of repetition. The results of application of one pass of SPIHT algorithm on table 4.2 have been shown in the table 4.3. Table 4.3 is clearly indicating about the data coded and updating of the control lists. It can be observed that after one pass SPIHT algorithm has used 29 bits without using any other kind of entropy encoding which are 4 bits less as compared to EZW.

	0	1	2	3	4	5	6	7
0	63	–34	49	10	7	13	–12	7
1	–31	23	14	–13	3	4	6	–1
2	15	14	3	–12	5	–7	3	9
3	–9	–7	–14	8	4	–2	3	2
4	–5	9	–1	47	4	6	–2	2
5	3	0	–3	2	3	–2	0	4
6	2	–3	6	–4	3	6	3	6
7	5	11	5	6	0	3	–4	4

Table4.2: Set of Image Wavelet Coefficients used by example. The numbers outside the box indicate the set of co-ordinates used.

Comm.	Pixel or Set Tested	Output Bit	Action	Control Lists
(1)				LIS = {(0,1)A,(1,0)A,(1,1)A} LIP = {(0,0),(0,1),(1,0),(1,1)} LSP = ∅
(2)	(0,0)	1+	(0,0) to LSP	LIP = {(0,1),(1,0),(1,1)} LSP = {(0,0)}
	(0,1)	1−	(0,1) to LSP	LIP = {(1,0),(1,1)} LSP = {(0,0),(0,1)}
	(1,0)	0	none	
	(1,1)	0	none	
(3)	$\mathcal{D}$(0,1)	1	test offspring	LIS = {**(0,1)A**,(1,0)A,(1,1)A}
	(0,2)	1+	(0,2) to LSP	LSP = {(0,0),(0,1),(0,2)}
	(0,3)	0	(0,3) to LIP	LIP = {(1,0),(1,1),(0,3)}
	(1,2)	0	(1,2) to LIP	LIP = {(1,0),(1,1),(0,3),(1,2)}
	(1,3)	0	(1,3) to LIP	LIP = {(1,0),(1,1),(0,3),(1,2),(1,3)}
(4)			type changes	LIS = {(1,0)A,(1,1)A,(0,1)B}
(5)	$\mathcal{D}$(1,0)	1	test offspring	LIS = {**(1,0)A**,(1,1)A,(0,1)B}
	(2,0)	0	(2,0) to LIP	LIP = {(1,0),(1,1),(0,3),(1,2),(1,3),(2,0)}
	(2,1)	0	(2,1) to LIP	LIP = {(1,0),(1,1),(0,3),(1,2),(1,3),(2,0),(2,1)}
	(3,0)	0	(3,0) to LIP	LIP = {(1,0),(1,1),(0,3),(1,2),(1,3),(2,0),(2,1),(3,0)}
	(3,1)	0	(3,1) to LIP	LIP = {(1,0),(1,1),(0,3),(1,2),(1,3),(2,0),(2,1),(3,0),(3,1)}
			type changes	LIS = {(1,1)A,(0,1)B,(1,0)B}
(6)	$\mathcal{D}$(1,1)	0	none	LIS = {**(1,1)A**,(0,1)B,(1,0)B}
(7)	$\mathcal{L}$(0,1)	0	none	LIS = {(1,1)A,**(0,1)B**,(1,0)B}
(8)	$\mathcal{L}$(1,0)	1	add new sets	LIS = {(1,1)A,(0,1)B,(2,0)A,(2,1)A,(3,0)A,(3,1)A}
(9)	$\mathcal{D}$(2,0)	0	none	LIS = {(1,1)A,(0,1)B,**(2,0)A**,(2,1)A,(3,0)A,(3,1)A}
(10)	$\mathcal{D}$(2,1)	1	test offspring	LIS = {(1,1)A,(0,1)B,(2,0)A,**(2,1)A**,(3,0)A,(3,1)A}
	(4,2)	0	(4,2) to LIP	LIP = {(1,0),(1,1),(0,3),(1,2),(1,3),(2,0),(2,1),(3,0),(3,1),(4,2)}
	(4,3)	1+	(4,3) to LSP	LSP = {(0,0),(0,1),(0,2),(4,3)}
	(5,2)	0	(5,2) to LIP	LIP = {(1,0),(1,1),(0,3),(1,2),(1,3),(2,0),(2,1),(3,0),(3,1),(4,2),(5,2)}
	(5,3)	0	(5,3) to LIP	LIP = {(1,0),(1,1),(0,3),(1,2),(1,3),(2,0),(2,1),(3,0),(3,1),(4,2),(5,2),(5,3)}
(11)			(2,1) removed	LIS = {(1,1)A,(0,1)B,(2,0)A,(3,0)A,(3,1)A}
(12)	$\mathcal{D}$(3,0)	0	none	LIS = {(1,1)A,(0,1)B,(2,0)A,**(3,0)A**,(3,1)A}
	$\mathcal{D}$(3,1)	0	none	LIS = {(1,1)A,(0,1)B,(2,0)A,(3,0)A,**(3,1)A**}
(13)				LIS = {(1,1)A,(0,1)B,(2,0)A,(3,0)A,(3,1)A} LIP = {(1,0),(1,1),(0,3),(1,2),(1,3),(2,0),(2,1),(3,0),(3,1),(4,2),(5,2),(5,3)} LSP = {(0,0),(0,1),(0,2),(4,3)}

Table 4.3:SPHIT Process.

Entropy Encoding (Huffman Encoding)

Chapter 5

Entropy Encoding

5.1 Introduction

It has been observed that bit- stream that happens to be the output of SPIHT consists of a long series of zeroes which can be compressed further, thus SPIHT is not advocated to be used as sole mean of compression. Therefore, wavelet transformed images have been initially compressed by using SPIHT technique and to attain more compression, the output bit streams of SPIHT are then fed to entropy encoders; Huffman and Arithmetic encoders, for further de-correlation. The experimental results of these cascading demonstrate that SPIHT combined with Arithmetic coding yields better compression ratio as compared to SPIHT cascaded with Huffman coding. Whereas, SPIHT once combined with Huffman coding is proved to be comparatively efficient. In this research we have concatenated SPIHT with Huffman encoding the detailed description of which has been discussed in this chapter.

5.2 Huffman Coding

Huffman encoding is a kind of entropy encoding which is lossless in itself. It works depending upon the probability of statistical data and in terms of the image it can be said that it is based on frequency of emergence or appearance of the pixels in an image. It distinct feature is that it should allocate lesser number of bits to encode a data that appears more number of time. Code book is used to store the codes. This code book can be constructed separately for each image or a set of images. To enable decoding it must be ensured in all cases that the encoded data and the code book must be transmitted. Huffman encoding has following features:-

1) Fixed length of symbols is allotted variable length code-words.
2) Symbols are decoded uniquely i.e. not even a single code word posses the prefix of the previous code word. Flow of algorithm is represented in Figure 5.1.

5.2.1 The Basic Principles

The Huffman man coding is based on few key principles which are as under:-

1) Least number of bits is used to represent the greater probability symbols whereas more number of bits is required to represent the lesser probability symbols. To fixed group of symbols a variable length cod-word is assigned.
2) While assigning code-word to the next symbol it must be ensured that it does not have any previous code-word as its part. This will make the Huffman coding a distinctively decodable scheme.
3) Every symbol should have a unique code-word.

5.2.2 Huffman Coding – Flow Chart

A flow chart in the Figure 5.1 has been used to express the sequence of algorithm. The symbols in this have been arranged keeping in view the values of probabilities in a decreasing order. The by joining the symbols with least probabilities a subgroup is being made and a bit '1' is assigned to upper symbol and '0' is assigned to lower symbol. Having done this now look for the next unmerged symbols if there exits any combine the two with keeping in view the same consideration as was done in the previous step. If there is no such symbol that can be combined then start generating the code-word for the symbols (Figure 5.1).

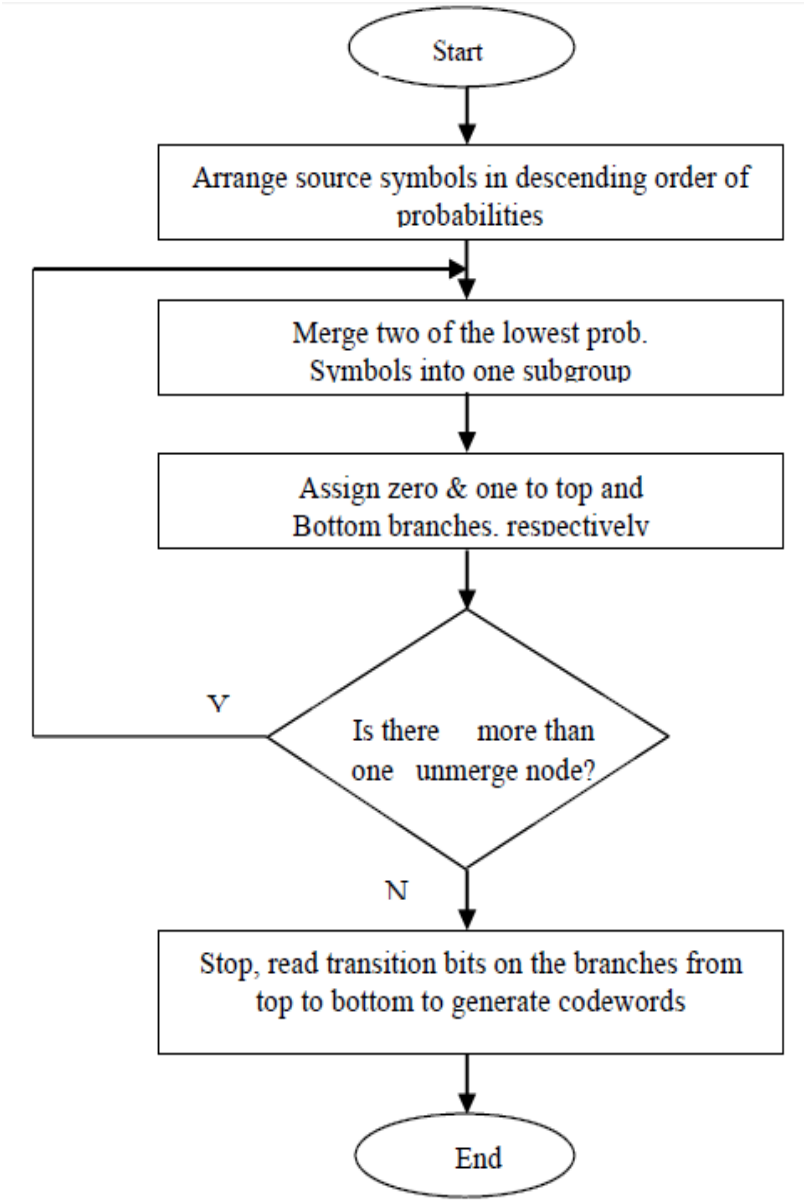


Figure 5.1: Flowcharts for Huffman Coding

5.2.3 Example-5.1

For detailed explanation and understanding of the aforementioned process let us solve an example. Down in table 5.1 frequency of occurrences of a set of symbols is given along with its respective symbol.

Symbols	Frequency
222	5
136	7
14	9
2	10
0	100

Table 5.1: Symbol with Frequency of Occurrences

Step-1: Symbols be arranged in accordance to their decreasing frequency of occurrence (Table 5.2)

Symbols	Frequency
0	100
2	10
14	9
136	7
222	5

Table 5.2: Table of arranged symbols in decreasing order of frequency

Step-2: To make subgroups merge the symbols with least frequencies and to get the total value all subgroups add their occurrences (Figure 5.2)

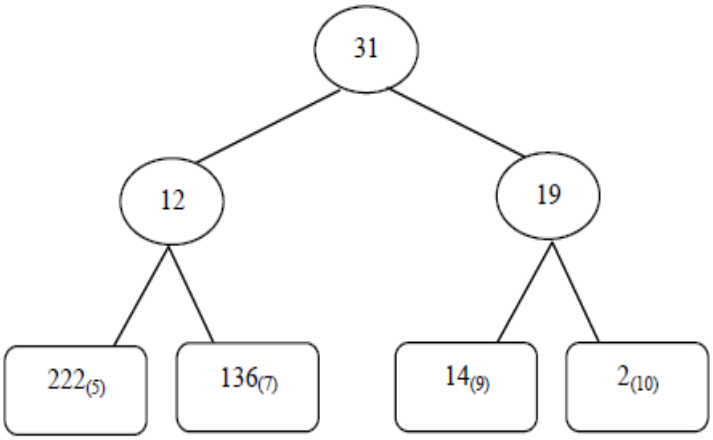


Figure 5.2: Formation of subgroups

Step-3: To ensure correctness and completeness see whether any unmerged node is present or not (Figure 5.3).

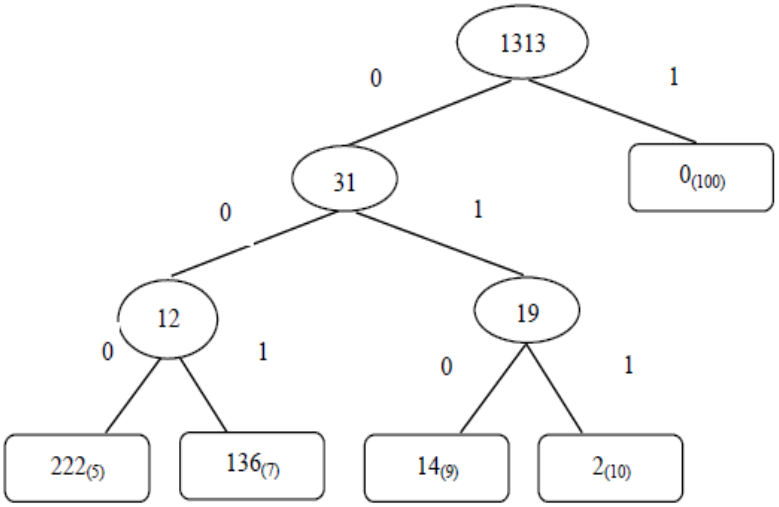


Figure 5.3: Huffman tree processing

Step-4: Symbols are assigned the code-words(Table 5.3)

Symbols	Code-word	Frequency
0	1	100
2	011	10
14	010	9
136	001	7
222	000	5

Table 5.3: Assignment of code-word

Results and simulations

Chapter 6

Results and Simulations

6.1 Introduction

In this section we have discussed as to how SPIHT technique has been concatenated with the Huffman entropy encoding scheme for further reduction in number of bits. This has paved the way towards saving more storage space. In this portion, a comparison is drawn between existing work and our proposed method to exhibit as to how new method saves more number of bits without degrading the quality of image while PSNR remained unchange. This research work is an extension of SPIHT cascaded with the Huffman. In that efforts have been made to improve the bit saving capacity of the aforementioned concatenation by making apt use of Huffman entropy coding. The results of existing and proposed methods have been compared by using various perimeters, like bit saving capacity, elapsed timings, PSNR etc

PSNR is used to compare the distortion. It has been done by using

$$\text{Peak Signal to Noise Ratio (PSNR)} = 10\ \log 10_{10}\left(\frac{\left(\max\left(f(x,y)\right)\right)^2}{\text{MSE}}\right) \qquad (6.1)$$

Input image is denote by equation 6.1 $f(x,y)$ and for the images of grey scale we usually consider

$$\left(\max\left(f(x,y)\right)\right) = 255 \qquad (6.2)$$

Where MSE depicts mean squared error i.e. between original image and the reconstructed image. MSE can be mathematically calculated by using as

$$\text{MSE} = \sum_{XY} \frac{f(x,y) - \tilde{f}(x,y)}{X \times Y} \qquad (6.3)$$

Here the equation 6.3 $\tilde{f}(x, y)$ is the compressed image which has been reconstructed after application of modified SPIHT algorithm.

6.2 Analysis of Concatenation of SPIHT with Huffman

Let us take the same old example 4.1 (Table 6.1(a) and 6.1(b) below), in that a 3-level DWT of an image is passed through the algorithm of SPIHT for one pass. In that case the initial threshold is going to be To = 25 due to largest coefficient being 63. With this, the binary output after one pass is going to be 29 bits in all, 11100011100010000001010110000. It is evident from the binary output steam that there is large series of zeros that can be further compressed. Moreover, same has been seen in a number of statistical analyses the bit steam that comes out as output consists of long series of zeros that can be further compressed, therefore SPIHT is not advocated to be used as sole mean of compression.

34.2329	22.9106	8.0819	-9.5783	2.4702	9.6024	17.4720	20.9260
3.1444	0.0473	-10.7578	5.7983	15.2621	5.7212	-6.8773	-26.2526
-9.4979	-7.2971	8.1126	10.0352	10.4049	3.1472	-12.044	-15.2028
6-7.7991	1.9334	12.7445	13.1993	11.3390	6.9783	4.1331	5.5305
12.7476	13.6053	24.2530	24.4590	21.2853	16.8028	13.9673	14.3350
0.4514	7.7005	16.9633	23.2157	20.2790	16.1249	13.9673	14.3350
7.9368	-4.9201	2.2329	0.4116	29.5710	22.8518	22.6126	6.1349

	0	1	2	3	4	5	6	7
0	63	-34	49	10	7	13	-12	7
1	-31	23	14	-13	3	4	6	-1
2	15	14	3	-12	5	-7	3	9
3	-9	-7	-14	8	4	-2	3	2
4	-5	9	-1	47	4	6	-2	2
5	3	0	-3	2	3	-2	0	4
6	2	-3	6	-4	3	6	3	6
7	5	11	5	6	0	3	-4	4

Table 6.1 (a): Original Matrix **(b): Dwt coefficients of arbitrary data set**

So it is concluded that still there is a redundancy that can be further removed to a great extent with the help of entropy encodings. This is done by

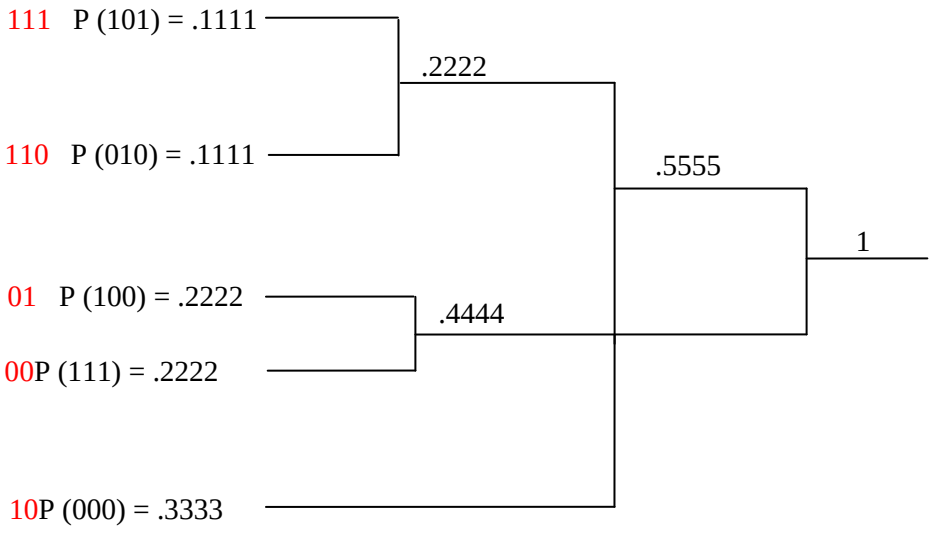
cascading the output of SPIHT with the entropy encodings such that Arithmetic and Huffman. The experimental results of these cascading demonstrate that SPIHT combined with Arithmetic coding yields better compression ratio as compared to SPIHT cascaded with Huffman coding. Whereas, SPIHT once combined with Huffman coding is proved to be comparatively efficient.

It has been observed that the output bit stream of SPIHT algorithm contains a large series of zeros situation and '000' seems to appear with maximum probability somewhat like ¼. Hence if the output of SPIHT is divided every 3 bits each to make it a symbol then there will be a total of eight different symbols with respective probabilities. Now if these symbols are entropy coded with the help of Huffman encoding the output of SPIHT is further compressed. This can be done in a following way:-

1) Make symbols or group of 3 bits each by dividing the output stream. It will result in "111,000,111,000,100,000,010,101,100,00". As evident there will remain zero, one or two bits after this division. These remaining bits will not be able to participate. To record these remaining number of bits for the purpose of unity and to avoid loss of information two bits are cost in the head of output bit stream of Huffman encoding to record the number of bits. These remaining bits come out as direct output in the end. The symbols along with the occurrence probabilities are shown as under:-

Ser No	Symbol	Probability
1	000	0.333
2	010	0.111
3	100	0.222
4	110	0
5	001	0
6	011	0
7	101	0.111
8	111	0.222

2) The output stream fed to Huffman encoding for further compression is shown below and the codeword book that results according to the probabilities enunciated above, by using Huffman encoding, is as in Table 6.1



Expected Codeword Length = L_C

It can be calculated a tree with minimum weight path length from the root.

Lc = 1 + 0 .5555 + 0.2222 + 0 .4444

L_C = 2.222

$H = \Sigma P_i \log_2 P_i$

= {2× (0.1111 $\log_2$0.1111) + 2× (0.2222 $\log_2$ 0.2222) + 0.3333 $\log_2$ 0.3333}

+ {0.2120 +0 .2903 +0 .1590}
0 .301
H = 2.197

$L_C \approx H$ **2.222 ≈ 2.197**

So

$$\mathbf{L_C \leq H \leq L_C + 1}$$

Var = 2×{0.1111(3-2.222)} + 2×{0.2222 (2-2.2222)} + 0.3333}

= 0.1729-0.0987-0.07399

= 0.000207

'C00' →	'01'	'100' →	'11'
'C01' →	'100000'	'101' →	'101'
'C10' →	'1001'	'110' →	'10001'
'C11' →	'10001'	'111' →	'00'

Table 6.2: Code word Comparison Table

From the code book given above we can have the output stream of corresponding codes: 10,00,01,00,01,11,01,1001,101,11,00,. These happens to be 25 bits in total. For the remainder bits, 10 lies in the head and appears at the start indicating that two bits were taken along as the remainder and these two bits are '00' that has appeared in the last of the code. It has been seen that 4 bits have been saved after carrying out this concatenation. Decoding can be done by adopting the reverse process.

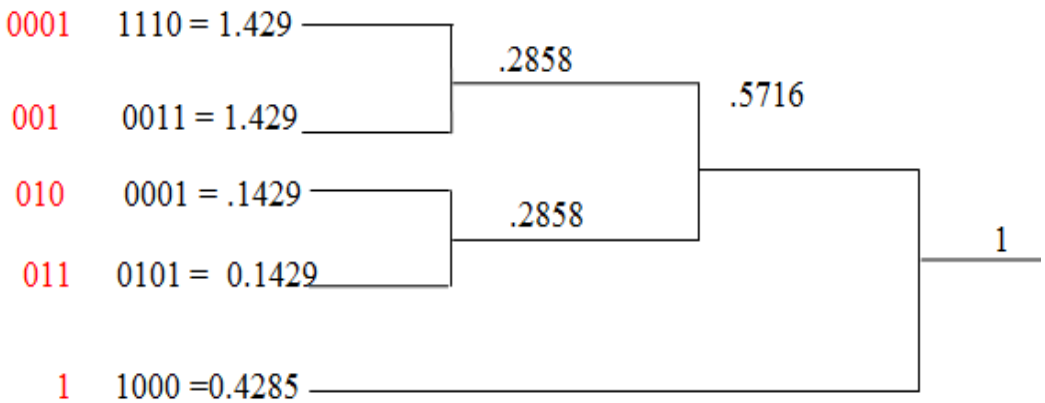
6.3 Proposed Method

Albeit, dividing binary output of SPIHT as 3 bit group, considering it as a symbol, and then passing through the Huffman coding ensures considerable reduction in the redundancy by discarding extra bits yet it is not the maximum what can be achieved. The room for improvements is always there. It is a known fact that the redundancy is the difference between Huffman average code length and its entropy i.e

Redundancy = Lc - H

Given the large alphabet such as a set of letters, digits and punctuations marked by natural language the largest symbol probability is around 15-20 %. Moreover, if the encoder simply writes the compressed data on a file the variance does not matter. A small variance Huffman code is preferable only in case where encoder transmit the compressed data, as its being generated over the network. In such case a code with large variance causes the encoder to generate bits at the rate that varies all the time. Code with larger variance needs larger buffer and that of with smaller variance needs smaller and transmit the bits with constant rate. With these considerations in mind several combination of bits group as symbol were taken and tested for various perimeters to draw the inferences as under :-

For 4 Bits



Expected length of Code

1 + .5710 + .2858 + .2858

$L_C = 2.1432$

H = {(4× (0.1429 $\log_2$0 .1429) + (0.4285 $\log_2$ .428)}

$$= \frac{.4829 + .1577}{.301}$$

H= 2.128 2.1432 ≈ 2.128

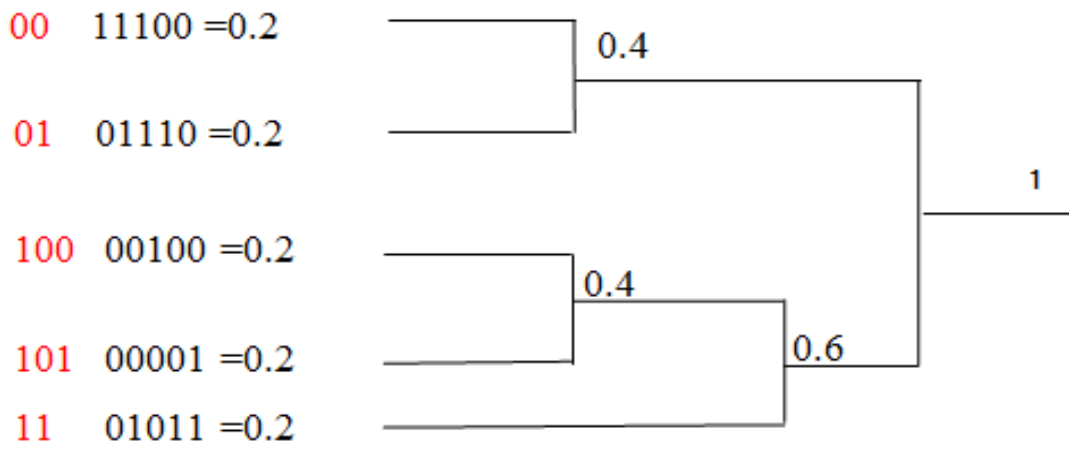
$$L_C \leq H \leq L_C + 1$$

Var = 4× [.1429× (3-2.1432)] + .4285 (1-2.1432)

= .4897 + (-.4899)

= - 0.0001612

For 5 Bits



Expected Codeword Length = L_C = 1+.6+.4+.4

= 2.4

Var = {3× (.2× (2-2.4)) + 2×(.2× (3-2.4))}

= 0

Entropy = H

$$H = \Sigma P_i \log_2 P_i$$

$$= - [5\times (0.2 \log_2 0.2)]$$

= - [5× (0.2×-0.322)]

H = 2.32

$L_C \leq H \leq L_C + 1$

For 6 Bits

11100011100010000001010110000

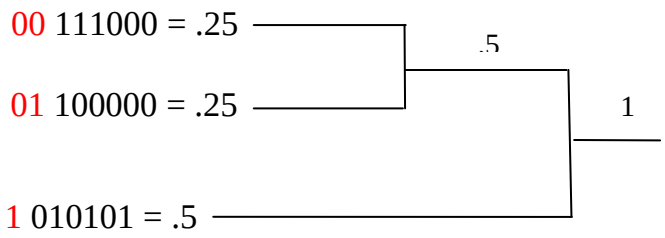
111000	111000	100000	010101 10000
A	A	B	C Remain

6 Bits mean that will 2^6 = 64 symbols

P (11000) = 2/4 = .5 P(A)

P (100000) = 1/4 = .25 P(B)

P (010101) = 1/4 = .25 P(C)



Expected Coded length = L_C

L_C = 1+1.5 =1.5

Entropy = H

= - [.5 $\log_2$ .5 + (.25 $\log_2$ .25)2]

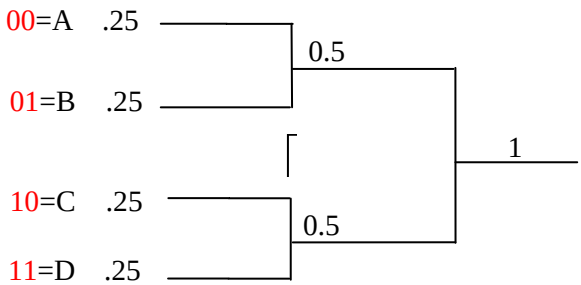
= - (-.5-1)

H = 1.5

Var = {1× (.5(1-1.5) +2× (.25× (2×1.5)}

For 7 Bits

As there will be 4 code of equal probability



Expected Codeword Length = L_C = 2

Entropy = H

H = -4[4× (0.25 $\log_2$ 0.25)] = 2

6.4 Simulations and Results

In this part, details of simulations and deduced results from the simulation have been included to display as to how proposed method proffers better results as compared to existing one. I have used MATLAB 7.11.0 (R2010b) and calculated some performance measures like number of bits saved, MSE, PSNR value, Bit Saving Capacity and Elapsed Time or Execution time for algorithm. Tables 6.3-6.7 exhibit the results using same 8x8 DWT matrix that has been used in examples to illustrate EZW and SPIHT in the previous chapters. Table 6.3 - 6.7 enunciate a no of performance measures like output bits, MSE, PSNR, Bit saving capacity and elapsed timings of 8 x 8 matrix of example 4.1 at various given bit rates for 3, 4, 5, 6, & 7 bit symbols. Efforts have been made to show the graphical representation of the results in figure 6.1 – 6.5 at 0.1, 0.3, 0.6, & 0.9 bit rates.

3 Bits Symbol

S/No	Rate	Output Bits	MSE	PSNR	Bit Saving Capacity	Elapsed Time (Sec)
1	.1	25	72.3906	29.5340	0.1379	0.390627
2	.2	31	49.8906	31.1506	0.0606	0.351054
3	.3	36	49.8906	31.1506	0.1000	0.374458
4	.4	37	49.8906	31.1506	0.2292	0.362652
5	.5	49	41.6406	31.9356	0.1250	0.413168
6	.6	55	32.6406	32.9932	0.1129	0.391448
7	.7	58	31.5156	33.1455	0.1077	0.420814
8	.8	68	26.2656	33.9369	0.0811	0.386800
9	.9	70	25.5156	34.0627	0.1026	0.364632
10	1.0	78	24.3906	34.2586	0.0824	0.425324

Table 6.3: Performance measures at given bit rates for 3 bits symbols

4 Bits Symbol

S/No	Rate	output Bits	MSE	PSNR	Bit Saving Capacity	Elapsed Time (Sec)
1	.1	18	72.3906	29.5340	0.3793	0.453431
2	.2	21	49.8906	31.1506	0.3636	0.375059
3	.3	25	49.8906	31.1506	0.3750	0.341746
4	.4	30	49.8906	31.1506	0.3750	0.328439
5	.5	40	41.6406	31.9356	0.2857	0.325473
6	.6	49	32.6406	32.9932	0.2097	0.404674
7	.7	51	31.5156	33.1455	0.2154	0.413081
8	.8	59	26.2656	33.9369	0.2027	0.421750
9	.9	63	25.5156	34.0627	0.1923	0.423469
10	1.0	73	24.3906	34.2586	0.1412	0.413899

Table 6.4: Performance measures at given bit rates for 4 bits symbols

5 Bits Symbol

S/No	Rate	output Bits	MSE	PSNR	Bit Saving Capacity	Elapsed Time (Sec)
1	.1	20	72.3906	29.5340	0.3103	0.427609
2	.2	20	49.8906	31.1506	0.3939	0.395824
3	.3	23	49.8906	31.1506	0.4250	0.359209
4	.4	31	49.8906	31.1506	0.3542	0.436104
5	.5	38	41.6406	31.9356	0.3214	0.439929
6	.6	45	32.6406	32.9932	0.2742	0.420876
7	.7	44	31.5156	33.1455	0.3231	0.380287
8	.8	54	26.2656	33.9369	0.2703	0.434703
9	.9	57	25.5156	34.0627	0.2692	0.439791
10	1.0	63	24.3906	34.2586	0.2588	0.363293

Table 6.5: Performance measures at given bit rates for 5 bits symbols

6 Bits Symbol

S/No	Rate	output Bits	MSE	PSNR	Bit Saving Capacity	Elapsed Time (Sec)
1	.1	15	72.3906	29.5340	0.4828	0.493891
2	.2	16	49.8906	31.1506	0.5152	0.502989
3	.3	22	49.8906	31.1506	0.4500	0.477635
4	.4	21	49.8906	31.1506	0.5625	0.435720
5	.5	30	41.6406	31.9356	0.4643	0.504025
6	.6	35	32.6406	32.9932	0.4355	0.497648
7	.7	39	31.5156	33.1455	0.4000	0.484547
8	.8	42	26.2656	33.9369	0.4324	0.498170
9	.9	43	25.5156	34.0627	0.4487	0.442348
10	1.0	50	24.3906	34.2586	0.4118	0.501804

Table 6.6: Performance measures at given bit rates for 6 bits symbols

7 Bits Symbol

S/No	Rate	output Bits	MSE	PSNR	Bit Saving Capacity	Elapsed Time (Sec)
1	.1	11	72.3906	29.5340	0.6207	0.691351
2	.2	17	49.8906	31.1506	0.4848	0.640270
3	.3	21	49.8906	31.1506	0.4750	0.687868
4	.4	26	49.8906	31.1506	0.4583	0.698405
5	.5	25	41.6406	31.9356	0.5536	0.646222
6	.6	34	32.6406	32.9932	0.4516	0.682178
7	.7	34	31.5156	33.1455	0.4769	0.734073
8	.8	40	26.2656	33.9369	0.4595	0.695936
9	.9	40	25.5156	34.0627	0.4872	0.701034
10	1.0	45	24.3906	34.2586	0.4706	0.704051

Table 6.7: Performance measures at given bit rates for 7 bits symbols

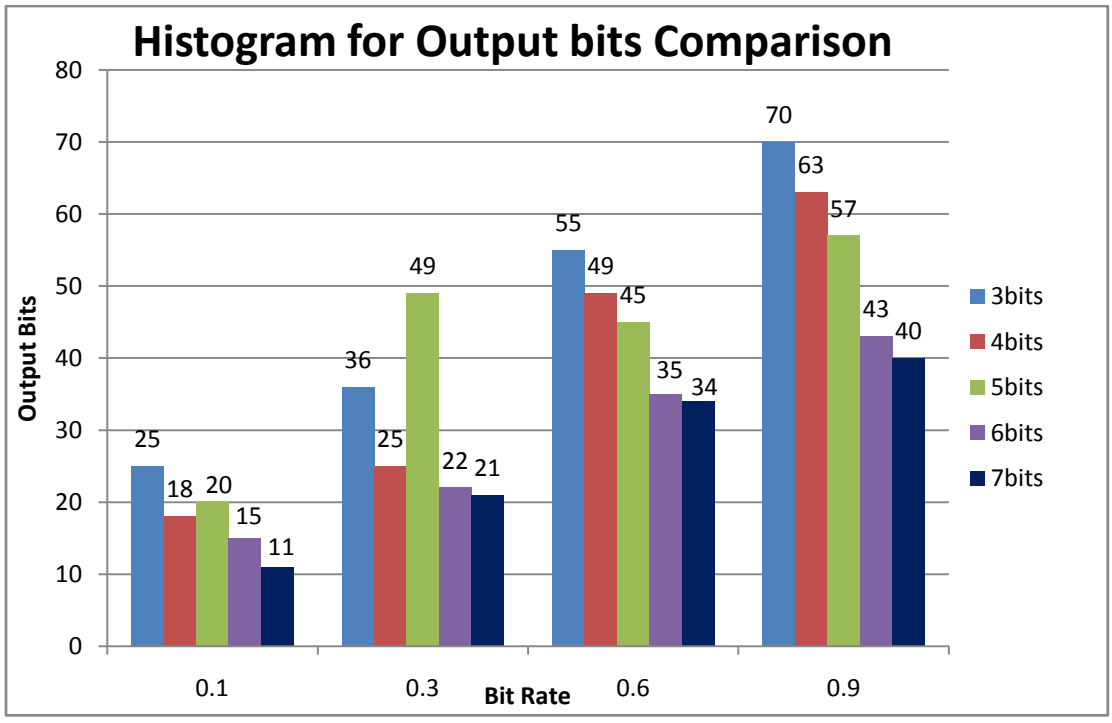


Figure 6.1: Output bit performance at given bit rates for 3, 4, 5, 6 & 7 bits symbol

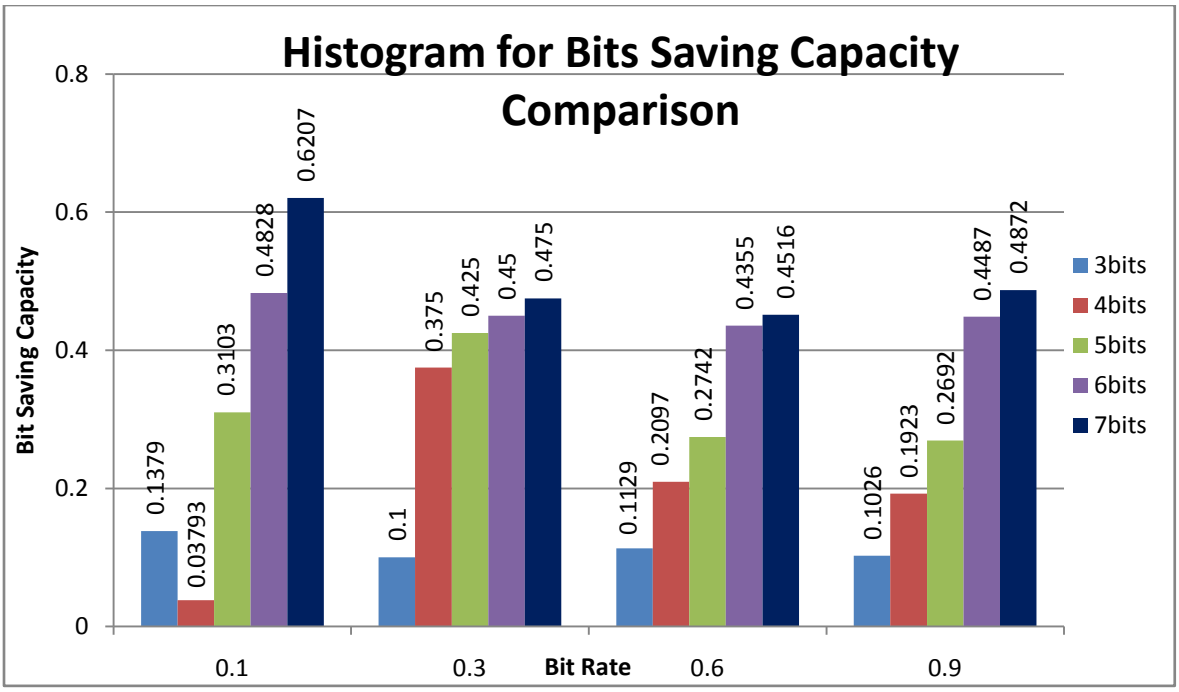


Figure 6.2: Bits saving capacity performance at given bit rates 3, 4, 5, 6& 7bits symbol

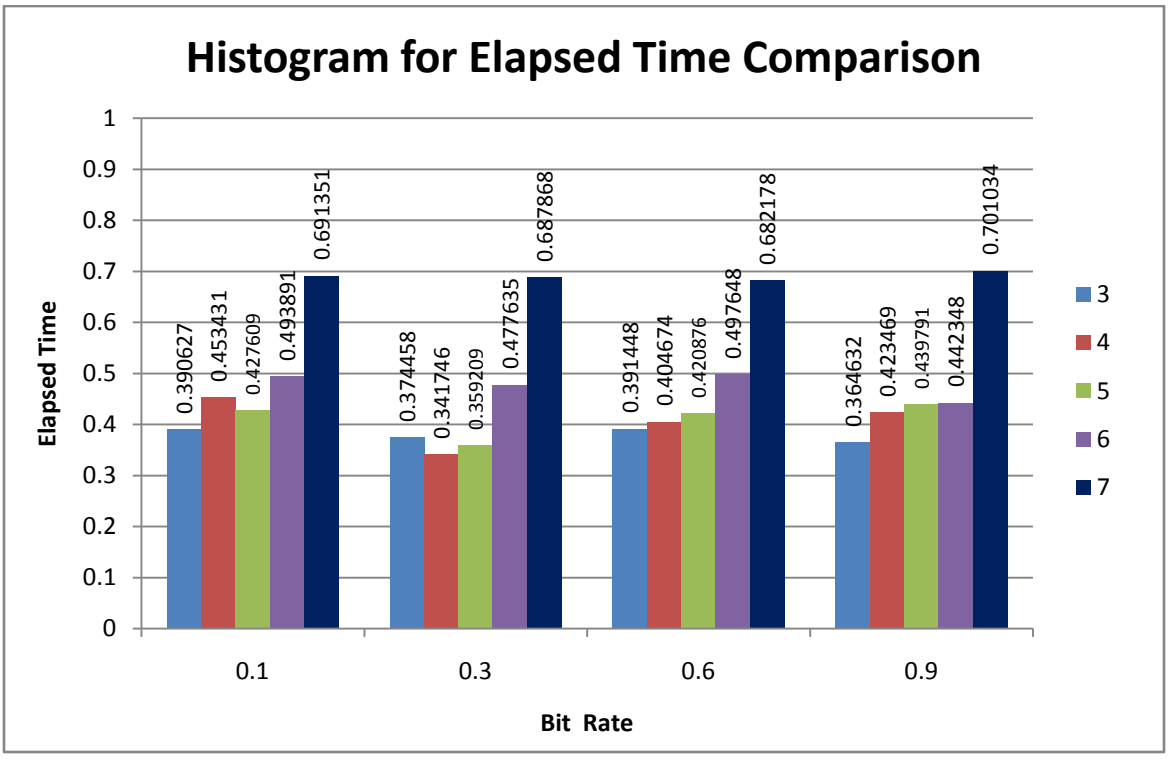


Figure 6.3: Elapsed timing performance at given bit rates 3, 4, 5, 6 & 7 bits symbol

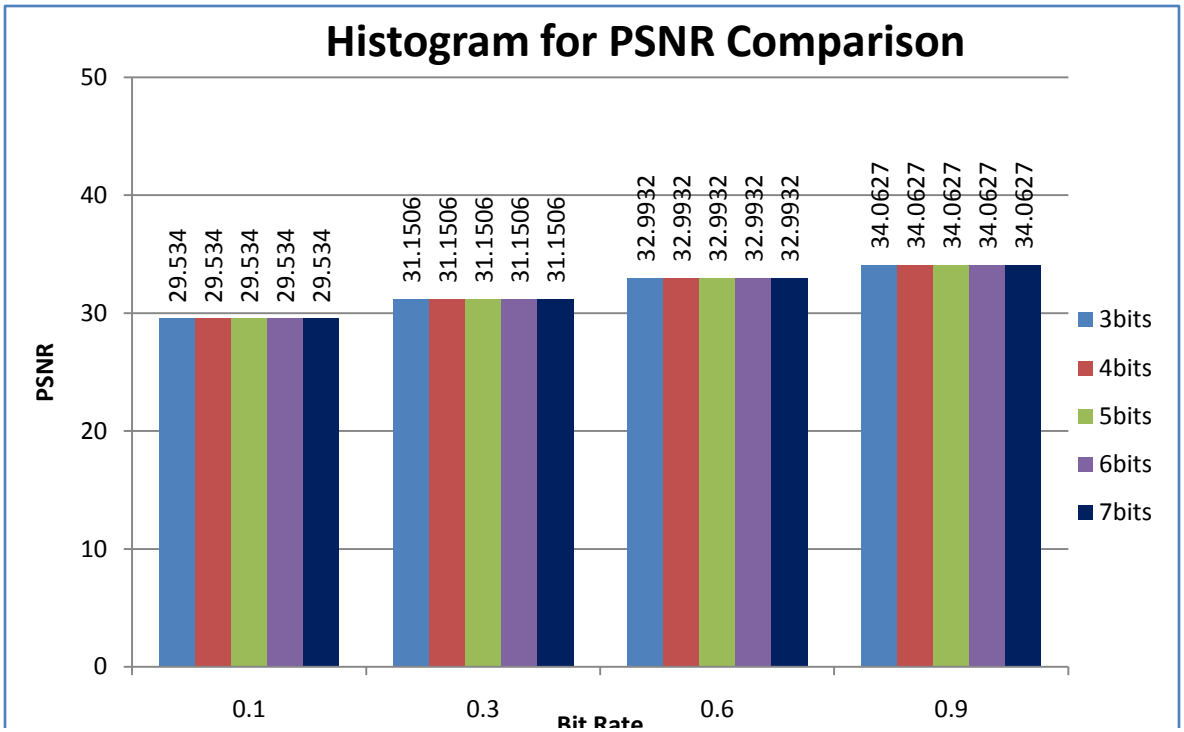


Figure 6.4: PSNR performance at given bit rates 3, 4, 5, 6 & 7 bits symbol

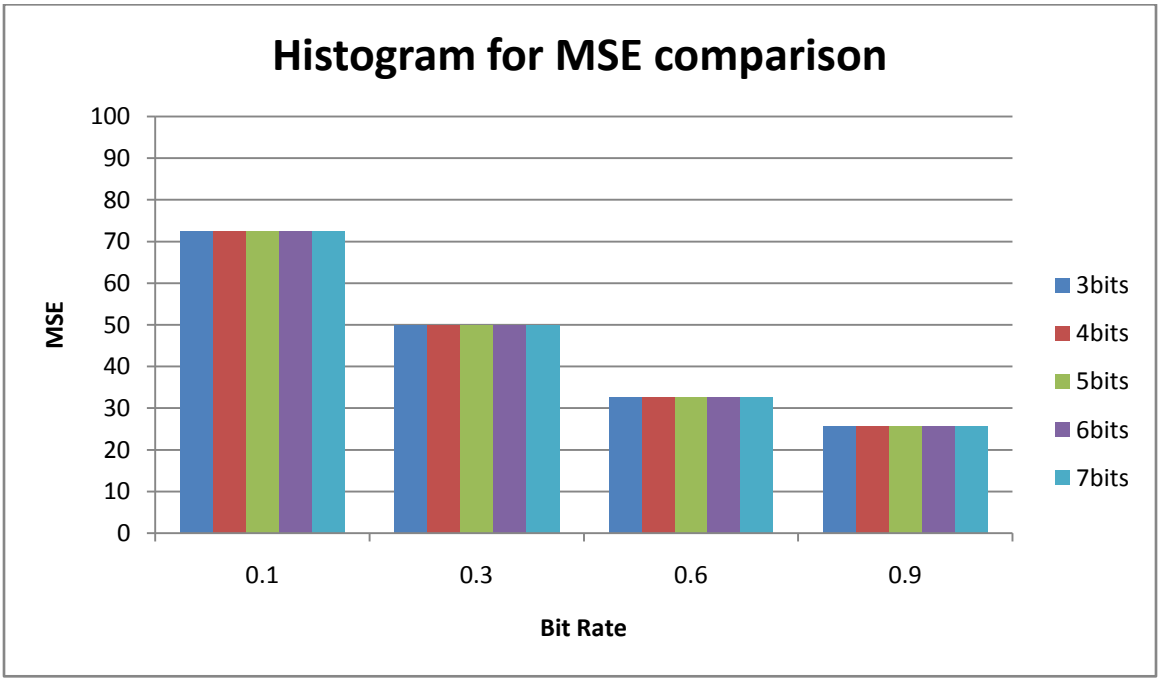


Figure 6.5: MSE performance at given bit rates 3, 4, 5, 6 & 7 bits symbol

It is evident from the results that 4 bits symbol offers improvement in saving more no of bits without degrading picture quality of original image (since PSNR and MSE remain constant). However, it has been observed there is an increase in the elapsed timings as we proceed further from 3 bits to 4 bits and so on. In order to verify the concluding statement, a lot of statistical analyses have been carried out by taking a number of images of various sizes and resolutions. Results of the few images like Lena and Barbra each of size 512x512 have been exhibited in ensuing tables 6.8 – 6.17. The reconstructed images at various rates have been shown in Figure 6.6 and 6.7. Improvements in image quality can be clearly witnessed with the increase in bit rate and if all the bit planes are used are then original images can be recovered completely with some rounding errors.

3 Bits Symbol

S/ No	Rate	Output Bits	MSE	PSNR	Bit Saving Capacity	Elapsed Time (Sec)
1	.1	10525	5.0625e+03	11.0872	0.5985	7.649042
2	.2	21757	1.2792e+03	17.0614	0.5850	14.615299
3	.3	32255	579.4686	20.5005	0.5899	22.183866
4	.4	44762	254.8278	24.0683	0.5731	28.930478
5	.5	62982	129.8626	26.9960	0.5380	37.400846
6	.6	79625	82.2883	28.9774	0.5020	45.515919
7	.7	110079	46.5642	31.4503	0.4401	57.005785
8	.8	121032	39.2820	32.1889	0.4229	60.376668
9	.9	144874	30.0629	33.3505	0.3859	69.899979
10	1.0	171885	22.5379	34.6017	0.3443	80.952381

Table 6.8: Performance measures at given bit rates for 3 bits symbols for Lena 512x512 image

4 Bits Symbol

S/ No	Rate	Output Bits	MSE	PSNR	Bit Saving Capacity	Elapsed Time (Sec)
1	.1	8477	5.0625e+03	11.0872	0.6766	5.658766
2	.2	17617	1.2792e+03	17.0614	0.6640	10.550501
3	.3	26326	579.4686	20.5005	0.6652	15.390950
4	.4	37322	254.8278	24.0683	0.6441	20.268120
5	.5	53661	129.8626	26.9960	0.6063	26.420480
6	.6	69756	82.2883	28.9774	0.5638	31.871435
7	.7	94226	46.5642	31.4503	0.5076	40.429685
8	.8	110958	39.2820	32.1889	0.4709	44.227125
9	.9	134795	30.0629	33.3505	0.4287	53.241057
10	1.0	162821	22.5379	34.6017	0.3789	62.602783

Table 6.9: Performance measures at given bit rates for 4 bits symbols for Lena 512x512 image

5 Bits Symbol

S/ No	Rate	Output Bits	MSE	PSNR	Bit Saving Capacity	Elapsed Time (Sec)
1	.1	7276	5.0625e+03	11.0872	0.7224	5.096015
2	.2	15227	1.2792e+03	17.0614	0.7096	9.804259
3	.3	22840	579.4686	20.5005	0.7096	13.921974
4	.4	32913	254.8278	24.0683	0.6861	18.457624
5	.5	48310	129.8626	26.9960	0.6456	24.732657
6	.6	64081	82.2883	28.9774	0.5993	29.489640
7	.7	88182	46.5642	31.4503	0.5392	38.142099
8	.8	105160	39.2820	32.1889	0.4986	41.826033
9	.9	128829	30.0629	33.3505	0.4539	50.345653
10	1.0	156649	22.5379	34.6017	0.4024	58.234868

Table 6.10: Performance measures at given bit rates for 5 bits symbols for Lena 512x512 image

6 Bits Symbol

S/ No	Rate	Output Bits	MSE	PSNR	Bit Saving Capacity	Elapsed Time (Sec)
1	.1	6488	5.0625e+03	11.0872	0.7525	5.032808
2	.2	13703	1.2792e+03	17.0614	0.7386	9.562132
3	.3	20495	579.4686	20.5005	0.7394	13.607368
4	.4	29862	254.8278	24.0683	0.7152	18.051060
5	.5	44788	129.8626	26.9960	0.6714	24.263317
6	.6	60066	82.2883	28.9774	0.6244	29.872365
7	.7	84064	46.5642	31.4503	0.5607	39.170604
8	.8	100875	39.2820	32.1889	0.5190	43.088888
9	.9	124696	30.0629	33.3505	0.4715	51.767213
10	1.0	152527	22.5379	34.6017	0.4182	61.764241

Table 6.11: Performance measures at given bit rates for 6 bits symbols for Lena 512x512 image

7 Bits Symbol

S/ No	Rate	Output Bits	MSE	PSNR	Bit Saving Capacity	Elapsed Time (Sec)
1	.1	5930	5.0625e+03	11.0872	0.7738	5.387633
2	.2	12629	1.2792e+03	17.0614	0.7591	10.214782
3	.3	18806	579.4686	20.5005	0.7609	14.677125
4	.4	27760	254.8278	24.0683	0.7353	19.689444
5	.5	42320	129.8626	26.9960	0.6895	27.119636
6	.6	57435	82.2883	28.9774	0.6408	33.816600
7	.7	81247	46.5642	31.4503	0.5754	45.736308
8	.8	98237	39.2820	32.1889	0.5316	50.475944
9	.9	121906	30.0629	33.3505	0.4833	60.489067
10	1.0	149598	22.5379	34.6017	0.4293	73.094481

Table 6.12: Performance measures at given bit rates for 7 bits symbols for Lena 512x512 image

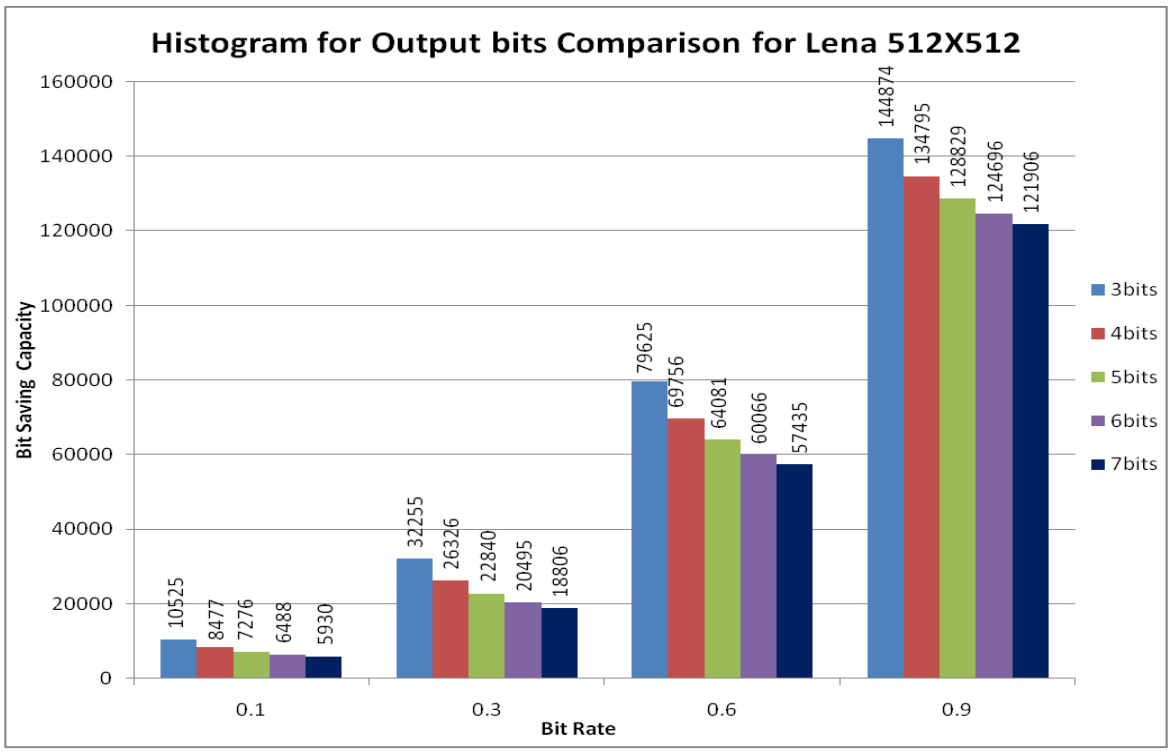


Figure 6.6: Output bit performance at given bit rates for Lena 512x512 Image

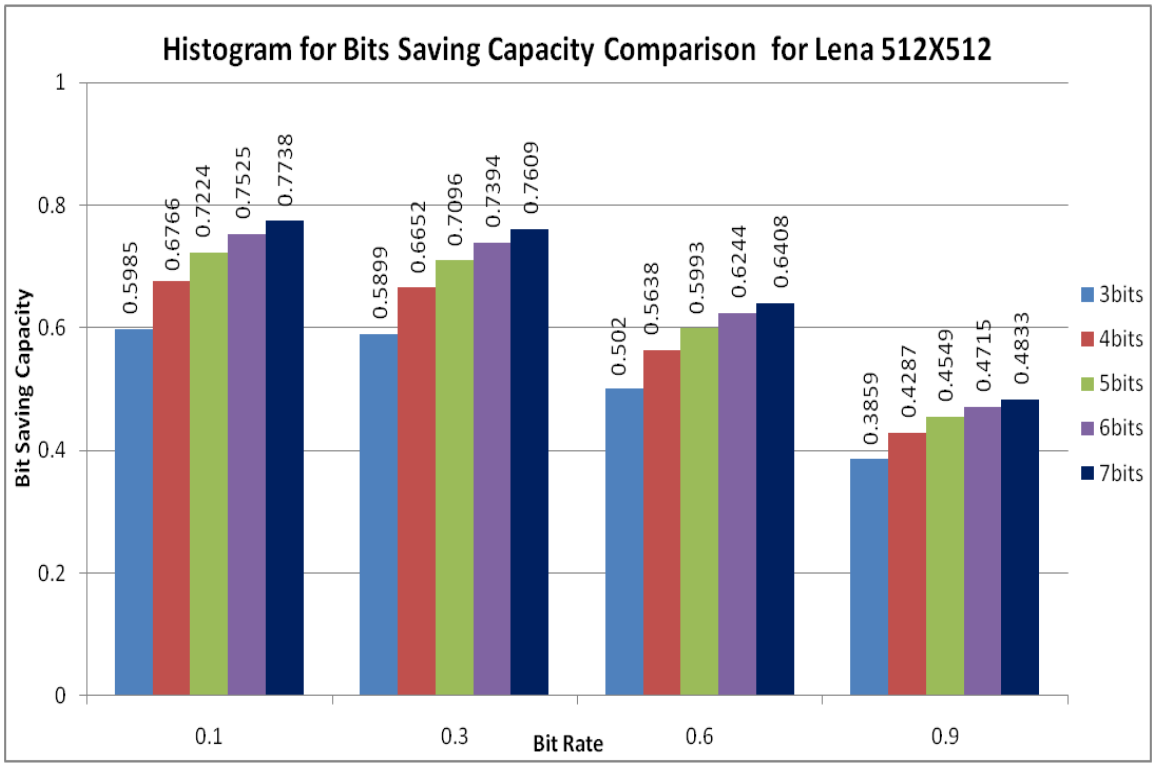


Figure 6.7: Bits saving capacity performance at given bit rates for Lena 512x512 image

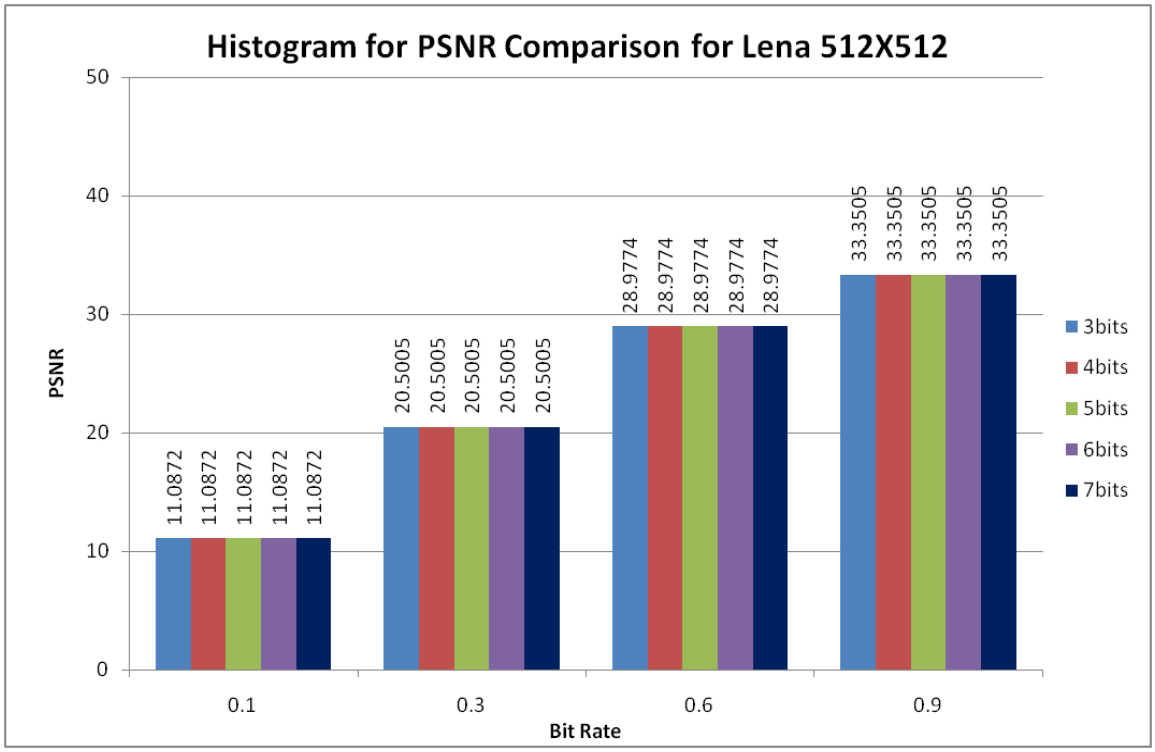


Figure 6.8: PSNR performance at given bit rates for Lena 512x512 image.

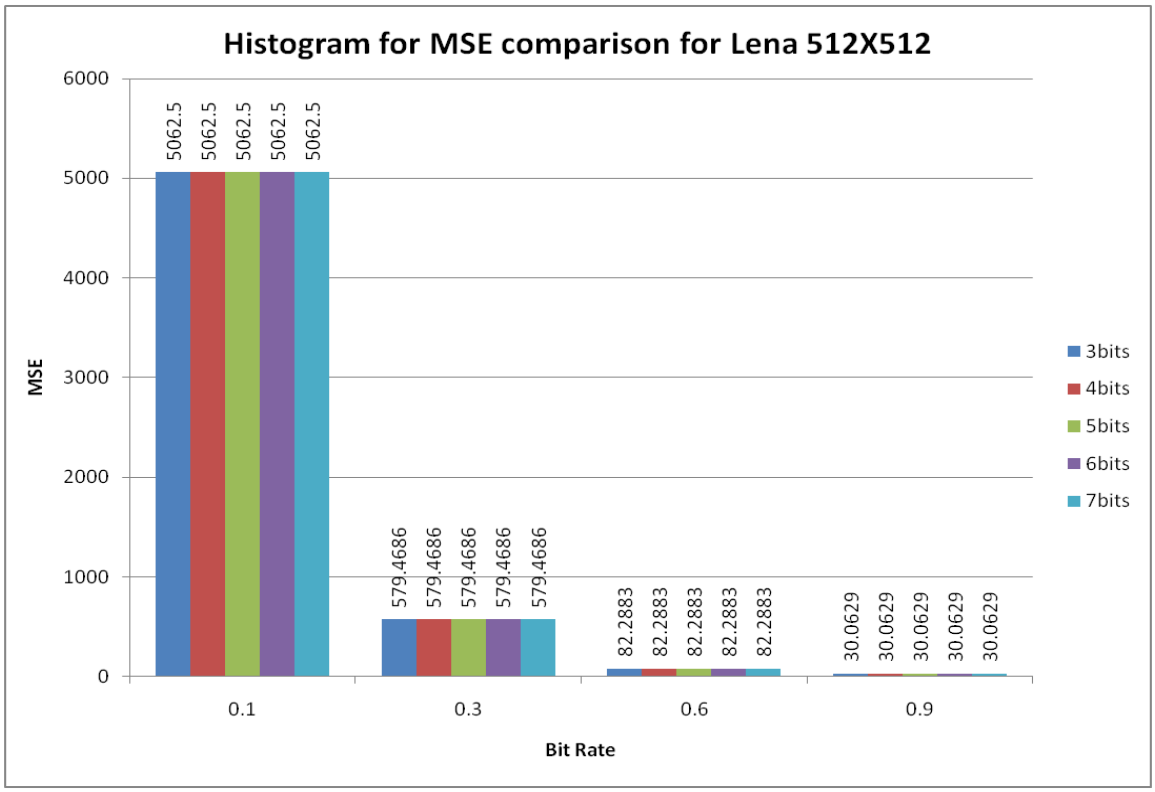


Figure 6.9: MSE performance at given bit rates for Lena 512x512 image

3 Bits Symbol

S/ No	Rate	Output Bits	MSE	PSNR	Bit Saving Capacity	Elapsed Time (Sec)
1	.1	10863	5.5360e+03	10.6988	0.5856	7.721870
2	.2	21902	1.4318e+03	16.5720	0.5822	14.622311
3	.3	31570	662.2012	19.9209	0.5986	21.392457
4	.4	43523	407.6395	22.0280	0.5849	29.768435
5	.5	59370	294.8762	23.4344	0.5470	39.137661
6	.6	79780	228.1006	24.5495	0.4928	44.865773
7	.7	103391	144.5186	26.5316	0.4366	53.703438
8	.8	127482	104.6319	27.9342	0.3921	71.394344
9	.9	155844	67.5941	29.8317	0.3394	76.300887
10	1.0	179412	51.9995	30.9708	0.3156	80.917493

Table 6.13: Performance measures at given bit rates for 3 bits symbols for Barbra 512x512 image

4 Bits Symbol

S/ No	Rate	Output Bits	MSE	PSNR	Bit Saving Capacity	Elapsed Time (Sec)
1	.1	8725	5.5360e+03	10.6988	0.6671	5.616361
2	.2	17647	1.4318e+03	16.5720	0.6634	10.610120
3	.3	25254	662.2012	19.9209	0.6789	15.310136
4	.4	35673	407.6395	22.0280	0.6598	20.098435
5	.5	50516	294.8762	23.4344	0.6146	25.704920
6	.6	70030	228.1006	24.5495	0.5548	33.405441
7	.7	94161	144.5186	26.5316	0.4869	39.181772
8	.8	118222	104.6319	27.9342	0.4363	48.466942
9	.9	147052	67.5941	29.8317	0.3767	59.609005
10	1.0	170747	51.9995	30.9708	0.3486	61.330953

Table 6.14: Performance measures at given bit rates for 4 bits symbols for Barbra 512x512 image

5 Bits Symbol

S/ No	Rate	Output Bits	MSE	PSNR	Bit Saving Capacity	Elapsed Time (Sec)
1	.1	7618	5.5360e+03	10.6988	0.7094	5.270766
2	.2	15244	1.4318e+03	16.5720	0.7092	9.713162
3	.3	21617	662.2012	19.9209	0.7251	13.931616
4	.4	31253	407.6395	22.0280	0.7019	18.358825
5	.5	44981	294.8762	23.4344	0.6568	23.578007
6	.6	64419	228.1006	24.5495	0.5904	30.695457
7	.7	88183	144.5186	26.5316	0.5194	38.248730
8	.8	112262	104.6319	27.9342	0.4647	44.340586
9	.9	141221	67.5941	29.8317	0.4014	53.475114
10	1.0	164985	51.9995	30.9708	0.3706	61.330953

Table 6.15: Performance measures at given bit rates for 5 bits symbols for Barbra 512x512 image

6 Bits Symbol

S/ No	Rate	Output Bits	MSE	PSNR	Bit Saving Capacity	Elapsed Time (Sec)
1	.1	6773	5.5360e+03	10.6988	0.7416	5.104215
2	.2	13563	1.4318e+03	16.5720	0.7413	10.290315
3	.3	19163	662.2012	19.9209	0.7563	13.919344
4	.4	28171	407.6395	22.0280	0.7313	17.569988
5	.5	41407	294.8762	23.4344	0.6841	22.570054
6	.6	60430	228.1006	24.5495	0.6158	30.179148
7	.7	84089	144.5186	26.5316	0.5417	37.240619
8	.8	107918	104.6319	27.9342	0.4854	45.774842
9	.9	136447	67.5941	29.8317	0.4217	56.494754
10	1.0	160165	51.9995	30.9708	0.3890	64.347909

Table 6.16: Performance measures at given bit rates for 6 bits symbols for Barbra 512x512 image

7 Bits Symbol

S/ No	Rate	Output Bits	MSE	PSNR	Bit Saving Capacity	Elapsed Time (Sec)
1	.1	6208	5.5360e+03	10.6988	0.7632	5.938539
2	.2	12445	1.4318e+03	16.5720	0.7626	10.754466
3	.3	17433	662.2012	19.9209	0.7783	16.405365
4	.4	25855	407.6395	22.0280	0.7534	20.880190
5	.5	38816	294.8762	23.4344	0.7039	25.730849
6	.6	57530	228.1006	24.5495	0.6342	34.154726
7	.7	81083	144.5186	26.5316	0.5581	42.436615
8	.8	104683	104.6319	27.9342	0.5008	52.922680
9	.9	132964	67.5941	29.8317	0.4364	64.424215
10	1.0	156885	51.9995	30.9708	0.4015	74.108488

Table 6.17: Performance measures at given bit rates for 7 bits symbols for Barbra 512x512 image

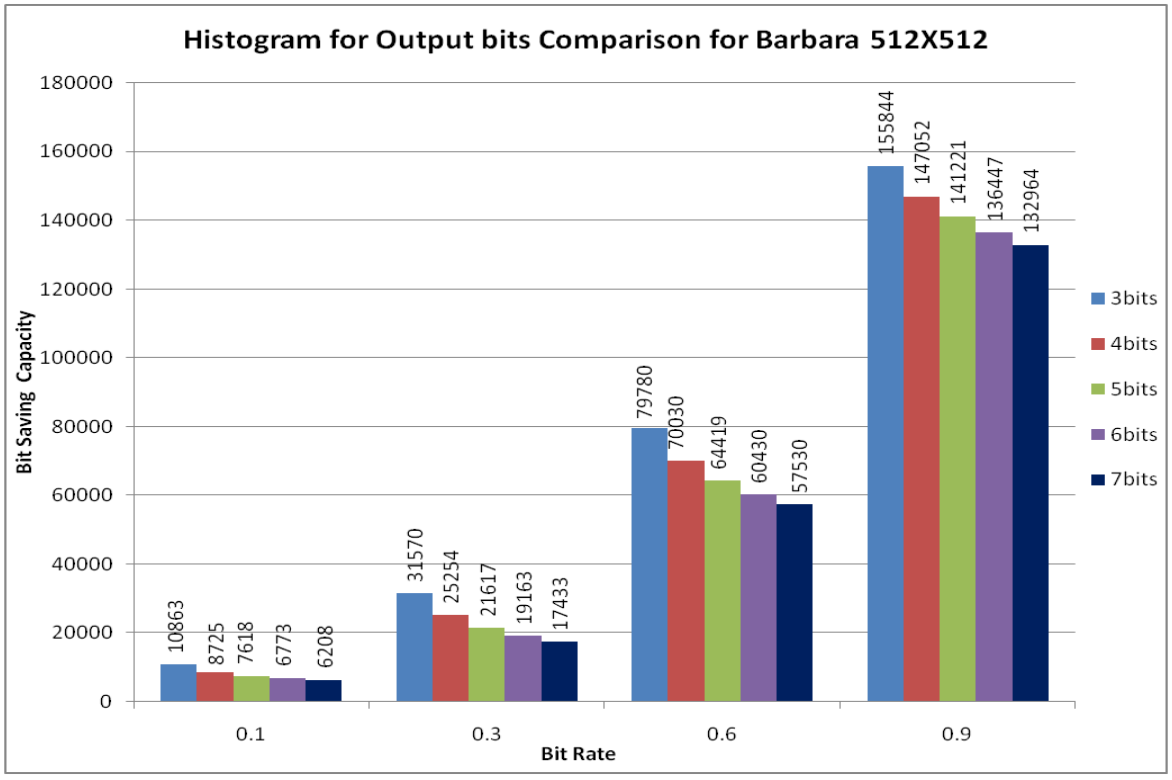


Figure 6.10: Output bit performance at given bit rates for Barbara 512x512 Image

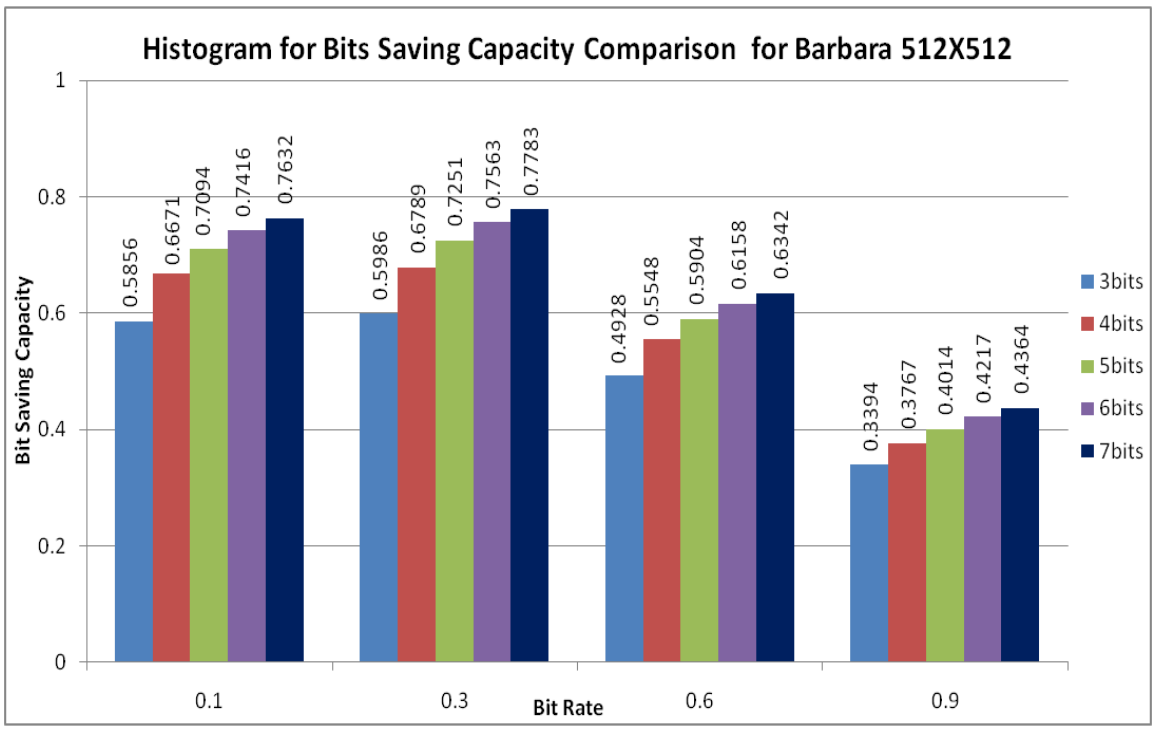


Figure 6.11: Bits saving capacity performance at given bit rates for Barbara 512x512 image

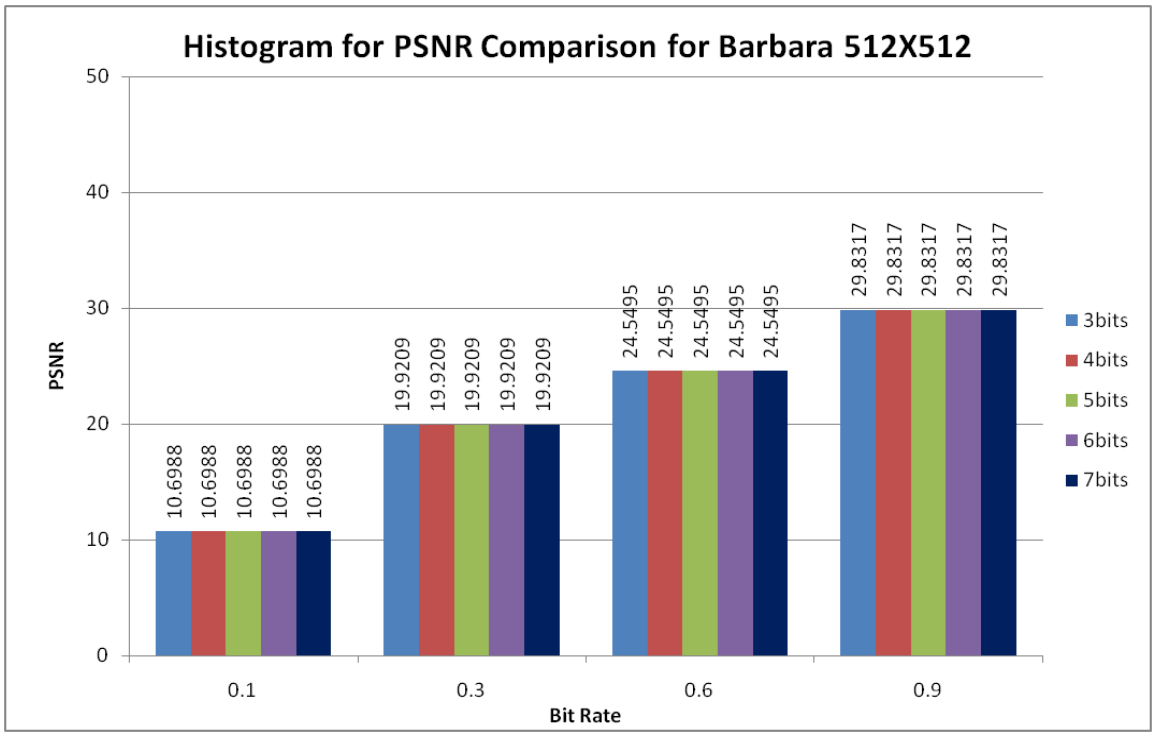


Figure 6.12: PSNR performance at given bit rates for Barbara 512x512 image

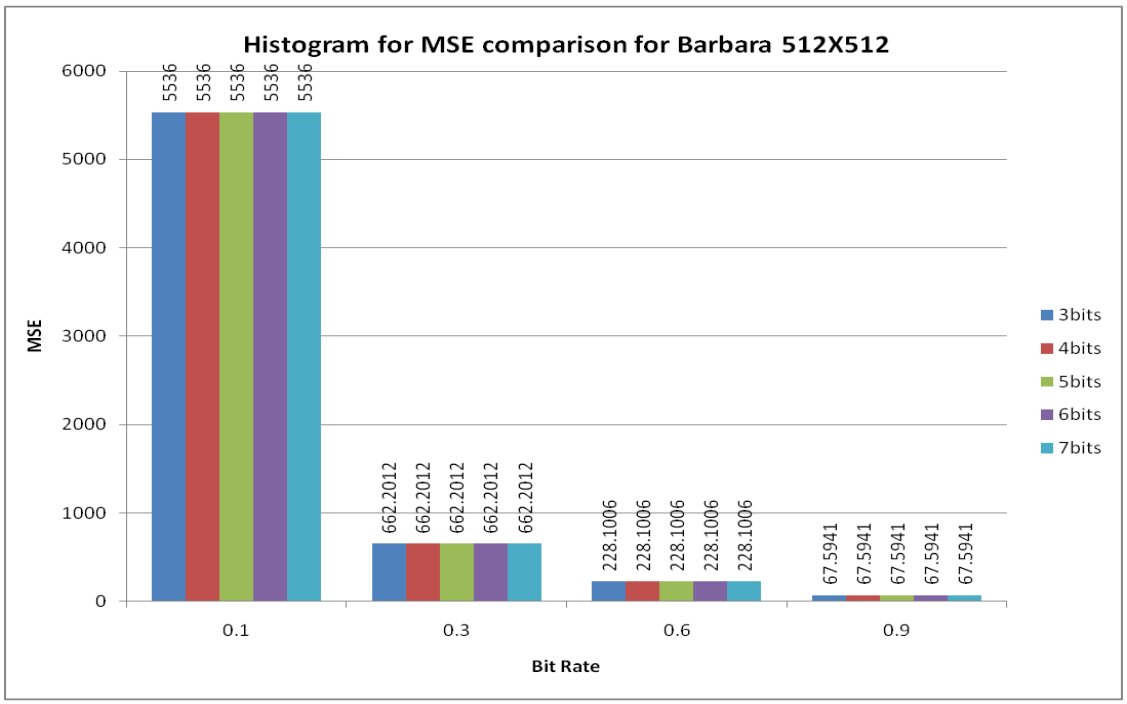


Figure 6.13: MSE performance at given bit rates for Barbra 512x512 image

(a) **Original image of 512x512**

(b) **Reconstructed image at 0.1 bpp**

(c) **Reconstructed image at 0.2 bpp**

(d) **Reconstructed image at 0.3 bpp**

(e) **Reconstructed image at 0.4 bpp**

(f) **Reconstructed image at 0.5 bpp**

(g) **Reconstructed image at 0.6 bpp**

(h) **Reconstructed image at 0.7 bpp**

(i) **Reconstructed image at 0.8 bpp**

(j) **Reconstructed image at 0.9 bpp**

Figure 6.14: Image compression using proposed algorithm on Lena image of size 512 x 512 using various rates.

(a) **Original image of 512x512**

(b) **Reconstructed image at 0.1 bpp**

(c) **Reconstructed image at 0.2 bpp**

(d) **Reconstructed image at 0.3 bpp**

(e) **Reconstructed image at 0.4 bpp**

(f) **Reconstructed image at 0.5 bpp**

(g) **Reconstructed image at 0.6 bpp**

(h) **Reconstructed image at 0.7 bpp**

(i) **Reconstructed image at 0.8 bpp**

(j) **Reconstructed image at 0.9 bpp**

(k) **Reconstructed image at 1.0 bpp**

Figure 6.15: Image compression using proposed method on Barbara image of 512 x 512 using various rates.

Conclusion and Future Work

Chapter 7

Conclusion and Future Work

7.1 Conclusion

A much simpler and fully embedded codec algorithm SPIHT has outperformed EZW and is widely used for the compression of wavelet transformed images. Since the output of SPIHT contains the redundancy therefore, it cannot be advocated to be used as the sole mean of compression. Reduction in redundancy in SPIHT output is very much possible by concatenation of entropy techniques as in Huffman. Albeit, dividing binary output of SPIHT as 3 bit and given as input to Huffman encoder ensures considerable reduction in the redundancy by discarding extra bits yet it is not the maximum what can be achieved. Code with large variance needs larger buffer and one with small variance needs smaller and transmits bits with constant rate. If encoder simply writes the compressed data on a file then variance of the code does not matter. But in case data being generated over NW, the code with large variance causes the encoder to generate bits at a rate that varies all the time. A small variance Huffman code is preferred where encoder transmit the compressed data. Dividing binary output of SPIHT as 4,5,6 &7 bits and given as input to Huffman encoder provides smaller variance than 3 bit symbol and ensure more reduction in the redundancy. It yields better compression ratio thereby enhance the storage space capacity and transmission speed without degradation of the picture quality as MSE and PSNR remain unchanged.

7.2 Future work

The research work concentrated on reduction of storage space and fast transmission by saving more number of bits by preserving MSE and PSNR. The proposed model is based in the empirical results. Future researchers are advocated to take on task of reduction of bits along with improvements in the picture quality that can be done by optimizing the PSNR and work on the mathematical model for the proposed solution.

References

[1] T. Brahimi, A. Melit, F. Khelifi and D. Boutana, "Improvements to SPIHT for Lossless Image Coding," 2006 2nd International Conference on Information & Communication Technologies, Damascus, 2006, pp. 1445-1450

[2] J. H. Zhao, W. J. Sun, Z. Meng, Z. H. Hao, "Wavelet transform characteristics and compression coding of remote sensing images," Optics and Precision Engineering, Vol. 12(2), pp.205-210, 2004.

[3] H. L. Xu, S. H. Zhong, "Image Compression Algorithm of SPIHT Based on Block-Tree," Journal of Hunan Institute of Engineering, Vol. 19(1), pp.58-61, 2009.

[4] R.C Gonzalez & R.E Wood, "Wavelets and Multi-resolution Processing, Digital Image Processing", 2nd edition, Chapter 7, 2002

[5] Marc Antonini, Michel Barlaud, Member, Pierre Mathieu, and Ingrid Daubechies, "Image Coding Using Wavelet Transform", IEEE Transactions on Image Processing, vol. i, no 2. April 1992.

[6] Anoop Mathew, Member, IEEE and Bensiker Raja Singh D, "Image Compression using Lifting Based DWT," International Journal of Computers, Information Technology and Engineering January-June 2009.

[7] B. Yan, H. Zhang, "SPIHT Algorithm and its Improvement," Computer Applications and Software, vol. 25(8), pp.245-247, 2008.

[8] Jose Oliver "A Fast Run-Length Algorithm for Wavelet Image Coding with Reduced Memory Usage" M.P. Malumbres Department of Computer Engineering (DISCA), Technical University of Valencia Camino de Vera 17, 46017, Spain.

[9] Nicholas Kolokotronis, AlikiVassilarakou, SergiosTheodoridis, DionisisCavouras "Wavelet-based medical image compression EleftheriosKofidis" Department of Informatics, Division of Communications and Signal Processing, University of Athens, Panepistimioupolis, TYPA Buildings, GR-15784 Athens, Greece .

[10] Rosenfeld, Azriel, ed. Multiresolution image processing and analysis. Vol. 12.Springer Science & Business Media, 2013.

[11] James S. Walke, "A Primer on Wavelets and Their Scientific Applications", Page No. 130.1999

[12] Introduction to the Discrete Wavelet Transform (DWT), 15,Feb,2004

[13] V.S.Shingate, T.R.Sontakke and S.N.Talbar “Still Image Compression using Embedded Zerotree Wavelet Encoding”

[14] Raid, A. M., W. M. Khedr, M. A. El-dosuky, and Wesam Ahmed. "Image Compression Using EZW." Signal & Image Processing 5, no. 6 (2014): 33.

[15] Shapiro, Jerome M. "Embedded image coding using zerotrees of wavelet coefficients." IEEE Transactions on signal processing 41, no. 12 (1993): 3445-3462.

[16] Li, Mei-shan, Yue Liu, and Hong Zhang. "Wavelet Transform-based Remote Sensing Image Compression." (2015).

[17] Tian, Jun, and Raymond O. Wells Jr. "Embedded image coding using wavelet difference reduction." In Wavelet image and video compression, pp. 289-301.Springer US, 2002.

[18] Shapiro, Jerome M. "Digital data compression system including zerotree coefficient coding." U.S. Patent 5,315,670, issued May 24, 1994.

[19] Rogers, Jon K., and Pamela C. Cosman. "Wavelet zerotree image compression with packetization." IEEE Signal Processing Letters 5, no. 5 (1998): 105-107.

[20] Cotronei, Mariantonia, DamianaLazzaro, Laura B. Montefusco, and LuigiaPuccio. "Image compression through embedded multiwavelet transform coding." IEEE Transactions on Image Processing 9, no. 2 (2000): 184-189.

[21] Zandi, Ahmad, James D. Allen, Edward L. Schwartz, and Martin Boliek. "CREW: Compression with reversible embedded wavelets." In Data Compression Conference, 1995.DCC'95. Proceedings, pp. 212-221. IEEE, 1995.

[22] Li, Jin, Po-Yuen Cheng, and C-C. Jay Kuo. "Improvements of embedded zerotree wavelet (EZW) coding."In Visual Communications and Image Processing'95, pp. 1490-1501.International Society for Optics and Photonics, 1995.

[23] Schwartz, Edward L., Ahmad Zandi, and TiborBoros. "Method and apparatus for spatially embedded coding." U.S. Patent 5,815,097, issued September 29, 1998.

[24] Oktem, Levent, RusenOktem, Karen Eguiazarian, and JaakkoAstola. "Efficient encoding of the significance maps in wavelet based image compression." In Circuits and Systems, 2000.Proceedings.ISCAS 2000 Geneva. The 2000 IEEE International Symposium on, vol. 3, pp. 25-28. IEEE, 2000.

[25] Islam, Asad, and William A. Pearlman. "Embedded and efficient low-complexity hierarchical image coder."In Electronic Imaging'99, pp. 294-305.International Society for Optics and Photonics, 1998.

[26] Da Silva, Eduardo AB, Demetrios G. Sampson, and Mohammed Ghanbari. "A successive approximation vector quantizer for wavelet transform image coding." IEEE Transactions on Image Processing 5, no. 2 (1996): 299-310.

[27] Sriram, Parhasarathy, and Michael W. Marcellin. "Image coding using wavelet transforms and entropy-constrained trellis-coded quantization." IEEE Transactions on Image Processing 4, no. 6 (1995): 725-733.

[28] Algazi, V.R and R.R. Estes, Analysis based coding of image transform and subband coefficients. Proceedings of SPIE, Vol 2564 (1995), p. 11-21.

[29] Basavanthaswami, V., and Somasekhar, T., "Image compression using SPIHT", International Journal of Innovative Research in Computer and Communication Engineering, Volume5, No.2, pp.1839-1844, 2017.

[30] Ghanbari, M., "Standard Codecs Image Compression to Advanced Video Coding", British library, London institute of information and technology, Chapter 4, pp. 80-97, 2011.

[31] Dodla, S., Raju, Y.D.S., and Mohan, K.V.M., "Image compression using wavelet and SPIHT Encoding Scheme", International Journal of Engineering Trends and Technology, Volume 4, No. 9, pp.3863-3865, 2013.

[32] Amir Said, Member IEEE and William A. Pearlman, Senior Member, IEEE,"A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees", IEEE Transaction on circuits and system for video technology, Vol 6,No 3, June 1996 (pp. 234-250)