LOSSLESS MEDICAL IMAGE COMPRESSION USING INTEGER TRANSFORMS AND PREDICTIVE CODING TECHNIQUE

By

DIVYA NEELA

B.E., Jawaharlal Nehru Technological University, India, 2007

A REPORT

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Electrical and Computer Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2010

Approved by:

Major Professor
Dr. Satish Chandra

Abstract

The future of healthcare delivery systems and telemedical applications will undergo a radical change due to the developments in wearable technologies, medical sensors, mobile computing and communication techniques. E-health was born with the integration of networks and telecommunications when dealing with applications of collecting, sorting and transferring medical data from distant locations for performing remote medical collaborations and diagnosis. Healthcare systems in recent years rely on images acquired in two dimensional (2D) domain in the case of still images, or three dimensional (3D) domain for volumetric images or video sequences. Images are acquired with many modalities including X-ray, positron emission tomography (PET), magnetic resonance imaging (MRI), computed axial tomography (CAT) and ultrasound. Medical information is either in multidimensional or multi resolution form, this creates enormous amount of data. Efficient storage, retrieval, management and transmission of this voluminous data is extremely complex. One of the solutions to reduce this complex problem is to compress the medical data losslessly so that the diagnostics capabilities are not compromised. This report proposes techniques that combine integer transforms and predictive coding to enhance the performance of lossless compression. The performance of the proposed techniques is evaluated using compression measures such as entropy and scaled entropy.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my Major Professor Dr. Satish Chandra for his constant help, support and patience in guiding me throughout the project. His encouragement made things very much easier and helped me complete the project with ease.

I would also like to thank Dr. Dwight Day and Dr. Don Gruenbacher for serving in my committee and for their valuable support and cooperation during the project.

Finally, I would like to thank my family and friends for their constant support and encouragement.

Chapter 1 - Introduction

Recent developments in health care practices and development of distributed collaborative platforms for medical diagnosis have resulted in the development of efficient technique to compress medical data. Telemedicine applications involve image transmission within and among health care organizations using public networks. In addition to compressing the data, this requires handling of security issues when dealing with sensitive medical information systems for storage, retrieval and distribution of medical data. Some of the requirements for compression of medical data include high compression ratio and the ability to decode the compressed data at various resolutions.

In order to provide a reliable and efficient means for storing and managing medical data computer based archiving systems such as Picture Archiving and Communication Systems (PACS) and Digital-Imaging and Communications in Medicine (DICOM) standards were developed. Health Level Seven (HL7) standards are widely used for exchange of textual information in healthcare information systems. With the explosion in the number of images acquired for diagnostic purposes, the importance of compression has become invaluable in developing standards for maintaining and protecting medical images and health records.

Compression offers a means to reduce the cost of storage and increase the speed of transmission, thus medical images have attained lot of attention towards compression. These images are very large in size and require lot of storage space. Image compression can be lossless and lossy, depending on whether all the information is retained or some of it is discarded during the compression process. In lossless compression, the recovered data is identical to the original, whereas in the case of lossy compression the recovered data is a close replica of the original with minimal loss of data. Lossy compression is used for signals like speech, natural images, etc., where as the lossless compression can be used for text and medical type images.

There has been a lot of research going on in lossless data compression. The most common lossless compression algorithms are run-length encoding, LZW, DEFLATE, JPEG, JPEG 2000, JPEG-LS, LOCO-I etc. **L**empel–**Z**iv–**W**elch is a lossless data compression

algorithm which can be used to compress images as shown in [18]. The performance of LZW can be enhanced by introducing three methods. The first two methods eliminate the frequent flushing of dictionary, thus lowering processing time and the third method improves the compression ratio by reducing number of bits transferred over the communication channel. The details of these methods are given in [19]. JPEG is most commonly used lossy compression technique for photographic images which can be converted into lossless by performing integer reversible transform. Lossless compression in JPEG [20] is achieved by performing integer reversible DCT (RDCT) instead of the floating point DCT used in original JPEG on each block of the image later using lossless quantization. Lossless JPEG does not allow flexibility of the code stream, to overcome this JPEG 2000[4-5] has been proposed. This technique performs lossless compression based on an integer wavelet filter called biorthogonal 3/5. JPEG 2000's lossless mode runs really slow and often has less compression ratios on artificial and compound images. To overcome this drawback JPEG-LS [7] has been proposed. This is a simple and efficient baseline algorithm containing two distinct stages called modeling and encoding. This technique is a standard evolved after successive refinements as shown in articles [10], [11], and [12]. JPEG-LS algorithm is more scalable than JPEG and JPEG 2000.

In this report, Chapter 1 concentrates on the fundamentals of lossless compression and the areas where it can be used effectively. In Chapter 2 different coding techniques such as Huffman, arithmetic and predictive are discussed in detail. Chapter 3 discusses the basics of wavelets, wavelet transforms both continuous as well as discrete and it also gives detailed description of how the integer wavelet transforms can be implemented using filter bank and lifting scheme. This chapter also discusses the derivation of perfect reconstruction conditions. Chapter 4 includes the implementation techniques and experimental results. These techniques are tested using different medical images, different predictor coefficients in predictive coding technique and different filter coefficients in lifting scheme. The last chapter includes the summary and suggestions for future improvements of the proposed techniques.

Chapter 2 - Lossless Compression Coding Techniques

Lossless compression allows extracting the original image to be perfectly reconstructed from the compressed image. Compression is a two step process, the first step creates a statistical model and the second step uses this model. The mapping of input data into bit sequences is done in such a way that the frequently encountered data will produce shorter output than less frequent data. There are two ways in constructing a statistical model one is called the static model and the other is the adaptive model. The simplest one among the two is the static model where in the data is analyzed and a model is constructed, then this model is stored with the compressed data. Even though this model is simple and modular it has some disadvantages. It is an expensive model for storage and forces using the same model for all the data compressed. It also performs poorly on files containing heterogeneous data. Adaptive models are the most popular type of models in practice. These models are dynamically updated as the data is compressed. Both the encoder and decoder begin with a trivial model yielding poor compression initially, but as they learn more about the data, performance is improved. Lossless compression methods may be categorized according to the type of data they are designed to compress. While, in principle any general-purpose lossless compression algorithm (*general-purpose* meaning that they can compress any bit string) can be used on any type of data, but many are unable to achieve significant compression on data that are not of the form for which they were designed to compress.

Lossless compression is preferred for artificial images such as technical drawings, icons or comics. This is because lossy compression methods when used especially at low bit rates, introduce compression artifacts. It can also be used for high value content, such as medical imagery or image scans in health industry where there is archiving of large number of images. Lossless compression increases the efficiency of sharing and viewing personal images, uses less storage space and is quicker in transmission and reception of images. It is also used in the chain of retail stores where the introduction of new products or the removal of discontinued items can be much more easily completed when all employees receive, view and process images in the same way. In federal government agency where viewing, storing and transmitting processes can reduce large amounts of time spent in explaining and solving the problem or issue, lossless compression is used. It is widely used in the security industry where it can greatly increase the efficiency of recording, processing and storing images. In Museums, where accurate

representation of museum images or gallery items is required, especially when uploading them to a website lossless image compression can be effectively useful.

In this section different coding techniques used to achieve lossless compression are discussed. The primary encoding algorithms used to produce bit sequences are entropy coding techniques of which the most efficient are Huffman coding (also used by DEFLATE) and arithmetic coding. We also go over lossless predictive coding technique.

2.1. Entropy Coding

Entropy measures the amount of information present in the data or the degree of randomness of the data. After the data has been quantized into a finite set of values it can be encoded using an entropy coder to achieve additional compression using probabilities of occurrence of data. This technique reduces the statistical redundancy. The entropy coder encodes the given set of symbols with the minimum number of bits required to represent them. It is a variable length coding which means that it assigns different number of bits to different gray levels. If the probability of occurrence is more, then fewer bits/sample will be assigned.

ENTROPY (H): Suppose we have M input levels or symbols (S1, S2…SM) with their probabilities (P1, P2…., PM)

$$H = -\sum_{K=1}^{M} PklogpPk = \sum_{K=1}^{M} Pklog2(\frac{1}{Pk})$$

In the **least random** case it takes only one value where

$$H=0$$

Most random case:

$$H = \log_2 M$$

The average number of bits per pixel needed with Huffman coding is given by

$$R = \sum_{k=1}^{M} PkNk$$

Where pk represent the probabilities of the symbols and Nk represent the number of bits per the code generated. Coding efficiency (ŋ) can also be calculated using H and R generated earlier

$$\eta = \frac{H}{R} * 100$$

2.2. Huffman Coding

Huffman coding is an entropy coding algorithm which is used in lossless compression. In this technique the two smallest probabilities are combined or added to form a new set of probabilities. This uses a variable length code table which is based on the estimated probability of occurrence for each possible value of the source symbol. This is developed by David. A. Huffman. In Huffman coding each symbol is represented in a specific method which expresses the most common characters with fewer strings than used for any other character. Huffman coding is equivalent to simple binary block encoding. Although Huffman's original algorithm is optimal for a symbol-by-symbol coding (i.e. a stream of unrelated symbols) with a known input probability distribution. It is not optimal when the symbol-by-symbol restriction is dropped, or when the probability mass functions are unknown, not identically distributed, or not independent.

The basic technique involves creating a binary tree of nodes which can be finally stored as an array. This size depends on the number of symbols which have given probabilities. Now the lowest two probabilities will be added and one probability will be represented by '0' and the other probability which is added will be assigned a '1'. This process is repeated until all the additions are completed leaving a sum of one. The simplest construction algorithm uses a priority queue where the node with lowest probability is given highest priority. The performance of the method is calculated using entropy as mentioned in Section 2.1.

2.3. Arithmetic Coding

Arithmetic coding is a form of variable-length entropy encoding that converts a string into another form that represents frequently used characters with fewer bits and infrequently used characters with more bits with the goal of using fewer bits in total. As opposed to other entropy encoding techniques that separate the input message into its component symbols and replace each symbol with a code word, arithmetic coding encodes the entire message into a single

number which is a fraction n where ($0.0 \leq n < 1.0$). It is also called a statistical technique since it is based on probabilities like all the other techniques. The code extracted for the entire sequence of symbols is called a tag or an identifier.

Arithmetic Coding is also a coding technique in which a message is encoded as a real number in an interval from one to zero. Arithmetic coding typically has a better compression ratio than Huffman coding as it produces a single symbol rather than several separate code words. To perform arithmetic coding we will be provided with symbols and probabilities for each symbol and a message that has to be coded. The arithmetic coding technique even though gives better performance than Huffman coding, it has some disadvantages. In this coding technique the whole codeword must be received for the symbols to be decoded, and if a bit is corrupted in the codeword, the entire message could become corrupt.

2.4. Run Length Coding (RLC)

The run length coding is used to compress data streams which contain long runs of the same value or symbol. This is most useful on data that contains many such runs for example simple graphic images such as icons, line drawings, and animations. Zero RLC is the most commonly used run length coding technique. In this technique only zeros are run length coded. The data in zero RLC can be compressed using two methods. In the first method the zeros before the number are zero run length coded and in the second method the zeros after the number are zero run length coded.

In the case of binary images Huffman encoding will not be able to compress properly, in such cases run length coding is very efficient. The most effective application of RLC is fax machine used in CCITT standard which uses 8 bits to encode. The drawback of RLC is that it is not useful with files that don't have many runs as it could potentially double the file size. To overcome these drawbacks an improved RLC technique has been developed called the dynamic window based RLC.

Dynamic window-based RLC (DW-RLC) [17] is a sophisticated RLC, unlike the basic RLC this algorithm does not use runs of same value but it allows the runs to fall within a gray-level range called a dynamic window range. This range is dynamic because it starts out with any

range narrowing down the maximum and minimum values to the actual range while encountering each pixel value.

2.5. Predictive Coding

Predictive coding was developed for the purpose of bit rate reduction in telephone calls. It was discovered that by quantizing and transmitting the residue sequence the bit rate could be substantially reduced while maintaining an acceptable level of fidelity at the receiver. In predictive coding we take the difference or prediction error into consideration rather than taking into account the original sample/image. The differences are taken between the original sample and the sample(s) before the original sample. Let f (n) be the original sample then the difference d (n) will be given by

$$d(n) = f(n) - f(n-1).$$

Figure 2.1 Original Histogram

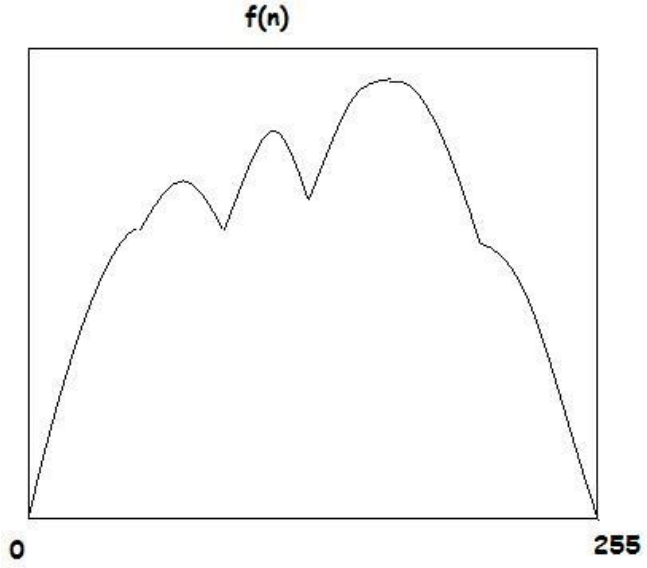


Figure 2.2 Histogram of the Difference

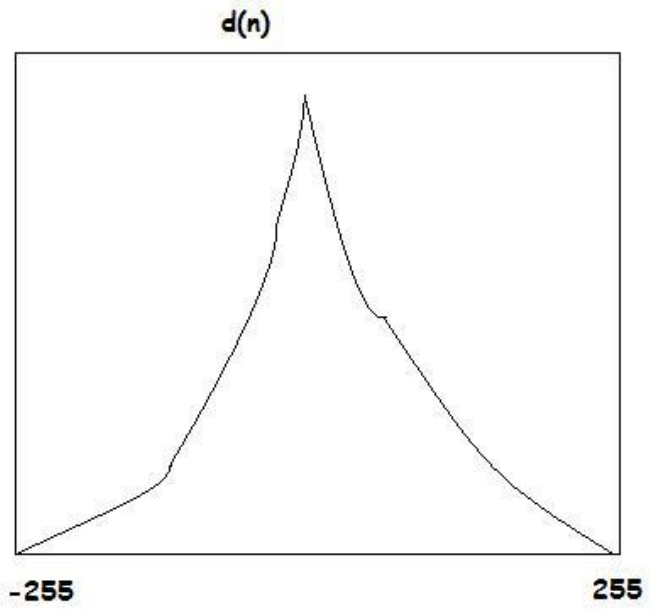


Fig.2.2 shows that it is easier to encode the difference rather than encoding the original sample because of less dynamic range.

2.6. Lossless Predictive Coding Technique

The principle behind predictive coding seems obvious when it is explained, but nothing has appeared in current literature providing a statement as to which filtering techniques can be used for lossless data compression.

There is no particular condition in specific for invertible filters but broad statements can be made about sufficient conditions for lossless filtering. The condition as given in [21] says "*The digital implementation of a filter is lossless if the output is the result of a digital one-to-one operation on the current sample and the recovery processor is able to construct the inverse operator".* It is important to note that the operations must be one-to-one not only on paper but also on the computer. Integer addition of the current sample and a constant is one-to-one under some amplitude constraints on all computer architectures.

Integer addition can be expressed as

$$e(n) = f_{of}\,(x(n) + s(n))$$

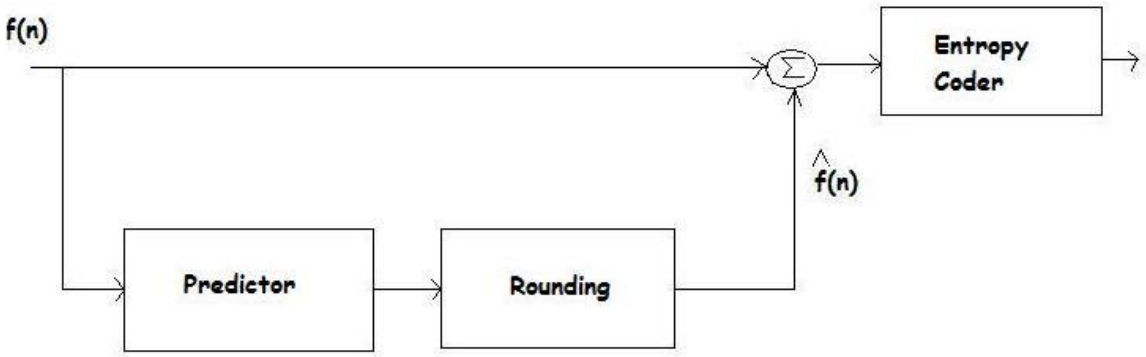
Where f_{of} is the overflow operator, *x (n)* is the current sample, *s (n)* an integer value which defines the transformation at time *n*, and *e (n)* is the current filter output.

The reverse operation is given by the equation

$$x(n) = f_{of}\,(e(n) - s(n))$$

This process always leads to an increase in number of bits required. To overcome this, rounding operation on the predictor output is performed making the predictor lossless. The lossless predictive encoder and decoder are shown in Fig.2.3 and Fig.2.4.

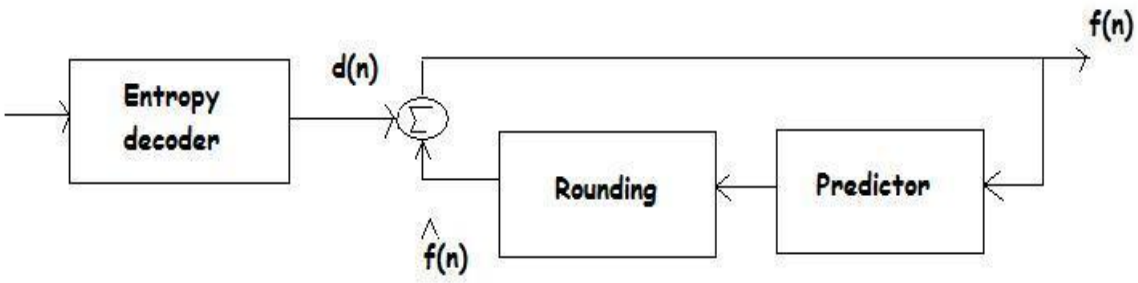
Figure 2.3 Predictive Encoder



Generally, the second order predictor is used which is also called Finite Impulse Response (FIR) filter. The simplest predictor is the previous value, in this experiment the predicted value is sum of the previous two values with alpha and beta being the predictor coefficients.

f^ (n) =<f (n-1)>

Figure 2.4 Predictive Decoder



In the process of predictive coding input image is passed through a predictor where it is predicted with its two previous values.

$$f^{\wedge}(n) = \alpha * f(n-1) + \beta * f(n-2)$$

f^(n) is the rounded output of the predictor, f(n-1) and f(n-2) are the previous values, α and β are the coefficients of the second order predictor ranging from 0 to 1. The output of the predictor is rounded and is subtracted from the original input. This difference is given by

$$d(n) = f(n) - f^{\wedge}(n)$$

Now this difference is given as an input to the decoder part of the predictive coding technique. In the decoding part the difference is added with the f^ (n) to give the original data.

$$f(n) = d(n) + f^{\wedge}(n)$$

Chapter 3 - Integer Wavelet Transform

Wavelet-based techniques are the latest development in the field of image compression. They offer multi-resolution capability that is not available in any other methods. The wavelet transform analyzes a signal in time and scale. The low frequency components in the signal that are spread out in time as well as the high frequency components that are localized in time are captured by a variable length window. The concept of wavelets in its present theoretical form was first proposed by Jean Morlet and the team at the Marseille Theoretical Physics Center working under Alex Grossmann in France. The main algorithm of wavelets was developed by Stephane Mallat in 1988.

3.1. Wavelet

Wavelets are small waves (unlike sine and cosine waves with infinite duration) with finite duration having finite energy and an average value of zero. It looks like a brief oscillation recorded by a seismograph or heart monitor. The wavelets posses' specific properties which make them much useful in signal processing. The process of convolution used in the wavelets typically involves a shift, multiply and sum technique to acquire information of the unknown signal by using the portions of the unknown signal. The wavelets can be used to attain information from any type of data that is not limited to only signals and images.

Generally sets of wavelets are needed to analyze data fully. A set of complementary wavelets can be used to get back the original data with least amount of loss in the data. This is the technique used in wavelet based compression and decompression algorithms.

The representation of a function f (t) using wavelet is given by the following formula

$$f(t) = \sum_{k}\sum_{j} \propto_{j,k} \Psi j,k(t)$$

Where, $\propto_{j,k}$ are the wavelet coefficients and Ψ (t) is called the mother wavelet. Equation for the wavelet coefficients is given by

$$\propto_{j,k} = \int_{-\infty}^{\infty} f(t)\ \psi_{j,k}\ (t)$$

3.2. Wavelet Transform

A wavelet transform is the representation of a function using wavelets called the daughter wavelets which are scaled and translated copies of the main oscillating waveform called the mother wavelet. These wavelet transforms have more advantages over the old Fourier transform. The wavelet transform can represent functions that have discontinuities and sharp peaks, and for accurately deconstructing and reconstructing finite, non-periodic and/or non-stationary signals along with periodic or stationary signals.

The wavelet transforms can be classified into Discrete Wavelet Transform (DWT) and a Continuous Wavelet Transform (CWT). Both of this wavelet transforms are analog transforms (continuous-time). CWTs operate on every scale and translation possible, whereas DWTs use a specific subset of scale and translation values or representation grid.

3.3. Continuous Wavelet Transform (CWT)

The CWT is used to divide a continuous time function into wavelets, this transform possess the ability to construct a time-frequency representation of a signal.

Mathematical representation of the CWT of a continuous wave x (t) at a scale $a > 0$ and translational value $b \in \mathbb{R}$ is given by

$$X_w(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x(t)\psi^* \left(\frac{t-b}{a}\right)\ dt$$

Where $\psi\ (t)$ represents a continuous function both in the time and frequency domain called the mother wavelet and $*$ represents operation of complex conjugate. The mother wavelet provides source function for the generation of the daughter wavelets.

3.4. Discrete Wavelet Transform (DWT)

In discrete wavelet transform the wavelets are discretely sampled. The DWT of a signal is calculated by passing the signal through series of filters called filter bank. The samples are simultaneously passed through a high pass filter and a low pass filter with impulse response *g* resulting in a convolution y[n]. Convolution is defined as the integral of the product of two functions after one is reversed and shifted. y[n] is the convolution of input signal with the impulse response.

The mathematical equation for y[n] is given by

$$y[n] = (x * g)[n] = \sum_{k=-\infty}^{\infty} x[k]g[n-k].$$

Figure 3.1 Block Diagram of Filter Analysis

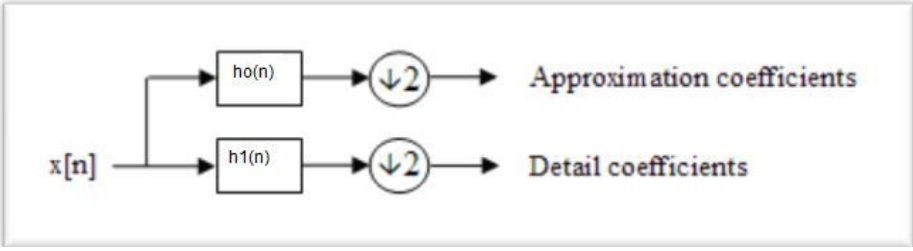
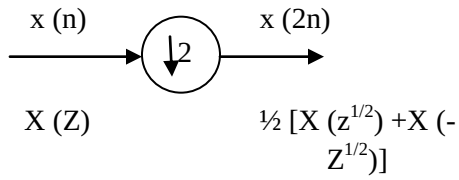
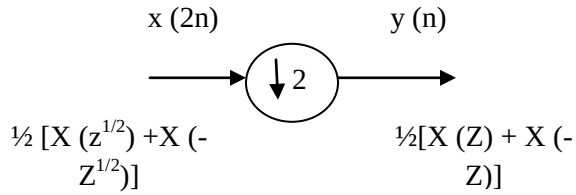


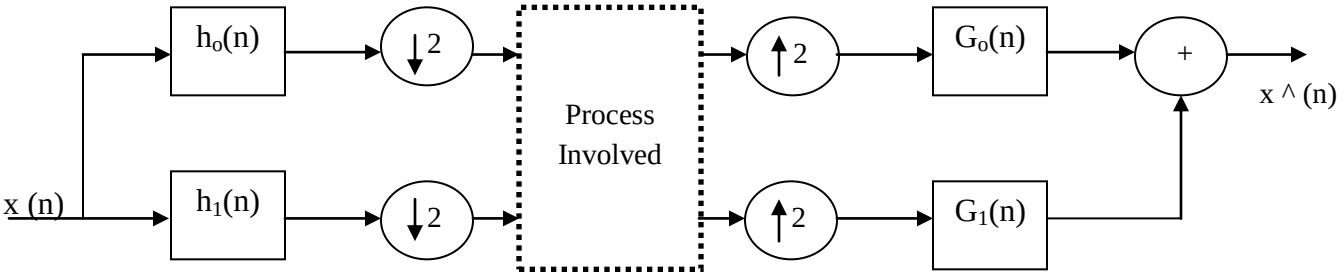
Fig.3.1 represent the DWT of a signal where x (n) is the input signal, h_o (n) is the low pass filter and h_1 (n) is the high pass filter. The output of the low pass filter will be approximate coefficients and the outputs of the high pass filter will be the detailed coefficients of the signal. The LPF and the HPF are related to each other and the relationship between these filters means that a perfect reconstruction is possible. That is, the two bands can then be upsampled, filtered again with the same filters and added together to reproduce the original signal exactly. The conditions for the perfect reconstruction are derived in frequency domain to avoid complex computations in time domain. The Z transforms representation of the input the downsampled and the upsampled signal are given by





The block diagram of subband coding is used to derive the conditions for perfect reconstruction.

Figure 3.2 Block Diagram for Subband Coding



The downsampled outputs are multiplied with the z transform of LPF and HPF called the analysis filters giving two different equations

$$\tfrac{1}{2} [H_0 (Z) X (Z) + H_0 (-Z) X (-Z)] \quad (1)$$

$$\tfrac{1}{2} [H_1 (Z) X (Z) + H_1 (-Z) X (-Z)] \quad (2)$$

The final output $X^{\wedge}(Z)$ is the sum of equations (1) and (2) multiplied by the synthesis filters and the input which is given by

$$X^{\wedge}(Z) = \tfrac{1}{2} [H_0 (Z) G_0 (Z) + H_1 (Z) G_1 (Z)] X (Z) + \tfrac{1}{2} [H_0 (-Z) G_0 (Z) + H_1 (-Z) G_1 (Z)] X (-Z) \quad (3)$$

In the case of perfect reconstruction the input and the output are the same therefore

$$X^{\wedge}(Z) = X (Z)$$

$$X^{\wedge}(Z) = \alpha Z^{-\beta} X (Z) \quad (4)$$

Where α is the scaling and β is the delay

$$[H_0 (-Z) G_0 (Z) + H_1 (-Z) G_1 (Z)] = 0 \quad (5)$$

Equation (5) is called an anti-aliasing condition, placing equation (5) in equation (3) we get

$$\tfrac{1}{2}\,[H_0(Z)\,G_0(Z) + H_1(Z)\,G_1(Z)] = \alpha\, Z^{-\beta}\, X(Z) \quad (6)$$

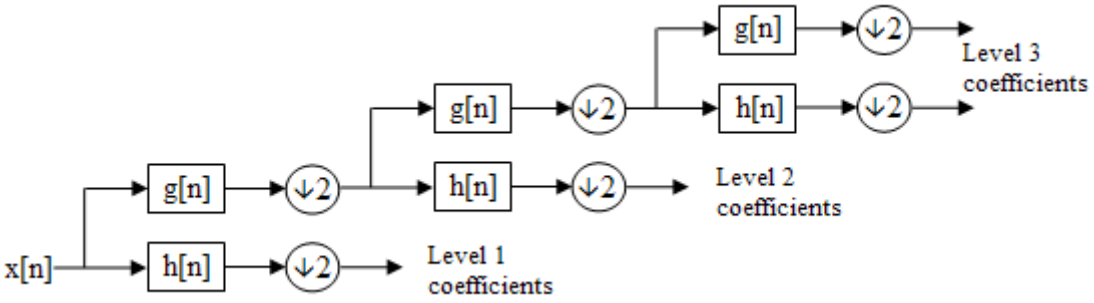
Filters used in DWT are called QMF (Quadrature Mirror Filters). A QMF gives critically sampled two-channel representation of the original signal. Two conditions that satisfy the QMF based on the perfect reconstruction described in equations (5) and (6) are

$$H_1(z) = H_0(-z)$$

$$H_0^2(z) - H_1^2(z) = \alpha\, Z^{-\beta}$$

The signal is further decomposed into high and low pass filters using filter banks at each level. This increases the frequency resolution of the signal. Due to so many decomposition involved the input signal must be a multiple of 2^n where *n* is the number of levels.

Figure 3.3 A 3 level filter bank



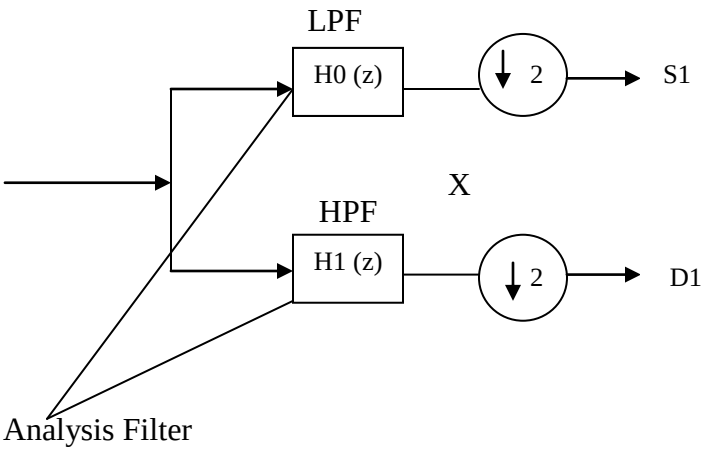
3.5. Implementation of Integer Wavelet Transform

In integer wavelet transform there is a mapping between integers to integers. In this section different ways of implementing integer wavelet transform have been discussed.

3.5.1. Implementation Using Filter Bank

In signal processing, a filter bank is an array of band-pass filters that separates the input signal into multiple components each one carrying a single frequency subband of the original signal. The process of decomposition performed by the filter bank is called *analysis* and the output of analysis is referred to as a subband signal with as many subbands as there are filters in the filter bank. The reconstruction process is called *synthesis*.

Figure 3.4 Filter Bank Implementation



After performing decomposition, the important frequencies can be coded with a fine resolution. The minimum requirement of the filters both the analysis filters and the synthesis filters are achieved by using the analysis filters and are derived from the perfect reconstruction conditions mentioned in Section 3.4.

$H_0(z)$ is chosen by the QMF rule there fore

$$H_1(z) = H_0(-z)$$

Now the synthesis filters will be given by

$$G_0(z) = H_0(z) \text{ and}$$

$$G_1(z) = -H_0(-z)$$

3.5.2. Lifting Scheme

The simplest lifting scheme is the lazy wavelet transform, where the input signal is first split into *even* and *odd* indexed samples.

$$(odd_{j-1}, even_{j-1}) = Split(s_j)$$

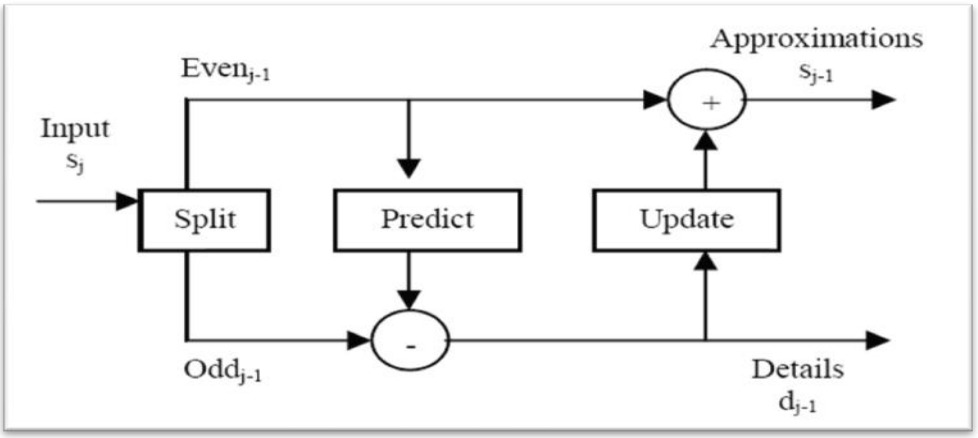
The samples are correlated, so it is possible to predict odd samples from even samples which in the case of Haar transform are even values themselves. The difference between the actual odd samples and the prediction becomes the wavelet coefficients. The operation of obtaining the differences from the prediction is called the lifting step. The update step follows the prediction step, where the even values are updated from the input even samples and the updated odd samples. They become the scaling coefficients which will be passed on to the next stage of transform. This is the second lifting step.

$$d_{j-1} = odd_{j-1} - P(even_{j-1})$$

$$s_{j-1} = even_{j-1} + U(d_{j-1})$$

Finally the odd elements are replaced by the difference and the even elements by the averages. The computations in the lifting scheme are done in place which saves lot of memory and computation time. The lifting scheme provides integer coefficients and so it is exactly reversible. The total number of coefficients before and after the transform remains the same.

Figure 3.5 Forward Lifting Scheme [16]

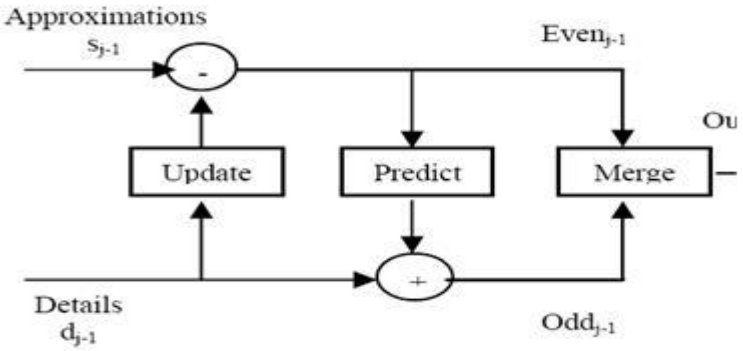


The inverse transform gets back the original signal by exactly reversing the operations of the forward transform with a merge operation in place of a split operation. The number of

samples in the input signal must be a power of two, and these samples are reduced by half in each succeeding step until the last step which produces one sample.

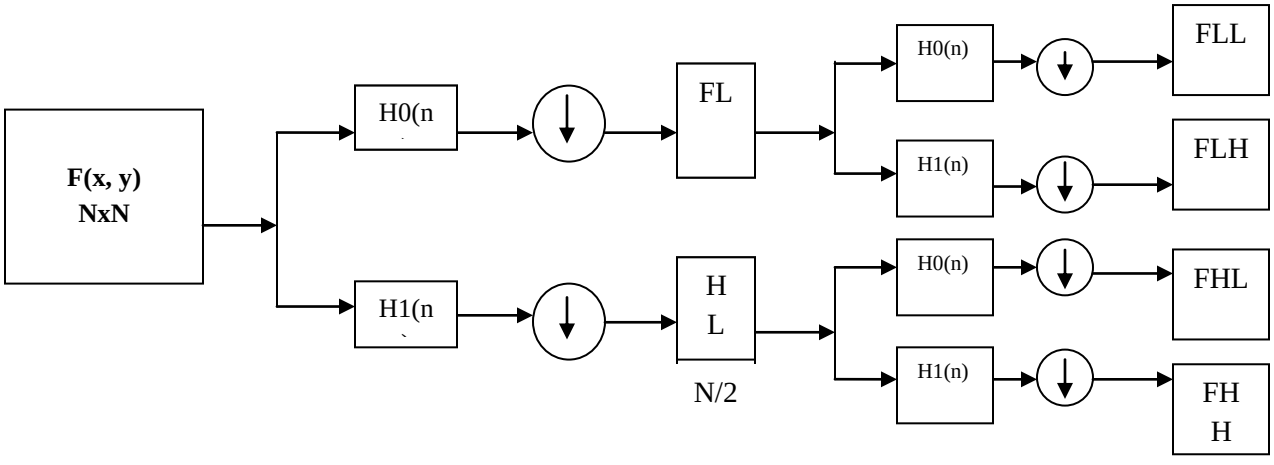
Figure 3.6 Inverse Lifting Scheme [16]

$$Even_{j-1} = s_{j-1} - U(d_{j-1})$$
$$Odd_{j-1} = d_{j-1} + P(Even_{j-1})$$
Finally, $$s_j = Merge(Even_{j-1}, Odd_{j-1})$$



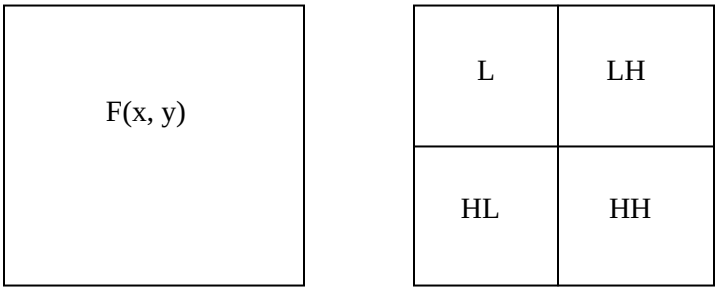
The Haar wavelet transform uses predict and update operations of order one. Using different predict and update operations of higher order, many other wavelet transforms can be built using the lifting scheme.

Figure 3.7 Steps for Decomposition Using Lifting



Basic steps involved in the decomposition are illustrated in Fig.3.7. Firstly the image/signal is sent through a low pass and band pass filter simultaneously (predict and update in case of lifting) and down sampled by a factor of 2. The process is repeated and the final four outputs are combined to from the transformed image as shown in Fig.3.8.

Figure 3.8 Input and Outputs of Lifting Scheme



Original Image Transformed Image

The transformed image in Fig.3.8 shows different subbands of which the first subband is called LL which represents the low resolution version of the image, the second subband is called LH which represents the horizontal fluctuations, the third band is called the HL which represents the vertical fluctuations, and the fourth subband is called the HH which represents the diagonal fluctuations. Same procedure can be followed to obtain different levels of image decomposition by changing the inputs given to the lifting or filter bank implementation techniques.

Chapter 4 – Implementation and Experimental Results

In this report the Integer Wavelet Transform (IWT) and the Predictive Coding Techniques are used to perform lossless image compression. The performance of the proposed techniques is calculated by finding the Entropy and scaled entropy of the compressed image. The performance is also measured using compression ratio which is given by the ratio of the bits in the original uncompressed data to the number of bits in the compressed data.

4.1. Procedure

The procedure of the implementation involves four methods of performing compression on the medical image. In the first method IWT is performed first followed by predictive coding technique on the transformed image. In the second method the predictive coding technique is applied first followed by the integer wavelet transform. The third method involves reduction of the filter coefficients by a factor of 3/2 and then applying integer wavelet transform followed by predictive coding technique. The fourth method also used the reduced filter coefficients and performing predictive coding followed by integer wavelet transform. All these methods use Haar filter in the lifting scheme and the filter coefficients are given by

h1= [-1 9 9 1]/ (16);

h2= [0 0 1 1]/ (-4);

Where h1 are the prediction filter coefficients and h2 are the update filter coefficients in the lifting scheme.

The reduced filter coefficients are given by

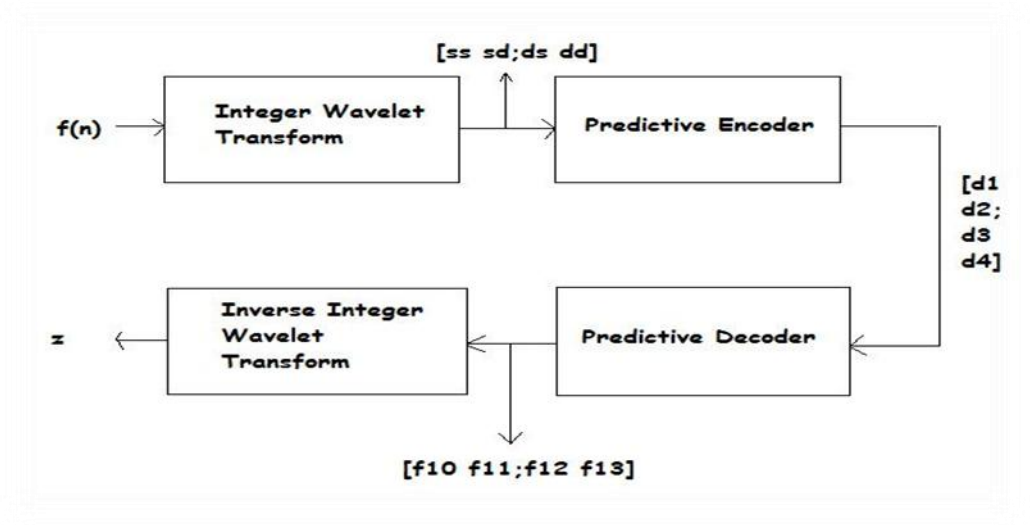
h1= [-1 9 9 1]/ (16*1.5);

h2= [0 0 1 1]/ (-4*1.5);

The implementation of all the four methods for five different medical images is also shown in the results. We implemented all the four methods using two other filters in lifting scheme called Daubechies 2 and Daubechies 3 and the results are presented.

4.2. Implementation Using Method 1

Figure 4.1 Block Diagram for IWT Followed by Predictive Coding



In this method integer wavelet transform is applied on the image which divides the image into four subbands ss, sd, ds, dd. Now predictive coding is applied on the four different bands separately giving outputs d1, d2, d3 and d4. The reconstruction process involves applying the predictive decoding followed by inverse integer transform. The reconstructed image is represented by z. To verify the perfect reconstruction the original and the reconstructed images are subtracted and the output is a dark image with maximum and minimum values as zero.

4.2.1. Outputs of Method 1

Figure 4.2 Original Image

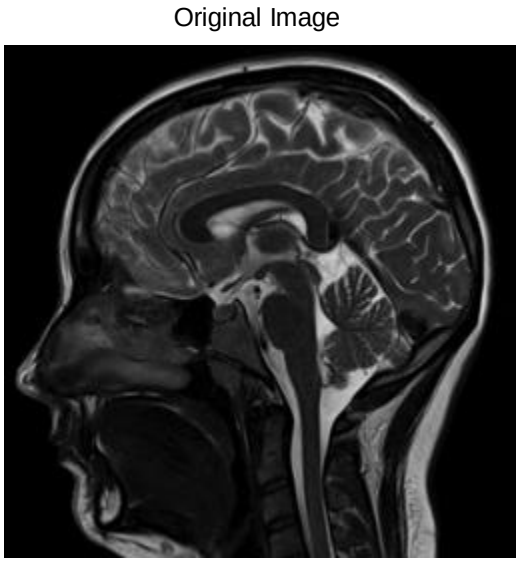


Figure 4.3 Image Obtained after Subband Coding

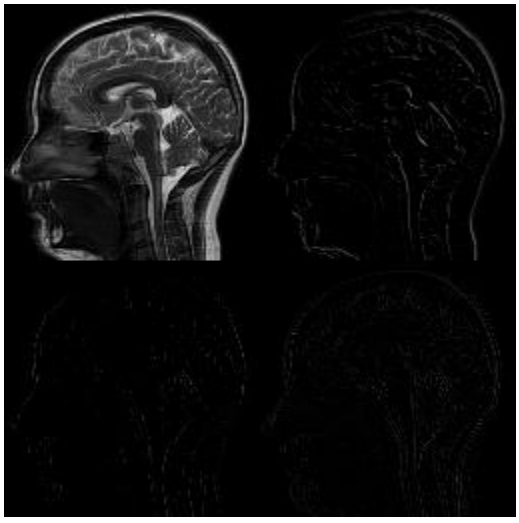


Figure 4.4 Encoded Image

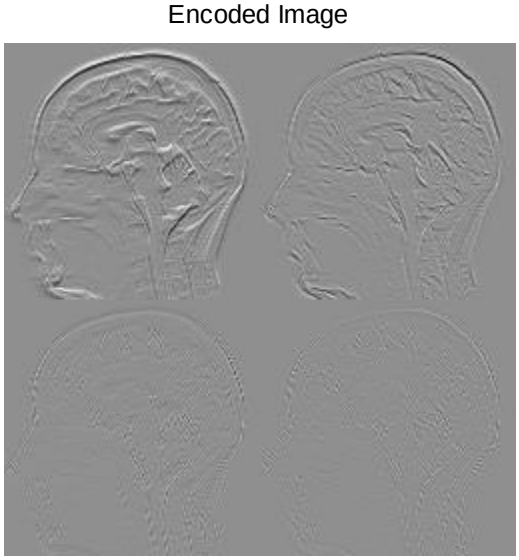


Figure 4.5 Decoded Image

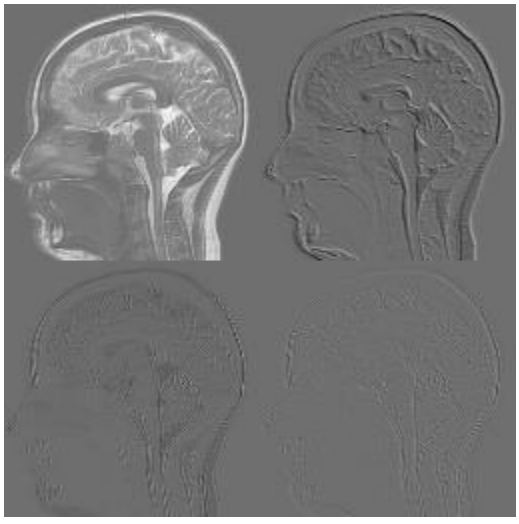
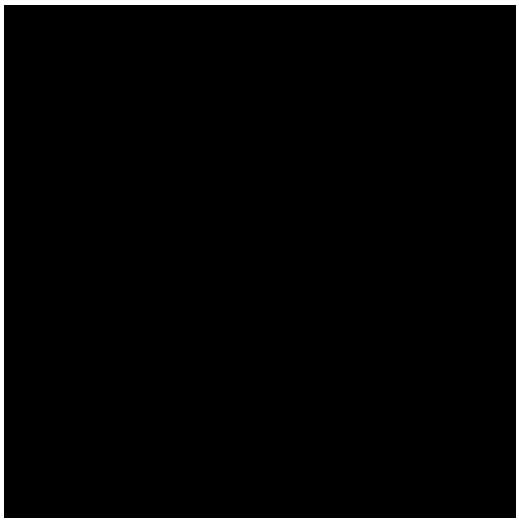


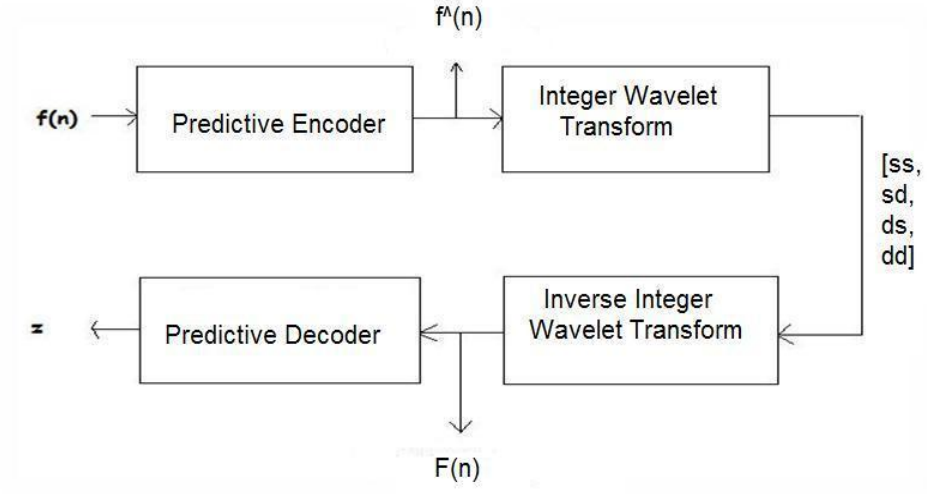
Figure 4.6 Reconstructed Image

Figure 4.7 Difference of the Original and Reconstructed Image



4.3. Implementation Using Method 2

Figure 4.8 Block Diagram for Predictive Coding Followed by IWT



In this method predictive coding is applied on the image first, this converts the image into f^ (n). Now on this output integer wavelet transform is applied which divides the image into four subbands ss, sd, ds, dd. The reconstruction process involves applying the inverse integer wavelet transform on the transformed image followed by applying predictive decoding on the output of the inverse transform F (n). The reconstructed image is represented by z.

4.3.1. Outputs of Method 2

Figure 4.9 Original Image

Figure 4.10 Predictive Encoder

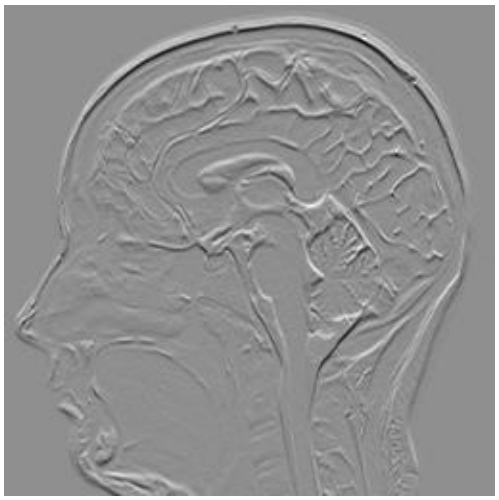


Figure 4.11 Image Obtained after Subband Coding

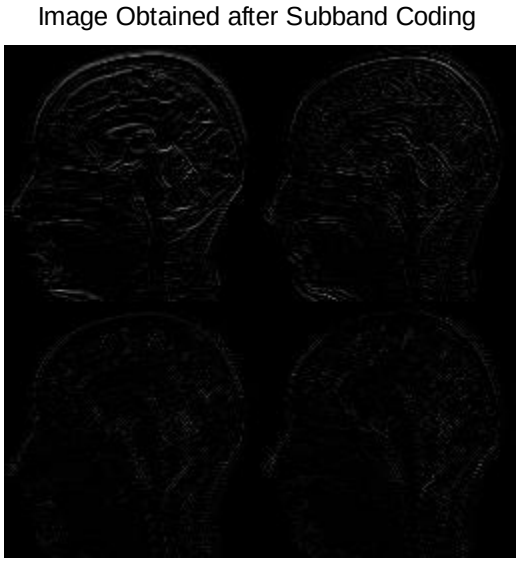


Figure 4.12 Image after Inverse IWT

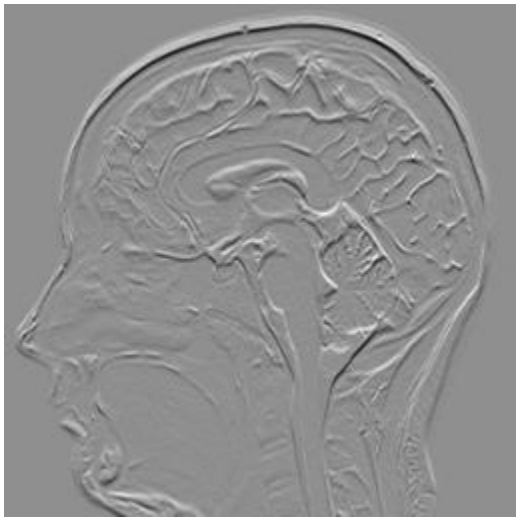
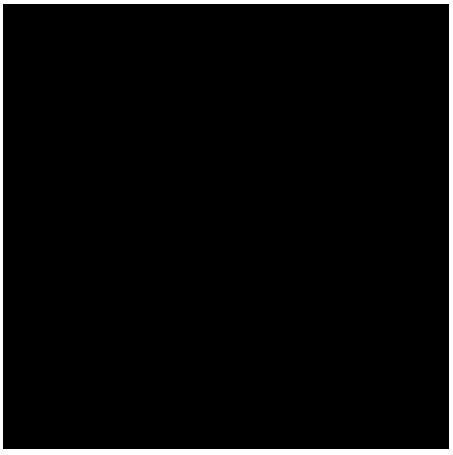


Figure 4.13 Predictive Decoder

Figure 4.14 Difference of Original and Reconstructed Image



4.4. Implementation Using Method 3

In this method the filter coefficients used in the integer wavelet transform using lifting scheme are reduced by a factor of 3/2 and the same steps mentioned in Section 4.2 are performed.

4.4.1. Outputs of Method 3

Figure 4.15 Original Image

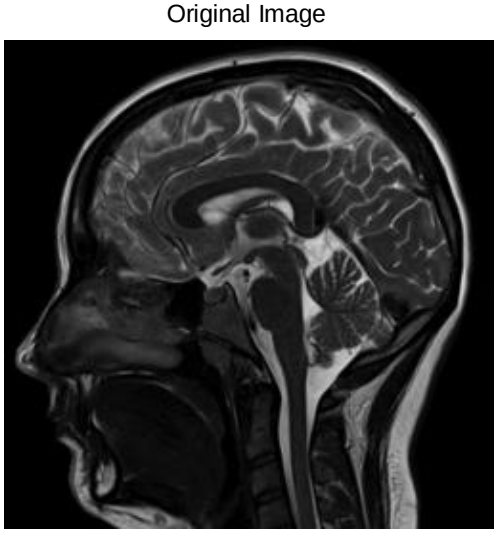


Figure 4.16 Image Obtained after Sub band Coding

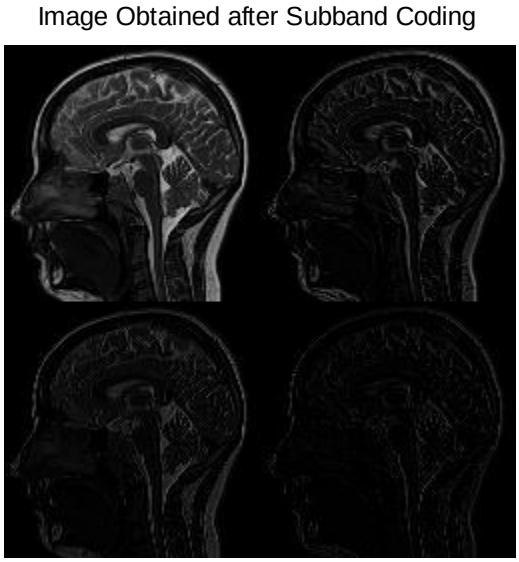


Figure 4.17 Encoder Image

Figure 4.18 Decoded Image

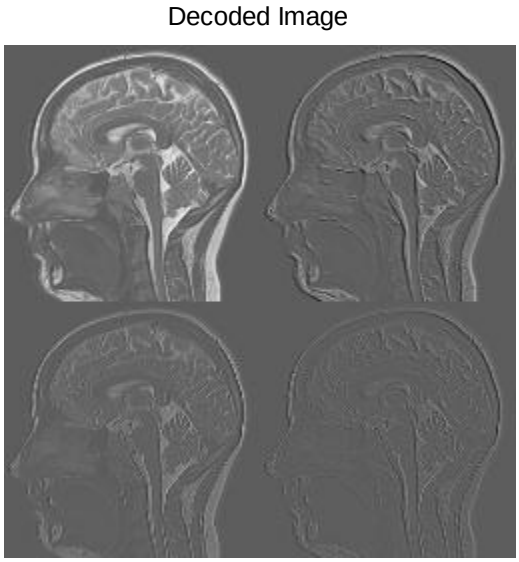
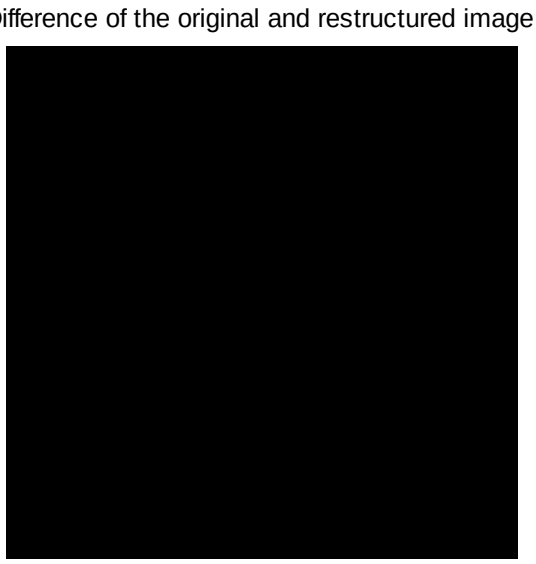


Figure 4.19 Reconstructed Image

Figure 4.20 Difference of the Original and Reconstructed Image



4.5. Implementation Using Method 4

In this method also the filter coefficients are reduced by a factor of 3/2 and same steps mentioned in Section 4.3 are implemented.

4.5.1. Outputs of Method 4

Figure 4.21 Original Image

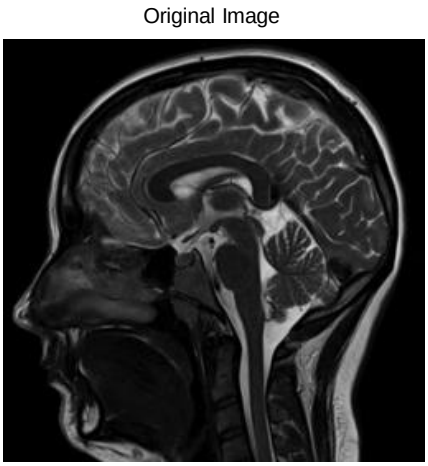


Figure 4.22 Predictive Encoder

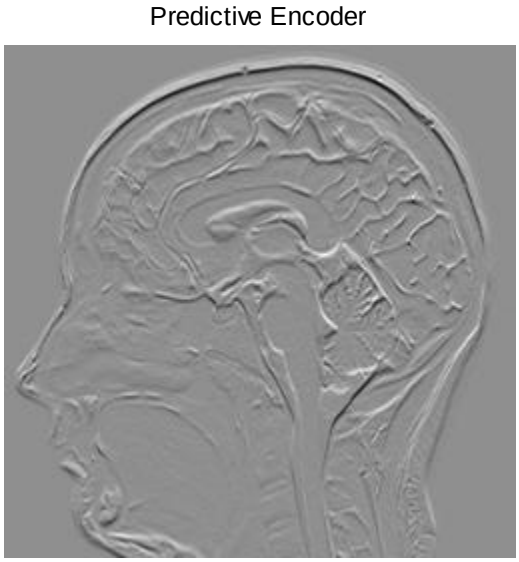


Figure 4.23 Image Obtained after Sub Band Coding

Figure 4.24 Image after Inverse IWT

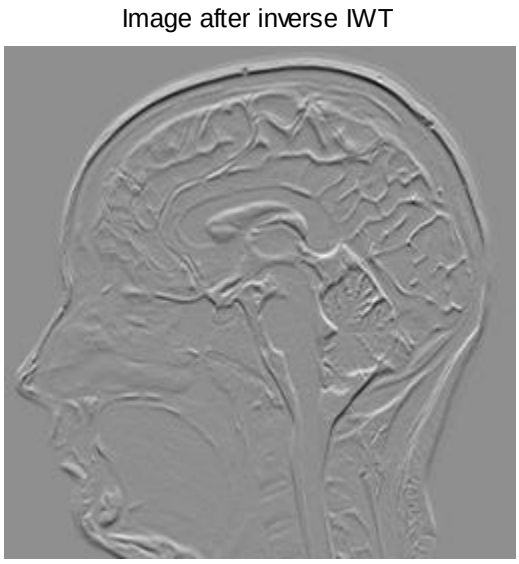


Figure 4.25 Predictive Decoder

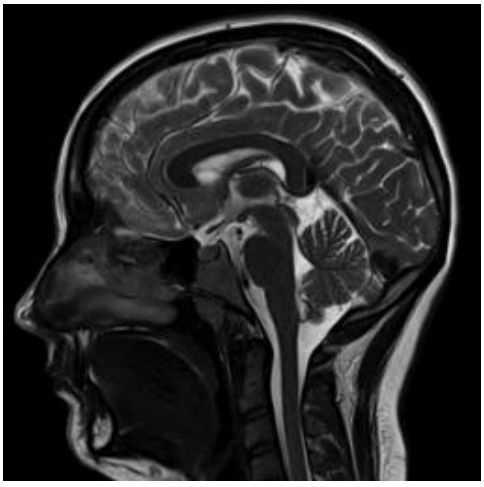


Figure 4.26 Difference of Original and Reconstructed Image

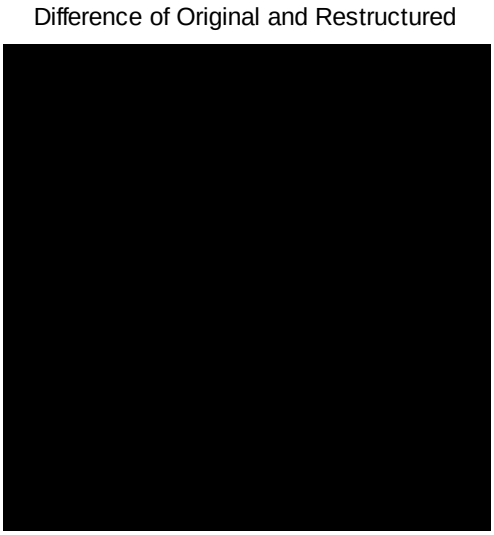


Table 4.1 Tabulations of Method 1 using different values of alpha and beta

S.NO	ALPHA	BETA	ORIGINAL ENTROPY	ENTROPY AFTER WAVELET TRANSFORM	ENTROPY AFTER PREDICTIVE CODING	SCALED ENTROPY
1	0.1	0.1	5.9604	5.6760	5.5322	5.1572
2	0.3	0.3	5.9604	5.6760	5.2844	5.0973
3	0.5	0.5	5.9604	5.6760	5.2578	5.1930
4	0.7	0.7	5.9604	5.6760	5.6104	5.4533
5	0.9	0.9	5.9604	5.6760	5.9678	5.7244
6	0.1	0.01	5.9604	5.6760	5.5971	5.1836
7	0.01	0.1	5.9604	5.6760	5.6076	5.2026
8	0.01	0.01	5.9604	5.6760	5.6672	5.2258

Table 4.2 Tabulations of Method 2 using different values of alpha and beta

S.NO	ALPHA	BETA	ORIGINAL ENTROPY	ENTROPY AFTER WAVELET TRANSFORM	ENTROPY AFTER PREDICTIVE CODING	SCALED ENTROPY
1	0.1	0.1	5.9604	5.4356	5.3224	5.1648
2	0.3	0.3	5.9604	5.4356	5.1512	5.0663
3	0.5	0.5	5.9604	5.4356	5.1896	5.1450
4	0.7	0.7	5.9604	5.4356	5.4968	5.4177
5	0.9	0.9	5.9604	5.4356	5.8104	5.6996
6	0.1	0.01	5.9604	5.4356	5.3673	5.1947
7	0.01	0.1	5.9604	5.4356	5.3897	5.2197
8	0.01	0.01	5.9604	5.4356	5.4299	5.2478

Table 4.3 Tabulations of Method 3 using different values of alpha and beta

SN0	ALPHA	BETA	ORIGINAL ENTROPY	ENTROPY AFTER PREDICTIVE	ENTROPY AFTER IWT	SCALED ENTROPY
1	0.1	0.1	5.9604	5.8193	5.4969	5.1020
2	0.3	0.3	5.9604	5.4284	5.1257	4.8988
3	0.5	0.5	5.9604	5.0654	4.9148	4.8723
4	0.7	0.7	5.9604	5.6886	5.4011	5.2283
5	0.9	0.9	5.9604	6.1320	5.8337	5.5640
6	0.1	0.01	5.9604	5.8594	5.5705	5.1422
7	0.01	0.1	5.9604	5.9145	5.5953	5.1802
8	0.01	0.01	5.9604	5.9487	5.6614	5.2185

Table 4.4 Tabulations of Method 4 using different values of alpha and beta

SN0	ALPHA	BETA	ORIGINAL ENTROPY	ENTROPY AFTER PREDICTIVE	ENTROPY AFTER IWT	SCALED ENTROPY
1	0.1	0.1	5.9604	5.8193	5.2627	5.0966
2	0.3	0.3	5.9604	5.4284	4.9286	4.8317
3	0.5	0.5	5.9604	5.0654	4.7891	4.7589
4	0.7	0.7	5.9604	5.6886	5.2158	5.1371
5	0.9	0.9	5.9604	6.1320	5.6163	5.4999
6	0.1	0.01	5.9604	5.8594	5.3276	5.1500
7	0.01	0.1	5.9604	5.9145	5.3591	5.1863
8	0.01	0.01	5.9604	5.9487	5.4234	5.2389

4.5.2. Comparing all the four methods using different Images

Figure 4.27 MRI of Brain

Figure 4.28 Iris Image

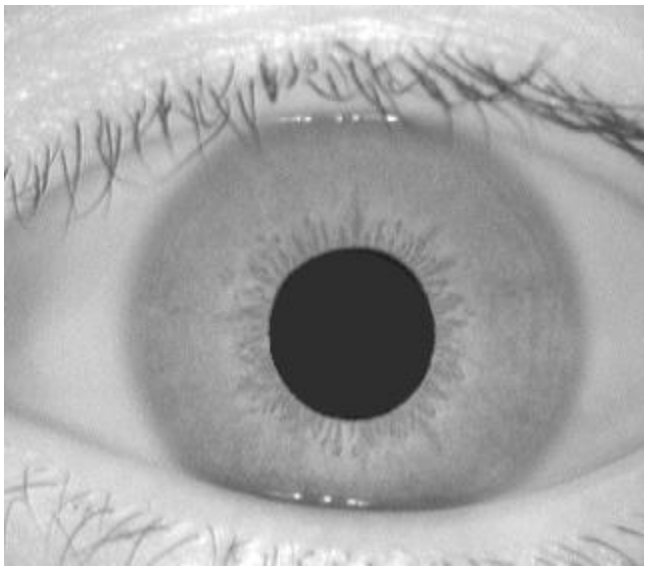


Figure 4.29 Nasal Fracture

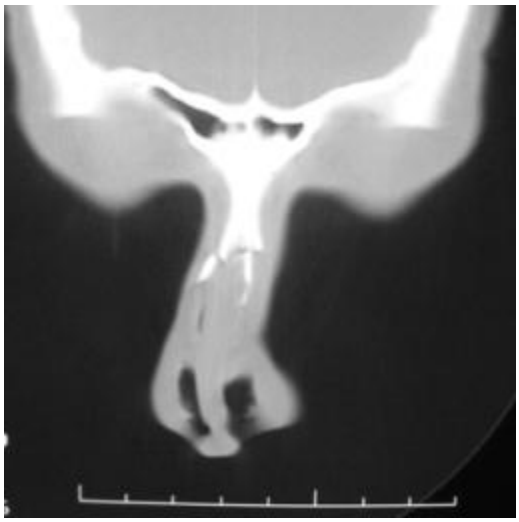


Figure 4.30 Throat Cancer

Figure 4.31 Nasal Fracture 2

Table 4.5 Comparison Table for different images using method 1

Image	Original Entropy	Entropy after IWT	Entropy after Predictive Coding	Scaled Entropy
Brain	5.9604	5.6760	5.2844	5.0973
Iris1	6.8416	7.0029	5.4107	5.3845
Nasal Fracture	6.4294	6.3516	4.0988	3.9990
Throat Cancer	7.2322	6.8841	5.2829	5.1932
Nasal Fracture 2	6.4444	6.5342	4.5150	4.4520

Table 4.6 Comparison Table for different images using method 2

Image	Original Entropy	Entropy after Predictive Coding	Entropy after IWT	Scaled Entropy
Brain	5.9604	5.0654	4.9148	4.8723
Iris1	6.8416	5.0425	5.2760	5.2556
Nasal Fracture	6.4294	3.9267	3.7370	3.6634
Throat Cancer	7.2322	5.1922	4.9891	4.9121
Nasal Fracture 2	6.4444	4.1847	4.1980	4.1576

Table 4.7 Comparison Table for different images using method 3

Image	Original Entropy	Entropy after IWT	Entropy after Predictive Coding	Scaled Entropy
Brain	5.9604	5.4356	5.1512	5.0663
Iris1	6.8416	6.8988	5.2047	5.1797
Nasal Fracture	6.4294	6.1483	4.0103	3.9568
Throat Cancer	7.2322	6.6987	5.2151	5.1505
Nasal Fracture 2	6.4444	6.3369	4.3953	4.3570

Table 4.8 Comparison Table for different images using method 4

Image	Original Entropy	Entropy after Predictive Coding	Entropy after IWT	Scaled Entropy
Brain	5.9604	5.0654	4.7891	4.7589
Iris1	6.8416	5.0425	4.9817	4.9632
Nasal Fracture	6.4294	3.9267	3.5275	3.4855
Throat Cancer	7.2322	5.1922	4.8434	4.7889
Nasal Fracture 2	6.4444	4.1847	3.9952	3.9681

4.5.3. Comparing the four methods using different filters

The other filters used are Daubechies 2 and Daubechies 3 in the lifting scheme. Daubechies 2 is an orthogonal wavelet with two vanishing moments and Daubechies 3 filter is also an orthogonal wavelet with 3 vanishing moments.

The coefficients of Daubechies 2 are given by

$$h1= [(1 + \sqrt{3})/4\sqrt{2}, (3 + \sqrt{3})/4\sqrt{2}, (3 - \sqrt{3})/4\sqrt{2}, (1 - \sqrt{3})/4\sqrt{2}]$$

$$h2= [-(1 - \sqrt{3})/4\sqrt{2}, (3 - \sqrt{3})/4\sqrt{2}, (3 + \sqrt{3})/4\sqrt{2}, (1 + \sqrt{3})/4\sqrt{2}]$$

The coefficients of Daubechies 3 are given by

h1= [0.3327 0.8069 0.4600 -0.1350 -0.085 0.0352]

h2 = [-0.0352 -0.085 0.1350 0.4600 -0.8069 0.3327]

Table 4.9 Comparison Table for different images using Daubechies 2 filter in method 1

Image	Original Entropy	Entropy after IWT	Entropy after Predictive Coding	Scaled Entropy
Brain	5.9604	5.8990	5.5205	5.4986
Iris1	6.8416	7.3438	5.3810	5.3619
Nasal Fracture	6.4294	6.9186	4.3993	4.3655
Throat Cancer	7.2322	7.2054	5.6088	5.5627
Nasal Fracture 2	6.4444	6.9887	4.7088	4.6776

Table 4.10 Comparison Table for different images using Daubechies 2 filter in method 2

Image	Original Entropy	Entropy after IWT	Entropy after Predictive Coding	Scaled Entropy
Brain	5.9604	5.0654	5.0273	5.0107
Iris1	6.8416	5.0425	4.9678	4.9504
Nasal Fracture	6.4294	3.9267	3.8059	3.7778
Throat Cancer	7.2322	5.1992	5.0953	5.0539
Nasal Fracture 2	6.4444	4.1897	4.1725	4.1512

Table 4.11 Comparison Table for different images using Daubechies 2 filter in method 3

Image	Original Entropy	Entropy after IWT	Entropy after Predictive Coding	Scaled Entropy
Brain	5.9604	5.8404	5.5227	5.4889
Iris1	6.8416	7.4059	5.3699	5.3437
Nasal Fracture	6.4294	6.7487	4.3438	4.3025
Throat Cancer	7.2322	7.0520	5.5766	5.5175
Nasal Fracture 2	6.4444	6.9194	4.7126	4.6789

Table 4.12 Comparison Table for different images using Daubechies 2 filter in method 4

Image	Original Entropy	Entropy after IWT	Entropy after Predictive Coding	Scaled Entropy
Brain	5.9604	5.0654	5.0204	5.0077
Iris1	6.8416	5.0425	4.9736	4.9562
Nasal Fracture	6.4294	3.9267	3.8226	3.7986
Throat Cancer	7.2322	5.1922	5.1103	5.0736
Nasal Fracture 2	6.4444	4.1847	4.1533	4.1344

Table 4.13 Comparison Table for different images using Daubechies 3 filter in method 1

Image	Original Entropy	Entropy after IWT	Entropy after Predictive Coding	Scaled Entropy
Brain	5.9604	6.1303	5.7089	5.6927
Iris1	6.8416	7.3861	5.4636	5.4417
Nasal Fracture	6.4294	6.9672	4.5255	4.4975
Throat Cancer	7.2322	7.2774	5.7492	5.7119
Nasal Fracture 2	6.4444	7.0513	4.8399	4.8143

Table 4.14 Comparison Table for different images using Daubechies 3 filter in method 2

Image	Original Entropy	Entropy after IWT	Entropy after Predictive Coding	Scaled Entropy
Brain	5.9604	5.0654	5.1926	5.1822
Iris1	6.8416	5.0425	5.0055	4.9895
Nasal Fracture	6.4294	3.9267	3.9046	3.8815
Throat Cancer	7.2322	5.1992	5.2270	5.1936
Nasal Fracture 2	6.4444	4.1897	4.2724	4.2543

Table 4.15 Comparison Table for different images using Daubechies 3 filter in method 3

Image	Original Entropy	Entropy after IWT	Entropy after Predictive Coding	Scaled Entropy
Brain	5.9604	6.1140	5.6658	5.6521
Iris1	6.8416	7.2264	5.4841	5.4629
Nasal Fracture	6.4294	6.8515	4.5225	4.4969
Throat Cancer	7.2322	7.2990	5.7402	5.7013
Nasal Fracture 2	6.4444	6.9912	4.8073	4.7826

Table 4.16 Comparison Table for different images using Daubechies 3 filter in method 4

Image	Original Entropy	Entropy after IWT	Entropy after Predictive Coding	Scaled Entropy
Brain	5.9604	5.0654	5.1356	5.1259
Iris1	6.8416	5.0425	5.0044	4.9876
Nasal Fracture	6.4294	3.9267	3.8853	3.8623
Throat Cancer	7.2322	5.1922	5.1865	5.1548
Nasal Fracture 2	6.4444	4.1847	4.2209	4.2030

4.6. Graphs

Figure 4.32 Graph between predictors and scaled entropy for method 1 and method 3

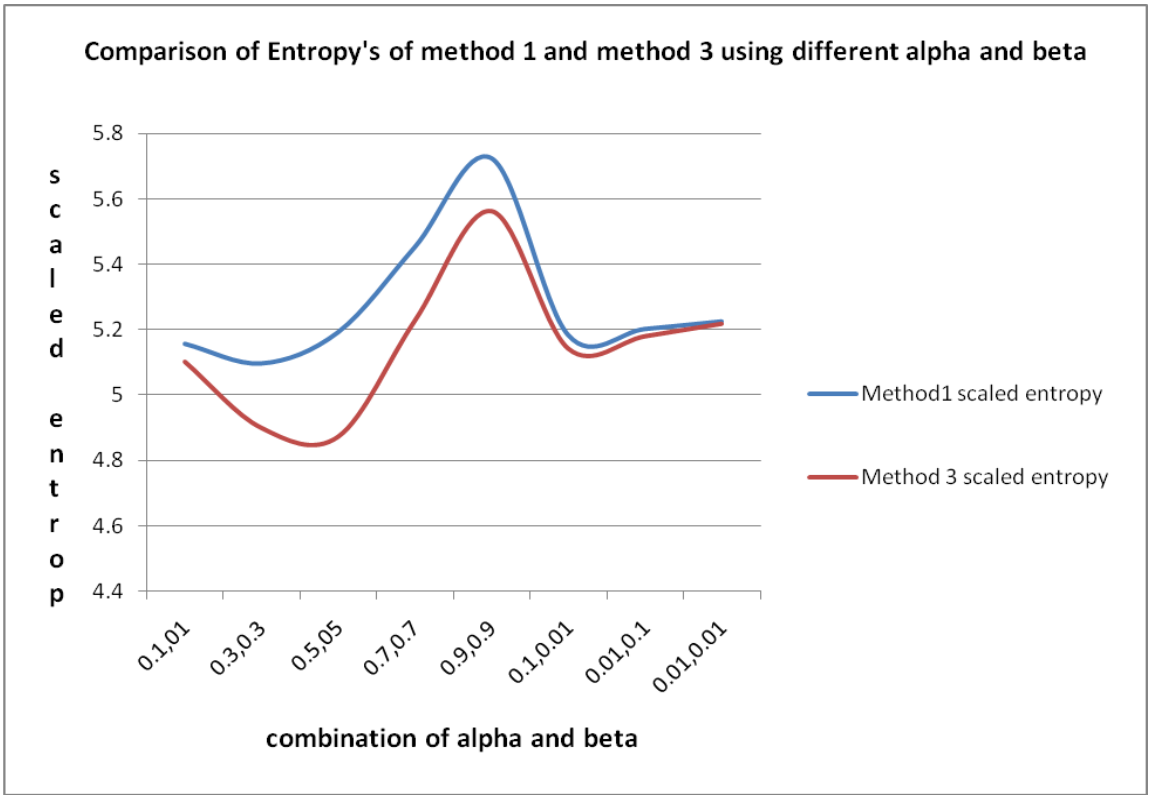


Figure 4.33 Graph between predictors and scaled entropy for method 2 and method 4

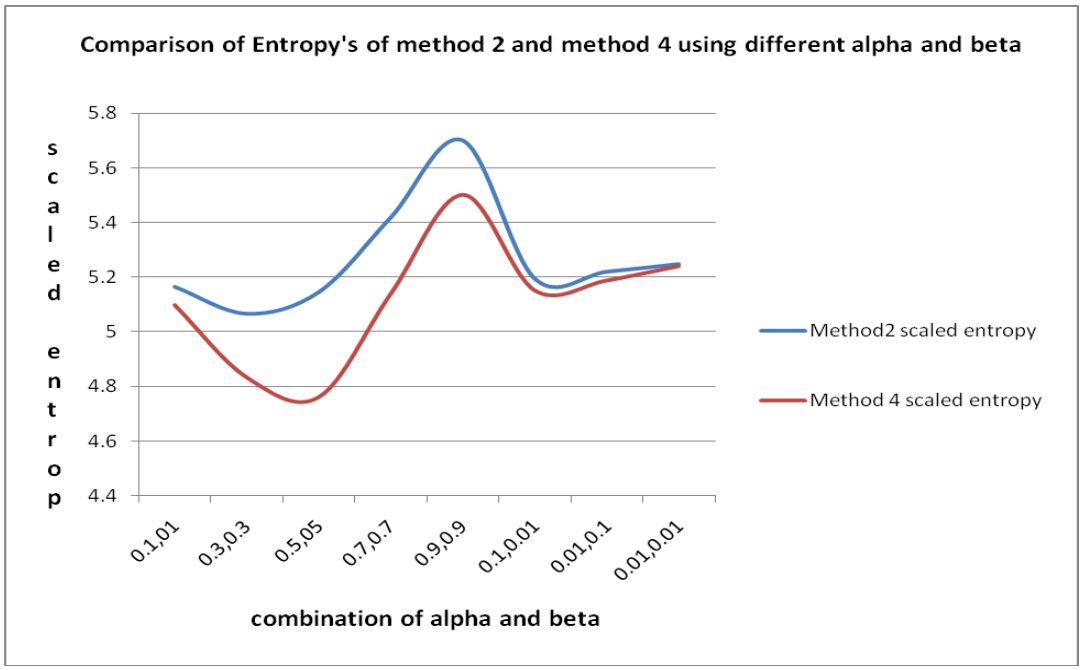


Figure 4.34 Comparing the behavior of different images with alpha and beta values as 0.5

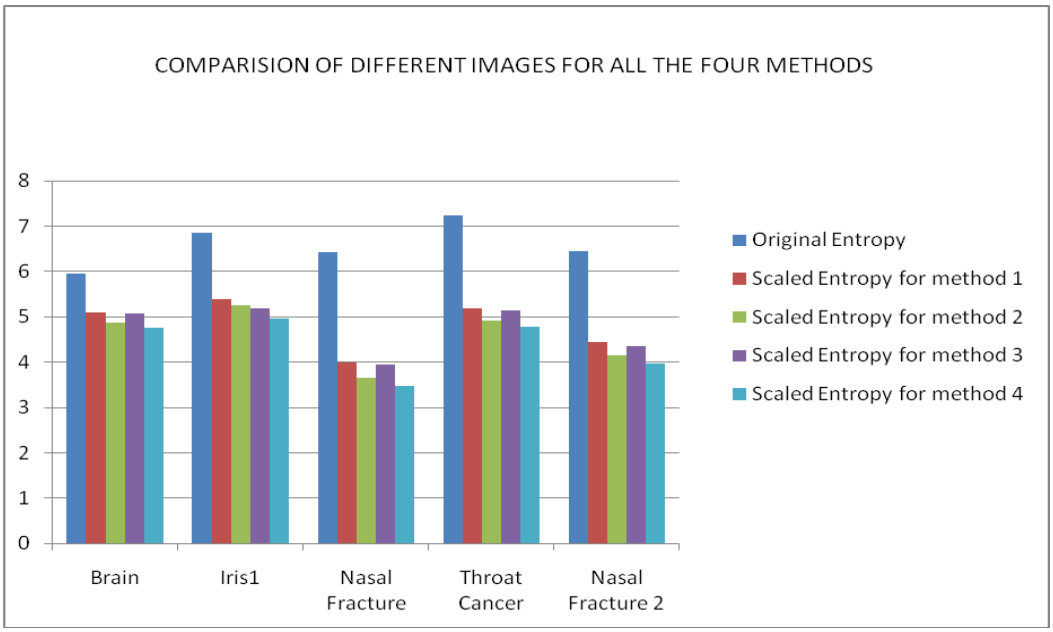


Figure 4.35 Behavior of different images using Daubechies 2 filter in all the four methods

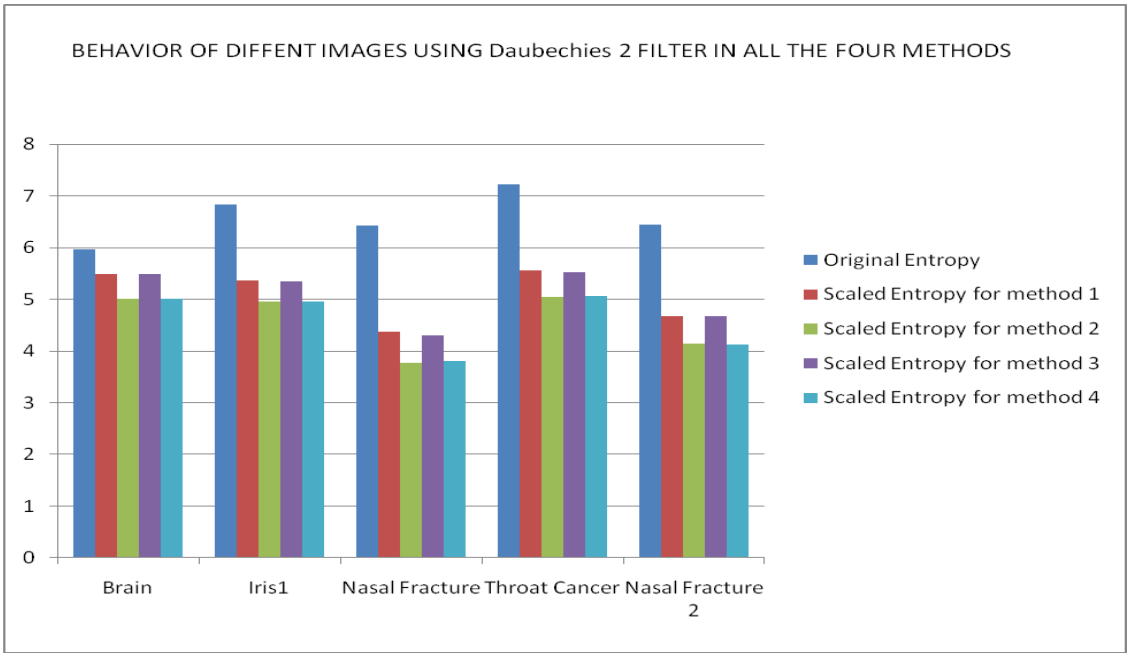


Figure 4.36 Behavior of different images using Daubechies 3 filter in all the four methods

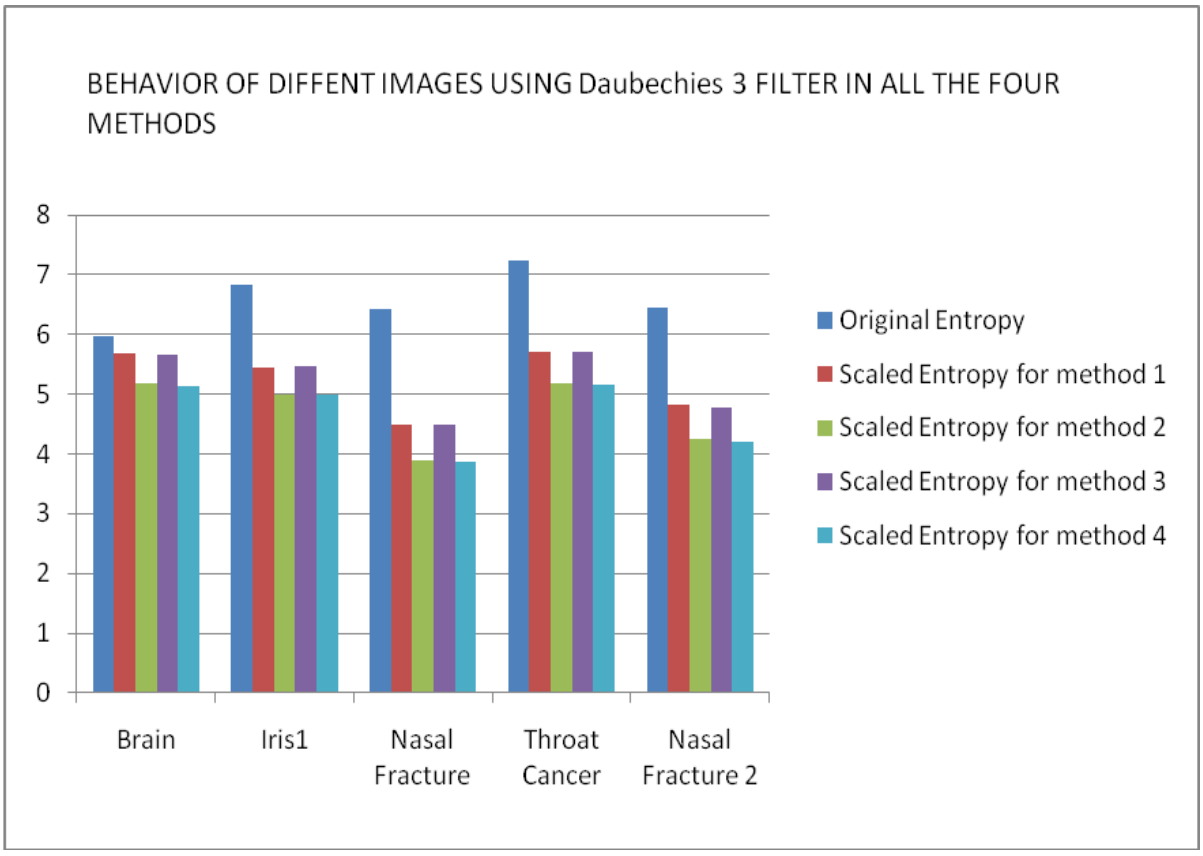
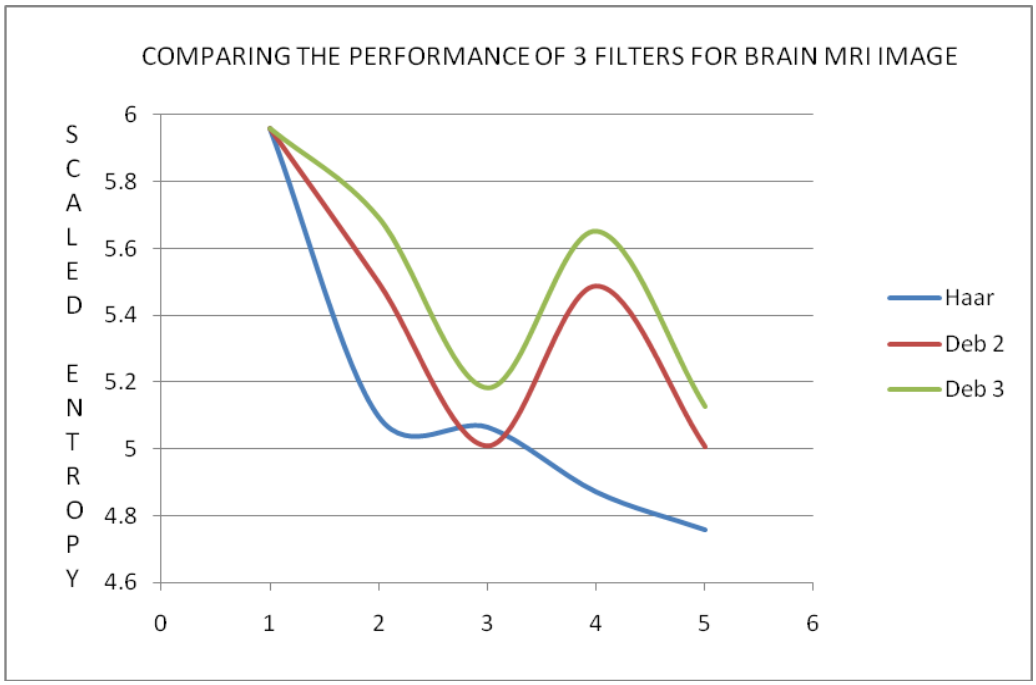


Figure 4.37 Comparing the Performance of 3 Filters For Brain MRI Image



Chapter 5 - Conclusion

5.1. Synopsis

This report presented four different methods for lossless medical image compression and these methods are tested using five different images obtained from the medical image database of North Carolina State University and University of Iowa. The images are compressed losslessly by performing integer wavelet transform using lifting technique as mentioned in the report of Daubechies and Wim Sweldens [22] and lossless predictive coding technique using second order predictors. In Daubechies method a technique for factoring wavelet transform into lifting steps has been developed. Lifting is achieved by performing simple filtering steps using finite filters such as Haar filter. In all our methods we have used first order Haar filter for performing lifting. In lossless predictive coding technique we take the difference or prediction error into consideration rather than taking into account the original sample/image, the differences are taken between the original sample and the sample(s) before the original sample as it is easier to encode the difference rather than encoding the original sample.

In the first lossless compression method, the image is transformed into four subbands using lifting technique, then predictive coding is applied to each subband using different predictor coefficients alpha and beta, giving an encoded image as output. Now the reverse process is applied to the encoded and the transformed image getting back the original image. The second method employs predictive coding technique on the image first followed by the integer wavelet transform giving a transformed output, which is then passed through the reverse techniques to get back the original image. The third and forth methods performed on the image involve reduction of the Haar filter coefficients by a factor of 3/2 and the methods one and two are implemented on all the images. The report also presents results when two other filters are used in lifting technique instead of Haar filter. The two other filters used are Daubechies 2 and Daubechies 3. Entropy and scaled entropy are used to calculate the performance of the system, which calculates the number of bits per pixel. A lower entropy and scaled entropy indicate higher performance of the system.

The analysis of the experimental results has given many conclusions. Choosing the predictor coefficients is more critical as the alpha and beta value can lie between 0 and 1, so different combinations of these coefficients are tested. The best combination in methods 1 and 2 are 03, 0.3 and methods 3 and 4 are 0.5, and 0.5. Among all the four methods, the forth method of performing predictive coding followed by integer wavelet transform using the reduced filter coefficients gave a better compression. This process reduced the entropy of the original image by almost 40% as shown in Table 4. This result can also be seen in Fig.44 and Fig.45 where the graphs are plotted for all the four methods. Of all the three filters used the first order Haar filter gave better performance as shown in Fig.49. Out of all the images used, the nasal fracture images gave the least entropy for all the four methods performed this can be seen in the tables present in Sections 4.5.2 and 4.5.3 and Fig.46.

5.2. Future Suggestions

The lifting scheme used in this report is only a two level lifting scheme. In order to improve the entropy of the transformed image, a multi level lifting scheme is to be implemented. The performance of the predictive coding can be increased by using higher order predictors with two dimensional predictions. Another possibility for improving the performance would be to use model-based and adaptive approaches.

The performance of the four lossless compression techniques proposed in the report can also be improved by performing different combinations of various transforms and coding techniques other than IWT and predictive coding, and realize the most optimal combination that gives the least entropy.

Bibilography

[1]. S.Arivazhagan1, W.Sylvia Lilly Jebarani, G.Kumaran, Performance Comparison of Discrete Wavelet Transform and Dual Tree Discrete Wavelet Transform for Automatic Airborne Target Detection, International Conference on Computational Intelligence and Multimedia Applications 2007, pp 495-500.

[2]. Serkan Hatipoglu, Sanjit K. Mitra and Nick Kingsbury, "Texture classification using Dual-Tree Complex Wavelet Transform", IEE Conference on Image Processing and its Applications, 465, 1999, pp. 344-347.

[3]. Panchamkumar D Shukla, "Complex Wavelet Transforms and their Applications", M.Phil.Thesis, Signal Processing Division, Department of Electronic and Electrical Engineering, University of Strathclyde, Glasgow G11XW, Scotland, United Kingdom, 2003.

[4]. K. R. Rao, Y. Huh, JPEG 2000 8th International Symposium on video/Image Processing and Multimedia Communication,2002.

[5]. C. Christopoulos, A Skodras and T. Ebrahimi, "The JPEG 2000 still image coding system: An overview", IEEE Trans. Consumer Electronics, vol 46, pp.1103-1127, Nov 2000.

[6]. X. Wu, *Context-Based, Adaptive, Lossless Image Coding*, IEEE Trans. Communications, vol. 45, No. 4, April 1997.

[7]. Jiang J., Guo B., Yang S.Y., "Revisit to JPEG-LS prediction scheme", IEE Proceedings: Vision, Image and Signal Processing, Vol. 147, No.6, (Dec. 2000).

[8]. ISO/IEC 14495-1**,** ITU T.87, "Information technology - Lossless and near-lossless compression of Continuous tone still images," 1999.

[9]. M. J. Weinberger, G. Seroussi, and G. Sapiro, "LOCO-I: A low complexity, context-based, lossless image compression algorithm," in *Proc. DCC'SG,* (Snowbird, Utah, USA), pp. 140-149, Mar. 1996.

[10]. I. Ueno and F. Ono. "Prouosed modification of LOCO-I for its improvement of the performance." ISOIIEC JTCl/SC29/WGl doc. N297. Feb. 1996.

[11]. M. 'J. Weinbeiger, G. Seroussi, and G. Sapiro. ISO/IEC JTCl/SC29/WGl docs. N341, N386, N412 (1996).

[12]. M. J. Weinberger, G. Seroussi, G. Sapiro, and E. Ordentlich, "JPEG-LS with limited-length code words." ISO/IEC JTCl/SC29/WGl doc. N538**,** July 1997.

[13]. Joel Sol´e Rojals, Optimization and Generalization of Lifting Schemes: Application to Lossless Image Compression. PhD Dissertation, Universitat Polit`ecnica de Cataluny, Barcelona April 2006.

[14]. Wade Spires, Lossless Image Compression via the Lifting Scheme. University of Central Florida, Nov 2005.

[15]. David Broman, Lossless data compression – methods for achieving better performance in a Wireless VPN. Master's Thesis, June 2001.

[16]. Vasuki.A and P.T Vasanta, Image compression using lifting and vector quantization. GVIP Journal, Volume 7, Issue 1, April, 2007

[17]. Umbaugh, S. B., Computer Vision and Image Processing: A Practical Approach Using CVIPtools, Prentice Hall PTR, 1998.

[18]. XIE Yao-hua, TANG Xiao-an, SUN Mao-yin, Image Compression Based on Classification Row by Row and LZW Encoding, 2008 Congress on Image and Signal Processing.

[19]. Parvinder Singh, Manoj Duhan, and Priyanka, Enhancing LZW Algorithm to Increase Overall Performance, Guru Jambheshwar University, India.

[20]. Ying Chen Pengwei Hao, Integer Reversible Transformation to Make JPEG Lossless, ICSP'O4 Proceedings.

[21]. McCoy, J.W., Magotra, N., Stearns,S. Lossless Predictive Coding. IEEE 1995.

[22]. I. Daubechies and W. Sweldens, Factoring Wavelet Transforms into Lifting Steps, Journal of Fourier Analysis and Applications, Vol. 4, No. 3, pp. 245-267, 1998.

APPENDIX A - Matlab Code

IWT followed by Predictive

```
clear all;
close all;
clc;

F=imread('Lena1.bmp');
F=rgb2gray(F);
figure(1);
imshow(F,[]);
title('Original Image');

%lifting scheme
i2=double(F);
h1=[-1 9 9 1]/(16*1.5);
h2=[0 0 1 1]/(-4*1.5);

%rows
o=i2(:,1:2:size(i2,2));
e=i2(:,2:2:size(i2,2));
y0=e-round(filter2(h1,o));
y1=o+round(filter2(h2,y0));
%column
y00=y1(1:2:size(y1,1),:);
y01=y1(2:2:size(y1,1),:);
sd=y01-round(filter2(h1,y00));
ss=y00+round(filter2(h2,sd));
%column
y10=y0(1:2:size(y0,1),:);
y11=y0(2:2:size(y0,1),:);
```

```
dd=y11-round(filter2(h1,y10));
ds=y10+round(filter2(h2,dd));
figure(2)
im=[ss sd;ds dd];
imshow(uint8(im));
title('Image Obtained after Subband Coding');

%compression
[m1 n1]=size(ss);
[m2 n2]=size(sd);
[m3 n3]=size(ds);
[m4 n4]=size(dd);
%predictive coding for ss subband
F1=zeros(m1,2);
F2=[F1 ss];
F3=zeros(2,n1+2);
F4=[F3;F2];
F5=double(F4);

alpha=0.5;
beta=0.5;
%taking the 2nd order predictor
for i=3:n1+2
   for j=3:n1+2
      F6(i-2,j-2)=round((alpha)*(F5(i,j-1)+(beta)*F5(i,j-2)));
      F6(i-2,j-2)=round((alpha)*F5(i-1,j)+(beta)*F5(i-2,j));
   end
end
D1=double(ss)-F6;

%predictive coding for sd subband
```

```
F11=zeros(m2,2);
F21=[F11 sd];
F31=zeros(2,n2+2);
F41=[F31;F21];
F51=double(F41);

%taking the 2nd order predictor
for i=3:n2+2
   for j=3:n2+2
      F7(i-2,j-2)=round((alpha)*(F51(i,j-1)+(beta)*F51(i,j-2)));
      F7(i-2,j-2)=round((alpha)*F51(i-1,j)+(beta)*F51(i-2,j));
   end
end

D2=double(sd)-F7;

%predictive coding for ds subband
F111=zeros(m3,2);
F211=[F111 ds];
F311=zeros(2,n3+2);
F411=[F311;F211];
F511=double(F411);

%taking the 2nd order predictor
for i=3:n3+2
   for j=3:n3+2
      F8(i-2,j-2)=round((alpha)*(F511(i,j-1)+(beta)*F511(i,j-2)));
      F8(i-2,j-2)=round((alpha)*F511(i-1,j)+(beta)*F511(i-2,j));
   end
end
```

```
D3=double(ds)-F8;

 %predictive coding for dd subband
F1111=zeros(m4,2);
F2111=[F1111 dd];
F3111=zeros(2,n4+2);
F4111=[F3111;F2111];
F5111=double(F4111);


%taking the 2nd order predictor
for i=3:n4+2
   for j=3:n4+2
      F9(i-2,j-2)=round((alpha)*(F5111(i,j-1)+(beta)*F5111(i,j-2)));
      F9(i-2,j-2)=round((alpha)*F5111(i-1,j)+(beta)*F5111(i-2,j));
   end
end

D4=double(dd)-F9;
figure(3);
imshow([D1 D2;D3 D4],[]);
title('Encoded Image');
im1=(([D1 D2;D3 D4]));
E=entropy(uint8([D1 D2;D3 D4]));
%finding entropy
E0=entropy1(im);
E=entropy1(im1);
E1=entropy1(D1);
E2=entropy1(D2);
E3=entropy1(D3);
E4=entropy1(D4);
```

```
total=(1/4)*(E1+E2+E3+E4);

% decoding ss
F10=D1+F6;

%decoding sd
F11=D2+F7;

%decoding ds
F12=D3+F8;

%decoding dd
F13=D4+F9;
figure(4);
imshow([F10 F11;F12 F13],[]);
title('Decoded Image');

%reconstruction
x00=F10-round(filter2(h2,F11));
x01=F11+round(filter2(h1,x00));

x0(1:2:2*size(x00,1),:)=x00;
x0(2:2:2*size(x01,1),:)=x01;

x10=F12-round(filter2(h2,F13));
x11=F13+round(filter2(h1,x10));

x1(1:2:2*size(x10,1),:)=x10;
x1(2:2:2*size(x11,1),:)=x11;

z0=x0-round(filter2(h2,x1));
```

```
z1=x1+round(filter2(h1,z0));

z(:,1:2:2*size(z0,2))=z0;
z(:,2:2:2*size(z1,2))=z1;

figure(5);
imshow(uint8((z)));
title('Reconstructed Image');
I=(z)-double(F);
figure(6);
imshow(I);
title('Difference of the original and restructured image')
```

Predictive Followed by IWT

```
 clear all;
close all;
clc;

F=imread('Lena1.bmp');
F=rgb2gray(F);
figure(1);
imshow(F,[]);
title('Original Image');

[M,N]=size(F);
F1=zeros(M,2);
F2=[F1 F];
F3=zeros(2,N+2);
F4=[F3;F2];
F5=double(F4);
```

```
figure(2);
imshow(F4);

alpha=0.5;
beta=0.5;
%taking the 2nd order predictor
for i=3:N+2
   for j=3:N+2
      F6(i-2,j-2)=round((alpha)*(F5(i,j-1)+(beta)*F5(i,j-2)));
      F6(i-2,j-2)=round((alpha)*F5(i-1,j)+(beta)*F5(i-2,j));
   end
end

 D=double(F)-F6;
 figure(3);
 imshow(D,[]);
 title('Predictive Encoder');
 Entropy=entropy1(D);

% lifting scheme
i2=double(D);
h1=[-1 9 9 1]/(16*1.5);
h2=[0 0 1 1]/(-4*1.5);
%rows
o=i2(:,1:2:size(i2,2));
e=i2(:,2:2:size(i2,2));
y0=e-round(filter2(h1,o));
y1=o+round(filter2(h2,y0));
%column
y00=y1(1:2:size(y1,1),:);
y01=y1(2:2:size(y1,1),:);
```

```
sd=y01-round(filter2(h1,y00));
ss=y00+round(filter2(h2,sd));
%column
y10=y0(1:2:size(y0,1),:);
y11=y0(2:2:size(y0,1),:);
dd=y11-round(filter2(h1,y10));
ds=y10+round(filter2(h2,dd));
figure(4)
im=[ss sd;ds dd];
imshow(uint8(im));
title('Image Obtained after Subband Coding');
%CALCULATING ENTROPY'S
E=entropy1(im);
E1=entropy1(ss);
E2=entropy1(sd);
E3=entropy1(ds);
E4=entropy1(dd);
total=(1/4)*(E1+E2+E3+E4);


%reconstruction
x00=ss-round(filter2(h2,sd));
x01=sd+round(filter2(h1,x00));

x0(1:2:2*size(x00,1),:)=x00;
x0(2:2:2*size(x01,1),:)=x01;

x10=ds-round(filter2(h2,dd));
x11=dd+round(filter2(h1,x10));

x1(1:2:2*size(x10,1),:)=x10;
```

```
x1(2:2:2*size(x11,1),:)=x11;

z0=x0-round(filter2(h2,x1));
z1=x1+round(filter2(h1,z0));

z(:,1:2:2*size(z0,2))=z0;
z(:,2:2:2*size(z1,2))=z1;

figure(5);
imshow(((z)),[]);
title('Image after inverse IWT');

%decoding
F7=(z)+double(F6);
figure(6);
imshow(F7,[]);
title('Predictive Decoder');
I=(F7)-double(F);
figure(7);
imshow(I,[]);
title('Difference of Original and Restructured');
```

Code for Entropy

```
function sum=entropy1(im1)
[n n]=size(im1);
x5=im1(:);
x6=im1(:);
r1=reshape((im1),1,n*n);%converting into a row matrix
[x1 i1]=sort((r1));%sorting and getting the indices
```

```
x2=r1(i1);%placing the sorted values in the indices
l1=length(x2);
 sum1=0;
for i=1:l1
    occur=0;
    for j=1:l1
        if( x1(i)==x2(j))
            occur=occur+1;
        end
    end
    p1(i)=occur/l1;
    sum1=((p1(i)*log2(p1(i)))/(occur))+sum1;
end
sum=(-sum1);
```

APPENDIX B - Matlab Program Listings

IMREAD:

Reads a grayscale or color image from the file specified by the string filename.

IMSHOW:

Displays the grayscale image I.-imshow (I).

RESHAPE:

Returns the m-by-n matrix B whose elements are taken column-wise from A. An error results if A does not have m*n elements.- B = reshape(A,m,n)

SORT:

Sort array elements in ascending or descending order.-B = sort (A)

SIZE:

Array dimensions- returns the sizes of each dimension of array X in a vector d with ndims(X) elements.- d = size(X).

FILTER2:

2-D digital filter filters the data in X with the two-dimensional FIR filter in the matrix h.

It computes the result, Y, using two-dimensional correlation, and returns the central part of the correlation that is the same size as X.- Y = filter2 (h,X)

ROUND:

Round to nearest integer rounds the elements of X to the nearest integers. For complex X, the imaginary and real parts are rounded independently.- Y = round(X).

LENGTH:

Length of vector. The statement length(X) is equivalent to max (size(X)) for nonempty arrays and 0 for empty arrays.- n = length(X)

DOUBLE:

Converts to double precision, double(x) returns the double-precision value for X. If X is already a double-precision array, double has no effect.- double(x)

ZEROS:

zeros(N) is an N-by-N matrix of zeros. ZEROS (M, N) or ZEROS ([M, N]) is an M-by-N matrix of zeros.

SUM

S = sum (X) is the sum of the elements of the vector X. If X is a matrix, S is a row vector with the sum over each column.

TITLE

Title ('text') adds text at the top of the current axis.

Unit8

I = Uint8(X) converts the elements of the array X into unsigned 8-bit integers.