Image Compression Using Wavelet Transform Based SPIHT Algorithm a nd Performance Analysis by Changing Parameter Simulated with QccPack

Major Project Report

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology

in

Electronics and Communication Engineering

(Communication Engineering)

By

Patel Daxesh Bhikhabhai

(08MECC13)

Department of Electronics & Communication Engineering,

Institute of Technology,

Nirma University,

Ahmedabad-382 481

May 2010

Image Compression Using Wavelet Transform Based SPIHT Algorithm and Performance Analysis by Changing Parameter Simulated with QccPack

Major Project Reprot

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology

in

Electronics and Communication Engineering

(Communication Engineering)

By

Patel Daxesh Bhikhabhai

(08MECC19)

Under the Guidance of

Prof. N.P.Gajjar

Department of Electronics & Communication Engineering,

Institute of Technology,

Nirma University,

Ahmedabad-382 481

May 2010

Declaration

This is to certify that

i) The thesis comprises my original work towards the degree of Master of Technology in Communication Engineering at Nirma University and has not been submitted elsewhere for a degree.

ii) Due acknowledgement has been made in the text to all other material used.

Patel Daxesh Bhikhabhai

Certificate

This is to certify that the Major Project entitled **"Image Compression Using Wavelet Transform Based SPIHT Algorithm and Performance Analysis By Changing Parameter Simulated With QccPack"** submitted by **Patel Daxesh Bhikhabhai (08MECC13)**, towards the partial fulfillment of the requirements for the degree of Master of Technology in Electronics & Communication (Communication) of Institute of Technology, Nirma University, Ahmedabad is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of our knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Date: Place: Ahmedabad

Internal Guide	External Guide
(Prof.N.P.Gajjar) Associate Professor, EC Dept., Nirma University, Ahmedabad	(Prof. V.H.Patel) Senior Faculty, BISAG,Gandhinagar
HOD	Director
(Prof. A. S. Ranade) Professor, EC	(Dr. K Kotecha) Director, IT, NU

Acknowledgements

I would like to express my gratitude and sincere thanks to Prof. A. S. Ranade Head of Electrical Engineering Department and Dr. D. K. Kothari Coordinator M.Tech Communication Engineering program for allowing me to undertake this thesis work and for his guidelines during the review process.

I am deeply indebted to my thesis supervisor Prof. N. P. Gajjar for his constant guidance and motivation. He has devoted significant amount of his valuable time to plan and discuss the thesis work. Without his experience and insights, it would have been very difficult to do quality work.

I would like to express my gratitude and sincere thanks to Mr. T. P. Singh, Director of BISAG (Bhaskaracharya Institute of Space Application and Geo-Informatics) for giving me an opportunity to work under to guidance of renowned people in the field of communications and for providing all the resources for the project development.

I would like to express my endless thanks to the external guide of my project thesis Prof. V. H. Patel, senior faculty of BISAG for his sincere and dedicated guidance throughout the project development.

I wish to thank all the people of BISAG and Nirma University, my classmates, Dr. Jay Patel and Dr. Jaikishan Lalwani for the valuable guidance and efforts behind the project.

Last, but not the least, no words are enough to acknowledge constant support and sacrifices of my family members because of whom I am able to complete the degree program successfully.

- Patel Daxesh Bhikhabhai (08MECC13)

Abstract

The study and analysis of wavelet image compression techniques focuses mainly on developing energy efficient, computation efficient and adaptive image compression technique, as the images will be a large part of future wireless data. Digital images are widely used in computer applications and multimedia applications. Uncompressed image required considerable storage capacity and transmission bandwidth. Set Partitioning In Hierarchical Tree (SPIHT) which uses wavelet transforms, represents a very effective form of entropy coding. The combination of integer lifting wavelet transform with SPIHT algorithm has been widely used in the field of image compression.

The thesis conveys a detail study and simulation of advance image compression algorithm named SPIHT, and modification of algorithm for more compression and good image quality after reconstruction.

In first part of work describes in brief about image compression and various compression techniques. Second part gives information about wavelet Filters, how DWT is implemented for 2D. The detail about SPIHT coding/decoding algorithm is described. This algorithm simulated with QccPack library and performance analysis is done by varying various parameters of SPIHT algorithm. Two modifications in SPIHT algorithm is done. First one is to select the threshold which is data dependent and second one is to make partitions adaptive that reduce amount of bit require than ordinary SPIHT.

Results provide a good reference for application developers to choose the wavelet, number of decomposition level and compression ratio with option to use arithmetic code or not for their application and modify algorithm as per their requirement.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

To enable modern high bandwidth required in wireless data services such as mobile multimedia, email, mobile internet access, mobile commerce, mobile data sensing in sensor networks, home and medical monitoring services, mobile conference and satellite communication for that there is a growing demand for rich content cellular data communication including Voice, Text, Image and Video.

One of the major challenges in enabling mobile multimedia data services will be the need to process and transmit very large volume of the data on wireless media. This will impose severe demands on the battery resources of multimedia mobile appliances as well as the bandwidth of the wireless network. While significant improvements in achievable bandwidth are expected with future wireless access technology, improvements in battery technology will lag the rapidly growing energy requirements of the future wireless data services. One approach to mitigate this problem is to reduce the volume of multimedia data transmitted over the wireless channel via data compression technique such as JPEG and MPEG . These approaches concentrate on achieving higher compression ratio without sacrificing the quality of the Image.

1.2 Introduction

The study and analysis of wavelet image compression techniques focuses mainly on developing energy efficient and adaptive image compression technique, as the images will be a large part of future wireless data. The Thesis conveys an implementation of advanced image compression algorithm using wavelet transform, consisting of technique to eliminate computation of certain High/Low pass filter coefficients of an image. The results from experiments explain the use of SPIHT(Set Partitioning In Hierarchical Trees) algorithm which can significantly reduce the no of bits require to represent the image. The reduction in energy is obtained with minimally perceptible loss in the Quality of the Image.

There is an identification of several parameters that can be varied in SPIHT algorithm and analyzed their effects image quality. Based on SPIHT algorithm and its dynamic parameters development of an adaptive image codec, which minimizes energy consumption and air time (Service/Bandwidth cost) needed for an Image based data service, while meeting the bandwidth constraints of the wireless network and the image quality and latency constraints of the wireless service. And the core importance of proposed modified SPIHT algorithm is to get more compressed image with same quality of image served by basic SPIHT Coder.

The thesis conveys describes in brief about image compression, compression techniques and its comparison, different wavelet transform based compression technique and its comparison. It will also gives information about different wavelet filters, how DWT (Discrete wavelet transform) is implemented for 2D? The detail about SPIHT Algorithm coding, decoding using SPIHT and also modification in SPIHT algorithm for more compression. The result of experiments are compared depending on PSNR (Pick Signal to Noise Ratio) and MSE.

1.3 Organization of Report

CHAPTER 2: This Chapter contains the basics of Image Compression and its needs, fundamental of image compression technique and types of compression methods.

CHAPTER 3: This Chapter contains comparison of different image compression methodology and with feature comparison it shows how SPIHT is better for the image compression.

CHAPTER 4: This Chapter contains the basics of wavelet filter and implementation steps of 2D-DWT (Discrete Wavelet Transform).

CHAPTER 5: This Chapter contains the detail study of SPIHT algorithm. It also contains the modification in algorithm of SPIHT coder with data dependent threshold and the scheme of Adaptive Partitions.

CHAPTER 6: This Chapter contains the introduction of QccPack, features of QccPack, script to run SPIHT with QccPack, performance analysis of SPIHT algorithm with changing parameters, testing different images with SPIHT algorithm and the result of modified SPIHT algorithm.

Chapter 2

Basics of Image Compression

2.1 Need of Compression

In the last decade, there has been a lot of technological transformation in the way to communicate. This transformation includes the growing internet, the explosive development in mobile communication and ever increasing importance of image communication. Data compression is one of the emerging technology in the field of multimedia revolution. Cellular phones would not be able to provide communication with increasing clarity without data compression. Data compression is an art and science of representing information in compact form.

Uncompressed multimedia (graphics, audio and video) data requires considerable storage capacity and transmission bandwidth, despite rapid progress in mass-storage density, processor speeds and digital communication system performance. Demand for data storage capacity and data-transmission bandwidth continues to outstrip the capabilities of available technologies. In a distributed environment large image files remain a major bottleneck within systems.

Image Compression is an important component of the solutions available for creating image file sizes of manageable and transmittable dimensions. Platform portability and performance are important in the selection of the compression/decompression technique to be employed.

2.1.1 Principle Behind Image Compression

Images have considerably higher storage requirement than Text and Audio. An image stored in an uncompressed file format, such as the popular BMP format, can be huge. An image with a pixel resolution of 640 by 480 pixels and 24-bit color resolution will take up $640 * 480 * 24/8 = 921,600$ bytes in an uncompressed format.

The huge amount of storage space is not only the consideration but also the data transmission rates for communication of continuous media are also significantly large. An image, $1024pixel * 1024pixel * 24bit$, without compression, would require 3 MB of storage and 7 minutes for transmission, utilizing a high speed, 64 Kbits/s, ISDN line.

Image data compression becomes still more important because of the fact that the transfer of uncompressed graphical data requires far more bandwidth and data transfer rate. For example, throughput in a multimedia system can be as high as 140 Mbits/s, which must be transferred between systems. This kind of data transfer rate is not realizable with today's technology, or in near the future with reasonably priced hardware. The figures in following table A [1] show the qualitative transition.

Often signals are wish to process in the time-domain, but in order to process them more easily, other information such as frequency is required. Mathematical transforms translates the information of signals into different representations. For example, the Fourier Transform converts a signal between the time and frequency domains, such that the frequencies of a signal can be seen. However the fourier transform cannot pro-

Image Type	**Size/Duration**	**Bits/Pixel**	**Original Size**	**Transmission Time (using a28.8K Modem)**
Gray Image	$512 * 512$	8bps	262KB	1min 13 sec
Colour Image	$512 * 512$	24bps	782KB	3min 39sec
Medical Image	$2048 * 1680$	12bps	5.16MB	23min 54sec
SHD Image	$2048 * 2048$	24bps	12.48MB	58min 15 sec

Table A: Size of Different Types of Uncompressed Images

vide information on which frequencies occur at specific times in the signal as time and frequency are viewed independently. To solve this problem the Short Time Fourier Transform (STFT) introduced the idea of windows through which different parts of a signal are viewed. For a given window in time the frequencies can be viewed. However Heisenberg's Uncertainty Principle states that *the resolution of the signal improves in the time domain, by zooming on different sections, the frequency resolution gets worse.* Ideally a method of multi-resolution is needed, which allows certain parts of the signal to be resolved well in time and other parts to be resolved well in frequency. The power and magic of wavelet analysis is exactly the multi-resolution.

2.1.2 Idea Behind Wavelet Transform

Wavelet analysis can be used to divide the information of an image into approximation and detail sub-signals. The approximation sub-signals shows the general trend of pixel values and three detail sub-signals show the vertical, horizontal and diagonal details or changes in the image. If these details are very small then they can be set to zero without significantly changing the image. *The value below which details are considered small enough to be set to zero is known as the threshold.* The greater the numbers of zeros the greater the compression that can be achieved. The amount of information retained by an image after compression and decompression is known as the 'energy retained' and this is proportional to the sum of the squares of the pixel values. If the energy retained is 100% then the compression is known as 'lossless', as the image can be reconstructed exactly. This occurs when the threshold value is set to zero, meaning that the detail has not been changed. If any values are changed then energy will be lost and this is known as 'Lossy' compression. Ideally, during compression the no. of zeros and the energy retention will be as high as possible. However, as more zeros are obtained more energy is lost, so a balanced between the two needs to be found.

2.2 Fundamentals of Image Compression Techniques

A digital image, or "bitmap", consists of a grid of dots, or "pixels", with each pixel defined by a numeric value that gives its color. The term data compression refers to the process of reducing the amount of data required to represent a given quantity of information. Now, a particular piece of information may contain some portion which is not important and can be comfortably removed. All such data is referred as redundant data. Data redundancy is a central issue in digital image compression. Image compression research aims at reducing the number of bits needed to represent an image by removing the spatial and spectral redundancies as much as possible.

A common characteristic of most images is that the neighboring pixels are correlated and therefore contain redundant information. The foremost task then is to find less correlated representation of the image. In general, three types of redundancy can be identified:

A. Coding redundancy. [2.2.1]

B. Inter pixel redundancy.[2.2.2]

C. Psycho visual redundancy.[2.2.3]

2.2.1 Coding Redundancy

If the gray levels of an image are coded in a way that uses more code symbols than absolutely necessary to represent each gray level, the resulting image is said to contain coding redundancy. It is almost always present when an images gray levels are represented with a straight or natural binary code. Let us assume that a random variable r_k lying in the interval [0, 1] represents the gray levels of an image and that each r_k occurs with probability $P_r(r_k)$.

$P_r(r_k) = N_k/n$
L = No. of gray level
N_k = No. of times that gray appears in that image
n = Total no. of pixels in the image
If no. of bits used to represent each value of r_k is i(r_k) the average no. of bits required to represent each pixel is
$L_avg = L(r_k)P(r_k)$

That is average length of code words assigned to the various gray levels is found by summing the product of the no. of bits used to represent each gray level and the probability that the gray level occurs. Thus the total no. of bits required to code an $M * N$ is $M * N * L_{avg}$

2.2.2 Inter Pixel Redundancy

The information of any given pixel can be reasonably predicted from the value of its neighboring pixel. The information carried by an individual pixel is relatively small. In order to reduce the inter pixel redundancies in an image, the 2-D pixel array normally used for viewing and interpretation must be transformed into a more efficient but usually non visual format. For example, the differences between adjacent pixels can be used to represent an image. These types of transformations are referred as mappings. They are called reversible if the original image elements can be reconstructed from the transformed data set.

2.2.3 Psycho Visual Redundancy

Certain information simply has less relative importance than other information in normal visual processing. This information is said to be Psycho Visually Redundant, it can be eliminated without significantly impairing the quality of image perception.

In general, an observer searches for distinguishing features such as edges or textual regions and mentally combines them in recognizable groupings. The brain then correlates these groupings with prior knowledge in order to complete the image interpretation process.

The elimination of psycho visually redundant data results in loss of quantitative information; it is commonly referred as quantization. As this is an irreversible process i.e. visual information is lost, thus it results in Lossy Data Compression. An image reconstructed following Lossy compression contains degradation relative to the original. Often this is because the compression scheme completely discards redundant information.

2.3 Image Compression Techniques

There are basically two methods of Image Compression:

A. Lossless coding techniques.[2.3.1]

B. Lossy coding techniques.[2.3.2]

2.3.1 Lossless Coding Techniques

In Lossless Compression schemes, the reconstructed image, after compression, is numerically identical to the original image. However Lossless Compression can achieve a modest amount of compression.

Lossless coding guaranties that the decompressed image is absolutely identical to the image before compression. This is an important requirement for some application domains, e.g. Medical Imaging, where not only high quality is in the demand, but unaltered archiving is a legal requirement. Lossless techniques can also be used for the compression of other data types where loss of information is not acceptable, e.g. text documents and program executables. Lossless compression algorithms can be used to squeeze down images and then restore them again for viewing completely unchanged.
Lossless coding techniques are as follows:

A. Run length encoding

B. Huffman encoding

C. Entropy encoding

D. Area encoding

2.3.2 Lossy Coding Techniques

Lossy techniques cause image quality degradation in each compression / de-compression step. Careful consideration of the Human Visual perception ensures that the degradation is often unrecognizable, though this depends on the selected compression ratio. An image reconstructed following Lossy compression contains degradation relative to the original. Often this is because the compression schemes are capable of achieving much higher compression. Under normal viewing conditions, no visible loss is perceived (Visually Lossless). Lossy Image Coding Techniques normally have three Components:

A. Image Modeling:

   It is aimed at the exploitation of statistical characteristics of the image (i.e. high correlation, redundancy). It defines such things as the transformation to be applied to the Image.

B. Parameter Quantization:

   The aim of Quantization is to reduce the amount of data used to represent the information within the new domain.

C. Encoding:

   Here a code is generated by associating appropriate code words to the raw produced by the quantizer. Encoding is usually error free. It optimizes the representation of the information and may introduce some error detection codes.

Chapter 3

Image Compression Methodologies

3.1 Basic Description of the Image Compression Methods

This chapter describes the four public domain image compression coders/decoders.

A. JPEG coding (DCT based coding) [3.1.1]

B. Wavelet Transformation Based Coding [3.1.2]

C. Vector Quantization coding [3.1.3]

D. Fractal Coding[3.1.4]

JPEG and Wavelet Coders use a transformation (DCT for JPEG and Wavelet for Wavelet) to obtain a representation of the image that is better for image compression purposes. This transformed image is then compressed by quantization and subsequent lossless compression of quantized coefficients.

Vector Quantization uses a so-called ”codebook” in which different blocks (vectors) are stored and tries to approximate each block of the image with a vector from the codebook.

The fractal method uses IFS (Iterative Function System) to describe each block of the image. It has a slight similarity to vector quantization because the IFS for a block, if found by mapping different parts of the image onto it.

3.1.1 JPEG : DCT-Based Image Coding Standard

The discovery of DCT in 1974, is an important achievement for the research community working on image compression [2]. The DCT can be regarded as a discrete time version of the Fourier Cosine Series. It is a close relative of DFT, a technique for converting a signal into elementary frequency components. An excellent analysis of DCT and related transforms and their applications can be found in [2].

In 1992, JPEG established the first international standard for still image compression where the encoders and decoders are DCT-based [3]. The JPEG standard specifies three modes namely sequential, progressive, and hierarchical for lossy encoding, and first mode of lossless encoding.. The 'baseline JPEG coder' which is the sequential encoding in its simplest form, will be briefly discussed here. Fig. 3.1[3] and 3.2[3] show the key processing steps in such an encoder and decoder for grayscale images. The DCT-based encoder can be thought of as essentially compression of a stream of 8 X 8 blocks of image samples. Each 8 X 8 block makes its way through each processing step, and yields output in compressed form into the data stream. Because adjacent image pixels are highly correlated, the 'Forward' DCT (FDCT) processing step lays the foundation for achieving data compression by concentrating most of the signal in the lower spatial frequencies. For a typical 8 X 8 sample block from a typical source image, most of the spatial frequencies have zero or near-zero amplitude and need not be encoded. In principle, the DCT introduces no loss to the source image samples; it merely transforms them to a domain in which they can be more efficiently encoded.

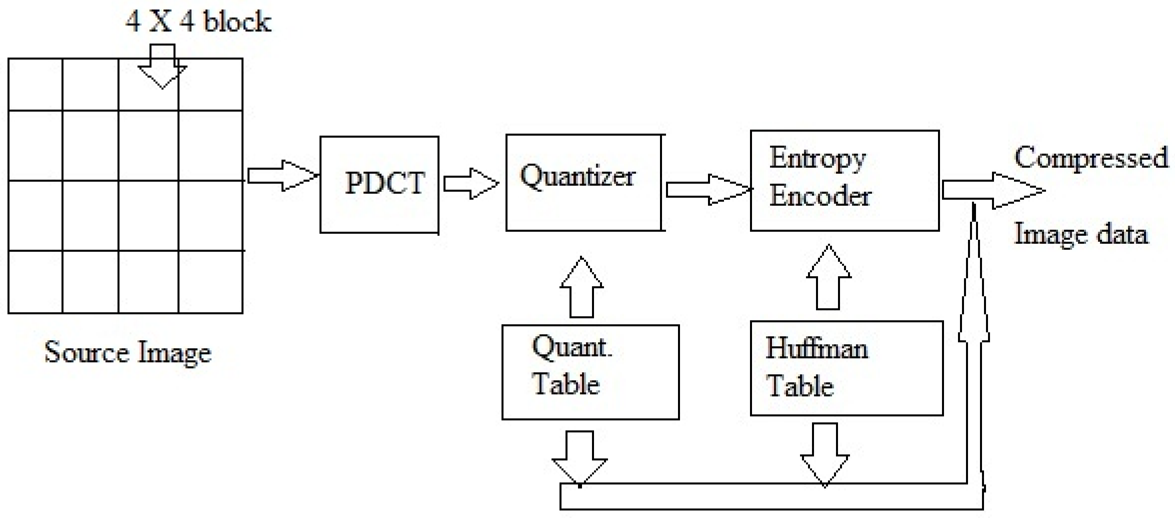


Figure 3.1: JPEG Encoder Block Diagram

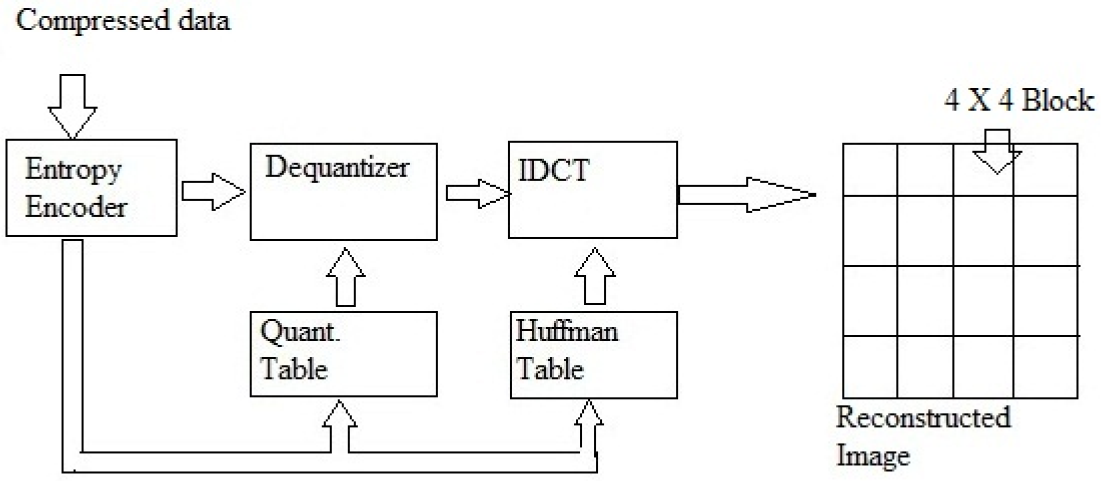


Figure 3.2: JPEG Decoder Block Diagram

After the output from the FDCT, each of the 64 DCT coefficients is uniformly quantized in conjunction with a carefully designed 64-element Quantization Table. At the decoder, the quantized values are multiplied by the corresponding QT elements to recover the original unquantized values. After quantization, all of the quantized coefficients are ordered into the "zig-zag" sequence as shown in Fig. 3.3 [3]. This ordering helps to facilitate entropy encoding by placing low-frequency non-zero coefficients before high-frequency coefficients. The DC coefficient, which contains a significant fraction of the total image energy, is differentially encoded. Entropy Coding (EC)

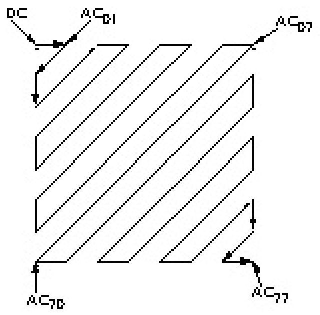


Figure 3.3: Zig-Zag Sequence

achieves additional compression losslessly by encoding the quantized DCT coefficients more compactly based on their statistical characteristics. The JPEG proposal specifies both Huffman Coding and Arithmetic Coding. The baseline sequential codec uses Huffman Coding, but codecs with both methods are specified for all modes of operation. Arithmetic Coding, though more complex, normally achieves 5-10% better as compared to the Huffman Coding.

3.1.2 Wavelet Transformation Based Coding

The Wavelet Compression Method [4] does not subdivide the image, the entire image is approximated by a sum of copies of an approximated function and a "Mother wavelet", which are resized (dilatation) and placed at different positions (translation) in the picture.

The following steps are performed:

A. Construction of a Detailed-Space-Pyramid: approximation of the image and the detailed needed to get the original image are calculated recursively.

   Along one axis,, the above would look like this:

   BBBBBBBB: Original image; which is split in to

   LLLLHHHH: LLLL is an approximation of the image, HHHH is the missing detail.

   LLHHHHHH: the approximation LLLL was split into another approximation LL and detail HH

   The L (approximation) data can be viewed best as information contained in the low pass filtered image, whereas the H (Detail) data as the information of the high pass-filtered image.

B. Along two axes, it look like this:

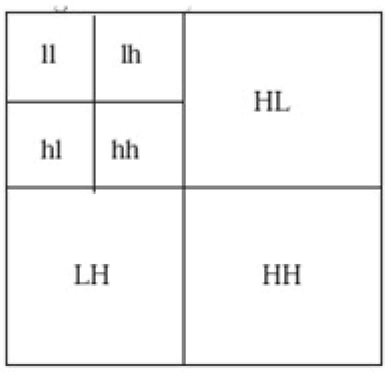


Figure 3.4: Image Decomposition

C. Now quantization is performed; intuitively the strength of quantization increases with growing distance of the square above from the top left square (bottom right square is most quantized).

D. The resulting frequency subband are similar to each other ;this can be used for further coding efficiency.

3.1.3 Vector Quantization Coding

The Predictive Residual Vector Quantizer (PVRQ) was chosen to investigate the Vector Quantization Method [4]. This coder operates as follows:

A. The image divided in to blocks of 4X4 or 6X6 pixels (4X4 for better quality but higher bit-rate).

B. Each block is predicted using the preceding block and a residual block is calculated by subtracting the prediction from the actual block.

C. The residual is quantized; that is, there is a "code-book" (a set of common residual block), among which the block most similar to the current residual block is chosen. There are 25 levels of quality, each working with its own code-book (the better the quality, the bigger the code-book).

D. The resulting data is packed by an arithmetic coder.

3.1.4 Fractal Coding

Fractal compression is a lossy image compression method using fractals. This method is best suited for textures and natural images, relying on the fact that parts of an image often resemble other parts of the same image. Fractal algorithms convert these parts into mathematical data called "fractal codes" which are used to recreate the encoded image. Fractal compression differs from pixel-based compression schemes such as JPEG, GIF and MPEG since no pixels are saved. Once an image has been converted into fractal code, the image can be recreated to fill any screen size without the loss of sharpness that occurs in conventional compression schemes. Fractal image representation can be described mathematically as an iterated function system (IFS).

3.1.5 Comparison and Conclusion

Ranking based on PSNR shown in Table A [4].

As table shows if the comparison done based on PSNR (quality of reconstructed image) the Wavelet transform method comes at top than PVRQ than JPEG coder and than Fractal coding comes at last.

PSNR rank	**Method**
1	Wavelet
2	PVRQ
3	JPEG
4	Fractal

Table A: Ranking by PSNR

Ranking based on Compression, Decompression time shown in Table B [4].

As table shows if the comparison done based on compression and decompression time require, JPEG comes at top and wavelet comes at last.

rank	**Compression Time**	**Decompression Time**
1	JPEG	JPEG
2	Fractal	PVRQ
3	PVRQ	Fractal
4	Wavelet	Wavelet

Table B: Ranking by Compression Decompression Time

By both table it is conclude that ”though wavelet requires more time for compression and decompression it gives more PSNR i.e good quality of reconstructed image”.

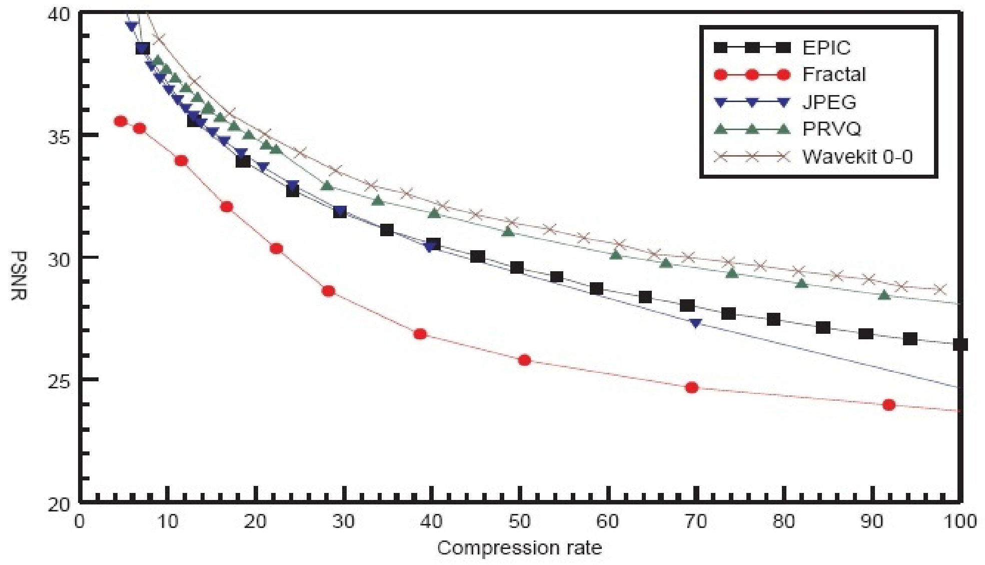


Figure 3.5: Comparison for LENNA 512 Image

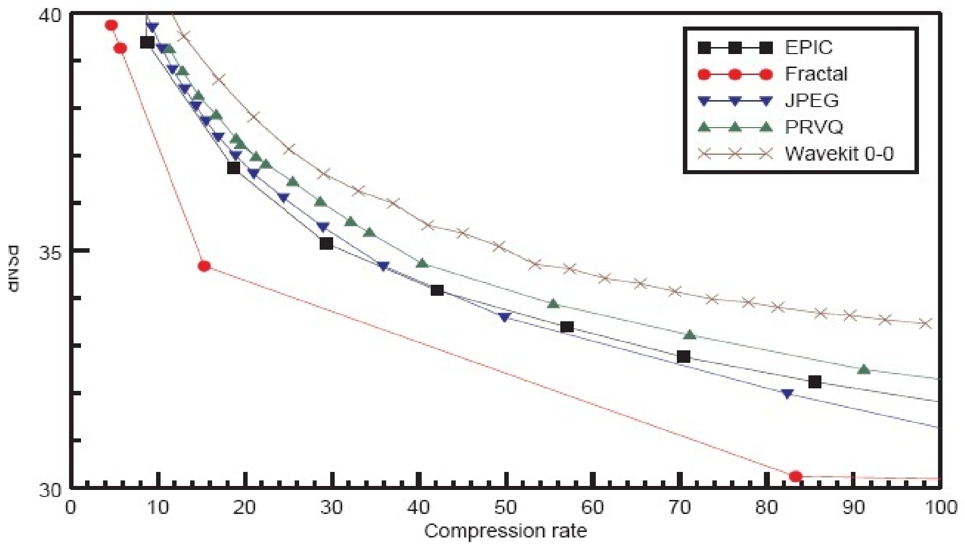


Figure 3.6: Comparison for Satellite Image

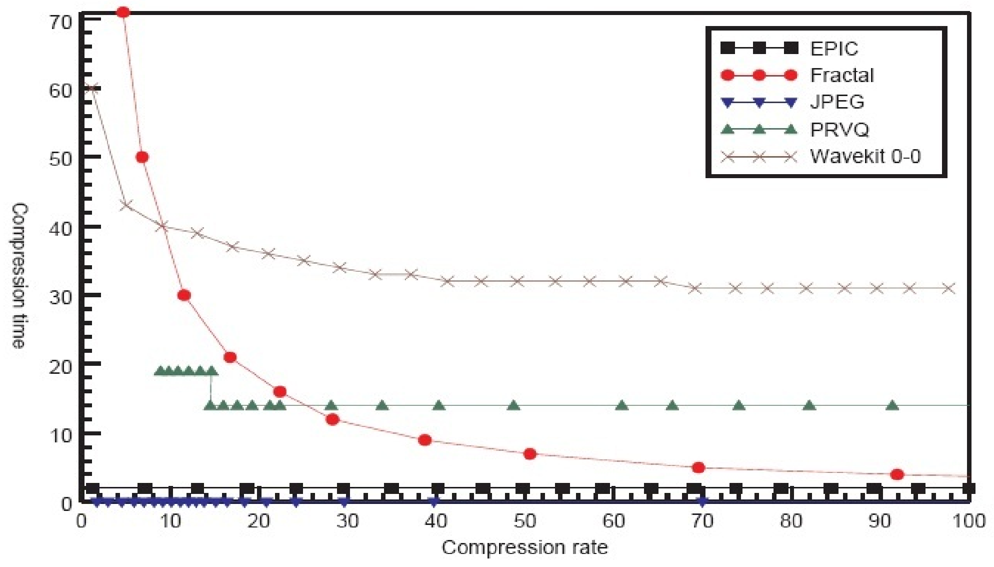


Figure 3.7: Comparison of Compression Time for LENNA512

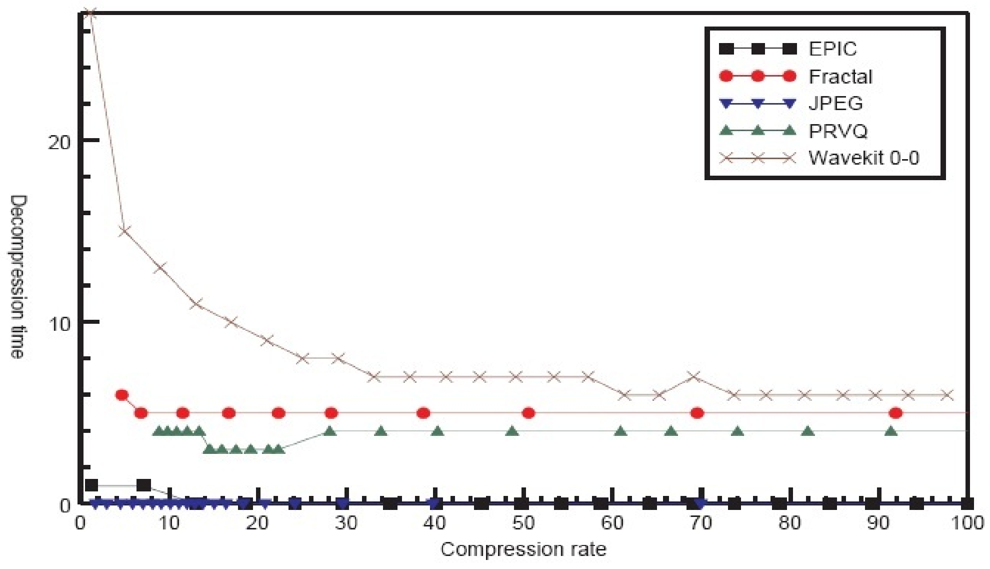


Figure 3.8: Comparison of Decompression Rate for LENNA512

3.2 Various Coding Methods of Wavelet Coefficient

One of the most successful applications of wavelet methods is transform-based image compression (also called coding). The coder is shown in the Fig. 3.9 [5] operates by transforming data to remove redundancy, then quantizing the transform coefficients (a lossy step),and finally entropy coding the quantizer output. Because of the superior energy compaction properties and correspondence with human visual system, wavelet compression methods have produced superior objective and subjective results. Since wavelet basis consists of functions with both short support (for high frequencies) and long support (for low frequencies), large smooth areas of an image may be represented with very few bits, and details are added where it is needed. The decoder side is the

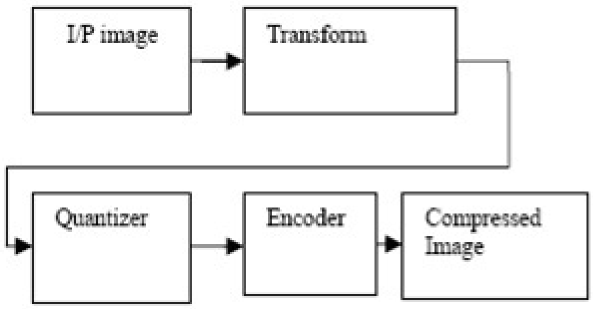


Figure 3.9: Block Diagram of a Transform Based Coder

reverse of the coder side. For Coding side discuss on following types EZW [3.2.1], SPIHT [3.2.2], SPECK [3.2.3] are done.

3.2.1 Embedded Zero Tree Wavelet (EZW) Coding

The EZW algorithm [5] was one of the first algorithms to show the full power of wavelet-based image compression. It was introduced in the groundbreaking paper of Shapiro. An EZW encoder is an encoder specially designed to use with wavelet transforms. The EZW encoder was originally designed to operate on images (2D-signals) but it can also be used on other dimensional signals.

The EZW encoder is based on progressive encoding to compress an image into a bit stream with increasing accuracy. This means that when more bits are added to the stream, the decoded image will contain more detail. Progressive encoding is also known as embedded encoding, which explains the E (Embedded) in EZW.

3.2.2 Set Partitioning In Hierarchical Trees (SPIHT) Coding

The SPIHT coder is a highly refined version of the EZW algorithm and is a powerful image compression algorithm that produces an embedded bit stream from which the best reconstructed images in the mean square error sense can be extracted at various bit rates. Some of the best results highest PSNR values for given compression ratios for a wide variety of images have been obtained with SPIHT. Hence, it has become the benchmark state-of-the-art algorithm for image compression.

Features of SPIHT

The SPIHT method is not a simple extension of traditional methods for image compression, and represents an important advance in the field. The method provides the following [5]:

A. Good image quality, high PSNR, especially for color images.

B. It is optimized for progressive image transmission.

C. Produces a fully embedded coded file.

D. Simple quantization algorithm.

E. Fast coding/decoding (nearly symmetric).

F. Has wide applications, completely adaptive.

G. Can be used for lossless compression.

3.2.3 Set Partitioned Embedded bloCK Coder (SPECK)

The image coding scheme the SPECK, is different from some of the above-mentioned schemes in that it does not use trees which span, and exploit the similarity, across different sub bands; rather, it makes use of sets in the form of blocks. The main idea is to exploit the clustering of energy in frequency and space in hierarchical structures of transformed images.

The SPECK algorithm can be said to belong to the class of scalar quantized significance testing schemes. It has its roots primarily in the ideas developed in the SPIHT, and few block coding image coding algorithms [5].

Features of The Coder

The SPECK image coding scheme has all the properties characteristic of scalar quantized significance testing schemes. In particular, it exhibits the following properties:

A. It is completely embedded :- A single coded bit stream can be used to decode the image at any rate less than or equal to the coded rate, to give the best reconstruction.

B. Of the image possible with the particular coding scheme.

C. It employs progressive transmission :- Source samples are coded in decreasing order of their information content.

D. It has low computational complexity :- The algorithm is very simple, consisting mainly of comparisons, and does not require any complex computation.

E. It has low dynamic memory requirements :- At any given time during the coding process, only one connected region (lying completely within a subband) is processed. Once this region is processed, the next region is then considered for processing.

F. It has fast encoding/decoding :- This is due to the low-complexity nature of the algorithm.

3.2.4 Summary of Above Coding Techniques

Type	**Features**	**Demerits**
EZW	A. Employs progressive and embedded transmission B. Uses zerotree concept C. Tree coded with single symbol D. Uses predefined scanning order	A. Transmission of coefficient position is missing B. No real compression C. Followed by arithmetic encoder
SPIHT	A. Widely used- high PSNR values for given CRs for variety of images B. Quad- tree or hierarchical trees are set partitioned C. Employs progressive and embedded transmission D. Superior to JPEG in perceptual image quality and PSNR	A. Only implicitly locates position of significant coefficients B. More memory requirements due to 3 lists C. Transmitted information is formed of single bits D. Suits variety of natural images E. Perceptual quality not optimal
SPECK	A. Uses blocks- rectangular regions B. Employs progressive and embedded transmission C. Low computational complexity D. Low memory requirements due to 2 lists	A. As layers increase, performance decreases

Table C: Summary of Various Coding Techniques

Chapter 4

Wavelet Filters and DWT

The most basic definition of a wavelet is simply a function with a well-defined temporal support that "wiggles" about the x-axis (it has exactly the same area above and below the axis). The basic wavelet transform is similar to the well-known fourier transform. Like the fourier transform, the coefficients are calculated by an inner product of the input signal with a set of orthonormal basis functions (this is a small subset of all available wavelet transforms though). The difference comes in the way these functions are constructed, and more importantly in the types of analysis they allow.

A wavelet system expansion can be written as following: [6]

$$f\,(t) = \sum_{k=-\infty}^{\infty} (C_{j0,k} 2^{\frac{j0}{2}} \emptyset(2^{j_0} t - k)) + \sum_{k=-\infty}^{\infty} (\sum_{j=j_0}^{\infty} (d_{j,k} \gamma(2^{j} t - k)))$$

The key difference is that the *Wavelet Transform is a multi-resolution transforms that is, it allows a form of time-frequency analysis* [1] (or translation-scale in wavelet speak). When using the Fourier Transform the result is a very precise analysis of the frequencies contained in the signal, but no information on when those frequencies occurred. From wavelet transform, information about when certain features occurred, and about the scale characteristics of the signal are taken. *Scale is analogous to frequency, and is a measure of the amount of detail in the signal. Small scale generally means coarse details, and large scale means fine details* (scale is a number related to the number of coefficients and is therefore counter-intuitive to the level of detail). The Discrete Wavelet Transform can be described as a series of filtering and sub-sampling (decimating in time) as depicted in Figure 4.1[1]. In each level in this series, a set of 2j-1 coefficients are calculated, where $j < J$ is the scale and N = 2J is the number of samples in the input signal. The coefficients are calculated by applying a high-pass wavelet filter to the signal and down sampling the result by a factor of 2. At the same level, a low-pass scale filtering is also performed (followed by down-sampling) to produce the signal for the next level. Both the wavelet and scale filters can be obtained from a single Quadrature Mirror Filter (QMF) function that defines the wavelet. Each set of scale-coefficient corresponds to a "smoothing" of the signal and the removal of details, whereas the wavelet-coefficients correspond to the "differences" between the scales. Wavelet theory shows that from the coarsest scale-coefficients and the series of the wavelet-coefficients the original signal can be reconstructed. The total number of coefficients (scale + wavelet) equals the number of samples in the signal.

4.1 Implementation DWT

For many natural signals, the wavelet transform is a more effective tool than the fourier transform. The wavelet transform provides a multiresolution representation using a set of analyzing functions that are dilations and translations of a few functions (wavelets). The wavelet transform comes in several forms. The critically sampled form of the wavelet transform provides the most compact representation; however, it has several limitations. For example, it lacks the shift-invariance property, and in multiple dimensions it does a poor job of distinguishing orientations, which is important in image processing. For these reasons, it turns out that for some applications

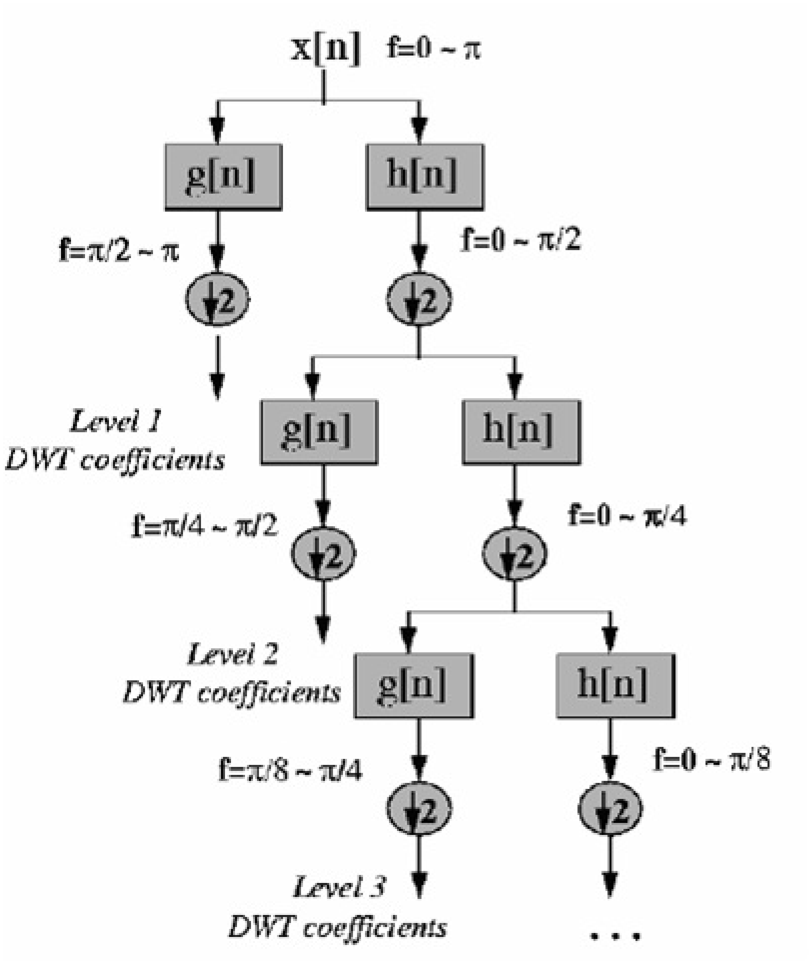


Figure 4.1: Discrete Wavelet Transform

improvements can be obtained by using an expansive wavelet transform in place of a critically sampled one. (An expansive transform is one that converts an N-point signal into M coefficients with $M > N$.) There are several kinds of expansive DWTs; here description and an implementation of the dual-tree complex discrete wavelet transform. The dual-tree complex wavelet transform overcomes these limitations - it is nearly shift-invariant and is oriented in 2D. The 2D dual-tree wavelet transforms produces six subbands at each scale, each of which is strongly oriented at distinct angles. In addition to being spatially oriented, the 3D dual-tree wavelet transform is also motion selective - each subband is associated with motion in a specific direction.

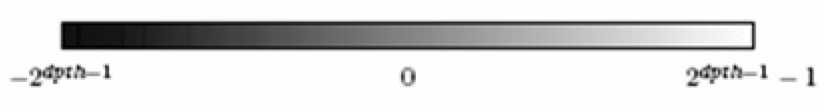
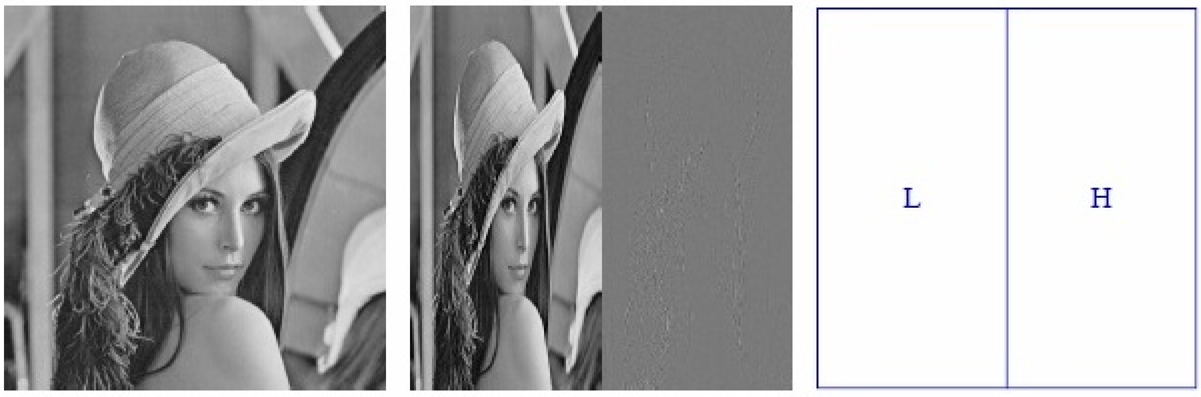


Figure 4.2: Gray Scales and the Corresponding Pixel Values for Dpth-Bit Resolution

The 3D dual-tree isolates in its subbands motion in distinct directions.

4.1.1 2D-DWT

Now one can able to discuss the separable two dimensional wavelet transform in detail. Consider again a row r of a given image of size N X N. After one level of transform and obtain N/2 low pass coefficients c(0,l) and N/2 high pass coefficients d(0,k) These are given in interleaved order, that is
(c(0,0), d(0,0), c(0,1), d(0,1), . . . , c(0,N-1), d(0,N-1))
because of the split in odd and even indexed positions in the Lifting Scheme. Usually the row of above is rearranged to
r(0) = (c(0,0), c(0,1), . . . , c(0,N-1), d(0,0), d(0,1), . . . , d(0,N-1)),
because in next applying the transform to the low frequency coefficients c(0,l) recursively. Suppose there has been already transformed and rearranged all rows of a given image as described above. If store the computed coefficients in place, which is in the memory space of the original image, can obtain a new array with a structure as shown in Figure 4.3[1]. On the left the well known benchmark image lena1 is shown. To the right of it applied the CDF(2,2)wavelet transform to the rows of the image. The corresponding result is interpreted as image again (Figure4.3 (b)) and is composed of a coarse and scaled version of the original and the details, which are necessary to reconstruct the image under consideration. On the right illustrated this the interpretation as low and high frequency coefficients blocks, denoted by L and H, respectively. Remark that most of the high frequency coefficients d(0,k) are shown in grey color, which corresponds to small values around zero. The one dimensional wavelet trans-



(a) *lena*, $N = 512$, dpth = 8

(b) CDF(2,2) wavelet transform applied to the rows of (a)

(c) low (L, $c_{0,k}$) and high (H, $d_{0,k}$) frequency coefficient blocks

Figure 4.3: One Dimensional CDF(2,2) Wavelet Transform Applied to the Rows and Columns of the Benchmark Image Leena with Reflection at the Boundaries

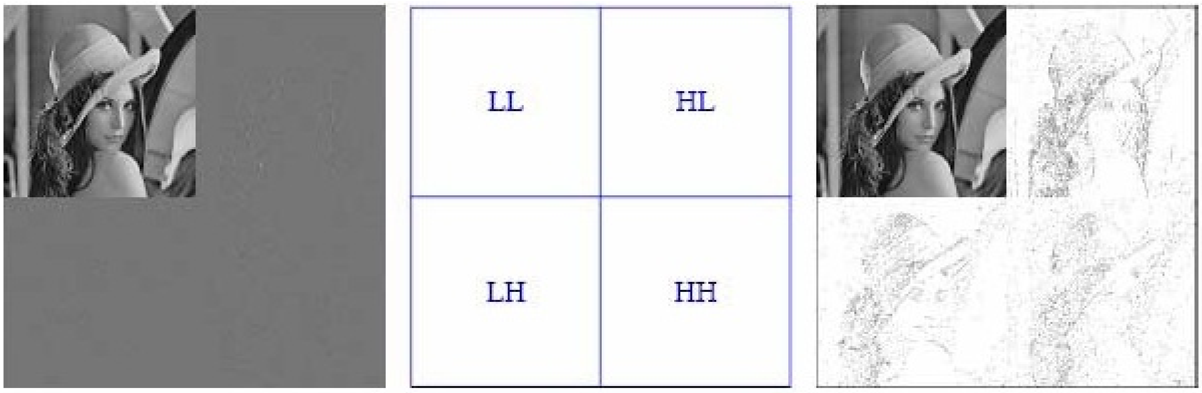
form can be applied to the columns of the already horizontal transformed image as well. The result is shown in Figure 4.4 and is decomposed into four quadrants with different interpretations.

A. LL: The upper left quadrant consists of all coefficients, which were filtered by the analysis low pass filter h along the rows and then filtered along the corresponding columns with the analysis low pass filter h again. This subblock is denoted by LL and represents the approximated version of the original at half the resolution.

B. HL/LH: The lower left and the upper right blocks were filtered along the rows and columns with $\hat{h}$ and $\hat{g}$, alternatively. The LH block contains vertical edges, mostly. In contrast, the HL blocks shows horizontal edges very clearly.

C. The lower right quadrant was derived analogously to the upper left quadrant but with the use of the analysis high pass filter $\hat{g}$ which belongs to the given wavelet. Now interpret this block as the area, where the find edges of the original image in diagonal direction.

The two dimensional wavelet transform can be applied to the coarser version at half the resolution, recursively, in order to further decorrelate neighboring pixels of the input image. For an illustration refer to Figure 4.4[1]. The subbands in the next higher transform levels l will be denoted by LL(l), LH(l), HL(l), and HH(l), where LL(1) = LL, LH(1) = LH, HL(1) = HL, and HH(1) = HH, respectively. Thereafter



(a) CDF(2,2) wavelet applied as tensor product to the rows and columns of the image *lena*

(b) the different frequency blocks

(c) set all coefficients v in LH, HL, HH with $-20 < v < 20$ to white color

Figure 4.4: One Dimensional CDF(2,2) Wavelet Transform Applied to the Rows of the Benchmark Image Leena with Reflection at the Image Boundaries

the coefficient in the upper left corner represents the average greyscale value of the whole image and is called *DC coefficient*. In practice, usually four up to six level of wavelet transform level will be performed. Due to the local similarities between neighboring pixels, many coefficients in the LH, HL, and HH sub bands at different scales will be small. As a consequence only a few samples, especially those of the LL block at the coarsest scale, represents most out of the *images energy*.

This observation is the starting point of wavelet based image compression algorithms.[1]

The focus on the tree structured decomposition as shown in Figure 4.5, where only

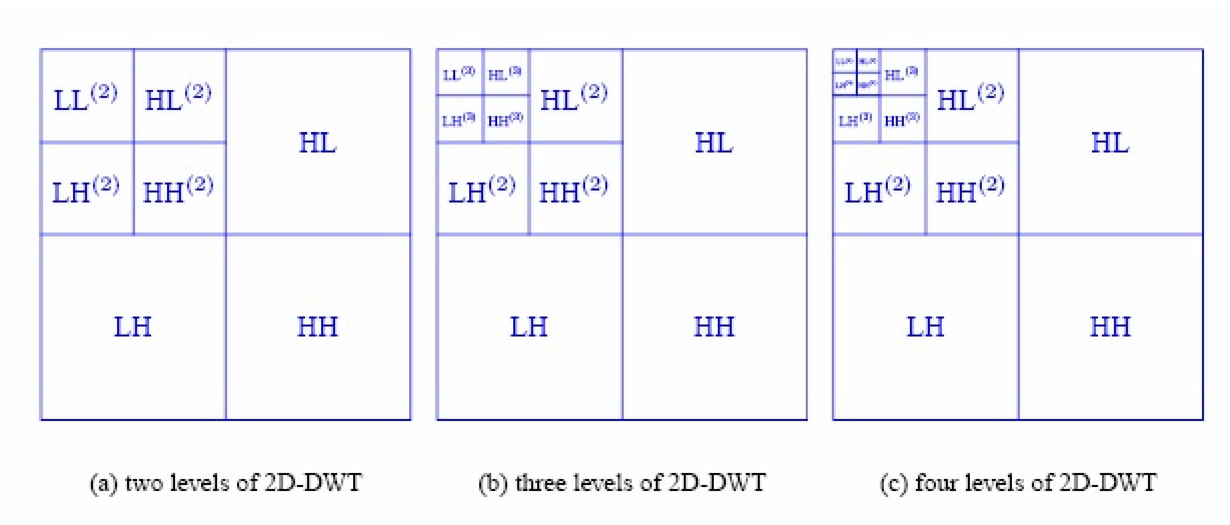


Figure 4.5: Multiresolution Scheme After Several Levels of Wavelet Transform

the LL blocks are subdivided. This type of image decomposition is also known as multiresolution scheme and multiscale representation. Other decomposition types are possible and known under the terms of wavelet packets.

Chapter 5

SPIHT Algorithm And Modification

5.1 SPIHT Algorithm

One of the most efficient algorithms in the area of image compression is the Set Partitioning In Hierarchical Trees (SPIHT). In essence it uses a sub-band coder, to produce a pyramid structure where an image is decomposed sequentially by applying power complementary low pass and high pass filters and then decimating the resulting images. These are one-dimensional filters that are applied in cascade (row then column) to an image whereby creating a four-way decomposition: LL (low-pass then another low pass), LH (low pass then high pass), HL (high and low pass) and finally HH (high pass then another high pass). The resulting LL version is again four-way decomposed, as shown in Figure 5.1[7]. This process is repeated until the top of the pyramid is reached.

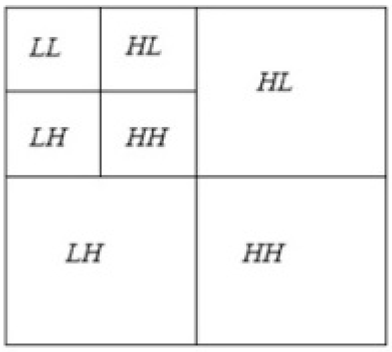


Figure 5.1: Image Decomposition Using Wavelets

There exists a spatial relationship among the coefficients at different levels and frequency sub-bands in the pyramid structure. A wavelet coefficient at location (i,j) in the pyramid representation has four direct descendants (off-springs) at locations:

$$O(i, j) = (2i, 2j), (2i, 2j + 1), (2i + 1, 2j), (2i + 1, 2j + 1)$$

and each of them recursively maintains a spatial similarity to its corresponding four off-spring. This pyramid structure is commonly known as spatial orientation tree. For example, Figure 5.2[7] shows the similarity among sub-bands within levels in the wavelet space. If a given coefficient at location (i,j) is significant in magnitude then some of its descendants will also probably be significant in magnitude. The SPIHT algorithm takes advantage of the spatial similarity present in the wavelet space to optimally find the location of the wavelet coefficient that are significant by means of a binary search algorithm.

The SPIHT algorithm sends the top coefficients in the pyramid structure using a progressive transmission scheme. This scheme is a method that allows obtaining a high quality version of the original image from the minimal amount of transmitted data. As illustrated in Table A, the pyramid wavelet coefficients are ordered by magnitude and then the most significant bits are transmitted first, followed by the next bit plane and so on until the lowest bit plane is reached. It has been shown that

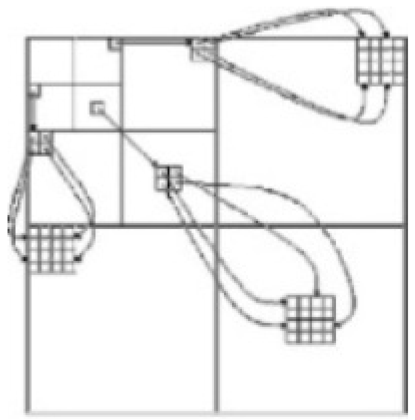


Figure 5.2: Off-Spring Dependencies in the Pyramid Structure

progressive transmission can significantly reduced the Mean Square Error (MSE) distortion for every bit-plane sent.

To take advantage of the spatial relationship among the coefficients at different levels and frequency bands, the SPIHT coder algorithm orders the wavelets coefficient according to the significance test defined as:

$$\max_{(i,j)\in\tau_m} |C_{i,j}| \geq 2^n$$

Where C is the wavelet coefficient at the nth bit plane, at location (i,j) of the τ_m subset of pixels, representing a parent node and its descendants. If the result of the significance test is yes an S flag is set to 1 indicating that a particular test is significant. If the answer is no, then the S flag is set to 0, indicating that the particular coefficient is insignificant.

Bit row	sign	s	s	s	s	s	s	s	s	S
MSB	5	1	1	0	0	0	0	0	0	0
	4			1	1	0	0	0	0	0
	3					1	1	1	0	0
	2									0
	1									
	0									

Table A: Bit-Plane Ordering and Transmission Scheme

$$S_n(\tau) = \left\{ \begin{array}{ll} 1, & \max_{(i,j)\epsilon\tau} |C_{i,j}| \geq 2^n \\ 0, & \qquad\qquad otherwise \end{array} \right\}$$

Wavelets coefficients which are not significant at the nth bit-plane level may be significant at (n-1)th bit-plane or lower. This information is arranged, according to its significance, in three separate lists: List of Insignificant Sets (LIS), the List of Insignificant Pixels (LIP) and the List of Significant Pixels (LSP). In the decoder, the SPIHT algorithm replicates the same number of lists. It uses the basic principle that *if the execution path of any algorithm is defined by the results on its branching points, and if the encoder and decoder have the same sorting algorithm then the decoder can recover the ordering information easily.*

The SPIHT algorithm can be summarized as follows [8].

Algorithm:1

A. Initialization:

(1)

$$n = \left\lfloor \log_2(max_{(i,j)} \{|C_{i,j}|\}) \right\rfloor$$

(2) LSP – Empty

(3) LIP – All pixels of LL band

(4) LIS – Root of tree in LL band

B. Sorting Pass:

(AA)

```
(1) Each entry of LIP tested for significance w.r.t n.
    If (significant)
    1 and Sign bit are transmitted and pixel is moved to LSP
    Else
    0 is transmitted
(2) Each entry of LIS is tested for existence of significant descendants.
    IF (there are non)
    0 is transmitted
    Else
    1 is transmitted'
    then next 4 bits are tested for significance,
    and according bits are transmitted
    If (descendants greater than 4)
    this pixel move at the end of LIS as type (BB)
```

(BB)

i. Here Grand child of that pixel is test for significant.

ii. Than this entry is removed from the list.

iii. Its descendents are appended to the end of the list as entries of type (AA)

C. Refinment Pass: The nth MSB of the magnitude of each entry of LSP, except those added in the current sorting pass, is transmitted.

D. THAN N IS DECREMENTED BY 1 AND ALGORITHM REPEAT FROM STEP B

5.2 Modification in SPIHT Algorithm

5.2.1 Data-Dependent Thresholds

Using a set of fixed thresholds obviates the transmission of these thresholds as side information. Choices of thresholds and their number have an impact on compression efficiency. One wants to identify as many newly significant sets as possible between successive thresholds, but at the same time have these sets be small and identify large insignificant sets. Choosing too many thresholds may result in sending redundant information, since too few changes may occur between two thresholds. Choosing too few thresholds will inevitably lead to large significant sets that require many splitting steps to locate the significant elements. The best thresholds for a given source are necessarily data-dependent and must decrease as rapidly as possible without producing too many large significant sets.

Consider the thresholds to be the maximum (magnitude) elements among the sets split at a given stage. For example, say the maximum value in a set and then split this set into four subsets. Find the maximum elements in each of these four subsets.

Split into four any subset having an element whose value equals the set maximum. Such a subset is said to be significant. Now repeat the procedure for each of these four new subsets and so on until the final splits that produce four single elements. These single elements are then encoded and ouputted to the codestream. Clearly, after creating subsets, the ones not further split, whose elements are capped in value by the full set maximum. These may be called the insignificant sets. The maximum of each subset that is not further split is less than the previous maximum. Then each of these subsets is split, just as the full set, but with a lesser maximum value to guide further splitting. These subsets should be tested from smallest to largest, that is, in reverse of the order they were generated. The reason is that one can more quickly locate significant single elements, thereby finding earlier value information, when the sets are smaller and closer to those already found. Clearly, there are production of a series of decreasing thresholds, as so-called insignificant sets are generated. The steps in the modification of the fixed threshold algorithm, Algorithm 1, to an adaptive one are presented in detail in Algorithm 2. [9]

One major difference from the previous preset thresholds is that the thresholds generated here depend on the original data, so have to be encoded. Prior to splitting of any set, its maximum is found. An efficient scheme is to generate a series of 4-bit binary masks, where a "1" indicates a significant set (having an element equal to the maximum) and "0" an insignificant set (no element within having a value equal to the maximum). A "1" in one of these masks specifies that the maximum is to be found in this set. A "0" indicates that the maximum element is not in the associated set and that a new (lesser) maximum has to be found and encoded. The masks themselves have to be encoded and outputted, so that the decoder can follow the same splitting procedure as the encoder. The maxima can be coded quite efficiently through differences from the global maximum or a similar differencing scheme. Each 4-bit mask can be encoded with the natural binary digits or with a simple, fixed Huffman code of 15 symbols (four "0"s cannot occur).

The above scheme works well when there are relatively large areas with 0-value elements, as found naturally in the usual image transforms. The compression results are comparable to those yielded from fixed thresholds. If the probabilities of values are approximately or partially known beforehand, it is advisable to index the values, so that the highest probability value has 0 index, the next highest 1, and so forth. Then this set partitioning and encoding procedure operates on these indices. The association of the indices to the actual values must be known beforehand at the decoder to reconstruct the source data.

Algorithm:2

A. Initialization:

(1) Find v_{max}, the largest magnitude among the source elements.

(2) Create a list LIS(List of Insignificant Sets) initially empty.LIS will contain "0"-labeled (called insignificant) sets.

(3) Put coordinates of LL(upper left corner) source block on to LIS.Label set with "0".

(4) Set threshold t = v_{max} and bit-plane index n = (log_2t). Encode t and write to codestream buffer.

B. Testing and Partitioning

(1) For each multiple element set on the LIS(all "0"-labeled set with more than one element), do

i. If maximum element less than t, write "0" to codestream buffer, else write "1" to code-stream buffer and do the following

A. Remove set from LIS and devide the set in to 4 equal quadrants.

B. Find maximum element of each quadrant v_q, q = 1,2,3,4. For each quadrant, label with "1", if $v_q = t$, otherwise lebel with "0" if $v_q < t$.(This step creats a 4-bit binary mask.)

C. Add coordinates of "0"-labeled quadrant on to LIS. Encode 4-bit binary mask on to codestream buffer.

D. For each quadrant of size 2X2 or greater labeled with "1", go to B(1)iA.

E. When quadrants have been split in to single elements, encode "0"-labeled elements with $n + 1$ bits and write to codestream buffer.("1"-labeled single elements have values equal to the known t.)

C. If $t > 0$ and there are multiple sets on the LIS, reset t equal to maximum of all the maxima of LIS sets,encode t, set $n = (log_2 t)$,and return to B(1).

D. If t = 0, stop.

5.2.2 Adaptive Partitions

The methods described above always split the set into fixed-shape subsets, such as quadrants of a block or children and grand-descendant sets of a tree. Also the iterative splitting into subsets numbering four seems somewhat arbitrary and other numbers may prove better. Since the attempt is to find clusters of insignificant (or significant) elements, why not adjust the partition shape to conform better to the outer contours of the clusters. As stated before, methods whose sets are exactly these clusters need too much location information as overhead. But there are methods that try to vary the number of clusters and the splitting shape in limited ways that prove to be quite effective for compression.

Step no.	Test element (in braces)	Output code
1.	[0 0 0 0 0 1 1 0]	1
2.	[0 0 0 0] 0 1 1 0	0
3.	0 0 0 0 [0 1] 1 0	1
4.	0 0 0 0 [0] 1 1 0	0
5.	0 0 0 0 0 1 [1 0]	1
6.	0 0 0 0 0 1 [1] 0	1
7.	0 0 0 0 0 1 1 [0]	0

Table B: Size of different types of uncompressed images

One such method is a form of group testing[9]. The basic procedure is to start with a group of k linearly ordered elements in the set K.To test this set as before to locate significant elements and clusters of insignificant elements. Again, the envision of a threshold test of a set that declares "1" as significant when there is a contained element whose value passes the test and declares "0" as insignificant when all contained elements fail the test. If the set K is significant, it is then split into two nearly equal parts, K_1 and K_2 of k_1 and k_2 elements, respectively. The smaller part, say K_1, is first tested for significance. If significant, K_1 is further split into two nearly equal subsets. If insignificant, then the other part, K_2, must be significant, so it is split into two nearly equal parts. This testing and splitting is repeated until the significant single elements are located. Whenever the threshold test is enacted, a "1" or "0" is emitted to indicate a significant or insignificant result, respectively. This procedure is called a group iteration, where K is the group.

Consider the example of coding the binary sequence K = 00000110 of k = 8 elements in Table B. Here the threshold is 1, so "0" denotes an insignificant element and "1" a significant one. The test of K produces an output "1," because there is at least one "1" in the sequence. Therefore, it is split into the subsequences 0000 and 0110, whereupon the test of the first produces a "0" output, because it is insignificant. The second subsequence must be significant, so it is split into the dibits 01 and 10. The first dibit is significant, so "1" is the output, and it is split into 0 and 1. Since the

0 is insignificant, the second element must be significant, so only a "0" is the output for 0 and 1. The 10 is significant, so only a "1" is the output and it is likewise split into single digits 1 and 0, Since the first digit is 1, a "1" is the output, followed by the second digit's "0", since it too has to be tested. The resulting code is 1010110, seven digits compared to the original eight in this example.

Now when a large set of elements is there, the above iteration with the full set never started, because there is an optimum group size for the above iteration, based on the probability θ of a "0" or insignificant element. It turns out there is an equivalence between the encoding procedure above and the Golomb code, where it has been shown that a good rule for choosing group size k is k that satisfies $\theta^k = \frac{1}{2}$ but not less than $\frac{1}{2}$.

One usually do not know a priori the value of θ and it is often too inconvenient or practically impossible to pass through all the data to estimate it. Furthermore, this value may vary along the data set. Therefore, an adaptive method, based on a series of estimates of θ is a possible solution. One method that seems to work well is to start with group size k = 1 and double the group size on the next group iteration if no significant elements are found on the current iteration. Once significant elements are identified, we can estimate the probability θ as the number of insignificant elements identified so far divided by the total of elements thus far identified. For example, in the single group iteration of Table B, the probability estimate would be $\frac{6}{8} = 0.75$. Therefore, the next group size k = 2 is taken, according to the approximate formula above with $\theta = 0.75$. The accumulation of the counts of insignificant and total elements thus far identified and update these estimates as it pass through the data with its varying group sizes.

Chapter 6

Application and Simulation With QccPack Library

6.1 Application

6.1.1 ECG image compression

ELECTROCARDIOGRAM (ECG) [10] signal is a very useful source of information for physicians in diagnosing heart abnormalities. With the increasing use of ECG in heart diagnosis, such as 24 hour monitoring or in ambulatory monitoring systems, the volume of ECG data that should be stored or transmitted, has greatly increased. For example, a 3 channel, 24 hour ambulatory ECG, typically has storage requirement of over 50 MB. Therefore it is needed to reduce the data volume to decrease storage cost or make ECG signal suitable and ready for transmission through common communication channels such as phone line or mobile channel. So, it needs an effective data compression method.

The main goal of any compression technique is to achieve maximum data reduction while preserving the significant signal morphology features upon reconstruction. Wavelet transform is applied to the frames of ECG signal to 6 levels of decomposition.

The wavelet used is biorthogonal 9/7. The assumption is that each wavelet coefficient is represented by a fixed-point binary format, so it treat as an integer, because SPIHT algorithm works on integer values. The termination of encoding algorithm is specified by a CR (Compression Rate) value determined in the program. The output of the algorithm is a bit stream (0 and 1). This bit stream is used for reconstructing signal after compression. From it and by going through inverse of SPIHT algorithm the reconstructed of ECG signal will made.

6.1.2 DICOM Image Compression

DICOM (Digital Imaging and Communications in Medicine) standard [11] has been developed due to the emergence of different medical imaging modalities, which requires some standards for the purpose of storage and transmission. DICOM format has a header that contains information about the image, imaging modality and information about the patient. In addition to this, there is a field in the header which defines the Transfer Syntax Unique Identification that indicates the type of compression applied to the image data. The image data then follows the header. DICOM images are usually stored in the uncompressed raw data format. The disadvantage of this is that it increases the storage size and the bandwidth requirement for transmission. Currently, DICOM standard supports run-length coding (RLE), lossy JPEG, lossless JPEG, JPEG-LS (lossless and near lossless) and JPEG-2000 for compression of the image data.

Here, DICOM images are compressed using a progressive transmission coder, namely, SPIHT. The DICOM image header is transmitted first and the image details are then transmitted in successive stages. The receiver can terminate the transmitting data at any instant once the required information is satisfactory.

6.2 Introduction of QccPack

QccPack [12] is intended for use in the development of prototypes of coding and compression systems that extend established compression theory and algorithms into new application areas. It will also be of great use in academic research that employs or builds upon fundamental compression techniques. Its main reason for existence is to obviate the incessant reimplementation of common algorithms for the sake of application-specific details. For example, a developer should not re implement arithmetic coding every time the symbol-alphabet size changes or the number of contexts in the arithmetic-coder adaptive model is modified. Instead, QccPack allows the same implementation of arithmetic coding to be used, via a library call, regardless of application at hand. The benefits of this approach lie in the obvious efficiency associated with software reuse as well as in less obvious reliability-implementation details, which can often be complicated and not thoroughly described in academic literature, are figured out only once, for the general case, during the algorithm's implementation in QccPack. Subsequent developers can then have confidence that the implementation of a certain data-compression algorithm is correct and focus attention, instead, on higher-level application-design issues. QccPack has been released to the public as open-source software, meaning that researchers and developers have uninhibited access to the QccPack source code. This licensing scheme is an invaluable asset for academicians seeking to deploy data compression technology into new application areas-the developer can "peer under the hood" to see how things are done.

QccPack library routines are designed to be very general, it could possible that the exact functionality required by a developer is not supported by the current release of QccPack; in this case, the developer can modify, at will, the code as needed. Finally, the open-source paradigm encourages the developer to contribute such modifications and additions to the code for inclusion in future releases of QccPack so that others can benefit from the work as that has been benefited from past versions of QccPack.

6.2.1 Description of The QccPack Software

The essential component of the QccPack collection is a library (a static-linked library, libQccPack.a, and, if supported by the operating system, a dynamic-linked library, libQccPack.so) of procedures implementing a large variety of compression and coding algorithms. Application programs may make use of the QccPack library routines by linking the application against the library during compilation. Each library function is very general in its implementation so to be useful in a large variety of applications. Additionally, much of the functionality of the library routines has been provided in the form of stand-alone executable programs. Probably the prime importance these utility programs is that they provide examples of how to interface with many of the QccPack library routines. For rapid prototyping, the utility programs can be called from shell scripts to simulate the operation of complex coding and compression systems, before implementing all the system functionality into one stand-alone program. In most cases, the utility programs provided with QccPack merely handle initialization (reading input data files, initializing needed dynamic memory, etc.) while the true heart of the processing takes place in one or more library routines. QccPack is written entirely in ANSI C and currently consists of over 55,000 lines of source code implementing over 500 library routines and over 50 utility-program executables.

6.2.2 The QccPackSPIHT Module

The QccPackSPIHT module is a optional module for the QccPack library that provides an implementation of the Set Partitioning In Hierarchical Trees (SPIHT) algorithm for image compression. Since the SPIHT algorithm is patented, it is made available here under special license terms different from the GPL/LGPL licensing used for the main distribution of QccPack. The QccPackSPIHT license contains certain restrictions governing the terms and conditions for use, copying, distribution, and modification of the SPIHT algorithm implementation-briefly, only use in academic and non-commercial research is permitted, while all commercial use is prohibited.

The QccPackSPIHT module includes two utility executables, spihtencode and spihtdecode, to perform SPIHT encoding and decoding, respectively, for grayscale images.

The SPIHT coder has been modified extensively in recent versions of the QccPackSPIHT; users of versions prior to 0.24 are strongly encouraged to upgrade. Most notably, support for arithmetic coding has been added. Additionally, the performance of the "binary-uncoded" mode (i.e., encoding without arithmetic coding) has been improved and brought closer to the performance originally reported by Said and Pearlman.21 An extensive quantitative comparison between the QccPackSPIHT coder and the original reported results appears in the man-page documentation for the QccPackSPIHT module; to summarize, though, QccPackSPIHT is typically only 0.15 dB in PSNR below reported results in both binary-uncoded and arithmetic-coded modes.

The major functionalities currently implemented include:

A. Entropy coding.

   (1) Arithmetic coding including multiple-context adaptive and nonadaptive models.

   (2) Huffman coding.

   (3) Golomb and adaptive Golomb coding.

B. Scalar Quantization (SQ).

   (1) Uniform SQ.

   (2) Dead-zone SQ.

   (3) u-law and A-law SQ.

   (4) Lloyd algorithm for optimal SQ design.

C. Vector quantization (VQ).

(1) Generalized Lloyd algorithm (GLA) for VQ-codebook design.

(2) Full-search VQ encoding and decoding.

(3) Entropy-constrained-VQ (ECVQ) training, encoding, and decoding.

(4) Multistage VQ (MSVQ) (also called residual VQ (RVQ)) training, encoding, and decoding.

D. Adaptive vector quantization (AVQ).

(1) The generalized-threshold-replenishment (GTR) algorithm.

(2) The Paul algorithm.

(3) Gersho-Yano algorithm.

(4) Coding of side information.

E. Wavelet transforms, wavelet-based subband coding.

(1) Discrete wavelet transform (DWT) using first-generation filter banks and popular orthonormal and biorthogonal wavelets.

(2) Lifting implementations of DWT for popular wavelets.

(3) Two-dimensional DWT in the form of dyadic subband pyramids.

(4) Three-dimensional DWT in the form of dyadic subband pyramids as well as a packet transform.

(5) Shape-adaptive DWT (SA-DWT) for 1D and 2D signals.

(6) Redundant DWT (RDWT), aka, the algorithme a trous.

(7) The SR algorithm for wavelet-based image coding.

(8) The SFQ algorithm for wavelet-based image coding.

(9) The WDR algorithm for wavelet-based image coding.

(10) The 3D-WDR algorithm for wavelet-based image-cube coding.

(11) The tarp-filter algorithm for wavelet-based image coding.

(12) The 3D-tarp algorithm for wavelet-based image-cube coding.

(13) The TCE algorithm for wavelet-based image coding.

(14) The BISK algorithm for wavelet-based shape-adaptive image coding.

(15) The 3D-BISK algorithm for wavelet-based image-cube coding.

F. Error-correcting codes.

(1) Field arithmetic, including Gaussian-elimination matrix inversion.

(2) Reed-Solomon encoding and decoding.

(3) CRC codes.

(4) Trellis codes.

(5) Hard and soft Viterbi decoding.

G. Image processing.

(1) Routines for reading and writing gray and color still images and sequences of images (via PGM and PPM formats).

(2) Routines for reading and writing image-cube volumes.

(3) Image and image-sequence deinterlacing.

(4) Image differential-pulse-code modulation (DPCM).

(5) Color-space conversions: RGB, YUV, CIE XYZ, CIE UCS, CIE modified UCS.

(6) Block-based DCT and inverse DCT.

H. Video coding.

(1) The spatial-block algorithm for image-sequence coding.

(2) The RDWT-block algorithm for image-sequence coding.

(3) The RWMH algorithm for image-sequence coding.

(4) Block-based motion estimation and motion compensation.

(5) Encoding and decoding of motion-vector fields.

In addition to the standard functionalities listed above, there exist optional modules that can be added to the QccPack library.

A. QccPackSPIHT
The Set Partitioning in Hierarchical Trees (SPIHT) algorithm for wavelet-based image coding.

B. QccPackSPECK
The Set-Partitioning Embedded Block (SPECK) algorithm for wavelet-based image coding.

6.3 Simulation of SPIHT Algorithm With QCC-PACK

Syntax for SPIHT in QccPack
Encoder
spihtencode [-w wavelet] [-b boundary] [-pw pcpfile] [-nl num levels] [-noac] [-m mask] [-rd rd file] [-vo] rate imgfile bitstream
Decoder
spihtdecode [-w wavelet] [-b boundary] [-pw pcpfile] [-m mask] [-r rate] bitstream img-file

Script for SPIHT in QccPack

% >spihtencode -w CohenDaubechiesFeauveau.9-7.lft -b symmetric nl 5 1.830322

lenna.pgm.gz lenna.spiht.bit

Target rate: 1.830322 bpp

Actual rate: 1.830326 bpp

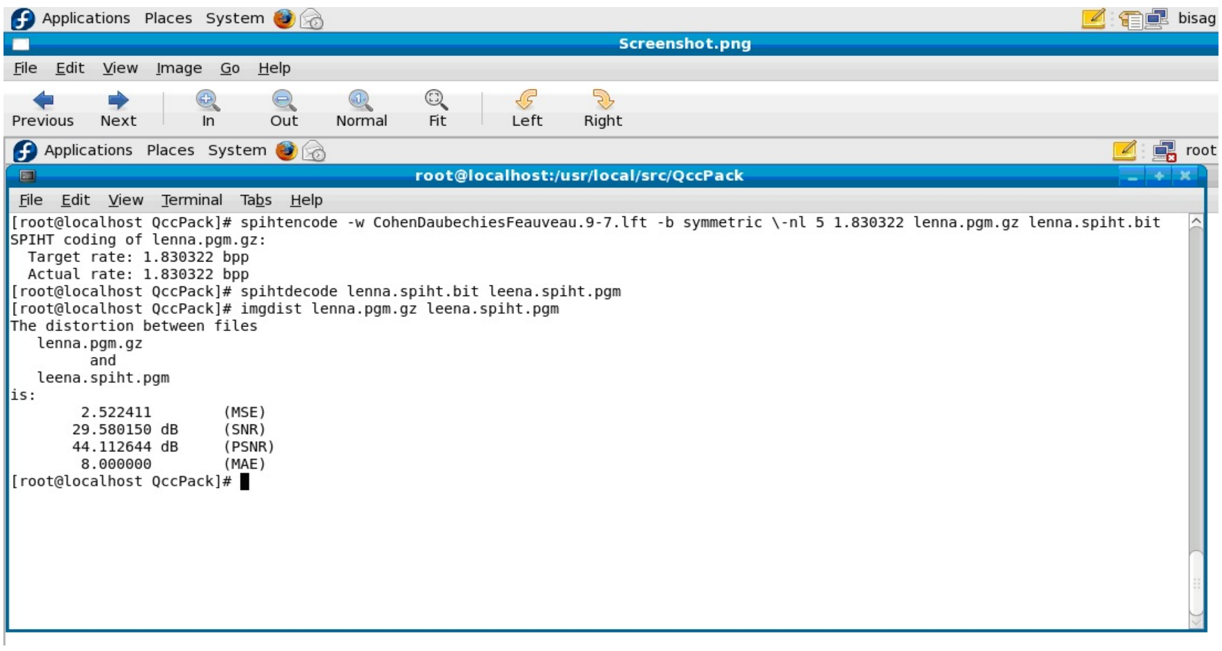
% >spihtdecode lenna.spiht.bit lenna.spiht.pgm

% >imgdist lenna.pgm.gz lenna.spiht.pgm

2.530071 (MSE)

29.566982 dB (SNR)

44.099476 dB (PSNR)


Applications Places System
bisag
Screenshot.png
File Edit View Image Go Help
Previous Next In Out Normal Fit Left Right
Applications Places System
root
root@localhost:/usr/local/src/QccPack
File Edit View Terminal Tabs Help

```
[root@localhost QccPack]# spihtencode -w CohenDaubechiesFeauveau.9-7.lft -b symmetric \-nl 5 1.830322 lenna.pgm.gz lenna.spiht.bit
SPIHT coding of lenna.pgm.gz:
  Target rate: 1.830322 bpp
  Actual rate: 1.830322 bpp
[root@localhost QccPack]# spihtdecode lenna.spiht.bit leena.spiht.pgm
[root@localhost QccPack]# imgdist lenna.pgm.gz leena.spiht.pgm
The distortion between files
   lenna.pgm.gz
         and
   leena.spiht.pgm
is:
        2.522411        (MSE)
       29.580150 dB     (SNR)
       44.112644 dB     (PSNR)
        8.000000        (MAE)
[root@localhost QccPack]# 
```

Figure 6.1: Snapshoot of Simulator

6.3.1 Simulation Result

Performance Analysis of SPIHT Algorithm by Changing Different Parameter

Changing compression rate.

CR	MSE	SNR(dB)	PSNR(dB)
1:2	0.07	45.63	59.24
1:4	1	33.25	46.89
1:8	4.3	28.13	41.73
1:16	10.0	24.29	37.93

Table A: Result of Changing Compression Rate for Lenna 512X512 with CDF 9-7 Wavelet and Arithmetic Coding @ Level 5

As figures in table A shows with the more compression rate the PSNR decrease means quality of image degrades. But when consideration done at 1:4 or 1:8 compression ratio SPIHT gives batter PSNR. The compression is consider as lossless if we get PSNR 45dB or more.

Changing wavelet

	CDF 5-3	CDF 9-7	LWT
MSE	410.51	1	4381
SNR(dB)	7.5	28.10	-2.8
PSNR(dB)	21	46.00	11.11

Table B: Result of Changing Wavelet for Lenna 512X512 with CR 1:8 @ Level 5 and Arithmetic Coding Used

The figures in table B shows PSNR of Image with CR 1:8 @ level 5 and Arithmetic coding used and the type of wavelet differs. The result shows CDF 9-7 give batter PSNR that is batter quality of reconstructed image. The important reason why CDF 9-7 gives batter result than CDF 5-3 is the order of filter used for DWT by that wavelets. so further simulations are done with CDF 9-7 only.

Changing no of level

	@Level = 2	@Level = 4	@ level = 5	@Level = 8
MSE	5.9	4.8	4.3	4.8
SNR(dB)	26.70	26.74	28.13	25.87
PSNR (dB)	40.40	41.23	41.73	41.27

Table C: Result of Changing no of Level for Lenna 512X512 with CDF 9-7 Wavelet and CR 1:8 and Arithmetic Coding Used

The figures of table C shows result of Changing no of level For lenna 512X512 with CDF 9-7 wavelet and CR 1:8 and Arithmetic coding used. The change of no of decomposition level results very little change in MSE and PSNR. So further simulations are done with 5 level of decomposition.

Arithmetic code used or not

	With Arithmetic Code	Without Arithmetic Code
MSE	3.54	5.4
SNR(dB)	28.10	26.25
PSNR(dB)	42.63	40.78

Table D: Result of Arithmetic Code Used or not for Lenna 512X512 with CDF 9-7 Wavelet and CR 1:8

The figures of table D shows Result of Arithmetic code used or not For lenna 512X512 with CDF 9-7 wavelet and CR 1:8. With use of Arithmetic coding on SPIHT coded bit stream gives batter PSNR. so for further simulations we are using Arithmetic coding on SPIHT coded bit stream.

Compression of Binary Image

Scanned Signature

	Signature Image 1	Signature Image 2	Signature Image 3
CR	1:8	1:8	1:8
MSE	0.6	6.99	0.34
SNR(dB)	34.24	28.63	39.54
PSNR(dB)	49.63	39.68	52.75

Table E: Scanned Signature Image

Scanned Fingerprint

	Fingerprint Image 1	Fingerprint Image 2	Fingerprint Image 3
CR	1:8	1:8	1:8
MSE	3.39	1.3	1.28
SNR(dB)	29.21	36.94	37.27
PSNR(dB)	42.81	46.69	47.04

Table F: Scanned Fingerprint

Figures in table E and F shows the PSNR for the binary images like fingerprint image and scanned signature of person. The PSNR of both images are mostly more than 45dB so it can consider as lossless image.

Compression of ECG Image

	ECG image with grid and single cycle		ECG image with grid and multiple cy cle		ECG image without grid and single cycle		ECG image without grid and multiple cycle	
CR	1:8	1:16	1:8	1:16	1:8	1:16	1:8	1:16
MSE(dB)	21.29	216.8	43	118.3	0	1.29	0.9	1.9
PSNR(dB)	34.84	31.73	31.73	27.39	60	47.11	50.2	45.19

Table G: Results of Compression of ECG Images

Figures in table G shows the PSNR of compressed ECG images. Result shows the PSNR of ECG image without grid is batter than with grid and if compression is done at 1:8 compression ratio, PSNR exceeds 45 dB. So by removing grid from ECG image and sending its scale information as well as starting and ending point simultaneously with Image data we get more compressed image and batter image quality of reconstructed image. ”There is no information loss in reconstructed image.” this is the comment given by doctors after observing both original and reconstructed image.

Compression of DICOM Images

	X-ray of chest		X-ray of brain		X-ray of hand	
CR	1:8	1:16	1:8	1:16	1:8	1:16
MSE(dB)	0.15	0.62	5.2	23.70	9.09	38.53
PSNR(dB)	56.25	50.18	40.89	34.38	38.54	32.27

Table H: Results of Compression of DICOM Images

Figures in table H shows the PSNR of compressed DICOM images. In result mostly PSNR exceeds 45 dB. And ”There is no information loss in reconstructed image.” this is the comment given by doctors after observing both original and reconstructed image.

Result of Modification of threshold

	Actual CR	Target CR
ECG image 1	1.0000	1.0
ECG image 2	0.4998	0.5
ECG image 3	1.4985	1.5
DICOM image 1	0.9991	1.0
DICOM image 1	0.4984	0.5
DICOM image 1	1.4983	1.5

Table I: Result of Threshold Modification

The more compression is taken by making threshold data dependant instead of fixed number.

Result of Adaptive Partitioning

	Actual CR	Target CR
ECG image 1	0.9963	1.0
ECG image 2	0.4979	0.5
ECG image 3	1.4942	1.5
DICOM image 1	0.9962	1.0
DICOM image 1	0.4981	0.5
DICOM image 1	1.4959	1.5

Table J: Result of Adaptive Partitioning

The figures in table J shows more compression is taken by making partition adaptive.

Chapter 7

Conclusion and Future Scope

7.1 Conclusion

The various wavelet based image coding schemes are discussed in this Thesis. Each of these schemes finds use in different applications owing to their unique characteristics. Though there a number of coding schemes available, the need for improved performance and wide commercial usage, demand newer and better techniques to be developed. Here presented an algorithm that operates through set partitioning in hierarchical trees (SPIHT) and accomplishes completely embedded coding. The results shows the performance of SPIHT algorithm for gray images. Also modification in selection of Threshold and making partitions adaptive gives the more compression with same reconstructed image quality. As results shows for ECG and DICOM images this algorithm gives excellent result after compression and this is accepted by Doctors also. The results of the coding algorithm with its embedded code and fast execution are so impressive that it is used for standardization in future image compression systems.

7.2 Future Scope

In future this algorithm can further modified by Progressive Transmission, Bit-Plane Coding, and source coding so that one can get encrypted bit stream for secure transmission of data. With proper modification and selection of parameters we can use this algorithm for Satellite image compression.

Appendix A

QccPack Installation Procedure

==

Q C C P A C K

Version 0.58, 13-feb-2009

Copyright (C) 1997-2009 James E. Fowler

*** Warning: this is pre-release code ***

*** There are probably bugs ***

==

AVAILABILITY:

For updated versions of QccPack, or optional modules for QccPack, consult the QccPack website:

http://qccpack.sourceforge.net

==

INSTALLATION:

General Installation Notes:

The following gives the steps for the building and installation of QccPack. It should be a relatively straightforward process to get QccPack to compile on any UNIX system, and should be thoroughly operational on such systems, although it has only really been tested on Linux. QccPack is supported on Microsoft Windows, but only through use of Cygwin, a free UNIX emulation environment for Windows.
The general procedure for installation of QccPack is described below. Platform-specific installation instructions follow. Windows users, in particular, be sure to read the instructions concerning installation of Cygwin.
Refer to documentation accompanying optional QccPack modules for information on the installation of those modules.

General Installation Steps:

A. Untar the QccPack distribution to a directory in which building is to be performed. The directory /usr/local/src/QccPack is suggested, but use whatever location you want.

B. Copy the file QccPack.config.xxxx to QccPack.config, where "xxxx" is the operating system that you are using. For example, if using Linux, type
cp QccPack.config.linux QccPack.config
If a QccPack.config.xxxx file is not provided for your operating system, use the QccPack.config.linux file and pay close attention to the various configuration options during the configuration-editing process which is the next step.

C. Edit the file QccPack.config :

(1) Set QCCPACK to be the absolute path of directory into which the files were untarred; the default for this path is
/usr/local/src/QccPack

(2) Change QCCPACK_INSTALL so that QCCPACK_INSTALL_LIB_PATH,
QCCPACK_INSTALL_INCLUDE_PATH,
QCCPACK_INSTALL_BIN_PATH,
QCCPACK_INSTALL_MAN_PATH,
and QCCPACK_INSTALL_DATA_PATH point to the desired location of the final installation of the libraries, header files, binaries, man pages, and data files, respectively. These default to lib, include, bin, man directories in /usr/local, and
/usr/local/share/QccPack for the data files, but change them if you like.

(3) Uncomment the "define" lines for the optional QccPack modules that you wish to install with QccPack. Source code for these modules must be downloaded separately from QccPack; refer to installation instructions specific to each module.

(4) QccPack comes with HTML versions of its man pages. By default, these will be installed in (QCCPACK_INSTALL)/doc/QccPack. If you do not want these pages installed, comment out (with XCOMM's) the definition of HTMLMAN.

(5) Define CC and CFLAGS as needed for your desired ANSI C compiler and associated flags. The GNU compiler, gcc, is recommended.

(6) If your system/compiler supports the creation of shared objects (i.e. dynamic libraries), set SHARED_OBJ_FLAGS and LD appropriately. Alternatively, you can define NO_DYNAMIC_LIB to suppress the creation of the dynamic library and cause the static linking of the utility executables.

(7) Set the definitions of QCCCOMPRESS and QCCUNCOMPRESS to point to where gzip and gunzip reside on your system (absolute path). Option-

ally, these definitions can be changed to use compress and uncompress, if so desired.

D. Type "imake".

E. Type "make Makefiles".

F. Type "make".

G. When you're happy that things compiled OK, type "make install" to put the libraries, header files, binaries, and man pages into their final installation locations.

H. A "make clean" will perform a complete uninstallation of QccPack: the build directory tree will be cleaned of all executables and object files, AND the libraries, header files, binaries, and man pages will be removed from the final installation locations. You can then remove the build tree with "rm -rf ¡dir¿".

I. Type "man QccPack" or point your browser to file: (QCCPACK_INSTALL)/doc/QccPack.3.html to view the available documentation.

==

ACKNOWLEDGMENTS:

A large portion of the QccPack library was written while James E. Fowler held a postdoctoral position at Laboratoire I3S, Universit de Nice-Sophia Antipolis, Sophia Antipolis, France, and was based upon work supported by the National Science Foundation under Grant No. INT-9600260. Subsequent support was provided by the National Science Foundation under Grant Nos. ACI-9982344 and CCR-0310864.

Appendix B

List of Abbreviations

SPIHT	Set Partitioning In hierarchical Tree
DWT	Discrete wavelet transform
PSNR	Peak Signal to Noise Ratio
MSE	Mean Square Error
DCT	Discrete Cosine Transform
DFT	Discrete Fourier
JPEG	Joint Photographic Expert Group
IFS	Iterative Function System
PVRQ	Predictive Residual Vector Quantizer
EZW	Embedded Zero Tree Wavelet
SPECK	Set Partitioned Embedded bloCK Coder
ECG	Electrocardiogram
DICOM	Digital Imaging and Communications in Medicine
ECVQ	Entropy-constrained-VQ

References

[1] Rupali Arjun More Veera Pratiksh Mahendra. ”implementation of advanced image compression algorithm using wavelet transform”. Technical report, Department of Electronics Engineering, L.T. college of engineering, Navi Mumbai, 2006.

[2] Natarajan T. Ahmed, N. and Rao K. R. ”discrete cosine transform”. *IEEE Trans. Computers, vol. C-23*, January 1974.

[3] W. B. Pennebaker and J. L. Mitchell. *”JPEG - Still Image Data Compression Standards”*. Van Nostrand Reinhold, 1993.

[4] P.Schneider B.Jarabek and A.uhl . ”comparison of lossy image compression method applied to photorealistic and graphical images”. Technical report.

[5] Ms R Karthiga R.Sudhakar and S.Jayaraman. ”image compression using coding of wavelet coefficients a survey”. Technical report, Department of Electronics and Communication Engineering, PSG College of Technology Peelamedu,Coimbatore.

[6] Danny Lazar Amir Averbuch and Moshe Israeli. ”image compression using wavelet transform and multiresolution decomposition”. *IEEE Transactions on Image Processing, Vol. 5*, JAN 1996.

[7] Aldo Morales and Sedig Agili. ”implementing the spiht algorithm in matlab”. Technical report, Department of Electrical Engineering, Penn State University at Harrisburg.

[8] Upena Dalal Mayana J. Shah and Hiren Mewada. ”low bit rate spiht coding for wavelet based image compression”.

[9] William A. Pearlman and Amir Said. ”set partition coding: Part i of set partition coding and image wavelet coding systems”. Technical report, Department of Electrical, Computer and System Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180-3590, USA, 2008.

[10] Morteza Moazami-Goudarzi Mohammad Pooyan, Ali Taheri and Iman Saboori. "wavelet compression of ecg signals using spiht algorithm". Technical report, World Academy of Science, Engineering and Technology, 2005.

[11] B. Ramakrishnan and N. Sriraam. "compression of dicom images based on wavelet and spiht for telemedicine application". Technical report, Faculty of Information Technology, Multimedia University, Malaysia.

[12] James E. Fowler. "qccpack: An open-source software library for quantization, compression, and coding". Technical report, Dept. of Electrical and Computer Engineering, Mississippi State University, Mississippi State, MS, 2000.