



GRADUATE SCHOOL
EAST TENNESSEE STATE UNIVERSITY

East Tennessee State University
Digital Commons @ East
Tennessee State University

Electronic Theses and Dissertations

Student Works

12-2001

Image Compression by Wavelet Transform.

Panrong Xiao
East Tennessee State University

Follow this and additional works at: <https://dc.etsu.edu/etd>

 Part of the [Computer Sciences Commons](#)

Recommended Citation

Xiao, Panrong, "Image Compression by Wavelet Transform." (2001). *Electronic Theses and Dissertations*. Paper 58. <https://dc.etsu.edu/etd/58>

This Thesis - unrestricted is brought to you for free and open access by the Student Works at Digital Commons @ East Tennessee State University. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of Digital Commons @ East Tennessee State University. For more information, please contact digilib@etsu.edu.

Image Compression By Wavelet Transform

A thesis
presented to
the faculty of the Department of Computer and Information Sciences
East Tennessee State University

In partial fulfillment
of the requirements for the degree
Master of Science in Computer Science

by
Panrong Xiao
August 2001

Martin Barrett, Chair
Dong Hong, Co-chair
Bob Riser

Keywords: Wavelet, Transform, Compression, EZW, Coding

ABSTRACT

Image Compression By Wavelet Transform

by

Panrong Xiao

Digital images are widely used in computer applications. Uncompressed digital images require considerable storage capacity and transmission bandwidth. Efficient image compression solutions are becoming more critical with the recent growth of data intensive, multimedia-based web applications.

This thesis studies image compression with wavelet transforms. As a necessary background, the basic concepts of graphical image storage and currently used compression algorithms are discussed. The mathematical properties of several types of wavelets, including Haar, Daubechies, and biorthogonal spline wavelets are covered and the Embedded Zerotree Wavelet (EZW) coding algorithm is introduced. The last part of the thesis analyzes the compression results to compare the wavelet types.

ACKNOWLEDGEMENTS

I would like to thank Dr. Barrett, Dr. Hong, and Prof. Riser who helped in making this thesis a reality.

CONTENTS

	Page
ABSTRACT	2
ACKNOWLEDGEMENTS	3
LIST OF TABLES	7
LIST OF FIGURES	8
 Chapter	
1. INTRODUCTION	10
2. LITERATURE REVIEW	12
Color Space and Human Perception	12
RGB Space	12
YUV Space	13
YIQ Space	14
YCrCb Space	14
Comparison of Color Spaces	14
Compression Techniques	16
Lossless Compression	16
Run Length Encoding	16
Huffman Coding	16
Lempel-Ziv-Welch (LZW) Encoding	17
Arithmetic Coding	17
Lossy Compression Methods	18
Vector Quantization	18
Predictive Coding	18
Fractal Compression	19
Transform Based Image Compression	19
DCT Based Transform Coding	20
MPEG Video Standard	20
Wavelet Transform	20

Reason to Use Wavelet Based Compression	22
Summary	23
3. WAVELET THEORY.....	24
Fourier Transform	24
Windowed Fourier Transform	24
Wavelet Transform	25
Multiresolution Analysis and the Scaling Function	26
Discrete Wavelet Transform Implementation Using Filters	27
Compact Support Orthogonal Wavelets	27
Haar Wavelet	29
Daubechies Orthogonal Wavelets	30
Biorthogonal and Spline Wavelets	33
Spline Filters	34
A Spline Variant with Less Dissimilar Lengths	37
Comparison of Wavelet Properties	38
Summary	39
4. WAVELET APPLIED IN IMAGE COMPRESSION	40
Baseline Schema	40
Discrete Wavelet Transform (DWT)	41
Daubechies 4-tap Wavelet	41
Biorthogonal Spline2_2 Wavelet	42
Decomposition and Composition Algorithms	43
Embedded Zerotree Wavelet (EZW) Coding	47
Subbands in the Wavelet Transform Blocks	47
EZW Coding	49
EZW Algorithm	50
Entropy Coding	54
Specification of the Wavelet Image File: LET	54
Summary	55
5. EXPERIMENTS	56
Objective Distortion Criteria	56
Objective Quality	57

Subjective Quality	58
Quantization Methods	59
JPEG V.S. LET	60
Timing Analysis	63
Summary	65
6. CONCLUSION	66
BIBLIOGRAPHY	68
VITA	70

LIST OF TABLES

Table	Page
1. Coefficients for the 4-tap Daubechies Low-pass Filter	31
2. Coefficients for the 6-tap Daubechies Low-pass Filter	31
3. Coefficients for the 10-tap Daubechies Low-pass Filter	32
4. Filter Coefficients with $k = 2, \tilde{k} = 2$	36
5. Filter Coefficients with $k = 4, \tilde{k} = 2$	37
6. Filter Foefficients with $l = k = 4$	37
7. Property Comparison of Three Kinds of Wavelets	38
8. Comparison of Compression Results by Using Different Wavelets	58
9. Comparison of Subjective Quality	58
10. Comparison of Compression Results by Using Different Quantization Methods	59
11. Comparison of Compression Results between JPEG and LET	60

LIST OF FIGURES

Figure	Page
1. RGB Space	13
2. Example of YUV Space	13
3. The Comparisons of the RGB and the $YCbCr$ Color Spaces	15
4. Haar Wavelet	21
5. (a) Original Lena Image (b) Reconstructed image to show blocking artifacts	23
6. Harr Scaling Function	30
7. Harr Mother Wavelet	30
8. Daubechies 4-tap Scaling and Wavelet Functions	31
9. Daubechies 6-tap Scaling and Wavelet Functions	32
10. Daubechies 10-tap Scaling and Wavelet Functions	32
11. Filter Structure and the Associating Wavelets	34
12. Spline Examples with $k = 2, \tilde{k} = 2$	35
13. Spline Examples with $k = 4, \tilde{k} = 2$	36
14. Spline Examples with $l = k = \tilde{k} = 4$	38
15. The Baseline Schema of MinImage	40
16. The 1-D Wavelet Decomposition Algorithm in MinImage	44
17. The 1-D Wavelet Composition Algorithm in MinImage	45
18. The 2D Wavelet Decomposition Algorithm	46
19. The 2D Wavelet Decomposition of An Image	47
20. Subbands after the 1-D Wavelet Decomposition	48
21. Subbands in a Wavelet Transform Block after the 2-D Wavelet	48
22. The Relations between Wavelet Coefficients in Different Subbands as Quad-trees	50
23. Algorithm of the Main Loop	51
24. Two Different Scan Orders	52
25. Algorithm to Determine if A Coefficient is A Root of Zerotree	52
26. Algorithm for the Dominant Pass	53
27. Algorithm for the Subordinate Pass	54

28. LET File Format	54
29. LET File Header Format	55
30. Lenna 256*256	57
31. Comparison of JPEG and LET	61
32. Original 512*512 Grayscale Lenna	62
33. Comparison of Decomposition and Composition Time	63
34. Comparison of EZW Coding and Decoding Time	64
35. Comparison of EZW Coding Time with Different Wavelets and Different Compression Ratios	65

CHAPTER 1

INTRODUCTION

Data compression is the process of converting data files into smaller files for efficiency of storage and transmission. As one of the enabling technologies of the multimedia revolution, data compression is a key to rapid progress being made in information technology. It would not be practical to put images, audio, and video alone on websites without compression

What is data compression? And why do we need it? Many people may have heard of JPEG (Joint Photographic Experts Group) and MPEG (Moving Pictures Experts Group), which are standards for representing images and video. Data compression algorithms are used in those standards to reduce the number of bits required to represent an image or a video sequence. Compression is the process of representing information in a compact form. Data compression treats information in digital form, that is, as binary numbers represented by bytes of data with very large data sets. For example, a single small 4" × 4" size color picture, scanned at 300 dots per inch (dpi) with 24 bits/pixel of true color, will produce a file containing more than 4 megabytes of data. At least three floppy disks are required to store such a picture. This picture requires more than one minute for transmission by a typical transmission line (64k bit/second ISDN). That is why large image files remain a major bottleneck in a distributed environment. Although increasing the bandwidth is a possible solution, the relatively high cost makes this less attractive. Therefore, compression is a necessary and essential method for creating image files with manageable and transmittable sizes.

In order to be useful, a compression algorithm has a corresponding decompression algorithm that, given the compressed file, reproduces the original file. There have been many types of compression algorithms developed. These algorithms fall into two broad types, lossless algorithms and lossy algorithms. A lossless algorithm reproduces the original exactly. A lossy algorithm, as its name implies, loses some data. Data loss may be unacceptable in many applications. For example, text compression must be lossless because a very small difference can result in statements with totally different meanings. There are also many situations where loss may be either unnoticeable or acceptable. In image compression, for example, the exact reconstructed value of each sample of the image is not necessary. Depending on the quality required of the reconstructed image, varying amounts of loss of information can be accepted.

This thesis is concerned with a certain type of compression that uses wavelets. Wavelets, introduced by J. Morlet [23], are used to characterize a complex pattern as a series of simple patterns and coefficients that, when multiplied and summed, reproduce the original pattern. There are a variety of wavelets for use in compression.

Several methods are compared on their ability to compress standard images and the fidelity of the reproduced image to the original image.

In order to compare wavelet methods, a software tool called MinImage was used. MinImage was originally created to test one type of wavelet [6]. Additional functionality was added to MinImage to support other wavelet types. The results of MinImage's performance and output were then analyzed to compare the wavelet types.

The remainder of this thesis is organized as follows. Chapter 2 surveys the compression algorithms currently in use. These include both lossless and lossy methods. Because many compression algorithms are applied to graphical images, the basic concepts of graphical image storage are also discussed.

Chapter 3 covers the mathematical properties of wavelets. Several types of wavelets are discussed, including Haar, Daubechies, and biorthogonal spline wavelets.

Chapter 4 discusses how to apply wavelet theory to image compression. The Embedded Zerotree Wavelet (EZW) coding algorithm is introduced to code the transformed wavelet coefficients.

Chapter 5 analyzes the compression results. These include the subjective and objective qualities of reconstructed images for different wavelet types, timing of composition and decomposition for different wavelet types, and timing of EZW coding algorithm for different compression ratios.

Chapter 6 summarizes the results of the study and provides suggestions for future research.

CHAPTER 2

LITERATURE REVIEW

In this chapter, we present the some common compression algorithms, which include both lossless and lossy methods. Since many compression algorithms are applied to graphical images, the basic concepts of graphical image storage (color space) are also discussed.

Color Space and Human Perception

A digital color image can be viewed as a three valued (channels) positive function $I=I(x,y)$ defined onto a plane. Its algebraic representation is obtained through an N by M by 3 matrix A . Thus, each entry of A is a three component integer vector (pixel color) expressing an intensity value at discrete location (x,y) with a precision p (for instance, one bit for each channel). Each component or layer of the image can be viewed as a single channel image, which, under particular conditions, can be analyzed independently from the others. This is not the case for RGB space, for example (see section 2.2.1), because, if two channels are fixed, human visual perception is very sensitive to small changes of the value of the remaining channel. Thus, even though RGB is the most common storage format for images, other formats may be better for compression.

The key step in lossy data compression in which data cannot be recovered exactly is the quantization phase, which exploits a data reduction based on their low information content. This is not optimal for RGB images. Nevertheless, for three layers, this can lead to the elimination of some low coefficients in a channel in a certain spatial location, even though the corresponding coefficients in the other layers are not eliminated because they carry high information content. When reconstructing the image at that location, a high visual distortion is introduced. The assumption of analyzing the three layers separately is valid only if they are not correlated with respect the visual appearance. [12]

RGB Space

RGB is perhaps the simplest and most commonly used color space (see Figure 1). It uses proportions of red, green, and blue that are scaled to a minimum and maximum values for each component (for example, 0x00 through 0xFF, or 0.0 through 1.0). Most colors in the visible spectrum can be recreated, although not completely. This scheme is based on the additive properties of color. [12]

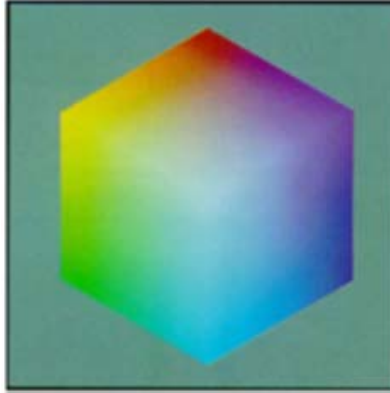


Figure 1. RGB Space [12]

YUV Space

YUV was originally used for PAL (European standard) analog video (see Figure 2). To convert from RGB to YUV spaces, the following equations can be used:

$$Y = 0.299 R + 0.587 G + 0.114 B$$

$$U = 0.492 (B - Y)$$

$$V = 0.877 (R - Y)$$

Any errors in the resolution of the luminance (Y) are more important than the errors in the chrominance (U, V) values. The luminance information can be coded using higher bandwidth than the chrominance information [12].

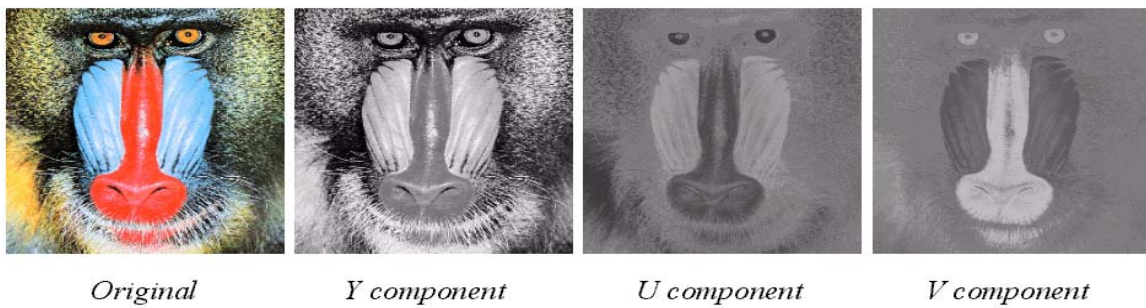


Figure 2. Example of YUV Space [12]

YIQ Space

YIQ is used in the U.S. television standard, NTSC (National Television System Committee). It is similar to YUV, except that its color space is rotated 33 degrees clockwise, so that I is the orange-blue axis, and Q is the purple-green axis. The equations to convert from RGB to YIQ are [12]:

$$Y = 0.299 R + 0.587 G + 0.114 B$$

$$I = 0.74 (R - Y) - 0.27 (B - Y) = 0.596 R - 0.275 G - 0.321 B$$

$$Q = 0.48 (R - Y) + 0.41 (B - Y) = 0.212 R - 0.523 G + 0.311 B$$

YCrCb Space

YCrCb is a subset of YUV that scales and shifts the chrominance values into the range of 0 to 1. The linear transform from RGB to YCrCb generates one luminance space Y and two chrominance (Cr and Cb) spaces [12].

$$Y = 0.299 R + 0.587 G + 0.114 B$$

$$Cr = ((B - Y) / 2) + 0.5$$

$$Cb = ((R - Y) / 1.6) + 0.5$$

Comparison of Color Spaces

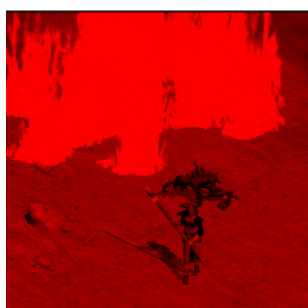
Among these three color spaces, only YCrCb is the proper one used in the digital image processing because

- 1) There is no correlation among the spaces of YCrCb, so each space can be analyzed separately.
- 2) Human eyes are more sensitive to the change of brightness than of color, so Cr and Cb spaces can be compressed more heavily than Y space to get better compression ratio.

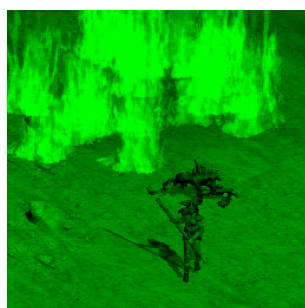
Figure 3 is the comparison of RGB space and YCrCb space. [6]



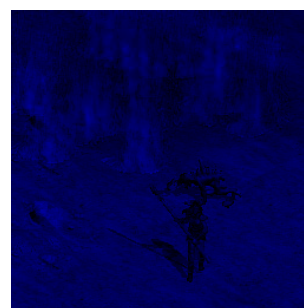
Original image



R



G



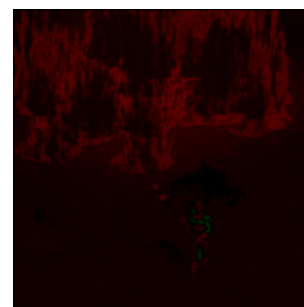
B



Y



Cb



Cr

Figure 3. The Comparisons of the RGB and the $YCbCr$ Color Spaces

Compression Techniques

Compression takes an input X and generates a representation X_C that hopefully requires fewer bits. There is a reconstruction algorithm that operates on the compressed representation X_C to generate the reconstruction Y . Based on the requirements of reconstruction, data compression schemes can be divided into two broad classes. One is lossless compression, in which Y is identical to X . Examples of lossless methods are Run Length coding, Huffman coding, Lempel/Ziv algorithms, and Arithmetic coding. The other is lossy compression, which generally provides much higher compression than lossless compression but allows Y to be different from X .

Lossless Compression

If data have been losslessly compressed, the original data can be recovered exactly from the compressed data. It is generally used for applications that cannot allow any difference between the original and reconstructed data.

Run Length Encoding. Run length encoding, sometimes called recurrence coding, is one of the simplest data compression algorithms. It is effective for data sets that are comprised of long sequences of a single repeated character. For instance, text files with large runs of spaces or tabs may compress well with this algorithm. Old versions of the arc compression program used this method. [3]

RLE finds runs of repeated characters in the input stream and replaces them with a three-byte code. The code consists of a flag character, a count byte, and the repeated characters. For instance, the string ``AAAAAABBBBCCCC" could be more efficiently represented as ``*A6*B4*C5". That saves us six bytes. Of course, since it does not make sense to represent runs less than three characters in length with a code, none is used. Thus ``AAAAAABBBCCDDDD" might be represented as ``*A6BBCCC*D4". The flag byte is called a sentinel byte.

Huffman Coding. Huffman coding, developed by D.A. Huffman [3], is a classical data compression technique. It has been used in various compression applications, including image compression. It uses the statistical property of characters in the source stream and then produces respective codes for these characters. These codes are of variable code length using an integral number of bits. The codes for characters having a higher frequency of occurrence are shorter than those codes for characters having lower frequency. This simple idea causes a reduction in the average code length, and thus the overall size of compressed data is smaller than the original. Huffman coding is based on building a *binary tree* that holds all characters in the source at its *leaf nodes*, and with their corresponding characters' probabilities at the side. The tree is built by going through the following steps:

1. Each of the characters is initially laid out as leaf node; each leaf will eventually be connected to the tree. The characters are ranked according to their weights, which represent the frequencies of their occurrences in the source.

2. Two nodes with the lowest weights are combined to form a new node, which is a parent node of these two nodes. This parent node is then considered as a representative of the two nodes with a weight equal to the sum of the weights of two nodes. Moreover, one child, the left, is assigned a "0" and the other, the right child, is assigned a "1".

3. Nodes are then successively combined as above until a binary tree containing all of nodes is created.

4. The code representing a given character can be determined by going from the root of the tree to the leaf node representing the alphabet. The accumulation of "0" and "1" symbols is the code of that character.

By using this procedure, the characters are naturally assigned codes that reflect the frequency distribution. Highly frequent characters will be given short codes, and infrequent characters will have long codes. Therefore, the average code length will be reduced. If the count of characters is very biased to some particular characters, the reduction will be very significant.

Lempel-Ziv-Welch (LZW) Encoding. This original approach is given by J. Ziv and A. Lempel in 1977 [17]. T. Welch's refinements to the algorithm were published in 1984 [18]. LZW compression replaces strings of characters with single codes. It does not do any analysis of the incoming text. Instead, it just adds every new string of characters it sees to a table of strings. Compression occurs when a single code is output instead of a string of characters.

The code that the LZW algorithm outputs can be of any arbitrary length, but it must have more bits in it than a single character. The first 256 codes (when using eight bit characters) are by default assigned to the standard character set. The remaining codes are assigned to strings as the algorithm proceeds.

There are three best-known applications of LZW: UNIX compress (file compression), GIF (image compression), and V.42 bis (compression over Modems). [3]

Arithmetic Coding. Arithmetic coding is also a kind of statistical coding algorithm similar to Huffman coding. However, it uses a different approach to utilize symbol probabilities, and performs better than Huffman coding. In Huffman coding, optimal codeword length is obtained when the symbol probabilities are of the form $(1/2)^x$, where x is an integer. This is because Huffman coding assigns code with an integral number of bits. This form of symbol probabilities is rare in practice. Arithmetic coding is a statistical coding method that solves this problem. The code form is not restricted to an integral number of bits. It can assign a code as a fraction of a bit.

Therefore, when the symbol probabilities are more arbitrary, arithmetic coding has a better compression ratio than Huffman coding. In brief, this can be considered as grouping input symbols and coding them into one long code. Therefore, different symbols can share a bit from the long code.

Although arithmetic coding is more powerful than Huffman coding in compression ratio, arithmetic coding requires more computational power and memory. Huffman coding is more attractive than arithmetic coding when simplicity is the major concern. [3]

Lossy Compression Methods

Lossy compression techniques involve some loss of information, and data cannot be recovered or reconstructed exactly. In some applications, exact reconstruction is not necessary. For example, it is acceptable that a reconstructed video signal is different from the original as long as the differences do not result in annoying artifacts. However, we can generally obtain higher compression ratios than is possible with lossless compression.

Vector Quantization. Vector Quantization (VQ) is a lossy compression method. It uses a codebook containing pixel patterns with corresponding indexes on each of them. The main idea of VQ is to represent arrays of pixels by an index in the codebook. In this way, compression is achieved because the size of the index is usually a small fraction of that of the block of pixels.

The main advantages of VQ are the simplicity of its idea and the possible efficient implementation of the decoder. Moreover, VQ is theoretically an efficient method for image compression, and superior performance will be gained for large vectors. However, in order to use large vectors, VQ becomes complex and requires many computational resources (e.g. memory, computations per pixel) in order to efficiently construct and search a codebook. More research on reducing this complexity has to be done in order to make VQ a practical image compression method with superior quality. [3]

Predictive Coding. Predictive coding has been used extensively in image compression. Predictive image coding algorithms are used primarily to exploit the correlation between adjacent pixels. They predict the value of a given pixel based on the values of the surrounding pixels. Due to the correlation property among adjacent pixels in image, the use of a predictor can reduce the amount of information bits to represent image.

This type of lossy image compression technique is not as competitive as transform coding techniques used in modern lossy image compression, because predictive techniques have inferior compression ratios and worse reconstructed image quality than those of transform coding. [3]

Fractal Compression. The application of fractals in image compression started with M.F. Barnsley and A. Jacquin [19]. Fractal image compression is a process to find a small set of mathematical equations that can describe the image. By sending the parameters of these equations to the decoder, we can reconstruct the original image.

In general, the theory of fractal compression is based on the contraction mapping theorem in the mathematics of metric spaces. The Partitioned Iterated Function System (PIFS), which is essentially a set of contraction mappings, is formed by analysing the image. Those mappings can exploit the redundancy that is commonly present in most images. This redundancy is related to the similarity of an image with itself, that is, part A of a certain image is similar to another part B of the image, by doing an arbitrary number of contractive transformations that can bring A and B together. These contractive transformations are actually common geometrical operations such as rotation, scaling, skewing and shifting. By applying the resulting PIFS on an initially blank image iteratively, we can completely regenerate the original image at the decoder. Since the PIFS often consists of a small number of parameters, a huge compression ratio (e.g. 500 to 1000 times) can be achieved by representing the original image using these parameters. However, fractal image compression has its disadvantages. Because fractal image compression usually involves a large amount of matching and geometric operations, it is time consuming. The coding process is so asymmetrical that encoding of an image takes much longer time than decoding.

Transform Based Image Compression. The basic encoding method for transform based compression works as follows:

1. Image transform

Divide the source image into blocks and apply the transformations to the blocks.

2. Parameter quantization

The data generated by the transformation are quantized to reduce the amount of information. This step represents the information within the new domain by reducing the amount of data. Quantization is in most cases not a reversible operation because of its lossy property.

3. Encoding

Encode the results of the quantization. This last step can be error free by using Run Length encoding or Huffman coding. It can also be lossy if it optimizes the representation of the information to further reduce the bit rate.

Transform based compression is one of the most useful applications. Combined with other compression techniques, this technique allows the efficient transmission, storage, and display of images that otherwise would be impractical. [6]

DCT-Based Transform Coding. The Discrete Cosine Transform (DCT) [7] was first applied to image compression in the work by Ahmed, Natarajan, and Rao. It is a popular transform used by the JPEG (Joint Photographic Experts Group) image compression standard for lossy compression of images. Since it is used so frequently, DCT is often referred to in the literature as JPEG-DCT, DCT used in JPEG.

JPEG-DCT is a transform coding method comprising four steps. The source image is first partitioned into sub-blocks of size 8x8 pixels in dimension. Then each block is transformed from spatial domain to frequency domain using a 2-D DCT basis function. The resulting frequency coefficients are quantized and finally output to a lossless entropy coder. DCT is an efficient image compression method since it can decorrelate pixels in the image (since the cosine basis is orthogonal) and compact most image energy to a few transformed coefficients. Moreover, DCT coefficients can be lossily quantized according to some human visual characteristics. Therefore, the JPEG image file format is very efficient. This makes it very popular, especially in the World Wide Web. However, JPEG may be replaced by wavelet-based image compression algorithms, which have better compression performance.

MPEG Video Standard. MPEG (Motion Picture Expert Group) was set up in 1988 to develop a set of standard algorithms for applications that require storage of video and audio on digital storage media. The basic structure of compression algorithm proposed by MPEG is simple.

An input image is divided into blocks of 8 X 8 pixels. For a given 8 X 8 block, we subtract the prediction generated using the previous frame. The difference between the block being encoded and the prediction is transformed using a DCT. The transform coefficients are quantized and transmitted to the receiver. [3]

Wavelet Transform

Wavelets are functions defined over a finite interval. The basic idea of the wavelet transform is to represent an arbitrary function $f(x)$ as a linear combination of a set of such wavelets or basis functions. These basis functions are obtained from a single prototype wavelet called the mother wavelet by dilations (scaling) and translations (shifts).

The purpose of wavelet transform is to change the data from time-space domain to time-frequency domain which makes better compression results.

The simplest form of wavelets, the Haar wavelet function (see Figure 4) is defined as:

$$\psi(x) = \begin{cases} 1 & 0 \leq x < 1/2 \\ -1 & 1/2 \leq x < 1 \\ 0 & \text{otherwise} \end{cases}$$

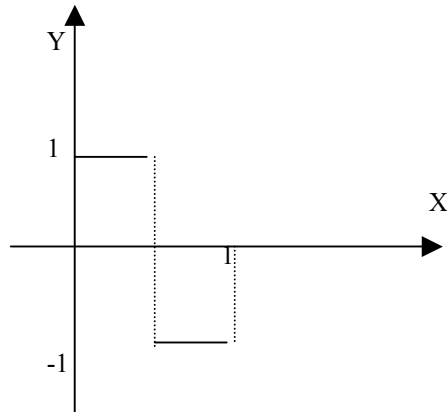


Figure 4. Haar Wavelet

The following is a simple example to show how to perform Haar wavelet transform on four sample numbers. Assume we have four numbers

$$x(0) = 1.2 \quad x(1) = 1.0 \quad x(2) = -1.0 \quad x(3) = -1.2.$$

Let us perform Haar wavelet transform on these four numbers.

$$\begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ y(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} = \begin{bmatrix} 2.2 \\ 0.2 \\ -2.2 \\ 0.2 \end{bmatrix}$$

Notice we can always do inverse transform from x to y :

$$\begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ y(3) \end{bmatrix}$$

If 0.2 is below our quantization threshold, it will be replaced by 0.

Then, reconstructed x will be [1.1, 1.1, -1.1, -1.1].

After first transform, we keep $y(1)$ and $y(3)$ at the finest level and iterate the transform on $y(0)$ and $y(2)$ again.

$$z(0) = y(0) + y(2) = 0 \text{ and } z(2) = y(0) - y(2) = 4.4.$$

Those four numbers become [0, 0.2, 4.4, 0.2]. After quantization, they could be [0, 0, 4, 0], which are much easier to be compressed.

We will discuss more detail on wavelet theory in Chapter 3.

Reason to Use Wavelet Based Compression

As discussed earlier, for image compression, loss of some information is acceptable. Among all of the above lossy compression methods, vector quantization requires many computational resources for large vectors; fractal compression is time consuming for coding; predictive coding has inferior compression ratio and worse reconstructed image quality than those of transform based coding. So, transform based compression methods are generally best for image compression.

For transform based compression, JPEG compression schemes based on DCT (Discrete Cosine Transform) have some advantages such as simplicity, satisfactory performance, and availability of special purpose hardware for implementation. However, because the input image is blocked, correlation across the block boundaries cannot be eliminated. This results in noticeable and annoying “blocking artifacts” particularly at low bit rates as shown in figure 5. [13]



Figure 5. (a) Original Lena Image (b) Reconstructed image to show blocking artifacts

Over the past ten years, the wavelet transform has been widely used in signal processing research, particularly, in image compression. In many applications, wavelet-based schemes achieve better performance than other coding schemes like the one based on DCT. Since there is no need to block the input image and its basis functions have variable length, wavelet based coding schemes can avoid blocking artifacts. Wavelet based coding also facilitates progressive transmission of images. [13]

Summary

In this chapter, we discussed some commonly used compression algorithms including both lossless and lossy methods. We also briefly introduced DCT-based compression methods, such as JPEG and MPEG, and basic concept of wavelet transform. Among the different lossy compression methods, wavelet based coding outperforms the others in image compression. In next chapter, we will discuss more detail on wavelet theory.

CHAPTER 3

WAVELET THEORY

In this chapter, we cover the mathematical properties of wavelets. Several types of wavelets are discussed, including Haar, Daubechies, and biorthogonal spline wavelets. For a better understanding the need of wavelet transform, we also briefly discuss the Fourier transform, which is most popularly used for signal processing, especially in electrical engineering.

Fourier Transform

In 19th century, the French mathematician, J. Fourier, showed that any periodic function can be expressed as an infinite sum of periodic complex exponential functions. Many years after this remarkable property of periodic functions was discovered, the ideas were generalized to non-periodic functions, and then to periodic or non-periodic discrete time signals. After this, Fourier transform became a very famous tool for computer calculations.

The equation

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

is generally called the Fourier Transform. The equation

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega)e^{j\omega t} d\omega$$

is called the inverse Fourier Transform.

Note that in the Fourier Transform equation, the integration is from minus infinity to plus infinity over time. So, no matter when the component with frequency ω appears in time, it will affect the result of the integration equally as well. The lack of time information is one serious weakness of Fourier Transform. That is why Fourier transform is not suitable if the signal has time varying frequency, i.e., the signal is non-stationary.

Windowed Fourier Transform

To solve the above problem, the Windowed Fourier Transform is used. The basic idea is to divide the signal into small enough segments, where these segments can be assumed to be stationary. The width of this window must be equal to the segment of the signal where this assumption is valid.

The Windowed Fourier Transform has several problems. If we use a window of infinite length, we get the Fourier Transform, which gives perfect frequency resolution, but no time information. On the other hand, in order to obtain the a stationary sample, we must have a small enough window in which the signal is stationary. The narrower

we make the window, the better the time resolution, and better the assumption of stationarity, but poorer the frequency resolution. However, the Wavelet transform solves the dilemma of resolution to a certain extent, as we will see in the next part.

Wavelet Transform

The fundamental idea behind wavelets is to analyze the signal at different scales or resolutions, which is called multiresolution. Wavelets are a class of functions used to localize a given signal in both space and scaling domains. A family of wavelets can be constructed from a mother wavelet. Compared to Windowed Fourier analysis, a mother wavelet is stretched or compressed to change the size of the window. In this way, big wavelets give an approximate image of the signal, while smaller and smaller wavelets zoom in on details. Therefore, wavelets automatically adapt to both the high-frequency and the low-frequency components of a signal by different sizes of windows. Any small change in the wavelet representation produces a correspondingly small change in the original signal, which means local mistakes will not influence the entire transform. The wavelet transform is suited for non-stationary signals, such as very brief signals and signals with interesting components at different scales. [15]

Wavelets are functions generated from one single function ψ , which is called mother wavelet, by dilations and translations

$$\psi_{a,b}(x) = |a|^{-1/2} \psi\left(\frac{x-b}{a}\right)$$

where ψ must satisfy $\int \psi(x)dx = 0$.

The basic idea of wavelet transform is to represent any arbitrary function f as a decomposition of the wavelet basis or write f as an integral over a and b of $\psi_{a,b}$.

Let $a = a_0^m, b = nb_0 a_0^m$, with $m, n \in$ integers, and $a_0 > 1, b_0 > 0$ fixed. Then the wavelet decomposition is

$$f = \sum c_{m,n}(f) \psi_{m,n}$$

Let $a_0 = 2, b_0 = 1$, we have an orthonormal basis, so that

$$c_{m,n}(f) = \langle \psi_{m,n}, f \rangle = \int \psi_{m,n}(x) f(x) dx$$

In image compression, we are dealing with sampled data that are discrete in time. We would like to have discrete representation of time and frequency, which is called the discrete wavelet transform (DWT). Before we start the DWT, we need to study another concept, multiresolution analysis.

Multiresolution Analysis and the Scaling Function

Let us define a function $\phi(x)$ that we call a scaling function. A very important property of the scaling function is it can be represented by the dilations of itself.

$$\phi(t) = \sum_{k=0}^N h_k \sqrt{2} \phi(2t - k) \text{ is called the multiresolution analysis (MRA) equation.}$$

We define $\phi_k(t) = \phi(t - k)$. The set of all functions that can be obtained by linear combination of the set $\{\phi_k(t)\}$ is called the span of the set $\{\phi_k(t)\}$ or $\text{Span}\{\phi_k(t)\}$. The closure of $\text{Span}\{\phi_k(t)\}$ denoted by $\overline{\text{Span}\{\phi_k(t)\}}$ is obtained by adding all functions that are limits of sequences of functions in $\text{Span}\{\phi_k(t)\}$. Let us call the closure set V_0 . To generate a function at a higher resolution, we can use dilations of the mother scaling function. Define $\phi_{j,k}(t) = 2^{j/2} \phi(2^j t - k)$, with the first index referring the resolution while the second one denotes the translation.

Notice that any function that can be represented by the translates of $\phi(t)$ can also be represented by a linear combination of translates of $\phi_{1,0}(t)$. The converse is not true.

Defining $V_1 = \overline{\text{Span}\{\phi_{1,k}(t)\}}$, we have $V_0 \subset V_1$. Similarly, we can get the nested spaces

$$V_0 \subset V_1 \subset V_2 \cdots \subset V_n.$$

We denote the set of functions that can be obtained by a linear combination of the translates of the mother wavelet as W_0 . It can be shown that a function in V_1 can be decomposed into a function in V_0 and a function in W_0 or

$$V_1 = V_0 \oplus W_0.$$

That means that any functions in V_1 can be represented by functions in V_0 and W_0 .

For a chosen scaling function, the mother wavelet $\psi(t)$ can be written as

$$\psi(t) = \sum_k w_k \phi_{1,k}(t)$$

or

$$\psi(t) = \sum_k w_k \sqrt{2} \phi(2t - k)$$

Similarly, for any functions that can be represented at resolution $j + 1$, we define W_j as the closure of the span of $\phi_{j,k}(t)$. We can show that

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

Therefore, we obtain

$$V_{j+1} = V_0 \oplus W_j \oplus W_{j-1} \oplus \dots \oplus W_0.$$

Discrete Wavelet Transform Implementation Using Filters

We can write $c_{m,n}(f) = \langle \psi_{m,n}, f \rangle = \int \psi_{m,n}(x) f(x) dx$ as the following algorithm.

$$\begin{aligned} c_{m,n}(f) &= \sum_k g_{2n-k} a_{m-1,k}(f) \\ a_{m,n}(f) &= \sum_k h_{2n-k} a_{m-1,k}(f) \end{aligned} \quad (\text{Equation 1})$$

where $g_l = (-1)^l h_{-l+1}$ and $h_n = 2^{1/2} \int \varphi(x-n)\varphi(2x)dx$. When f is given by sampled form, then we can assume those samples as the highest order resolution approximation coefficient $a_{0,n}$. Equation 1 gives the coding algorithm on these sampled values with low-pass filter h and high-pass filter g . For orthonormal wavelet bases, these filters give exact reconstruction by the following equation:

$$a_{m-1,n}(f) = \sum_n [h_{2n-l} a_{m,n}(f) + g_{2n-l} c_{m,n}(f)].$$

Compact Supported Orthogonal Wavelets

In this section, we will determine the filter coefficients for compact supported orthogonal wavelets.

We begin with the scaling function

$$\phi(t) = \sum_k h_k \sqrt{2} \phi(2t - k).$$

By integrating both sides, we have

$$\int_{-\infty}^{\infty} \phi(t) dt = \int_{-\infty}^{\infty} \sum_k h_k \sqrt{2} \phi(2t - k) dt.$$

Substituting $x = 2k - t$ and $dx = 2dt$ on the right hand side, we get

$$\int_{-\infty}^{\infty} \phi(t) dt = \sum_k h_k \sqrt{2} \int_{-\infty}^{\infty} \phi(x) \frac{1}{2} dx = \sum_k h_k \frac{1}{\sqrt{2}} \int_{-\infty}^{\infty} \phi(x) dx .$$

Then, divide both sides by the integral and we have

$$\sum_k h_k = \sqrt{2} . \quad (\text{Equation 2})$$

We use the orthogonality condition on the scaling function to get another condition on $\{h_k\}$:

$$\begin{aligned} \int |\phi(t)|^2 dt &= \int \sum_k h_k \sqrt{2} \phi(2t - k) \sum_m h_m \sqrt{2} \phi(2t - m) dt \\ &= \sum_k \sum_m h_k h_m 2 \int \phi(2t - k) \phi(2t - m) dt \\ &= \sum_k \sum_m h_k h_m \int \phi(x - k) \phi(x - m) dx \end{aligned}$$

where in the last equation, we substitute $2t$ by x .

The integral on the right hand side is zero except when $k = m$. When $k = m$, we obtain

$$\sum_k h_k^2 = 1. \quad (\text{Equation 3})$$

By using the orthogonality of the translates of scaling function we have

$$\int \phi(t) \phi(t - m) dt = \delta_m .$$

Substituting $\phi(t)$ by scaling function, we get

$$\begin{aligned} &\int \left[\sum_k h_k \sqrt{2} \phi(2t - k) \right] \left[\sum_l h_l \sqrt{2} \phi(2t - 2m - l) \right] dt \\ &= \sum_k \sum_l h_k h_l 2 \int \phi(2t - k) \phi(2t - 2m - l) dt. \end{aligned}$$

By substituting $x = 2t$, we get

$$\begin{aligned} \int \phi(t) \phi(t - m) dt &= \sum_k \sum_l h_k h_l \int \phi(x - k) \phi(x - 2m - l) dx \\ &= \sum_k \sum_l h_k h_l \delta_{k-(2m+l)} = \sum_k h_k h_{k-2m} . \end{aligned}$$

Therefore, we have equation

$$\sum_k h_k h_{k-2m} = \delta_m \quad (\text{Equation 4})$$

By using equations 2, 3 and 4, we can generate filter coefficients for scaling function.

For $k = 2$, from Equation 2 and 3, we have

$$h_0 + h_1 = \sqrt{2}$$

$$h_0^2 + h_1^2 = 1.$$

The only solution is $h_0 = h_1 = \frac{1}{\sqrt{2}}$, which is the Harr scaling function.

For $k = 4$, from Equation 2, 3, and 4, we have

$$h_0 + h_1 + h_2 = \sqrt{2}$$

$$h_0^2 + h_1^2 + h_2^2 = 1$$

$$h_0 h_2 + h_1 h_3 = 0.$$

The solutions to these equations include the 4-tap (filter length is 4) Daubechies scaling function.

$$h_0 = \frac{1 + \sqrt{3}}{4\sqrt{2}}, h_1 = \frac{3 + \sqrt{3}}{4\sqrt{2}}, h_2 = \frac{3 - \sqrt{3}}{4\sqrt{2}}, h_3 = \frac{1 - \sqrt{3}}{4\sqrt{2}}.$$

We know the wavelet function is

$$\psi(t) = \sum_k w_k \sqrt{2} \phi(2t - k).$$

If the wavelet is orthogonal to the scaling function at the same scale, we have

$$\int \phi(t - k) \psi(t - m) dt = 0.$$

Then, we can obtain the wavelet filter coefficients from the scaling filter coefficients [5].

$$w_k = \pm(-1)^k h_{N-k}.$$

Haar Wavelet. The Haar scaling function is defined as:

$$\phi(x) = \begin{cases} 1 & 0 \leq x < 1 \\ 0 & \text{otherwise} \end{cases}$$

The scaling function is plotted in Figure 6.

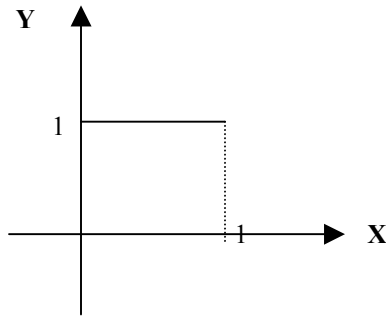


Figure 6. Haar Scaling Function

In the scaling equation $\varphi(x) = \sum_{k=0}^N C_k \varphi(2x - k)$, only $c_0 = c_1 = 1$, all the other coefficients are zeros.

Haar wavelets are defined as:

$$\psi_{a,b}(x) = 2^{a/2} \psi(2^a x - b), \quad b = 0, 1, \dots, 2^a - 1,$$

$$\text{where } \psi(x) = \begin{cases} 1 & 0 \leq x < 1/2 \\ -1 & 1/2 \leq x < 1 \\ 0 & \text{otherwise} \end{cases} \text{ is the Haar mother wavelet.}$$

The Haar mother wavelet function is plotted in Figure 7.

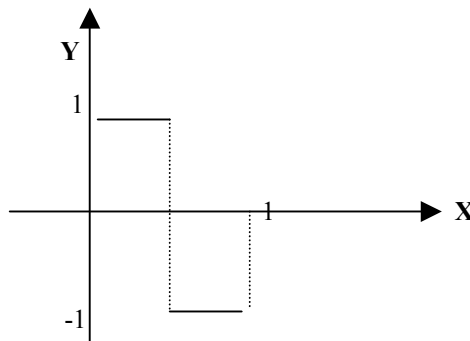


Figure 7. Haar Mother Wavelet

Daubechies Orthogonal Wavelets. There are no explicit expressions for Daubechies compact support orthogonal wavelets and corresponding scaling functions.

Table 1, 2, and 3 present the wavelet filter coefficients for Daubechies 4, 6, and 10 taps wavelets.

Figure 8, 9, and 10 are their scaling and wavelet functions respectively.

Note that the longer the filter, the smoother the scaling and wavelet functions.

Table 1. Coefficients for the 4-tap Daubechies Low-pass Filter [9]

H_0	.4829629131445341
H_1	.8365163037378077
H_2	.2241438680420134
H_3	-.1294095225512603

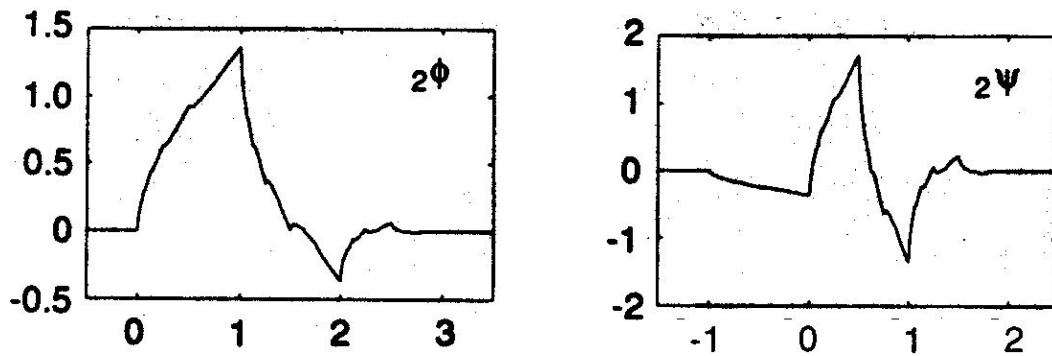


Figure 8. Daubechies 4-tap Scaling and Wavelet Functions

Table 2. Coefficients for the 6-tap Daubechies Low-pass Filter

H_0	.3326705529500825
H_1	.8068915093110924
H_2	.4598775021184914
H_3	-.1350110200102546
H_4	-.0854412738820267
H_5	.0352262918857095

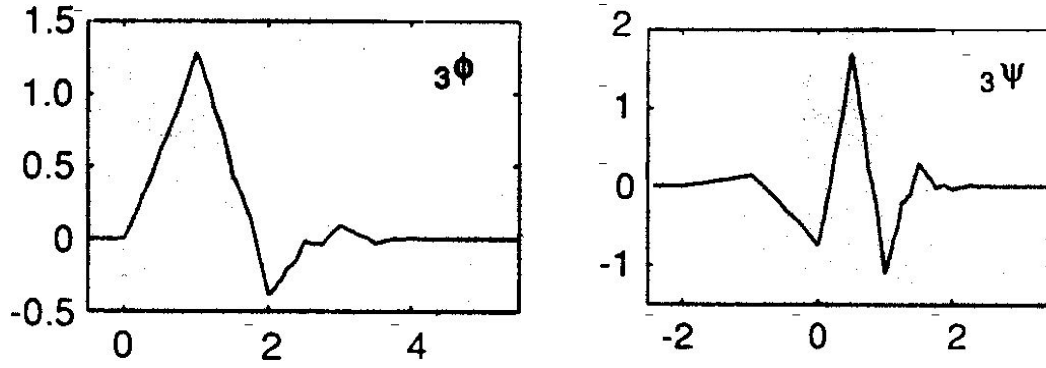


Figure 9. Daubechies 6-tap Scaling and Wavelet Functions

Table 3. Coefficients for the 10-tap Daubechies Low-pass Filter

H_0	.1601023979741929
H_1	.6038292697971895
H_2	.7243085284377726
H_3	.1384281459013203
H_4	-.2422948870663823
H_5	-.0322448695846381
H_6	.0775714938400459
H_7	-.0062414902127983
H_8	-.0125807519990820
H_9	-.0033357252854738

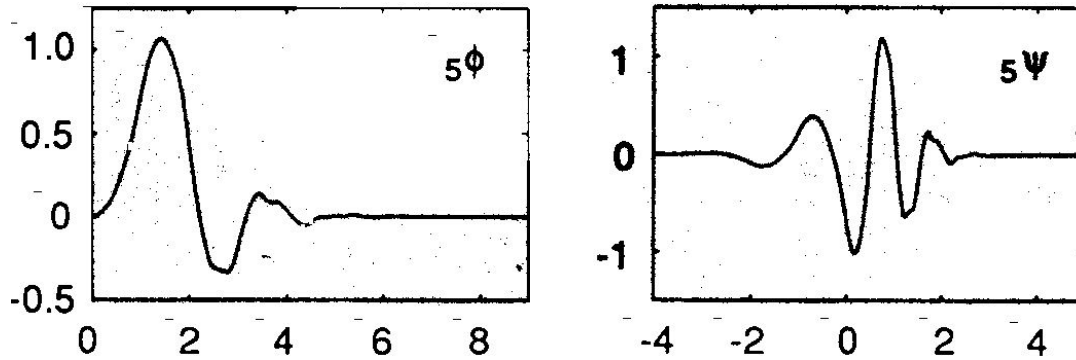


Figure 10. Daubechies 10-tap Scaling and Wavelet Functions

Biorthogonal and Spline Wavelets.

As we know, most images are smooth. It is reasonable to use smooth mother wavelet for image analysis. On the other hand, it is also desirable that the mother wavelet is symmetric so that the corresponding wavelet transform can be implemented using mirror boundary conditions that reduces boundary artifacts. Unfortunately, except for the Harr wavelet (trivial example), no wavelets are both orthogonal and symmetric.

To achieve the symmetric property, we can relax the orthogonality requirement by using a biorthogonal basis. In this case, we keep the same decomposition as in Equation 1. The reconstruction becomes

$$a_{m-1,f}(f) = \sum_n [\tilde{h}_{2n-l} a_{m,n}(f) + \tilde{g}_{2n-l} c_{m,n}(f)]$$

where \tilde{h} and \tilde{g} may be different from h, g to obtain exact reconstruction. The relations between them are given by the following equations.

$$\tilde{g}_n = (-1)^n h_{-n+1}, \quad g_n = (-1)^n \tilde{h}_{-n+1}, \quad \text{and} \quad \sum_n h_n \tilde{h}_{n+2k} = \delta_{k,0}. \quad (\text{Equation 5})$$

Define

$$\phi(x) = \sum_n h_n \phi(2x - n) \quad \text{and} \quad \tilde{\phi}(x) = \sum_k \tilde{h}_k \tilde{\phi}(2x - n).$$

Also define

$$\psi(x) = \sum_n g_n \phi(2x - n) \quad \text{and} \quad \tilde{\psi}(x) = \sum_k \tilde{g}_k \tilde{\phi}(2x - n).$$

Then we can rewrite $a_{m,n}(f)$ and $c_{m,n}(f)$ as:

$$a_{m,n}(f) = \langle \phi_{m,n}, f \rangle = 2^{-m/2} \int \phi_{m,n}(x) f(x) dx$$

$$c_{m,n}(f) = \langle \psi_{m,n}, f \rangle = 2^{-m/2} \int \psi_{m,n}(x) f(x) dx$$

and reconstruction is $f = \sum_{m,n} \langle \psi_{m,n}, f \rangle \tilde{\psi}_{m,n}$.

The following figure shows the relation between the filter structure and wavelets functions.

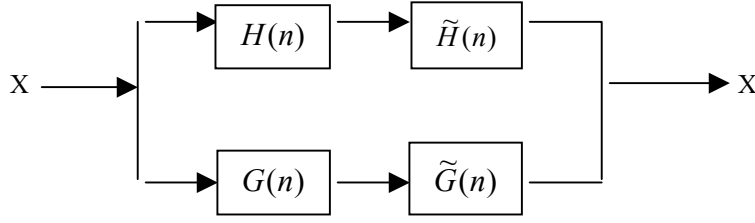


Figure 11. Filter Structure and The Associating Wavelets

For symmetric filters, the exact reconstruction condition on h and \tilde{h} given in Equation 5 can be represented by

$$H(\xi)\tilde{H}(\xi) + H(\xi + \pi)\tilde{H}(\xi + \pi) = 1$$

where $\tilde{H}(\xi) = 2^{-1/2} \sum_n \tilde{h}_n e^{-jn\xi}$ and $H(\xi) = 2^{-1/2} \sum_n h_n e^{-jn\xi}$.

Together with the divisibility of H and \tilde{H} , respectively, by $(1 + e^{-j\xi})^k$ and $(1 + e^{-j\xi})^{\tilde{k}}$, we have [12]

$$H(\xi)\tilde{H}(\xi) = \cos(\xi/2)^{2l} \left[\sum_{p=0}^{l-1} \binom{l-1+p}{p} \sin(\xi/2)^{2p} + \sin(\xi/2)^{2l} R(\xi) \right] \quad (\text{Equation 6})$$

where $R(\xi)$ is an odd polynomial in $\cos(\xi)$ and $2l = k + \tilde{k}$.

Spline Filters. Let us choose $R \equiv 0$ with $\tilde{H}(\xi) = \cos(\xi/2)^k e^{-jk\xi/2}$ where $k = 0$ if \tilde{k} is even, $k = 1$ if \tilde{k} is odd. We have

$$H(\xi) = \cos(\xi/2)^{2l-\tilde{k}} e^{jk\xi/2} \left[\sum_{p=0}^{l-1} \binom{l-1+p}{p} \sin(\xi/2)^{2p} \right]$$

Then, we get the corresponding function $\tilde{\phi}$ which is a B-spline function.

Note that a B-spline of degree n is defined as:

$$\phi^n(x) = \underbrace{\phi^0(x) * \dots * \phi^0(x)}_{(n+1) \text{ times}}$$

where $\phi^0(x) = \begin{cases} 1 & 0 \leq x < 1 \\ 0 & \text{otherwise.} \end{cases}$

We present two examples from this family. They correspond to $k = 2, \tilde{k} = 2$ and $k = 4, \tilde{k} = 2$. Table 4 and 5 list the their coefficients H_n and \tilde{H}_n . The corresponding scaling and wavelet functions are plotted in Figure 9 and 10 respectively.

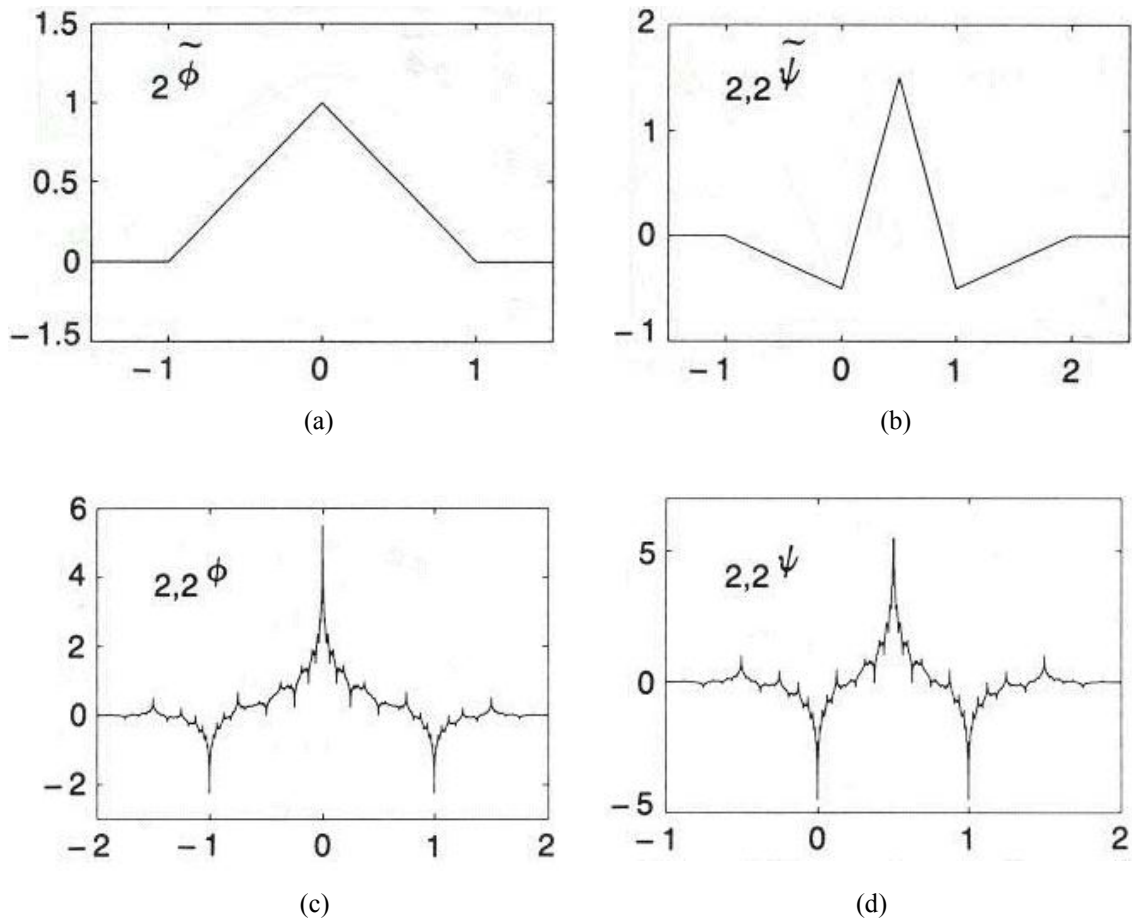
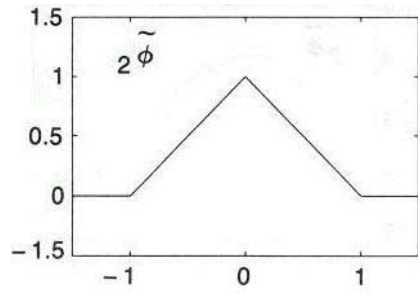


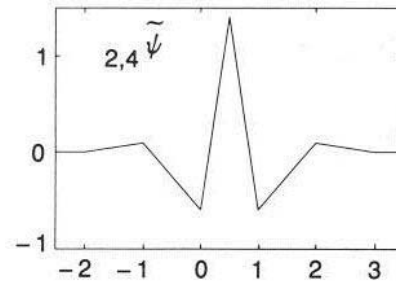
Figure 12. Spline Examples with $k = 2, \tilde{k} = 2$ (a) Scaling Function $\tilde{\phi}$
 (b) Wavelet Function $\tilde{\psi}$ (c) Scaling Function ϕ (d) Wavelet Function ψ

Table 4. Filter Coefficients with $k = 2, \tilde{k} = 2$

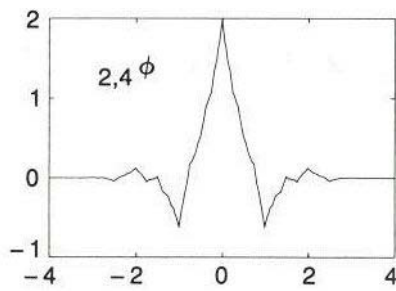
n	0	± 1	± 2
H_n	-1/8	1/4	3/4
\tilde{H}_n	1/2	1/4	0



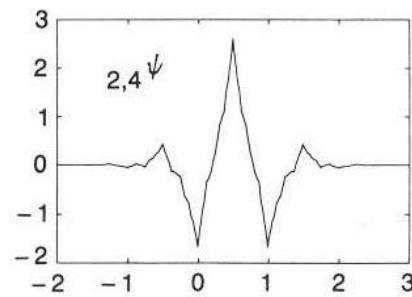
(a)



(b)



(c)



(d)

Figure 13. Spline Examples with $k = 4, \tilde{k} = 2$. (a) Scaling Function $\tilde{\phi}$
 (b) Wavelet Function $\tilde{\psi}$ (c) Scaling Function $\tilde{\phi}$ (d) Wavelet Function ψ

Table 5. Filter Coefficients with $k = 4, \tilde{k} = 2$

n	0	± 1	± 2	± 3	± 4
H_n	45/64	19/64	-1/8	-3/64	3/128
\tilde{H}_n	1/2	1/4	0	0	0

A Spline Variant with Less Dissimilar Lengths. We choose $R \equiv 0$, and factor the right hand side of Equation 6 to break the polynomial into a product of two polynomials in $\sin(\xi/2)$. Allocate the polynomials to H and \tilde{H} respectively to make the length of h and \tilde{h} as close as possible.

The following example is the shortest one in this family (shortest h and \tilde{h}). It corresponds to $l = k = 4$.

Table 6. Filter Coefficients with $l = k = 4$

n	0	± 1	± 2	± 3	± 4
H_n	0.602949	0.266864	-0.078223	-0.016864	0.026749
\tilde{H}_n	0.557543	0.295636	-0.028772	-0.045636	0

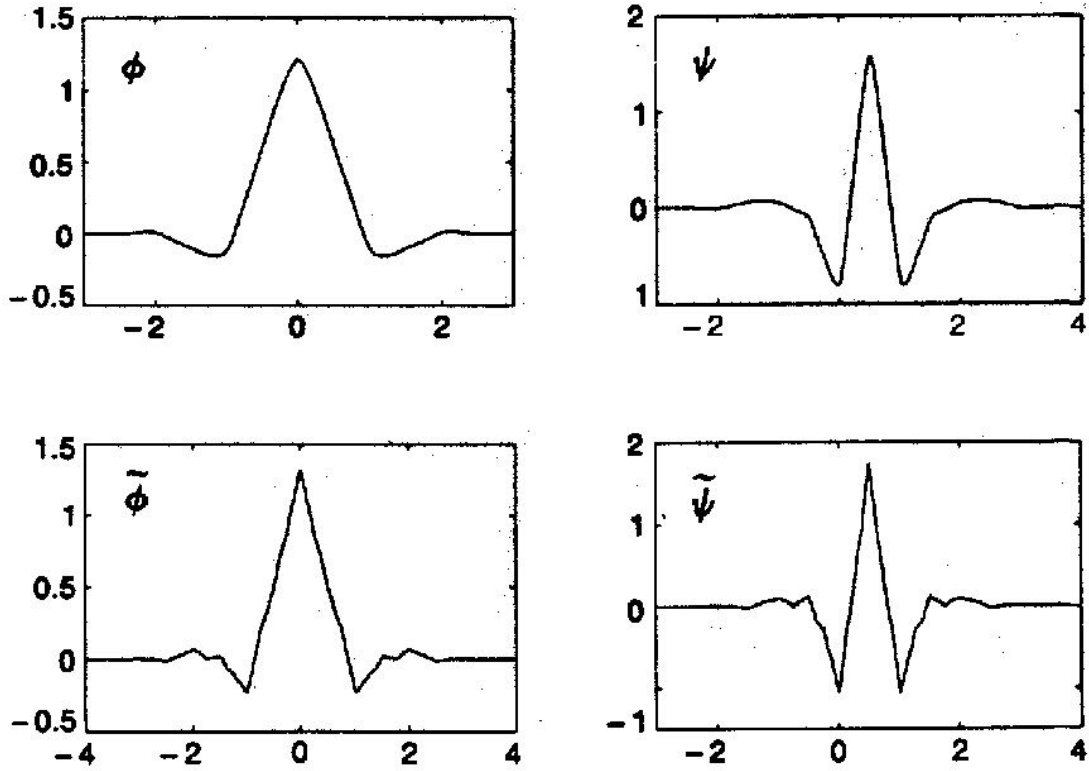


Figure 14. Spline Examples with $l = k = \tilde{k} = 4$

Comparison of Wavelet Properties

The following Table 7 shows the property comparison of three kinds of wavelets.

Table7. Property Comparison of Three Kinds of Wavelets

Property	Haar	Daubechie	Biorthogonal Spline
Explicit Function	Yes	No	Yes
Orthogonal	Yes	Yes	No
Symmetric	Yes	No	Yes
Continuous	No	Yes	Yes
Compacted support	Yes	Yes	Yes
Maximum regularity for order L	No	No	Yes
Shortest scaling function for order L	Yes	No	Yes

Haar and Daubechie's wavelets have orthogonality, which has some nice features.

- 1) The scaling function and wavelet function are the same for both forward and inverse transform.
- 2) The correlations in the signal between different subspaces are removed.

Among the three kinds of wavelets, the Haar wavelet transform is the simplest one to implement, and it is the fastest. The major disadvantage of the Haar wavelet is its discontinuity, which makes it difficult to simulate a continuous signal.

Daubechie found the first continuous orthogonal compact support wavelet. Note that this family of wavelets is not symmetric. The advantage of symmetry is that the corresponding wavelet transform can be implemented using mirror boundary conditions that reduces boundary artifacts. That is why we introduce the biorthogonal spline wavelet.

For the biorthogonal spline wavelet, the scaling function is a B-spline. The B-spline of degree N is the shortest possible scaling function of order $N-1$ and B-splines are the smoothest scaling functions for a filter of a given length [2]. Because splines are piecewise polynomial, they are easy to manipulate. For example, it is simple to get spline derivatives and integrals.

Summary

To understand the need for wavelet transforms, in this chapter, we first introduced the most popularly used Fourier Transform. Then, we covered the mathematical properties of several types of wavelets including Haar, Daubechies, and biorthogonal spline wavelets. In the next chapter, we will discuss how to apply the wavelet transform to image compression.

CHAPTER 4
WAVELET APPLIED IN IMAGE COMPRESSION

In order to compare wavelet methods, a software tool coded in C++ called MinImage was used. MinImage was originally created to test one type of wavelet [6]. Additional functionality was added to MinImage to support other wavelet types, and the EZW coding algorithm was implemented to achieve better compression results. In this chapter, we discuss some implementation details of MinImage.

Baseline Schema

The wavelet image compressor, MinImage, is designed for compressing either 24-bit true color or 8-bit gray scale digital images. It was originally created to test Haar wavelet using subband coding. To compare different wavelet types, other wavelet types, including Daubechies and biorthogonal spline wavelets were implemented. Also, the original subband coding were changed to EZW coding to obtain better compression results.

A very useful property of MinImage is that different degrees of compression and quality of the image can be obtained by adjusting the compression parameters through the interface. The user can trade off between the compressed image file size and the image quality. The user can also apply different wavelets to different kind of images to achieve the best compression results.

Figure 15 is the baseline structure of MinImage compressing sechema.

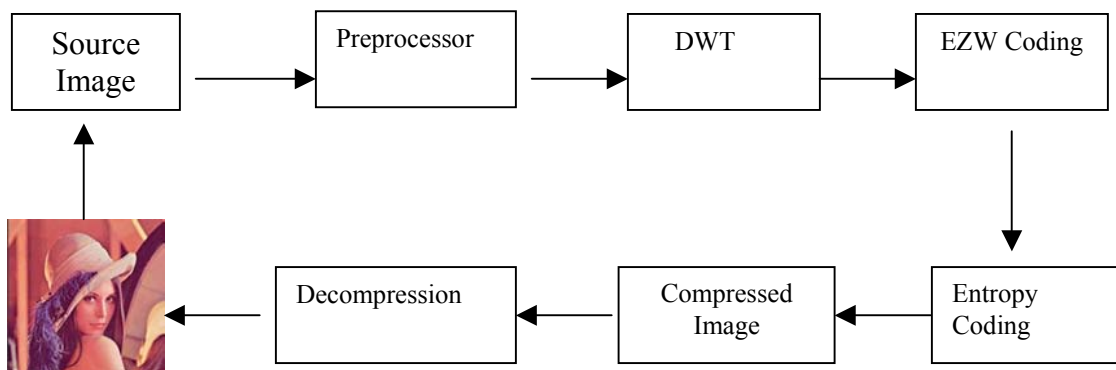


Figure 15. The Baseline Schema of MinImage

For more details on the Preprocessor, see [6]. In the rest of the chapter, we will focus on implementation of discrete wavelet transform (DWT) and EZW coding.

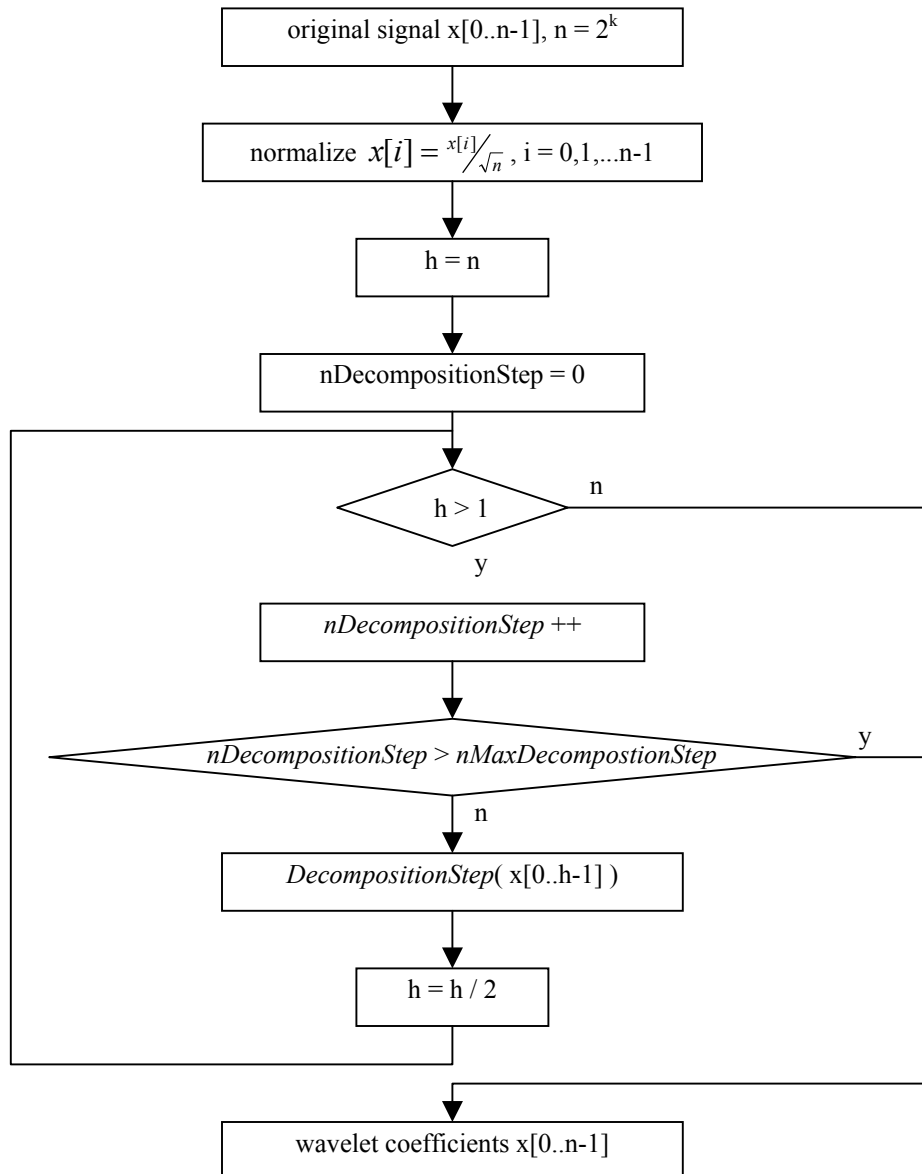


Figure 16. The 1-D Wavelet Decomposition Algorithm in MinImage [6]

Note: The $nMaxDecompositionStep$ is defined to control when the wavelet decomposition should be stopped. For a complete wavelet decomposition of the signal X^n with 2^n entries, the $nMaxDecompositionStep$ can be defined in the domain of $[1, n]$. $DecompositionStep$ implements one step of the wavelet decomposition in an array of data. [6]

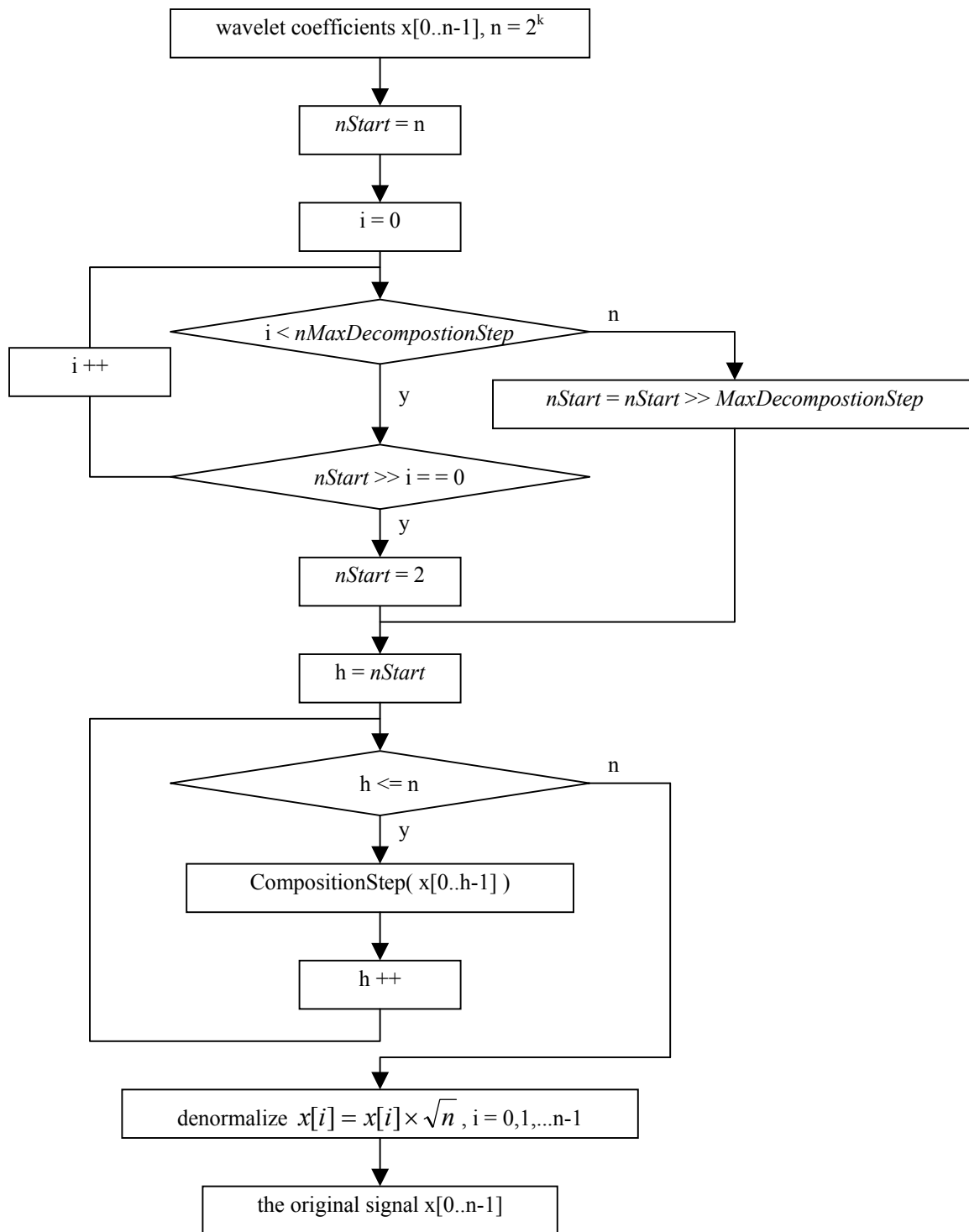


Figure 17. The 1-D Wavelet Composition Algorithm in MinImage [6]

As we know, images are 2-D signals. A simple way to perform wavelet decomposition on an image is to alternate between operations on the rows and columns. First, wavelet decomposition is performed on the pixel values in each row of the image. Then, wavelet decomposition is performed to each column of the previous result. The process is repeated to perform the complete wavelet decomposition.

The following figure shows the 2-D wavelet decomposition algorithm.

```

PROCEDURE 2D-Decomposition( ImageData[ 0..n-1, 0..n-1 ] )
  FOR row = 0 TO n-1 DO
    FOR col = 0 TO n-1 DO
      ImageData[ row, col ] /= n
      h = n
      WHILE h > 1 DO
        FOR row = 0 TO h-1 DO
          DecompositionStep( ImageData[ row, 0..h-1 ] )
        END FOR
        FOR col = 0 TO h-1 DO
          DecompositionStep( ImageData[ 0..h-1, col ] )
        END FOR
        h = h / 2
      END WHILE
    END FOR
  END FOR
END PROCEDURE

```

Figure 18. The 2D Wavelet Decomposition Algorithm [6]

Figure 19 illustrates each step of the 2-dimensional wavelet decomposition. The shaded parts are wavelet coefficients.

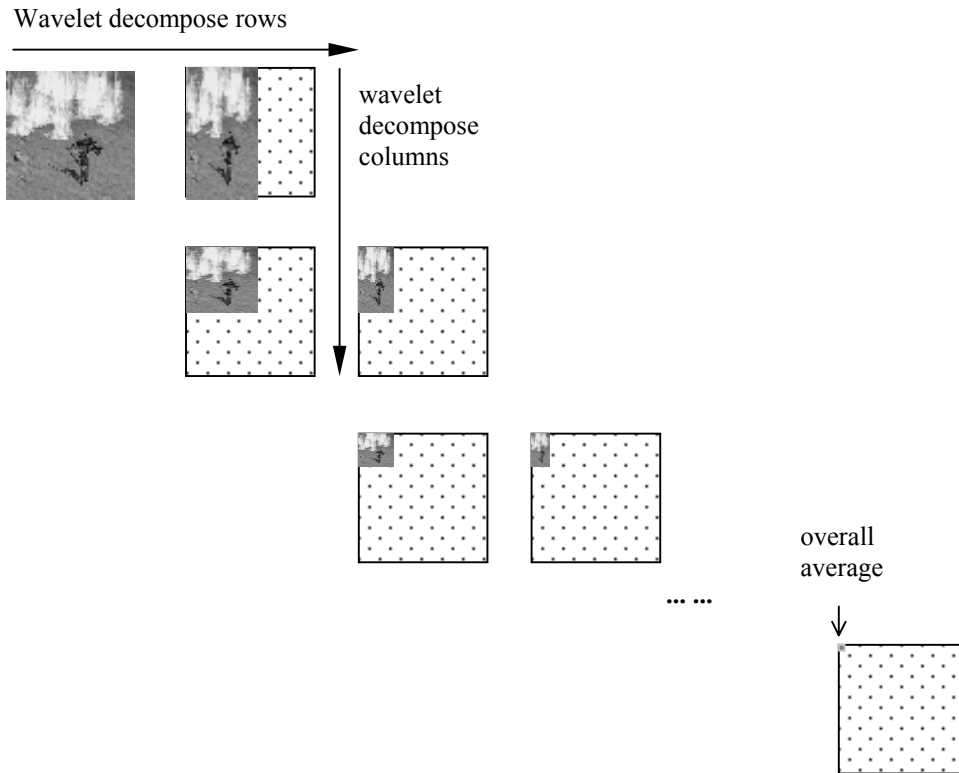


Figure 19. The 2D Wavelet Decomposition of An Image [6]

Embedded Zerotree Wavelet (EZW) Coding

After the 2-D wavelet decomposition, the wavelet transform blocks contain the wavelet coefficients. This section introduces the Embedded Zerotree Wavelet coding to code the transformed wavelet coefficients.

Subbands in the Wavelet Transform Blocks

For a 1-D wavelet transform, a vector of the wavelet coefficients can be divided into subbands after the wavelet decomposition as shown in the following figure:

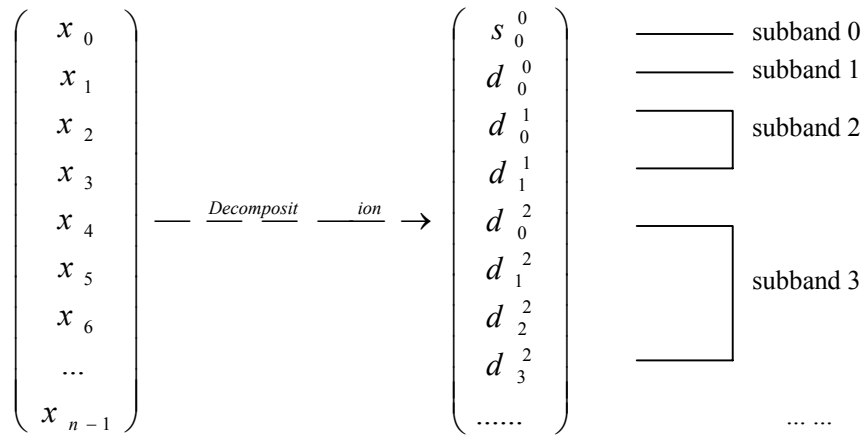


Figure 20. Subbands after the 1-D Wavelet Decomposition [6]

Similarly, a block of the two-dimensional wavelet coefficients can be divided into subbands as follows:

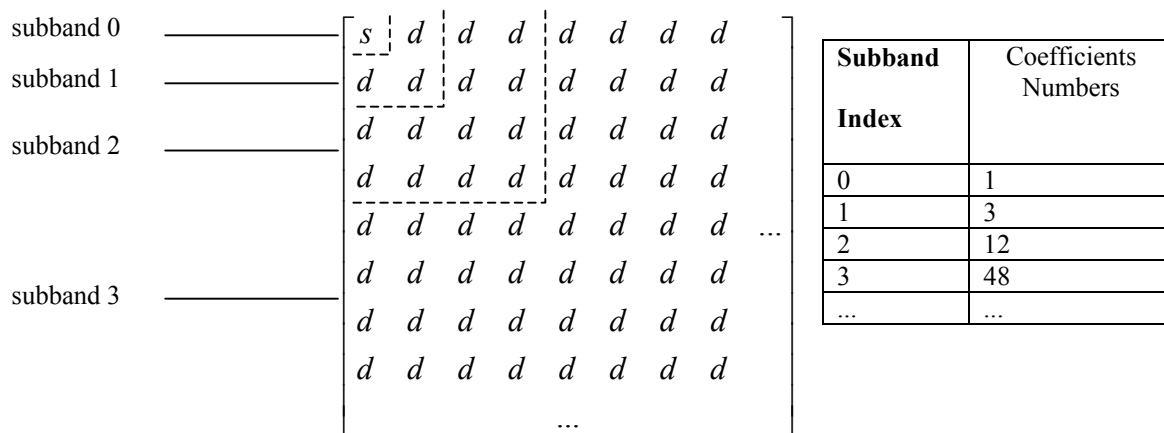


Figure 21. Subbands in A Wavelet Transform Block after the 2-D Wavelet [6]

EZW Coding

An EZW encoder was specially designed by Shapiro [1] to use with wavelet transforms. In fact, EZW coding is more like a quantization method. It 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, a property similar to *JPEG* encoded images. Progressive encoding is also known as embedded encoding, which explains the E in EZW.

Coding an image using the EZW scheme, together with some optimizations, results in a remarkably effective image compressor with the property that the compressed data stream can have *any* bit rate desired. Any bit rate is only possible if there is information loss somewhere, so that the compressor is lossy. However, lossless compression is also possible with an EZW encoder, but of course with less spectacular results.

The EZW encoder is based on two important observations:

1. Natural images in general have a low pass spectrum. When an image is wavelet transformed, the energy in the subbands decreases as the scale decreases (low scale means high resolution), so the wavelet coefficients will, on average, be smaller in the higher subbands than in the lower subbands. This shows that progressive encoding is a very natural choice for compressing wavelet transformed images, since the higher subbands only add detail.

2. Large wavelet coefficients are more important than small wavelet coefficients.

These two observations are used by encoding the wavelet coefficients in decreasing order, in several passes. For every pass, a threshold is chosen against which all the wavelet coefficients are measured. If a wavelet coefficient is larger than the threshold, it is encoded and removed from the image; if it is smaller it is left for the next pass. When all the wavelet coefficients have been visited, the threshold is lowered, and the image is scanned again to add more detail to the already encoded image. This process is repeated until all the wavelet coefficients have been encoded completely or another criterion has been satisfied (maximum bit rate for instance).

A wavelet transform transforms a signal from the time domain to the joint time-scale domain. This means that the wavelet coefficients are two-dimensional. If we want to compress the transformed signal, we have to code not only the coefficient values, but also their position in time. When the signal is an image, then the position in time is better expressed as the position in space. After wavelet transforming an image, we can represent it using trees because of the subsampling that is performed in the transform. A coefficient in a low subband can be thought of as having four descendants in the next higher subband (see figure 22). One of each four descendants also has four descendants in the next higher subband and we see a *quad-tree* emerge: every parent has four children. [14]

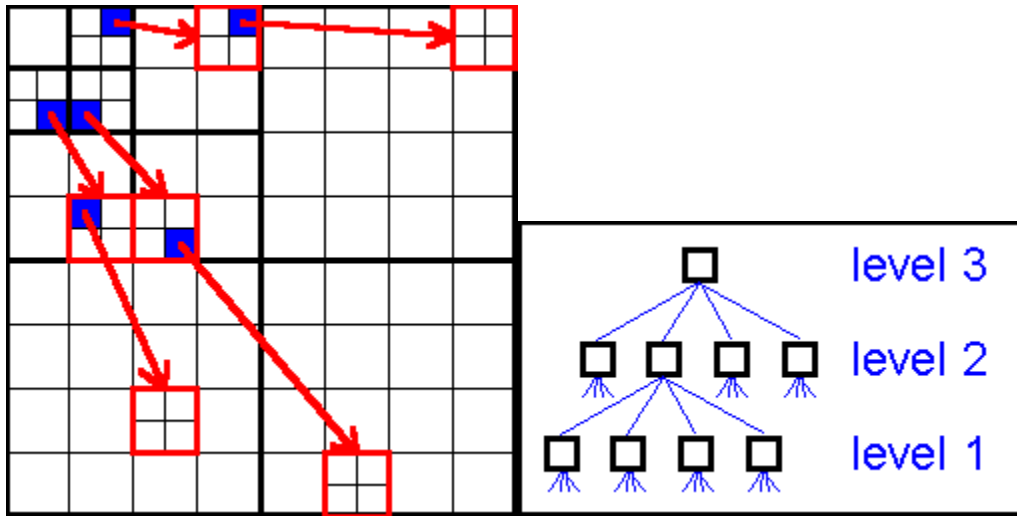


Figure 22.

The Relations between Wavelet Coefficients in Different Subbands as Quad-trees.

A zerotree is a quad-tree of which all nodes are equal to or smaller than the root. The tree is coded with a single symbol and reconstructed by the decoder as a quad-tree filled with zeroes. To clutter this definition we have to add that the root has to be smaller than the threshold against which the wavelet coefficients are currently being measured.

The EZW encoder exploits the zerotree based on the observation that wavelet coefficients decrease with scale. It assumes that there will be a very high probability that all the coefficients in a quad tree will be smaller than a certain threshold if the root is smaller than this threshold. If this is the case, then the whole tree can be coded with a single zerotree symbol. Now if the image is scanned in a predefined order, going from high scale to low, implicitly many positions are coded through the use of zerotree symbols. [14]

EZW Algorithm

The first step in the EZW coding algorithm is to determine the initial threshold. It can be calculated by the following equation:

$$t_0 = 2^{\lfloor \log_2(\text{MAX}(|C(x,y)|)) \rfloor}$$

where MAX(.) means the maximum coefficient value of the image and C(x,y) denotes the coefficient. We start the coding loop with this initial threshold.

```

MAIN PROCEDURE
threshold = initialThreshold;
DO
{
dominantPass();
subordinatePass();
threshold = threshold / 2;
}
WHILE (threshold > stopThreshold);
END MAIN PROCEDURE

```

Figure 23: Algorithm of the main loop

From the above algorithm, we see that two passes are used to code the image. In the first pass, the *dominant pass*, the image is scanned and a symbol is outputted for every coefficient. Notice two different scan orders (see figure 4.3) can be applied to this algorithm and Morton scan is implemented because it is simpler to code.

Only four symbols are used to code the coefficients.

- 1) If the coefficient is larger than the threshold a **P** (positive) is coded
- 2) Else if the coefficient is smaller than minus the threshold an **N** (negative) is coded.
- 3) Else if the coefficient is the root of a zerotree then a **T** (zerotree) is coded.
- 4) Else if the coefficient is smaller than the threshold but it is not the root of a zerotree, then a **Z** (isolated zero) is coded. This happens when there is a coefficient larger than the threshold in the subtree.

The effect of using the **N** and **P** codes is that when a coefficient is found to be larger than the threshold (in absolute value or magnitude), its two most significant bits are outputted.

Notice that in order to determine if a coefficient is the root of a zerotree or an isolated zero, we will have to scan the whole quad-tree. This process is time consuming. Also, to prevent outputting codes for coefficients in identified zerotrees, we will have to keep track of them. This means memory for book keeping.

Finally, all the coefficients that are in absolute value larger than the current threshold are extracted and placed without their sign on the subordinate list, and their positions in the image are filled with zeroes. This will prevent them from being coded again.

The second pass, the *subordinate pass*, is the refinement pass. We output the next most significant bit of all the coefficients on the subordinate list.

The main loop ends when the threshold reaches the stop threshold value.

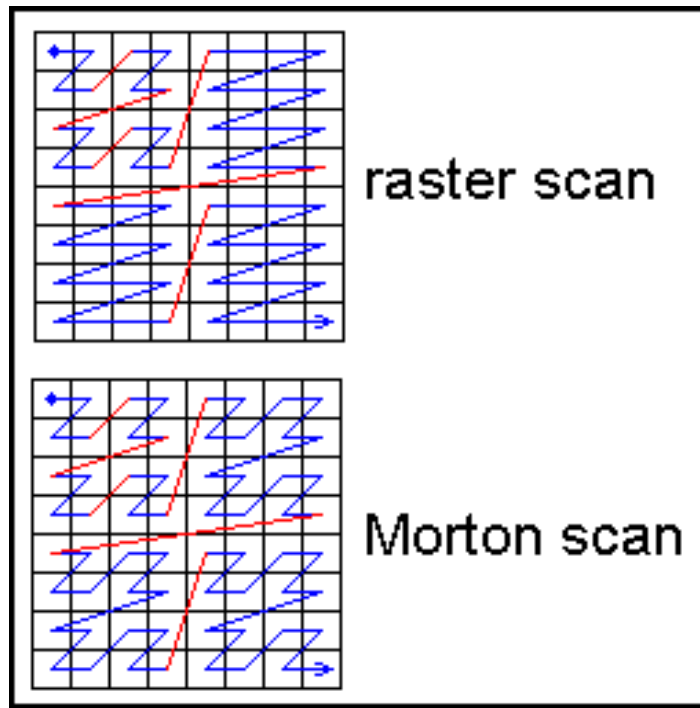


Figure 24. Two Different Scan Orders[14]

```

PROCEDURE Bool Zerotree(int threshold, Coefficient CurrentCoefficient)
BEGIN
    Put the current efficient into Children coefficient list;
    WHILE ( Children coefficient list is not empty)
    BEGIN
        Get a coefficient from the Children coefficient list;
        IF coefficient >= threshold, THEN
            RETURN FALSE
        ELSE
            Put the four children into the Children coefficient list;
        END IF
    END
    RETURN TR
END PROCEDURE

```

Figure 25. Algorithm to Determine If A Coefficient is A Root of Zerotree

```

PROCEDURE DominantPass()
BEGIN
    initialize coded coefficient list;
    WHILE (coefficient list not empty)
        get one coded coefficient from the list;
        IF coefficient was coded as P, N or Z then
            code next scanned coefficient;
            put the coefficient into coded coefficient list;
            IF coefficient was coded as P or N then
                add abs(coefficient) to subordinate list;
            set coefficient position to zero;
        END IF
    END WHILE
END PROCEDURE

```

Figure 26. Algorithm for the Dominant Pass

Here we use a coefficient list to keep track of the identified zerotrees. To start this loop, we have to initialize the coefficient by manually adding the first quad-tree root coefficients to the coefficient list. The step of “code next scanned coefficient” checks the next uncoded coefficient of the image, indicated by the scanning order and outputs a **P**, **N**, **T** or **Z**. After coding the coefficient, it is put in the coded coefficient list. Thus, the coded coefficient list contains only coefficients which have already been coded, i.e. a **P**, **N**, **T**, or **Z** has already been outputted for these coefficients. Finally, if a coefficient is coded as a **P** or **N**, we remove it from the image and place it on the subordinate list.

This loop stops as long as the coefficients at the last level, the highest subbands, are coded as zerotrees.

The subordinate pass follows the dominant pass:

```

PROCEDURE SubordinatePass
BEGIN
    subordinateThreshold = currentThreshold / 2;
    FOR all elements on subordinate list do
        BEGIN
            IF (coefficient > subordinateThreshold)
                output a one;
            coefficient = coefficient - subordinateThreshold;
        ELSE output a zero;
        END FOR
    END PROCEDURE

```

Figure 27. Algorithm for the Subordinate Pass

Entropy Coding

The basic idea of entropy coding is to apply one or more lossless compression methods on EZW coded data to obtain a better compression ratio.

Zlib is a free compression library coded in C and downloaded from the Internet. It compresses the source data by LZ77 (Lempel-Ziv1977) and/or Huffman entropy encoding. Its deflation algorithms are similar with those used by PKZIP (an MSDOS based software by PKWARE, Inc.). A wrapper class is written to apply Zlib compression engine as the last stage of MinImage to compress all the wavelet coefficients to a final wavelet image file with the extension LET. [6]

Specification of the Wavelet Image File: LET

A LET file is defined as the wavelet image file read and written by MinImage.

The following is the Let file format.

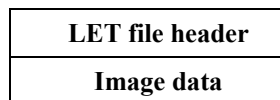


Figure 28. LET File Format

Note: The LET file header contains the information used to uncompress the image. The image data is the EZW coded wavelet coefficients compressed by Zlib engine.

Figure 29 shows the detail of the LET file header.

LET file header	Size(bytes)	Content
<i>Identifier</i>	2	“TE”
<i>ImageType</i>	1	0 = grayscale 1 = truecolor
<i>Width</i>	2	Pixel width of the whole image
<i>Height</i>	2	Pixel height of the whole image
<i>SampleMode</i>	1	0 = H1V1 1 = H2V1 2 = H2V2
<i>WaveletBlockBase</i>	1	\log_2 WaveletTransformBlockWidth
<i>WaveletType</i>	1	0 = Standard Haar 1 = Nonstandard Haar 2 = Daubechies 4-tap 3 = Daubechies 6-tap 4 = Daubechies 8-tap 5 = Daubechies 10-tap 6 = Spline 2_2 7 = Spline 2_4 8 = Spline 4_4
<i>ImageDataSize</i>	4	The size (bytes) of the image data in the Let file.
<i>StopThreshold</i>	6	The stop threshold for EZW coding

Figure 29. LET File Header Format

Summary

In this chapter, we discussed the baseline and detail of implementation of MinImage. First, the image is sampled and preprocessed, then, we apply wavelet transform to preprocessed data to get wavelet coefficients. After this, wavelet coefficients are coded by EZW coding algorithm. Finally, Zlib entropy coding is used to perform a lossless compression to obtain a better compression ratio.

CHAPTER 5 EXPERIMENTS

In this chapter, we analyze the compression results produced by MinImage. These include the subjective and objective qualities of reconstructed images for different wavelet types, timing of composition and decomposition for different wavelet types, and timing of EZW coding algorithm for different compression ratios. As a necessary background, the basic concept of objective distortion criteria is introduced.

Objective Distortion Criteria

A natural way to determine the fidelity of a recovered image is to find the difference between the original and reconstructed values. Two popular measures of distortion are the squared error measure and the absolute difference measure, which are called *difference distortion measures*. If $\{x_n\}$ is the source output and $\{y_n\}$ is the reconstructed sequence, the squared error measure is given by

$$d(x, y) = (x - y)^2$$

and the absolute difference measure is given by

$$d(x, y) = |x - y|.$$

Practically, it is difficult to examine the difference on a term-by-term basis. Some average measures are used to summarize the information. The most often used average measure is the average of squared error measure. This is called the *mean squared error* (MSE) and is often denoted by the symbol δ_d^2 :

$$\delta_d^2 = \frac{1}{N} \sum_{n=1}^N (x_n - y_n)^2.$$

If we are interested in the size of the error relative to the signal, we can find the ratio of the average squared value of the source output and the MSE. This is called the *signal-to-noise-ratio* (SNR).

$$SNR = \frac{\delta_x^2}{\delta_d^2}$$

The SNR is often measured on a logarithmic scale and the units of measurement are *decibels* (dB).

$$SNR(dB) = 10 \log_{10} \frac{\delta_x^2}{\delta_d^2}$$

Sometimes we are more interested in the size of the error relative to the peak value of the signal x_{peak} than the size of the error relative to the average squared value of the signal. This ratio is called the peak-signal-to-noise-ratio (PSNR) and is calculated by the following equation:

$$PSNR(dB) = 10 \log_{10} \frac{x_{peak}^2}{\delta_d^2}.$$

PSNR is the most commonly used value to evaluate the objective image compression quality.

Objective Quality

To compare the objective quality for different wavelet types, we use the standard testing image (see Figure 30).



Figure 30. Lenna 256*256

Table 8 illustrates one set of compression results.

Table 8. Comparison of Compression Results by Using Different Wavelets

Wavelet	Compression Ratio	Compressed File size	PSNR
Harr	9.177 %	18048 bytes	35.230 dB
D 4	8.613 %	16938 bytes	36.515 dB
D 6	8.469 %	16709 bytes	35.246 dB
D 8	8.426 %	16570 bytes	36.946 dB
D 10	8.336 %	16393 bytes	35.643 dB
SP 2_2	8.321 %	16365 bytes	36.962 dB
SP 2_4	8.278 %	16280 bytes	37.152 dB
SP 4_4	8.235 %	16196 bytes	37.644 dB

From the above table, we observe that biorthogonal spline wavelets achieve better compression results than orthogonal wavelets. Haar wavelet has the worst result.

Subjective Quality

To compare the subjective quality for different wavelet types, we choose five different standard testing images. All compressed images have the same compression ratio (1%). For each image, we present the reconstructed images in random order to some volunteers to evaluate the quality based on the original image (graded as 10). The following table 9 shows the result, which is conform with the objective result.

Table 9. Comparison of Subjective Quality

Wavelet	Average grade
Haar	3.2
D 4	5.5
D 6	5.4
D 8	5.4
D 10	5.6
SP 2_2	5.9
SP 2_4	6.0
SP 4_4	6.2

Haar wavelet obtains the worse subjective quality because of its discontinuity, which results in the block artifact in reconstructed images. It is hard to tell the difference of reconstructed images compressed by different Daubechies orthogonal wavelets. Birorthogonal spline wavelets achieve the best subjective quality because those reconstructed images are smoother than the others.

Quantization Methods

Originally MinImage used Uniform Quantization to code the transformed wavelet coefficients [6]. EZW coding algorithm is adopted to achieve better compression result.

Table 9 illustrates the compression result for these two different methods.

Table 10. Comparison of Compression Results by Using Different Quantization Methods

Wavelet	Quantization	Compression Ratio	Compressed File size	PSNR
Harr	UQ	9.177 %	18048 bytes	35.230 dB
	EZW	8.788 %	17282 bytes	35.710 dB
D 4	UQ	8.613 %	16938 bytes	36.515 dB
	EZW	8.443 %	16605 bytes	36.433 dB
D 6	UQ	8.469 %	16709 bytes	35.246 dB
	EZW	8.058 %	15874 bytes	36.580 dB
D 8	UQ	8.426 %	16570 bytes	36.946 dB
	EZW	8.396 %	16511 bytes	37.140 dB
D 10	UQ	8.336 %	16393 bytes	35.643 dB
	EZW	7.942 %	15619 bytes	36.381 dB
SP 2_2	UQ	14.550 %	28615 bytes	40.650 dB
	EZW	14.510 %	28536 bytes	41.190 dB
SP 4_4	UQ	12.917 %	25403 bytes	40.710 dB
	EZW	12.756 %	25086 bytes	41.271 dB

From this table, we can verify that EZW coding obtains better objective compression results than Uniform Quantization method for every wavelet type.

JPEG V.S. LET

JPEG stands for Joint Photographic Experts Group, the original name of the committee that wrote the standard. JPEG is a standardized image compression standard that uses the DCT (Discrete Cosine Transform). As we discussed above, wavelet based compression method should obtain better compression ratio than the DCT based compression method. The following test confirms this result.

The standard 512*512 grayscale Lenna image (see Figure 32) is used for this test . The LET files on the right side (see Figure 31) are compressed by using the SP4_4 wavelet and the EZW coding.

The left side images (see Figure 31) are compressed by using the JPEG standard algorithm called CJPEG, version 6 (2-Aug-95) by Thomas G. Lane (Copyright (C) 1995) [16].

Table 10 lists results for four different compression ratios.

Table 11. Comparison of Compression Results between JPEG and LET.

Compression Method	Compression Ratio	Compressed File size	PSNR
JPEG	1.60 %	4213 bytes	21.93 dB
LET	1.19 %	3129 bytes	28.20 dB
JPEG	3.07 %	8078 bytes	30.40 dB
LET	2.97 %	7812 bytes	32.05 dB
JPEG	6.36 %	16752 bytes	34.74 dB
LET	6.06 %	15944 bytes	35.24 dB
JPEG	12.40%	32640 bytes	37.81 dB
LET	12.39 %	32618 bytes	38.47 dB



CJPEG

Compressed File Size: 4213 bytes

PSNR: 21.93 dB

(a)



LET

Compressed File Size: 3219 bytes

PSNR: 28.20 dB

(b)



CJPEG

Compressed File Size: 8078 bytes

PSNR: 30.40 dB

(c)



LET

Compressed File Size: 7812 bytes

PSNR: 32.05 dB

(d)

Figure 31. Comparison of JPEG and LET

From the Table 10, we can see that LET has better objective (PSNR) quality for all different compression ratios. Because JPEG applies DCT on 8*8 blocks, the block artifact (see image (a) in Figure 32) can be seen from the reconstructed image, especially for low quality image.



Figure 32. Original 512*512 Grayscale Lenna

Timing Analysis

In this section, we show the decomposition and composition time comparison(see Figure 32), EZW coding and decoding time comparison, and EZW coding time with different compression ratio (see Figure 33). All timing data were collected from a Dell Pentium III desktop machine running on Windows NT 4.0.

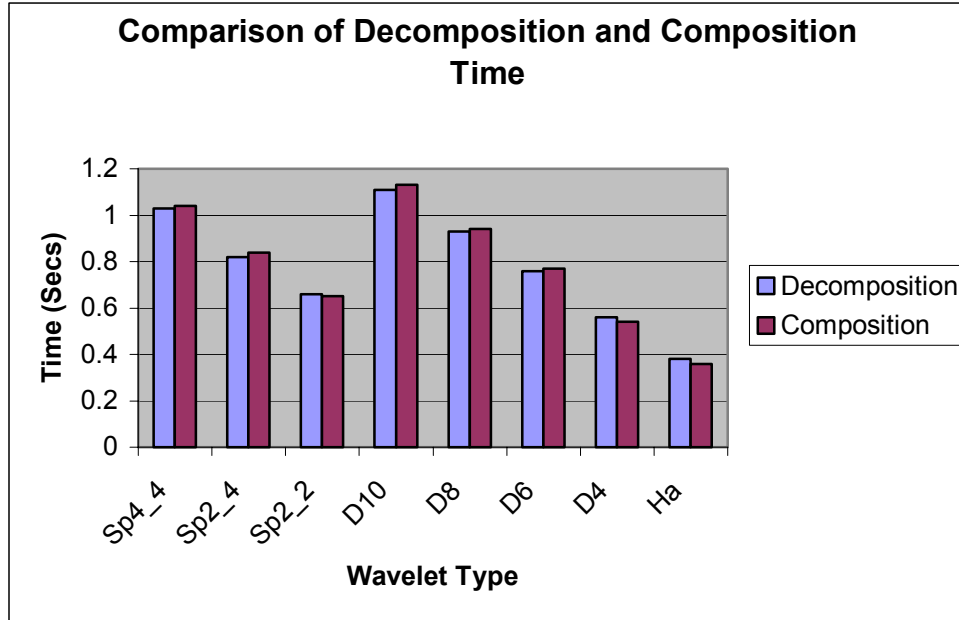


Figure 33. Comparison of Decomposition and Composition Time

Because the wavelet decomposition and composition times are entirely determined by the filter length, Haar wavelet (2 taps) takes the least time, and Daubechies 10-tap orthogonal wavelet takes the most time.

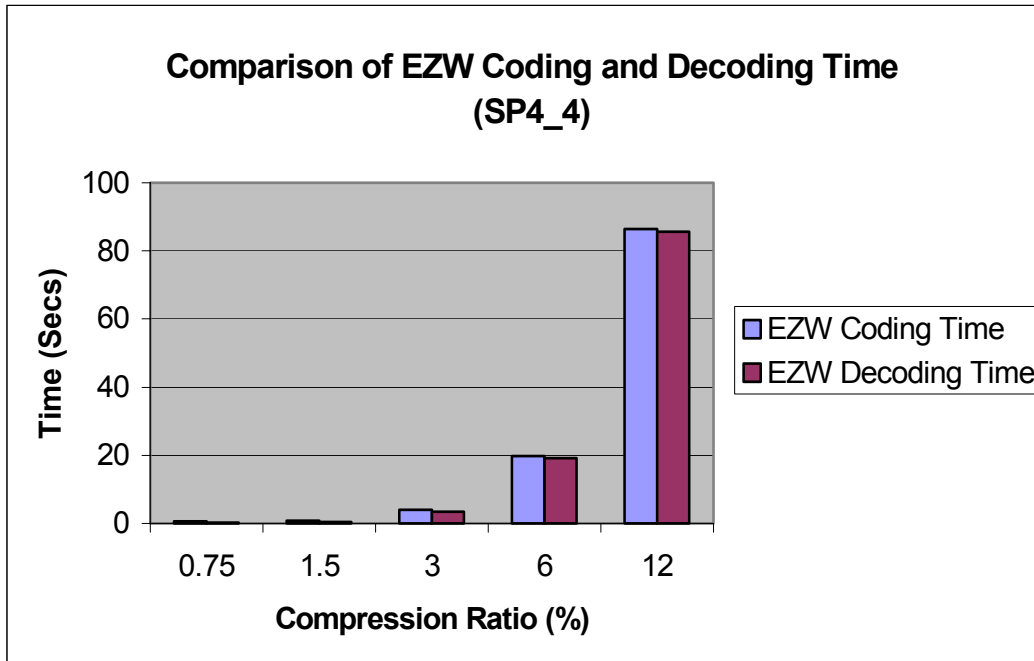


Figure 34. Comparison of EZW Coding and Decoding Time

Figure 33 shows that the EZW coding and decoding take about the same time for same the compression ratio. However, when the compression ratio increases, the EZW coding and decoding time will increase significantly.

To explain this, let us recall the EZW coding algorithm. When the stop threshold is high, for every main loop, the transformed wavelet coefficients maintain a good zerotree structure, which makes EZW coding algorithm efficient. On the other hand, with lower stop threshold, a lot of time will be wasted to check the potential zerotree structures, which are not most of the time for a small threshold.

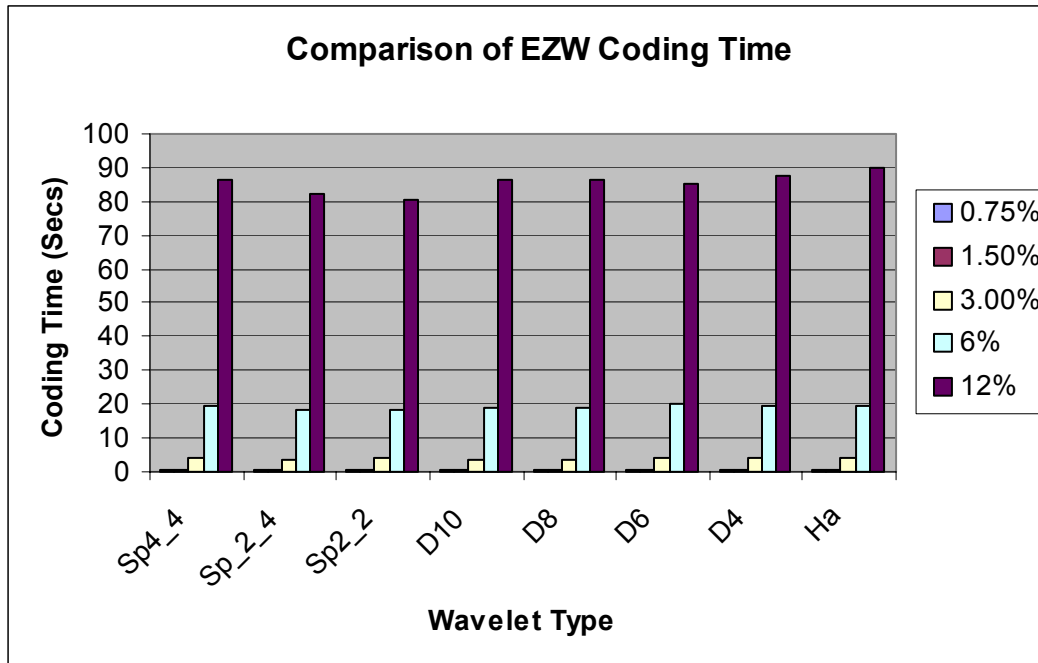


Figure 35. Comparison of EZW Coding Time with Different Wavelets and Different Compression Ratios

Because the EZW coding takes so much time, especially for higher compression ratios, the total compression time will be dominated by the compression ratio. From Figure 34 we can see that there is not much difference in the EZW coding time between different wavelet types for a same compression ratio.

Summary

In this chapter, we first introduced the basic concept of objective distortion criteria. Then, we analyzed the compression results using MinImage. We compared the subjective and objective qualities of reconstructed images for different wavelet types, composition and decomposition time for different wavelet types, and the EZW coding time for different compression ratios.

CHAPTER 6

CONCLUSION

This thesis introduced the basic wavelet theory, derived some orthogonal and biorthogonal wavelet filters and discussed the EZW coding algorithm. A software tool called MinImage was used to compare different wavelet methods.

The Haar wavelet transform is the simplest one to implement, and it is the fastest. However, because of its discontinuity, it is not optimal to simulate a continuous signal. Based on our experiments, Haar wavelet obtained the worst compression result, which proves the above statement.

Daubechie found the first continuous orthogonal compactly supported wavelet. Note that this family of wavelets is not symmetric. The advantage of symmetry is that the corresponding wavelet transform can be implemented using a mirror boundary condition, which reduces boundary artifacts.

It was proved that no wavelet could be compactly supported, symmetric, continuous, and orthogonal at the same time. To get the symmetry property, we introduced biorthogonal spline wavelet by relaxing the orthogonal condition. For the biorthogonal spline wavelet, the scaling function is a B-spline. The B-spline of degree N is the shortest possible scaling function of order $N-1$ and B-splines are the smoothest scaling functions for a filter of a given length. Our experiment also proved that biorthogonal spline wavelet outperformed the Daubechies orthogonal wavelet.

Compared with the current JPEG image compression standard, MinImage obtains better compression results. However, the run time of MinImage increases significantly to achieve a high quality compression ratio because of the EZW coding algorithm.

As an application, MinImage gives the user an easier way to do a variety of tests about wavelet image compressions. It could be improved in several ways.

1. To get the symmetric wavelets, one solution is to relax the orthogonal condition. We derived the biorthogonal spline wavelet, which gives better compression results than orthogonal wavelets. Another solution is called multiwavelets [20]. The basic idea is to use a vector of scaling functions to generate multiwavelets with orthogonality, symmetry, continuity, and compact support properties at the same time. MinImage could be more powerful if multiwavelets transform methods are added.
2. The EZW coding method is one of the more efficient quantization and coding methods for wavelet coefficients. However, MinImage could be more powerful if some more efficient quantization methods, such as space-frequency quantization [21] or context-based entropy coding [22], can be implemented.

3. The entropy coding library, Zlib, is imported to MinImage. Since it uses traditional coding methods, it may not be the best coding method for EZW coded wavelet coefficients.

BIBLIOGRAPHY

- [1] J. M. Shapiro. "Embedded Image Coding Using Zerotrees of Wavelet Coefficients", IEEE Transactions on Signal Processing, Vol. 41, No. 12 (1993), pp. 3445-3462.
- [2] M. Unser, "Splines, A Perfect Fit For Signal/Image Processing", to appear in IEEE Signal Processing Magazine.
- [3] K. Sayood, *Introduction to Data Compression*, Second edition, Academic Press, 2000.
- [4] A. Fournier, "Wavelets and their Applications in Computer Graphics", University of British Columbia, SIGGRAPH'95 Course Notes, 1995.
- [5] C.S. Burrus, R.A. Gopinath, and H.Guo. *Introduction to Wavelets and Wavelet Transforms*, Englewood Cliffs, NJ: Prentice Hall, 1998.
- [6] H. Gu, "Image Compression Using the Haar Wavelet Transform", Masters thesis, East Tennessee State University, 2000.
- [7] G. Strang, "The Discrete Cosine Transform", Society for Industrial and Applied Mathematics, 1999.
- [8] P. Morton and A. Petersen, *Image Compression Using the Haar Wavelet Transform*, College of the Redwoods, 1997.
- [9] I. Daubechies, *Ten lectures on wavelets*, Society for Industrial and Mathematics, Philadelphia, PA, 1992.
- [10] M.Unser, Aldroubi and M. Eden, "A family of polynomial spline wavelet transforms", Signal Proecssing, Vol.30, No.2, pp.141-162, January 1993.
- [11] N.S. Jayant and P.Noll. *Digital Coding of Waveforms*, Englewood Cliffs, NJ:Prentice Hall, 1984.
- [12] S. Dunn, "Digital Color", <http://davis.wpi.edu/~matt/courses/color>
- [13] S. Saha, "Image Compression - from DCT to Wavelets"
<http://www.acm.org/crossroads/xrds6-3/sahaimgcoding.html>
- [14] C. Valens, "EZW Coding", <http://perso.wanadoo.fr/polyvalens/clemens/ezw>
- [15] B.B. Hubbard, *The World According to Wavelets*, A K Peters Wellesley, Massachusetts, 1995.
- [16] C. McCrosky, "Demo of wavelet compression and JPEG",
<http://www.cs.usask.ca/faculty/mccrosky/dwt-demo/index.html>
- [17] J. Ziv, A. Lempel, "A Universal Algorithm for Sequential Data", Compression, IEEE Trans. Inform. Theory, 1977, vol. 23, no. 3, pp. 337-343.
- [18] T.A. Welsh, "A Technique for High-Performance Data Compression", Computer, 1984, vol. 17, no. 6, pp. 8-19.
- [19] M.F. Barnsley, A. Jacquin, F. Malassenet, L. Reuter, and A.D. Sloan, "Harnessing chaos for image synthesis", Computer Graphics, vol 22, no. 4 pp. 131-140, 1988.
- [20] D. Hong and A. Wu, "Orthogonal Multivavelets of Multiplicity Four", Computer and Mathematics with applications, 2000, vol 20, pp. 1153-1169.

- [21] Z. Xiong, K. Ramchandran, and M. T. Orchard, "Space-frequency quantization for wavelet image coding", to appear in IEEE Trans. Image Processing, 1997.
- [22] C. Chrysafis and A. Ortega, "Efficient context-based entropy coding for lossy wavelet image compression", DCC, Data Compression Conference, Snowbird, UT, March 25 - 27, 1997.
- [23] J. Morlet, "Sampling theory and wave propagation", NATO ASI Series, Vol. 1, Issues in Acoustic signal/Image processing and recognition, Berlin, pp. 233-261, 1983

VITA

PANRONG XIAO

- Personal Data: Date of Birth: October 25, 1970
 Place of Birth: Zigong, P.R. China
 Marital Status: Married
- Education: Zhongshan University, Guangzhou, P.R. China;
 Applied Mathematics, B.S., 1990
 East Tennessee State University, Johnson City, Tennessee;
 Mathematics, M.S., 2000
 East Tennessee State University, Johnson City, Tennessee;
 Computer Science, M.S., 2001
- Professional
Experience: Programmer, Marine Surveying Bureau, Guangzhou, P.R. China, 1990-1993
 System Engineer, China Construction Bank, Panzhihua, P.R. China, 1993 – 1998
- Publications: J. Boland, J. Kuffman, P. Xiao, (2000) “Mode Graphs”, Congressus Numerantium, Volume 142,
 2000, pp. 161-172.
 J. Boland and P. Xiao, “On a Conjecture Concerning the Eccentricity”,
 to appear in Bulletin of the Institute of Combinatorics and its Applications.
- Honors and
Awards: 1998-2001, Outstanding International Student,
 East Tennessee State University
 2000-2001, Outstanding Computer Science Graduate Student,
 East Tennessee State University