Dynamic Region-based Wavelet Coding for Telemedicine Applications

by

Hui Li
B.Sc. (Comp. Sci.) Chengdu University of Science and Technology

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
March 1997

APPROVAL

Name: Hui Li

Degree: Master of Science

Title of Thesis: Dynamic Region-based Wavelet Coding for Telemedicine Applications

Examining Committee: Dr. Fred Popowich

Chair

Dr. M. Stella Atkins

Senior Supervisor

Dr. Jacques Vaisey

Supervisor

Dr. Ze-Nian Li

Examiner

Date Approved:

Abstract

In this thesis, a novel image compression scheme is presented which is specially well suited for image transmission over the narrow-band networks typically required for telemedicine to remote regions. A wavelet compression algorithm is enhanced with the ability to dynamically compress different regions of the image. This feature is provided while keeping the algorithm's embedding ability, which leads to an 'importance' embedding rather than the traditional 'energy' based embedding. To incorporate region selection in a wavelet-based compression algorithm, the region edges are carefully tuned to eliminate the negative influence that the wavelet transform has on the region algorithm. Tests of this new algorithm on standard test images and ultrasound images showed that both the embedding and region-based features can be dynamically incorporated into the wavelet algorithm with only a small overhead.

Acknowledgments

I would like to thank my senior supervisor, Dr. Stella Atkins. She Guided and supported me in the entire process of this thesis research work. Special thanks to Dr. Jacques Vaisey for his valuable suggestions during my research. Thanks also to Dr. Ze-Nian Li for taking the time to act as thesis examiner and to Dr. Fred Popowich for chairing the examining committee.

I would like to thank Lijuan Chen who help me preparing the test data and Victor Jung and Glen Scott of EYETEL who help me with the PC version implementation.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Telemedicine [7] is the provision of health care services via interactive audio visual and data communications. It is a digitized and computerized process incorporating many technologies like communications, databases, user-interfaces and medical science while the foundation of it is communication. As the medical images may be very big, the transmission and storage of the medical image often cause difficulties.

For example, a single 2048×2048 X-ray image may use 4 megabytes, and transmitting it over a telephone line operating at 9600 bits per second (bps) may take one hour, which would be very inefficient. So, if we want to get better performance, we'll have to either increase the bandwidth of the communication channel or apply some compression during transmission.

Furthermore, the situation of narrow-band communication can't be totally eliminated in the near future. In many remote countryside places, wide-band communication service may be unavailable; in moving vehicles, ships or planes, it is hard to achieve wide-band communication because of the nature of the channel (e.g. fading). So a compression ratio of at least 10:1 is highly required, and better could reach 30:1. Then for the previous example, it means that the image could be transmitted in only a few minutes.

There already have been many very successful works on image compression, and a large variety of algorithms have been proposed. A standard compression algorithm, JPEG, is available which will get good results on most images except when the compression ratio is high. Recently, the wavelet transform was proposed and it can achieve a better compression ratio without increasing computational complexity.

When using wavelet algorithms to compress an image with a ratio of 10:1 or even 30:1, down to below one bpp, the decompressed image's quality is enough for most uses. But for medical usage there are still some problems. First, as the medical image's quality may influence the diagnosis result, lossy compression has not yet been completely accepted for use in diagnostic usage. Second, unlike ordinary usage, which is mainly concerned with the overall impression of the image, medical imaging may be very concerned about the detail at some region (e.g. a pathologically important region), so the deviation caused by a 30:1 or 10:1 compression at that region may be unacceptable.

By considering the strict requirement of medical imaging and the fact of low-band communication in the future decade, the current image compression technology is still not adequate for the task. Some new technology is required.

1.1.1 Embedded Compression and Region Based Compression

One of the recent valuable achievements for providing better service on image transmission is *embedded zerotree wavelet image coding* (EZW) [20, 22, 21]. 'Embedded' here means that the image is coded in such a way that encodings of the image at lower bit rates are always at the beginning of the bit streams of the encoding of the same image at higher bit rates. This leads to a result that a small leading part of the encoding bit-stream can provide a coarse reconstruction of the original image. With the use of embedded coding, we could already make our image transmission system more acceptable to the medical users. For example, we could first transmit a lower-quality image in a very short time for initial viewing and could get a quick idea for the following steps of diagnosis, then with a following slower transmission of further detail to double check and draw a formal and legal conclusion; or we could let the remote expert browse the compressed images in the local image database by just transmitting the head part of them and then only the relevant images are transmitted in their entirety.

Interest in a medical image often can be confined to a specific region. A different bit-rate can be allocated to different spatial regions according to their 'importance' to the user. This kind of bit-rate allocation, different from the widely used optimal bit allocation [2] to minimize the total distortion such as the *mean squared error* (MSE) of the image, is a *region based bit-rate allocation.* A region based method provides another way solving the conflict between the high fidelity requirement for medical images and the low bandwidth communication.

1.1.2 Dynamic Region-based Image Coding

We have seen that both embedded coding and region based compression are helpful for our target of improving the image transmission performance for telemedicine usage, but up to now, we haven't found any research work which will work on both of them. In this paper, a new very flexible and high performance compression scheme called *dynamic region-based wavelet* (DRW) coding is implemented. It is an algorithm which provides region based compression; meanwhile, it is an embedded coding algorithm which also supports progressive transmission. Furthermore, this is a completely dynamic algorithm which can choose and change the *region of interest* (ROI) to encode on the fly. If no region of interest is selected, the algorithm will progressively encode the whole image until some target requirement such as bit-rate or distortion rate is reached. If a region of interest is selected, the successive bit-stream only progressively encodes that region(of course it will stop if some requirement is reached). The ROI can be changed to any shape and position in the middle of sending the previous ROI. If needed, after the encoding of the ROI, the remaining image can continue to be coded and transmitted until a bit-rate or some other requirement is reached.

As mentioned previously, wavelet transform based compression has very good performance on both compression ratio and speed. The DRW algorithm inherits the high performance of wavelets and adds the novel feature of dynamic embedded coding.

1.2 Research Methodology

In this thesis a state-of-the-art still-image compression method, embedded zerotree wavelet(EZW) coding [20] is researched. Comparing with other image compression scheme, such as JPEG, this wavelet-transform based sub-band coding has better rate-distortion performance. In the mean time, it has many attractive features like bit-rate control, progressive coding. We adapt this EZW scheme and enhance it with the dynamic region-based feature to fit it for telemedicine applications.

The research work of this thesis were implemented in the WiT ™ visual programming language [4, 5]. WiT is a powerful package for designing complex imaging algorithms. In WiT, the work is done by building graphs, called igraphs, with icons and links. The icons represent operators and the data objects flow on the links between operators. An operator can generate objects, process objects or do flow control of the objects. The igraphs could be executed efficiently by exploiting inherent parallelism using a server/client model to distribute computation across a network of resources. It has a large operator (icon in igraph) library which provides basic and advanced image procession calls. An efficient mechanism is presented to the programmer to add their own special purpose operators.

1.3 Outline of the Thesis

In Chapter 2, the background and previous work on telemedicine, image compression and wavelets is discussed.

In Chapter 3, Wavelets and their features for image compression usage is discussed.

In Chapter 4, the EZW algorithm, which is fundamental for the later work, is discussed. Our implementation and test results are also given in this chapter.

In Chapter 5, our new dynamic region-based wavelet coding is presented together with our implementation and test results.

In Chapter 6, the summary and the potential future work will be given.

Chapter 2

Telemedicine and Image Compression

2.1 Telemedicine

Telemedicine is the provision of health care services via electronic transmission of medical information among different sites[6]. It is now being used or tested in many areas of health care as pathology, surgery, physical therapy, radiology and so on.

Telemedicine makes it possible that the doctor and patient can be in different places. Time and expense can be saved in transporting the patient to the health care expert in the central city hospital or transporting the expert to the patient. Thus, health care service quality is improved while the cost is simultaneously reduced. Another application is to let doctors have rapid access to patient records and be able to retrieve vital patient information in multiple formats.

The development of telemedicine can be traced back to the 1960s when it was used to monitor astronauts' vital signs. It's becoming more and more attractive with the improvements in computer technology and the growth of the information super-highway. Now there are many telemedicine research or application projects under-way. In Bosnia, communication and medical equipment is used by the U.S. military to provide high quality medical care services.

2.2 The Basics of Image Compression

With the continuing growth of modern communications technology, demand for image transmission and storage is increasing rapidly. The improvement of computer hardware including processing power and storage power has made it possible to utilize many sophisticated signal processing techniques in advanced image compression algorithms. Many technologies are arising in this area, and a practical image compression algorithm is usually a composition of more than one technique. This chapter provides some background information on image compression.

Generally, image data compression is concerned with minimization of the number of information carrying units used to represent an image[10].

2.2.1 Redundancy

Image compression takes advantage of the fact that there is a lot of redundant information contained in the original image. Basically there are three kinds of redundancy: Psycho-visual, Inter-pixel and Coding:

- Inter-pixel Redundancy: the images we are to deal with are not just random pixels, they have many kinds of structures. This means there are statistical dependencies between pixels, especially between neighboring pixels. This kind of dependence is a redundancy which may lead to a compression.

- Psycho-visual Redundancy: the human eye does not respond with equal sensitivity to all image signals; some are even not perceivable. Certain information simply has less relative importance than other information in our visual processing. Therefore, eliminating or distorting some information may be acceptable.

- Coding Redundancy: the uncompressed image usually is coded with each pixel having a fixed length code. For example, an image with 256 gray scales is represented by an array of 8-bit integers. It is convenient for processing the image, but usually it's space wasting. Using some variable length code scheme such as Huffman coding[8] or arithmetic coding[26] may save space.

There are different methods for dealing with the different kinds of redundancy, therefore, an image-compressor is usually a multi-step algorithm in order to reduce these redundancies.

Image compression may also be classified into lossy or lossless compression.

- Lossless compression permits no loss or distortion from the original image. It will ensure the fidelity but it's not so "powerful". The compression ratio reached is often lower than 3:1 or 2:1.

- Lossy compression permits distortion over the original image, and thus can achieve higher compression. Theoretically, it can compress an image at any ratio. However, as the compression ratio goes higher, the degradation of the image will become serious. For many images a ratio of 10:1 to 30:1 is acceptable, and sometimes it's even hard to see any difference with our eyes.

The reason why lossy compression and its higher compression ratio is applicable is that for the three kinds of redundancy mentioned above, the psycho-visual redundancy can only be removed in lossy compression.

In this thesis research, the focus is on lossy compression because of its high compression ratio and wider work space to be improved on. But as the legal issue of lossy compression in telemedicine haven't been settled, the lossless compression problem will still be discussed in this chapter.

2.2.2 Lossless Compression

In some applications, lossless or error-free compression is the only acceptable schema. As the lossless compressors can only achieve compression ratios around 2:1 to 3:1 for most images, the compressed image is still large. A lossless compressor is usually comprised of two steps (see Figure 2.1(a)): step 1 transforms the original image to some other format in which the inter-pixel redundancy is reduced; step 2 uses an entropy encoder to get rid of the coding redundancy. The lossless decompressor is an accurate reverse process of the lossless compressor.

2.2.3 Lossy Compression

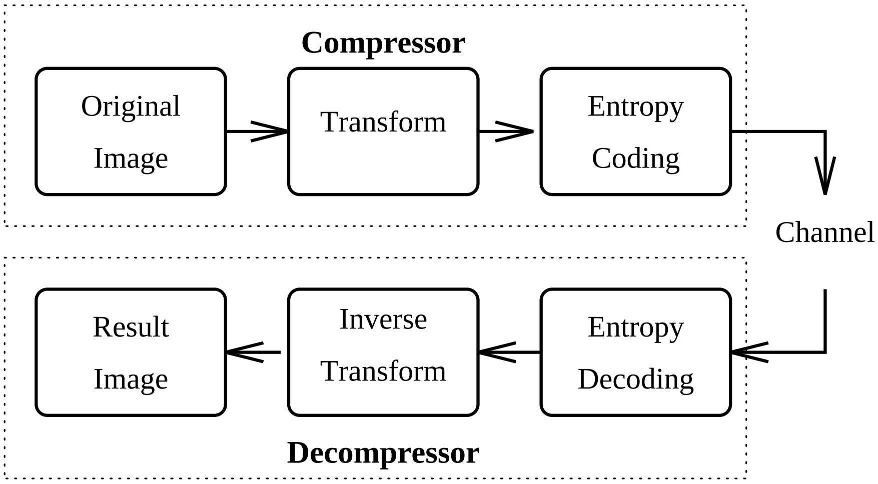
Generally most lossy compressors are three step algorithms (see Figure 2.1(b)) which could be regarded to be in accordance with the three kinds of redundancy mentioned above.

The first stage is a transform which is used to eliminate the inter-pixel redundancy. After the transform, the information becomes more packed. Many kinds of transform could be used; one kind of transform is performed on small blocks of the image, such as the variation of Fourier transform – discrete cosine transform (DCT) which is used in JPEG image file format; another kind of transform is the sub-band decomposition, which is usually applied to a whole image to split it into a series of sub-bands(sub-images) according to frequency[16]. An example of sub-band decomposition is the wavelet based decomposition filters.

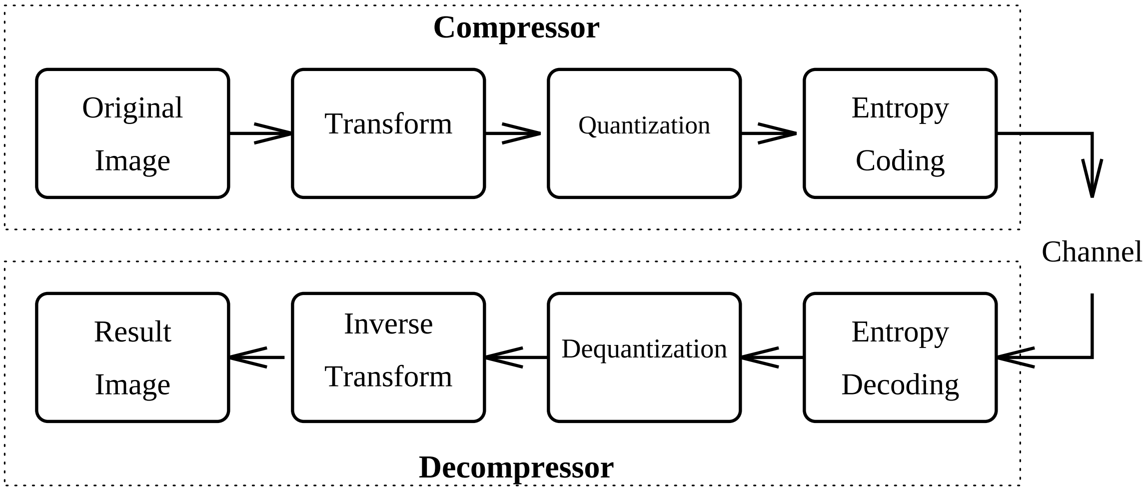
The transform result signal is then sent to a quantizer to represent it with as few bits as possible, where the psycho-visual redundancy is removed.

The quantized bits are then efficiently encoded to get more compression from the coding redundancy.

In the decompressor shown in Figure 2.1(b), the dequantization can not reverse exactly the process of quantization, so this is where the loss compression arises.



(a) Lossless image compression



(b) Lossy image compression

Figure 2.1: Diagram of lossless and lossy image compression

2.2.4 Distortion Measurement

Another important concept needed to be known about lossy compression is the distortion measurement. Lossy compression is a trade off between image distortion and the compression ratio, so when using an lossy algorithm, we need to measure the quality of the reconstructed image as well as the compression ratio. One straightforward measure is the mean square error (MSE), σ^2 between the original image f and the reconstructed image $\hat{f}$ which is denoted as:

$$\sigma^2 = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (\hat{f}_{i,j} - f_{i,j})^2 \tag{2.1}$$

here, image size is $m \times n$. Practically, the peak signal-to-noise ratio (PSNR) which is based on MSE is also widely used:

$$\mathrm{PSNR} = 10 \log_{10} \frac{f_{\max}^2}{\sigma^2} \tag{2.2}$$

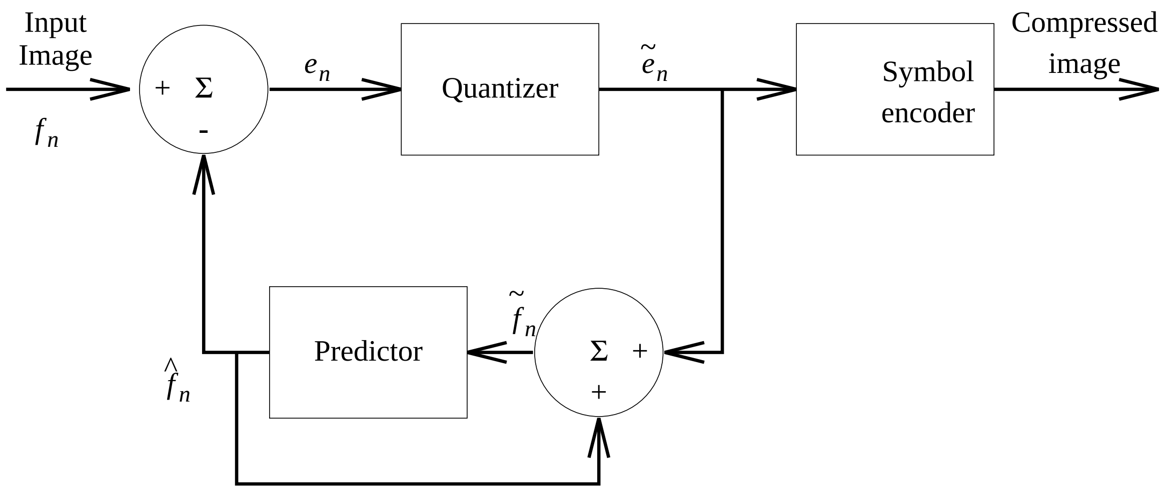
where the $f_{\max}$ is the maximum pixel value. These measurements, which are statistical terms derived from the image data, are called *objective measurements*. As the images are usually for human viewing, MSE based objective measurement does not represent our human vision process. This leads to *subjective measurement* based on subjective comparisons to tell how "good" the decoded image looks to a human viewer. One fact needing to be pointed out is that, although objective measurement, like MSE is not perfect, it is still widely used and is the most acceptable because it is quantitative, easy to compare and stable.

2.3 Survey of Current Compression Methods

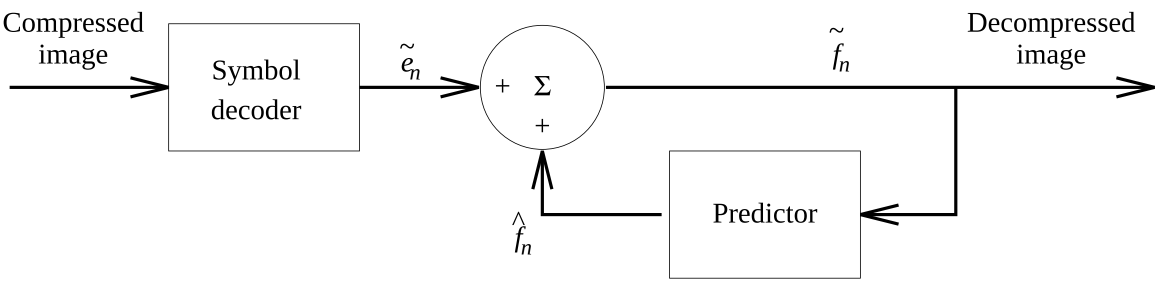
2.3.1 Predictive Coding

As mentioned before, the first step of image compression is to represent an image by uncorrelated data. One easy-to-implement method is predictive compression[10, 11, 24, 25]. In predictive compression, every pixel is encoded with a prediction error rather than its original value. Diagrams of the predictive encoder and decoder are shown in Figure 2.2. The input image f_n denotes each successive pixel of the input image; e_n is the prediction error from the original value of the pixel; $\tilde{e_n}$ is the prediction error after quantization; $\tilde{f_n}$ is the pixel value recovered from the quantized prediction error; and $\hat{f_n}$ is the prediction made for the next pixel. Note that $\tilde{e_n} \neq e_n$, so there will be distortion after the quantization, which providing the distortion of the compression. If we eliminate this quantization procedure, then $\tilde{e_n} = e_n$ and $\tilde{f_n} = f_n$, and it becomes a lossless compression.

The predictor here usually makes prediction of a pixel's value from its neighbor pixels, typically employing a linear combination of some previous pixels. There have been a lot of research works done on predictor and quantizer design in predictive compression. Predictive compression may have very low algorithm complexity which is fit for hardware implementation. The drawback of a pure predictive coding is that its performance is not as high as some other scheme such as transform coding (see subsection 2.3.2). However, it could be combined with other methods to get a hybrid coding algorithm which has higher performance[11, 28].



(a) encoder of predictive coding



(b) decoder of predictive coding

Figure 2.2: predictive coding model: (a) encoder; (b)decoder.

2.3.2 Transform Coding

In predictive coding, the image is transformed pixel by pixel and then compressed. In transform coding, a reversible transform (usually linear) is first applied onto each block of the image or the whole image to get a set of transformed coefficients, then the coefficients are quantized and symbol coded.

There are many kinds of transform; the most commonly used are discrete Fourier (DFT), discrete cosine (DCT), Karhunen-Loeve (KL) and the wavelet transform. Commonly, these transforms have the information-packing property, that is, most of the information about the original image is packed into a small portion of the transform coefficients.

Taking the advantage of the information-packing, the quantizer can selectively eliminate or more coarsely quantize the coefficients that correspond to less information of the original image. This process of selecting and quantizing the transform coefficients to minimize the distortion is called *bit allocation*. JPEG and EZW use *zonal coding* and *threshold coding* respectively.

- *Zonal coding*: choose the coefficients to code and set the number of bits to quantize a coefficient according to its maximum variance, because coefficients of greater variance tend to carry more picture information and should be coded with more bits; coefficients of smaller variance carry less picture information and could be coded with fewer bits or even not encoded. Usually, zonal coding uses fixed bits to select to code specific coefficients.

- *Threshold coding*: choose the coefficients to code and set the number of bits to quantize a coefficient according to its magnitude. The magnitude of each transform coefficient varies from image to image, so threshold coding is therefore an adaptive method to choose the coefficients and its encoding bits.

Transform coding achieves relatively larger compression than predictive methods. Although transform coding algorithms have higher complexity, with the fast development of hardware, this disadvantage is less and less important.

2.3.3 JPEG

A joint ISO/CCITT[1] committee known as JPEG (Joint Photographic Experts Group) has established the first international transform-based compression standard for continuous tone still images [14, 23]. A generic scheme is specified as a baseline JPEG method for lossy compression. The method is based on a division of the image into blocks of 8x8 pixels, after which each block is transformed with a discrete cosine transform (DCT). The transform coefficients are quantized such that the higher frequency coefficients are always quantized less accurately than lower ones, since errors in these coefficients are less visible to the eye. The quantization accuracy of the coefficients is also decided by the required image quality; when higher quality is needed, they are quantized more accurately. The quantized coefficients are reordered using a zigzag pattern to form a 1-D sequence of quantized coefficients. In this reordering, the coefficients are ordered according to increasing frequency and also from one coefficient to its neighbor. This reordering will normally result long runs of zero coefficients, and

[1]ISO standards for International Standardization Organization. CCITT standards for International Consultative Committee for Telephone and Telegraph.

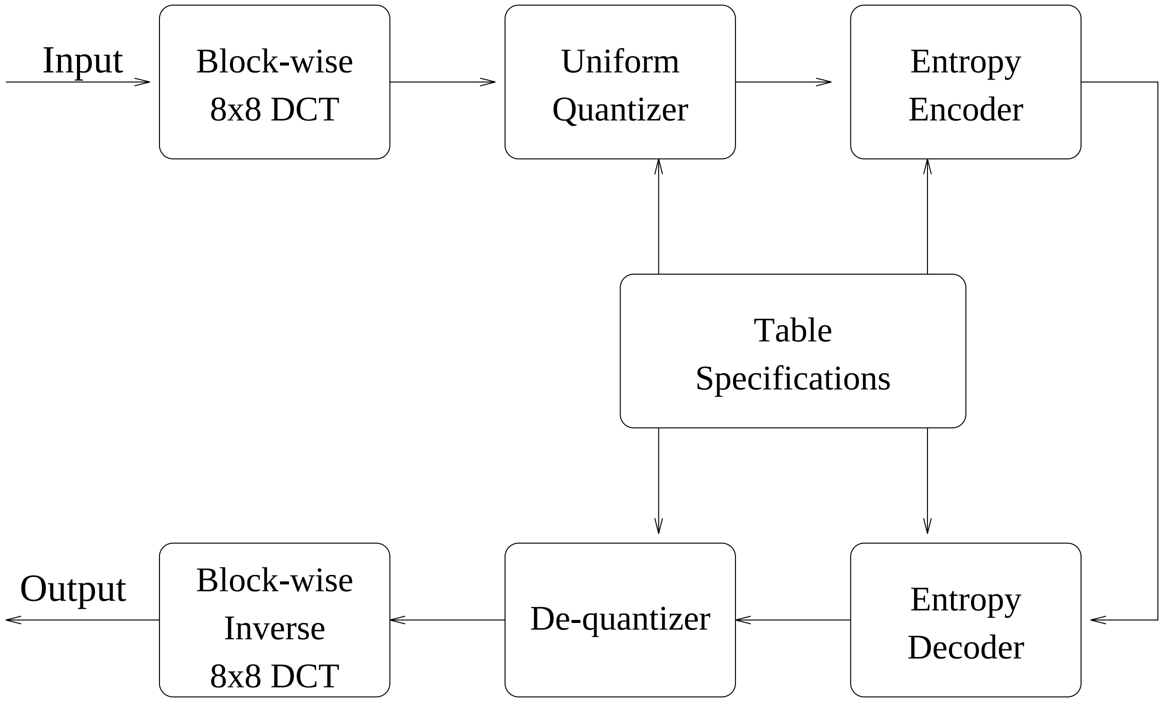


Figure 2.3: JPEG baseline coding algorithm, Here Table Specifications are the specification of bit-allocation of the quantizer and the Huffman code table which should be the same for both encoder and decoder

to take advantage of this, the AC[2] coefficient are coded using a variable-length code that defines the coefficient's value and number of preceding zeros. The DC[3] coefficient is difference coded relative to the DC coefficient of the previous sub-image. The lengths of the zeros and values of this encoding are all Huffman coded. The JPEG method performs well at high or medium bit rates, but it introduces perceptibly annoying blocking artifacts (due to the block-wise transform coding) at low bit rates (see Figure 4.10 (b)). The diagram of the JPEG algorithm is shown in Figure 2.3.

[2]AC here means all transform coefficients except DC coefficient
[3]DC here means the zero frequency coefficient

2.3.4 Lossless Compression

Lossless compression doesn't distort the image; its main focus is on how to represent an image with as low an entropy as possible and then entropy code it. Decorrelation of an image usually will lower the entropy. Following are some of the popular lossless compression algorithms currently in use:

Run length coding: represents a long run of a symbol with the symbol plus the repeating times. However, for ordinary gray-scale images, the occurrence of a long run of a symbol is rare. A solution to this is to apply run length coding on the bit-plane.

Lossless predictive coding: by removing the quantization procedure of the predictive coding mentioned in subsection 2.3.1, we will get lossless predictive coding. It has good performance on gray-scale images, and lossless JPEG [14] compression format is based on lossless predictive coding.

Multi-resolution method: the image is reversibly transformed into a group of different resolution sub-images. Usually, this reduces the entropy of the image. Then, by exploiting the tree structure which undermines this multi-resolution method, some kind of tree representation could be used to get far more compression. A successful example is in [18] [17].

Although lossless compression can only achieve 2:1 to 3:1 compression on most images, in many applications, it's the only acceptable way.

2.3.5 Entropy Coder

An entropy coder is usually an essential part of a successful image compression algorithm. It reduces the coding redundancy which is normally presented in any encoding of an image. If we construct a code book which assigns shorter code words to the more probable symbols, we may reduce the average code words length of the whole encoding; this is called *variable-length coding*. Huffman coding [8] is the most classical variable-length coding. In the situation that the source symbols are coded one at a time, Huffman coding is the optimal solution.

The entropy coder which is very popular now is called *arithmetic coding*. It doesn't code just one symbol at a time; if the source symbol length is infinite, it could achieve performance which is infinitely approaching the limit of zero-order entropy at any source symbol possibility. Alternatively Huffman coding may have poor performance on small symbol set; if the symbol set size is increased for higher compression, its adapting speed for adaptive coding will be a problem. Thus arithmetic coding is specially favorable for small symbol set adaptive coding. More descriptions about arithmetic coding are in [15, 26].

Chapter 3

Wavelets and Image Compression

Wavelets are functions that satisfy certain mathematical requirements which could be used to represent other functions or signals. Wavelets have been developed for a very long-time, but only in the last ten years, as they spread into many other fields, was their "power" recognized.

One of the very hot applications of wavelets is in image processing and analysis, as wavelets have advantages over traditional Fourier methods in many cases including image compression. In this chapter, we will give an introduction to wavelets and their relationship to image compression.

3.1 Subband Decomposition

Subband decomposition uses a linear transformation to split an image into sub-images containing different frequency information; these sub-images are called subband images. This decomposition allows processing of different frequency information separately. Recently, much research has been devoted to the discrete wavelet transform (DWT) for subband coding of images.

3.2 Wavelets

3.2.1 Introduction

Wavelets are a group of functions generated from one single function ψ by dilation and translations:

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

Here ψ is what we call the *mother wavelet or analyzing wavelet*; $\psi_{a,b}$ is the wavelet functions; a and b are dilation and translation coefficients respectively. The mother wavelet ψ has to satisfy

$$\int \psi dt(t) = 0$$

which implies at least some oscillations. Technically, it's also required that $\psi(t)$ decays faster than $|t|^{-1}$ for $t \to \infty$ which imply its local property.

The purpose of the wavelet transform is to represent an arbitrary function f as a superposition of wavelets. In practice one prefers to write f as a weighted sum of $\psi_{a,b}$ thus:

$$f(t) = \sum_{a,b} c_{a,b}\psi_{a,b}(t),$$

where $c_{a,b}$ are the wavelet transformed coefficients. However, not all wavelets are applicable in digital signal processing. According to the works of Mallat [13] and Daubechies [9], a set of wavelet orthonormal basis functions which have good performance have been developed.

3.2.2 Wavelet's Relation to Image Compression

Previously, the transforms that were widely used were Fourier-like transforms, which use periodical basis functions to transform an image from the spatial domain to the frequency domain. The transformed image has the property of energy-packing which is very important for image compression. However, for image compression, this kind of transform is not perfect, because the transformed image contains only the global frequency information. For most images, there are many discontinuities or spikes (like edges). These spikes may disperse into many coefficients of the transform, and are easy to be ignored. Sometimes the transform is only applied on small blocks of the image, such as is used in the JPEG algorithm. But, this method still has drawbacks, as the image contains information of different frequencies, and this block transformation can not deal with the low frequency information which is wider than the block-size.

On the other hand, if the block-size is increased, its spatial domain resolution for high frequency information will degrade. Another problem the block-transform algorithm may have is the block-shaped distortion to which human eyes are very sensitive.

For the purposes of signal compression, we wish to have a decomposition which contains information about both the time and frequency information of the signal. In other words, we want to know the frequency information of the signal at each point. However, we can't increase the spatial resolution and lower the frequency at the same time. This means that we must trade off between frequency and space resolution. This situation could also be understood by thinking low frequency information is spread out on the spatial domain, while high frequency information is more concentrated.

Therefore, a better way to achieve our purpose of getting local frequency information is to have a decomposition which has low resolution for low frequency information and high resolution for high frequency information. This arrangement is realized by carefully choosing the basis functions for decomposition. What we seek is a set of basis function $\{\psi\}$, each with different "width". The different widths allow us to trade off time and frequency resolution in different ways; for example, a wide basis function can examine a large region of spatial space, and resolve low frequency information, while narrow basis functions can be used to resolve the spatial position of high frequency information. These basis functions can be obtained by scaling and translating the mother wavelet. The scaling factor controls the "width" of the function. If the scale factors are power of 2, then we get a multi-resolution decomposition of the original signal [13].

$$f(t) = \sum_v \sum_k c_{v,k} \psi_{v,k}(t)$$

where

$$\psi_{v,k}(t) = 2^{v/2} \psi(2^v x - k)$$

The $c_{v,k}$ can be obtained by the wavelet transform, which is the inner product of the signal $f(t)$ with the basis function $\psi_{v,k}(t)$. It needed to be mentioned that for the purpose of compression, more restrictions need to be add to the wavelet ψ to obtain a nonredundant representation through an orthonormal or biorthogonal basis.

3.2.3 Discrete wavelet transform

In order to make the wavelet computable, a discrete algorithm is needed. In [13] Mallat proposed a discrete-time algorithm for the computation of the wavelet series given in the previous section.

3.2.4 Implementation

The wavelet transforms can be treated and implemented as bandpass filters. The forward and inverse wavelet transforms can be implemented by a pair of appropriately designed filters. We denote the forward filter pair as (H, G), where H is the low-pass filter and G is the high-pass filter.

One-dimensional implementation

If discrete time function $f(t)$ is convolved with both H and G and down-sampling the results by 2, we get $f_L(t)$ which we called *average signal* and $f_H(t)$ which we called *detail signal*. The reason we could do the down-sampling by 2 after each filtering is because this transform had split the signal's bandwidth in half. By iteratively applying the filters on the low-frequency band, we get the discrete wavelet transform mentioned at the end of previous section.

In the real application, this decomposition can't be infinitely continue, what we get is a $n - level$ wavelet decomposition.

The reverse of this wavelet decomposition is computed by up-sampling $f_L(t)$ and $f_H(t)$ by 2 and convolving them with the reverse filter pair $(\tilde{H}, \tilde{G})$ respectively. Fig 3.1 is a diagram of the forward and reverse one level wavelet decomposition. Note that the output and input are the same. In the real signal-compression applications, distortion comes from the quantization.

Two-dimensional implementation

The two-dimensional (2-D) wavelet decomposition is realized by cascading one-dimensional (1-D) wavelet filters. Figure 3.2 illustrates a one-step 2-D *dyadic* discrete wavelet decomposition which is the most widely used realization of the two-dimensional wavelet

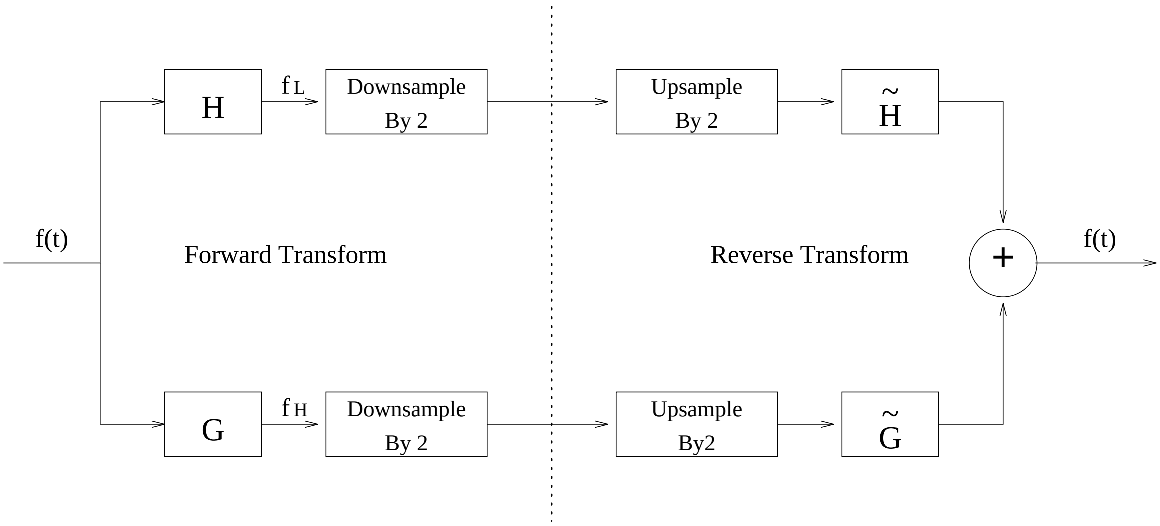


Figure 3.1: Block diagram of one level 1-D wavelet decomposition and composition. Note that f, f_L and f_H are discrete signals.

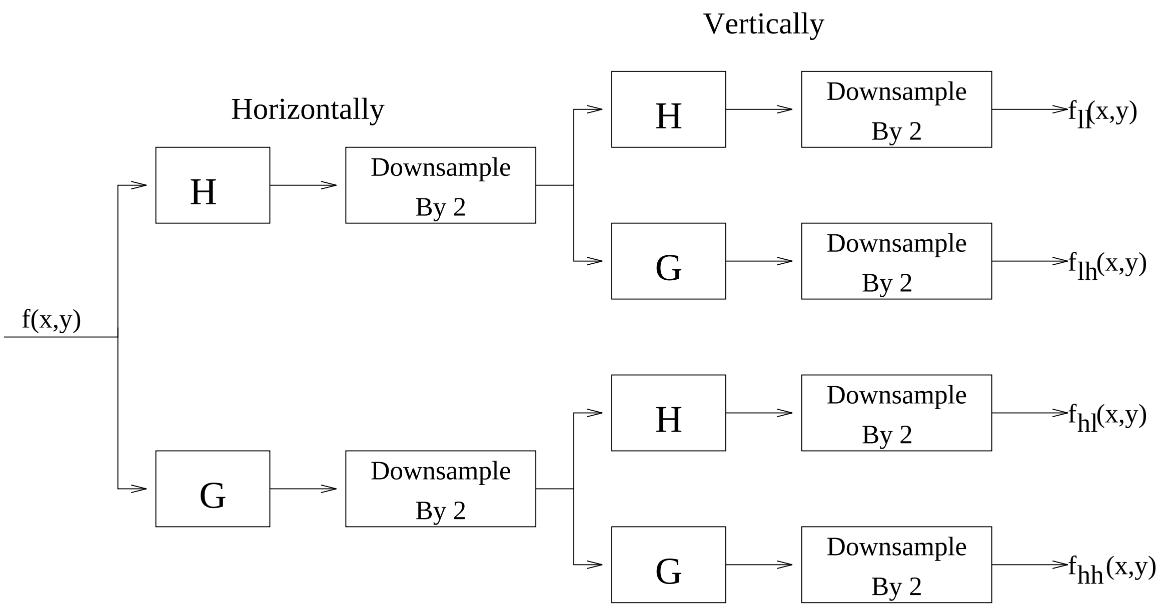


Figure 3.2: Block diagram of one level 2-D wavelet decomposition

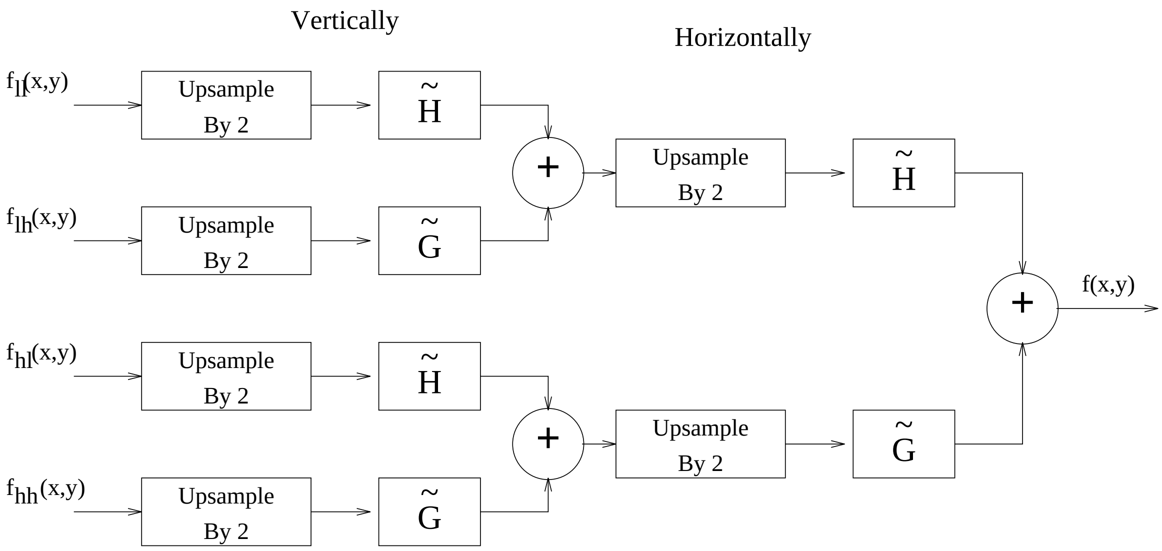


Figure 3.3: block diagram of one level 2-D wavelet composition

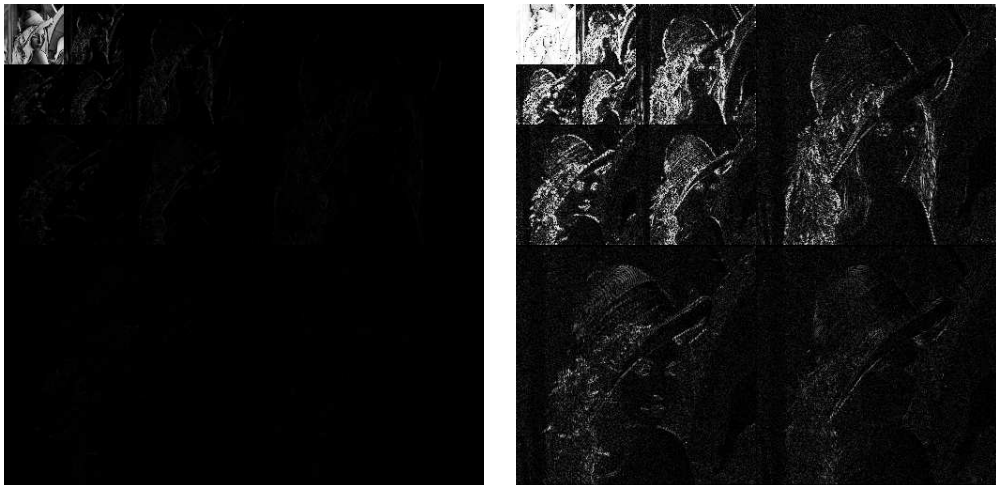
decomposition. The image $f(x, y)$ is first filtered along the horizontal direction, resulting in a lowpass image and a highpass image. Here G and H are 1-D low-pass and high-pass filters respectively. Both sub-images are down-sampled by two (dropping every other filter value) along the horizontal direction, then both sub-images are filtered and down-sampled along the vertical direction resulting four sub-images: $f_{ll}(x, y)$, $f_{lh}(x, y)$, $f_{hl}(x, y)$ and $f_{hh}(x, y)$ whose total size is the same as the original image. $f_{ll}(x, y)$ is called the *average signal* or average subband and the other three are *detail signals* or *detail subbands* which correspond to the horizontal, vertical and diagonal directions respectively.

The corresponding composition process is illustrated in Figure 3.3. The subbands are up-sampled, filtered and summed along two directions to yield a reconstructed image.

Figure 3.4 shows the one step wavelet decomposition on sample image “lena”.

(a) (b)

Figure 3.4: One level wavelet decomposition of lena. The brightness of a point is linear with the magnitude of the coefficients. The detail subbands are nearly dark because their magnitude is much smaller compare with the average subband, not because they are zero. (a) is the result of directly decomposing the image; (b) is the result of subtracting the mean value from the image then doing the wavelet decomposition.



(a) (b)

Figure 3.5: Three level wavelet decomposition of Lena. The mean value is subtracted before transform. (a) is the display with brightness linear to magnitude (b) is the equalized display, in which the detail subbands have been brightened

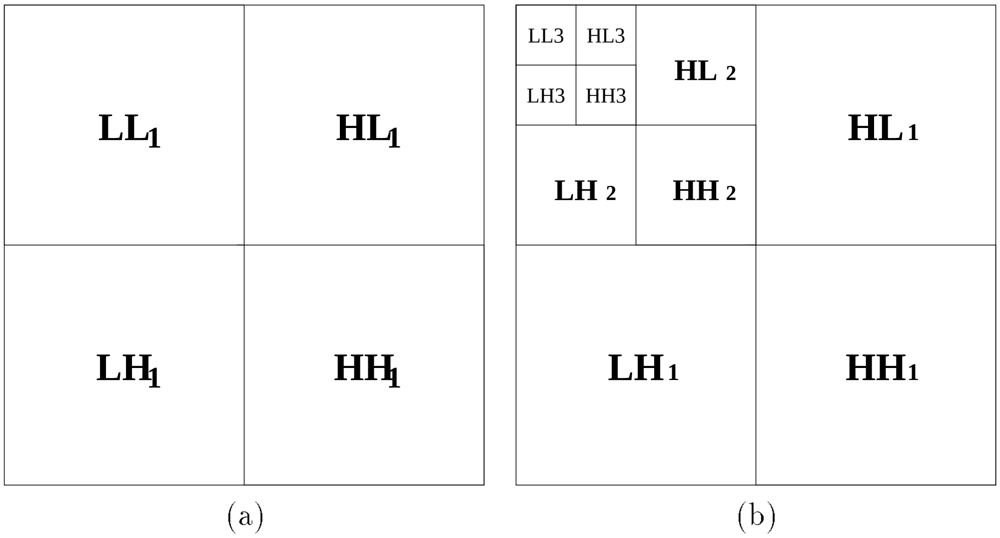


Figure 3.6: Subband of wavelet coefficients, (a) one level transformation; (b) three levels transformation

Figure 3.6 (a) shows how the subbands are named; in Figure 3.4 the visible image appears in the LL band. The detail subbands are nearly dark because their magnitude is much smaller compare with the average subband. Note that Figure 3.4 (a)'s LL band is different to that of Figure 3.4 (b). This is because the mean value of the original image is an zero frequency information which will be kept in the LL subband, so the mean value of the LL band is above zero. By subtracting the mean value from the image and then doing the decomposition, the mean value of the LL band may be close to zero. This usually leads to more near zero coefficients in LL band and lower sum magnitude of the coefficients in LL subband, which is valuable for the data compression. the net effect is shown in Figure 3.4 (b) where the LL band is overall darker than in Figure 3.4 (a), and the other bands in Figure 3.4 (b) are lighter than in Figure 3.4 (a) is because that the maximum magnitude of the LL band is smaller.

For image compression, we usually recursively decompose the average subband LL until its size is reasonably small. Figure 3.5 shows the 3-level wavelet decomposition of "lena". Figure 3.6 (b) shows how we name the subbands in multi-level decomposition. The reason we use this decomposition is that after one step of the 2-D wavelet decomposition, the energy is usually compacted into the average sub-image (the LL band), and as we will describe later, this decomposition has a tree structured relationship among the coefficients. However, there are still other kinds of multi-level decomposition like *wavelet-packets* [12] [27] which will recursively decompose every subband, but its computational complexity is much higher, the encoding of the coefficients is also more complex, and the further compression which might be gained from this decomposition may not be very much.

3.3 Wavelet Image Compression

After wavelet decomposition, we get several sub-images (sub-bands). Their total size is still the same as the original, and except the LL_n sub-image, each of them only shows a specific frequency information from the original image. From this subband wavelet decomposition, energy-packing is achieved (see Figure 3.7) while the spatial domain (sometimes also called time domain) information is retained as much as possible. As the wavelet is applied to a whole image, the block-shaped distortion observed in JPEG compression is avoided. The energy of the subband transformed image is compacted to the average subband and the low frequency detail subbands. The high-frequency subbands contain mostly near-zero coefficients and a small fraction of high coefficients which usually correspond to edges. Compression can be achieved by allocating lower bitrates to the high-frequency bands while protecting the edge points.

Presently, many wavelet-based image compression schemes have been proposed.

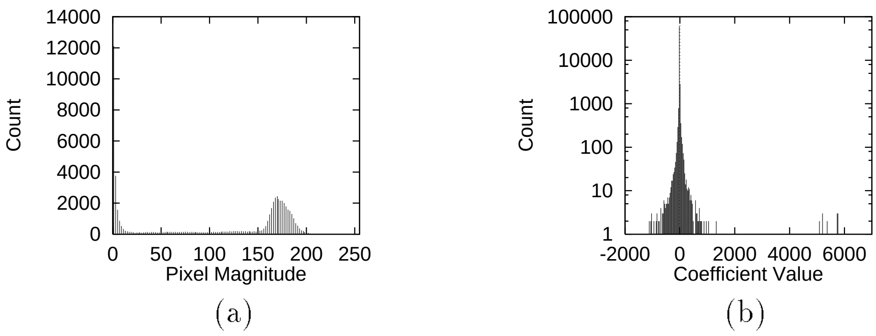


Figure 3.7: (a) Histogram of the original image lena (b) Histogram of its wavelet filtered image.

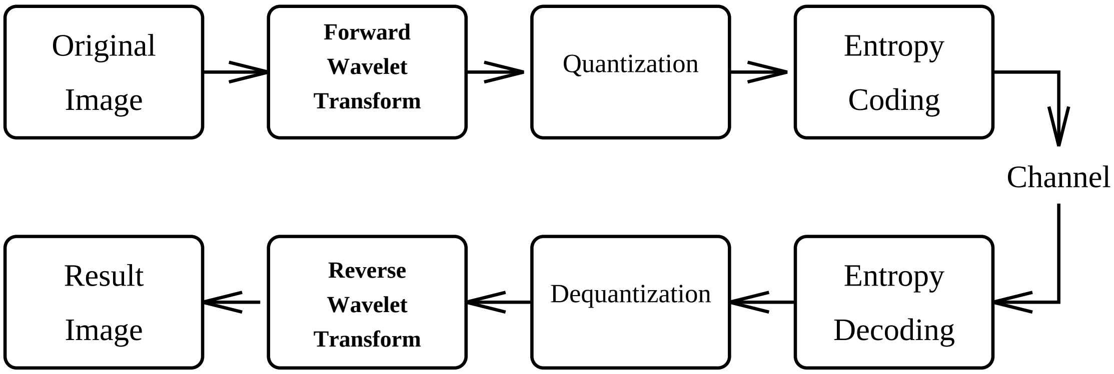


Figure 3.8: Diagram of wavelet-based image compression

Generally they first apply a wavelet decomposition to the image, and then many different kinds of method are used to quantize and encode the wavelet image, such as vector quantization[2] and tree encoding[20]. The general procedure of wavelet compression is shown in Figure 3.8.

Chapter 4

Embedded Zerotree Coding

In this chapter, we will introduce the *embedded zerotree wavelet algorithm* (EZW) [22]. EZW is an algorithm based on the wavelet subband decomposition. Its compression performance is enhanced by exploiting the self-similarity inherent in images, and entropy-coded successive-approximation quantization gives this algorithm the embedding ability. EZW produces compression results which are competitive with most current algorithms.

4.1 Zerotree for Wavelet

Zerotree is the fundamental concept for EZW image coding. This algorithm uses zerotrees to exploit the self-similarity of the wavelet subbands. Looking at the result of wavelet transform in Figures 3.4 and 3.5, we find two features:

- there are similarities between wavelet subbands; if there are edges at low frequency bands, usually there are similar structures at the higher frequency bands.
- the magnitude of a coefficient at a lower frequency band is usually higher than that of the coefficients of the higher frequency band at the same spatial position.

These features are considered to be redundant among the wavelet subbands. Based on the assumption that most images in-use today have the previous properties, the zerotree concept can be used to exploit these redundancies.

To clarify the zerotree concept, we first begin with the definition of *descendants*. For a coefficient at all subbands except LL_n and the highest frequency subbands, the set of four coefficients at the next finer scale of similar orientation which have the same spatial position are called *children* of this coefficient, and this coefficient is called the *parent* of these children. For a coefficient at the average subband LL_n, three coefficients of the same spatial position from LH_n, HL_n and HH_n respectively are called its children. The parent-child relationship is shown in Figure 4.1. For a given parent, its descendants are then defined to be its children and its children's descendants.

If a coefficient x and all its descendants are smaller than a given threshold value T, and if x's parent doesn't meet this condition or x doesn't have parents, x and all its descendants form a zerotree for threshold T, where x is the root of this zerotree.

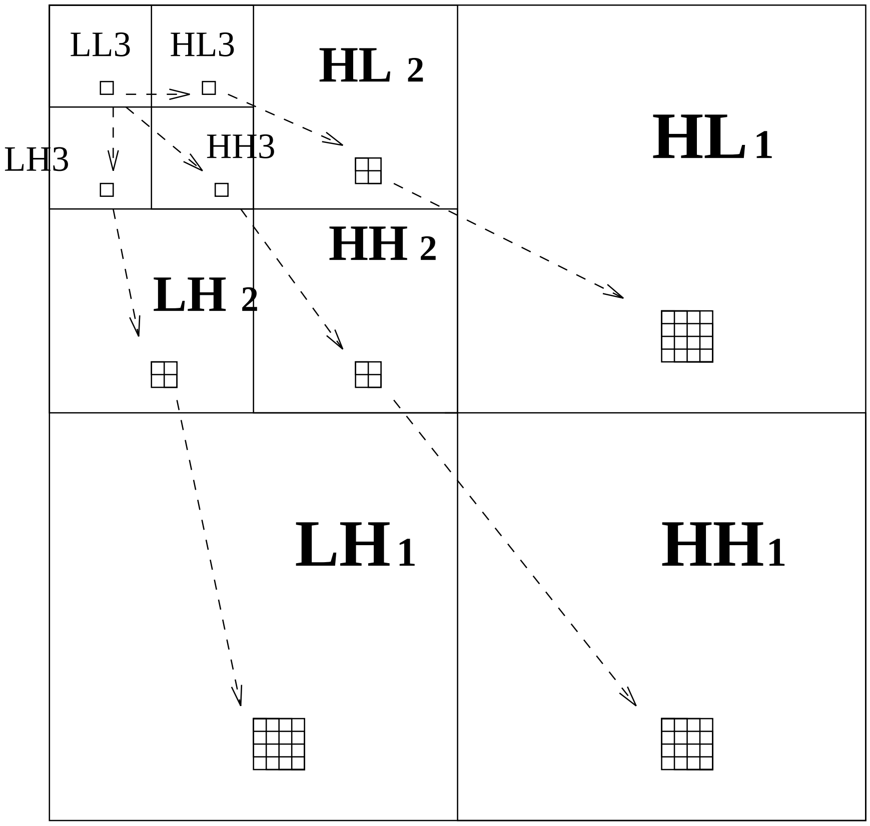


Figure 4.1: Parent-child relationship of wavelet coefficients. Arrows points from parents to the children. In this graph, a coefficient in the LL_3 band and all its descendants are plotted.

4.2 Algorithms

In general, zerotree encoding is a successive approximation method which is realized by first coding with a very coarse threshold or quantization step, then iterating by refining the threshold and redoing the coding. In each pass of the zerotree coding, the wavelet coefficients must be scanned through in such an order that a coefficient won't be scanned before its parent. In practice, we choose to scan subband by subband, first the average subband LL_n, then the detail subbands from coarse ones to the finer ones. This scan order also has an advantage that it is processing from the higher energy subbands to the lower energy one.

While scanning through a coefficient at the ith pass, a reference threshold T_i is used to determine its encoding as *positive significant*, *negative significant*, *zerotree root* or *separate zero* respectively according to the rules in Table 4.1:

Table 4.1: Zerotree Encoding Rules in Dominant Pass

Encode to be	Under the condition of
Positive Significant	$x \geq T_i$
Negative Significant	$x \leq -T_i$
Zerotree Root	x is a zerotree root for T_i
Separate Zero	x is neither significant coefficient nor zerotree root

The initial threshold T_0 is chosen to be larger or equal to one-half the magnitude of the largest coefficient; the threshold in subsequent iterations is given by $T_i = T_{i-1}/2$.

If x is a significant coefficient, its position and value is added to a list called the *significant list*, and x is marked as significant and set to zero, which requires that in the following passes, it won't be scanned. Setting x to zero will still allow its ancestors to form a zerotree in later iteration.

If x is a zerotree root, all its descendants are marked and won't be scanned and coded in this pass. In fact, this is the important part where the compression comes from, because a large block of data is now represented with only one symbol.

This scanning through of the coefficients is called the *dominant pass*. After each dominant pass, there is a *subordinate pass* in which the significant list is scanned through and its quantization refined by $T_i/2$. For each magnitude on the subordinate list, this refinement can be encoded using a binary alphabet with a *refine up* symbol indicating that the true value falls in the upper half of the old uncertainty interval defined with T_i, and a *refine down* symbol indicating that the true value falls in the lower half.

The string of symbols generated during the dominant and subordinate passes are all entropy coded. The entropy coder here is an adaptive arithmetic coder of Witten *et al* [26]. In adaptive arithmetic coding, the coder is separate from the model, which in [26], is in fact a learned occurrence probability of the symbols in the stream. In the practical coder used in this experiment, three different models are chosen for different situations because they have different symbols and probabilities. In the dominant pass, if the coefficient is in the finest detail subband (HL_1, LH_1 and HH_1), only 3 symbols are needed, because it cannot have a descendent, so there is no difference between zerotree root and separate zero; it uses a different model from the other coefficients because it can't have descendants. The subordinate pass also has its own model, with only 2 symbols.

The encoding process continues to alternate between dominant passes and subordinate passes where the threshold is halved before each dominant pass.

In the decoding operation, from each decoded symbol, we may decide or refine an uncertainty interval in which the original value of one coefficient or a set of coefficients in a zerotree may lie. Usually, we choose to set the decode value to be the center of the uncertainty interval to reduce the mean square error.

The process of encoding and decoding gradually refines the precision of the transform coefficients in order of their magnitudes. The encoding or decoding may be terminated at any point, and the coded bit stream is a prefix of all lower-rate encodings; this is referred to as *embedded coding.* Thus, compression is achieved by terminating the transmission or storage of the embedded code at some point in the bit stream, and the exact bit rate is controlled by choosing the point at which this termination takes place. Embedded coding schemes naturally suit a progressive mode of transmission of the image; at any time, images reconstructed from the decoded bit stream can be displayed as increasingly good approximations of the original image.

4.3 Implementation

4.3.1 WiT visual dataflow image processing environment

We used WiT ™, a visual programming package for designing complex imaging algorithms in areas of image processing [4, 5] as our development platform. In WiT, the programs are like a graph, called *igraphs.* An igraph is a set of *operators* connected by *links.* Operators do the processing functions on data objects, which flow along the links from operator to operator. Standard image-processing functions are realized in WiT as pre-installed operators. Figure 4.2 is a simple igraph. The operators are

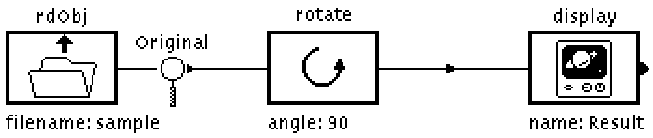


Figure 4.2: Example of a WiT igraph.

shown as icons. They are connected by links with a small arrow on them showing the dataflow direction (usually from left to right). Each operator's name is displayed above its icon; its parameters and their values are shown below. In this igraph the rdObj operator reads an image object from a WiT file called "sample"; this object contains the images size, type and data; the rotate operator with its parameter set to be 90 will rotate the image by 90 degrees, and the display operator will display the resulting image. A probe labeled original causes the raw image to be displayed as it passes along the first link.

4.3.2 Image Padding for Wavelet and Zerotree

It may be noticed that both wavelet transform and zerotree representation presume that the image width and hight is divisible by some power of 2. For a n step dyadic wavelet transform (Section 3.2.4), the original image size must be divisible by 2^n. In order to make the algorithm applicable to any sized image, image padding is necessary. In this implementation, the image width and height are padded to be divisible by 8 for images of which either the width and height is less than 256; the image width and height are padded to be divisible by 16 for images of which both the width and height are larger than 256. The padded pixels are set to be the average of its neighbour.

Table 4.2: The fastWav operator.

Inputs:	image	Image for transform.
Parameters:	none	
Outputs:	wavelet features	wavelet image original image size, and the mean value

Table 4.3: The fastRevWav operator.

Inputs:	wavelet features	wavelet image original image size, and the mean value
Parameters:	none	
Outputs:	wavelet	reconstructed image

Table 4.4: The fastZTcompress operator.

Inputs:	wavelet features bytepp	wavelet image to encode original image size, and the mean value original image's pixel size in bytes
Parameters:	max_bytes	byte budget for the encoding
Outputs:	out_stream bytes_coded	encoded stream size of the stream

Table 4.5: The fastZTexpand operator.

Inputs:	in_stream	encoded stream
Parameters:	max_bytes	maximum decode bytes
Outputs:	wavelet features bytepp bytes_decoded	decoded wavelet image original image size, and the mean value original image's pixel size in bytes number of bytes actually decoded

4.3.3 WiT Implementation

The embedded zerotree image coder and decoder are implemented in WiT on Sun workstations (the code has also been ported to Microsoft Visual C++ on PC). Two WiT operators fastWav and fastRevWav are implemented to perform multi-level forward and reverse two-dimensional wavelet transform. The wavelet transform core of these operators is from Said *et al* [19]. fastZTcompress and fastZTexpand are implemented for the zerotree coding and decoding of the wavelet coefficients. The encoded bit stream of fastZTcompress is output as a string and could be written to and read from a disk file or it could be passed directly to the fastZTexpand operator as its input. The specification of these operators are summarized in Table 4.2, 4.3, 4.4 and 4.5. Note there are input or output of features in those operators; these features include overhead values for the decoder:

- original image size. In order to apply the wavelet transform on to image of any size, we may have to pad the image, so the original image size is needed to unpad the recovered image.
- mean value of wavelet image. In order to get as many "zero" value coefficients as possible, the mean value of the wavelet image is subtracted before the forward wavelet transform. This value is needed after the reverse transform.

Figure 4.3 is an igraph we used to test the operators and animate the EZW coding and decoding process. In this igraph, the image lena is read and changed into floating point format which is acceptable by the wavelet transform operator. WiT automatically gets its dimension 512×512. The image is then sent to the fastWav

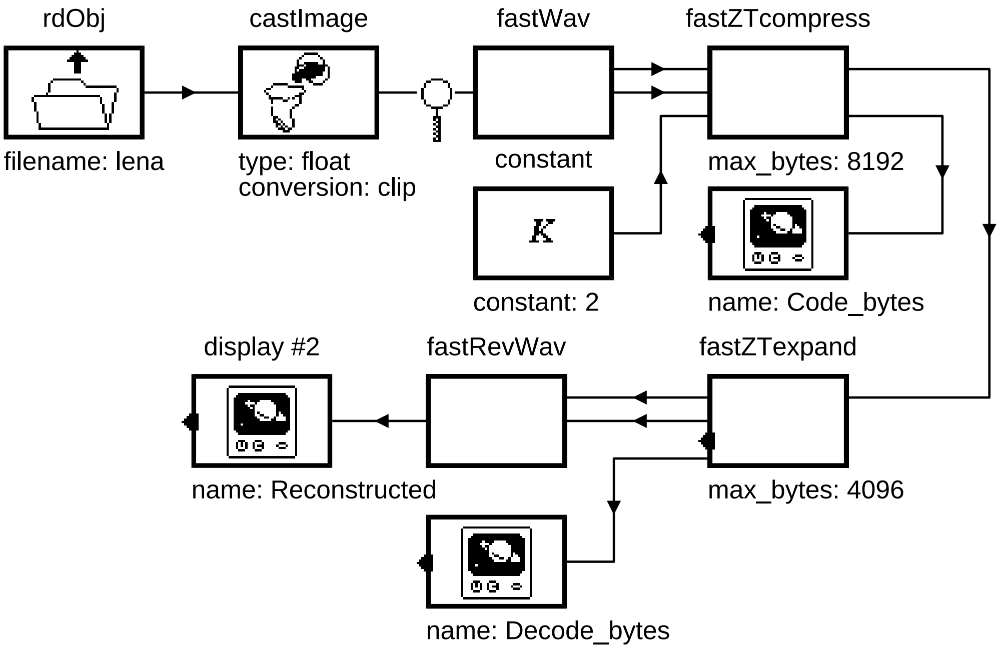


Figure 4.3: Igraph of EZW codec

operator which makes the forward wavelet transform. The resulting wavelet image and additional features are then sent to the zerotree coder fastZTcompress. The coder will generate a 8192 bytes stream as specified by the max_byte parameter which is a 0.25bpp compression of the original 8 bit image; this byte stream is sent to the decoder fastZTexpand, the decoder here will only decode the first 4096 bytes to get the additional features and the reconstructed wavelet image. The number of bytes to decode could also be any value smaller or equal to 8192. Finally the reverse wavelet transform is applied and a 0.125bpp reconstruction of the original image is shown in a window generated by the display operator.

4.4 Results

The EZW algorithm has been tested on standard images 'lena' and 'goldhill' and ultrasound images 'US1' and 'US2'. 'lena' and 'goldhill' are 512×512 8-bit greyscale images as shown in Figure 4.4 (a) and Figure 4.5 (a); 'US1' and 'US2' are 640×480 8-bit greyscale images as shown in Figure 4.6 (a) and Figure 4.7 (a). Each image has been compressed at 1.0, 0.75, 0.6, 0.5, 0.4, 0.3, 0.25, 0.20, 0.15 and 0.10 bpp. Figures 4.4, 4.5, 4.6 and 4.7 show some of the reconstructed images. The performance measured by PSNR are given in Tables 4.6 and 4.7; the performance of the JPEG standard algorithm are also presented in the tables as a reference (As JPEG can't control the compression ratio accurately, the results of JPEG here are all that of compression ratios lower than and close to the required ones). Figure 4.8 and 4.9 are plotted PSNR versus compression-ratio graph using data from Tables 4.6 and 4.7.

(a)

(b)

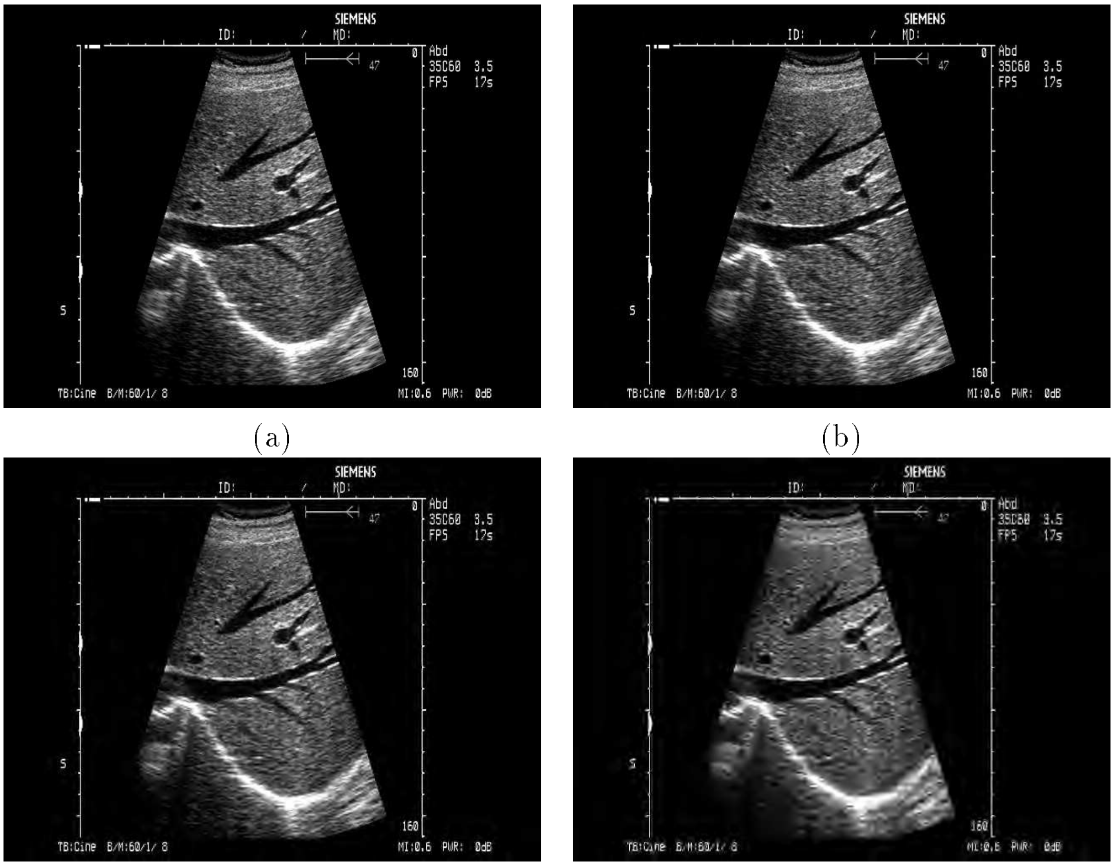
(c)

(d)

Figure 4.4: Result images of EZW on 'lena', (a) Original image, 512×512 8-bit greyscale; (b)Reconstruction of 1.0 bpp, 8:1 compression, PSNR=39.31dB; (c)Reconstruction of 0.25 bpp, 32:1 compression, PSNR=33.15dB; (d)Reconstruction of 0.10 bpp, 80:1 compression, PSNR=29.50dB

(a) (b) (c) (d)

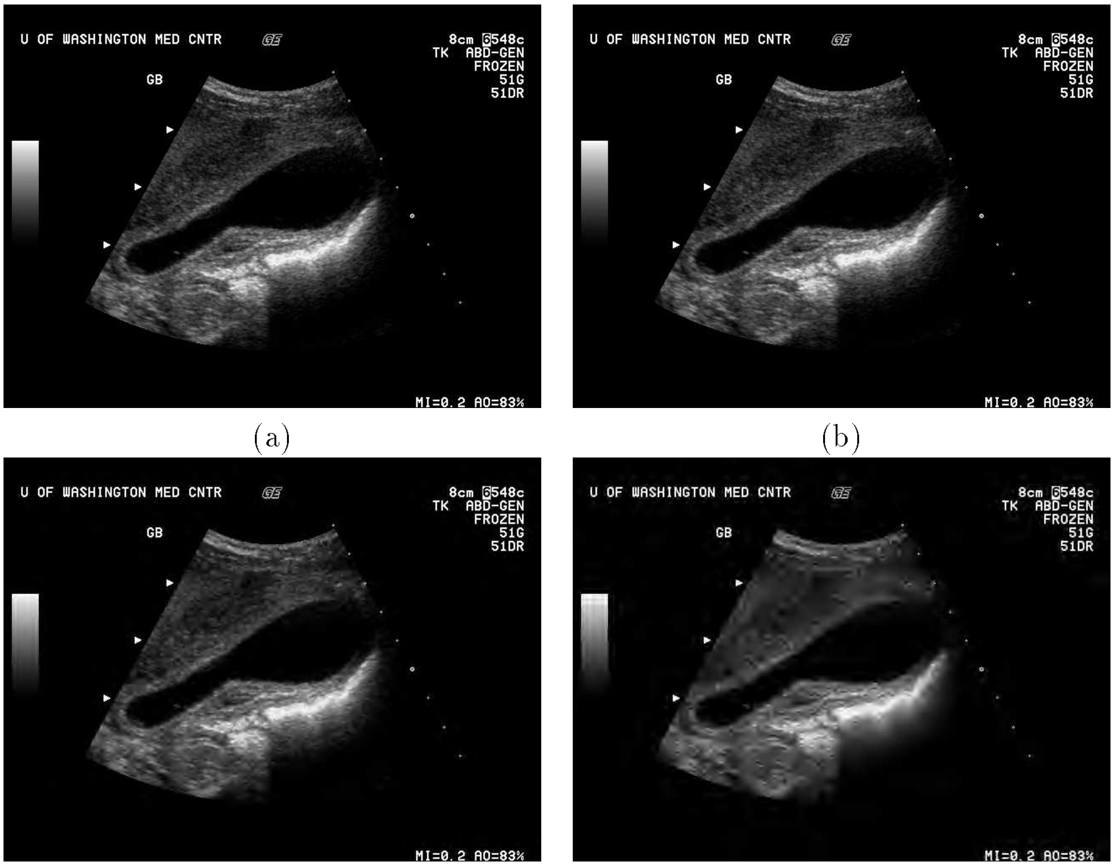
Figure 4.5: Result images of EZW on 'Goldhill', (a) Original image, 512×512 8-bit greyscale; (b)Reconstruction of 1.0 bpp, 8:1 compression, PSNR=35.44dB; (c)Reconstruction of 0.25 bpp, 32:1 compression, PSNR=30.09dB; (d)Reconstruction of 0.10 bpp, 80:1 compression, PSNR=27.71dB



(a) (b)

(c) (d)

Figure 4.6: Result images of EZW on 'US1', (a) Original image, 640×480 8-bit greyscale; (b)Reconstruction of 1.0 bpp, 8:1 compression, PSNR=41.72dB; (c)Reconstruction of 0.25 bpp, 32:1 compression, PSNR=28.78dB; (d)Reconstruction of 0.10 bpp, 80:1 compression, PSNR=24.12dB



(a) (b)

(c) (d)

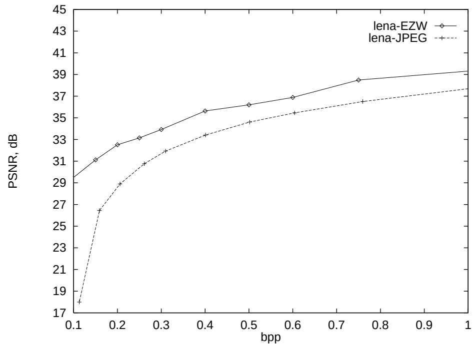
Figure 4.7: Result images of EZW on 'US2', (a) Original image, 640×480 8-bit greyscale; (b)Reconstruction of 1.0 bpp, 8:1 compression, PSNR=48.21dB; (c)Reconstruction of 0.25 bpp, 32:1 compression, PSNR=34.08dB; (d)Reconstruction of 0.10 bpp, 80:1 compression, PSNR=28.75dB

Table 4.6: Test PSNR of EZW on 'lena' and 'goldhill' in comparison with JPEG.

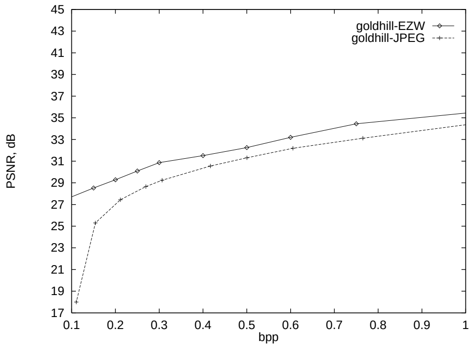
compression ratio	bit rate (bpp)	EZW on lena PSNR (dB)	JPEG on lena PSNR (dB)	EZW on goldhill PSNR (dB)	JPEG on goldhill PSNR (dB)
8:1	1.00	39.31	37.71	35.44	34.40
10.67:1	0.75	38.49	36.50	34.45	33.11
13.33:1	0.60	36.88	35.46	33.20	32.19
16:1	0.50	36.20	34.61	32.25	31.31
20:1	0.40	35.63	33.41	31.51	30.55
26.67:1	0.30	33.92	31.93	30.86	29.24
32:1	0.25	33.15	30.76	30.09	28.65
40:1	0.20	32.51	28.89	29.28	27.44
53.33:1	0.15	31.12	26.44	28.52	25.30
80:1	0.10	29.50	18.00	27.71	18.00

Table 4.7: Test PSNR of EZW on 'US1' and 'US2' in comparison with JPEG.

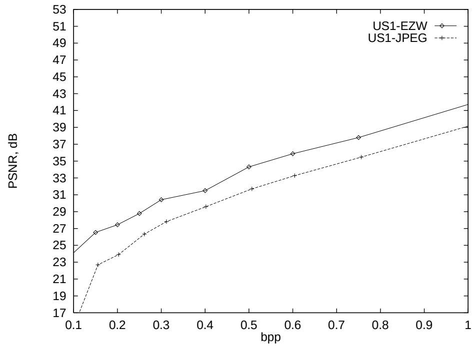
compression ratio	bit rate (bpp)	EZW on US1 PSNR (dB)	JPEG on US1 PSNR (dB)	EZW on US2 PSNR (dB)	JPEG on US2 PSNR (dB)
8:1	1.00	41.72	39.42	48.21	46.06
10.67:1	0.75	37.79	35.47	44.11	41.96
13.33:1	0.60	35.87	33.27	42.42	39.02
16:1	0.50	34.31	31.70	39.88	36.62
20:1	0.40	31.49	29.58	37.83	34.04
26.67:1	0.30	30.41	27.82	35.58	30.92
32:1	0.25	28.78	26.33	34.08	29.78
40:1	0.20	27.44	23.91	33.36	27.81
53.33:1	0.15	26.54	22.69	30.63	25.39
80:1	0.10	24.12	16.83	28.75	17.25



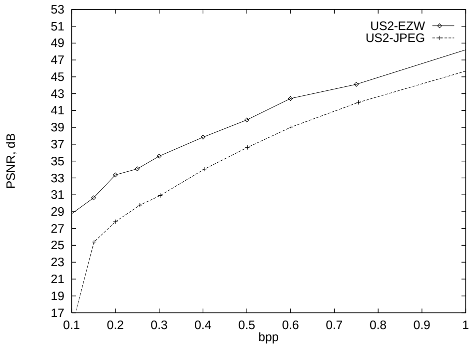
(a)



(b)



(c)



(d)

Figure 4.8: Comparison of PSNR of EZW with JPEG. (a) lena; (b) goldhill; (c) US1; (d) US2

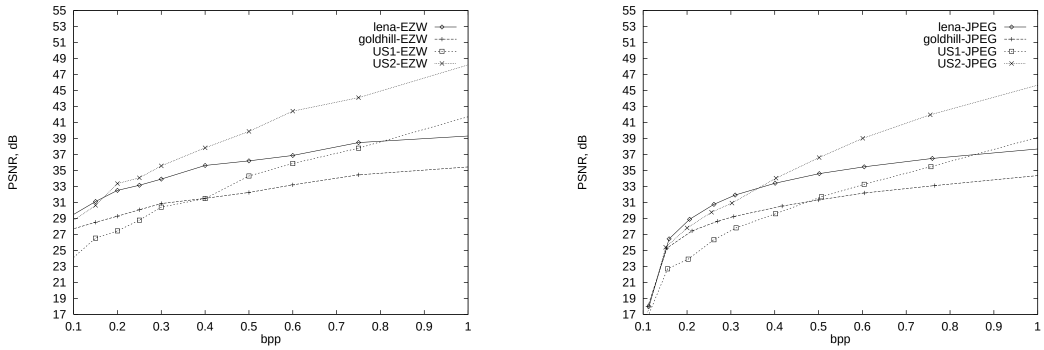


Figure 4.9: Comparison of PSNR of lena, goldhill US1 and US2. (a) EZW (b) JPEG

(a)

(b)

Figure 4.10: EZW and JPEG images at same PSNR, (a) EZW compression of 'goldhill', 0.093 bpp, 27.44dB; (b) JPEG compression of 'goldhill', 0.23 bpp, same PSNR

4.5 Discussion

Qualitatively, Figures 4.4 – 4.7 show how the images degrade as the bitrate goes down. We find that for the bitrate of 1 bpp, the image quality is always very good. At the medium bitrate of 0.25 bpp, the image qualities are fairly good but some loss of details; at low 0.1 bpp, the image quality degradation is huge, many details and textures have lost.

From Figure 4.8 we find that measured by PSNR, the performance of EZW is always better than JPEG. When at higher bitrate (higher than 0.2 bpp) the difference is about 1dB to 3dB depending on images. At lower bitrates, the performance of JPEG deteriorates much faster than EZW, and there is an almost 10 dB difference between these two methods. This proves that JPEG performs worse than EZW at any bitrate, and that it performs very poorly at low bitrates. With the decrease of bitrate, the reconstruction image of both algorithms will decay, but they decay in a different manner. Figure 4.10 shows the 27.40 dB reconstructions of both EZW and JPEG, where, the JPEG image is like a mosaic of the original image while the EZW image is like a blurring of the original image.

Another interesting result is noticed in Figure 4.9; measured by PSNR the ultrasound images degrade faster than the standard images. For example, at 1.0 bpp, US1's image quality is better than both "lena" and "goldhill" with either EZW or JPEG compression; however, at bitrate lower than 0.4 bpp, US1's image quality becomes worse than both of them! This suggests that the ultrasound images have their own properties such as the large black areas with white text, finding some special treatment for them may improve the performance.

Chapter 5

Dynamic Region-based Wavelet Compression

5.1 ROI Partition for Region Based Coding

We have expanded the EZW algorithm to a region based *dynamic embedded wavelet coding* (DEW).

It is very natural that different regions of an image may have different importance. Therefore an intuitive way to gain more compression is to partition the important regions from the unimportant ones and code them with appropriate accuracy. A region-based coding has to do the following extra work:

- Region partition

- Region encoding
- Bitrate allocation

The region partition is a binary mask function M over the image pixels (i, j):

$$M(i,j) = \begin{cases} 1 & if\ (i,j) \in\ Region\ of\ Interest \\ 0 & otherwise \end{cases}$$

Region encoding means representing this partition with fewer bits to send to the decoder. Bitrate allocation is the process of applying different quantization bit-rates to the different regions, according to $M(i, j)$.

5.1.1 Transform-domain Partition

A region could be incorporated into a spatial domain coding algorithm which scans through each pixel one by one and applies a mask function M to each pixel, then a quantizer of different bitrate is chosen according to $M(i, j)$. For an image coding algorithm based on small block transformations or spatial domain vector quantization, a region-based coding region-based feature could also be added in a similar way by using a partition mask function over blocks rather than over the single pixels.

For subband image codings such as wavelet coding, the information of each pixel is now stored in different subbands and has spread to its neighbour, so it is impossible to specify a quantization rate for a pixel. One way is to physically segment the image

$f(i,j)$ into two images $f_{in}(i,j)$ and $f_{out}(i,j)$ where

$$f_{in}(i,j) = \begin{cases} f(i,j) & M(i,j) = 1 \\ 0 & M(i,j) = 0 \end{cases}$$

and

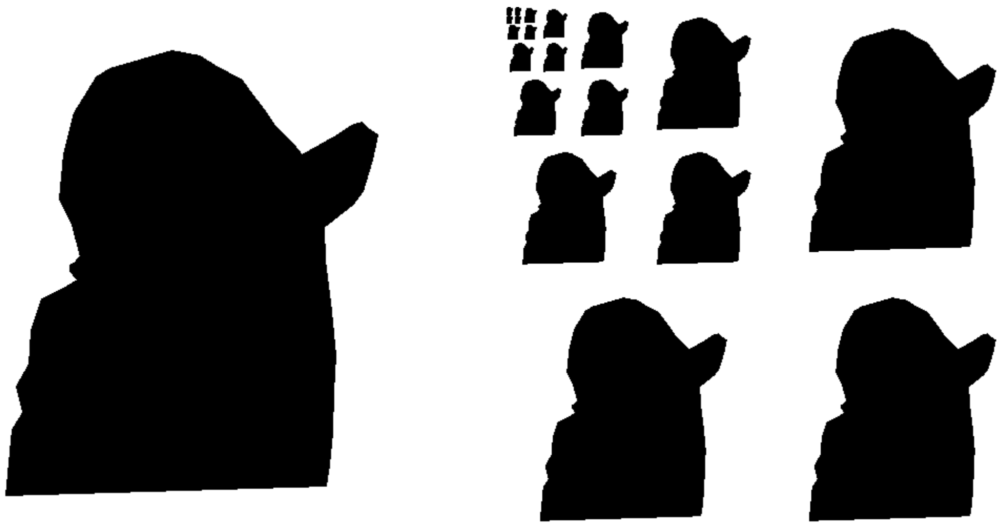
$$f_{out}(i,j) = \begin{cases} 0 & M(i,j) = 1 \\ f(i,j) & M(i,j) = 0 \end{cases}$$

then coding $f_{in}(i,j)$ and $f_{out}(i,j)$ with appropriate compression ratios. At the decoder side, the reconstructed images of $f_{in}(i,j)$ and $f_{out}(i,j)$ are added together. However, this method has some obvious drawbacks:

1. the number of images to be coded is now two, which is inefficient;

2. this segmentation will introduce artifacts at the edge of the mask, because after the segmentation, artificial sharp edges are introduced at the edges in both $f_{in}(i,j)$ and $f_{out}(i,j)$, and these sharp edges are hard to be perfectly coded, so artifacts are formed after the two images are merged together. This effect was observed in [1].

This leads to the idea that the region partition will have to be done in the transform domain. As the wavelet transform coefficients still have spatial localization, the spatial mask over the original image could be adapted to a mask over the transformed image. As the wavelet transformation represents an image with a group of subband images, a reasonable mask transform is to down sample the spatial mask to fit in each subband of the wavelet coefficient. Figure 5.1 shows an original mask and its transform domain counter-part.



(a) (b)

Figure 5.1: Mask transformation from Space to Wavelet Domain. (a) ROI Mask in space domain; (b) its transformation in wavelet domain

However, in a wavelet transform, besides "splitting" the energy of one point into a group of subbands, the energy is also "spread" into its neighbor points because of the convolution operation of the wavelet transform. The straightforward way of solving this is to enlarge the mask by a proper degree. Tests have shown that enlarging the spatial mask usually gets poor partition performance, even if very wide enlargements were applied to the mask. This is because the size of this spread neighborhood is dependent on the wavelet filter and the frequency of information. As the low frequency energy spreads much wider than the high frequency one, we couldn't just enlarge the mask at fixed width in the spatial domain. We found that enlarging the mask in the transform domain gets good results, because it gives different subbands the least necessary enlargement, this will be discussed in more detail in section 5.3.

5.2 DRW – Dynamic Region-based Wavelet Coding

As shown in Chapter 4, EZW is an embedded coding algorithm, which means it could progressively render an image at the decoder side. This embedding feature is very useful and it is desirable to keep this feature while adding the region-based ability. The result is our *dynamic region-based wavelet* (DRW) coding. Dynamic here means that the codec process could operate dynamically, in that the original image is encoded in such a way that the decoder could reconstruct any length of the coded bitstream and get a reconstructed image utilizing almost every bit (there might be a few bits which couldn't be reconstructed, because of the arithmetic coding). Furthermore, at a different period, a different coding mask could be set in the coder to force it to only encode information in the region specified by the mask.

The DRW algorithm is a very flexible algorithm, and it is easy for it to realize the functionality of region-based coding. For example, an image is assigned a *Region of interest* (ROI) which we wish to code with a higher fidelity requirement than the out of region part (we call it non-ROI). Using our algorithm, first the coding mask is set to be the whole image, and the image is coded and transmitted until the required fidelity of non-ROI is reached. Then the mask is reduced to the ROI, and the image is coded and transmitted until the fidelity requirement under the coding mask is fulfilled.

In fact, this algorithm is so flexible that different schemes could be used to encode an image and get the same final decoded image. In the previous example, the compression could also be realized by first setting the coding mask to the ROI and coding until the requirement for ROI is reached, and then setting the mask to non-ROI and coding until the requirement for non-ROI is reached.

The differences between the schemes may lead to a small difference of the bitstream length due to the fact that different schemes may lead to different zerotree coding order, different region-coding overheads and different performance of the adaptive entropy coder. However, tests have shown that although this difference is usually very small, encoding the whole image and then coding the important regions always gets a better result, because fewer iterations of the coding pass are needed.

In the situation that the reconstructed image is only displayed after all the data is received and decoded, the differences between the schemes don't show up. However, in the situation that the decoded image will be displayed on the fly, a progressive rendering of the image will take place, and this ability to choose different schemes may have merit.

Originally, a progressive rendering of the image produces a blurred image which contains the main part of energy of the original image, and then additional details which are related to less energy will be rendered to refine the previous result. We could call this rendering based on energy. In our new algorithm, we could change the scheme to first code the important region or ROI, then code the non-important region, and at the decoder side we get progressive transmission of the image which is not just based on the energy, but also based on the regional "importance" of the information.

Another merit of this dynamic algorithm is as we mentioned in Section 1.1.2, while this algorithm is used in image transmission over narrow band networks, the decode side could choose a ROI after looking at the gross reconstruction from the first short part of the transmission, then the coder could change its mask to only code the ROI which could save a lot of transmitting time.

5.2.1 Algorithms

Motivation

Before giving the details of the algorithm, we would like to discuss the motivation of this algorithm we get from the EZW.

In the embedded zerotree wavelet (EZW) coding, the most critical concept is *zero tree* and *successive approximation.*

Successive approximation is realized by first coding with a very coarse threshold, then iterating by halving the threshold and redoing the coding. The fidelity is then revised iteration after iteration. Looking at the successive approximation process of the EZW algorithm, we may regard the threshold as a 'fidelity requirement'. Different thresholds mean different fidelity requirements for the reconstructed image. Region-based image compression may be described as a compression which has different fidelity requirements for different regions. In the conventional EZW algorithm, the threshold is unique at every iteration, in other words, there is one fidelity requirement for the whole image. We conclude that, if we could specify different thresholds for different regions, the EZW coding could be upgraded to a region-based algorithm.

In the DRW method, for each coefficient in the wavelet domain, a unique threshold

$T_{i,j}$ is assigned, here, (i, j) is the address of the coefficient in the wavelet transformed images as in Figure 3.6. These initial values are all set to be half the maximum magnitude. There is also a mask image $M_{i,j}$ over the wavelet transformed image, which is initialized to 1 which means it is in the coding mask. The coding operates by scanning through the coefficients with the same order as in EZW. The coding has two main operations, *mask encoding* and *data encoding* .

Mask encoding

In the beginning of each scanning pass of the image, one bit is used to specify if a different mask from the previous (or initial) mask was selected, if not, a new pass of data encoding will continue with the previous (or initial) mask.

If a new mask is selected, it needs to be encoded. The mask could be any shape. Currently we support rectangle and polygon shaped masks. A rectangle is encoded as its upper-left and bottom-right corner; a polygon is represented by all its vertices. The mask is converted into the transform domain using the method mentioned in section 5.1.1, and put into $M_{i,j}$ giving a result as in Figure 5.1. Then the mask needs to be enlarged (or dilated) in the transform domain because of the filtering. A test in Section 5.3 has shown that for the wavelet filter we use (a 9/7-tap filter of [2]), enlargement of 2 pixels is enough for a good partition result.

After the new mask image M is generated, we set the maximum threshold of the in-mask coefficients t_{max} by:

$$t_{max} = max\{T_{i,j} | M_{i,j} = 1\}$$

t_{max} is used in the data encoding procedure to restrict the encoding only of the coefficients which have the coarsest quantization step.

Data encoding

The data encoding procedure is similar to a coding pass of EZW coding. In the dominant pass only a coefficient which is in the mask and whose threshold $T_{i,j}$ is equal to t_{max} will be scanned and encoded to *significant positive*, *significant negative*, *zerotree root* or *separate zero* respectively according to the rules in Table 4.1 , where now we use t_{max} instead of T_i. Operations of moving a significant coefficient into the significant list and skipping the points which are in a zerotree will also be done in the same way as in EZW coding. The things needed to be considered here are:

1. Sometimes, we may have case that when coding an in-mask coefficient, some of its descendents might be outside the mask. These coefficients are taken as zero to increase the possibility of forming zerotrees.

2. if one coefficient is encoded, its related threshold $T_{i,j}$ needs to be halved.

3. if a coefficient is encoded to be a zerotree root, its descendents' thresholds also need to be halved, but only for those which are in the mask and are not points in the significant list. This is because the out-of-mask coefficients are just taken as zero, not really coded and points in the significant list will be dealt with only in the subordinate pass.

After the dominant pass, the t_{max} is halved, and the subordinate pass will process the points in the significant list. The difference from the subordinate pass in EZW coding is that only the points which are in the mask and have threshold $T_{i,j} \geq t_{max}$ will be processed and coded to generate *refine up* or *refine down* symbols indicating that the trues value fall in the upper or lower half of the old uncertainty interval whose size is defined by $T_{i,j}$.

The process continues to alternate between mask encoding and data encoding, where mask could be changed at every mask encoding process. Note that if the receiver wants to change the encoding mask, another channel must be established from the receiver to the user, or a two-way channel is needed.

5.2.2 Implementation

The DRW coding has been implemented on WiT for algorithm prototyping research, and a Microsoft Visual C++ class is created which can be used for dynamic region-based PC image transmission.

WiT Implementation

Two WiT operators dynamicZTcompress and dynamicZTexpand are implemented for the dynamic region-based zerotree coding and decoding of the wavelet coefficients. The specification of the operators are summarized in Table 5.1 and 5.2.

The DRW encoding operator dynamicZTcompress has one more input, RegionOfInterest, than the EZW encoding operator fastZTcompress. RegionOfInterest specifies the encoding region; the other inputs are the same. dynamicZTcompress also has two more parameters than fastZTcompress, start_bytes and progress_speed which are used

Table 5.1: The dynamicZTcompress operator.

Inputs:	wavelet	wavelet image to encode
	features	original image size, and the mean value
	RegionOfInterest	ROI to encode on
	bytepp	original image's pixel size in bytes
Parameters:	max_bytes	maximum byte budget for the encoding
	start_bytes	bytes budget to coded the first frame of data
	progress_speed	animation speed of progressive transmission
Outputs:	out_stream	encoded stream
	bytes_coded	size of the stream

Table 5.2: The dynamicZTexpand operator.

Inputs:	in_stream	encoded stream
Parameters:	No	
Outputs:	wavelet	decoded wavelet image
	features	original image size, and the mean value
	bytepp	original image's pixel size in bytes
	bytes_decoded	number of bytes actually decoded

to set points to stop encoding. Each time encoding is stopped, the current result of progressive transmission could be seen at the receiver side, and the ROI could be changed at this time. start_bytes orders the encoder to prepare to stop the first time after the required bytes are coded; then progress_speed r will order the encoder to stop after r round of scans over the original image, so progress_speed must be an integer greater or equal to 1, and for a good animation result, it should between 1 and 5. Another special feature of the dynamicZTcompress operator is that it operates on the "firing on any" strategy rather than "firing on all". "firing on all" means that the operator will start processing only after all the inputs have arrived; "firing on any" means that the operator could start processing for any input (and might generate output). See [3] for more detail of firing strategies. Here, this dynamicZTcompress operator will start compression after the three inputs wavelet, features and bytepp arrived. It will stop compression and output the current compressed stream after it reached start_bytes and finished a full coding pass. The successive compression of dynamicZTcompress operator will be triggered by a single input RegionOfInterest, noting that if the input is a null object, i.e. the user didn't select any region, the ROI will be unchanged from the previous pass.

Figure 5.2 is an igraph used to test the operators and animate the process of DRW coding and decoding. The test image "lena" is read from a wit image file by rdObj; then it is casted into a floating point format and a wavelet transform is applied. The resulting wavelet image and additional features are then sent to the operator dynamicZTcompress. The coded bitstream output from dynamicZTcompress is directed to dynamicZTexpand which decodes the stream and recovers the wavelet image and the additional features which are used by fastRevWav to reconstruct original image. Note on the igraph that the features output is directed to an icon with two black dots, which is a repeat operator which will continuously send copies of the object it received. This is needed because the additional features are only encoded and decoded in the first codec round but are needed every time the reverse wavelet transform is called. The reconstructed image from fastRevWav is sent to a getData operator which displays the image and lets the user define graphic objects like lines, circles and outputs their graphic vector description. Here the user could choose rectangle and/or polygon regions which are acceptable to the dynamicZTcompress operator. The dynamicZTcompress will be active again after it receives the ROI description from getData, and the cycle repeats. If the user does not want to change the ROI, a null region can be selected or the whole animation could be stopped.

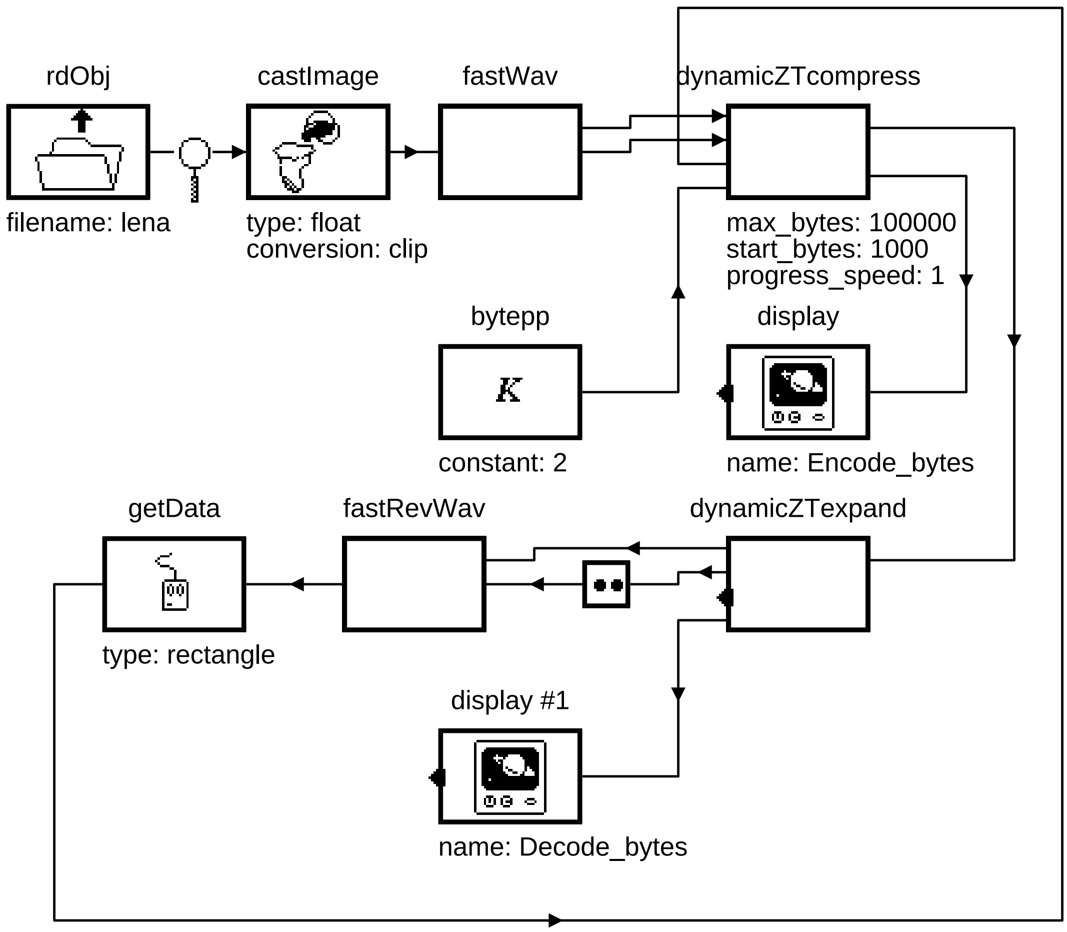


Figure 5.2: igraph of DRW codec testing

Visual C++ Class Implementation

This DRW algorithm has been ported to Microsoft Visual C++ on a pentium PC running Windows NT 3.5. The coder and the decoder is realized as a C++ class which is easy to incorporate into MS-Windows applications. The class interface is given below:

```
class CWavelet : public CObject
{
public:
/* construction and deconstruction function */
CWavelet();
~CWavelet();

    /* encoder function */
ERRORS StartCoder(HDIB & originalImage);
ERRORS SetRegion(RECT & regionOfInterest);
HGLOBAL GetData(long requireBytes, long *targetSize);

    /* decoder function */
ERRORS StartDecoder(HGLOBAL bitStream);
ERRORS PutData(HGLOBAL bitStream);
ERRORS Recover(HDIB & recoverImage);
};
```

"CWavelet" and "~Cwavelet" are the construction and deconstruction functions. At the encoder side, "StartCoder" is called to set the image to be coded, then "SetRegion" called to set the ROI and then "GetData" is called to start coding and return the required bytes of the coded stream. As the actual coded stream may be larger than the required size, "*targetSize" returns the coded stream size. At the decoder side, "StartDecoder" is called to decode the first piece of the coded stream it received, then "PutData" called to decode the successive streams. Finally, "recover" is called to output the reconstructed image using the most recently received data.

This class is being utilized into a teleconference software package to add progressive and region-based features. [1]

5.3 Tests and Discussion

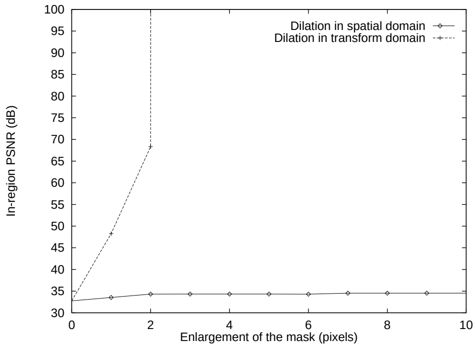
One test is done to show the advantage of enlarging the mask in transform domain rather than spatial domain in the process of region partition, one speed test is given to show DRW's speed performance, and three dynamic transmission tests on 'lena', 'US1' and 'MR1' are conducted to show the PSNR performance of the DRW coding algorithm.

5.3.1 Regional PSNR

To measure the fidelity of region-based image compression, the regional PSNR with respect to a mask M is needed. In the following test, we use $PSNR_M$ to denote the

[1]This is a technology transfer project supported by the British Columbia Science Council.

(a)



(b)

Figure 5.3: Comparison between enlarging in the spatial domain and the transform domain

PSNR of the part of image in M:

$$PSNR_M = 10 \log_{10} \frac{f_{\max}^2}{\frac{\sum_{i,j \in M} (\hat{f}_{i,j} - f_{i,j})^2}{S(M)}} \tag{5.1}$$

Here $f_{i,j}$ and $\hat{f}_{i,j}$ are original and reconstructed images, $M_{i,j}$ is the mask function and $S(M)$ are the number of coefficients in the mask M.

5.3.2 Region Partition Test

Figure 5.3 shows a comparison between enlarging in the spatial domain and enlarging in the transform domain. The mask is as selected in Figure 5.3 (a). For each method, the recovered image's in-region PSNR is plotted in Figure 5.3 after applying the mask to partition out non-ROI data. We found that if we enlarge the mask in the

Table 5.3: Speed of DRW and EZW on lena, in DRW the coding mask is fixed to the whole image, the times given are the sum of encoding and decoding time

compression	bpp	PSNR	time EZW (seconds)	time DRW (seconds)
16:1	0.5	36.20	11.68	13.14
32:1	0.25	33.15	9.97	11.13
64:1	0.125	30.25	8.52	10.07

spatial domain and then transform it to the transform domain, the performance won't increase much even if we choose an enlargement of 10 pixels. On the other hand, if we enlarge the mask in the transform domain, the partition performance is good enough using just 2 pixels (65dB). In fact, for dilation larger than 2 pixels the partion won't cause any distortion, as it includes all the in-ROI information. This test is based on the 9/7-tap filter [2] we use; for different filters, the results may be different, but the transform domain enlarging is always much better than the spatial domain enlarging.

5.3.3 Speed Test

To test the speed of DRW in comparison with the EZW algorithm, the mask is fixed to the whole image, so DRW will provide exactly the same compression result as EZW. Table 5.3 compares timing results on 'lena' for the two algorithms at different bitrates, where in DRW the mask is fixed to whole image. It is shown that to provide the same compression ratio and PSNR the speed overhead of DRW in comparison with EZW is acceptable.

Table 5.4: Test result on lena – dynamic rendering, ROI is shown on Figure 5.4 (b)

Result Figure	bytes	bpp	PSNR	PSNR in ROI
Fig 5.4 (a)	1314	0.04	26.22	25.62
Fig 5.4 (b)	3645	0.11	26.63	36.91
Fig 5.4 (c)	3645	0.11	29.56	29.13
Fig 5.4 (d)	29056	0.89	37.31	37.04

5.3.4 Dynamic Transmission Tests on 'lena'

Figure 5.4 shows an example of dynamic transmission of the lena image. Table 5.4 lists the PSNR performance of the example of dynamic transmission in Figure 5.4. Figure 5.4(a) shows a 0.04 bpp compression of the whole image; Figure 5.4(b) and Figure 5.4(c) are both 0.11 bpp compression. In this case the area of the ROI is 7.48% of the whole image area. Observe in Figure 5.4(c) how poor the selected region looks compared with Figure 5.4(b) after the same total number of bytes has been transmitted.

In fact, if we go from Figure 5.4(b) and set the ROI back to the whole image and go on with the transmission, we could get exactly the same result as Figure 5.4(d) after 29472 bytes, compared with 29056 bytes of Figure 5.4(d). This shows that the region-based dynamic transmission has very small overhead.

5.3.5 Dynamic Transmission Tests on 'US1'

The dynamic transmission tests were also performed on ultrasound image 'US1' of Figure 4.6(a). Ultrasound images have text and large black spaces, which makes their compression characteristics rather different to the lena-like images, where meaningful

(a) (b) (c) (d)

Figure 5.4: DRW Dynamic Compression of Lena:
(a) Mask set to whole image, first 1314 bytes
(b) Follows (a) , after a ROI is selected, and a total of 3645-bytes have been transmitted
(c) shows the first 3645 bytes if the whole image is transmitted
(d) shows without the ROI, 29056 bytes are needed to create the same fidelity within the ROI as shown in (b)

data occupies the whole image. For these ultrasound images, it is reasonable to choose the ultrasound data as the ROI, as shown in Figure 5.6. Here, the DRW scheme gives the non-ROI only enough fidelity requirement to recognize the texts and scales. The saved bytes budget is used to make the ROI more detailed. In practice, the mask is initialized to be the whole image, and after the non-ROI part is clear enough, the mask is set to the ROI for the successive coding. In this case the area of the ROI is 25.72% of the whole image area.

The in-ROI PSNR at different stages of the dynamic compression are listed in Table 5.5, and the bitrates needed for EZW to achieve the same in-ROI PSNR are also listed as a comparison. The data in Table 5.5 are also plotted in Figure 5.5. It is noticed that the DRW method always out-performs the conventional whole-image EZW compression. Note that for an in-region PSNR of 30.81, the DRW method uses only 0.40 bpp whereas the EZW method requires 0.61bpp; from the original 8bpp data, the DRW method provides a 20 times compression, whereas the EZW method provides only 13.1 times compression. Figure 5.6 shows the result of DRW at 0.21 bpp, which has the same in-region PSNR as the result of EZW at 0.34 bpp, hence 38% more compression is realized. Similar results were obtained for other grey-scale ultrasound images.

5.3.6 Dynamic Transmission Test on 'MRI1'

A similar test is directed on an MRI lung image 'MRI1' as shown in Figure 5.7. The compression starts with the mask set to the whole image, and after the receiver side could recognize the lung area, the ROI is drawn as shown in Figure 5.8. Then

Table 5.5: DRW test result on US1 (Figure 4.6a)

in-region PSNR (dB)	bpp/compression ratio DRW	bpp / compression ratio EZW
19.51	0.060/133:1	0.060/133:1
21.92	0.10/80:1	0.15/53.3:1
25.59	0.21/38.1:1	0.34/23.5:1
30.81	0.40/20:1	0.61/13.1:1
36.51	0.69/11.6:1	1.01/7.92:1
42.64	1.02/7.8:1	1.46/5.48:1
49.29	1.37/5.84:1	1.94/4.12:1
56.92	1.70/4.71:1	2.42/3.31:1

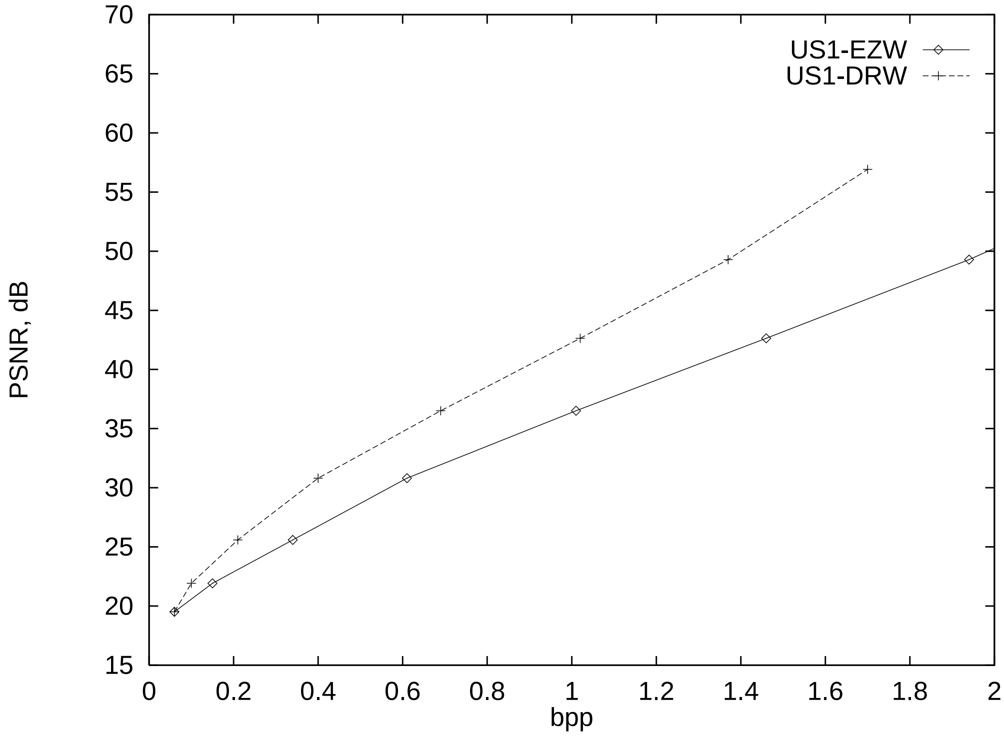


Figure 5.5: Plot of DRW and EZW performance on US1 (Figure 4.6(a))
*the PSNR of the region as plotted in Figure 5.6

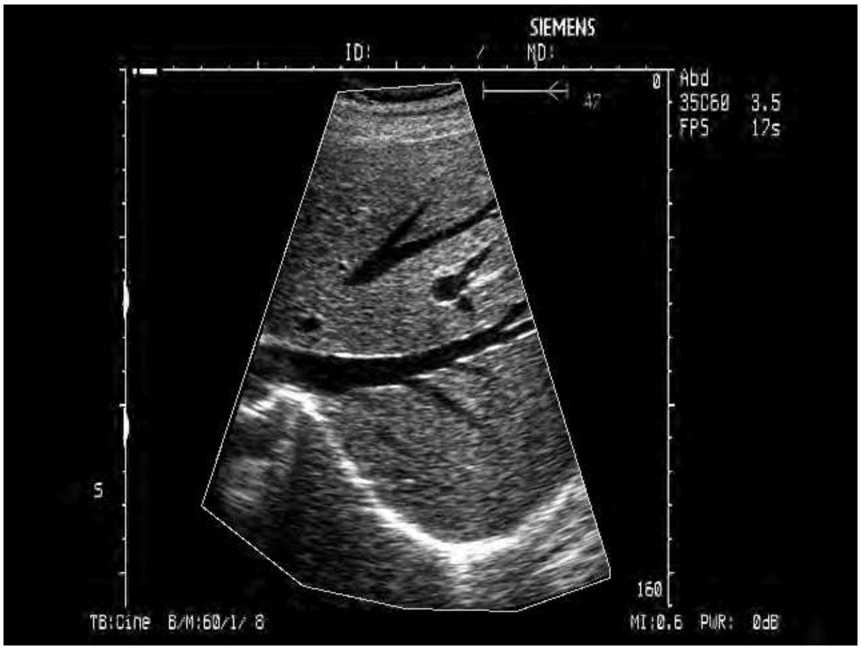


Figure 5.6: US1 compressed with DRW at 0.21 bpp, mask as plotted.

coding is done with the mask set to the ROI. The same image is also compressed and progressively transmitted with EZW for comparison. The test results are listed in Table 5.6 and plotted in Figure 5.9. The area of the ROI is 24.37% of the total image area.

In this test we found that the difference between the two results of DRW and EZW is larger than for the 'US1' ultrasound image, even though the ROI areas have almost the same percentage of the whole image. This is because there is a lot of information in the non-ROI area including the strong noise in the barred region of the middle of the image. Not coding them will definitely save much of the bit budget. In this example, on achieving the same image quality of the ROI, the DRW algorithm may only use one fifth of the EZW data size, although the region is almost one fourth of the image area.

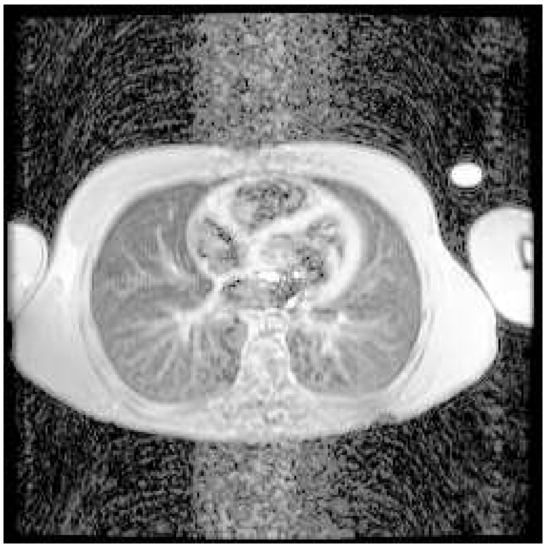


Figure 5.7: Original image of MRI2, lung image

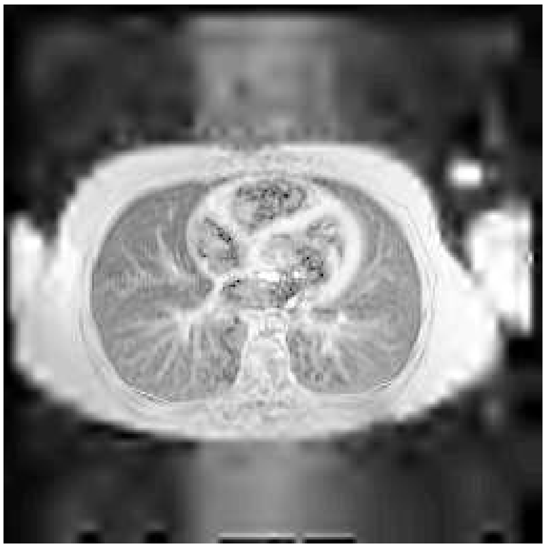


Figure 5.8: MRI2 compressed with DRW at 0.21 bpp, mask as plotted.

Table 5.6: DRW test result on MRI2 (Figure 5.7)

in-region PSNR (dB)	bpp DRW	bpp EZW
21.36	0.035	0.035
23.57	0.063	0.10
26.55	0.11	0.39
30.02	0.22	1.09
34.61	0.40	2.11
39.93	0.67	3.22
45.64	1.01	4.36

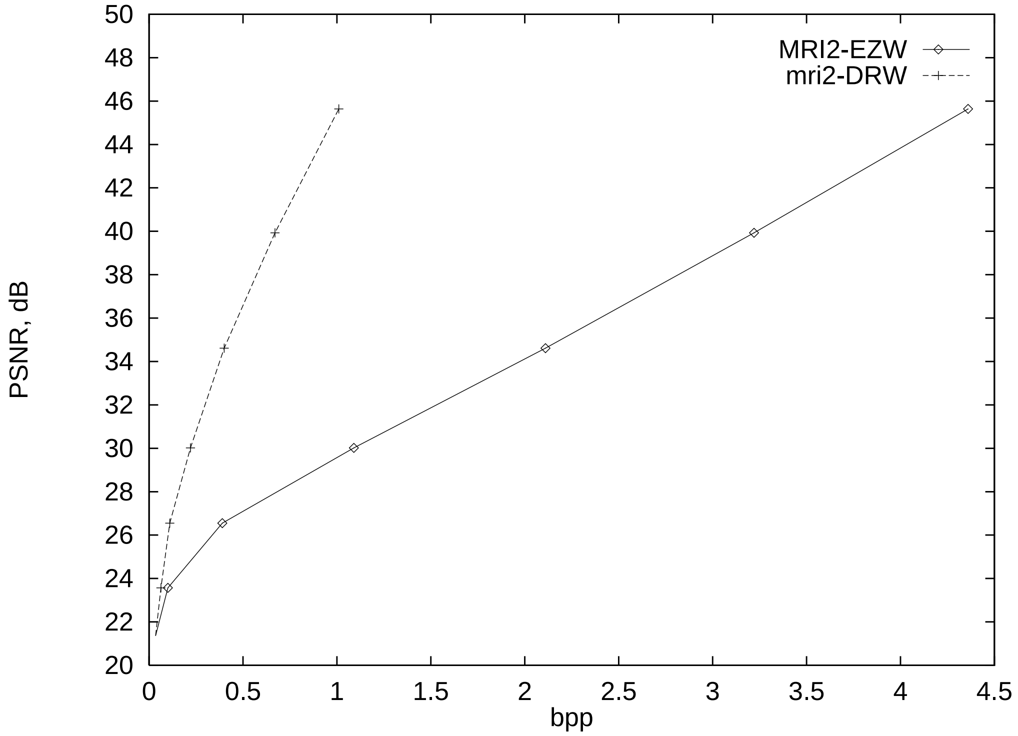


Figure 5.9: Plot of DRW and EZW performance on MRI2 (Figure 5.7)
*the PSNR of the region as plotted in Figure 5.8

Chapter 6

Summary

6.1 Review

In this thesis, a general, high-performance wavelet compression algorithm called EZW coding has been studied and implemented. Based on the research work on the EZW's algorithm and performance, a new method is proposed, called DRW coding, which incorporates a dynamic region-based feature into wavelet coding.

EZW is a coding which has the embedding feature to support progressive transmission, but it doesn't support region-based coding. In our DRW algorithm, not only is the region-based feature added, but also the regions can be changed dynamically while the progressive coding and decoding is on-going. To solve the partition edge artifacts problem which occurs for transform based coding, the mask dilation based on the transform domain partition is introduced, and tests have shown its good result.

Both EZW and DRW are implemented in the WiT visual programming environment. Their performance on both standard test image and medical image are analyzed and compared.

It has been found that the DRW algorithm could either act the same as EZW or could gain far more compression by turning on the region-based feature. As this region-based coding could be triggered on the fly, and even could be fixed up to get a normal whole image compression with little extra expanse, we propose that this algorithm could have its merits to supplying telemedicine service over narrow band communication networks.

6.2 Future Work

In this thesis, we have shown that the image transmission quality and efficiency could be improved by adopting our DRW scheme, and some future works are also incurred.

The choice of the wavelet filter is an important aspect of tuning any wavelet-based compression system. For region-based compression, the region segmentation performance of a filter could be take into consideration.

A performance improvement may occur if we incorporate the dynamic region-based concept into other zerotree based algorithms such as the Said-Perlman algorithm [19] which has better performance than EZW.

From the test of EZW on ultrasound images, we found the strange PSNR performance in compare with ordinary greyscale images. The spectral characteristics of different medical image modalities may require different wavelet basis functions or different filters, and different wavelet compression techniques.

Another aspect concerns the design of the Graphical User Interface for interactively drawing the ROI; we are working on designs for the GUI, including the possibility of automatic or semi-automatic segmentation of the ROI in ultrasound images to aid in data storage and display.

Bibliography

[1] Mark C. Anderson. Task-oriented lossy compression of magnetic resonance images. M.Sc. thesis, Simon Fraser University, August 1995.

[2] Marc Antonini and Michel Barlaud. Image coding using wavelet transform. *IEEE Transactions on Image Processing*, 1(2):205–220, April 1992.

[3] T. Arden and J. Poon. *WIT Programmer's Guide*. Logical Vision Ltd., Burnaby, BC, October 1993. Version 4.1.

[4] T. Arden and J. Poon. *WIT User's Guide*. Logical Vision Ltd., Burnaby, BC, October 1993. Version 4.1.

[5] Chitra Balasubramaniam. Dataflow image processing. *Computer*, 27(11):81–84, November 1994.

[6] Peter A. Ensminger. Tutorial on telemedcine at syracus. Northeast Parallel Architecture Center, http://www.npac.syr.edu/ users/ensmingr/ TMED.html, April 1996.

[7] Kansas Telemedicine Policy Group. Telemedicine accessing the kansas enviroment. Unpublished, Funded by the Kansas Health Foundation, Wichita, Kansas., November 1993.

[8] David A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(10):1098–1101, September 1952.

[9] I.Daubechies. Orthonormal bases of compactly supported wavelets. *Comm. Pure Appl. Math.*, 41:906–966, 1988.

[10] Anil K. Jain. Image data compression: a review. *Proceedings of the IEEE*, 69(3):349–389, March 1981.

[11] Anil K. Jain. *Fundamentals of Digital Image Processing.* Prentice-Hall, Inc., University of California, Davis, 1989.

[12] Björn Jawerth and Wim Sweldens. An overview of wavelet based multiresolution analyses. *SIAM Review*, 36(3):377–412, September 1994.

[13] S. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11:674–693, 1989.

[14] William B. Pennebaker and Joan L. Mitchell. *JPEG Still Image Data Compression Standard.* Van Nostrand Reinhold, New York, 1993.

[15] William B. Pennebaker, Joan L. Mitchell, G. G. Langdon, Jr., and R. B. Arps. An overview of the basic principles of the Q-coder adaptive binary arithmetic coder. *IBM Journal of Research and Development*, 32(6):717–726, November 1988.

[16] T.A. Ramstad, S. O. Aase, and J. H. Husoy. *Subband Compression of Images: Principles and Examples*. Advances in image Communication. Elsevier Science B.V., 1995.

[17] Amir Said. An image multiresolution representation for lossless and lossy compression. Submitted to *IEEE Transactions on Image Processing*, ftp: ipl.rpi.edu pub/EW_Code, 1995.

[18] Amir Said and William A. Pearlman. Reversible image compression via multiresolution representation and predictive coding. In *SPIE Symposium on Visual Communications and Image Processing*, November 1993.

[19] Amir Said and William A. Pearlman. A new fast and efficient image codec based on set partitioning in hierarchical trees. Submitted to *IEEE Transactions on Circuits and Systems for Video Technology*, ftp: ipl.rpi.edu pub/EW_Code, April 1995.

[20] Jerome M. Shapiro. An embedded wavelet hierarchical image coder. In *Proceedings of ICASSP'92*, volume IV, pages 657–660. IEEE, 1992.

[21] Jerome M. Shapiro. An embedded hierarchical image coder using zerotrees of wavelet coefficients. In James A. Storer and Martin Cohn, editors, *DCC '93 : Data Compression Conference*, pages 214–223. IEEE, April 1993.

[22] Jerome M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41(12):3445–3462, December 1993.

[23] G. Wallace. The JPEG still-picture compression standard. *Communications of the ACM*, 34, April 1991.

[24] S. A. Werness. Statistical evaluation of predictive data compression systems. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-35(8):1190, 1987.

[25] S. A. Werness. Correction to "statistical evaluation of predictive data compression systems". *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-36(1):140, 1988.

[26] Ian H. Witten, Radford M. Neal, and John G. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, June 1987.

[27] Zixiang Xiong and M. T. Orchard. Wavelet packets-based image coding using joint space-frequency quantization. In *Proceedings of ICIP'94, Austin, Texas*, 1994.

[28] Jianhua Xuan, Tülay Adah, Yue Wang, and Richard Steinman. Predictive tree-structured vector quantization for medical image compression and its evaluation with computerized image analysis. In Yongmin Kim, editor, *Medical Imaging 1995: Image Display*, Proceedings of SPIE 2431. SPIE, 1995.