A Fuzzy and Wavelet-Based Image Compression Algorithm

A thesis presented in partial fulfillment
of the requirements for the degree of
Master
In
Computer Science
At Massey University, Albany,
New Zealand

Li Huang

2006

Abstract

Nowadays, the Internet and digital image widespread are used in the industry, commerce, military, traffic and all walks of life. However, this kind of general use resulted in required for less transmission time and storage space. Image compression can address the problem of reducing the amount of data to represent a digital image. The image will satisfy the transmission and the preserved request after the compression. With the increasing use some technologies in the image processing, image compression also requires new technology to get the high compression ratio and more better image quality. Therefore, a new standard has been developed by Joint Photographic Experts Group (JPEG). Apart from JPEG, there are other algorithms developed for image compression, Normally, EZW, SHIPT and VQ algorithms. However, they all deal with the calculation of coefficients with too much complexity; as a consequence, compressing still image takes too much time. In the light of these problems, this thesis introduces a new method for dealing with the requirements of the coefficients while retaining the important detail in the image, by employing a Fuzzy Logic technique reduce the number of the coefficients, and then utilizes the Huffman or LZW algorithm to complete the image compression. The algorithm developed in this research, called IWF algorithm, is based on four key techniques: 1) a wavelet transform for decomposition. This technique allows the combination of lossless and lossy compression with extremely high compression rate and image quality. 2) Quantization, this technique generally works by compressing a range of value to a single quantum value. By reducing the number of discrete symbols in a given stream, the stream becomes more compressible. This step in the IWF process is a lossy transformation. 3) Adaptation of Fuzzy logic techniques. This step uses the Fuzzy Logic techniques to handle the wavelet coefficients, enable the wavelet coefficients to have the same value in the high subbands. 4) Adaptation of Lossless data compression techniques.

Keywords: Image Compression, Fuzzy Logic, Wavelet transforms, Decomposition, Haar Wavelet transform,

Acknowledgments

I thank Reyes,Napoleon for his invaluable support and guidance as my supervisor. His readiness to give help and suggestions throughout the year has been highly appreciated.

I would also like to thank my family for their support throughout this year.

Li Huang

Massey University
March 2006

List Figures

List Tables

List of Abbreviation

1-D - 1 Dimension

2-D - 2 Dimension

ANN - Artificial neural Network

DCT - Discrete Cosine Transformation

DFT - Discrete Fourier Transformation

DHT - Discrete Hada Masurium transformation

DM - Delta Modulation

DPCM - Differential Pulse Code Modulation

EZW - Embedded Zerotree Wavelet

GIF - graphic interchange format

ISDN - Integrated Services Digital Network

IWF - Image Compression Based on Wavelet and Fuzzy Logic

IZ - Isolated Zero

LIP - List of Insignificant Pixels

LIS - List of Insignificant Set

LSP - List of Significant Pixels

LZW - Lempel Ziv Welch

MSE - Mean Square Error

N - Negative

P - Positive

PCM - Pulse Code Modulation

PDF - portable document format

PSNR - Peak Signal to Noise Ratio

RMSE - root mean square error

SPHIT - Set Partitioning in Hierarchical Trees

TIFF - tagged image file format

VQ - Vector Quantification

ZTR - Zero Tree Root

Chapter 1

Introduction

1.1 Overview of the Current State of Technology

The main goal of image compression is to minimise the data volume of an image with no significant loss of information. The reduction in file size allows more images to be stored in a given amount of disk or memory space, and it also reduces the transmission cost of images sent over the Internet or downloaded from Web pages.

Several different methods exist for the compression of images. All basic image compression groups have advantages and disadvantages. Methods for image compression fall into several major categories:

- Transform-based methods
- Vector quantisation methods
- Pyramid-based techniques
- Hybrid compression methods
- Huffman encoding

Transform-based techniques are renowned to better preserve subjective image quality and are less susceptible to statistical image property changes both inside a single image, and between images. One attractive quality of transform-based techniques is their insensitivity to transmission channel noise. If one transform coefficient is altered during transmission, the resulting image distortion is distributed homogeneously through the image or image section and therefore undesirable disruptive effects are minimised. Predictive compression techniques is transmitted, the error causes image distortion not only in a particular pixel, but within a neighbourhood of pixels because

the predictor involved has a considerable visual effect in a reconstructed image.are plagued with the problem of propagating difference value transmission errors. When an erroneous difference value

Vector quantisation methods have the advantage of only employing a single decoding scheme; comprising of a look-up table only. Vector quantisation methods, however, require complex code, and parameters are very sensitive to image data, often blurring the edges of images.

Pyramid-based techniques have a natural compression ability, and are suitable for dynamic image compression and smart transmission approaches. Pyramid-based methods have shown the potential for further improvement of compression ratios.

Hybrid compression methods generally combine the different dimensionalities of transform compressions with predictive compressions. Hybrid compression methods can also combine predictive approaches with vector quantisation.

Huffman encoding falls into a separate category of image compression, and is distinguished by its provision of optimal compression and error-free decompression.

1.2 Research Objectives

The focus of this study is on developing a new image compression method based on Wavelet transforms and Fuzzy Logic techniques. Many researches have been conducted using Wavelet transforms, however, only very few researches have been done using the fuzzy approach, and the full potential of Fuzzy Logic in the field of image compression has not yet been fully explored.

The general objectives of this study are:

1. To understand the basic concepts of wavelet transform operation.
2. To learn about image compression techniques developed using the Wavelet

transform and Fuzzy Logic.

3. To combine Wavelet transforms and Fuzzy techniques to reduce the computational cost of image compression.

The specific objectives of this study are:

1. To develop a Hybrid Wavelet approach to image compression that would be faster and more efficient than other Fuzzy Logic approaches.
2. To develop a fast and simple Fuzzy coding technique for generating the Wavelet coefficients.

1.3 Scope and Limitations of Research

The scope of this study will include only static images. Both grey-scale and colour images will be used. The scope of the study will not, however, be extended to testing the algorithm on video sequences.

1.4 Assumptions

It is assumed that the performance of the algorithm developed in this research can be compared against other algorithms by compressing and decompressing the same set of images used by other algorithms and inspecting speed, compression ratios, and root mean square errors.

1.5 Significance of Research

Image compression addresses the problem of reducing the amount of data required to represent a digital image. From the information theory viewpoint, image compression is the removal of redundant information, namely keeping the non-definite information, and removing the determined information. From a mathematical viewpoint, image compression means to transform a 2-D pixel array into a statistically uncorrelated data set, where such transformation is applied to storage or transmission of the image. At some rate, the compressed image is decompressed to reconstruct the original image or an approximation of it.

Wavelet compression [Antonini, M., Barlaud, M., Mathieu, P., & Daubechies, I. (1992)] is one of the most effective methods of image compression. The algorithm is based on multi-resolutional analysis. Like conventional JPEG compression, the JPEG compression algorithm is based on the DCT transform, and separated the whole image to small sub-images. This method however, returns grainy compressed images, or those reflecting the "block effect". On the other hand, Wavelet compression algorithm presents an image as sets of real coefficients. After the wavelet decomposition step, the most important image information is located at the low frequency subband. In addition, the other Wavelet coefficients of a typical image are nearly zero, and the image is thus well-approximated with a small number of high-valued Wavelet coefficients. The advantage of Wavelet compression is that, in contrast to JPEG, the Wavelet algorithm does not divide the image into small blocks, but takes and analyzes the whole image. The distinguishing characteristic of Wavelet compression is that it allows arriving at the best compression ratio.

Fuzzy Logic techniques are mainly employed in digital imaging for Contrast Adjustment, Image Enhancement, Image Segmentation, and Image Edge Detection [Tizhoosh, H. R. (1997)].

By and large, Wavelet image compression is gaining acceptance in the internet community for its robustness in the presence of noise during the transmission of images. Unlike predictive methods, Wavelet-based techniques do not generate square image compression artifacts. On the other hand, one advantage of predictive methods over Wavelet-based techniques is that they achieve larger compression ratios in a much less expensive way. In light of this problem, this research amends the weaknesses of pure Wavelet image compression techniques by injecting Fuzzy techniques to speed up the calculation of Wavelet coefficients. As a result, image compression is achieved by utilizing Fuzzy logic in the quantization of Wavelet coefficients, allowing for the preservation of important image details while discarding less significant ones.

1.6 Research Methodology

In devising a new Fuzzy and Wavelet-based image compression algorithm, the following steps have been taken:

1. Study of the basic concepts of image compression.
2. Study of the different colour spaces and implementation of colour space transformation equations.
3. Study of the Run Length Coding.
4. Study and implementation of the Huffman algorithm.
5. Study and implementation the LZW algorithm.
6. Study of Wavelet transforms.
7. Study and implementation of the EZW algorithm.
8. Study of the SPHIT algorithm.
9. Comparison of EZW and SPHIT.
10. Study of JPEG and JPEG 2000.
11. Comparison of JPEG and JPEG 2000.
12. Study of Fuzzy Logic and Fuzzy image processing.

13. Study and analysis of other image compression algorithms based on Wavelet transforms.
14. Study and analysis of other existing image compression algorithms based on Fuzzy logic.

1.7 Organisation of the Thesis

This work is divided into five chapters and is described briefly as follows:

Chapter 2 provides a theoretical framework for the issues relating to different kinds of image compression and Fuzzy Logic techniques (Basic image concepts, image coding, transform coding, Fuzzy Logic and Fuzzy image processing).

Chapter 3 presents a review of related literature, and discusses relevant algorithms in detail with analysis.

Chapter 4 presents a brief overview of the algorithm developed. The chapter starts by introducing the principles of the algorithm, and then follows with the intricacies of the algorithm applied to both grey-scale and colour images, including analysis and experimental results.

Lastly, in Chapter 5, conclusions and future works are presented.

Chapter 2

Theoretical Framework

2.1 Basic Image Compression Concepts

2.1.1 Image Types

From the point of view of compression, images can be classified into the following categories:

- Binary images
- Grey-scale images
- Colour images
- Video images

Binary images are those images where pixels have only two possible intensity values. They are normally displayed as black and white. Numerically, the two values are often 0 for black and 1 for white.

Grey-scale images are coded using one number per pixel representing one of 256 different gray tones ranging from black to white.

Colour images use three numbers for each pixel, allowing possible use of millions of colours within the image. These images are often refereed to as *true color* RGB images.

Video Images captured from video cameras are in color. Adjustments in contrast, color, sharpness, etc., are commonly made on these to raw video images.

Grey-scale, colour, and video images are differ from binary images in that compression methods designed for grayscale images also apply to colour image and video image, but do not usually apply to the binary images.

2.1.2 Colour Space

A colour space [Gonzalez & Woods, 2002] is a model for representing colour in terms of intensity values. There are a lot of colour spaces in common usage depending on the different application or different facility. A colour space specifies how colour information is represented. It defines a one, two, three, or four-dimensional space whose dimensions, or components, represent intensity values. A colour component is also referred to as a colour channel. For example, RGB space is a three-dimensional colour space whose components are the red, green, and blue intensities that make up a given colour ["Advanced Color Imaging on the Mac OS," 1996].

Colour images are encoded as triplets of values. RGB is an additive colour model that is used for light-emitting devices (e.g. CRT displays). Two common colour models in imaging processing are RGB and CMY, two common colour models in video are YUV and YIQ. The YUV colour usually uses in the colour image compression. YUV uses properties of the human eye to prioritize information. Y represents the black and white (luminance) image; U and V are the colour difference (chrominance) images. YIQ uses a similar idea [Li & Drew, 2004].

Besides the hardware-oriented colour models (i.e., RGB, CMY, YUV, YIQ), HSB (Hue, Saturation, and Brightness) and HLS (Hue, Lightness, and Saturation) are also commonly used [Li & Drew, 2004].

In image compression, colour is usually dealt with in two forms (RGB and YUV):

- Red, Green and Blue (RGB) colour images. RGB is perhaps the simplest and most commonly used colour space. It use 8 bits to hold each primary colour and then these

colours are combined to produce true colours that can be displayed on a monitor or television.

• Luminance, chrominance and saturation (YUV). This is an abstract form of the RGB system that is more useful for image compression. To convert from RGB to YUV space, the following equations can be used:

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

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

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

• YIQ is used in U.S. television standard, and NTSC (National Television System Committee). YIQ is similar to YUV, except that it is colour space is rotated 33 degrees clockwise. The Y component represents the luminance information, while I and Q represent the chrominance information. The equations to convert from RGB to YIQ are:

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

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

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

• YCbCr is a family of colour spaces used in video systems. Y is the luminance component, while, Cb and Cr are the chrominance components. YCbCr is a subset of YUV that scales and shift the chrominance value into the range of 0 to 1. The equations to convert from RGB to YCbCr are:

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

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

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

Fig.1 is the comparison of RGB, YUV, YIQ and YCrCb space (The test image is the standard of image compression).

Original image (image of Lena widespread used in image processing)

R

G

B

Y

U

V

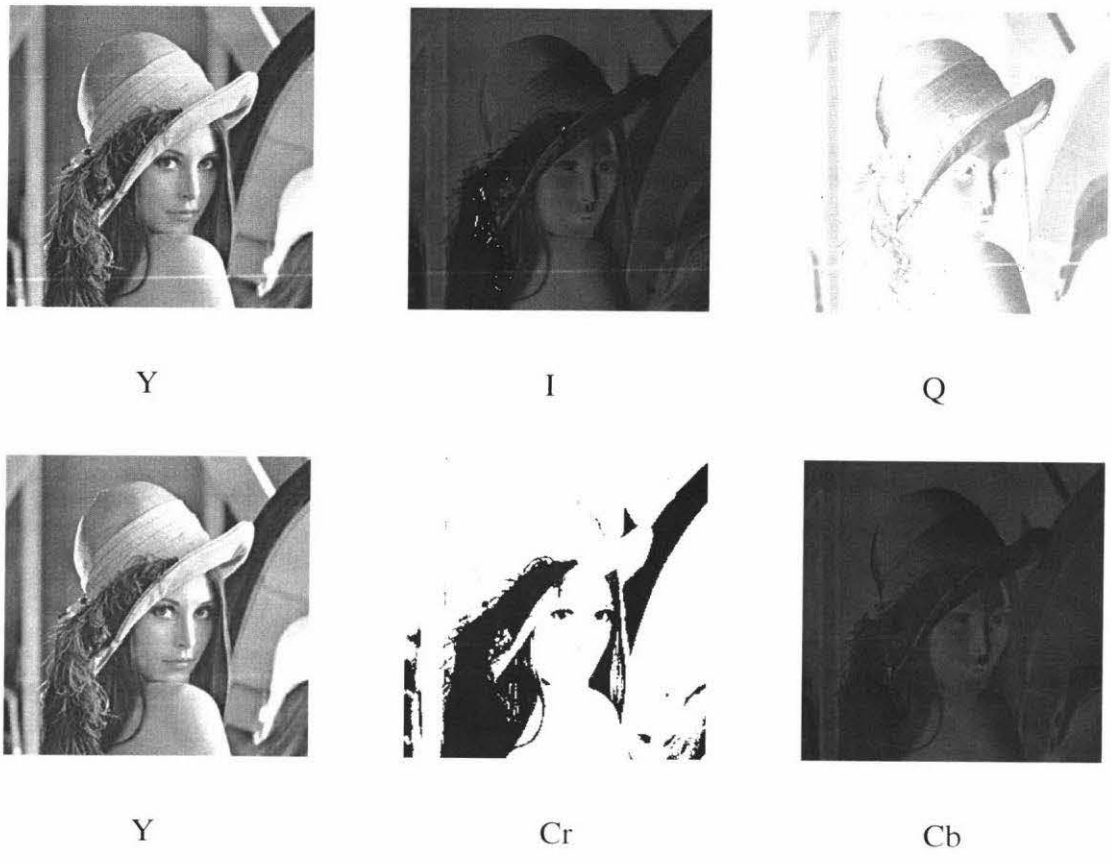


Fig.1 The Comparison of RGB, YUV, YIQ and YCrCb Colour Space

The YUV mode is the popular color space for the colour image compression, because the luminance (Y) image carries the most of important spatial information of the original image and is indeed just a weighted average of the original red, green, and blue digital count values for each pixel. The other two individual chrominance (U and V) images show very little spatial information. This is advantageous for the goal of colour image compression.

2.1.3 Data Redundancy

In digital image compression, three kinds of basic data redundancies exist [Gonzalez & Woods, 2002]: coding redundancy, interpixel redundancy and psychovisual redundancy. If we can reduce or eliminate one or many of these redundancies, we can obtain the data compression effect and also we can compress image.

Usually in a digital image, the number of bits used for the representation of each pixel is constant for all the pixels, regardless of the value of the pixel and the frequency of occurrence of that value in the image [Zehtab-Fard]. In general, coding redundancy is present in an image if the possible values are coded in such way that they use more code symbols than absolutely necessary. Coding redundancy is almost always present in images that are coded without taking into consideration the occurrence frequency of each value [Zehtab-Fard].

Interpixel redundancy is another form of data redundancy, which is related to interpixel correlation within an image. Usually the value of certain pixel in the image can be reasonably predicted from the values of a group of other pixels in the image [Bogomjakov, 1998]. Thus, the value of the individual pixel carries relatively small amount of information and much more information about pixel value can be inferred on the basis of its neighbors' values. These dependencies between pixels' values in the image are called interpixel redundancy [Bogomjakov, 1998].

Human eye does not respond equally to all visual information, information of less relative importance is psychovisual redundancy. Removing this kind of redundant information does not obviously reduce image quality for an observing human, but in actuality may result in the loss of a large amount of image data.

According to the three categories of basic data redundancies, the encoder includes below the three sequences independent operations, but corresponds the decoding contains the two counter-foreword independent operations, like the chart shows (Fig.2) [Gonzalez & Woods, 2002]:

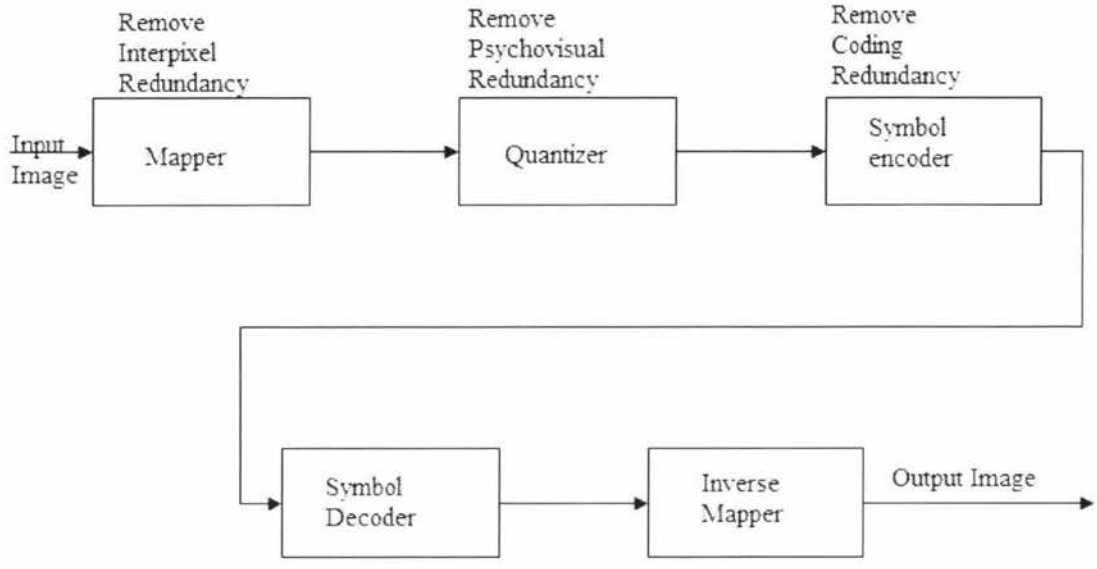


Fig.2 Source Encoder and Decoder Model [Gonzalez & Woods, 2002]

As can be seen in Fig.2, the Quantizer is no longer needed in the decompression stage. This is because the quantization operation is not reversible.

2.1.4 Two Categories of Image Compression Techniques

There are two basic types of image compression [Gonzalez & Woods, 2002]

- Lossless compression.
- Lossy compression.

Lossless coding techniques guarantee that the decompressed image is absolutely identical to the image before compression. This is an important requirement for some application domains (e.g. medical imaging, where not only high quality is in demand, but unaltered archiving is a legal requirement). Lossless techniques can also be used for the compression of other data types where loss of information is not acceptable (e.g. text documents and program executables).

In most applications, images (and music too) need not be reproduced exactly. An approximation of the original image is enough for most purposes, as long as the error

between the original and the compressed image is tolerable. As a result, lossy compression techniques schemes allow redundant and unnecessary information to be lost. It will affect the image quality, but this kind of lost the human eye can acceptance. Typically with lossy schemes, there is a tradeoff between compression ratio and image quality, normally the high compression ratio always resulted in the bad image quality. In addition, lossy compression techniques are typically more complex and require more computations [Mai].

2.1.5 Compression Ratio and Error Metrics

The compression ratio is a key feature of any discussions of data compression. A compression ratio is simply, the size of the original data divided by the size of the compressed data. (For a given image, the greater the compression ratio, the smaller the final image will be.)[Gonzalez, R. C., & Woods, R. E. (2002)]

Compression ratio = original data / compressed data

Two of the error metrics used to compare the various compression techniques are the Mean Square Error (MSE) and the Peak Signal to Noise Ratio (PSNR). The MSE is the cumulative squared error between the compressed and the original image, whereas PSNR is a measure of the peak error. Mathematical formulae for two are [Gonzalez, R. C., & Woods, R. E. (2002)][Kumar, 2001]

$$MSE = \frac{1}{N} \sum_{i,j} \left(f(i,j) - f'(i,j) \right)^2 \qquad [1]$$

$$PSNR = -10 \log_{10} \left(\frac{MSE}{Max \left| f(i,j) \right|^2} \right) \qquad [2]$$

Where N is the number of pixels in the image

f (i,j) if the original image pixel intensity at i,j

and f'(i,j) is the compressed image pixel intensity at i,j.

A lower value for MSE means lesser error, and as seen from the inverse relation between the MSE and PSNR, this translates to a high value of PSNR [Kumar, 2001].

2.2 Image Coding

2.2.1 An Introduction to Image Coding

Most image coding methods fall into one of four major categories:

- Image element coding
- Predictive coding
- Transformation coding
- Other coding.

Image element coding individually codes each image element, and does not consider relationships between image elements. The most commonly utilised methods fall into four sub-categories: The Pulse Code Modulation (PCM), the Entropy Coding, the Run Length Coding, and the Bit Plane Coding.

Predictive coding only carries on the code to the new information, thus removes the relevance and redundancy between image elements. The most commonly utilised methods fall into two sub-categories: The Delta Modulation (DM), and the Differential Pulse Code Modulation (DPCM).

Transformation coding transforms the image to another data field (for example frequency area), to enable a mass of information to have less data to represent. The most commonly utilized methods fall into three sub-categories: the Discrete Fourier Transformation (DFT), the Discrete Cosine Transformation (DCT), and the Discrete Hada Masurium Transformation (DHT).

Other coding methods commonly include: The Hybrid Coding, Vector Quantification (VQ), LZW algorithm, and coding methods which appeared in recent years, such as the Artificial Neural Network (ANN) algorithm, the Fractal algorithm, the Wavelet, the Object-Based algorithm, the Model-Based algorithm and so on.

Huffman coding and Arithmetic Coding belong to statistical coding (the coding method does not consider the relationship between elements; it just considers the appearance probability of pixel grey value and the characteristic of distribution. This kind of coding method is called statistical coding).

2.2.2 Run Length Encoding

Run length encoding [Bourke, 1995] is one of the simplest and a straightforward way of data compression methods, so that it takes up less space. This technique is taking advantage of repetitive data. These repeating data are called *runs*. In the image compression some images have large areas of constant colour. *Runs* are represented with a count and the original data byte [Mai]. For example, a source string of

AAAAAABBBBBCCCCCCCCDDDEEEE

Could be represented with

6A5B8C3D4E

(Six A are represented as 6A. Five B are represented as 5B...)

This example represents 26 bytes of data with 10 bytes, achieving a compression ratio of:

$$\text{Compression ratio} = 26 \text{ bytes} / 10 \text{ bytes} = 2.6$$

Run length encoding is most useful on data that contains many runs; it works very well for simple images such as icon, line drawings and cartoons. However for natural images it doesn't work just as well, because run length encoding capitalises on characters repeating more than three times [Mai], it also doesn't work well with English text. A method can make run length encoding get better results is one that uses fewer bits to represent the most frequently occurring data. Data that occurs less frequently would require more bits [Mai].

2.2.3 Huffman Coding

This algorithm developed by D.A. Huffman [Gonzalez, R. C., & Woods, R. E. (2002)], and is based on the fact that in an input stream certain tokens occur more often than others. Based on this knowledge, the algorithm builds up a binary tree according to their frequency of occurrence [Image Compression techniques]. Each node of this tree is assigned a new code word, where the length of the code word is determined by its position in the tree. Therefore, the token which is most frequent and becomes the root of the tree is assigned the shortest code. Each less common element becomes the leaf of the tree and is assigned a longer code word. The least frequent element is assigned a code word which may have become twice as long as the input token [Image Compression techniques].

The Huffman algorithm builds the binary tree will according many steps. The first step in creating Huffman codes is to create an array of character frequencies, in this step just to count the frequency of each character, this step is very important for build the binary tree, because it will decide the root and leaf of the binary tree. This step is as simple as scanning your data and incrementing each corresponding array element

for each character encountered. After first step the binary tree can easily be constructed by recursively grouping the lowest frequency characters and nodes. The algorithm is as follows [Mai]:

Step1. All characters are initially considered free nodes [Mai].

Step2. The two free nodes with the lowest frequency are assigned to a parent node with a weight equal to the sum of the two free child nodes [Mai].

Step3. The two child nodes are removed from the free nodes list. The newly created parent node is added to the list [Mai].

Step4. Steps 2 through 3 are repeated until there is only one free node left. This free node is the root of the tree [Mai].

Let us create a binary tree to explain the Huffman algorithm. The 8 x 8 pixel image is small to keep the example simple. The letters represent the colours Red, Green, Cyan, Magenta, Yellow, and Black [Mai].

R	K	K	K	K	K	K	K
K	K	K	R	R	K	K	K
K	K	R	R	R	R	G	G
K	K	B	C	C	C	R	R
G	G	G	M	C	B	R	R
B	B	B	M	Y	B	B	R
G	G	G	G	G	G	G	R
G	R	R	R	R	G	R	R

Table.1 Sample 8 x 8 screen of red, green, blue, cyan, magenta, yellow, and black pixels

Before building the binary tree, the frequency table for Table.01 must be generated

Colour	Frequency
red	19
black	17
green	16
blue	5
cyan	4
magenta	2
yellow	1

Table.2 Frequency Table for Table.01

Create the Binary tree creation (Fig.3)

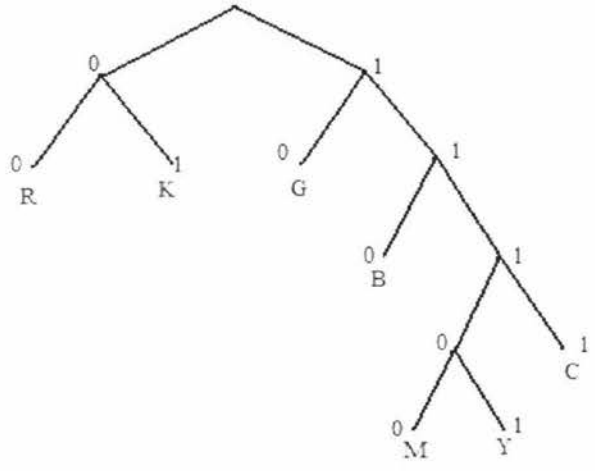


Fig.3 Binary Tree Creations

The resulting Huffman coding are shown in Table.03

red	00
black	01
green	10
blue	110
cyan	1111
magenta	11100
yellow	11101

Table.3 Huffman Codes for Table.01

The main disadvantage of Huffman coding is that encoding requires two scans over the whole data. The first scan is that accumulates the character frequency data, then compression on the second scan. It will waste the compression time. But the drawback can be solved by many ways. One way to remove the first scan is to always use one fixed table, but this method will not be optimized for every data set that will be compressed [Mai].

The second disadvantage is that the decoder must use the same binary tree as the encoder. This will result in take more store space to keep the binary tree. This drawback also can solve by providing the standard binary tree to the decoder to use, but this way may not be optimum for the code being compressed [Mai]. Another option is to store the binary tree with the data. Rather than storing the tree, the character frequency could be stored and the decoder could regenerate the tree. However, this would increase decoding time; because the decoder must recreate the binary tree when it decodes the compressed data [Mai].

Here are some sample results returned by the Image compression system developed (using Delphi) to implement the Huffman algorithm:

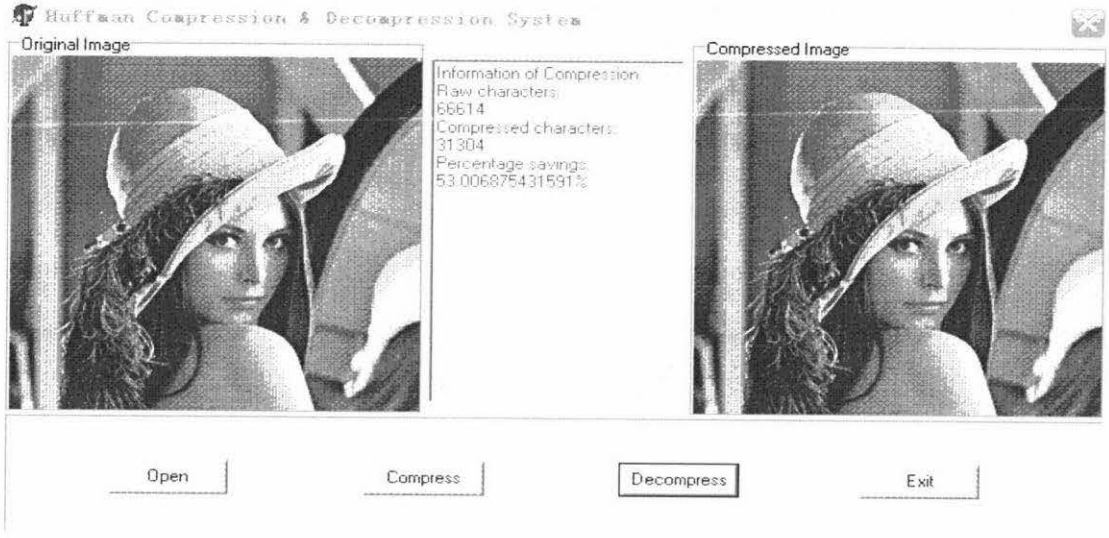


Fig.4 Delphi Implementation the Huffman Algorithm

Original image

compressed image

Fig.5 Comparison the between Original Image and Reconstructed Image (Huffman algorithm)

2.2.4 LZW Coding

In 1977, a paper was published by Abraham Lempel and Jacob Ziv laying the foundation for the next big step in data compression [Gonzalez, R. C., & Woods, R. E. (2002)]. While Huffman coding achieved good results, however the Huffman coding was typically limited to coding one character at a time. This algorithm requires a lot of time on the compression and decompression step. On the other hand, Lempel and Ziv proposed a scheme for encoding strings of data [Mai]. The main different between the two compression algorithm is that the Huffman coding one character at once, but the LZW coding strings of data at once. The main different resulted in the LZW algorithm more effect than the Huffman algorithm. This technique also resulted in greatly reducing the time of compression and decompression.

The LZW algorithm is published by Terry Welch in 1984. This algorithm improved upon the original by proposing a code table that could be created the same way in the compressor and the decompressor [Mai]. This algorithm was implemented in myriad applications [Mai]. It has been integrated into a variety of mainstream imaging file formats, including the GIF, TIFF and PDF.

LZW coding is conceptually very simple. At the onset of the coding process, a string table or "dictionary" containing the source symbol to be coded is constructed [Mai]. It seeks to replace strings of characters with single codewords that are stored in a string table. Most implementations of LZW used 12-bit codewords to represent 8-bit input characters. The dictionary is 4096 locations, since that is how many unique locations you can address with a 12-bit index. The first 256 locations are initialized to the single characters (location 0 stores 0, location 1 stores 1, and so on), for the monochrome images, the first 256 locations are assigned to the grey values 0, 1, 2... 255. As new combinations of characters are parsed in the input stream, these strings are added to the string table, and will be stored in locations 256 to 4095 in the table [Mai].

The compression algorithm is as follows [Mai][Dipperstein, 2005]

Step1. Initialize table with single character strings and the encoded string with the first char if the input stream.

Step2. Read the next char from the input stream.

Step3. If char is end of the input stream goto step 6.

Step4. If concatenating the char to the encoded string produces a new string that is in the string table.

- Concatenate the char to the encoded string
- goto step 2.

Step5. If concatenating the char to the encoded string produces a new string that is not in the string table.

- Output the code for String.
- Add the new string to string table.
- Set the encoded string equal to the new char

Step6. Output code for String

The decompression algorithm is also simple [Mai][Dipperstein, 2005]:

Step1. Initialize table with single character strings

Step2. Read the first input char to the Old_code and output translation of Old_code.

Step3. Read the next input char to the New_code

Step4. If the New_code is end of input stream exit.

Step5. If the New_code is not in the string table

- Set string equal to the translation of Old_code and concatenate char to the string

Step6. If the New_code is in the string table.

- Set string equal to the translation of New_code.

Step7. Output string

Set the char equal to the first character of the string

Concatenating the char to the Old_code produces a new string.

Add new string to the string table.

Set the Old_code equal to the New_code.

Step8 goto step 3

Let us take a small image to explain the algorithm how it works. Consider the following 4 x 4, 8-bit image [Gonzalez, R. C., & Woods, R. E. (2002)]:

39 39 126 126

39 39 126 126

39 39 126 126

39 39 126 126

The image is encoded by processing its pixels in a left-to-right, top-to-bottom manner. The first step is assumed a string table.

Codeword	String
0	0
1	1
...	...
255	255
256	---
...	...
4096	---

Table.4 A String Table for LZW Algorithm

Following the above algorithm, we set String equal to 39 and Character equal to 39. We then output 39 and add 39-39 to the string table. Since 0 to 255 have been initialized to single characters in the string table, the first available entry is 256. The new String is set to 39 and we start at the top of the While loop. This process is repeated until the input stream is exhausted. As we encode the data we output codes and create a string table as shown:

Encoded output		String table	
Pixels of image	Output code	Codeword	String
39			
39	39	256	39-39
126	39	257	39-126
126	126	258	126-126
39	126	259	126-39
39			
126	256	260	39-39-126
126			
39	258	261	126-126-39
39			
126			
126	260	262	39-39-126-126
39			
39	259	263	126-39-39
126			
126	257	264	39-126-126
	126		

Table.5 A LZW Compression Table

So the output stream is

<39><39><126><126><256><258><260><259><257><126>

The LZW decompressor creates the same string table during decompression. It starts with the first 256 table entries initialized to single characters. The string table is updated for each character in the input stream, except the first one. After the character has been expanded to its corresponding string via the string table, the final character

of the string is appended to the previous string. This new string is added to the table in the same location as in the compressor's string table [Mai].

Let us decompress the compressed datas

<39><39><126><126><256><258><260><259><257><126>.

First we input the first character 39 into Old - Code and output the translation 39. We read next character 39 into New-Code. Since New-Code is in the string table we set String = 39. Then output 39. Character is set to 39 and 39-39 is the first entry in the string table. Old-Code gets set to 39 and jump to the beginning of the While loop. The process continues until we have processed all the compressed data. The decompression process yields output and creates a string table like that shown below.

Decoder output		String table	
input	Output	Codeword	String
39	39		
39	39	256	39-39
126	126	257	39-126
126	126	258	126-126
256	39-39	259	126-39
258	126-126	260	39-39-126
260	39-39-126	261	126-126-39
259	126-39	262	39-39-126-126
257	39-126	263	126-39-39
126	126	264	39-126-126

Table.6 A LZW Decompression Table

The decoder produces the string like this:

39-39-126-126-39-39-126-126-39-39-126-126-39-39-126-126.

Because the image normally is a 2-D array, so we can arrange the result like this:

39	39	126	126
39	39	126	126
39	39	126	126
39	39	126	126

The main advantage of LZW algorithm over Huffman coding is that it can compress the input stream in one single pass. This will result in greatly reducing the time of compression and decompression; because the Huffman algorithm scans the input stream two times, but the LZW algorithm just scans the input stream only once. It maybe cuts the compression and decompression time by half. Another advantage is its simplicity, allowing fast execution [Mai]. A unique feature of the LZW coding is that the coding dictionary or string table is created while the data are being encoded. Remarkably an LZW decoder builds an identical decompression dictionary as it decodes simultaneously the encoded data stream.

Here is the example using the Delphi to implement the LZW algorithm:

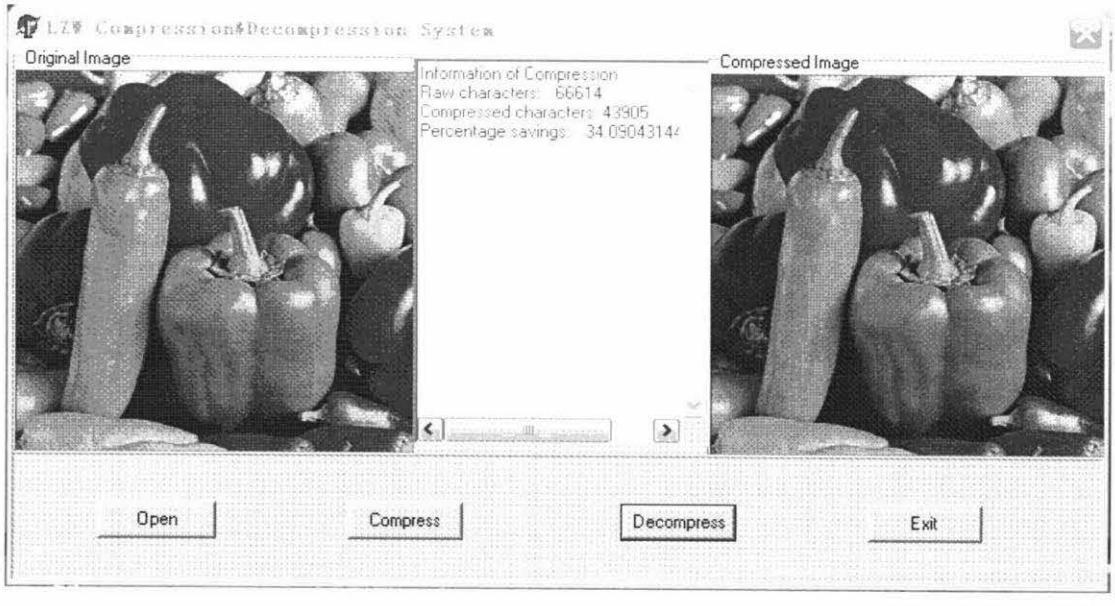
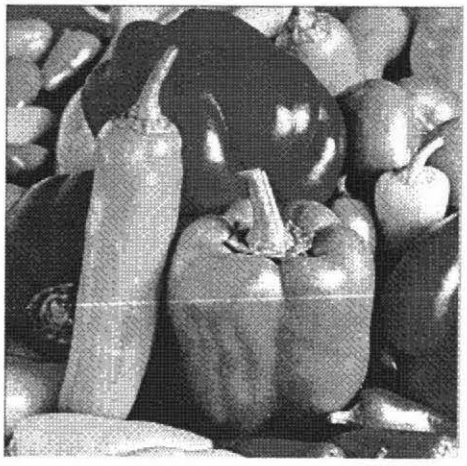
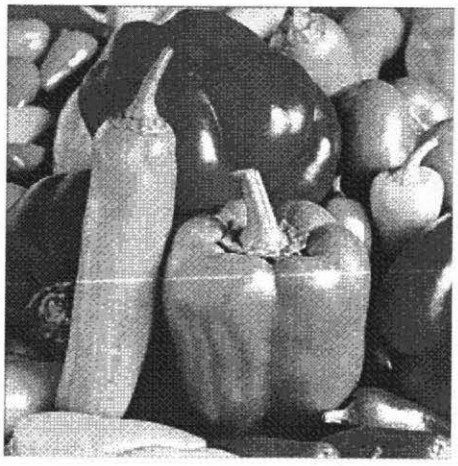


Fig.6 Delphi Implementation the LZW Algorithm



Original Image



Compressed Image

Fig.7 Comparison the between Original Image and Reconstructed Image (LZW algorithm)

2.2.5 Huffman Algorithm Vs LZW Algorithm (Results)

Image file (256 x 256)	Original File size (Byte)	Compressed file size (Byte)		Compression ratio	
		Huffman	LZw	Huffman	LZW
lena.bmp	66614	31304	45597	2.13	1.46
zelda.bmp	66614	29470	40957	2.26	1.63
sailboat.bmp	66614	32313	47267	2.06	1.41
peppers.bmp	66614	32326	43905	2.06	1.52
goldhill.bmp	66614	31584	44337	2.11	1.50
frog512.bmp	66614	28502	51721	2.34	1.29
elain.bmp	66614	31536	44046	2.11	1.51
couple.bmp	66614	29438	44045	2.26	1.51
boat.bmp	66614	29744	41847	2.24	1.59
baboon.bmp	66614	30507	56897	2.18	1.17
aerial.bmp	66614	28291	54645	2.35	1.22
barb.bmp	66614	31127	50469	2.14	1.32

Table.7 Comparison the between Huffman and LZW Algorithms

Based on the results of the experiment, it is evident that the Huffman algorithm performs better than LZW for all test images used. The compressed file size returned by the Huffman algorithm is always smaller than the values returned LZW, and the compression ratio is bigger than the LZW.

2.3 Transform coding

2.3.1 Introduction

In this section, we consider other image compression techniques that are based on modifying the transform of an image. In transform coding, a reversible, linear transform is used to map the image into a set of transform coefficients, which are then quantised and coded. A variety of transformations, including DFT, DCT, and WT, can be used to transform the image data. Fig.8 shows a typical transform coding system.

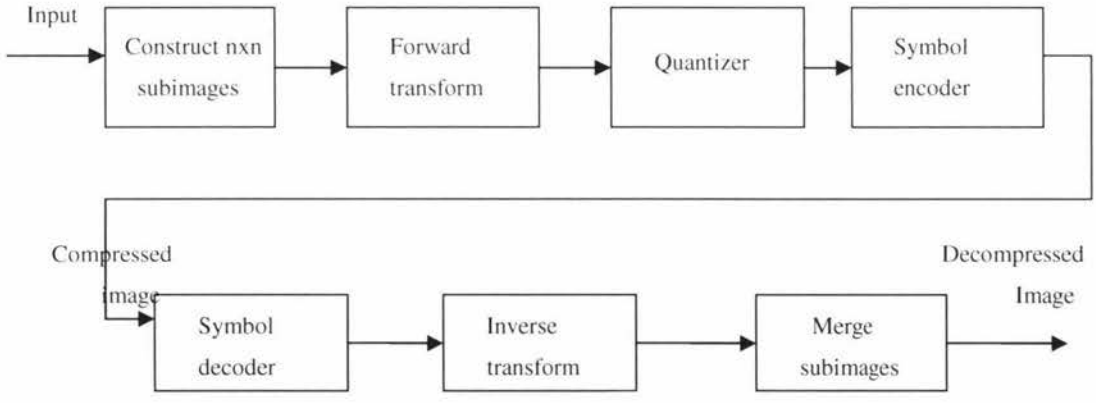


Fig.8 The Transform Coding System

Transform coding can be generalised into four stages:

- Image subdivision (some transform coding doesn't need this stage)
- Image transformation
- Coefficient quantization
- Symbol encoding

For a transform coding scheme, logical modelling is done in two steps: segmentation, in which the image is subdivided in bi-dimensional vectors (possibly of different sizes) and a transformation step, in which the chosen transform (e.g. DFT, DCT, and WT) is applied [Image Compression techniques].

Quantisation can be performed in a number of ways. Most classical approaches use "zonal coding", consisting in scalar quantisation of the coefficients belonging to a predefined area (with a fixed bit allocation), and "threshold coding", the coefficients of each block characterized by an absolute value exceeding a predefined threshold. Another possibility, that leads to higher compression factors, is to apply a vector quantization scheme to the transformed coefficients [Image Compression techniques].

In the symbol encoding step the same type of encoding algorithm is used for each transform coding method. In most cases a classical Huffman code and LZW code can be used successfully.

2.3.2 Discrete Cosine Transform

The Discrete Cosine Transform was introduced by Ahmed, Natarjan, and Rao in 1974 [Rao & Yip, 1990]. The DCT is an orthogonal transform having some features of a transformation to the frequency domain.

The 1-D discrete cosine transform (DCT) is defined as [Franti, 2002]

$$C(u) = \alpha(u)\sum_{x=0}^{N-1} f(x)\cdot\cos\left[\frac{(2x+1)u\pi}{2N}\right] \qquad [3]$$

Similarly, the inverse DCT is defined as [Franti, 2002]

$$f(x) = \sum_{u=0}^{N-1}\alpha(u)C(u)\cdot\cos\left[\frac{(2x+1)u\pi}{2N}\right] \qquad [4]$$

where

$$\alpha(u) = \begin{cases} \sqrt{1/N} & \text{for} \quad u = 0 \\ \sqrt{2/N} & \text{for} \quad u = 1,2,\ldots,N-1 \end{cases} \qquad [5]$$

The corresponding 2-D DCT, and the inverse DCT are defined as [Franti, 2002]

$$C(u,v) = \alpha(u)\alpha(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\cdot\cos\left[\frac{(2x+1)u\pi}{2N}\right]\cdot\cos\left[\frac{(2y+1)v\pi}{2N}\right] \qquad [6]$$

and

$$f(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1}\alpha(u)\alpha(v)C(u,v)\cdot\cos\left[\frac{(2x+1)u\pi}{2N}\right]\cdot\cos\left[\frac{(2y+1)v\pi}{2N}\right] \qquad [7]$$

The advantage of DCT is that it can be expressed without complex numbers. 2-D DCT is also separable (like 2-D Fourier transform), i.e. it can be obtained by two subsequent 1-D DCT in the same way than Fourier transform [Franti, 2002]. See Fig.9 for an example of basis functions of the 1-D DCT, Fig.10 for an example of the basis functions of the 2-D DCT [Franti, 2002].

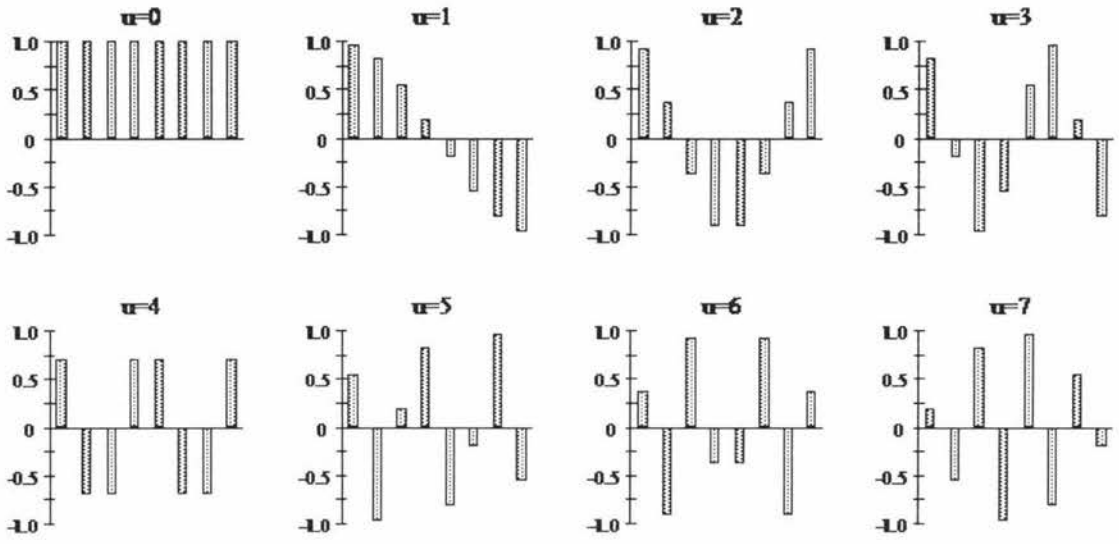


Fig.9 1-D DCT Basis Functions for N=8 [Franti, 2002].

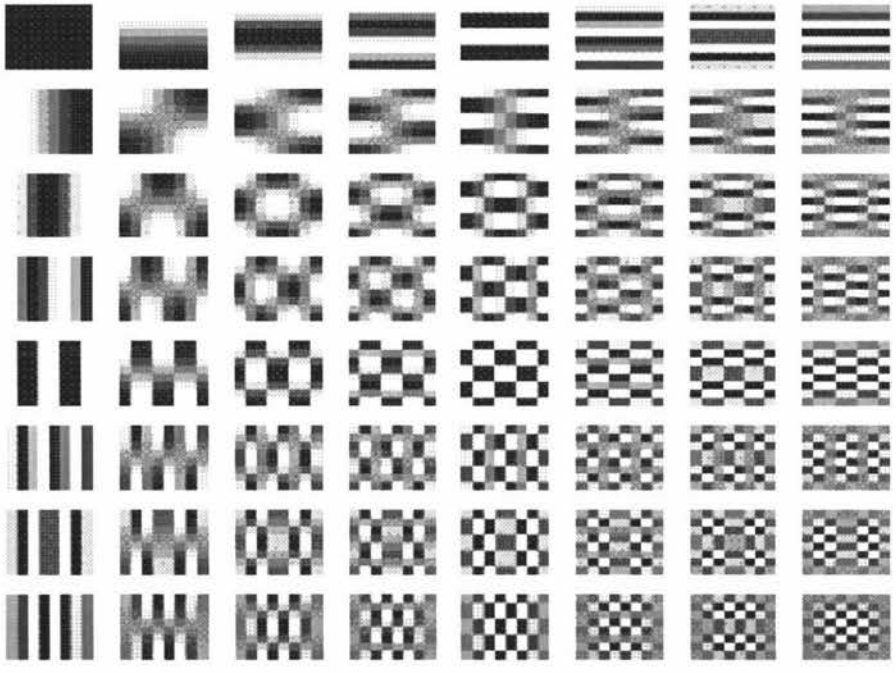


Fig.10 Basis Function of an 8x8 DCT [Franti, 2002]

2.3.3 Wavelet Transform

Wavelet transform [Polikar, 1995] analysis has emerged as a major new time-frequency decomposition tool for data analysis. The wavelet transform has been found to be particularly useful for analysing signals which are transitory,

discontinuous, noisy, and so on. Its ability to examine the signal in both time and frequency resolution is distinctive and makes a myriad of applications possible that traditional signal analysis tools such as Fourier transforms cannot handle. It has now been applied to diverse realm of data analysis or process: climate analyses, financial indices analysis, signal denoising, characterisation, feature extraction, data compression, and so on.

Wavelet transform is an unsmooth signal analysis method. The basic idea of the wavelet transform is using a set of function $\psi_{a,b}(t)$ to represent or to approach arbitrary function $f(t)$ the set of function is called wavelet function. In practice, for convenience, numbers are usually entered into wavelet transformations using base-two notation. For any arbitrary function $f(t)$ in the $L^2(R)$ space, the wavelet transform is [Qing, H., & Shu-ling, Z. (2004)]:

$$C_{m,n} = \int_{-\infty}^{+\infty} f(t)\psi_{m,n}(t)dt \qquad [8]$$

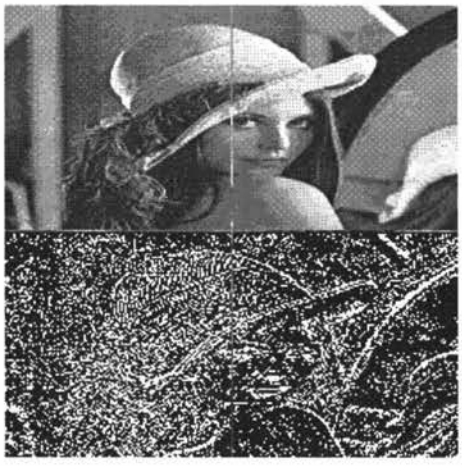
The 1-D wavelet transform is expanded to a 2-D wavelet transform for use in image processing. Through the horizontal and vertical filter, the 2-D wavelet transform decomposes the original image to horizontal, vertical, diagonal and low frequency four subbands. The low frequency subbands can continually be decomposed. See Fig.11–Fig.13 for an example of wavelet transform.

LL1	HL1
LH1	HH1

Fig.11 Wavelet Decomposition

Original Image

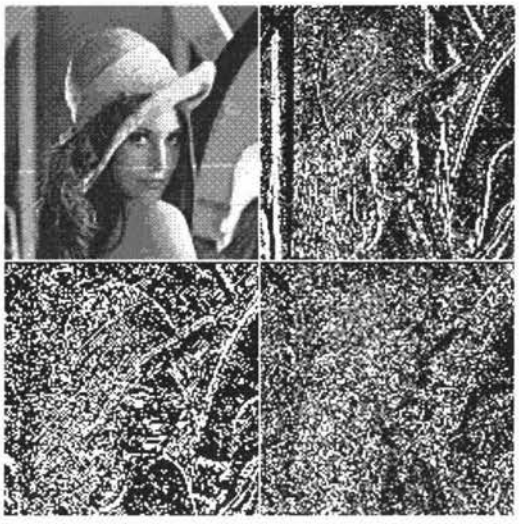
Vertical Decomposition Image



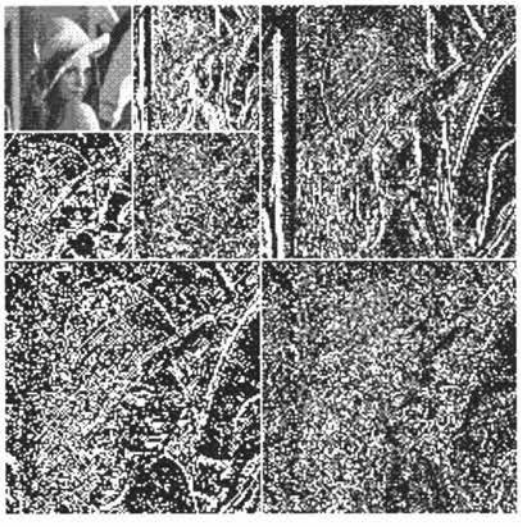
Horizontal Decomposition Image

Fig.12 Vertical and Horizontal Decomposition Image

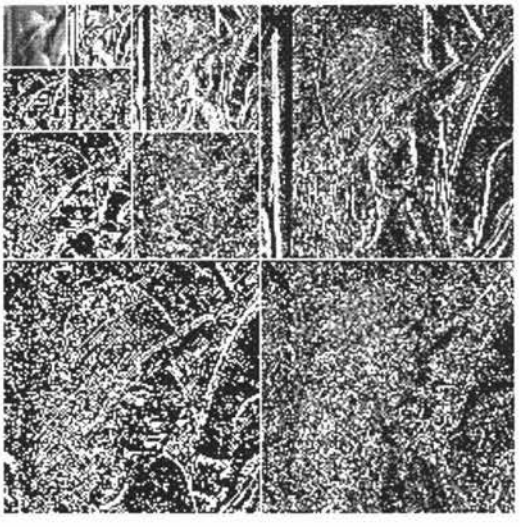
Original Image



1-Level Decomposition Image



2-Level Decomposition Image



3-Level Decomposition Image

Fig.13 1 to 3 Levels Wavelet Decomposition Image

In the next sections we discuss some of the most popular wavelets.

2.3.3.1 Haar Wavelet

Haar wavelet [Polikar, 1995] is derived from the Haar function, The Haar function is a set of orthogonal functions.

$$\varphi(t) = \begin{cases} +1, & 0 \le t < \frac{1}{2} \\ -1, & \frac{1}{2} \le t < 1 \end{cases} \qquad [9]$$

The Fig.14 shows the wave shape of Haar wavelet [TheMathWorks]:

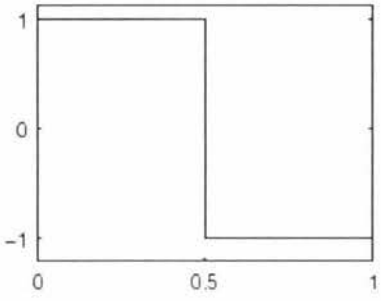


Fig.14 Haar Wavelet [TheMathWorks]

The haar wavelet is discontinuous in the time domain, so the property is not suitable for use as a core function. The main advantage of the wavelet is that computation is simple. The key properties of the wavelet are:

- Orthogonal
- Compact support
- The scaling function is symmetric
- The wavelet function is anti-symmetric
- It has only one vanishing moment (a minimum)

2.3.3.2 Morlet Wavelet

Morlet wavelet [Polikar, 1995] is a modulated Gaussian: $\varphi(t) = e^{-t^2} * e^{-2\alpha\pi}$. it satisfies the admissibility condition only approximately. The error can be neglected numerically if α > 5.5. The Fig.15 shows the wave shape of Morlet wavelet [TheMathWorks]:

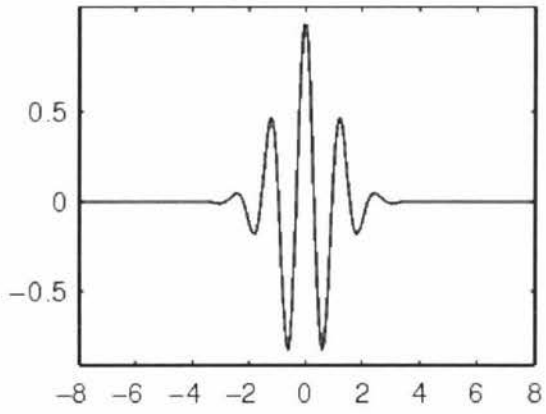


Fig.15 Morlet Wavelet [TheMathWorks]

2.3.3.3 Marr Wavelet

Marr wavelet [Polikar, 1995] is the second derivative of the Gaussian. It satisfies the admissibility condition and has two vanishing moments.

$$\psi(t) = \left(t^2 - 1\right)e^{-\frac{1}{2}t^2} \qquad [10]$$

The Fig.16 shows the wave shape of Marr wavelet [TheMathWorks]:

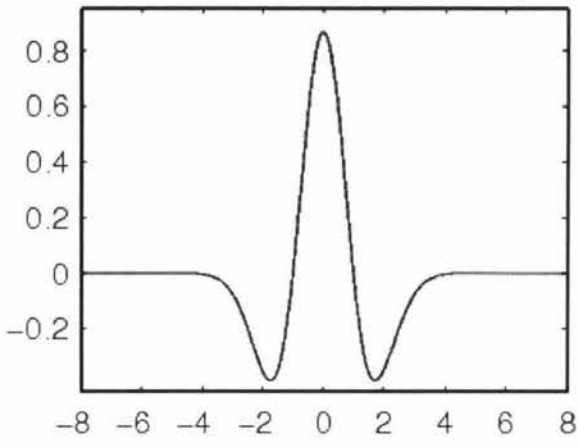


Fig.16 Marr Wavelet [TheMathWorks]

2.3.4 Typical Image Compression Algorithms based on Wavelet Transform

2.3.4.1 EZW Algorithm [Valens, 1999-2004]

1. Algorithm principle:

Coding is embedded in the encoder to allow the image information in the bit stream to be treated differently according its level of importance; it can terminate the encoding at any point thereby allowing a target rate or target distortion metric to be met exactly. This means that when more bits are added to the stream, the decoded image will contain more detail information. (For example, like π, every digit we add would mean an increase in the accuracy of the number; however, we can limit the accuracy at any decimal place we desire.) [Shapiro, J. M. (1993)]. The order of transmission, consistent with the embedded coding, is from the most important (highest) position, to the least important (lowest) position. Progressive encoding is also known as embedded encoding

One image passed through three levels of wavelets decomposes to produce ten subbands, like what Fig.17 shows. The wavelet coefficient distribution characteristic is that coefficients are bigger in low frequency, and also includes more image information. As shown in the subband in Fig.17 coefficients are smaller in the high

frequency subband and include less image information. The low frequency subband reflects important information from the image, and it is therefore important to human perception of the image. The high frequency subbands reflect less important image information, and are therefore not of great importance to human perception. Because the wavelet coefficient has these characteristics, it complements the embedded image coding algorithm

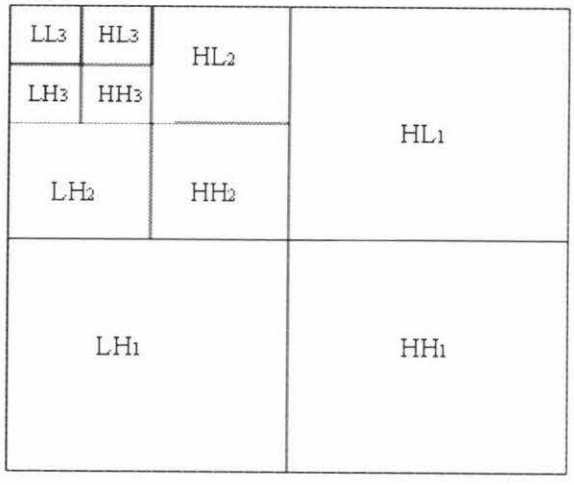


Fig.17 Three Level Wavelet Decomposition

2. EZW algorithm

The EZW algorithm is a simple, yet remarkably effective image compression algorithm. It has the property that the bits in the bit stream are generated in order of importance yielding a fully embedded code [Shapiro, 1993]. (A zerotree is a quad-tree of which all nodes are equal to or smaller than the root. A quad-tree is a tree where each leaf corresponds to a uniform part of the image and each interior node has exactly four children.)There are four key concepts in EZW algorithm [Shapiro, 1993]:

- A wavelet transform or hierarchical subband decomposition.
- Prediction of the absence of significant information across scales by exploiting the self-similarity inherent in images.
- Entropy-coded successive-approximation quantization .

• Universal lossless data compression which is achieved via adaptive arithmetic coding

The first step in the EZW coding algorithm is to calculate the initial threshold *th* [Valens, C. (1999-2004)].

$$th = 2^{\lfloor \log_2 (Max (|f(x,y)|)) \rfloor} \qquad [11]$$

Here Max (...) is the maximum coefficient value in the image and f (x,y) denotes the coefficient. The EZW algorithm is as follows [Shapiro, 1993].

Step1. The entire input image is input into the discrete wavelet transform, then produces the wavelet coefficients.

Step2. Calculating the initial threshold *th* (Equation [11]).

Step3. If the *th* is larger then zero.

- All the wavelet coefficients pass the dominant_pass() function;
- Set the *th* = *th* /2;
- A part of wavelet coefficients pass the subordinate_pass() function;

Step4. goto step 3

Two passes are used to code the image in EZW algorithm. In the first pass, or the dominant pass, the coefficients are scanned in raster order or morton order (see Fig.19) within the subbands. The scan starts with the subbands of the highest transform level. In each transform level, the subbands are scaned in the order LL, HL, LH, and HH. The coefficients are coded by symbol P, N, ZTR, and IZ [Shapiro, 1993].

• P, if the coefficient is greater than the given threshold and it is positive [Valens, C. (1999-2004)].

• N, if the absolute value of coefficient is greater than the given threshold and it is negative [Valens, C. (1999-2004)].

• ZTR, if the absolute value of coefficient is small the given threshold and the absolute value of all coefficients in the corresponding quad tree are smaller than the threshold too [Valens, C. (1999-2004)].

• IZ, if the absolute value of coefficient is smaller than the given threshold and there exists at least one coefficient in the corresponding quad tree that is greater than the given threshold with respect to the absolute value [Valens, C. (1999-2004)].

In the subordinate pass, each coefficient that has been code as P or N in the previous dominant pass, is now refined. If the coefficient is belong to [T, T+T/2) then output the 0, if the coefficient is belong to (T+T/2, 2T] then output the 1. (T=th/2). Fig.18 shows the flow chat for encoding a coefficient [Shapiro, 1993].

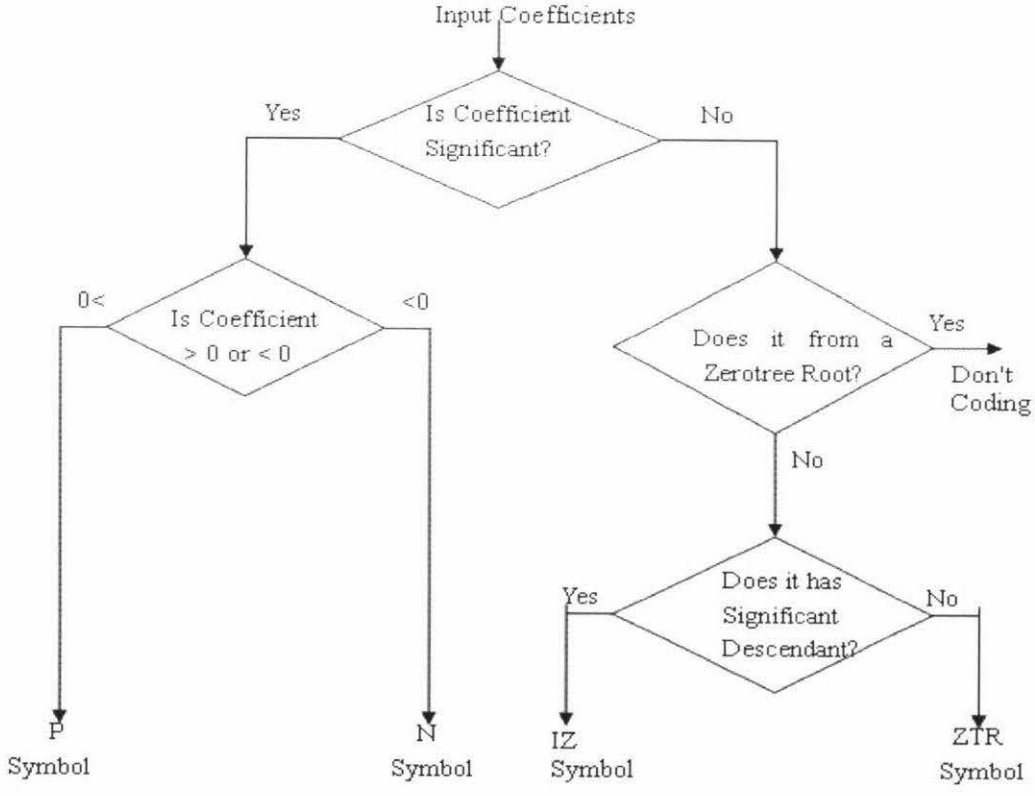


Fig.18 Flow Chart for Encoding a Coefficient [Shapiro, 1993]

3. Algorithm analysis

The core idea of EZW is to unceasingly scan the image to produce many zerotrees to code image. At the each scan , all the coefficients that are in absolute value larger than the current threshold are extracted and placed without their sign on the subordinate list and their positions in the image are filled with zeroes. This will prevent them from being coded again.

The existing problems of the EZW algorithm are:

The EZW algorithm forms many zerotrees, and consequently must scan the coefficients many times, reducing the efficiency of the algorithm. Moreover, each zerotree must form after the preceding tree has formed. It is therefore very difficult to optimize the algorithm by forming zerotrees concurrently.

The EZW algorithm executes the same important coding on all frequency subbands; it cannot fully use the characteristic of wavelet transformation. One of the improvements is that EZW separates the lowest frequency subband from other subbands, and carries on lossless coding for the lowest frequency subband.

The relationship between zerotrees has not been taken into account by the EZW algorithm, and is a major disadvantage of the algorithm.

Through analysis of wavelet coefficients, we find a relationship between pixels within the same subband. It is possible to achieve compression by organizing low-value coefficients together, especially those in the high-frequency subband. The EZW does not effectively make use of this relationship.

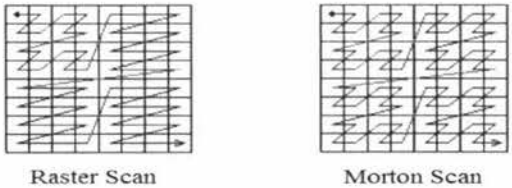


Fig.19 Two Kinds of Scan Ordering [Shapiro, 1993]

For example [Shapiro, 1993]. Encode the wavelet transform image given in Fig.20 using the EZW algorithm, at the first step we need obtain the initial threshold. According to the Equation [11]:

The initial threshold $th = 32$.

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

Fig.20 Coefficients

Thus, in the first dominant pass, the encoding starts as follows:

Scanning LL3: read 63 => output P

Scanning HL3: read -34 => output N

Scanning LH3: read -31 => output IZ

Scanning HH3: read 23 => output ZTR

Sacnning HL2: read 49 => output P

read 10 => output ZTR

read 14 => output ZTR

read -13 => output ZTR

Scanning LH2: read 15 => output ZTR

read 14 => output IZ

read -9 => output ZTR

read -7 => output ZTR

Scanning HL1: read 7 => output ZTR

read 13 => output ZTR

read 3 => output ZTR

read 4 => output ZTR

Scanning LH1: read -1 => output ZTR

read 47 => output P

read -3 => output ZTR

read 2 => output ZTR

In the subordinate pass we have to refine four coefficients, which were encoded with the symbols P and N in the previous dominant pass. After the subordinate pass producing the output:

1 0 1 0

The process continues on the second dominant pass at the new threshold of 16. During this pass, it produces the symbol string:

IZ->ZTR->N->P->ZTR->ZTR->ZTR->ZTR->ZTR->ZTR->ZTR->ZTR

The second subordinates pass produces the output:

100110

The processing continues alternating between dominant and subordinate passes and it can stop at any time or until the threshold is equal to the zero [Shapiro, 1993].

2.3.4.2 SPIHT Algorithm

1. Algorithm principle:

EZW is a kind of embedded image coding algorithm based on zerotrees. The zerotree is a more efficient structure for representing the insignificant coefficient. Considering the structures of zerotrees, only the tree root is of significance to EZW, other nodes are insignificant. Therefore EZW does not take full advantage of the structure of the zerotree. A.Said and W.A.Pearlman according to the basic idea of EZW to develop a new and more efficient algorithm: SPHIT (Set Partitioning in Hierarchical Trees). It uses the SOT(Spatial Orientation Tree), set D(i,j) and L(i,j) to present this kind of tree structure(mention in before),thereby improve the efficient of coding.

The following concepts are used in the SPHIT algorithm:

- H: set of coordinates of all spatial orientation tree roots.
- Z(i,j): set of coordinates of all offspring of node (i,j) and node (i,j).
- O(i,j): set of coordinates of all offspring(immediate children) of node (i,j).
- D(I,j): set of coordinates of all descendants of node (i,j).
- L(i,j) = D(i,j) – O(i,j).

Improves the basic zerotree coding algorithm of EZW by using slightly different tree structures, the new structure is called spatial orientation tree, the set partitioning rules are simply:

- Z(i,j) = c(i,j) + D(i,j)

- $D(i,j)=O(i,j)+L(i,j)$

- $L(i,j)=\sum D(k,l)$ $(k,l)\in O(i,j)$

2. Sorting procedure

SPIHT uses three lists, called list of insignificant sets (LIS), list of insignificant pixels (LIP), and list of significant pixels (LSP). In all lists each entry is identified by a coordinate (i,j), which in the LIP and LSP represents individual pixels, and in the LIS represents either the set D(i,j) or L(i,j). The sorting pass checks the significance of the LIP and LIS, and moves significant coefficients to the LSP. When a significant coefficient is a member of a set, the set partitioning rule is used to split the set.

SPHIT algorithm [Miguel, 1999]

Step1. Compute the threshold. Initialize the three lists: LIP to all root nodes coefficients, LIS to all tress, and LSP to an empty set.

Step2. Check the significance of all coefficients in LIP:

If significant, output 1 followed by a sign bit and move it to the LSP.

If not significant, output 0.

Step3. Check the significance of all trees in the LIS according to the type of set:

·for a set of type D:

·If it is significant, output 1, and code its children:

·if a child is significant, output 1, then a sign bit and add it to the LSP

·if a child is insignificant, output 0and add it to the end of LIP.

·if the children have descendants, move the tree to the end of LIS as type L, otherwise remove it from LIS

·if it is insignificant, output 0.

·for a set of type L:

·if it is significant, output 1, add each of the children to the end of LIS as an entry of type D and remove the parent tree from the LIS.

·if it is insignificant, output 0.

Step4. Decrease the threshold and go back to step 2.

The initialisation process and the split process of SPIHT algorithm are similar to those of the EZW algorithm, but have improved the expression method for EZW's zerotree. Also of importance is the change to how coefficients order information in tables, which causes the expression to become more simplified, thus enhancing the coding efficiency.

2.4 Fuzzy Logic and Fuzzy Image Processing

2.4.1 What is Fuzzy Logic?

Fuzzy logic is an extension of Boolean logic dealing with the concept of partial truth. Whereas classical logic holds that everything can be expressed in binary terms (0 or 1, black or white, yes or no). Fuzzy logic replaces Boolean truth values with degree of truth [Tizhoosh, H. R. (1997)].

Degrees of truth are often confused with probabilities, although they are conceptually distinct, because fuzzy truth represents membership in vaguely defined sets, not

likelihood of some event or condition. Fuzzy logic is mainly responsible for representation and processing of vague data (ill-defined, fuzzy). Probability theory is mainly representation and processing of uncertainty (randomness). Following table clarifies the different between the two theories [Tizhoosh, H. R. (1997)].

Probability Measure	Membership Function
Calculates the probability that an ill-known variable X ranging on U hits the well-known set A	Calculates the membership of a well-known variable X ranging on U hits the ill-known set A
Before an event happens	After it happened
Measure Theory	Set Theory
Domain is 2U(Boolean Algebra)	Domain is [0,1]U(cannot be a Boolean Algebra)

Table.8 Comparing the Probability and Fuzzy Theory [Tizhoosh, H. R. (1997)]

2.4.2 What is Fuzzy Set Theory?

Fuzzy set theory is the extension of conventional (crisp) set theory. It handles the concept of partial truth (truth values between 1 (completely true) and 0 (completely false)). It was introduced by Prof. LotfiA.Zadeh of UC/Berkeley in 1965 as a mean to model the vagueness and ambiguity in complex system [Tizhoosh, H. R. (1997)].

The idea of fuzzy sets is simple and natural. For example [Tizhoosh, H. R. (1997)], we want to define a set of gray levels that share the property dark. In classical set theory, we define a threshold is 100. All gray levels between 0 and 100 are element of the set; the others do not belong to the set (left image in Fig.21). But the darkness is the matter of degree. So, a fuzzy can model this property much better. To define this set, we also need two thresholds 50 and 150. All gray levels that are less than 50 are the full member of the set, all gray levels that are greater than 150 are not the member of the set. The gray levels between 50 and 150, however, have a partial membership in the set (right image in Fig.21) [Tizhoosh, H. R. (1997)].

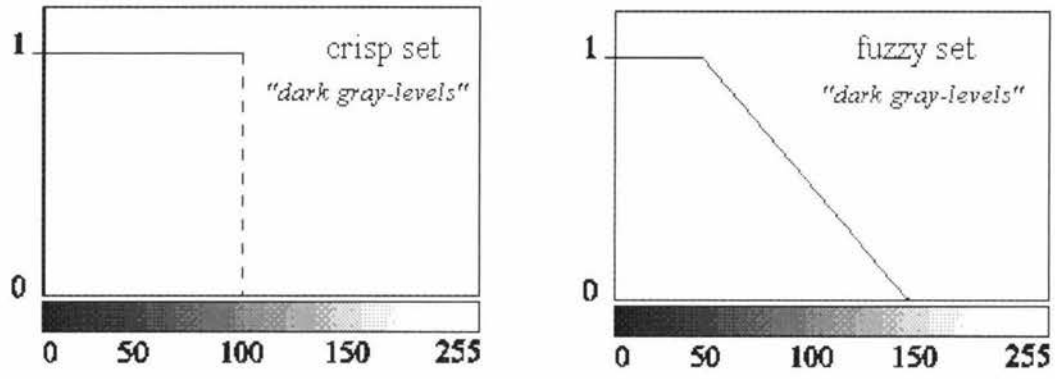


Fig.21 Representation of "dark gray-levels" with a Crisp and a Fuzzy Set [Tizhoosh, H. R. (1997)]

2.4.3 Fuzzy Membership Functions

The fuzzy membership function is selected by trial and error. There are four basic membership functions namely: Triangular, Trapezoidal, Gaussian and Generalized bell [Short Term Load Forecasting Using Neural Networks and Fuzzy logic].

The Triangular membership function defined as:

$$trimf\,(x;a,b,c) = Max\left(Min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b} \right), 0 \right) \qquad [12]$$

It has three parameters a (minimum), b (middle) and c (maximum) that determine the shape of the triangle.

Fig.22 shows the triangular function of triangle (x, 10,50,90):

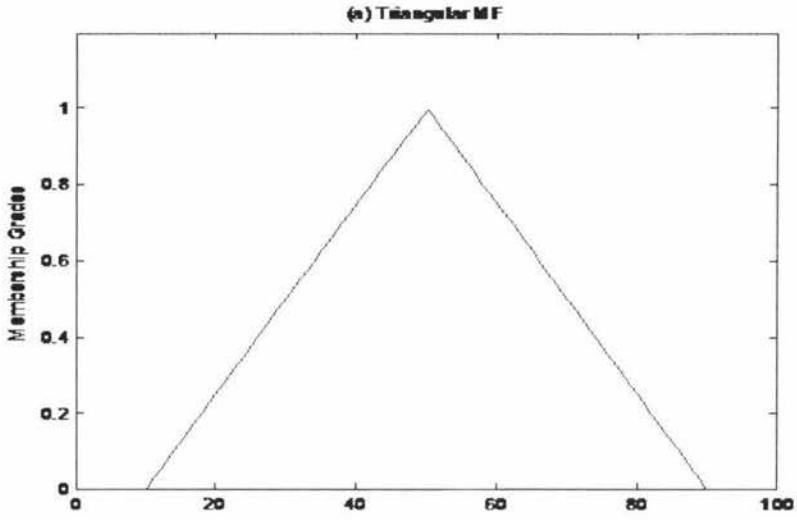


Fig.22 The Triangular Membership Function

The Trapezoidal membership function defined as:

$$trapmf\ (x; a, b, c, d) = Max\left(Min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c} \right), 0 \right) \quad [\ 13\ ]$$

Fig.23 Trapezoidal function of trapezoid (x, 10, 20, 60, 95):

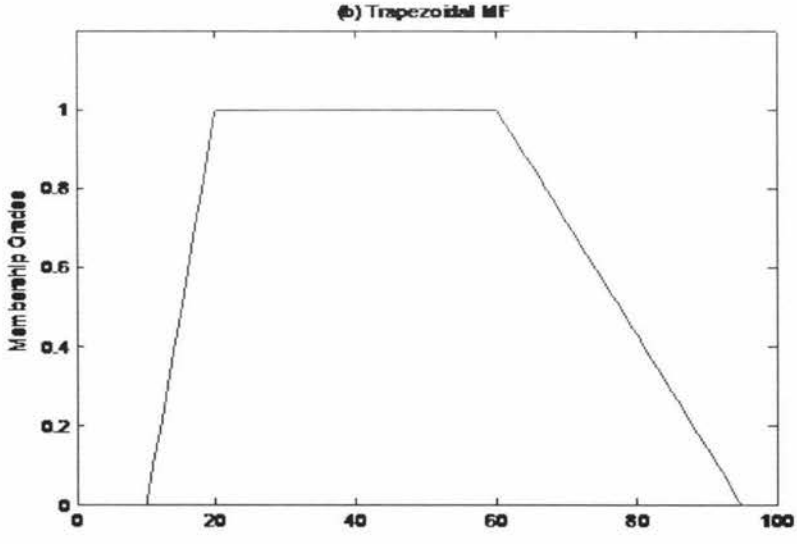


Fig.23 The Trapezoidal Membership Function

The Gaussian membership function defined as:

$$gaussmf(x;a,b,c) = e^{-\frac{1}{2}\left(\frac{x-c}{\sigma}\right)^2} \qquad [14]$$

Fig.24 Gaussian function of Gaussian (x, 50,20):

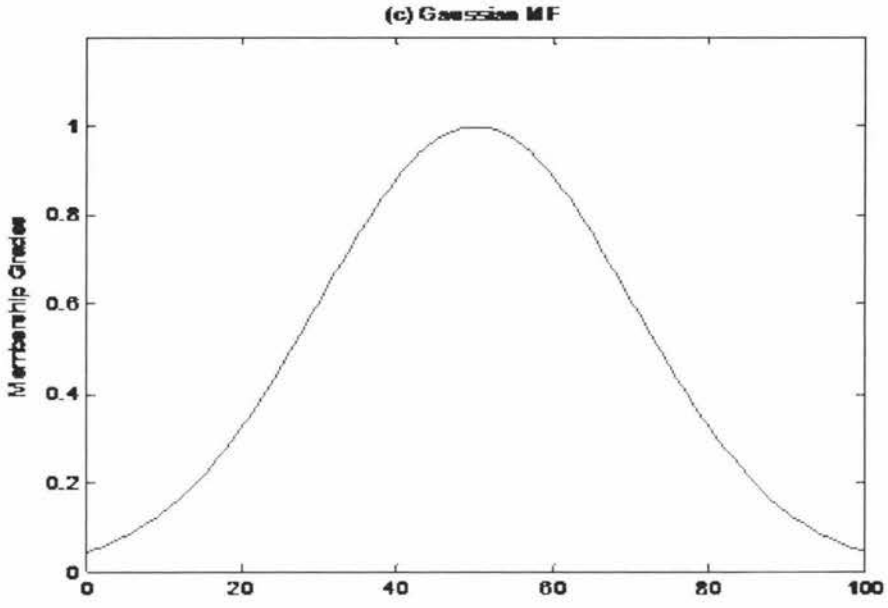


Fig.24 The Gaussian Membership Function

The Generalized bell membership function defined as:

$$gbellmf(x;a,b,c,d) = \frac{1}{1+\left|\frac{x-c}{b}\right|^{2b}} \qquad [15]$$

Fig.25 Generalized bell function of Generalized_bell(x, 20, 4, 50):

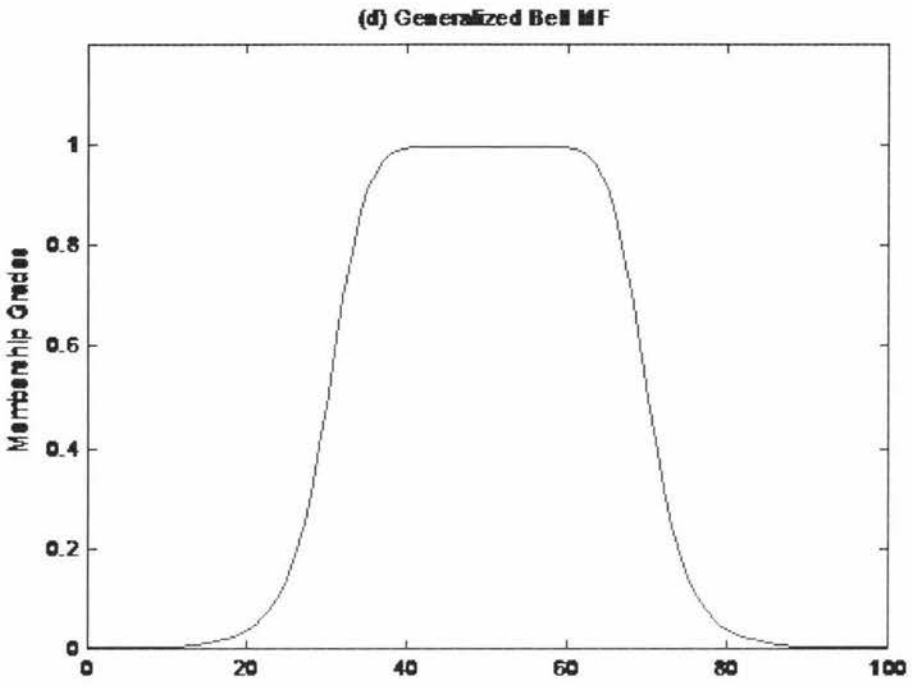


Fig.25 The Generalized bell Membership Function

2.4.4 Applications of Fuzzy Techniques in Image Processing

In general, there are three image processing based on fuzzy techniques: Fuzzy Image Enhancement, Fuzzy Image Segmentation and Fuzzy Edge Detection. [Tizhoosh, H. R. (1997)]

2.4.4.1 Fuzzy Image Enhancement.

Fuzzy image enhancement is based on the gray-level mapping into a fuzzy plane, using a membership transformation [Tizhoosh, H. R. (1997)]. The aim is to generate an image of higher contrast than the original image. An image f(x,y) of size M x N and L gray levels can be considered as an array of fuzzy singleton, each having a value of membership denoting its degree of brightness relative to some brightness levels [Hassanien & Badr, 2003]. The principle of fuzzy enhancement scheme is illustrated in Fig.26.

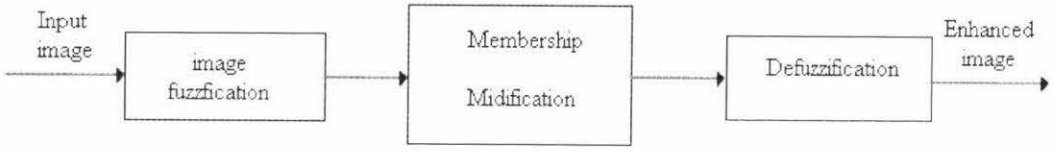


Fig.26 The Main Principle of Fuzzy Image Enhancement [Hassanien & Badr, 2003]

In recent years, many researchers have applied the fuzzy set theory to develop new techniques for contrast improvement [Hassanien & Badr, 2003]. In this thesis only mention the name of five algorithms is:

1).Contrast Improvement with INT-operator.

2).Contrast improvement using Fuzzy Expected Value.

3).Contrast Improvement with Fuzzy Histogram Hyperbolization.

4).Contrast Improvement based on Fuzzy if-Then Rules.

5).Locally Adaptive Contrast Enhancement.

2.4.4.2 Fuzzy Image Segmentation.

The different theoretical components of fuzzy image processing provide us with diverse possibilities for development of new segmentation techniques. In this thesis only mention some of fuzzy approaches to image segmentation [Tizhoosh, H. R. (1997)]:

1). Fuzzy Clustering Algorithms.

2). Fuzzy Rule-Based Approach.

3). Fuzzy Integrals.

4). Measures of Fuzziness and image information.

5). Fuzzy Geometory.

2.4.4.3 Fuzzy Edge Detection.

In this section, only mention the approaches to image edge detection [Tizhoosh, H. R. (1997)].

1). Definition of appropriate membership function.

2). Rule-Based fuzzy Edge detection.

Chapter 3

Review of Related Literature

3.1 YCC Colour Space and Image Compression

In [Bourke, 2000], the YCC colour space and an Image compression algorithm based on YCC colour space are introduced. This algorithm's core idea is that it reduces the stored space of two chrominance channels to get the goal of image compression.

Consider an RGB image with Nx horizontal pixels and Ny vertical pixels. If each pixel is represented as one byte, the image size in bytes is Nx×Ny×3. Consider storing the image in YCC space where the luminance channel Y is stored as one byte for each pixel but the two chrominance channels are only stored for each block of say 4x4 pixels. The resulting image would be Nx Ny + 2 (Nx/4 + Ny/4) [Bourke, 2000].

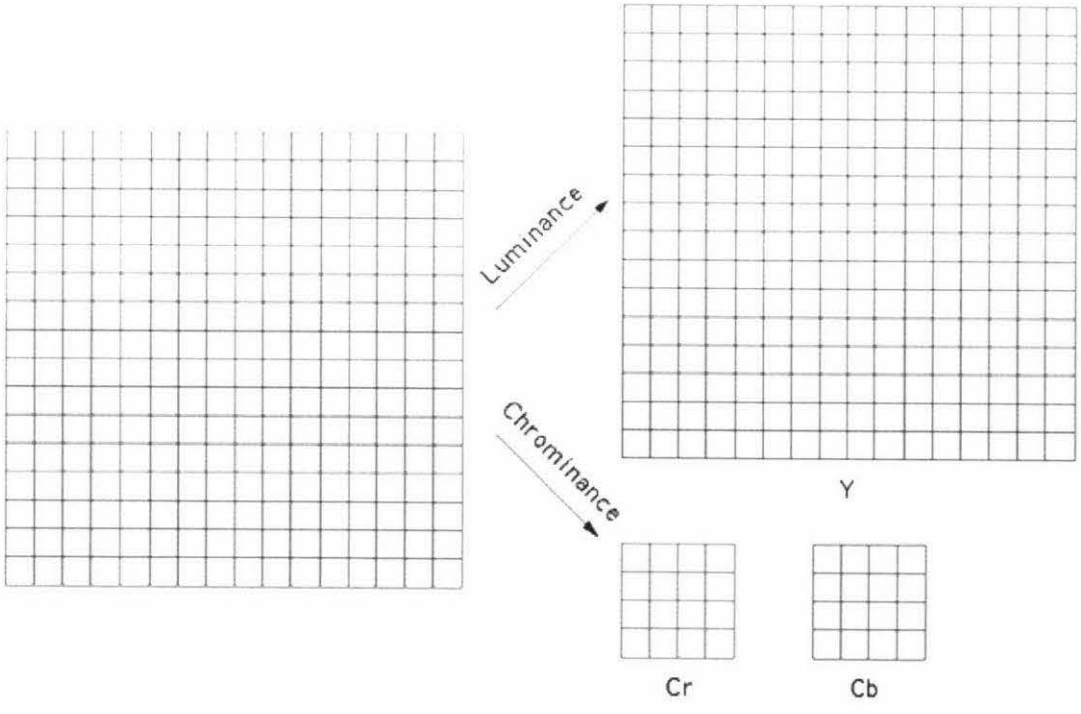


Fig.27 Basic Idea of Compression Scheme [Bourke, 2000],

3.2 JPEG Compression Techniques

JPEG [Gonzalez, R. C., & Woods, R. E. (2002)] is the image compression standard developed by Joint Photographic Expert Group. It works best on the natural images. Baseline JPEG compression consists of five basic steps:

- Transform image to YUV colour space.
- Reduce the colour componments (optional).
- Division image into 8 x 8 pixel blocks and perform the DCT on each block.
- Quantize resulting DCT coefficients.
- Entropy codes the reduced coefficients (normally using Huffman and LZW algorithm).

JPEG compression takes advantage of a limitation of the human visual system. The human eye can perceive small changes in brightness better than small changes in colour [Mai]. This allows JPEG to remove some colour information.

The Fig.28 shows the JPEG process.

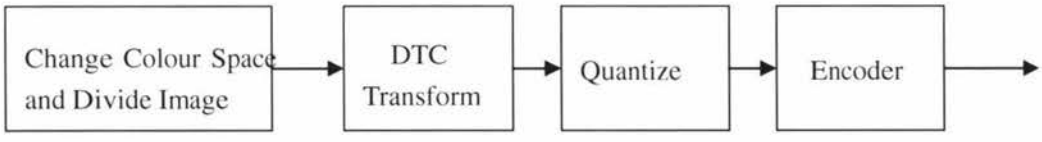


Fig.28 Block Diagram of JPEG Compression

3.3 JPEG 2000: The Next Compression Standard Using Wavelet Technology

JPEG2000 [Gonzalez, R. C., & Woods, R. E. (2002)] is an ISO specification (ISO/IEC 15444) for a wavelet based lossy compressed format for storing images. It is intended to superceed traditional JPEG format for many applications, providing better compression and a more flexible imaging model.

JPEG can be called the father of JPEG2000, but JPEG soon proved to have limitations. The compression system is lossy, meaning that whenever an image is compressed and decompressed. It loses data. This eventually affects the image quality, and can distort the image. Another limitation of JPEG is it is inability to handle sharp edges within the image. Sharp edges created by text and geometric shapes would appear fuzzy and murky. The stronger the compression rate, the worse it got [Elzinga & Feenstra, 2001].

JPEG2000 was created to address these specific problems, JPEG200 uses wavelet transform, complex mathematical formulas representing image data, to compress image at a high rate with a small of data. This allows higher quality images that can handle sharp edges better and reduce the amount of lost data form compression. JPEG2000 has a long list of features, a subset of which is [Marcellin, Gormish, Bilgin, & Boliek, 2000]:

- State-of-the art low bit-rate compression performance
- Progressive transmission by quality, resolution, component, or spatial locality
- Lossy and lossless compresson
- Region of interest coding by progression
- Limited memory implementation.

The Fig.29 shows the JPEG2000 process.

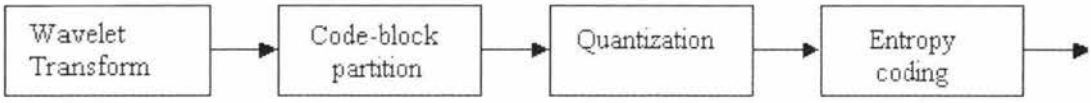


Fig.29 Block Diagram of JPEG2000 Compression

3.4 Image Compression Techniques

This paper introduces the basic techniques of image compression; it also introduces some lossless coding techniques and lossy coding techniques. (See Sec.2 for more details).

3.5 Image Compression Method of Vector Coding Based on Wavelet Transform

In [Hassanien & Badr, 2003], a new vector classified searching method is developed, it can accelerate searching speed to the code book. The number of code book vector will vary with the differential image and the classified result is modified self-adaptively under the distorted value is given. After code book is found, the vectors are organized according to subband sequences, and use the Run Length Coding and Entropy Coding.

The Fig.30 shows the image compression process [Hassanien & Badr, 2003]

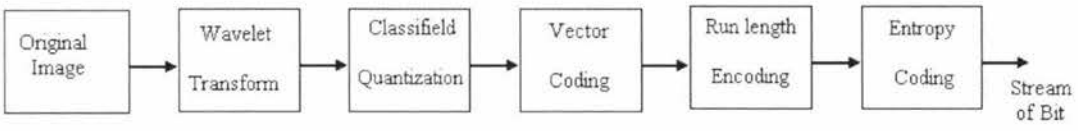


Fig.30 The Image Compression Process [Hassanien & Badr, 2003]

3.6 Embedded Image Coding Using Zerotrees of Wavelet Coefficients

In [Shapiro, 1993], an EZW algorithm based on the wavelet transform is introduced in detail. The EZW is the typical image compression algorithm based on the wavelet transform. The paper also introduces a series of image compression algorithms based on the wavelet transform according to the EZW algorithm (SPHIT). See Sec. 2.3.4 for more details.

3.7 A Comparative Study on Digital Mamography Enhancement Algorithm Based on Fuzzy Theory

In [Hassanien & Badr, 2003], some algorithms of fuzzy image enhancement has been introduced. The fuzzy image enhancement is based on gray level mapping into a fuzzy plane, using a membership transformation function. The aim is to generate an image of higher contrast than the original image. In this paper, five algorithms for the image enhancement were presented [Hassanien & Badr, 2003].

- Possibility Distribution Algorithm [Hassanien & Badr, 2003] (See Sec.4.4.1)

- Contrast Improvement with Intensification Operator [Hassanien & Badr, 2003] (See Sec.4.4.2)

- Contrast Improvement with Fuzzy Histogram Hyperbolozation [Hassanien & Badr, 2003] (See Sec.4.4.3)

- Contrast Improvement based on fuzzy If-Then Rules [Hassanien & Badr, 2003] (See Sec.4.4.4)

- Local Adaptive Contrast Enhancement [Hassanien & Badr, 2003]

The principle of fuzzy enhancement scheme used in this paper is illustrated in Fig.26. These researches served as the basis in determining the most suitable fuzzification function (Eq. 25) for IWF.

3.8 An algorithm for Image Clustering and Compression

In [KAYA, 200], a new method for image compression based on fuzzy clustering is introduced. The new image compression algorithm includes six steps [KAYA, 200]:

- Pre-filtering
- Fuzzy image enhancement
- Separation of image to 4x4 blocks and transformation of each block by using discrete cosine transform
- Selection of peak values of membership functions from transformed 4x4 blocks by zig-zag method
- Finding of membership values, and cluster centroids by cosine membership functions
- Creation of segmented image, and one dimension Run Length Coding

The new image clustering and compression method based on fuzzy logic and discrete cosine transform provides betters compression ratio and performing time.

3.9 Image Coding Using Wavelet Transform

In [Antonini, Barlaud, Mathieu, & Daubechies, 1992], a new method for image compression has been introduced, this new image compression algorithm includes three steps [Antonini, Barlaud, Mathieu, & Daubechies, 1992]:

- It uses a wavelet transform to get a set of biorthogonal subbands of images

- According to Shannon's rate distortion theory , the wavelet coefficients are vector quantized using a multiresolution codebook

- It uses a progressive transmission

3.10 On Embedded Zerotree Wavelets Coding and other Improved Algorithms

In [Wei-guo & Bao-long, 2000], the EZW algorithm and other improved algorithms based on the EZW. Like SPHIT, SPECK and CREW were introduced. This paper also discuss the principles and performances of these algorithms, it explains the research tendency in the area of embedded image coding.

3.11 Wavelet-based Image Compression

In [Walker], it concentrates on wavelet-based lossy compression of gray-scale still image. It introduces the general idea of image compression based on the wavelet transform, and also introduces some typical image compression algorithms (EZW, SPHIT, WDR, and ASWDR).

3.12 EZW Encoding

In [Valens, 1999-2004] the implementation of an EZW (Embedded Zerotree Wavelet) encoder is explained in detail.

3.13 Colour Image Compression/Reconstruction by YUV Fuzzy Wavelets

In [Nobuhara & Hirota, 2004], a new algorithm based on the Fuzzy Wavelet is presented, it also discusses the compression of original image and reconstruction compressed image on different colour spaces (RGB and YUV colour space). The

paper comes to the conclusion that the image compression and reconstruction on YUV colour space is more efficient on the RGB colour space.

Chapter 4

The Image Compression Algorithm based on Wavelet and Fuzzy Logic (IWF)

4.1 Central Thesis

An image is passed through the wavelet transform and is decomposed to produce subbands. The wavelet coefficient distribution characteristic is that coefficients are large at low frequency, and also include more image information. The coefficient is smaller in the high frequency subband and less image information is included. The low frequency subband reflects the low frequency information of the image, it is important to human perception, so we can use the loseless algorithm for compression this part of the coefficients (Huffman algorithm or LZW algorithm). The high frequency subbands reflect the high frequency information of image, and is therefore of reduced importance to human vision. The IWF algorithm is then concerned with this part of the coefficient.

The high frequency subband's coefficients uses the formula (Equation [24]) is used for mapping the coefficients to the [0..255] range, after that we can use fuzzy logic techniques to blur the coefficients within the [0..1] range. After these steps, most coefficients in high frequency subbands will represent the same value. Therefore we can easily perform the coding of the coefficients in the high frequency subbands. Lastly, the Huffman or LZW algorithm is employed to compress again.

4.2 General System Architecture

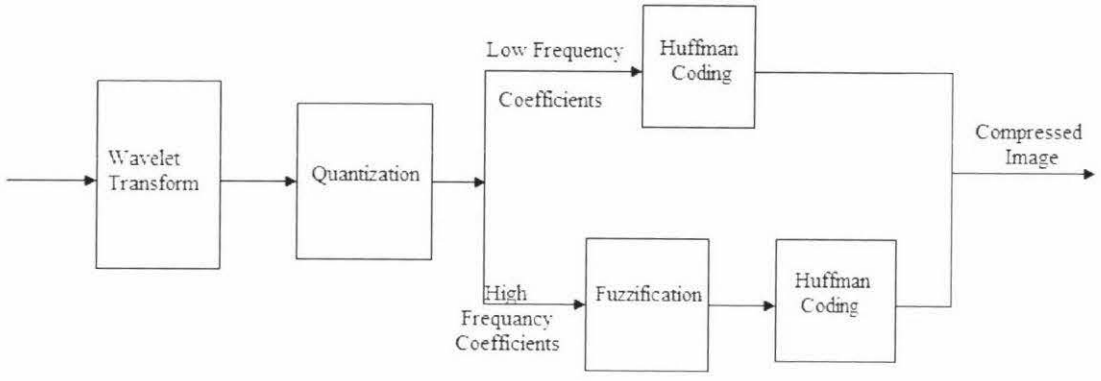


Fig.31 General Architecture of a Fuzzy and Wavelet-Based Image Compression

The figure depicts the general components of a Fuzzy and wavelet-based image compression algorithm. Initially, the whole image is decomposed by performing a wavelet transform, returning the wavelet coefficients. Next, the wavelet coefficients are quantized using Eqn (23) to derive values representing each pixel within the range [0, 255]. Based on the level of decomposition, the different subbands are encoded in various ways. The bulk of image details would always spring from the low-subband and so the coefficients in this subband is encoded without undergoing fuzzification anymore. The rest of the other subbands (high frequency) would contain not so important details, and are therefore fuzzified to reduce the number of coefficients and increase image compression. The last components are two Huffman coding components, one for encoding the important image details (low frequency), and the other for encoding minor image details (high frequency).

4.3 Wavelet Transform (Haar Wavelet)

In my research, the wavelet transform is the first step in my algorithm. In order to implement the algorithm more easily, also the Haar wavelet transform was chosen. Nevertheless, other wavelet transforms can also be used instead.

The Haar Wavelet is the earliest known example of wavelet basis, and perhaps one of the simplest orthogonal bases. Hierarchical decomposition via filter banks transforms a signal in terms of the new basis. Thereby, the dilation equation [Willam, Rhee, & Beylkin].

$$\phi(t) = \sqrt{2}\sum_{k} h_0(k)\phi(2t-k) \qquad [16]$$

and the wavelet equation

$$\psi(t) = \sqrt{2}\sum_{k} h_1(k)\phi(2t-k) \qquad [18]$$

are the main repositories where $\phi(t)$ is called the scaling function, and $\psi(t)$ is called the wavelet function. The dilation and wavelet equation must hold for all t [Willam, Rhee, & Beylkin]. Illustrate the wavelet basis construction from the dilation equations using Haar filter bank. The low pass Haar filter is defined by $h_0(0) = h_0(1) = 1/\sqrt{2}$ while all other coefficients are zero [Willam, Rhee, & Beylkin]. Substituting the Haar low pass filter into equation [16], we get

$$\phi(t) = \phi(2t) + \phi(2t-1) \qquad [19]$$

The solution to this recurrence is the Haar scaling function [Willam, Rhee, & Beylkin].

$$\phi(t) = \begin{cases} 1 & \text{if} \quad t \in [0,1); \\ 0 & \text{otherwise.} \end{cases} \qquad [20]$$

The high pass Haar filter is defined by $h_1(0) = 1/\sqrt{2}$ and $h_1(1) = -1/\sqrt{2}$. Substituting into equation [18] yields [Willam, Rhee, & Beylkin].

$$\psi(t) = \phi(2t) - \phi(2t-1) \qquad [21]$$

So the Haar wavelet function is [Willam, Rhee, & Beylkin]

$$\psi(t) = \begin{cases} 1 & \text{if} \quad t \in [0, \frac{1}{2}); \\ -1 & \text{if} \quad t \in [\frac{1}{2}, 1); \\ 0 & \text{otherwise.} \end{cases} \qquad [22]$$

4.4 Fuzzification Function.

In this research, in order to deal with the wavelet coefficients, the range of the brightness level [0....255] is normalised into a closed unit interval [0, 1]. This section describes the fuzzification formulas from different Fuzzy image enhancement algorithms. They are scrutinized to find the most suitable fuzzification step-formula for the IWF algorithm.

The algorithms described in this section are applied by scanning the whole image pixel by pixel, calculating the fuzzified value, and recording the results into a 2-D array.

The initial step is described as follows:

For an image of size M x N, inspect each pixel (i, j), where i runs from [0, M-1], and j runs from [0, N-1], extract its value (or R,G,B values for coloured images), then perform the fuzzification operation.

4.4.1 Possibility Distribution algorithm [Hassanien & Badr, 2003]:

This routine depicts an algorithm extracted from the Possibility Distribution algorithm that shows how pixels could be fuzzified to reflect a value between [0,1], instead of [0, 255], and is therefore scrutinized.

Set $\beta_1 = \frac{\min + mid}{2}$; $\beta_2 = \frac{\max + mid}{2}$

Check all the pixels (i,j)

If min<=data[i][j] and data[i][j] < β_1

Compute FuzzyData[i][j] = pow((2*(pow((data[i][j] – min)/(mid - min),2))),2)

If β_1 <=data[i][j] and data[i][j] < mid

Compute FuzzyData[i][j] = pow((1-(2*(pow((data[i][j] – mid)/(mid - min),2)))),2)

If mid<=data[i][j] and data[i][j] < β_2

Compute FuzzyData[i][j] = pow((1-(2*(pow((data[i][j] – mid)/(mid - min),2)))),2)

If β_2 <=data[i][j] and data[i][j] < max

Compute FuzzyData[i][j] = pow((2*(pow((data[i][j] – mid)/(max - mid),2))),2)

4.4.2 Contrast Improvement with Intensification Operator [Hassanien & Badr, 2003]:

Similarly, this routine, extracted from the Intensification Operator is a good candidate for fuzzifying the pixel value to fall within the range [0,1], and is likewise scrutinized.

Firstly set $F_e = 2$ and $F_d = \frac{g_{\max} - g_{mid}}{0.5^{\frac{-1}{F_e - 1}}}$

Then compute the FuzzyData[i][j]:

$$g'_{i,j} = \left[1 + \frac{g_{\max} - g_{i,j}}{F_d} \right]^{-F_e}$$

Where

$g'_{i,j}$ is the new fuzzified value for (i,j)

$g_{\max}$ is the maximum pixel value

$g_{i,j}$ is the pixel value for (i,j)

4.4.3 Contrast Improvement with Fuzzy Histogram Hyperbolization [Hassanien & Badr, 2003]:

In the same way, this routine, also fuzzifies the pixel value to fall within the range [0, 1], and is likewise scrutinized.

Fuzzification technique:

If data[i][j] < 100

FuzzyData[i][j] = 0

Else if 100<=data[i][j]<=200

FuzzyData[i][j] = (0.01 * data[i][j]) -1

Else if 200<data[i][j]<=255

FuzzyData[i][j] = 1

4.4.4 Contrast Improvement based on Fuzzy If-Then Rules [Hassanien & Badr, 2003]:

This algorithm starts with the initialisation of the image parameters; minimum and maximum grey levels. It then sets grey levels by way of a fuzzification technique as described below [Hassanien & Badr, 2003]:

Find the minimum and maximum gray level

Calculating the mid gray level = (max + min)/2

If 0<=data[i][j]<min then

    Fuzzydata[i][j] = 0;

Else if min<=data[i][j]<mid

    Fuzzydata[i][j]= (data[i][j]/(max-mid)) -1;

Else if max<=data[i][j]<=255 then

    Fuzzydata[i][j]=1;

4.4.5 Experiment Results

Fig.32 shows the experimental results of the reconstruct the image after the fuzzification function. Fig.32-1 shows the original image. Fig.32-2 shows the result of applying the fuzzification function from the Possibility Distribution algorithm. Likewise, Fig.32-3 shows the result of the fuzzification function from Contrast Improvement with Intensification Operator. Also, Fig.32-4 shows the result of the fuzzification function from Contrast Improvement with Fuzzy Histogram Hyperbolization. Lastly, Fig.32-5 depicts the result of the fuzzification function from Contrast Improvement based on Fuzzy If-Then Rules.

(1) original image (2) Result#1 (3)Result#2

(4) Result#3 (5) Result#4

Fig.32 Results using different fuzzification functions

Comparing the results, the fuzzification function from the Contrast Improvement with Intensification Operator algorithm was chosen (because this kind of fuzzification function was found to be the best function for my image compression algorithm, as the quality of the reconstructed image is the best one).

4.5 IWF Algorithm

The strength of the IWF algorithm is in its simplicity of computation and therefore, offers a less expensive way of implementing. The IWF algorithm is as follows.

Step1. A wavelet transform decomposition.(for easy to implement, in this thesis we using the Haar wavelet transform.)

Step2. Use the following formula for mapping the wavelet coefficients.

$$W'(i,j) = \left[\frac{W(i,j) - Min}{(Max - Min)} * 255 \right] \qquad [23]$$

Where

w'(i,j) is the new coefficient' s value of (i,j), the value range is [0..255].

w(i,j) is a coefficient's value of (i,j).

min is a minimum of coefficients.

max is a maximum of coefficients.

Step3. Use the Huffman or LZW algorithm to compress the low frequency subband's coefficients.(in this thesis, using Huffman to implement it).

Step4. Use the following formula to blur the coefficients.

$F_e = 2$

and

$$F_d = \frac{g_{\max} - g_{Mid}}{0.5^{-1/F_e} - 1} \qquad [24]$$

The fuzzification function was derived from [Hassanien & Badr, 2003]. However, the parameters now depict new meanings.

$$g'_{i,j} = \left[1 + \frac{g_{\max} - g_{i,j}}{F_d} \right]^{-F_e} \qquad [25]$$

Where

$g'_{i,j}$ is the new coefficient (i,j)

$g_{\max}$ is the maximum of wavelet coefficients

$g_{i,j}$ is the wavelet coefficient (i,j)

Fig.32 shows the fuzzification function:

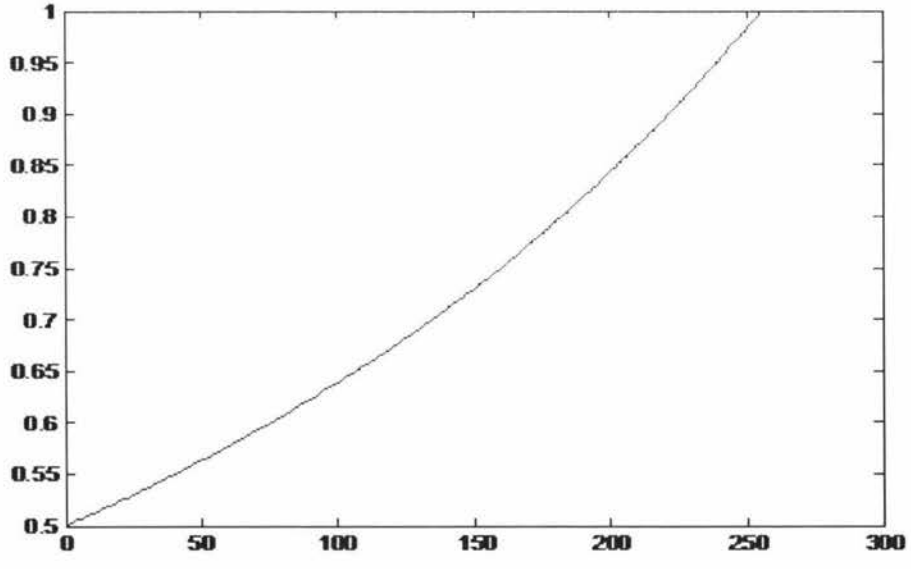


Fig.33 The Fuzzification Function

Step5. Using RLE coding the new coefficients and using Huffman or LZW to compress it (in this thesis, using Huffman algorithm)

4.6 Sample Application of the IWF Algorithm

Let us take a small image to explain the main idea of IWF algorithm. Consider the following 8 by 8, 8-bit image:

181	171	175	184	197	200	192	196
201	198	195	201	206	202	196	189
202	201	193	192	194	190	182	181
195	192	183	180	185	188	187	189
189	190	187	188	188	194	196	199
194	193	192	195	194	197	197	203
197	197	197	200	201	204	208	210
206	207	210	213	211	212	212	211

Fig.34 8x8 Image Data

After the wavelet decomposition:

760	767	8	-20	23	18	5	-1
788	808	24	25	-8	-11	-5	6
-9	-12	-11	3	4	6	4	2
2	8	4	-2	9	13	9	2
-6	7	0	-1	3	-1	-3	-5
-2	-2	0	0	-1	-1	3	1
0	2	4	4	-1	1	-1	1
0	3	2	0	0	0	-1	-1

Fig.35 2-level Wavelet Decomposition

After mapping the coefficients changes to:

240	242	9	0	13	12	8	6
249	255	14	14	4	3	5	8
3	2	3	7	7	8	7	7
7	9	7	6	9	10	9	7
4	8	6	6	7	6	5	5
6	6	6	6	6	6	7	6
6	7	7	7	6	6	6	6
6	7	7	6	6	6	6	6

Fig.36 Coefficients after Mapping

After the fuzzification, the coefficients change to:

0.91	0.92	0.26	0.25	0.27	0.26	0.26	0.26
0.96	1.00	0.27	0.27	0.25	0.25	0.26	0.26
0.25	0.25	0.25	0.26	0.26	0.26	0.26	0.26
0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26
0.25	0.26	0.26	0.26	0.26	0.26	0.26	0.26
0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26
0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26
0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26

Fig.37 Coefficients after Fuzzification

After the fuzzification process, the coefficients in high frequency subbands are basically same, so according to the property we can easily compress the high subband's coefficients.

4.7 Grayscale Image Compression

4.7.1 Introduction

Grayscale images are images without colour, or achromatic images. The levels of a grayscale range from 0 (black) to 1 (white). Each pixel value in a grayscale image corresponds to an amount or quantity of light. In the realm of physics, the terms intensity or luminance are used to describe the energy density of light. The term brightness is often used to describe perceived intensity in the psychological sense of visual perception [Gray Scales].

The relationship between pixel value and intensity can be nonlinear. For example, there is a non-linear dependence of the luminance produced by a CRT on the voltage of the input signal to the CRT controller since energy is proportional to the square of voltage [Gray Scales].

The human eye is sensitive to the ratios of intensity levels rather than to absolute values in intensity [Gonzalez, R. C., & Woods, R. E. (2002)]. Brightness (perceived by the eye) and loudness (perceived by the ear) are both logarithmic functions of the intensity of the corresponding wave (light or sound) [Gray Scales].

Gray-scale images are very common, in part because much of today's display and image capture hardware can only support 8-bit images. additionally, gray-scale images are entirely sufficient for many tasks and so there is no need to use more complicated and harder-to-process colour images.

4.7.2 IWF Algorithm (grayscale image)

The IWF gray-scale image compression algorithm follows the system shown in Fig. 31.

4.7.3 Experimental Results

4.7.3.1 Time and Compression Ratio Analysis

In this section, the standard image (lena.bmp 512x512) was used in IWF algorithm to analyze the total time expenditure and compression ration.

Fig.38 Original Image (lena.bmp 512 × 512)

The total time includes the decomposition time, the compression time, the decompression time and the reconstruction time. The following graphs will demonstrate the relationship between the each kind of time and the level of wavelet decomposition.

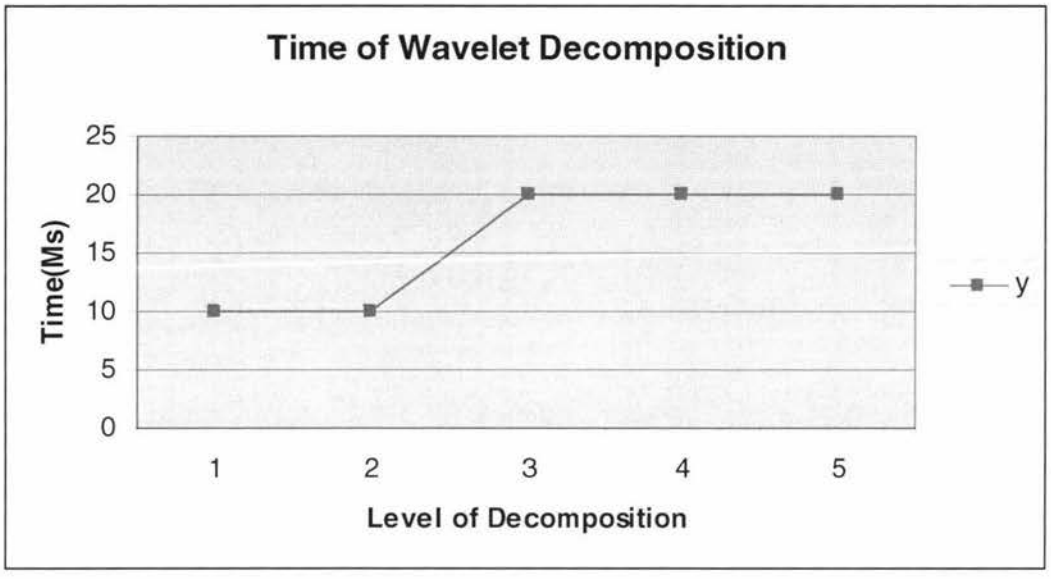


Fig.39 Relationship between the Time and the Level of Wavelet Decomposition

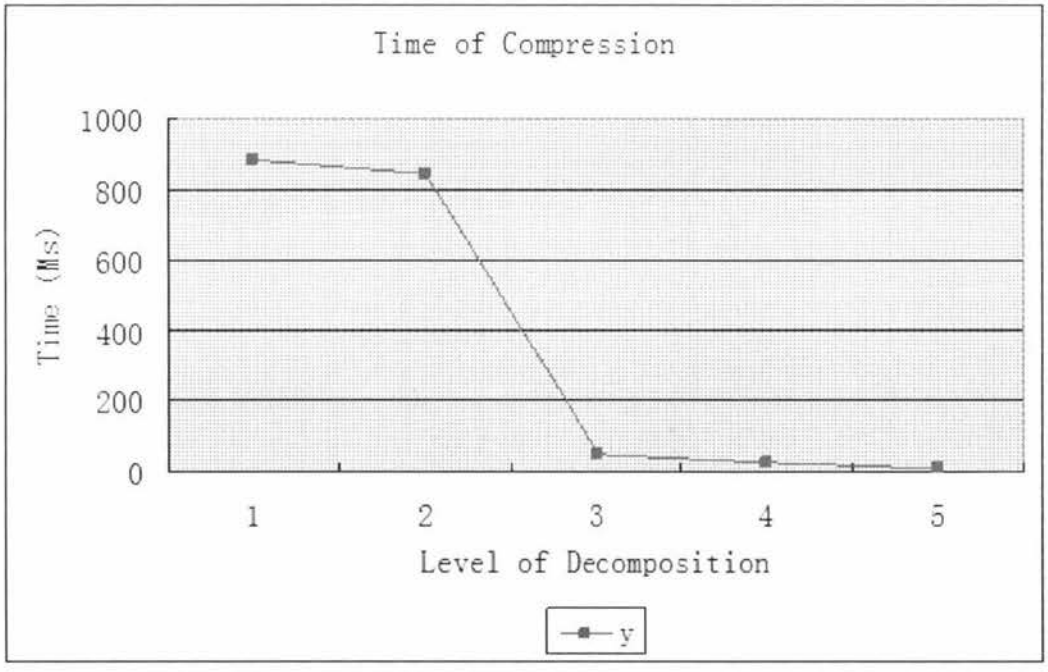


Fig.40 Relationship between the Time and the Compression

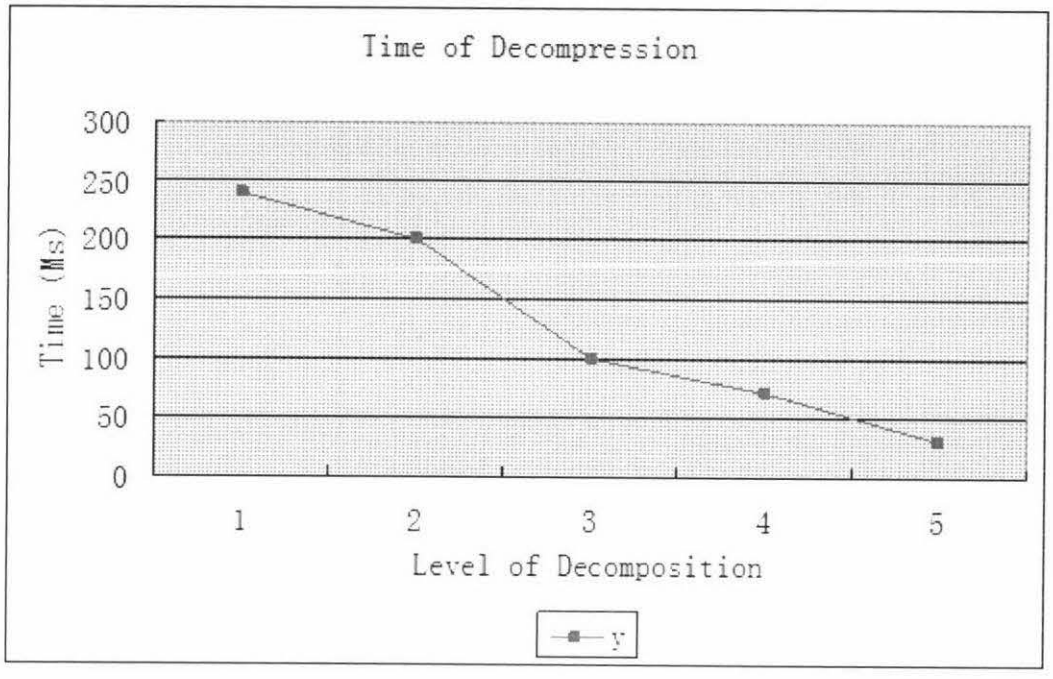


Fig.41 Relationship between the Time and the Decompression

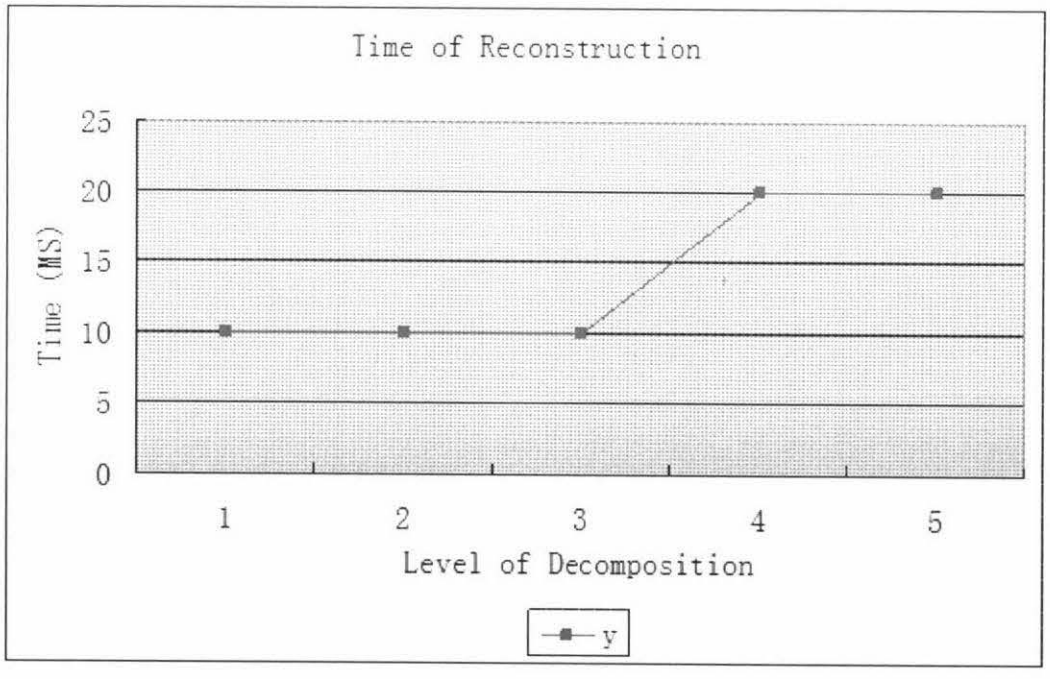


Fig.42 Relationship between the Time and the Reconstruction

Based on the above graph, the decomposition and reconstruction time are smaller than the compression and decompression time. Therefore we can ignore the decomposition and reconstruction time, and just care about the compression and

decomposition time. The below graph demonstrates the relationship between the total time (compression and decompression time) and level of decomposition.

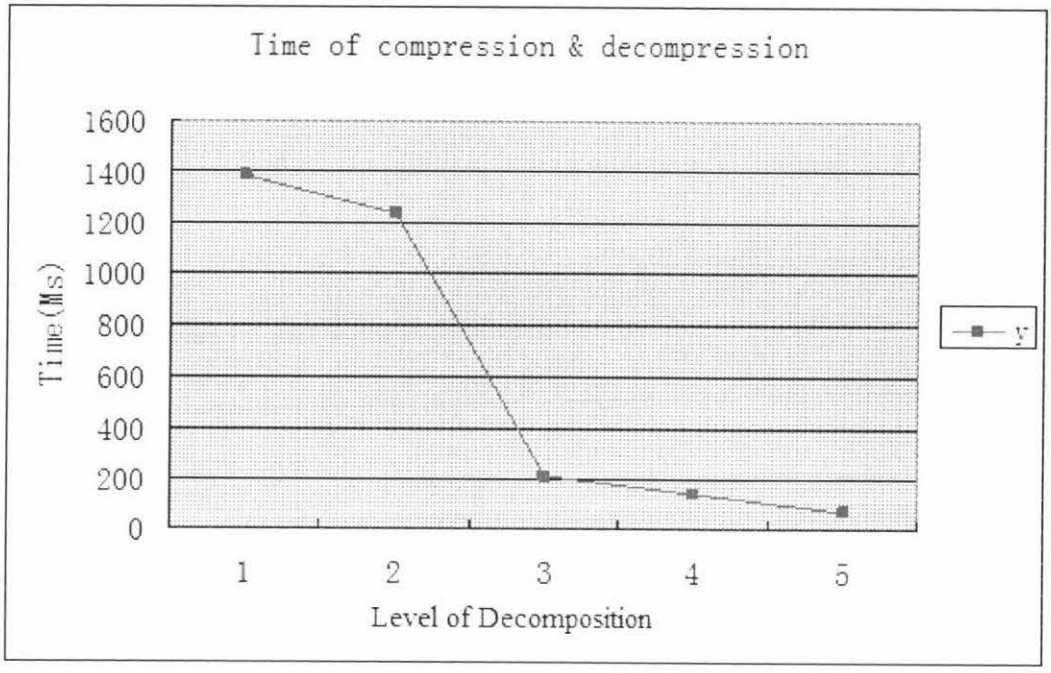


Fig.43 Relationship between the Total Time and the Level of Decomposition

The graph below demonstrates the relationship between the compression ratio and level of decomposition.

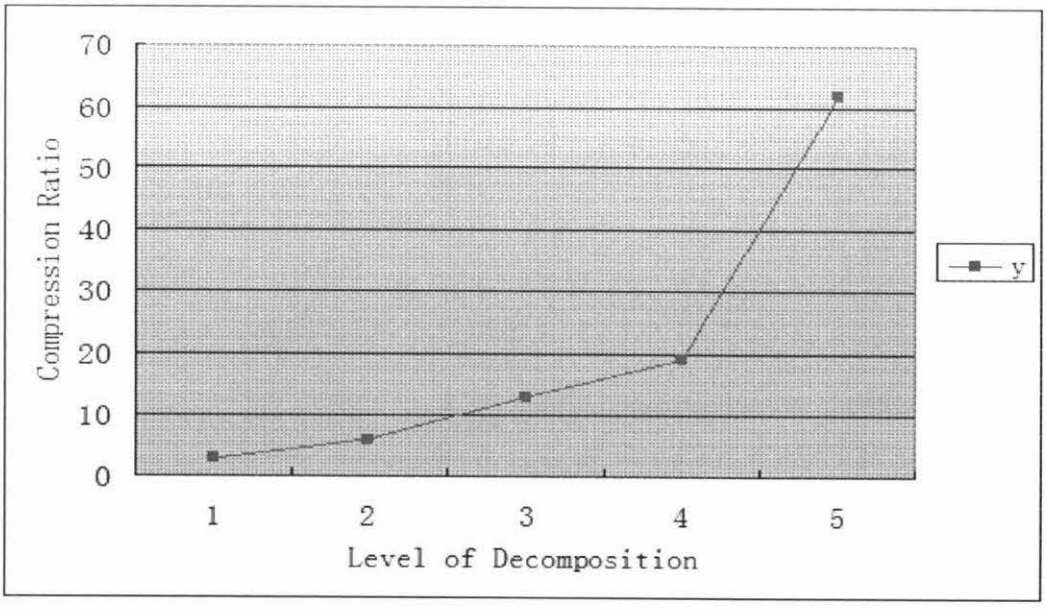


Fig.44 Relationship between the Compression Ratio and the Level of Decomposition

4.7.3.2 Image Compression Results

In the experiments performed, eight standard images were used. They are all kinds of different gray-level images including natural scenes, geometric image, portraits, and are shown in Fig.45. The decompression images (3-level wavelet decomposition) are shown in Fig.46.

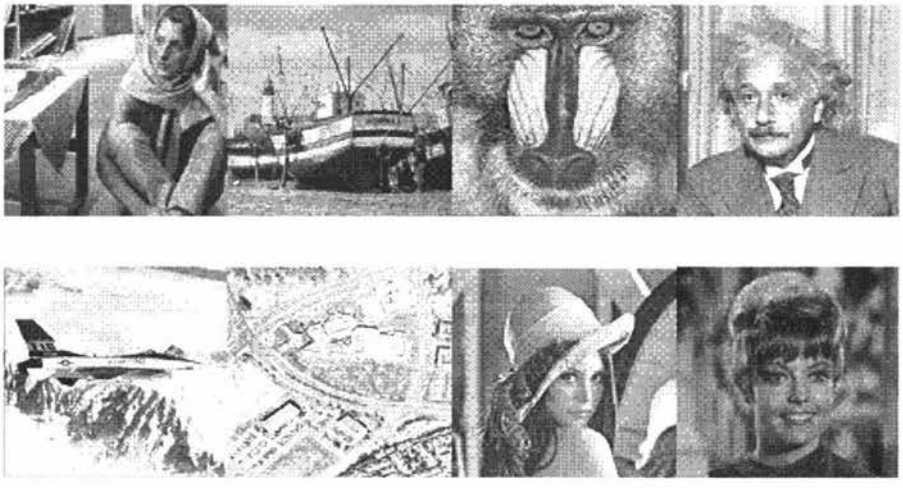


Fig.45 Test Images (Standard test images for image processing)

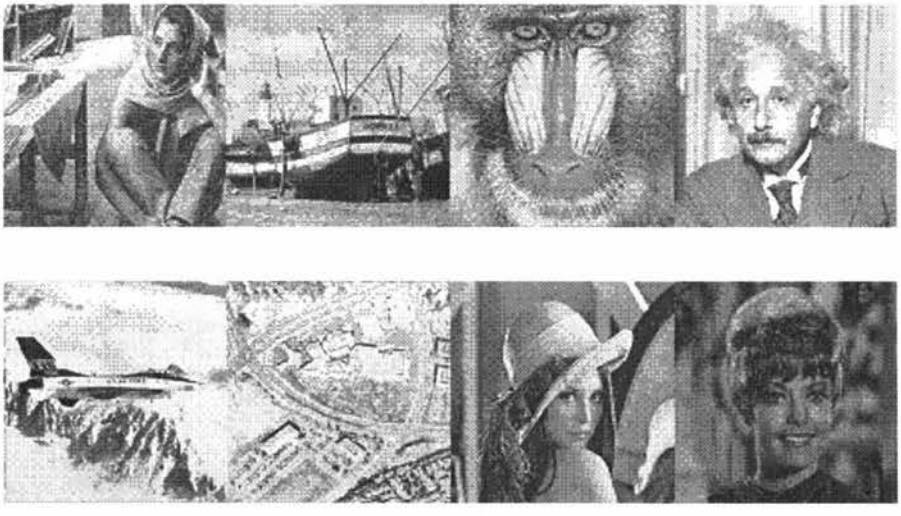


Fig.46 Compressed Images (3-Level of Wavelet Decomposition)

The graph below demonstrates the relationship between the compression ratio and different levels of decomposition.(the test image shown in Fig.45)

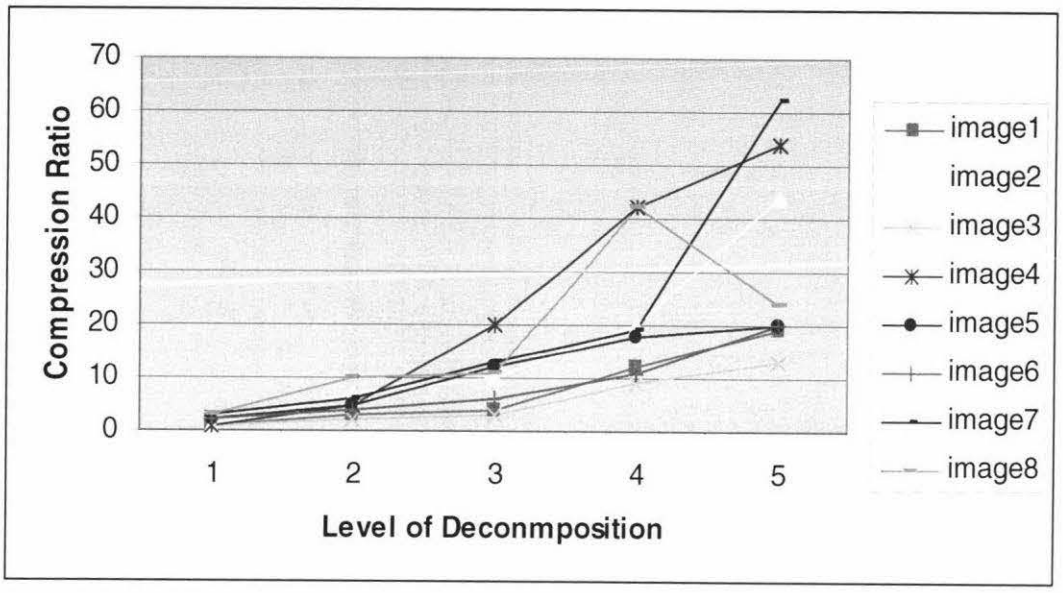


Fig.47 Relationship between the Compression Ratio and the Level of Decomposition

The following images demonstrated the quality of decompression image and compression ration on different level wavelet decomposition.

Fig.48 Original Test Image (lena.bmp 512 x 512)

Fig.49 Compressed Image

(1-Level Wavelet Decomposition, Compression Ration 3:1)

Fig.50 Compressed Image

(2-Level Wavelet Decomposition, Compression Ration 6:1)

Fig.51 Compressed Image

(3-Level Wavelet Decomposition, Compression Ration 13:1)

Fig.52 Compressed Image

(5-Level Wavelet Decomposition, Compression Ration 62:1)

4.8 Colour Image Compression

4.8.1 Introduction

The human eye has only three different colour (red, green and blue) receptors, each of them sensible to one of the three colours, so it is possible to construct all visible colours by combining the three colours. Different combinations in the stimulation of the receptors enable the human eye to distinguish approximately *350000* colours. A RGB colour image is a multi-spectral image with one band for each colour red, green and blue, thus producing a weighted combination of the three primary colours for each pixel [Conversion of a Color Image to a Grayscale Image].

A full 24-bit colour image contains one 8-bit value for each colour, thus being able to display $2^{24} = 16777216$ different colours [Gonzalez, R. C., & Woods, R. E. (2002)]. There are an infinite number of possible colour spaces instead of the common RGB (Red, Green, and Blue) system most commonly used in computer graphics [Bourke, 2000]. Many of these other colour spaces are derived by applying linear functions of r, g, and b [Bourke, 2000]:

$$v1 = Ar_1 R + Ag_1 G + Ab_1 B$$

$$v2 = Ar_2 R + Ag_2 G + Ab_2 B$$

$$v3 = Ar_3 R + Ag_3 G + Ab_3 B$$

Where the Ar, Ag, Ab are constants for a particular colour space. Similarly for any such system there are linear functions to go back to RGB space (Dr, Dg and Db are constants for a special colour space) [Bourke, 2000]:

$$R = Dr_1 v1 + Dg_1 v2 + Db_1 v3$$

$$G = Dr_2 v1 + Dg_2 v2 + Db_2 v3$$

$$B = Dr_3 v1 + Dg_3 v2 + Db_3 v3$$

4.8.2 IWF Algorithm (for Colour Image)

The IWF algorithm can also be applied to colour image compression. The algorithm is as follows.

Step1. Transform the RGB colour space to the YUV colour space (See Sec.2.1.2 for more details).

Step2. A wavelet transforms decomposition the Y, U and V.

Step3. Using mapping function to deal with the wavelet coefficients

Step4. Using the Huffman or LZW algorithm to compress the low frequency subband's coefficients.(in this thesis, using Huffman to implement it).

Step5. Using the fuzzification function handle the new coefficients to get the fuzzy coefficients.

Step6. Using RLE coding the new coefficients and using Huffman or LZW to compress the fuzzy coefficients

The Fig.52 shows the IWF colour image compression algorithm. As compared to the IWF algorithm for gray-scale images, the general system architecture now includes a component for transforming the R,G,B values of a pixel into its YUV equivalent. As a consequence, this system now employs the Fuzzification function independently for each colour component.

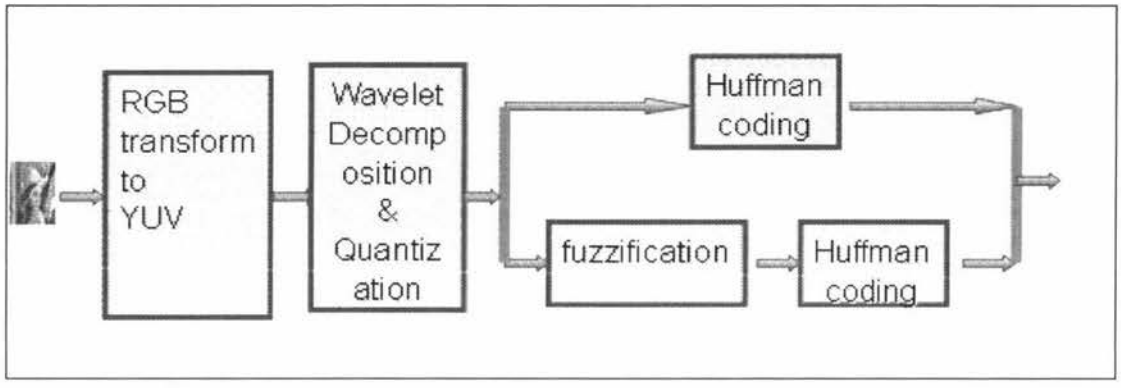


Fig.53 The General Architecture of the IWF Colour Image Compression

Let us take a small image to explain the main idea of IWF algorithm how to compress the colour image. Consider the following 8 by 8, 24-bit image:

For the first step, transform the RGB values depicting the image into YUV colour components.

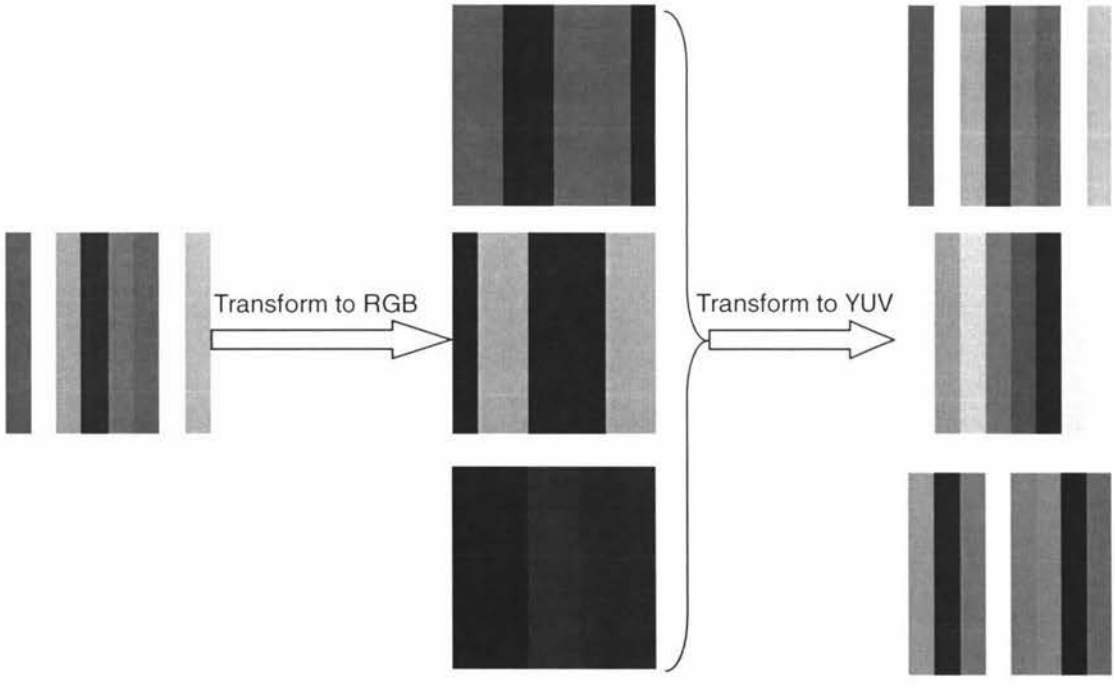


Fig.54 The Transform the Colour Space

After the colour space transform, the colour image compression steps are the same as the gray-scale image compression. However, the mapping and fuzzification steps require processing separately the Y, U and V colour components.

4.8.3 Experimental Results

In the colour image experiments, using the different colour images shown in (Fig.54) to test the IWF algorithm to compress the colour image in different levels of wavelet decomposition (Fig.55-Fig.57) [USC-SIPI Image Database - Miscellaneous], it was observed that the level of image compression depends on the image content. In particular, if there is a distinguishable colour difference (by way of human vision judgment) between the background and objects (e.g. brightly-lit background with objects that are easily identifiable), then the resulting level of image compression is high.

Fig.55 original colour images

Fig.56 Compressed colour images (1-level wavelet decomposition)

Fig.57 Compressed colour images (2-level wavelet decomposition)

The following images demonstrated the quality of decompression image and compression ration on different level wavelet decomposition

Fig.58 Original Test Image (lena.bmp 256x256)

The process of image compression is demonstrated in the following image (Fig.54 –Fig.57)

Fig.59 Compressed Image

(1-Level Wavelet Decomposition, Compression Ratio 2:1)

Fig.60 Compressed Image

(2-Level wavelet Decomposition, Compression Ratio 3.2:1)

Fig.61 Compressed Image

(3-Level Wavelet Decomposition, Compression Ratio 7.2:1)

4.8.4 Comparison between IWF Algorithm and Other Algorithms (based on Colour Images)

By using the original colour image 'Lenna' (256 x 256 pixels) shown in Fig 7.14 extracted from standard image database (SIDBA), an experimental comparison of the new image compression based on Wavelet & Fuzzy Logic and colour image compression-reconstruction by Fuzzy Wavelet. In the experiment, the main factor considered in the analysis is the root mean square error (RMSE). The RMSE defined as

$$RMSE=\sqrt{\frac{1}{3*M*N}\sum\left(\left(R^{'}-R\right)^{2}+\left(G^{'}-G\right)^{2}+\left(B^{'}-B\right)^{2}\right)} \quad [26]$$

Where R, G, B and $R^{'}$, $G^{'}$, $B^{'}$ stands for the original image and reconstructed one. Using the same image, the RMSE of the new image compression based on Wavelet & Fuzzy Logic is lower than that of colour image compression-reconstruction by Fuzzy Wavelet. The table below shows the compression ratio and the RMSE of two algorithms.

For the IWF algorithm, it is worth-noting that based on the different levels of decomposition used, as shown in Figures 59-61, the best level of decomposition was found to be the 3-level, and so it is used for comparison in the following table. It is evident in Table 9 that IWF performs better both in terms of compression ratio, and RMSE. This means that IWF returns a more compact form of the original image, consuming lesser space, as well as a more accurate representation of the original image; that is, lower RMSE value.

Algorithm	Compression ratio	RMSE
The new image compression based on Wavelet & Fuzzy Logic	16	20.32
colour image compression-reconstruction by Fuzzy Wavelet [Nobuhara, 2004]	10	25.96

Table.9 Comparing the RMSE of two Algorithms

Chapter 5

Conclusions and Future Works

5.1 Wavelet Image Compression

Wavelet compression is one of the transform coding methods. Wavelet compression algorithm presents an image as set of real coefficients, after the wavelet decomposition, the great value coefficients are located at the low subband, the other low coefficients are located at the high subbands. The coefficients in the low subband represent the most important image information; the other coefficients in high subbands are nearly zero. In contrast to the traditional JPEG, wavelet algorithm uses the wavelet transform to replace the DCT transform in the JPEG, and wavelet algorithm analyze the whole image, this feature of wavelet compression allows getting the high compression ratio and better image quality. On the other hand, the JPEG must divide the image into 8x8 small blocks; this will cause the block effect. The main different between the wavelet and JPEG resulted in wavelet-based image compression fast development.

All common wavelet-based image compression algorithms include five basic steps [Betz, Bhagat, Murphy, & Stengler]:

1. Digitize the source image into signals, which is a string of numbers.

2. Decompose the signal into a sequence of wavelet coefficients W.

3. Use thresholding to modify the wavelet coefficients form W to another sequence W'.

4. Use quantization to covert the wavelet coefficients W' to a sequence Q.

5. Apply entropy coding to compress Q into a sequence E.

In the thesis, it also uses the basic technique of Wavelet decomposition to deal with the digitized image. After decomposition, it uses different formula (equation [24]) and the Fuzzy Logic techniques to treat the Wavelet coefficients.

The stages of compression and decompression are shown in Fig.58 and Fig.59. All of the steps shown in the compression diagram are invertable, except for the quantization step [James S, W].

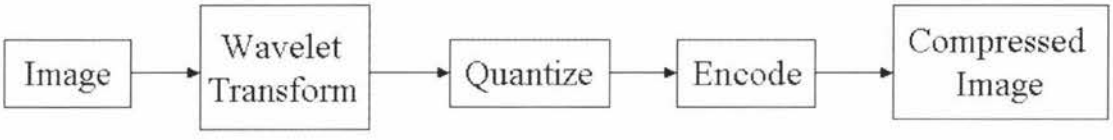


Fig.62 Compression of an Image [James S, W]

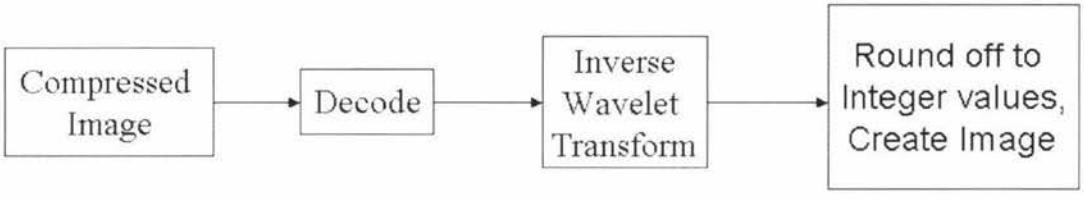


Fig.63 Decompression of an Image [James S, W]

5.2 Fuzzy Image Compression

In recent years subband image coding, especially wavelet-subband image coding, emerged as an efficient method for image compression. There is also an increasing interest in image compression which provides desired image quality or compression rate for the particular application [Planinsie, 2000].

To avoid the indicated constraints and the repeat scanning of the Wavelet coefficients, a more computationally efficient method which involves the blurring of the Wavelet coefficients by fuzzy techniques are developed [Planinsie, 2000]. The fuzzy techniques are used for adjusting the quantization steps. Uses of fuzzy techniques allow simple solving of the computationally expensive and the scanning problem.

This can be determined computationally efficient analytically using the so called low distortion models, although at higher compression rate the better result is achieved by fuzzy logic [Planinsie, 2000].

5.3 Synopsis

In this thesis, a new image compression algorithm called IWF based on wavelet and Fuzzy logic techniques is presented. The key idea is to use wavelet transform to decompose the image, and apply Fuzzy Logic to optimize the calculation of wavelet coefficients. The main advantage of the IWF algorithm is that it is computationally inexpensive as compared to other algorithms when dealing with coefficients that do not reflect much image information.

This research contributes in the field of image compression in that it provides a new image compression algorithm called IWF. Furthermore, this research extends the repertoire of fuzzy logic techniques in image processing. It simplifies the scanning of the wavelet coefficients by reducing it to no more than two times. Therefore, it provides a fast and simpler method for the image compression. In addition, as compared to other techniques, nearly almost other image compression algorithms scan the coefficients more than three times. Lastly, another advantage is that the IWF algorithm is much easier to implement as it does not rely on the associations between the wavelet coefficients.

5.4 Research Papers Published

The following is a list of research papers published as conference proceeding, as a result of this research untaking.

1. Li Huang, N.H.Reyes, (2006). A Fuzzy and Wavelet-based Image Compression. Proceedings. The 1st Korean-NewZealand joint Workshop on Advance of Computational Intelligence Methods and Applications. AUT Technology Park, Auckland,New Zealand, 17, Feb. 2006.

2. Li Huang. (2005). Image Compression based on Fuzzy Technique and Wavelet Transform. Proceedings of the second Annual Post graduate Conference of the Institute of information and mathematical Science, Massey University, Auckland, New Zealand, 27, Oct. 2005.

5.5 Suggestions for Future Work

There are many areas touched on this thesis that could be expanded for future research works. For one, the wavelet transform used in this research employed the Haar wavelet, but other wavelets could be used as well. Also the fuzzy logic technique used for the calculation of the coefficients could be modified to improve its accuracy. This algorithm also loses some information in the quantization and fuzzification step, so in future work, determining a better quantization and fuzzification scheme are most important.

Appendix (the main code for thesis):

Wavelet transform function

```
procedure TForm1.HaarWaveletTransform(pic: TCoefficient; wl: TCoefficient;
dx,dy,depth: Integer);
var
  Row,Col,i,j : Integer;
  A,B,C,D,Ho,Hx,Hy,Hc : Integer;
  Temp: TCoefficient;
begin
  SetLength(Temp,Bmp_Width,Bmp_Height);
  Row:=0;
  Col:=0;
  while Row <= dy - 1 do
  begin
    while Col <= dx - 1 do
    begin
      A:=pic[Row][Col];
      B:=pic[Row][Col+1];
      C:=pic[Row+1][Col];
      D:=pic[Row+1][Col+1];

      Ho:=(A+B+C+D) div 2;
      Hx:=(D+C-B-A) div 2;
      Hy:=(D-C+B-A) div 2;
      Hc:=(D-C-B+A) div 2;

      temp[Row div 2][Col div 2]:=Ho;
      temp[Row div 2][Col div 2 + dx div 2]:=Hx;
      temp[Row div 2 + dy div 2][Col div 2]:=Hy;
      temp[Row div 2 + dy div 2][Col div 2 + dx div 2]:=Hc;

      Col:=Col+2;
    end;
    Col:=0;
    Row:=Row+2;
  end;

  for i:=0 to dy - 1 do
  begin
    for j:=0 to dx - 1 do
    begin
      wl[i][j]:=temp[i][j];
```

```
        temp[i][j]:=0;
      end;
    end;
    if depth > 0 then
    begin
      HaarWaveletTransform(wl,wl,dx div 2,dy div 2,depth - 1);
    end;
end;

procedure TForm1.InHaarWaveletTransform(wl: TCoefficient; pic: TCoefficient;
dx,dy,depth: Integer);
var
  Row,Col,i,j : Integer;
  A,B,C,D,Ho,Hx,Hy,Hc : Integer;
  Temp: TCoefficient;
begin
  if depth>0 then
    InHaarWaveletTransform(wl,wl,dx div 2,dy div 2,depth-1);

  SetLength(Temp,Bmp_Width,Bmp_Height);
  Row:=0;
  Col:=0;
  i:=0;
  j:=0;
  while Row <= dy div 2 - 1 do
  begin
    while Col <= dx div 2 - 1 do
    begin
      A:=wl[Row][Col];
      B:=wl[Row][Col+dx div 2];
      C:=wl[Row+dy div 2][Col];
      D:=wl[Row+dy div 2][Col+ dx div 2];

      Ho:=(A+D-B-C) div 2;
      Hx:=(A+C-B-D) div 2;
      Hy:=(A+B-C-D) div 2;
      Hc:=(A+B+C+D) div 2;

      temp[i][j]:=Ho;
      temp[i][j+1]:=Hx;
      temp[i + 1][j]:=Hy;
      temp[i + 1][j + 1]:=Hc;

      j:=j+2;
```

```
            Col:=Col+1;
        end;
        Col:=0;
        i:=i+2;
        j:=0;
        Row:=Row+1;
    end;

    for i:=0 to dy - 1 do
    begin
        for j:=0 to dx - 1 do
        begin
            pic[i][j]:=temp[i][j];
            temp[i][j]:=0;
        end;
    end;
end;
```

Fuzzification, compression and decompression function:

```
function TForm1.set_fuzzydata(scale,count: Integer):integer;
var
    x,y,width,height: Integer;
begin
    width:=Bmp_Width shr scale;
    Height:=Bmp_Height shr scale;

    for y:=0 to Height - 1 do
    begin
        for x:=width to 2*width - 1 do
        begin
            de_fuzzy[x,y]:=de_data[count];
            count:=count+1;
        end;
    end;

    for y:=height to 2*Height - 1 do
    begin
        for x:=0 to width - 1 do
        begin
            de_fuzzy[x,y]:=de_data[count];
            count:=count+1;
        end;
    end;
```

```
    for y:=height to 2*Height - 1    do
    begin
      for x:=width to 2*width - 1 do
      begin
        de_fuzzy[x,y]:=de_data[count];
        count:=count+1;
      end;
    end;
    set_fuzzydata:=count;
end;

procedure TForm1.Compress_BtnClick(Sender: TObject);
var
    x,y,initial_value,i,num: Integer;
    Temp: Double;
    T,result_str: String;
    temp1: TCoefficient;
    StartTime, StopTime : Tsystemtime;
    difference: String;
    f: file of Byte;
begin
    num:=0;
    result_str:='';
    memo1.Lines.Clear;
    for y:=0 to Bmp_Height - 1 do
    begin
      for x:=0 to Bmp_Width - 1 do
      begin
        Temp:=Power((1 + (255 - New_Coefficient[x,y]) / F_d),-F_e);
        Temp:=Power(Temp,2);
        Str(Temp:2:2,T);
        Temp:=StrToFloat(T);
        Fuzzy_Value[x,y]:=Temp;
      end;
    end;
    ....
    for i:=level+1 downto 1 do
    begin
      result_str:=result_str + form1.Compression(i);
    end;
    ...
end;
```

```
function TForm1.Compression(scale: Integer):string;
var
  x,y,count,width,height: Integer;
  new_str: String;
  temp: double;
  stream: TfileStream;
begin
  count:=0;
  new_str:='';
  width:=Bmp_Width shr scale;
  Height:=Bmp_Height shr scale;

  temp:=Fuzzy_Value[width,0];
  for y:=0 to Height - 1 do
  begin
    for x:=width to 2*width - 1 do
    begin
      if temp = Fuzzy_value[x,y] then
      begin
        count:=count + 1;
      end
      else
      begin
        new_str:=new_str + IntToStr(count) + fuzzydata2letter(temp);
        temp:=Fuzzy_Value[x,y];
        count:=1;
      end;
    end;
  end;
  if count >= 1 then
  begin
    new_str:=new_str + IntToStr(count) + fuzzydata2letter(temp);
    count:=0;
  end;

  temp:=Fuzzy_Value[0,height];
  for y:=height to 2*Height - 1 do
  begin
    for x:=0 to width - 1 do
    begin
      if temp = Fuzzy_value[x,y] then
      begin
        count:=count + 1;
      end
```

```
        else
        begin
          new_str:=new_str + IntToStr(count) + fuzzydata2letter(temp);
          temp:=Fuzzy_Value[x,y];
          count:=1;
        end;
      end;
    end;
    if count >= 1 then
    begin
      new_str:=new_str + IntToStr(count) + fuzzydata2letter(temp);
      count:=0;
    end;

    temp:=Fuzzy_Value[width,height];
    for y:=height to 2*Height - 1 do
    begin
      for x:=width to 2*width - 1 do
      begin
        if temp = Fuzzy_value[x,y] then
        begin
          count:=count + 1;
        end
        else
        begin
          new_str:=new_str + IntToStr(count) + fuzzydata2letter(temp);
          temp:=Fuzzy_Value[x,y];
          count:=1;
        end;
      end;
    end;
    if count >= 1 then
    begin
      new_str:=new_str + IntToStr(count) + fuzzydata2letter(temp);
      count:=0;
    end;
    Compression:=new_str;
end;

procedure TForm1.DeCompression(File_name: String);
var
    len,i,count,start,finish,j: Integer;
    str,tempstr,tempstr1: String;
    value : Double;
```

```
begin
   start:=0;
   Count:=0;
   tempstr:='';
   tempstr1:='';
   memo2.Lines.LoadFromFile(File_name);
   Str:=Memo2.Lines.Text;
   len:=Length(str);
   for i:=1 to len do
   begin
      case str[i] of
         'A':
         begin
             tempstr1:=tempstr1 + 'A';
         end;
         'B':
            begin
             tempstr1:=Tempstr1 + 'B';
         end;
         'C':
            begin
             tempstr1:=Tempstr1 + 'C';
         end;
         'D':
            begin
             tempstr1:=Tempstr1 + 'D';
         end;
         'E':
            begin
             tempstr1:=Tempstr1 + 'E';
         end;
         'F':
            begin
             tempstr1:=Tempstr1 + 'F';
         end;
         'G':
            begin
             tempstr1:=Tempstr1 + 'G';
         end;
         'H':
            begin
             tempstr1:=Tempstr1 + 'H';
         end;
         'I':
```

```
            begin
              tempstr1:=Tempstr1 + 'I';
            end;
          'J':
            begin
              tempstr1:=Tempstr1 + 'J';
            end;
        else
          if tempstr1 <> '' then
          begin
            Value:=letter2fuzzydata(tempstr1);
            finish:=start + StrToInt(tempstr) - 1;
            for j:=start to finish do
            begin
              de_data[j]:=Value;
            end;
            start:=start + StrToInt(tempstr);
            tempstr:='';
            tempstr1:='';
            tempstr:=tempstr + str[i];
          end
          else
          begin
            tempstr:=tempstr + str[i];
          end;
        end
      end;
end;
```

Other functions:

```
function TForm1.GetMaxValue(m: TCoefficient;dx,dy: Integer): Integer;
var
  Max,i,j: Integer;
begin
  Max:=0;
  for i:=0 to dy - 1 do
  begin
    for j:=0 to dx - 1 do
    begin
      if m[i][j] > Max then
        Max:=m[i][j];
    end;
  end;
```

```
    GetMaxValue:=Max;
end;

function TForm1.GetMinValue(m: TCoefficient;dx,dy: Integer): Integer;
var
    Min,i,j : Integer;
begin
    Min:=0;
    for i:=0 to dy - 1 do
    begin
        for j:=0 to dx - 1 do
        begin
            if m[i][j] < Min then
                Min:=m[i][j];
        end;
    end;
    GetMinValue:=Min;
end;

procedure TForm1.set_fuzzygray(temp: TFuzzy_gray; ini_value: Integer);
var
    x,y,num : Integer;
    total,average : double;
begin
    total:=0.0;
    num:=0;
    for y:=0 to ini_value - 1 do
    begin
        for x:=ini_value to 2*ini_value - 1 do
        begin
            total:=total + temp[x,y];
            num:=num + 1;
        end;
    end;
    average:=total / num;
    for y:=0 to ini_value - 1 do
    begin
        for x:=ini_value - 1 to 2*ini_value - 1 do
        begin
            temp[x,y]:=average;
        end;
    end;
end;
```

```
function TForm1.fuzzydata2letter(value: Double):string;
var
   temp,temp1 : String;
   len,i : Integer;
begin
   temp1:='';
   temp:=FloatToStr(value);
   len:=length(temp);
   for i:=3 to len do
   begin
      case temp[i] of
         '0':
            temp1:=temp1 + 'A';
         '1':
            temp1:=temp1 + 'B';
         '2':
            temp1:=temp1 + 'C';
         '3':
            temp1:=temp1 + 'D';
         '4':
            temp1:=temp1 + 'E';
         '5':
            temp1:=temp1 + 'F';
         '6':
            temp1:=temp1 + 'G';
         '7':
            temp1:=temp1 + 'H';
         '8':
            temp1:=temp1 + 'I';
         '9':
            temp1:=temp1 + 'J';
      end;
   end;
   fuzzydata2letter:=temp1;
end;

function TForm1.letter2fuzzydata(value: string):double;
var
   temp: String;
   len,i : Integer;
begin
   temp:='';
   len:=length(value);
   for i:=1 to len do
```

```
  begin
    case value[i] of
      'A':
        temp:=temp + '0';
      'B':
        temp:=temp + '1';
      'C':
        temp:=temp + '2';
      'D':
        temp:=temp + '3';
      'E':
        temp:=temp + '4';
      'F':
        temp:=temp + '5';
      'G':
        temp:=temp + '6';
      'H':
        temp:=temp + '7';
      'I':
        temp:=temp + '8';
      'J':
        temp:=temp + '9';
    end;
  end;
  temp:='0.'+temp;
  letter2fuzzydata:=StrToFloat(temp);
end;
...
```

REFERENCES:

Antonini, M., Barlaud, M., Mathieu, P., & Daubechies, I. (1992). Image coding using wavelet transform. *IEEE Transactions on Image Processing, 1.*

Berger, T. (1971). *Rate-distortion theory: A mathematical basis for data compression*: Englewood Cliffs,NJ: prentice-Hall.

Bourke, P. (1995). RLE - Run length Encoding [Electronic Version].

Bourke, P. (2000). YCC Colour Space and Image Compression [Electronic Version].

Bultheel, A. (2003). *Wavelets with application in signal and image processing.*

Chrysafis, C. (2000). *Wavelet Image Compression Rate Distortion Optimizations and Complexity Reductions.* University of Southern California.

Gakhal, R. S. (2004). Wavelet Based Progressive Image Coding.

Gonzalez, R. C., & Woods, R. E. (2002). *Digital Image Processing (2nd Edition)*: Prentice Hall.

Han-sheng, L., & Xiao-ping, Z. (2002). Facial Image Compression Based on Wavelet Transform and Vector Quantization. *Journal of Image and graphics, 7(A).*

Hassanien, A. E., & Badr, A. (2003). A Comparative Study on Digital Mamography Enhancement Algorithm Based on Fuzzy Theory. *Studies in infromatics and control, 12.*

James S, W. *Wavelet-based Image Compression*

Sub-chapter of CRC Press Book: Transforms and Data Compression.

Kaya, M. A New Image Clustering Nad Compression Method Based on Fuzzy Logic and Discrete Cosine Transform [Electronic Version].

KAYA, M. (2005). An Algorithm for Image Clustering and Compression. *Turk J Elee Engin, 13.*

LI-Bao, Z., & Ke, W. (2003). An Image Compression Algorithm Based on Integer Wavelet Transform. *2003 Journal of Software, 14.*

Nobuhara, H., & Hirota, K. (2004). Color Image Comression/Reconstruction by YUV Fuzzy Wavelet. *IEEE Transactions on Image Processing.*

Polikar, R. (1995). The Wavelet Tutorial.

Qing, H., & Shu-ling, Z. (2004). Image compression based on wavelet transform and classified vector quantization.

Rao, K. R., & Yip, P. (1990). *Discrete Cosine Transform: Algorithms, Advantages, Applications*. New York: Academic.

Rui, X., Chang-sheng, Y., & Guang-hua, S. (2003). Wavelet Image Coding based on Dependency Predication. *Journal of Zhejiang University(Engineering Science), 37*.

Said, A. (1999). Example of Application for Image Compression.

Said, A., & Pearlman, W. A. (1996). A New Fast and Efficient Image Codec Based On Set Partitioning in Hierarchical Trees. *IEEE Transactions on Circuits and Systems for Video Technology, 6*.

Shapiro, J. M. (1993). Embedded image coding using zerotrees of wavelet coefficient. *IEEE Transactions on signal Processing, 41*.

Sobrevilla, P., & Montseny, E. (2003). Fuzzy Sets in Computer Vision: an Overview. *Mathware & soft computing, 10*, 71-83.

Tizhoosh, H. R. (1997). *Fuzzy Image Processing: Introduction in Theory and Application*: Springer-Verlag.

Tuy, H. A., & An, N. V. (2005). Image Compression Using The Haar Wavelet. *Journal of Science. Mathematics- Physcis, 2*.

Valens, C. (1999-2004). EZW encoding.

Ville, D. V. D., Nachtegael, M., Weken, D. V. d., Kerre, E. E., Philips, W., & Lemahieu, I. (2003). Noise Reduction by Fuzzy Image Filtering. *IEEE Transactions on Fuzzy System, 11*.

Walker, J. S. Wavelet-Based Image Processing.

Wang, Z., & Bovik, A. C. (2001). Embedded Foveation Image Coding. *IEEE Transactions on Image Processing*.

Wei-guo, W., & Bao-long, G. (2000). On Embedded Zerotree Wavelets Coding and other Improved Algorithms.

Wolfgang, R. JPEG Tutorial.

Xiao, P. (2001). *Image Compression By Wavelet Transform.* East Tennessee State University.

Xing, M. Y., & Zhong, Y. S. (2003). Image Compression Method of Vector Coding Based on Wavelet Transform. *Computer Application, 23.*

Bogomjakov, D. P.-A. (1998). Lossless Image Compression

Conversion of a Color Image to a Grayscale Image. from http://www.codersource.net/csharp_conversion_color_gray_image.aspx

Franti, P. (2002). Image Compression

Gray Scales. from http://www.engr.udayton.edu/faculty/jloomis/ece563/notes/color/GrayScale/grays.html

Image Compression techniques. from http://www.ee.bgu.ac.il/~greg/graphics/compress

TheMathWorks from http://www.mathworks.com/access/helpdesk/help/toolbox/wavelet/wavelet.html.

Li, Z.-N., & Drew, M. S. (2004). Fundamentals of Multimedia: Prentice hall.

Mai, L. C.Introduction to Computer Vision and Image Processing

Marcellin, M. W., Gormish, M. J., Bilgin, A., & Boliek, M. P. (2000). An Overview of JPEG-2000. IEEE Data Compression Conference, 523-541.

Planinsie, P. (2000). Standard and Subjective Images Quality Control of Subband Coded Images using Fuzzy Logic and Neural Networks. Paper presented at the 8th Internation Workshop on System, Signal and image processing.

Shapiro, J. M. (1993). Embedded Image Coding Using Zerotrees of Wavelet Coefficients. IEEE Transactions on signal,processing, 41.

Short Term Load Forecasting Using Neural Networks and Fuzzy logic. from http://www4.eas.asu.edu/PowerZone/LoadForecast/fuzzy.htm

Willam, k., Rhee, I., & Beylkin, G. Multiresolution Analysis Elastic Degradation in Heterogenous materials. University of Colorado at Boulder, Boulder CO 80309-0428, USA.

Zehtab-Fard, P.Still Image Compression

Advanced Color Imageing on the Mac OS. (1996). In: Apple Computer.Inc.

Betz, S., Bhagat, N., Murphy, P., & Stengler, M.Wavelet-Based Image Compression, from
http://www.owlnet.rice.edu/~elec301/Projects00/wavelet_image_comp/img-compression-theory.html

Kumar, S. (2001). An Introduction to Image Compression, from
http://www.debugmode.com/imagecmp/

Miguel, A. F. (1999). *Set partitioning in Hierarchical Trees (SPIHT)*, from
http://dcl.ee.washington.edu/amiguel/spiht.html#SaidP:96

Elzinga, J., & Feenstra, K. (2001). *JPEG 2000: The Next Compression Standard Using Wavelet Technology*, from
http://www.gvsu.edu/math/wavelets/student_work/EF/index.html

Dipperstein, M. (2005). *Lempel-Ziv-Welch (LZW) Encoding Discussion and Implementation*

USC-SIPI Image Database - Miscellaneous.
http://sipi.usc.edu/database/database.cgi?volume=misc