2012

Lossy image compression using wavelet transform

Huda Al-Ghaib

LOSSY IMAGE COMPRESSION USING WAVELET TRANSFORM

by

HUDA AL-GHAIB

A THESIS

Submitted in partial fulfillment of the requirements

for the degree of Master of Science in Engineering

in

The Department of Electrical and Computer Engineering

to

The School of Graduate Studies

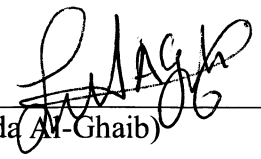
of

The University of Alabama in Huntsville

Huntsville, Alabama

2012

In presenting this thesis in partial fulfillment of the requirements for a master's degree from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department or the Dean of the School of Graduate Studies. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.



(Huda Al-Ghaib) Dec. 19. 2011 (date)

THESIS APPROVAL FORM

Submitted by Huda Al-Ghaib in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering and accepted on behalf of the Faculty of the School of Graduate Studies by the thesis committee.

We, the undersigned members of the Graduate Faculty of The University of Alabama in Huntsville, certify that we have advised and/or supervised the candidate on the work described in this thesis. We further certify that we have reviewed the thesis manuscript and approve it in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering.

12/19/2011
Committee Chair
(Date)

Heggere S. Ranganath 12-19-2011

Mark Tin 1-9-2012

Department Chair

College Dean

Rhonda Kay Shede 3/7/12 Graduate Dean

ABSTRACT

School of Graduate Studies
The University of Alabama in Huntsville

Degree Masters of Science in Engineering College/Dept. Engineering/Electrical and Computer Engineering

Name of Candidate Huda Al-Ghaib

Title Lossy Image Compression using Wavelet Transform

Image compression is the process of compacting data and maintaining the necessary information to represent an image. Image compression minimizes the memory requirements for storing data, especially when dealing with images that have large dimensions. This research focuses on image compression using wavelet transform to obtain "visually lossless" compression. A new technique which achieves a better compression ratio than the existing methods is presented. Different standards such as: BMP, JPEG2000 and JPEG are compared with the new algorithm.

Abstract Approval: Committee Chair

Department Chair

Graduate Dean Rhonda Kay Daede 3/7/12

Dedication

This thesis is dedicated to the people who taught me the alphabet,

To my mother, who taught me the noun

'God'

To my father, who taught me the verb

'Smile'

To my brother and sisters, who taught me the adjective

'Motivation'

To my brothers and sister in law, who taught me the adverb

'Hardworking'

To my niece and nephews, who taught me the characters

'L-O-V-E'

To my advisor, who taught me the sentence

'Lossy Image Compression using Wavelet Transform'

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

LIST OF CHARTS

Page

CHAPTER I

INTRODUCTION

Numerous applications use digital images as their main source of information, including magnetic resonance imaging (MRI), computed tomography or CT scans, and X-rays. Another example is forensic imaging, where studies are done on images to obtain information that is necessary for the protection of society. In the field of pattern recognition, digital images are used to detect and recognize different objects such as cars and human bodies. Digital images can also be used as a search tool. The need for high quality images in these different applications created an important field called "digital image processing," which is a subcategory of digital signal processing [1]. In this field, computer algorithms such as image filtering and de-nosing are performed in order to enhance images. Algorithms such as, image resizing, thresholding, and segmentation can be performed to manipulate images and to obtain extra information for a specific application. Once an image is enhanced or manipulated, it needs to be stored on a local storage device or transmitted. Digital images with good quality require a large storage space. Presentations that minimize the size of images are still required, especially when large amounts of images need to be archived. The transmission of a large amount of

images is a time consuming process. The need for storage mass reduction and an efficient transfer of images resulted in the field of image compression. Image compression is the process of compacting data contained within images while still maintaining the necessary information to represent them [2].

Generally, images contain redundant information. The image compression process tends to find a representation that either minimizes or removes the redundancy. Two-dimensional intensity arrays contain two types of redundant data: coding redundancy and spatial redundancy. Coding redundancy results from the specified storage unit, i.e., units of storage that are utilized to encode more information than is required. Adjacent cells within images are usually correlated. This correlation results in spatial redundancy. Additionally, two-dimensional intensity arrays contain irrelevant information that is usually ignored by the human visual system (HVS). In our survey, we will consider the irrelevant information as a part of data redundancy, since removing it would help to compress the data.

Image compression schemes tend to reduce the size of images by considering different data redundancies. Different compression algorithms have been developed for this purpose. The following figure represents the general scheme for image compression [31].

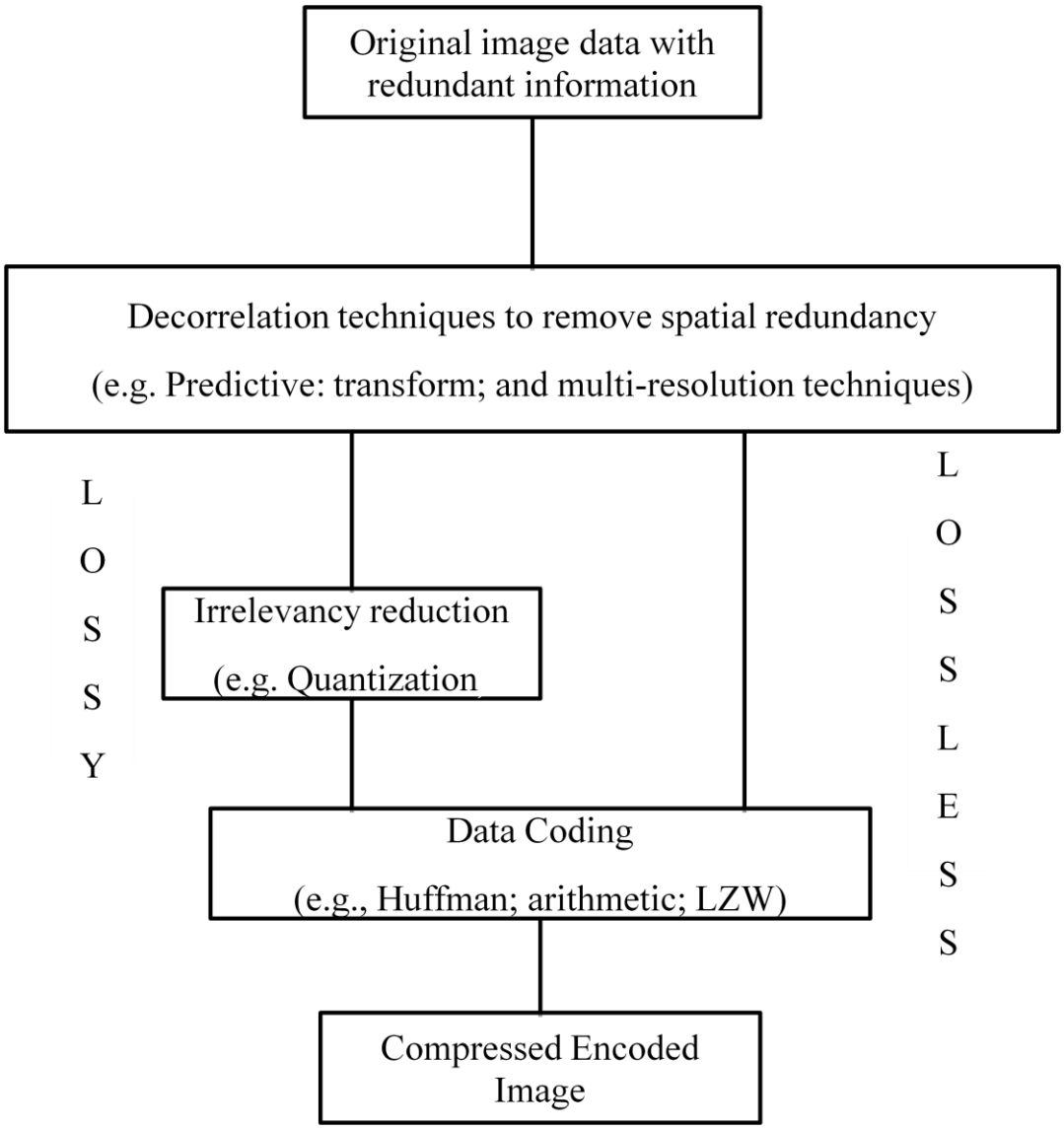


Figure 1.1 General scheme for image compression

The general procedure or algorithm for image compression consists of two steps: decorrelation and entropy coding [3-4]. An optional step (irrelevancy reduction) can be added to produce a lossy image compression; otherwise the compression is called lossless image compression. Lossless image compression is also known as reversible image compression or noiseless coding. It preserves all information, i.e., numerical differences

between the original and compressed pixels equal zero. The highest compression ratio that is achieved using lossless compression is 2:1 [2], but this can vary depending on the nature of the image. It is usually used with applications that do not allow the sacrificing of some information in order to obtain a higher compression ratio. Some examples of these applications are medical imaging, as well as biomedical, telemetry forensic, and geophysics applications [1], [3]. In medical images, losing some information may affect the diagnostic accuracy. Lossy compression sacrifices some information in order to obtain a higher compression ratio. It is an acceptable method of compression for applications that do not require an error free reproduction of the original image. Some lossy compressions provide "visually lossless" images, meaning that the actual data has been lost, but is not sensitive to visual appearances [4].

As shown in Figure 1, the first step in an image compression algorithm is decorrelation. Decorrelation tends to find a less correlated representation by removing spatial redundancy. There are different techniques for decorrelation, including predictive, transform and multi-resolution techniques. In a linear prediction technique, the prediction of each sample is calculated based on its neighboring samples. A transform technique is frequently employed for data compression, since it excels at decorrelation. An image transform finds a less decorrelated representation for data coefficients by transforming them from spatial domain to other domains. Some examples of these transforms are the discrete cosine transform (DCT) and discrete Fourier transform (DFT). These transformations are difficult to implement in lossless image compression because the resulting coefficients are real or complex values that must be quantized for the encoding stage. Transforms such as the Walsh-Hadamard transform (WHT) can be applied for

lossless compression because their coefficients are binary fractions that need no quantization. In a multi-resolution technique, the original data is represented with different resolutions, resulting in a hierarchical representation of data sets. Some examples of multi-resolution techniques are hierarchical interpolation (HINT), the Laplacian pyramid, and S-transform [3]. A hybrid approach can be performed for the decorrelation step by combining different techniques (such as the predictive and transform coding).

The second step in Figure 1 is irrelevancy reduction. This step is employed only for lossy compression, since it removes parts of the data contained within images. Quantization and/ or thresholding are employed to perform irrelevancy reduction. The quantization process quantizes data coefficients, while thresholding removes data that is not noticeable by the receiver.

The last step in image compression is data encoding. Processes such as Huffman coding or arithmetic coding are employed in order to reduce coding redundancy by reducing the number of units (i.e., bits) that are required to represent each pixel.

CHAPTER II

IMAGE COMPRESSION

2.1 Introduction

Digital communication has largely replaced analog communication. Digital images require large storage spaces. Also, the transmission of images to users in remote locations requires the transmission of large amount of data. A gray image that is 512x512 pixels requires 2Mega bits of memory to store it (when 8-bit per pixel (BPP) is used). For true color images, the initial BPP is 24, with three colors RGB (Red, Green, and Blue) being used to represent each pixel, and 8 bits being used to encode each of the three colors. Therefore, the total number of bits is equal to 512x512x24= 6 Mega bits. As a result, storing larger-sized images would require a larger memory. That being said, how does one represent digital images with less information and still maintain the images' quality? One way to do so is by considering the image compression approach. Data compression helps remove redundant information contained within images while still maintaining the necessary information to represent them [2].

Data and information represent two different terms. Data is used to convey information, i.e., different amount and kinds of data can be used to represent the same amount of information [6]. Generally, the data used to represent information within

images is a 2-D array with M rows and N columns. (x, y) is the discrete coordinate representing the pixel location, where $x = 0,1,2, \dots, M-1$ and $y = 0,1,2, \dots, N-1$. $f(x_i, y_j)$ represents the intensity value of an image at a given coordinate (x_i, y_j).

Data is said to be redundant if it contains irrelevant or repeated information. If b is the number of bits required to represent an image with redundant information, then b'be the number of bits used to represent that same image after compression. C the compression ratio between b and b'can be defined as:

$$C = \frac{b}{b'} \tag{2.1}$$

A compression ratio C with a value of 10 (or 10: 1) indicates that b has 10 units of information compare to one unit of b'.

C can be employed to find R, where R represents the relative data redundancy, i.e., the amount of redundant data within an image, as follows:

$$R = 1 - \frac{1}{C} \tag{2.2}$$

Data redundancy results from having repetitive information. Images contain three kinds of redundancy: coding redundancy, spatial redundancy, and irrelevant information. A code is a system of symbols such as letters, numbers, or bits that is used to represent different kinds of data. 8 BPP are typically used to represent digital images with 2-D intensity arrays. Since intensity values vary from 0-255 (where 0 represents black and 255 represents white), cells with low intensity values in the 2-D intensity arrays contain more bits than required to represent them. This results in coding redundancy. The correlation between neighboring pixels corresponds with spatial redundancy. Irrelevant information refers to information within images that is usually ignored by the viewer.

Sections 2.2 through 2.5 explain the different redundancies within images and illustrate an example for each one.

In chapter one, a general scheme for image compression has been explained. It consists of three steps that are used to remove different data redundancies. These steps are: decorrelation or mapper, irrelevancy reduction, and date coding. Combining the three steps together results in a system that is known as an *encoder*. A d*ecoder* accepts the output of an encoder and performs the inverse operation to retrieve the original data. A typical decoder consists of two operations, the symbol decoder and the inverse mapper. Two approaches can be used to compress images, lossy and lossless. The general scheme for image compression, in addition to an explanation of the terms *lossless* and *lossy* image compression, is given in Section 2.5. Sections 2.6 through 2.8 explain the main function for each block in the general scheme; techniques for each block are also given.

2.2 Coding Redundancy

Information is represented by using a system of symbols, such as letters or numbers. A system of symbols is known as a *code*. In general, information is subdivided into pieces and each piece is represented using a *code word*. Each code word contains a sequence of symbols, with the number of symbols representing the *code length*. Coding redundancy occurs when the code length is longer than is necessary to adequately represent the symbols. Each symbol within a system has a probability of occurrence, which is defined as the number of iterations for that symbol over the total number of symbols.

For an image of size MxN with an intensity value k that varies from$[0, L-1]$. The probability of the occurrence of intensity value r_k is defined as:

$$p_r(r_k) = \frac{n_k}{MN} \tag{2.3}$$

n_k represents the number of times the kth intensity appears in an image. The average number of bits that are required to represent each pixel is equal to:

$$L_{avg} = \sum_{k=0}^{L-1} l(r_k) p_r(r_k) \tag{2.4}$$

where, $l(r_k)$ is the number of bits required to represent each value of r_k. The total number of bits required to represent an image is equal to MNL_{avg}. If a code with a fixed length is used, then $l(r_k) = constant$.

Thus,

$$L_{avg} = \sum_{k=0}^{L-1} constant.\, p_r(r_k) \tag{2.5}$$

The constant value can be taken outside the summation, with the sum of $p_r(r_k)$ for $0 \leq k \leq L-1$ being equal to one. As a result, L_{avg} for a fixed length code is equal to the constant value.

To illustrate coding redundancy within an image, consider a 4x4 image with the following intensity values.

117	121	166	117
121	105	166	166
117	117	105	117
105	105	117	117

Figure 2.1 Image A with different intensity values

The image in Figure 2.1 can be decoded by using a code with either a fixed or variable size. Table 2.1 illustrates the probability distribution for the different intensity values. Two different codes with fixed and variable lengths are also used to decode the image in Figure 2.1.

$\mathbf{r_k}$	$\mathbf{p_r(r_k)}$	**Code 1**	$\mathbf{l_1(r_k)}$	**Code 2**	$\mathbf{l_2(r_k)}$
$\mathbf{r_{105}}$	0.25	01101001	8	01	2
$\mathbf{r_{117}}$	0.4375	01110101	8	1	1
$\mathbf{r_{121}}$	0.125	01111001	8	000	3
$\mathbf{r_{166}}$	0.1875	10100110	8	001	3
$\mathbf{r_k}$ for $k \neq 105,117,121,166$	0	-	8	-	0

Table 2.1 Probability distribution for intensity values in the image in Figure 2.1

The image in Figure 2.1 has four possible intensity values with a different probability of occurrence. If code 1 with fixed length that is equal to 8-bits, is used to represent them, then $l(r_k) = 8\ bits$ for all r_k and L_{avg} is equal to the average number of bits, i.e., 8 bits. On the other hand, if code 2 with a variable length is used to code the image, then the number of bits that is used to represent an intensity value depends on its probability of occurrence. In other words, fewer bits are assigned to the most probable intensity values and more bits are assigned to the lesser intensity values. r_{117} is the most probable intensity in the image in Figure 2.1, so it assigned a 1-bit code word with a value of 1 and a length of $l_2(r_{117}) = 1$. r_{166} (the least probable intensity) is assigned 3-bits code words (001) and a length of $l_2(r_{166}) = 3$. The average length of the encoding pixels for code 2 is calculated using equation (2.5) as follows:

$$L_{avg} = 0.25 * 2 + 0.4375 * 1 + 0.125 * 3 + 0.1875 * 3 = 1.875\ bits$$

The total number of bits required to represent the image is equal to MNL_{avg}=4*4*1.875=30. Equations (2.1) and (2.2) can be implemented to calculate the compression ratio and the relative redundancy.

$$C = \frac{8}{1.875} \approx 4.2667 \tag{2.6}$$

$$R = 1 - \frac{1}{4.2667} = 0.7656 \tag{2.7}$$

Using code 1 with an 8-bit length results in a representation that has 76.56% redundant information compared to code 2.

Code redundancy occurs when the code assigned for a set of events (or intensity in the case of images) does not take full advantage of the probability of occurrence of the events [6]. When a natural binary code is used to encode images, the code redundancy is almost always present. Generally, images contain objects with different shapes and as a result certain intensities have a higher probability of occurrence than others. Figure 2.2a and 2.2b, represent an image and its histogram. It is obvious from the non-uniform histogram that different intensity values have a different probability of occurrence. If a natural binary encoding was used to encode the image in Figure 2.2a, it would assign the same number of bits to both the most and least probable values. This would not minimize equation (2.4), thus resulting in coding redundancy.

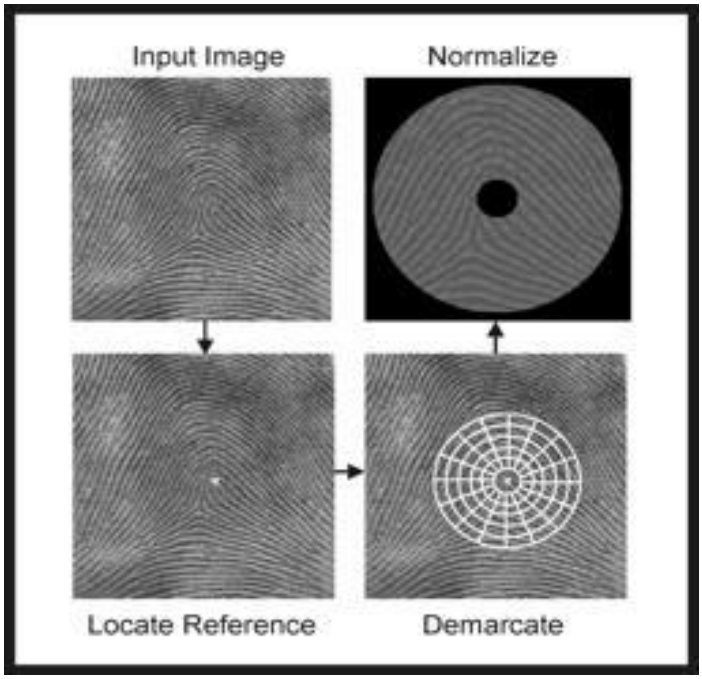


Figure 2.2a 255x255 image with coding redundancy

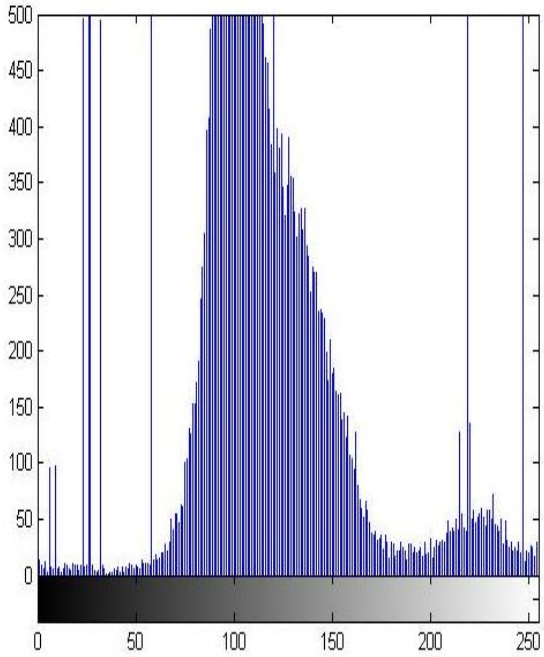


Figure 2.2b The histogram for the image in Figure 2.2a

In order to reduce coding redundancy, a code of variable length is needs to be used to encode the image in Figure 2.2a.

2.3 Spatial Redundancy

In general, pixels are correlated spatially, meaning that adjacent pixels in a uniform region within an image contain similar intensity values. Reducing the spatial redundancy helps to reduce the size of images significantly. Operations such as run length and the differences between adjacent pixels can be used to reduce spatial redundancy. These operations perform transformation on pixels in order to map them into a less correlated representation. Mapping is called reversible, if the value for each pixel in

the original 2-D intensity array can be reconstructed without error. Otherwise, the mapping is considered to be irreversible. Figure 2.3 represents a gray image [29].

Figure 2.3 Gray image

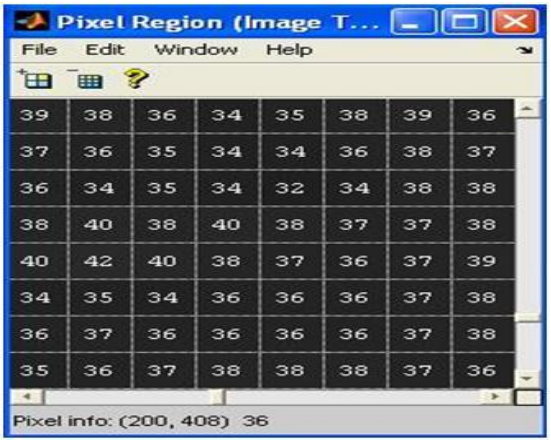


Figure 2.4 Intensity values for an 8x8 region from the image in Figure 2.3

Figure 2.4 represents the intensity values for a uniform region in the image in Figure 2.3. The intensity values in Figure 2.4 vary from 32-42 with a different probability of occurrence, for example $p_r(r_{36}) = 16$. The following chart represents the probability of occurrence for the pixels in Figure 2.4.

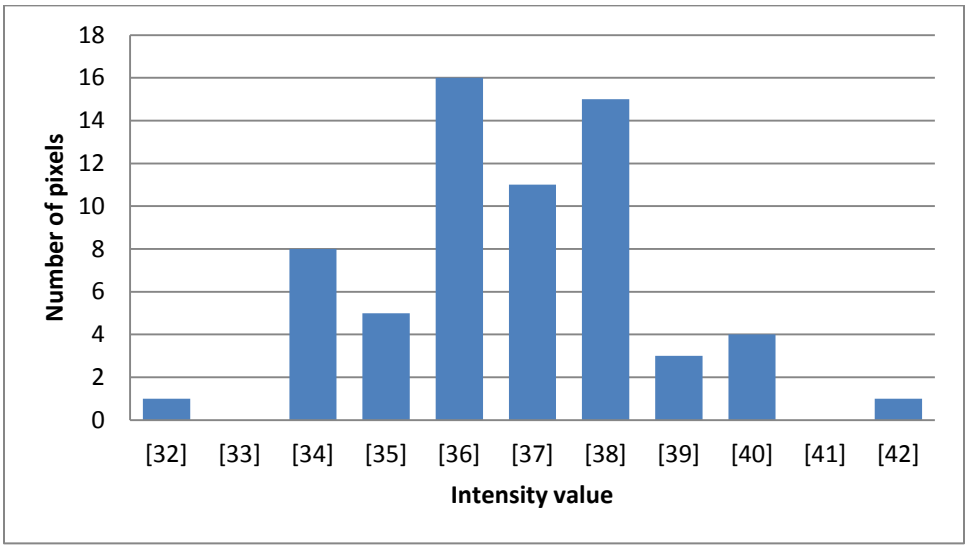


Chart 2.1 Probability of occurrence for the pixels in Figure 2.4

In Chart 2.1, the x-axis represents the intensity values, i.e., 32-42 and the y-axis represent the number of pixels with the corresponding intensity value. The pixels are correlated since the intensity values occur more than once. When a character or an intensity value occurs more than once, it is called a *run*, while the number of repetitions for that run is known as *length*. Representing that region by using a fixed-length code with 8 BPP results in a total number of bits equal to $MNL_{avg} = 8 * 8 * 8 = 512$. Since the pixels are correlated, compression can be performed to reduce the size of an image. One way to do this is by using *Run-length coding*. By using this code, any run of length can be represented using only two characters. For example the string rrrrrryuuuuu is equivalent to @r5t@u6 [2].

2.4 Irrelevant Information

Irrelevant information refers to images containing information that is usually ignored by the human visual system. The simplest way to compress these images is to omit this information. Figure 2.2b represents the histogram for the image in Figure 2.2a with the intensity values for the image in Figure 2.2a varying from 0-255. In this case, 0 represents black and 255 represents white. Pixels with small intensity values, such as 0-60 for the image in Figure 2.2a, are usually ignored by the HVS because this system averages the intensity values and thus perceives only the average value. Omitting these low intensity values would not affect the quality of the image. Figure 2.5 represents the same image in Figure 2.2a after setting pixels with intensity values less than 60 to zero.

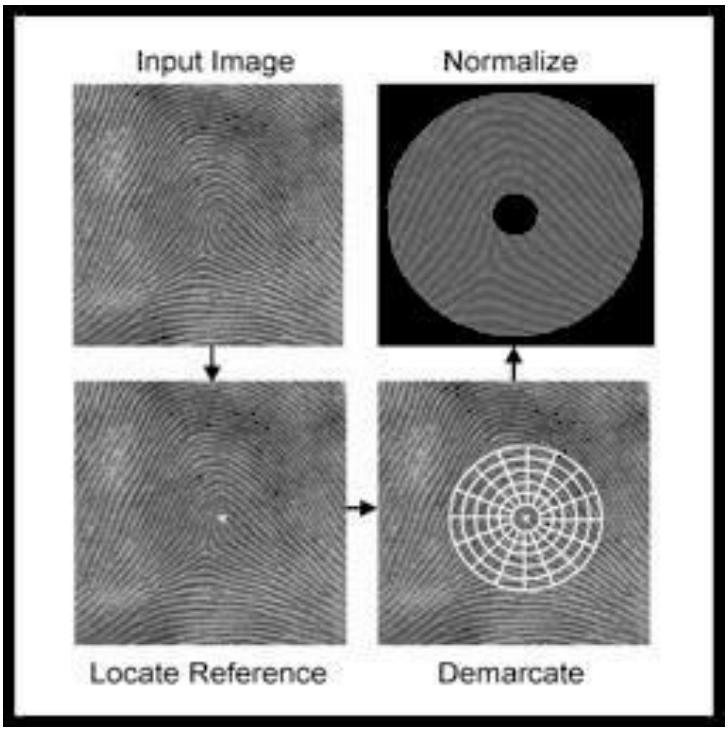


Figure 2.5 Image with omitted information

Processes, such as thresholding, can be used to omit this information. Thresholding increases the compression performance by omitting information with no significant importance. Nevertheless, the thresholding process is an application dependent operation. Some applications, such as X-ray archives and computerized electronic radiographs, consider all the information contained within images as being equally important and do not omit any information.

2.5 Image Compression Scheme

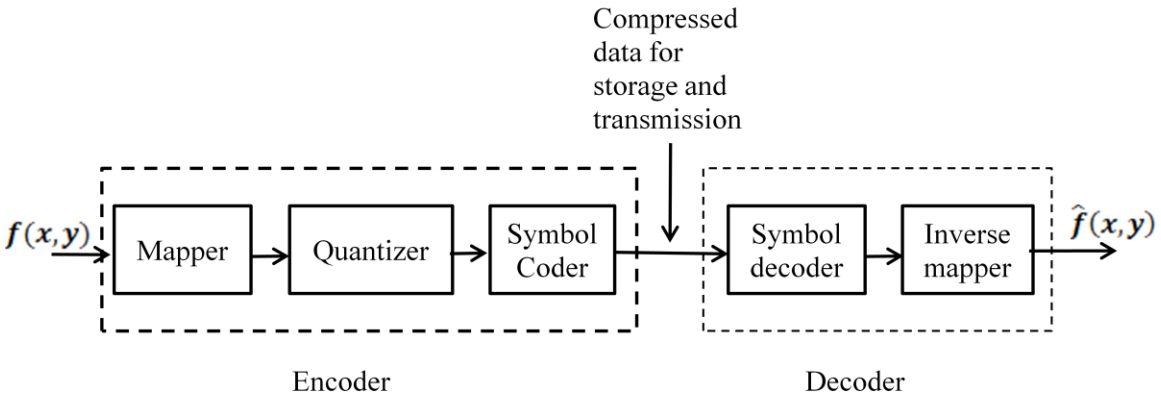


Figure 2.6 Image compression scheme

The image compression scheme consists of two sub-systems: the first sub-system is called the encoder and performs the compression operation. The second sub system is called the decoder and performs the complementary operation, i.e., decompression. If the same system performs both operations then it is known as codec. These two operations can be performed in either software or hardware [6].

The input to the encoder is an image $f(x,y)$ with redundant information. The encoder creates a compressed representation for $f(x,y)$, which is either stored locally or transmitted for later use in a remote location. Later on, the output of the encoder is fed to the decoder, where the operation of original image reconstruction $\hat{f}(x,y)$ is preformed. If $\hat{f}(x,y) = f(x,y)$, then the compression system is a reversible system that preserves information. This system is known as a lossless system and provides a complete reconstruction of original images. It is used with applications that do not sacrifice

information. Examples of these applications are medical imaging, as well as biomedical, telemetry, forensic, and geophysics applications. On the other hand, if $\hat{f}(x,y) \neq f(x,y)$, then the system is not reversible and is known as lossy system. Such a system may provide visually lossless images, meaning that it omits information that is not noticeable by the HVS.

The main function of the encoder is to remove data redundancies and produce a less relevant representation with minimum size. The encoder consists of three independent operations that together perform the main function of the encoder (compressing the data). The following three sections explain the main function for each operation.

2.5.1 Mapper

The first block in the encoder is the mapper. The main function of the mapper is to decorrelate the data. The mapper does not necessarily compress the data; it mainly reduces spatial redundancy. Predictive, transform, and multi-resolution techniques are used to perform the mapping operation.

2.5.1.1 Predictive Techniques

This technique is a linear prediction technique that performs mapping/ decorrelation based on the correlation between adjacent pixels, or blocks, in the spatial domain. It helps produce a less correlated representation by decorrelating/ eliminating the redundancy between adjacent pixels. In this technique, a predictor is employed to predict the value of each pixel and *new information* is calculated as the difference between the

actual and predicted value. Instead of coding the information contained within pixels, the *new information* is coded. Predictive coding can be used for both lossless and lossy image compression. Figure 2.7 represents the general scheme for a lossless predictive coding system [6].

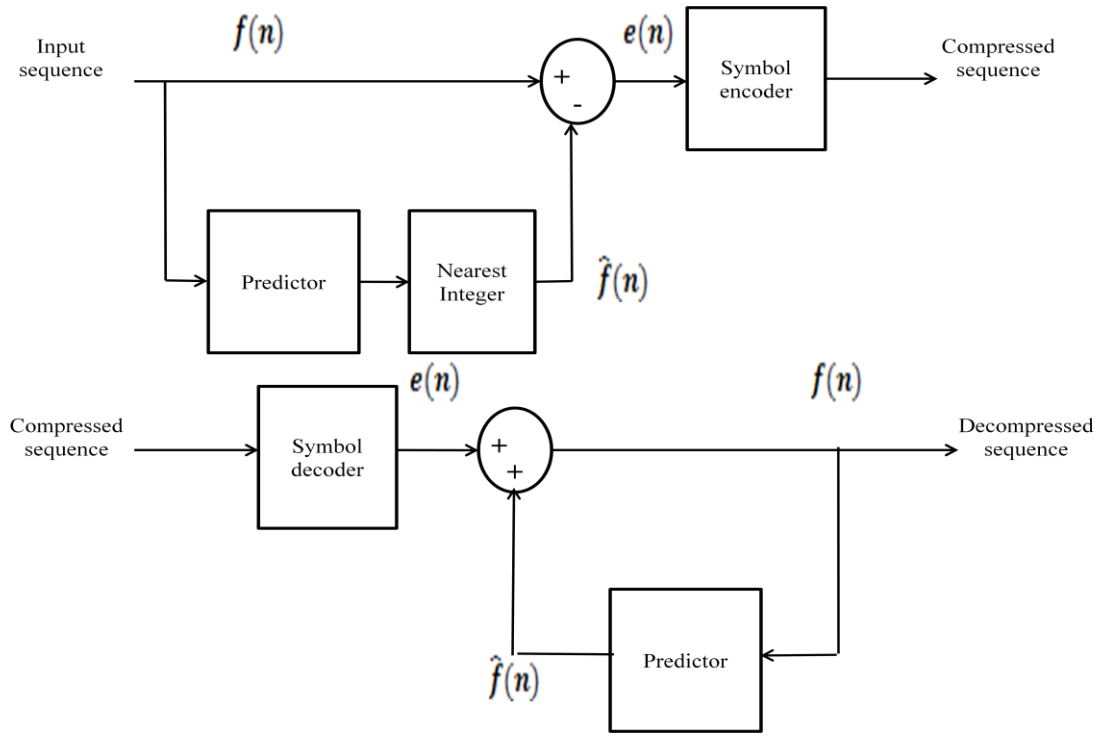


Figure 2.7 Lossless predictive coding system

The system in Figure 2.7, consists of the encoder and decoder. The discrete input signal $f(n)$ is fed to the encoder as a sequence of samples. The Predictor block generates the anticipated value for each sample based on a specified number of past samples. The output of the predictor is then rounded to the nearest integer value using the Nearest

Integer block. The output of the Nearest Integer block is denoted as $\hat{f}(n)$ and is used to calculate the new information value using the following equation:

$$e(n) = f(n) - \hat{f}(n) \tag{2.8}$$

$e(n)$ is also denoted as the prediction error. If the predictor is well chosen, then the result of equation (2.8) should be small. The entropy (data entropy is given in Section 2.5.3) of the prediction error should be much smaller than the entropy of the original data. A code with a variable length can be used to code the prediction error and generate the compressed data stream [6-7]. The decoder in Figure 2.7 reconstructs $e(n)$ from the received code. The original input sequence is recreated using the following equation:

$$f(n) = e(n) + \hat{f}(n) \tag{2.9}$$

Various methods can be employed to determine the best prediction value $\hat{f}(n)$ for an input sequence $f(n)$. In the following equation, the $\hat{f}(n)$ prediction is performed based on the linear combination of previous m samples.

$$\hat{f}(n) = round[\sum_{i=1}^{m} \propto_i f(n-i)] \tag{2.10}$$

Where m is the order of the linear predictor and $\propto_i$ for $i = 1,2, \dots, m$ are the prediction coefficients. For a 2-D signal like an image, the input to equations (2.9) and (2.10) is pixels instead of samples. Thus, equation (2.10) can be rewritten as follows:

$$\hat{f}(x,y) = round[\sum_{i=1}^{m} \propto_i f(x,y-i)] \tag{2.11}$$

In equation (2.11), the m samples are used to predict the value of each pixel. The m samples' results from the current and previous scan are denoted as 2-D linear predictive coding. For images, the prediction is a function of a left-to-right, top-to-bottom scan. As a

result, equation (2.11) cannot be evaluated for the first m pixels in each line; these pixels need to be encoded separately.

When compression performance is preferable to reconstruction accuracy, lossy compression is considered to be the desired approach. Figure 2.8 represents the lossy predictive coding system.

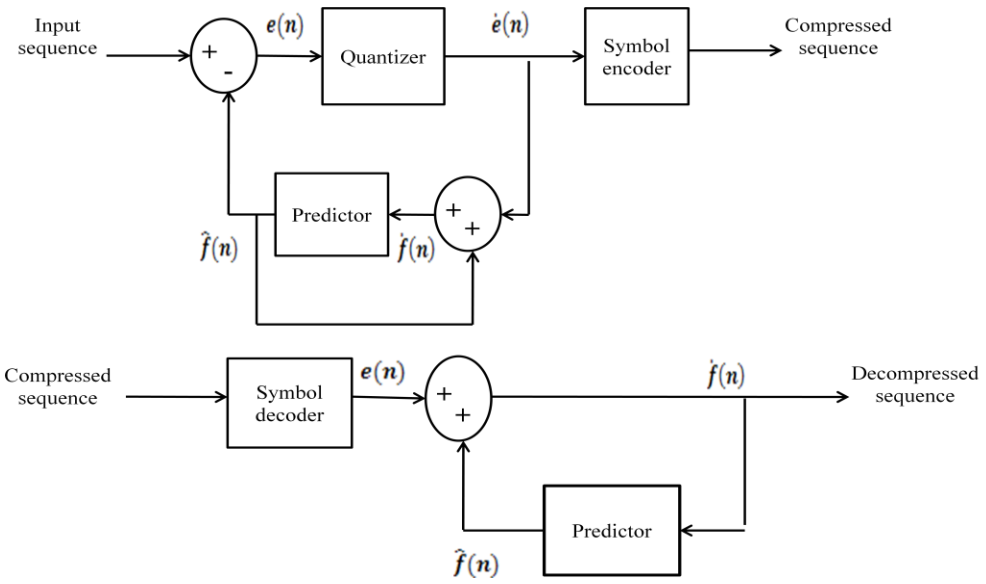


Figure 2.8 Lossy predictive coding

Lossy predictive coding has a Quantizer block that performs irrelevancy reduction. The Quantizer block maps the prediction error $e(n)$ into a limited range of output that is denoted as $\dot{e}(n)$. Since the prediction error generated by the encoder and decoder has to be equivalent, changes that occur in the data as a result of quantization must be altered to the encoder. A feedback loop for the encoder's predictor can be used for this purpose. The quantized new input is denoted as $\dot{f}(n)$ and can be generated as follows:

$$\dot{f}(n) = \dot{e}(n) + \hat{f}(n) \tag{2.12}$$

In equation (2.12), $\hat{f}(n)$ represents the past prediction, while $\dot{e}(n)$ represents the quantized error that is associated with it.

One example of lossy predictive coding is the Delta Modulation (DM), in which the predictor and quantizer are defined as follows:

$$\dot{f}(n) = \propto \dot{f}(n-1) \tag{2.13}$$

$$\dot{e}(n) = \begin{cases} +\zeta & for\ e(n) > 0 \\ -\zeta & otherwise \end{cases} \tag{2.14}$$

$\propto$ represents the prediction coefficients and is usually less than 1, with ζ being a positive constant. If the predictor is chosen in order to minimize the encoder's mean-square prediction error

$$E\{e^2(n)\} = E\{\left[f(n) - \hat{f}(n)\right]^2\} \tag{2.15}$$

then equation (2.15) is applicable only if:

$$\dot{f}(n) = \dot{e}(n) + \hat{f}(n) \approx e(n) + \hat{f}(n) = f(n) \tag{2.16}$$

and

$$\hat{f}(n) = \sum_{i=1}^{m} \propto_i f(n-i) \tag{2.17}$$

This would reduce the quantization error that is assumed to be negligible in this case, i.e., $\dot{e}(n) \approx e(n)$. The prediction would thus be constrained to a linear combination of previous m samples. This would decrease the computational complexity of the predictor. The predictors in this case are known as *optimal predictors*, and the resulting coding is known as *differential pulse code modulation* (DPCM).

2.5.1.2 Transform Techniques

Transformations such as DCT, DFT, and WHT can be used to transform images from the spatial domain to a different domain, i.e., 2-D DFT is used to transform an image from the spatial domain to the frequency domain. Transform-based coding subdivides an image into a number of blocks, with each block consisting of a number of pixels, and it is represented as a weighted sum of basis functions. The basis functions can be orthnormal, which means that the inner product $\langle f(t), f(t) \rangle$ of a function that has a length of one is normalized to one [8].

$$length\ squared = \int_{-\infty}^{\infty} (f(t))^2 dt = 1 \tag{2.18}$$

Image is transformed by projecting each block onto each basis function [9]. Chapter IV explains the generalized image transform and gives examples of different transfers.

2.5.1.3 Multi-Resolution Techniques

Images contain objects with different sizes. Objects with small sizes are usually low in contrast and need to be examined using a high resolution representation, while objects with large sizes are high in contrast and can be examined at a low resolution. This being the case, images can be studied using different resolutions. This is known as the theory of multi-resolution processing.

Multi-resolution process produces a representation that is known as image pyramid which consists of number of levels to represent an image with original size NxN using different resolutions. The base of the pyramid has an image with high resolution representation while the apex contains the same image with a low-resolution representation. The base, i.e., level J, is calculated as $J = log_2 N$. The apex, i.e., level 0, is

of size 1x1 and any level between base and apex is of size 2^jx2^j where $0 \leq j \leq J$. As a result, the pyramid is composed of $J + 1$ level. Image pyramid can be truncated into $P + 1$ levels,

where $1 \leq P \leq J$ and $j = J - P, \ldots, J - 2, J - 1, J$.

Each level j the approximation output for level $j - 1$ is found by using an approximation filter such as Gaussian filter and downsampling operation on dyadic base. A Gaussian filter of two variables has the basic form:

$$h(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.19}$$

Where σ is the standard deviation and x and y are integers.

The downsampling operation is defined as follows:

$$f_{2\downarrow}(n) = f(2n) \tag{2.20}$$

Downsampling on dyadic base is simply the process of discarding every other sample. The following system illustrates the process of obtaining an approximation output for level $j - 1$. Also, the prediction residual in Figure 2.9 represents the lost information from level j image. The prediction residual can be used later to retrieve the level j image with error free or small amount of error.

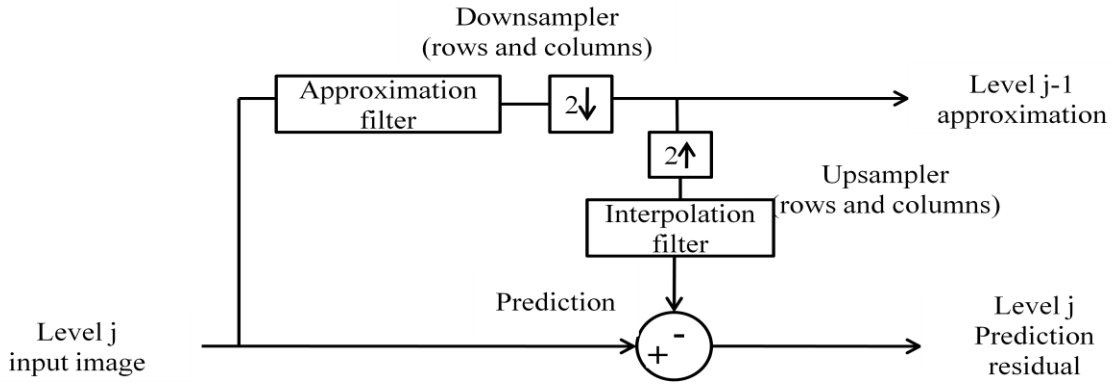


Figure 2.9 A system for creating approximation and prediction residual pyramids

The prediction residual for level j can be obtained from upsampling the level $j-1$ approximation on a dyadic base and using an interpolation filter such as bilinear interpolation filter. Upsampling operation is defined as follows:

$$f_{2\uparrow}(n) = \begin{cases} f\left(\frac{n}{2}\right) & if\ n\ is\ even \\ 0 & otherwise \end{cases} \tag{2.21}$$

Upsampling is the process of inserting a 0 after every sample in a sequence.

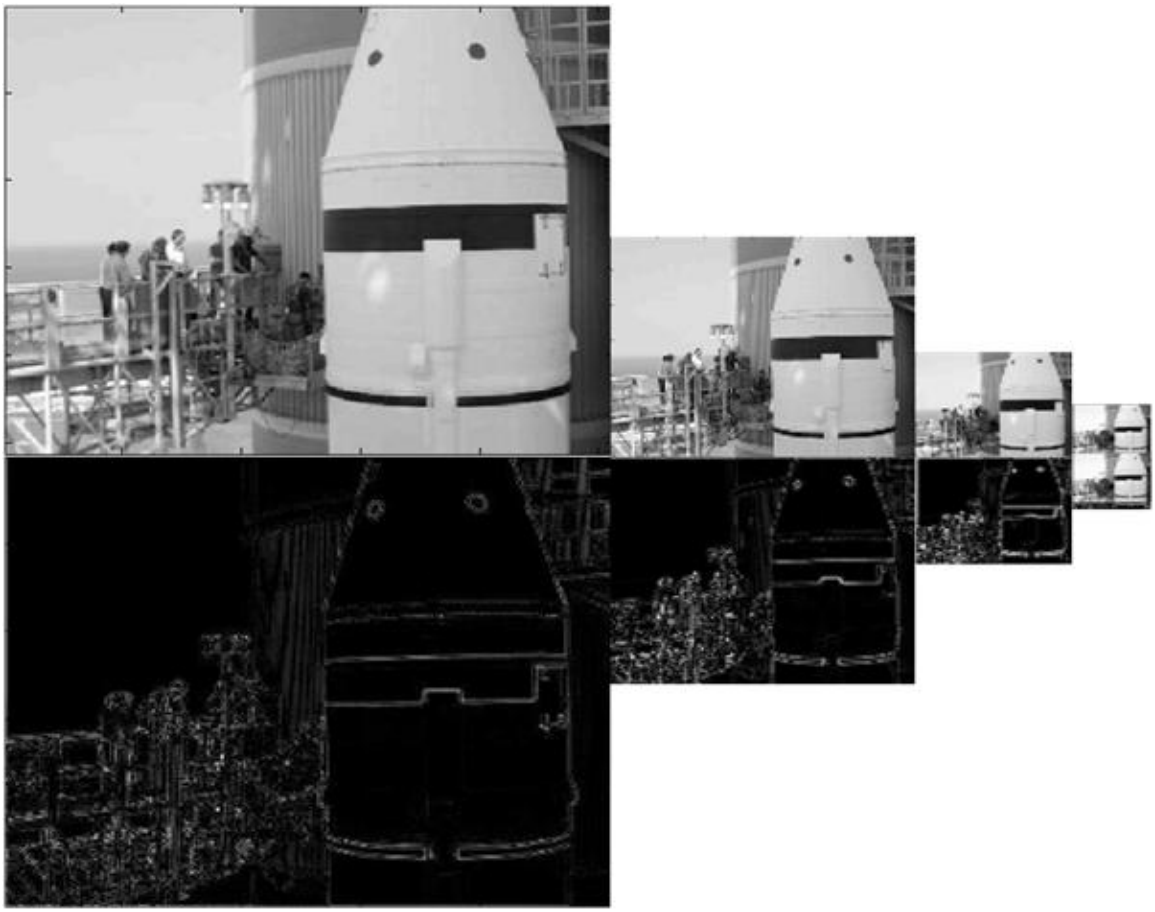


Figure 2.10 An approximation and prediction residual pyramids

2.5.2 Irrelevancy Reduction

The methods explained in section 2.5.1 are used to remove the spatial redundancy and decorrelate the data. Once the data is decorrelated, an extra compression can be obtained by removing parts of the information that are unnoticeable by the HVS. As this step changes, the data contained within an image is performed only in a lossy image compression.

Quantization can be employed for the purpose of irrelevancy reduction. Images generally have a continuous tone (continuous with respect to the x and y coordinates), with pixels being continuous in amplitude. Digitizing an image, (i.e., converting it from a continuous tone to a digital tone) is done by employing two processes, sampling and quantization. The process of taking samples from the x and y coordinates is known as sampling, while digitizing the amplitude value is known as quantization. Quantization helps to reduce the number of bits required to represent the intensity values. Thresholding, on the other hand, eliminates low intensity values that do not affect the quality of an image.

2.5.3 Entropy Coding

Once the data has been compressed, an extra compression can be obtained by coding the data. The basic idea behind entropy coding is to assign a short codeword for the most probable symbols or pixels (i.e., symbols or pixels that appear frequently in a context or image). Long code words are assigned for symbols and pixels with the least probability of occurrence. Some examples of entropy coding include Huffman coding, run-length coding, arithmetic coding and the Lempel-Ziv-Welch (LZW) coder.

2.5.3.1 Huffman Coding

Huffman Coding is one of the most popular techniques of data compression. In Huffman coding, each symbol in the information source is coded individually. Symbols may be either pixel intensities or the results of an intensity mapping operation.

The first step in Huffman coding is to create a series of source reductions by rearranging the symbols in a descending order according to their probability of occurrence. Then, the two least probable symbols are combined into a single symbol that replaces them in the next source reduction. This process is repeated until only two symbols are left [6]. This repetition generates a tree. Huffman code is then obtained through the tree labeling process [2]. The table below illustrates the first step in Huffman coding.

Original source		Source reduction			
Symbol	Probability	1	2	3	4
a_2	0.42	0.42	0.42	0.42	0.58
a_6	0.2	0.2	0.23	0.35	0.42
a_1	0.13	0.15	0.2	0.23	
a_4	0.1	0.13	0.15		
a_3	0.1	0.1			
a_5	0.05				

Table 2.2 Huffman source reductions

In the second step, a code is assigned for each reduced symbol. The coding procedure starts from the smallest source and goes back to the original source. The two smallest sources on the right side are assigned the binary code with the shortest length (i.e., length=1) that is equal to 0 or 1. For example, in Table 2.2, 0 binary code is assigned to 0.58, while 1 binary code is assigned to 0.42. Next, a binary code is assigned to symbols that are used to generate the reduced symbol with a probability of 0.58. Here, the same binary code 0 is assigned to these two symbols, and an arbitrary appended 0 and 1 are used to distinguish them from each other. This operation is repeated for each reduced source until the original source is reached. The following table illustrates the second step of Huffman coding.

Original source			Source reduction							
Symbol	Probability	Code	1		2		3		4	
a_2	0.42	1	0.42	1	0.42	1	0.42	1	0.58	0
a_6	0.2	000	0.2	000	0.23	01	0.35	00	0.42	1
a_1	0.13	001	0.15	001	0.2	000	0.23	01		
a_4	0.1	010	0.13	010	0.15	001				
a_3	0.1	0010	0.1	011						
a_5	0.05	0011								

Table 2.3 Huffman code

The code efficiency for Huffman coding can be calculated as follows:

$$\eta = \frac{H(r)}{L_{avg}} \tag{2.22}$$

where, $H(r)$ is the entropy of the source and L_{avg} is the average length of the code that can be calculated using equation (2.4) [30].

The entropy of a source represents the average amount of information obtained per symbol from the source. It can be calculated using the following equation;

$$H(r) = \sum_{i=0}^{I} P(r_i) log_2 \frac{1}{P(r_i)} \tag{2.23}$$

where, $P(r_i)$ represents the probability of occurrence of symbol r_i, and I is the number of symbols in the source. The more information associated with a source, the greater its entropy value.

The average length of the code in the above example can be calculated using equation (2.4) as follows:

$$L_{avg} = (0.42)(1) + (0.2)(3) + (0.13)(3) + (0.1)(3) + (0.1)(4) + (0.05)(4)$$

$L_{avg} = 2.31 bits/pixel$, where the entropy of the source is equal to $2.2532 bits/symbol$, and the code efficiency is equal to 0.9754 $pixel/symbol$.

In order to code the bits (010110110001) using Huffman code from Table 2.3, the bits stream is subdivided into number of code words. The first valid code word is '0101' which corresponds to a_4, while the second valid code word is '1,' which corresponds to a_2. This process is continued until the entire string is coded. The complete encoded message is $a_4 a_2 a_3 a_1$.

2.5.3.2 Run-length Compression

When an intensity value occurs more than once in an image, a run-length coding can be used to represent the identical intensities as *run-length pairs*. The *run* represents the repeated occurrence of an intensity value. The number of repetitions for a run is known as the length. In this scheme, the run of any length is represented using only two characters; one represents the intensity value, while the other represents the length (or the number of repetitions for that intensity value). Each pair represents the beginning of a new intensity value. An example of this type of data encoding is given in Section 2.3.

Chapter III

WAVELET TRANSFORM

3.1 Introduction

Transformations can be performed on signals and images to obtain extra information that is not available in the original domain. There are numerous types of transformation. Different applications utilize different kinds of transformations to focus on the specific properties of signals and images. Each transformation has its advantages and disadvantages. Examples of transformations include the Fourier transform, the short-time Fourier transform, the Hilbert transform, the Wigner transform and wavelet transform. Section 3.2 is employed to understand how generalized images transform and how different transformations are implemented on images. The kernels, or basis functions for a 2-D wavelet transform, are explained in Section 3.3. Section 3.3 also shows how the forward and inverse wavelet transform is employed on a 2-D array.

In a continuous case, a wavelet transform measures similarities between a given signal and a basis function using a correlation function. Mathematically, a correlation represents the inner product of two functions and can be expressed as follows:

$$< f(t), g(t) >= \int_a^b f(t).\, g^*(t) dt \tag{3.1}$$

Where, $g^*(t)$ represents the conjugate of $g(t)$

In the continuous wavelet transform, the transformed signal $CWT_x^{\psi}(\tau, s)$ is computed as the inner product of a signal $x(t)$ and the basis function $\psi_{\tau,s}(t)$. It is given by:

$$CWT_x^{\psi}(\tau, s) = \Psi_x^{\psi}(\tau, s) = \frac{1}{\sqrt{|s|}} \int x(t)\psi^*(t)dt \tag{3.2}$$

The basis function is called the wavelet function, or window, and is defined as:

$$\psi_{\tau,s} = \frac{1}{\sqrt{s}} \psi\left(\frac{t-\tau}{s}\right) \tag{3.3}$$

In order to obtain the wavelet transform of a signal, a window with an arbitrary size is employed to segment a signal and compute the wavelet transform for each segment. The transforming signal $\psi(t)$ in equation (3.2) is known as the 'mother wavelet'. The mother wavelet is classified as the prototype for generating other functions. A small wave is most typically used as a mother wavelet. The mother wavelet should be a semi-symmetric oscillated signal, compactly supported with a finite length and an area equal to one. It has two parameters tau τ and s. Tau represents the translation as the window shifts through a signal; translation refers to the current location of the window. The translation helps to present the time information of a signal in the transform domain. The scale represents the size of the window. Different window sizes are used to locate different frequencies within the signal.

The wavelet transform can be employed using any wavelet that satisfies the conditions explained in the above section. Based on these conditions, different families of wavelets have been defined. Generally, the wavelet families are subdivided into two groups: orthogonal and biorthogonal filters. Orthogonal filters preserve the energy contained within a signal, while biorthogonal filters do not preserve the energy contained within a signal [20].

The continuous wavelet transform represents the transform in a continuous case. In order to apply the wavelet transform to signals using machines, a solution must be found for its continuity. A discrete wavelet transform can be used to obtain the discrete wavelet representation of a discrete signal. In this case, convolution is employed to find the similarities between the input signal and digital filters at different frequency bands. Digital filters (such as high and low pass filters) are used to obtain signal coefficients at the different frequencies. Additionally, a downsampler can be used to down sample the number of samples (an upsampler is used for signal reconstruction). This operation is known as sub-band coding. Section 3.4 explains how to perform sub-band coding on 1-D and 2-D signals, respectively.

3.2 Generalized Image Transform

The default domain for images is the spatial domain (in which tasks are performed directly on pixels). Some tasks work faster or more efficiently in another domain. Therefore, it is more practical to transform an image to the desired transform domain, execute the task, and apply the inverse transform to place the image back in spatial domain. The general equation for transformation is given by;

$$T(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) r(x,y,u,v) \tag{3.4}$$

This equation is evaluated for $u = 0,1,2,\ldots,M-1$, and $v = 0,1,2,\ldots,N-1$, where $T(u,v)$ is the forwarded transform of an input image $f(x,y)$. M and N represent the number of rows and columns in $f(x,y)$ respectively, while x and y are the spatial variables. u and v are the transform variables, with $r(x,y,u,v)$ being the forward transform kernel.

Reconstruction of the original signal or image in 2-D is possible, as performing the inverse transform on $T(u,v)$ gives the recovered image $f(x,y)$ as follows;

$$\hat{f}(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} T(u,v) s(x,y,u,v) \tag{3.5}$$

Equation (3.5) evaluated for $x = 0,1,2,\ldots,M-1$, and $y = 0,1,2,\ldots,N-1$. $s(x,y,u,v)$ is the inverse transform kernel. If $\hat{f}(x,y) = f(x,y)$, then the transform is considered an invertible transformation.

Different kernels result in different kinds of transformations. If the kernel is equal to the infinite sum of periodic and complex exponential functions, then the transform is called the Fourier transform. The Fourier transform kernel is given by

$$r(x,y,u,v) = e^{-j2\pi(\frac{ux}{M}+\frac{vy}{N})} \tag{3.6}$$

Also, the inverse Fourier transform kernel is equal to

$$s(x,y,u,v) = \frac{1}{MN} e^{+j2\pi(\frac{ux}{M}+\frac{vy}{N})} \tag{3.7}$$

Substituting these two kernels in the general transform formulations, i.e., equations (3.4) and (3.5), gives:

$$T(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(\frac{ux}{M}+\frac{vy}{N})} \tag{3.8}$$

$$\hat{f}(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} T(u,v)e^{+j2\pi(\frac{ux}{M}+\frac{vy}{N})} \tag{3.9}$$

where equations (3.8) and (3.9) present the discrete Fourier transform and inverse discrete Fourier transform, respectively.

Another useful transformation is the Walsh-Hadamard transform (WHT). This transformation has identical kernels that are defined as follows:

$$r(x,y,u,v) = s(x,y,u,v) = \frac{1}{n}(-1)^{\sum_{i=0}^{m-1}\lfloor b_i(x)p_i(u)+b_i(y)p_i(v)\rfloor} \tag{3.10}$$

where $n = 2^m$. The summation of the exponent is carried out in module 2 arithmetic, while $b_k(z)$ is equal to the kth bit in the binary representation of z (from right to left). For example, if z=4 100 in binary, then $b_0(z) = 0$, $b_1(z) = 0$, and $b_2(z) = 1$. $p_i(u)$ is computed as follows:

$$
\begin{aligned}
p_0(u) &= b_{m-1}(u) \\
p_1(u) &= b_{m-1}(u) + b_{m-2}(u) \\
&\vdots \\
p_{m-1}(u) &= b_1(u) + b_0(u)
\end{aligned}
\tag{3.11}
$$

The above expressions are applied to $p_i(v)$. Again, both summations are performed in module 2 arithmetic.

One of the most important and frequently used transformations for performing image compression is discrete cosine transform DCT. DCT has equal kernels for the forward and inverse transform that are given by:

$$
\begin{aligned}
r(x, y, u, v) &= s(x, y, u, v) \\
&= \alpha(u)\alpha(v) \cos\left[\frac{(2x+1)u\pi}{2n}\right] \cos\left[\frac{(2y+1)v\pi}{2n}\right]
\end{aligned}
\tag{3.12}
$$

where

$$
\alpha(u) = \begin{cases} \sqrt{\frac{1}{n}} & for\ u = 0 \\ \sqrt{\frac{2}{n}} for\ u = 1,2, \dots, n-1 \end{cases}
\tag{3.13}
$$

Another important transformation for image processing is wavelet transform. In wavelet transform, a kernel of transformation consists of both scaling and wavelet function. The following section explicates the forward and inverse wavelet transform kernels and how the discrete wavelet transform is performed in 2-D.

3.3 2-D Wavelet Transform

A 2-D wavelet transform is performed on 2-D data such as images. It requires a two dimensional scaling function $\varphi(x,y)$, and three two dimensional wavelet functions $(\psi^H(x,y), \psi^V(x,y), \psi^D(x,y))$. The 2-D scaling and wavelet functions are obtained from 1-D functions such as the 1-D scaling function and 1-D wavelet function. The two dimensional separable scaling function is obtained as follows:

$$\varphi(x,y) = \varphi(x)\varphi(y) \tag{3.14}$$

where $\varphi(x)$ and $\varphi(y)$ are 1-D scaling functions. In addition, three functions of two-dimensional wavelets are obtained using the following equations:

$$\psi^H(x,y) = \psi(x)\varphi(y) \tag{3.15}$$

$$\psi^V(x,y) = \varphi(x)\psi(y) \tag{3.16}$$

$$\psi^D(x,y) = \psi(x)\psi(y) \tag{3.17}$$

The wavelet functions measure the intensity variation in different directions. $\psi^H(x,y)$ is a directional wavelet ψ^H that detects horizontal edges by measuring variations along columns, whereas $\psi^V(x,y)$ measures variations along rows, and detects vertical edges. $\psi^D(x,y)$ detects diagonal edges.

Thus, the two dimensional scaled and translated basis are defined as follows:

$$\varphi_{j,m,n}(x,y) = 2^{\frac{j}{2}}\varphi(2^j x - m, 2^j y - n) \tag{3.18}$$

$$\psi^i_{j,m,n}(x,y) = 2^{\frac{j}{2}}\psi^j\left(2^j x - m, 2^j y - n\right), i = \{H, V, D\} \tag{3.19}$$

Note that i is a subscript with values of $H, V, and\ D$ that identifies the directional wavelets.

The two-dimensional discrete wavelet transform is given by:

$$W_\varphi(j_o, m, n) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)\varphi_{jo,m,n}(x,y) \tag{3.20}$$

$$W^i_\psi(j, m, n) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)\psi^i_{j,m,n}(x,y), i = \{H, V, D\} \tag{3.21}$$

M and N represent the number of rows and columns of $f(x,y)$, while j_o is an arbitrary starting scale; normally $j_o = 0$. Equation (3.20) calculates the approximation of $f(x,y)$ at a scale of j_o , while the coefficients $W^i_\psi(j,m,n)$ in equation (3.21) add the horizontal, vertical, and diagonal details for the scale j (where $j > j_o$). Note that when $N = M = 2^J$, then $j = 0,1,2, \dots, J-1$ and $m = n = 0,1,2, \dots, 2^j - 1$ [6]. The inverse wavelet transform can be calculated using the following equation:

$$\begin{aligned} \hat{f}(x,y) = {} & \frac{1}{\sqrt{MN}} \sum_m \sum_n W_{\varphi,m,n}(x,y) \\ & + \frac{1}{\sqrt{MN}} \sum_{i=H,V,D} \sum_{j=j_o}^{\infty} \sum_m \sum_n \psi^i_\psi(j,m,n)\psi^i_j(x,y) \end{aligned} \tag{3.22}$$

Equations (3.20) and (3.21) are known as discrete wavelet transforms (DWT). Additional decomposition can be obtained by decomposing the approximation of $f(x,y)$ using the same equations [10-13]. The DWT is similar to sub-band coding [8], [15-16] in which a 1-D or 2-D signal is decomposed using digital filters and downsamplers. The

digital filter and sampler are known together as filter banks. The following section explains how DWT decomposes a two-dimensional signal.

3.4 The Sub-band Coding

In the discrete time, signal decomposition can be employed using filter banks. Signal decomposition consists of two operations, filtering and sub-sampling. Filtering is performed mathematically by convolving a signal with high and low pass filters as follows:

$$y_{high}[k] = \sum_{n} x[n].\, g[2k - n] \tag{3.23}$$

$$y_{low}[k] = \sum_{n} x[n].\, h[2k - n] \tag{3.24}$$

Where n is an integer. $y_{high}[k]$ represents the high pass version of the original signal $x[n]$. $y_{high}[k]$ is known as the detail coefficient of $x[n]$. $y_{low}[k]$ is the low pass version of $x[n]$ and is thus known as the approximation coefficient. A sub- sampling operation, i.e., down sampling, is used to reduce the number of samples for both detail and approximation coefficients. In other words, downsampling by the factor n decreases the number of samples in a signal by n. Usually, the DWT coefficients are down sampled on a dyadic grid, i.e., n=2. The following figure represents the general scheme for analyzing a signal using sub-band coders [18-19].

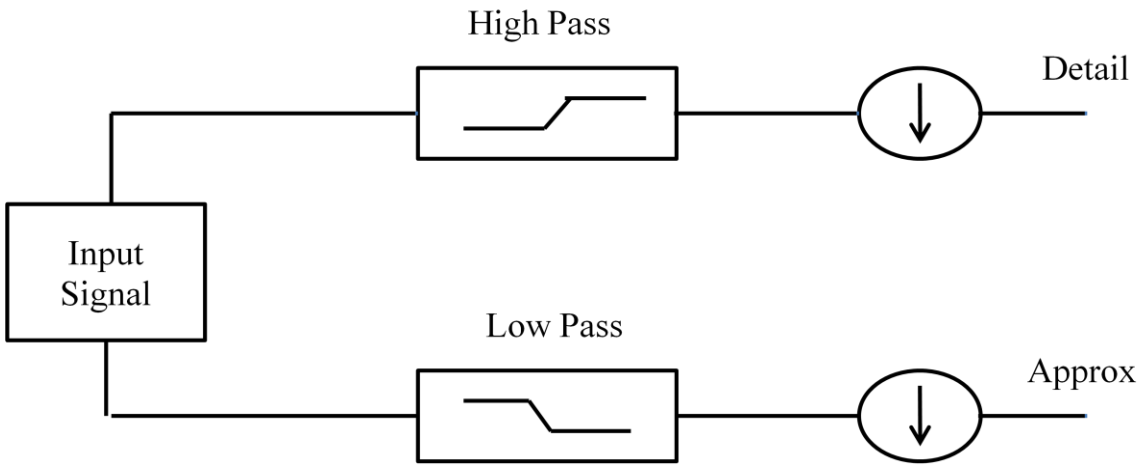


Figure 3.1 General scheme for analyzing a signal using sub-band coders

The signal can be further analyzed by passing the resultant approximation coefficients through the coder again; this yields another set of approximation and detail coefficients. The following figure illustrates the wavelet analysis tree.

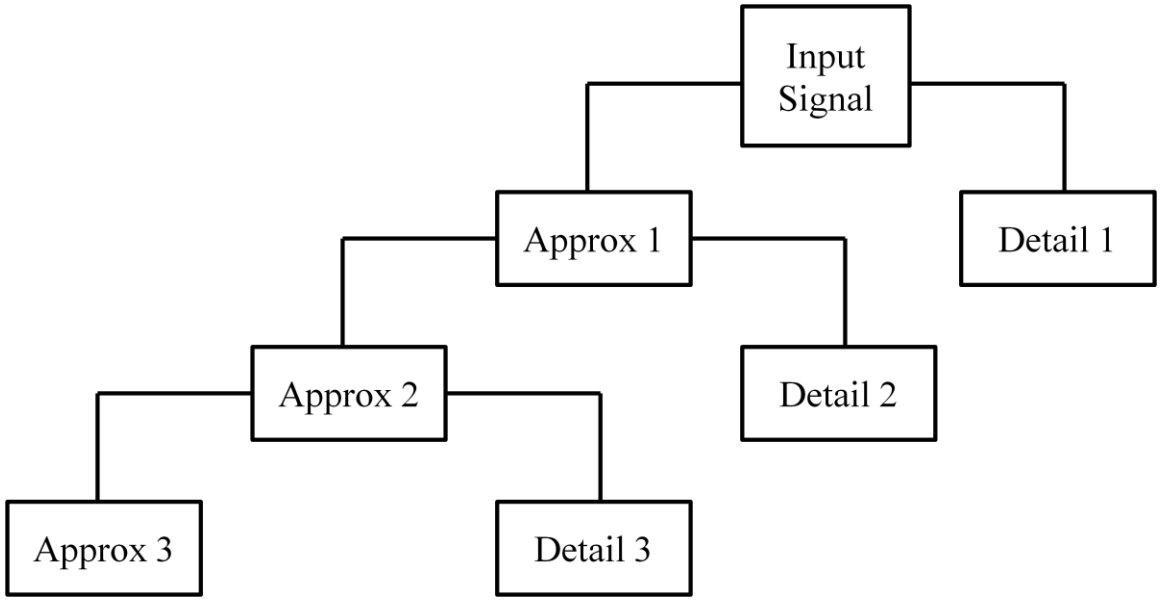


Figure 3.2 Wavelet signal analysis tree

Signal reconstruction is done in reverse order, meaning that the approximation and detail coefficients are first upsampled. The upsampled coefficients are then passed through high and low pass reconstruction filters; finally, the result is summed. The following figure represents the general scheme for reconstructing a signal using the approximation and detail coefficients.

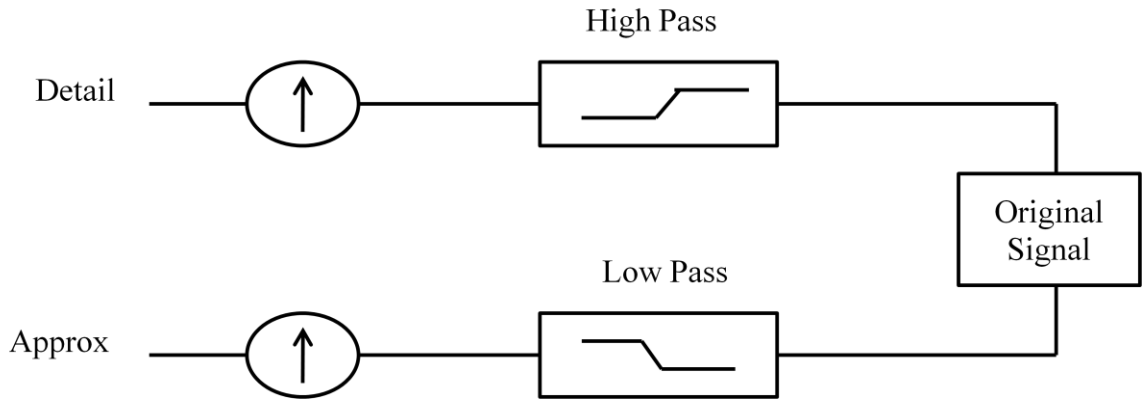


Figure 3.3 Signal synthesis using sub-band coders

An image is a 2-D signal that contains high and low frequencies. Image decomposition using filters can be employed to analyze or decompose image details into fine and coarse details. The separable two dimensional scaling and wavelet functions from equations (3.18) to (3.19) are used for this purpose, with the scaling function corresponding to a low pass filter and the wavelet function corresponding to a high pass filter [17]. The 2-D DWT decompositions are done in a similar approach to 1-D DWT. First the 1-D DWT are calculated for the rows of $f(x, y)$ and then for the resultant columns. Figure 3.4 and 3.5 illustrate the analysis and synthesis filter banks for the 2-D discrete wavelet transform [6].

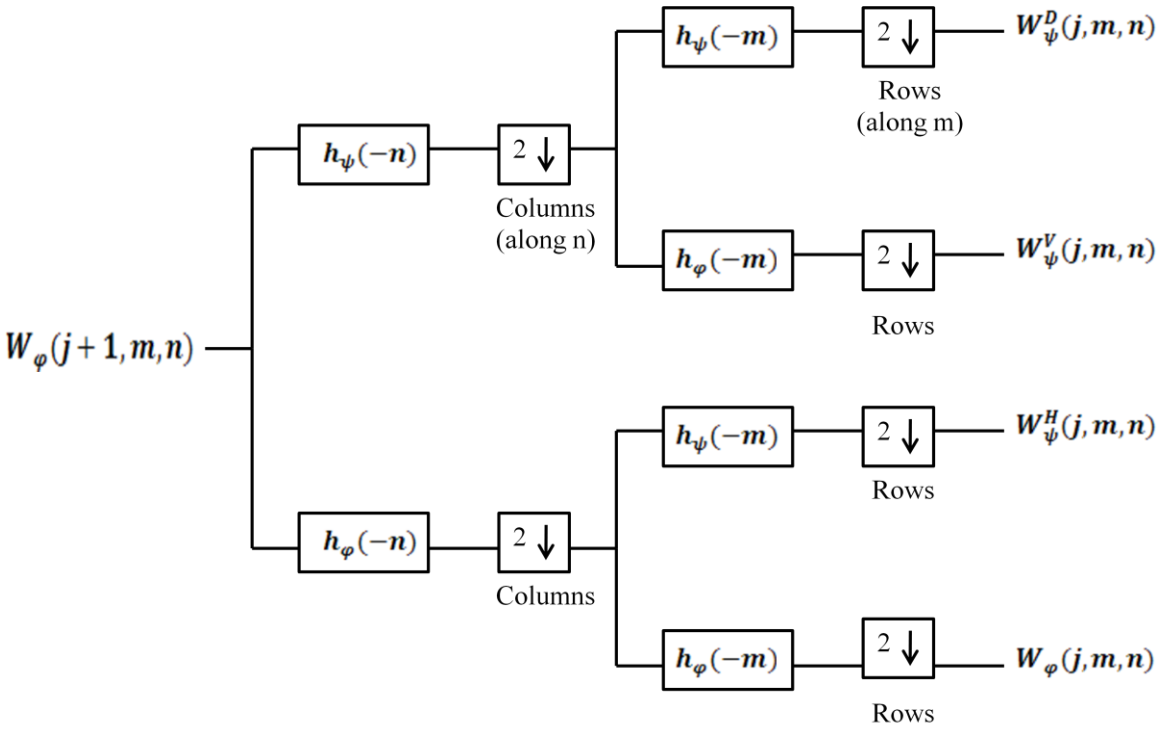


Figure 3.4 2-D wavelet transform using the analysis filter bank

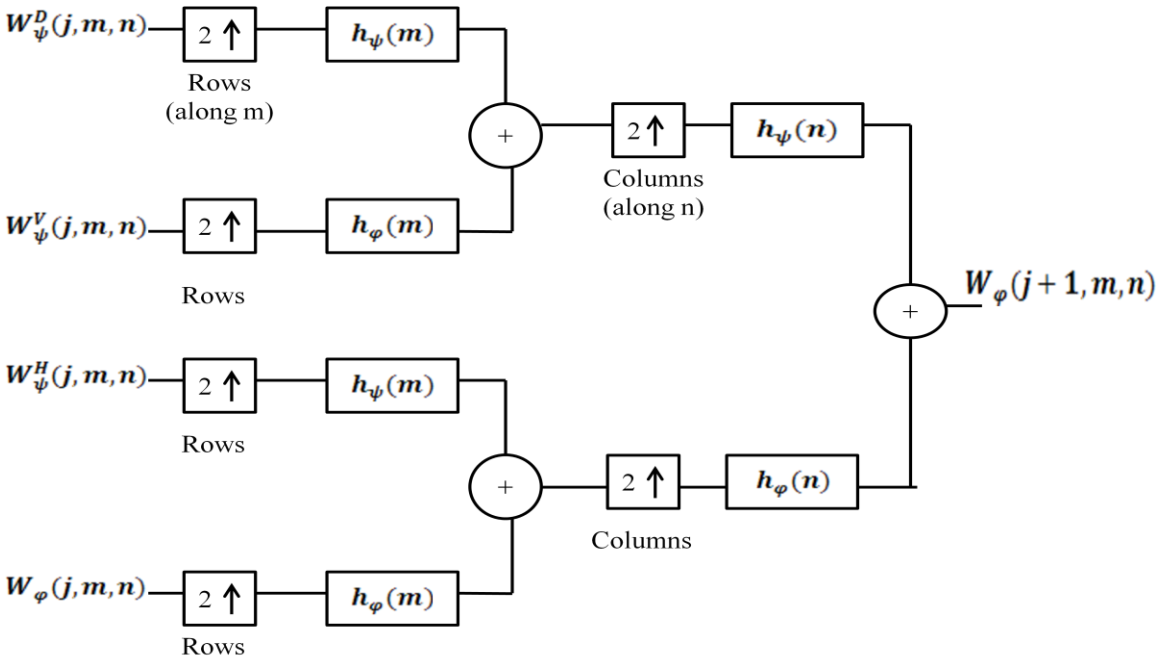


Figure 3.5 2-D inverse wavelet transform using the synthesis filter bank

Figure 3.6 demonstrates one level of decomposition of an image [6].

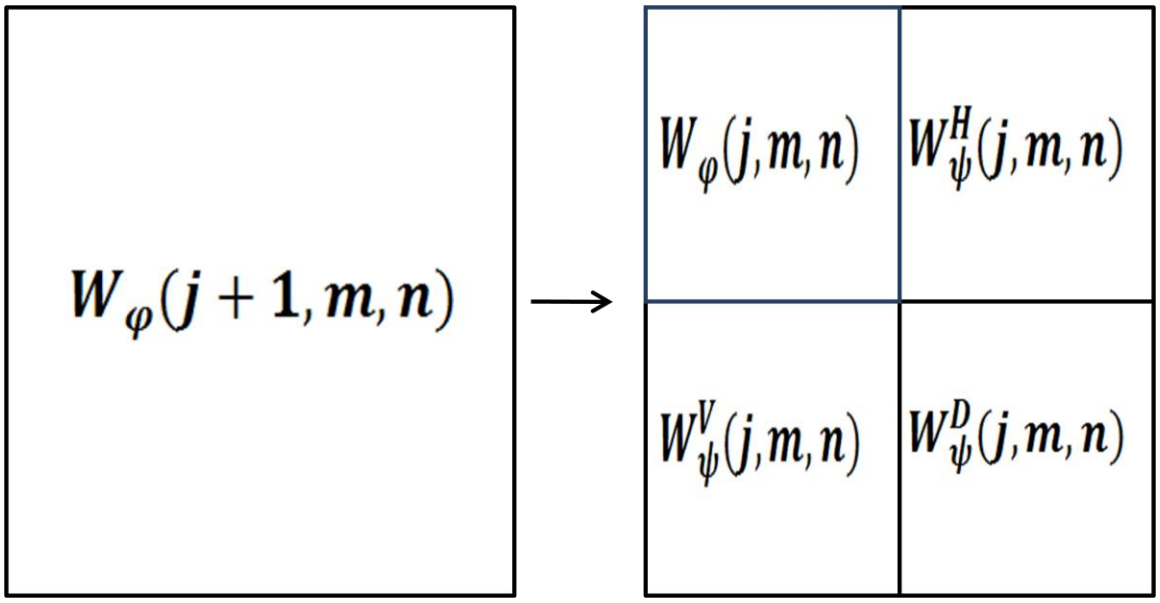


Figure 3.6 One level of decomposition of an image

In Figure 3.6, $W_\varphi(j+1,m,n)$ represents the original image. $W_\varphi(j,m,n)$ is the approximation sub-image that represents smooth variations. It is obtained from low passing the original image and decimating the results by factor 2. $W_\psi^H(m,n)$, $W_\psi^V(m,n)$ and $W_\psi^D(m,n)$ are the detail images that result from high passing the original image and decimating the results by factor 2. They give the horizontal, vertical, and diagonal edges of the approximation image. Additional decomposition can be obtained by decomposing the approximation image $W_\varphi(j,m,n)$ of level j. Figure 3.7 represents two levels of decomposition of an image.

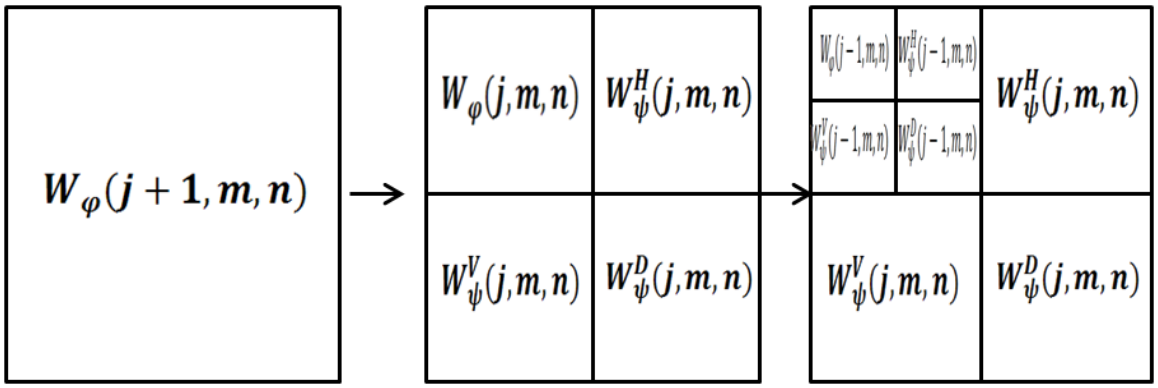


Figure 3.7 Two level of decomposition of an image

Chapter IV

LOSSY IMAGE COMPRESSION USING WAVELET TRANSFORM

4.1 Introduction

An algorithm of lossy digital image compression using wavelet transform is presented in this thesis. The following figure illustrates the general scheme for our algorithm.

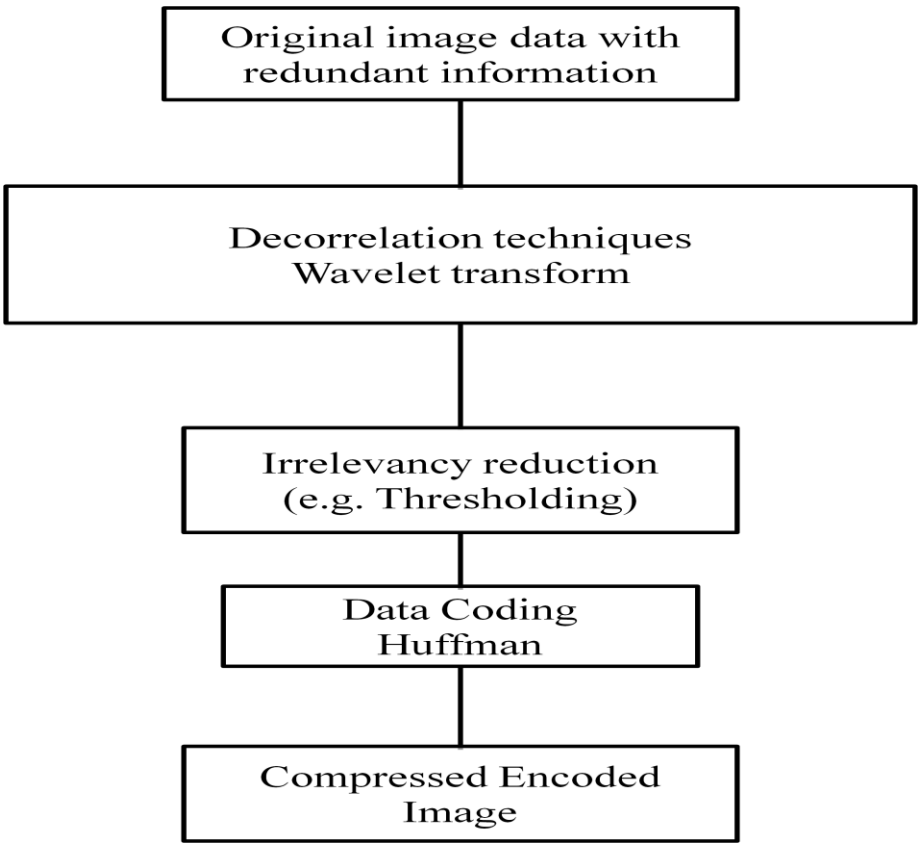


Figure 4.1 Lossy image compression using wavelet transform

In this algorithm, wavelet transform was selected as the desired method for data decorrelation. Section 4.2 explains the decorrelation technique using wavelet transform.

Since wavelet transform helps obtain the edges contained within images, a new approach for irrelevancy reduction was proposed in this algorithm. Additionally, Roberts and Sobel masks helped calculate diagonal edges using horizontal and vertical edges. We were able to apply these masks on the horizontal and vertical edges of the wavelet image and thereby retrieve the diagonal edges. As a result, the diagonal edges can be set to zero because they can be retrieved later in the reconstruction process using the horizontal and vertical edges. This approach saves a lot of memory, since ¼ of an image is set to zero. Section 4.3 explains the irrelevancy reduction procedure for this algorithm, while Section 4.4 illustrates the procedure followed by this algorithm in order to retrieve the diagonal edges. Since our algorithm results in a loss of information, an evaluation of the quality of the reconstructed image is required. Section 4.5 explains the different approaches to determining the quality of an image. The experimental results and conclusions are given in Sections 4.6 and 4.7 respectively.

4.2 The Wavelet Transform

In our algorithm, wavelet transform is used as a decorrelation tool. Usually, natural images are smooth (with the fine details being represented as sharp edges). In the chapter 3 analysis of an image, using 2-D DWT is explained. Figure 4.2 represents an original image of the size MxN. Transforming the image in Figure 4.2 (using one level of DWT) results in four sub-images in Figure 4.3.

Figure 4.2 An image in spatial domain

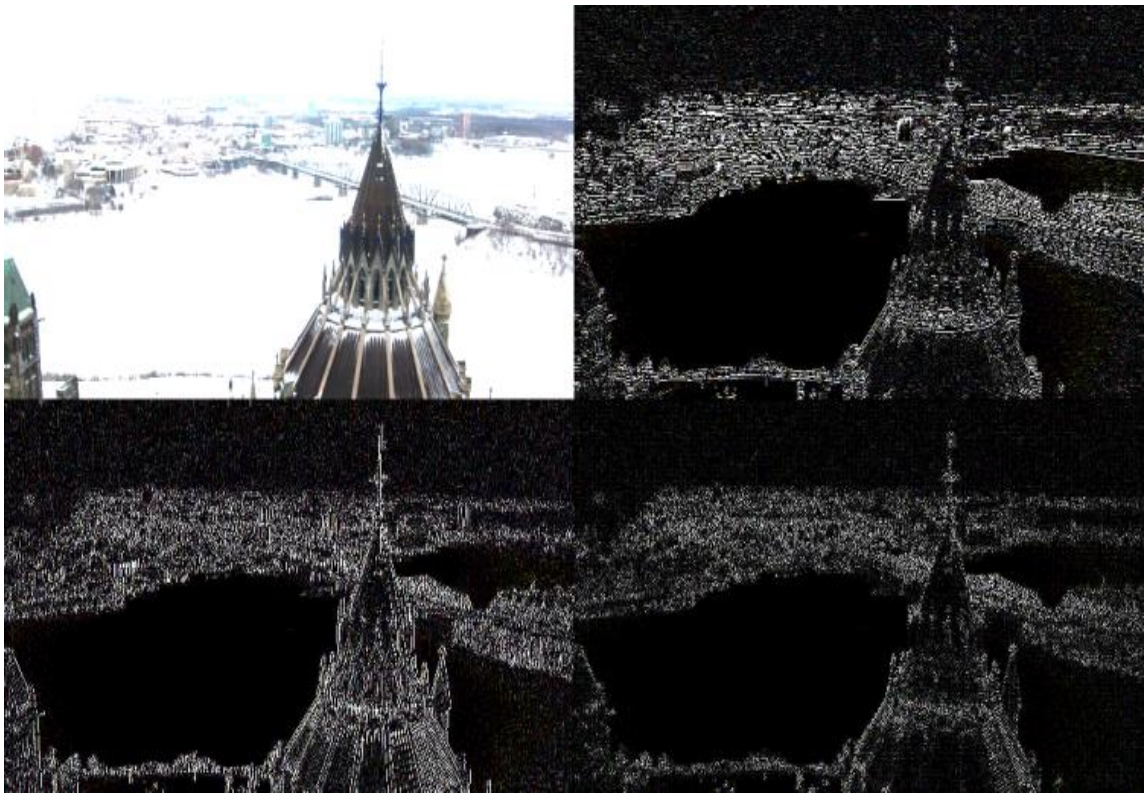


Figure 4.3 One level of decomposition using the wavelet transform

In Figure 4.3, the sub-image in the upper left corner represents the approximation image, with a size equal to $M/2xN/2$. The approximation image does not contain any sharp edges for this level of decomposition. The other three images, in Figure 4.3, contain the horizontal, vertical, and diagonal edges, respectively. Full reconstruction of the original image can be obtained by combining the approximation sub-image with the detail sub-images.

The analysis of the original image into sub-images helps to decorrelate data. Analyzing an image using wavelet transform gives an approximation sub-image and three detail sub-images. The approximation sub-image contains the fine details of the original image, while the detail sub-images consist of sharp details and black regions. The black

regions represent pixels with zero intensity value or small values.

As a result, less size is required to represent the original image. Since our algorithm produces an irreversible image, extra size reduction can be obtained by employing one of the irrelevancy reduction techniques to quantize the data contained within each pixel. Thresholding can be employed to eliminate small intensity values that are usually imperceptible by the HVS. The following section explains the quantization and thresholding processes that are employed in this algorithm.

4.3 Irrelevancy Reduction

Once the approximation and detailed images are obtained, the quantization step is performed in order to quantize the data as a part of the irrelevancy reduction process. Since wavelet transform produces numbers with fractions in addition to negative values, quantization is performed so as to round all values to the nearest integer. Thresholding is employed to eliminate intensity values that are less than zero and positive intensity values that are usually ignored by the HVS.

Generally, thresholding is employed to partition an image into regions using a threshold value. The threshold value is selected based on the histogram of an image. Figure 4.4, represents a gray image and the corresponding histogram.

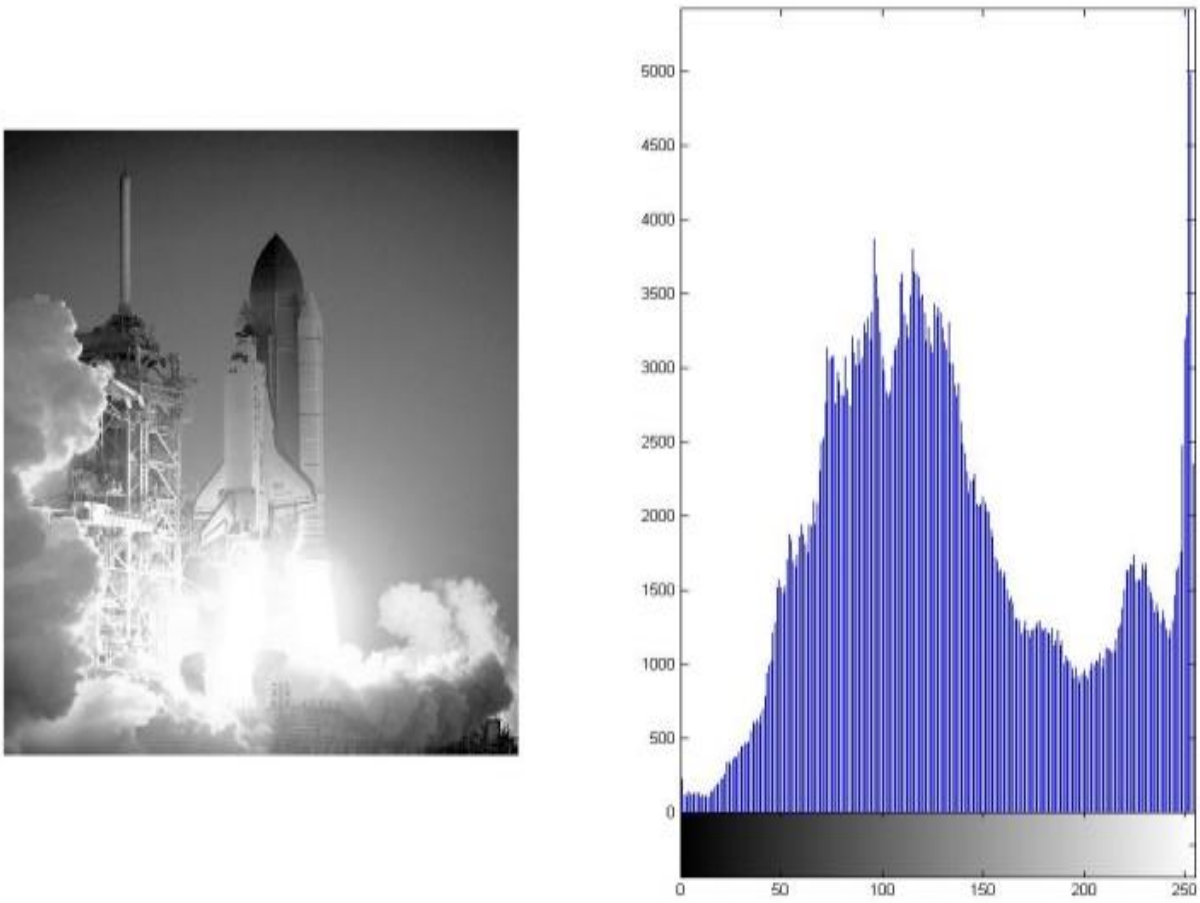


Figure 4.4 A gray image and its histogram

The image in Figure 4.4 is an 8 BPP image, i.e., 8 bit ($2^8 = 256$ gray levels). Thus, this image has intensity values $f(x, y)$ that vary from 0-255. Intensities with zero value represent black dots, while intensities with value= 255 represent white dots. Thresholding can be used to separate objects from the background. A threshold value T is used for this purpose, thus any (x, y) in the image with an intensity value of $f(x, y) > T$ is called an object point. Any intensity value $f(x, y)$< T is called a background point. The segmented image $g(x, y)$ is given by:

$$g(x,y) = \begin{cases} 1 \; if \; f(x,y) > T \\ 0 \; if \; f(x,y) \leq T \end{cases} \tag{4.1}$$

T can be a constant or variable value. For a constant value, the same value is used for the entire image and results in global thresholding. When T varies over an image, it is called variable thresholding [6].

In the field of image compression, thresholding can be employed to filter out pixels with intensity values that do not affect the quality of an image. In our algorithm, iterative global thresholding is the selected thresholding method, meaning that one single value is determined based on the histogram of an image. Thus, for different images, a different threshold value is employed. The following algorithm represents the approach implemented by our algorithm for selecting a global threshold value [6]:

1. Select an initial threshold value, T.
2. Group the intensity values within an image into two groups, G_1 and G_2 where $G_1 > T$ and $G_2 \leq T$.
3. Compute the mean intensity value for each group, m_1 and m_2.
4. Compute a new threshold value as follows:

$$T = \frac{1}{2}(m_1 + m_2) \tag{4.2}$$

5. Repeat steps 2-4 until the difference between values of T in successive iterations is smaller than the predefined parameter ΔT.

Thresholding is not applied to the approximation image, since it constitutes the base of an image. It is only performed on the detailed images, particularly on the horizontal and vertical edges. In this algorithm, the diagonal edges are set to zero. This saves extra memory and increases the compression rate. In the image reconstruction process, the

diagonal edges are retrieved using a 2-D mask. The following section explains the approach that is implemented by our algorithm for retrieving the diagonal edges.

4.4 Diagonal Edges Retrieval

Different kinds of masks such as the Roberts-cross gradient, Prewitt, and Sobel operators can be used to retrieve diagonal edges. Generally, these masks compute variations along different directions, (horizontal, vertical, and diagonal) using partial derivatives $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$. The partial derivatives measure the edge strength and direction at every pixel location. The digital approximation g_x and g_y for digital images can be computed as follows:

$$g_x = \frac{\partial f(x,y)}{\partial x} = f(x+1,y) - f(x,y) \tag{4.3}$$

$$g_y = \frac{\partial f(x,y)}{\partial y} = f(x,y+1) - f(x,y) \tag{4.4}$$

Computing variation along different directions is done by filtering an image with a mask (i.e., a Roberts-cross gradient, Prewitt, or Sobel mask). Figure 4.5 represents a 2x2 Roberts mask that is used to detect edges.

-1	**0**
0	**1**

0	**-1**
1	**0**

Figure 4.5 A Roberts mask

The mask in Figure 4.5 is considered to be one of the simplest masks. It is not very efficient in detecting edges compared to 3x3 masks that consider symmetric around the center point. One of the simplest 3x3 masks is the Prewitt mask. Figure 4.6 represents masks that are used to detect horizontal and vertical edges, respectively. Equations (4.5) and (4.6) compute the digital approximations of the partial derivatives using the same mask.

Z_1	Z_2	Z_3
Z_4	Z_5	Z_6
Z_7	Z_8	Z_9

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

Figure 4.6 Prewitt mask

$$g_x = (z_7 + 2z_8 + z_7) - (z_1 + z_2 + z_3) \quad (4.5)$$

$$g_y = (z_3 + 2z_6 + z_9) - (z_1 + z_4 + z_7) \quad (4.6)$$

Calculating the digital approximation using Prewitt operators would give a more accurate approximation than Roberts's operators. Another 3x3 mask is the Sobel mask. The Sobel mask is considered to be a good mask because it uses a 2 in the center of the mask that provides image smoothing. Figure 4.7 illustrates a 3x3 Sobel mask that is used to obtain the gradient at point z_5 [6].

-1	**-2**	**-1**
0	**0**	**0**
1	**2**	**1**

-1	**0**	**1**
-2	**0**	**2**
-1	**0**	**1**

Figure 4.7 Sobel masks for detecting horizontal and vertical edges

$$g_x = (z_7 + 2z_8 + z_7) - (z_1 + 2z_2 + z_3) \tag{4.7}$$

$$g_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \tag{4.8}$$

Masks in Figure 4.7 are used to detect the horizontal and vertical edges of the image in Figure 4.8 [29].

Figure 4.8 Image.jpg

Figure 4.9a Horizontal edges for the image in Figure 4.8
Figure 4.9b Vertical edges for the image in Figure 4.8

Figure 4.9a and 4.9b represent the horizontal and vertical edges for the image in Figure 4.8. Figure 4.10 illustrates the complete edges for the image in Figure 4.8. The complete edges are obtained from combining the edges in Figure 4.9a and 4.9b.

Figure 4.10 Edges for the image in Figure 4.8

Masks in Figure 4.7 can also be reshaped to detect diagonal edges. Figure 4.11 illustrates the reshaped Sobel filter mask [6].

0	**1**	**2**
-1	**0**	**1**
-2	**-1**	**0**

-2	**-1**	**0**
-1	**0**	**1**
0	**1**	**2**

Figure 4.11 Sobel masks for detecting diagonal edges

One of the two masks in Figure 4.11 can be employed for the purpose of diagonal edge detection. Figure 4.12 represents an image, while Figure 4.13 represents the corresponding diagonal edges for the same image when (the first mask on Figure 4.11 is applied on it) [29].

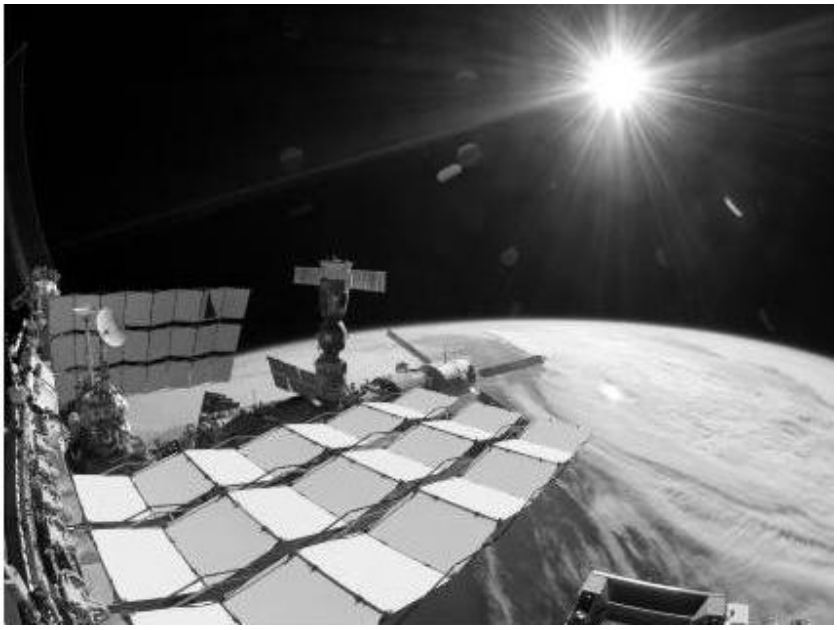


Figure 4.12 An image

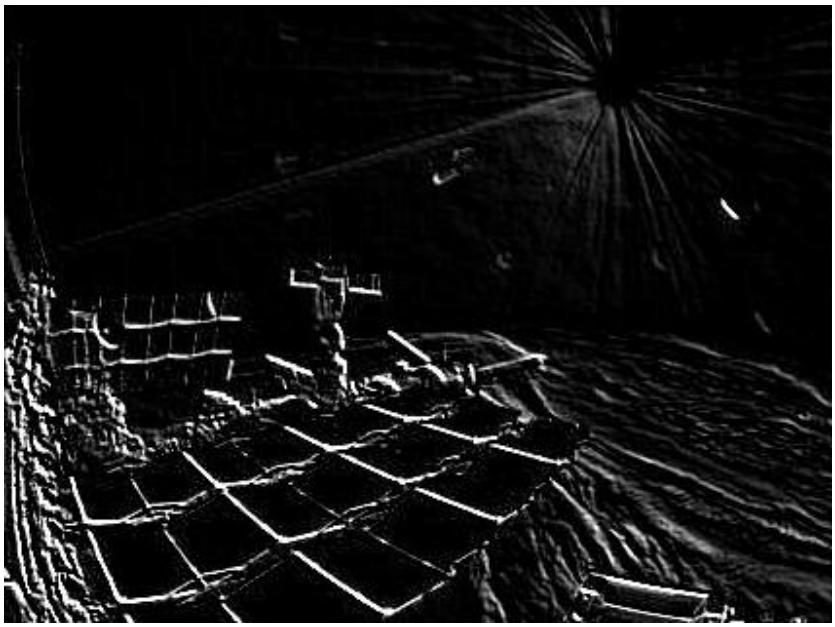


Figure 4.13 Diagonal edges for the image in Figure 4.12

Another approach to obtaining the diagonal edges is through the use of partial derivatives of g_x and g_y. The following equation is used to combine the gradient magnitude g_x and g_y.

$$M(x,y) = \sqrt{g_x^2 + g_y^2} \tag{4.9}$$

$M(x,y)$ represents the diagonal edges. Computational requirements for (4.9) can be reduced by using the following approximation:

$$M(x,y) \approx |g_x| + |g_y| \tag{4.10}$$

When wavelet transform is applied to an image, an image becomes classified into the corresponding horizontal, vertical, and diagonal edges. Figure 4.3 shows the edges that are obtained from applying wavelet transform to the image in Figure 4.2. Setting the diagonal edges to zero saves M/2xN/2 of image size. When an image is reconstructed, the diagonal edges need to be retrieved. One way to obtain the diagonal edges is by applying

the masks in Figure 4.11 to both sub-images (Figure 4.3) with horizontal and vertical edges. Doing so helps one to obtain the partial derivatives, while combining the results gives one the diagonal edges. Figure 4.14 represents the retrieved diagonal edges for the image in Figure 4.3, the horizontal and vertical edges from Figure 4.3, and the mask from Figure 4.11. In our algorithm, the diagonal edges are retrieved using equation (4.9) for better accuracy.

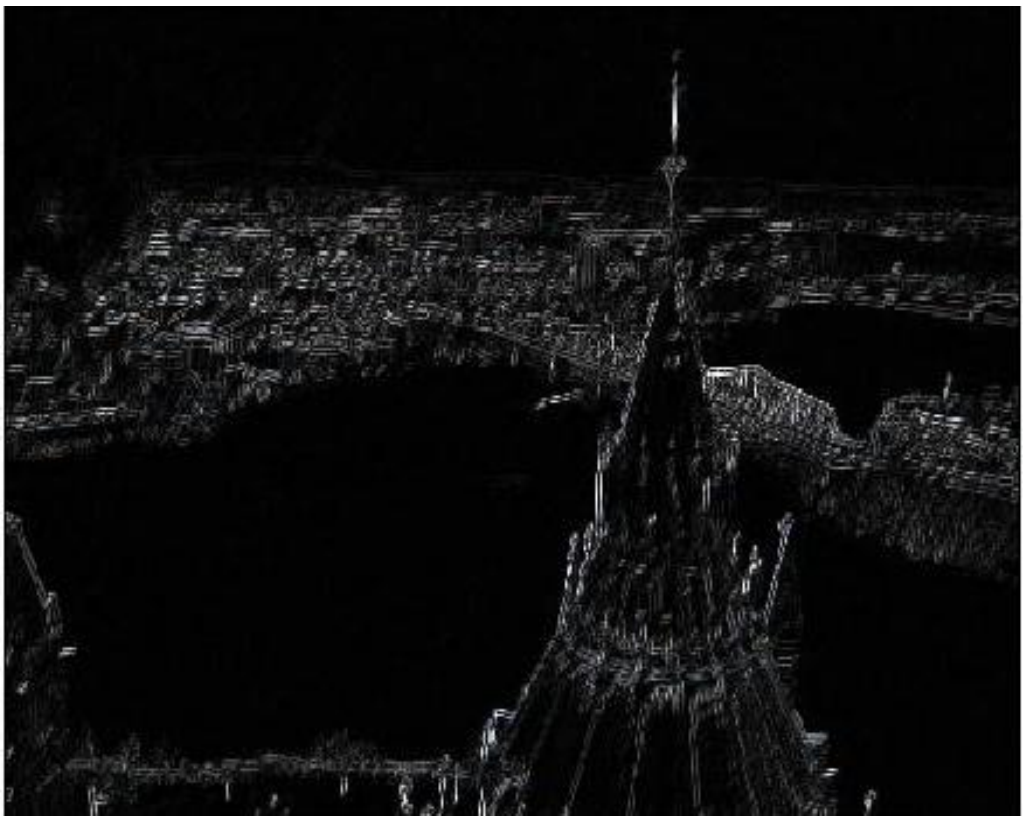


Figure 4.14 Retrieved diagonal edges

4.5 Fidelity Criteria

In lossy image compression, part of the information is removed. In this case, an assessment for nature of loss is required. Two methods can be used for the assessment: objective and subjective fidelity criteria.

In objective fidelity criteria, a mathematical function can be employed to determine the amount of information that is lost. The root mean square error (rms) between the two images (uncompressed and compressed) can be used for this purpose. rms can be computed as follows:

$$e(x,y) = \hat{f}(x,y) - f(x,y) \tag{4.11}$$

where $f(x,y)$ represents the original image and $\hat{f}(x,y)$ represents the compressed one. The total error between the two images is:

$$\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}[\hat{f}(x,y) - f(x,y)] \tag{4.12}$$

The root-mean-square error is equal to:

$$e_{rms} = [\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\hat{f}(x,y) - f(x,y)\right]^2]^{\frac{1}{2}} \tag{4.13}$$

If the result of equation (4.12) is equal to zero, then the error between the two images is equal to zero and they are the same. The rms value of the signal to noise ratio can be obtained as follows:

$$SNR_{rms} = \frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\hat{f}(x,y)^2}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\hat{f}(x,y) - f(x,y)\right]^2} \tag{4.14}$$

SNR_{rms} values greater than 40 indicate good image quality. Compressed images with a degraded quality have SNR_{rms} values less than 40.

In subjective fidelity, image quality is examined by humans. In this case, evaluation is done by presenting a compressed image to a set of viewers. The viewers usually compare the original image to the compressed image, with their evaluation being based on an absolute rating scale (i.e., much worse, worse, slightly worse, the same, slightly better, better, and much better).

4.6 Experimental results and analysis

The algorithm is applied to both grayscale and truecolor images. Figure 4.15 represents an uncompressed image with tif extension (tif stands for Tagged Image File Format). Chart 1 illustrates the size required to save the image using different file extensions or standards. The x-axis represents the compression standards, while the y-axis represents the size required to save the image using each standard in KB.

Figure 4.15 image.tif with no compression

Chart 4.1 The size required to save an image using different standards

Three standards are used for comparative purposes: BMP, JPEG2000, and JPEG. BMP is a Windows bitmap that represents an image with no compression or coding. JPEG2000 stands for Joint Photographs Experts Group 2000 and represents an image with a high quality and moderate compression rate. It uses the wavelet transform as its decorrelation technique. JPEG stands for Joint Photographs Experts Group and represents an image with visually lossless. JPEG provides a high compression rate and uses the discrete cosine transform as its decorrelation technique. JPEG and JPEG2000 are the latest ISO/ITU standards. They are used to compress images with a continuous-tone [21-26]. Mix JPEG is the standard used to compress images with our algorithm and to encode images using the JPEG standard. Figure 4.9 illustrates an image that is compressed using

each of the four standards. Note that the compression ratio is based on calculating the ratio between the compressed image and the uncompressed one.

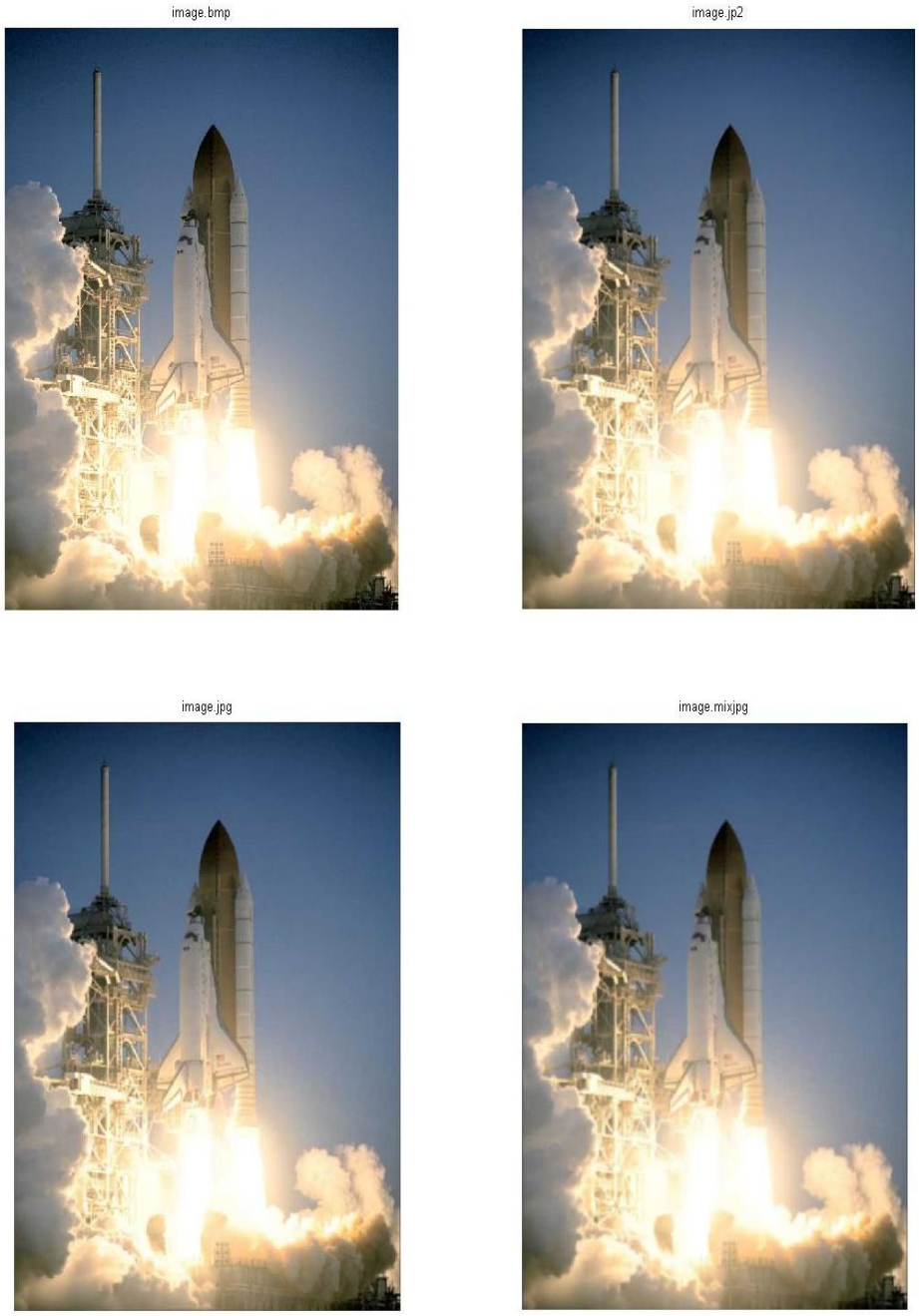


Figure 4.16 A colored image saved using bmp, jp2, jpg, and mix jpg extensions

Figure 4.17 presents the same image as a grayscale image that is compressed using the same four standards.

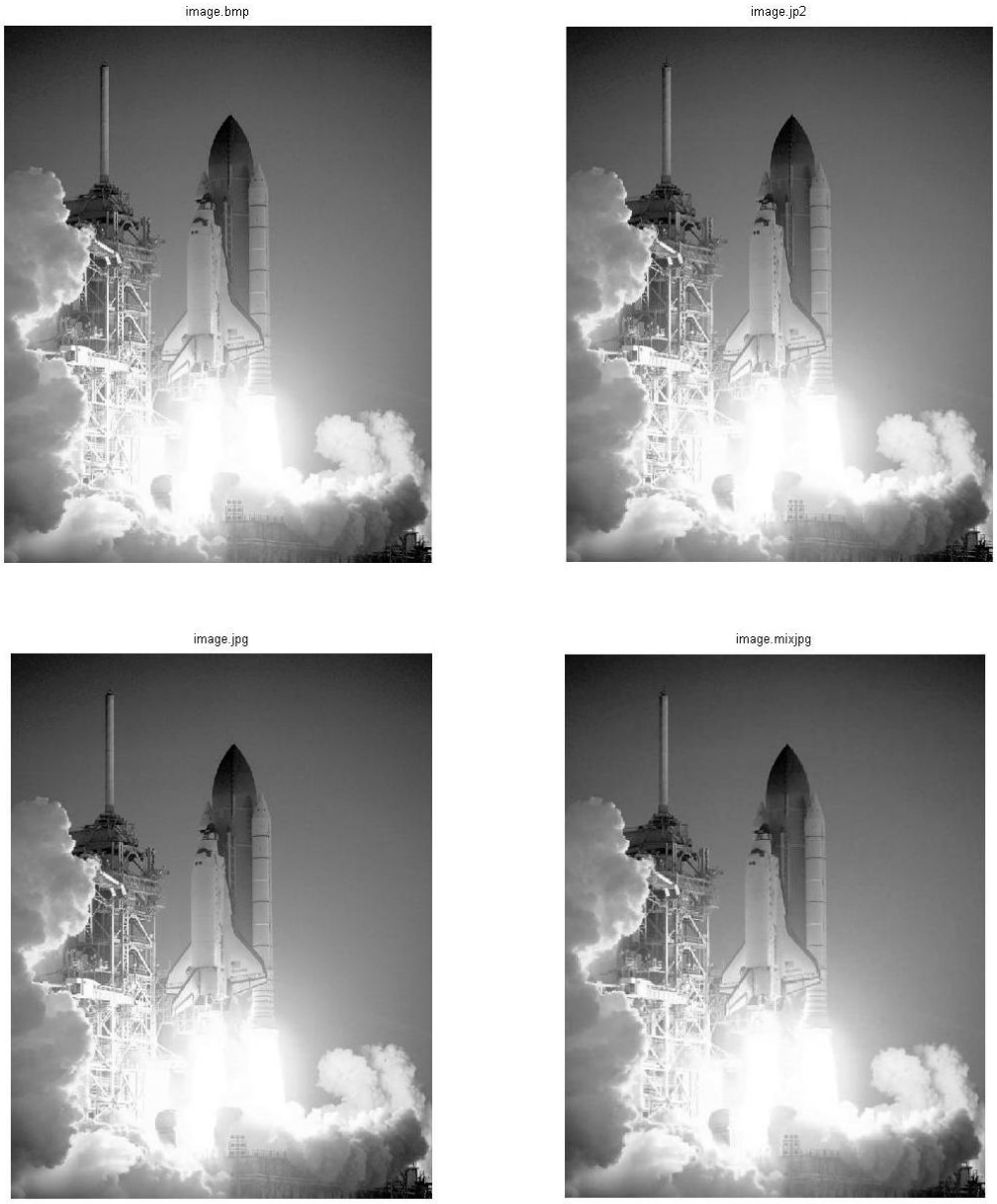


Figure 4.17 A grayscale image compressed with bmp, jp2, jpg, and mix jpg

While all the images in Figure 4.16 and 4.17 are the same, it is obvious that image.bmp has more details, since it uses 24 bits and 8 bits to represent colored and grayscale images. Chart 4.2 and 4.3 illustrate the compression ratio for images in Figure 4.16 and 4.17 respectively. For colored and grayscale images we were able to obtain the highest compression ratio compare to JEPG and JPEG2000.

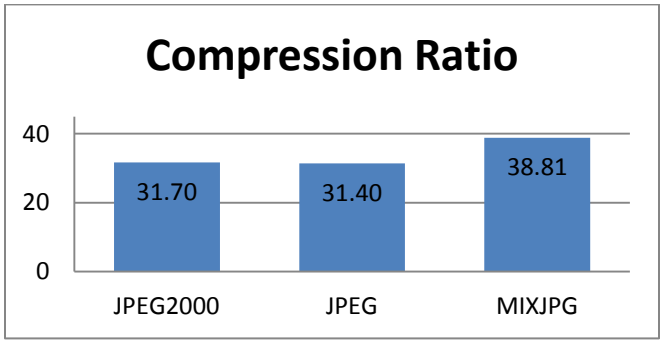


Chart 4.2 Compression ratio for a compressed colored image

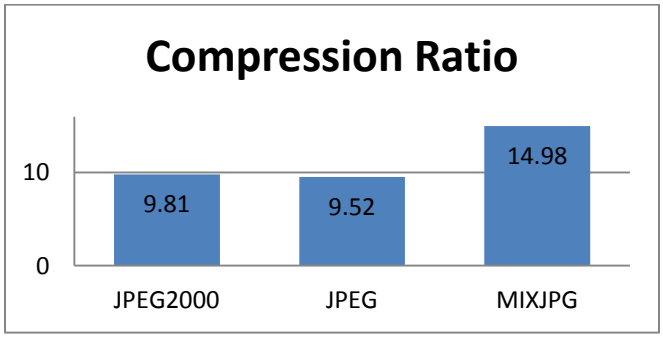


Chart 4.3 Compression ratio for a compressed grayscale image

The mean square error e_{rms} and signal-to-noise ratio (SNR) are used to evaluated the quality of the compressed images. Table 4.1 and 4.2 show e_{rms} and SNR_{rms} values for images in Figure 4.16 and 4.17 respectively. For colored images, this algorithm provides better quality compare to JPEG but slight less than the quality for JEPG2000. This is due to quantizing horizontal and diagonal edges. Eliminating the quantization part would slight increase the image size but provides better quality as it is shown in Table 4.1. For grayscale images, the quality of our image are still better compare to JEPG but less than that of JPEG2000. In both cases we were able to obtain better visual quality compare to JPEG.

	e_{rms}	SNR_{rms}	Size(KB)
BMP	-	-	1361
JPEG	1.96	45.21	42
JPEG 2000	0.92	48.5	42.3
MIX JPEG (quantization)	1.57	46.17	34.3
MIXJPEG (no quantization)	0.8	49.12	35.1

Table 4.1 Objective quality for images in Figure 4.16

	e_{rms}	SNR_{rms}	Size (KB)
BMP	-	-	457
JPEG	4.33	41.77	48
JPEG 2000	1.05	47.91	46.6
MIX JPEG (quantization)	5.16	41	30.5
MIXJPEG (no quantization)	4.27	41.81	31.7

Table 4.2 Objective quality measurement for images in Figure 4.17

CHAPTER V

CONCLUSIONS

In this survey, an algorithm for lossy image compression using wavelet transform was proposed. First, the wavelet transformation was selected as the desired transformation. Then, the method of setting the diagonal edges to zero was applied in order to increase the compression rate. Thresholding was performed on horizontal and vertical edges to increase the compression rate. For the image compression process, we retrieved the diagonal edges from the vertical and horizontal edges. This algorithm provided a higher compression rate compared to other standards such as JPEG2000 and JPEG. These standards provided a compression ratio that is equal to 31.7% and 31.4% for colored images and 9.81% and 9.52% for grayscale images. In contrast, our standard provided a compression ratio that is equal to 38.81% and 38.81% for the same colored and grayscale images.

REFERENCES

[1] R. Adhami, P.M. Meenen, III, D. Hite "Fundamental Concepts in Electrical and Computer Engineering with Practical Design Problems", 2nd edition, 2007.

[2] Asraf, R.; Akbar, M.; Jafri, N.; ,"Statistical Analysis of Difference image for Absolutely Lossless Compression of Medical Images," *Engineering in Medicine and Biology Society, 2006. EMBS '06. 28th Annual International Conference of the IEEE* , vol., no., pp.4767-4770, Aug. 30 2006-Sept. 3 2006.

[3] Ming Yang; Bourbakis, N.; , "An overview of lossless digital image compression techniques," *Circuits and Systems, 2005. 48th Midwest Symposium on* , vol., no., pp. 1099- 1102 Vol. 2, 7-10 Aug. 2005.

[4] Tzong-Jer Chen; Keh-Shih Chuang; , "A pseudo lossless image compression method," *Image and Signal Processing (CISP), 2010 3rd International Congress on* , vol.2, no., pp.610-615, 16-18 Oct. 2010.

[5] C.Alexopoulos and N.Bourbakis, An algebraic modeling of the SCAN language, Pattern Recognition Journal, to appear (also PhD thesis Univ. of P a m 1989).

[6] R. C. Gonzalez, R. E. Woods, "Digital Image Processing", 3rd edition, 2008.

[7] Zschunke, W.; , "DPCM Picture Coding with Adaptive Prediction," *Communications, IEEE Transactions on* , vol.25, no.11, pp. 1295- 1302, Nov 1977.

[8] Strang, G. and Nguyen, T. Wavelets and Filter Banks, Wellesley-Cambridge Press, Wellesley, MA, 1996, http://www-math.mit.edu/~gs/books/wfb.html.

[9] Berg, A.P.; Mikhael, W.B.; , "A survey of mixed transform techniques for speech and image coding ," *Circuits and Systems, 1999. ISCAS '99. Proceedings of the 1999 IEEE International Symposium on* , vol.4, no., pp.106-109 vol.4, Jul 1999.

[10] Mallat, S.G.; , "A theory for multiresolution signal decomposition: the wavelet representation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* , vol.11, no.7, pp.674-693, Jul 1989.

[11] Volkmer, H.; , "On the regularity of wavelets," *Information Theory, IEEE Transactions on* , vol.38, no.2, pp.872-876, Mar 1992.

[12] Hua Li; Yiming Zhu; , "Lossless Image Compression Based on DPCM-IWPT," *Computing, Communication, Control, and Management, 2008. CCCM '08. ISECS International Colloquium on* , vol.1, no., pp.157-160, 3-4 Aug. 2008.

[13] S. Hamidi and R. Adhami, Wavelet Transform for Image Data Compression, Proc. 25th SSST, 1993.

[14] Majid Rabbani and Paul Jones, "Digital Image Compression Techniques", SPIE Press, 1991.

[15] N. J. Fliege, "Multirate Digital Signal Processing: Multirate Systems – Filter Banks – Wavelets", Wiley &sons, Jan., 2000.

[16] K. P. Soman & K.I.Ramachandran "Insight into Wavelets from theory to practice", Prentice Hall India, New Delhi, 2002.

[17] Li-jen Du; Hoppel, K.; , "Multiresolution Analysis of Sar Images Using Wavelet Transform Representation," *Geoscience and Remote Sensing Symposium, 1991. IGARSS '91. Remote Sensing: Global Monitoring for Earth Management., International* , vol.3, no., pp.1417-1419, 3-6 Jun 1991.

[18] C. S. Burrus, R. A. Gopinath, and H. Guo, Introduction to Wavelets and Wavelet Transforms: A Primer, Prentice Hall, New Jersey, 1998.

[19] R. Polikar, "The Engineer's Ultimate Guide to Wavelet Analysis." http://www.public.iastate.edu/~rpolikar/WAVELETS/waveletindex.html Last Accessed 3-23-2000.

[20] Usevitch, B.E.; , "A tutorial on modern lossy wavelet image compression: foundations of JPEG 2000," *Signal Processing Magazine, IEEE* , vol.18, no.5, pp.22-35, Sep 2001.

[21] JPEG 2000 Part I Committee Draft Version 1.0, , Mar. 2000, ISO/IEC 1.29.15444-1.

[22] A. Samet, M. A. Ben Ayed, M. Loulou, and N. Masmoudi, "Comparison between JPEG and JPEG2000 still image compression standard," Proc. Vis., Imag., Image Process., Sep. 2002.

[23] D. Santa-Cruz, R. Grosbois, and T. Ebrahimi, "JPEG 2000 performance evaluation and assessment," Signal Process.: Image Commun., vol. 17, pp. 113–130, 2002.

[24] Penne baker, W. B.and Mitchell, J. L. JPEG – Still Image Data Compression Standards, Van Nostrand Reinhold, 1993.

[25] Wallace, G.K.,The JPEG still picture compression standard, Comm. of the ACM, 34(4), 30–44, 1991.

[26] Balster, E. J.; Fortener, B. T.; Turri, W. F.; , "Integer Computation of Lossy JPEG2000 Compression," *Image Processing, IEEE Transactions on* , vol.20, no.8, pp.2386-2391, Aug. 2011.

[27] T. Acharya and P.-S. Tsai, JPEG2000: Standard for Image Compression. Hoboken: Wiley, 2004.

[28] R. Sudhakar, Ms R Karthiga, S. Jayaraman, "Image Compression using Coding of Wavelet Coefficients- A Survey," ICGST-GVIP Journal, Vol. 5, Issue 6, June 2005.

[29] http://www.nasa.gov/multimedia/imagegallery/iotd.html.

[30] M. Rabbani, "Fundamentals and World Standards for Digital Image Compression,".