

*Republic of Iraq
Ministry of Higher Education and Scientific Research
Al-Nahrain University
College of Science*



Color Image Compression Based on Modified SPIHT Algorithm

*A Thesis Submitted to the College of Science, AL-Nahrain
University In Partial Fulfillment of the Requirements for
The Degree of Master of Science in Computer Science*

Submitted by:

Khyreia Saied Abd Al-Jibaar

(B.Sc. 2006)

Supervised by:

Dr. Ban Nadeem Dhannoon

April 2009

Rabia II 1430

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

((وَإِلَى اللَّهِ الْمَصِيرُ))

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Supervisor Certification

I certify that this thesis was prepared under my supervision at the Department of Computer Science/College of Science/Al-Nahrain University, by **Khyreia Saied Abd Al-Jibaar** as partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

Supervisor

Signature:



Name : **Dr. Ban Nadeem Dhannoon**

Title : **Assistance Professor**

Date : **8/4/2009**

The Head of the Department Certification

In view of the available recommendations, I forward this thesis for debate by the examination committee.

Signature:

Name : **Dr. Taha S. Bashaga**

Title : **Head of the department of Computer Science,
Al-Nahrain University.**

Date : / / 2009

Examining Committee Certification


We certify that we have read this thesis and as an examining committee, examined the student in its content and what is related to it and that in our opinion it meets the standard of a thesis for the degree of Master of Science in Computer Science.

Signature: 


Name : **Dr. Jane Jaleel Stephan**
Title : **Assistance Professor**
Date : **26/ 8 / 2009**
(Chairman)

Signature: 

Name : **Dr. Bushra K. AL-Abudi**
Title : **Assistance Professor**
Date : / / 2009
(Member)

Signature: 

Name : **Dr. Sawsan Kamal Thamer**
Title : **Lecturer**
Date : / / 2009
(Member)

Signature: 

Name : **Dr. Ban Nadeem Dhannoon**
Title : **Assistance Professor**
Date : **11/ 8 / 2009**
(Supervisor)

Approved by the Dean of the College of Science, Al-Nahrain University.

Signature:

Name : **Dr. Laith Abdul Aziz AL-Ani**
Title : **Assistance Professor**
Date : / / 2009
(Dean of College of Science)

A decorative border of repeating floral motifs surrounds the text. The motifs are stylized, symmetrical flowers with multiple petals, arranged in a continuous line.

Dedication

To My Parents . . .

Sister . . .

Brothers . . .

and Friends

To everyone

Taught me a letter

Khyreia

Acknowledgment

First of all, my great thanks to Allah who helped me and give me the ability to achieve this work.

My great thanks to my supervisor Dr.Ban Nadeem Thannoon for her guidance, supervision and efforts during this work.

Grateful for the Head of the department of computer science Dr.Taha S. Bashaga.

My deep gratitude to the employees and stuff of the computer science department.

Thanks to my parents, sister and my brothers for their help and patience during this work.

Finally thanks to all my friends for supporting and giving me advices.



List of Abbreviations

| Abbreviation | Meaning |
|---------------------|---|
| ASIC | Application Specific Integrated Circuit |
| BR | Bit Rate |
| BMP | Bitmap |
| CR | Compression Ratio |
| CMY | Cyan, magenta and yellow |
| DCT | Discrete Cosine Transform |
| DWT | Discrete Wavelet Transform |
| DPCM | Differential Pulse Code Modulation |
| ECG | Electro Cardio Graph |
| EZW | Embedded Zerotree Wavelet |
| GUI | Graphical User Interface |
| HWT | Haar Wavelet Transform |
| HVS | Human Vision System |
| HSV | Hue Saturation Value |
| HSL | Hue Saturation Luminance |
| JPEG | Joint photographic Expert Group |
| LTW | Lower-Tree Wavelet |
| LSP | List of Significant pixels |
| LIP | List of Insignificant Pixels |
| LIS | List of Insignificant Set |
| MSE | Mean Square Error |
| NTSC | National Television system Committee |
| NASA | National Aeronautics and Space Administration |
| PSNR | Peak Signal to Noise Ratio |
| PPM | Prediction with Partial string Matching |
| PAL | Phase Alternating Line |
| RLE | Run Length Encoding |
| SPIHT | Set Partitioning In Hierarchical Tree |
| SECAM | Sequential Couleur a Memoire |
| TC | Transform Coding |
| VLSI | Very Large Scale Integration |
| WP | Wavelet Packet |
| WT | Wavelet Transform |

Abstract

Image compression is the application of data compression on digital images. In effect, the objective is to reduce redundancy of image data in order to be able to store or transmit data in an efficient form. There have been many types of compression algorithms developed. This work exploits an image compression based on Set Partitioning In hierarchical Tree (SPIHT) methods.

In this work, the RGB image was transformed into YC_bC_r image, the goal of this transforming is to prepare the image for encoding process by eliminating any irrelevant information, the C_b and C_r bands are down sampled due to their poor spatial resolution. Then the compression technique is started by applying the wavelet transform on each component, the first sub band (i.e. LL sub band) is coded by using Discrete Cosine Transform (DCT), and uniform quantization. The other sub bands are coded by hierarchical uniform quantization, and then the SPIHT method was applied on each color band separately. At the end, some spatial coding steps were applied on List of Significant Pixels (LSP) like (Run Length Encoding (RLE) and Shift Coding) to gain more compression.

Six color images are used to test the system performance, At first, a comparison between Haar and Tap9/7 wavelet transforms were made, the results show that tap9/7 give better PSNR on the selected images, so this work continued by using Tap9/7. Then many tests were made to select the best parameter values for the uniform quantization. At last a comparison between the test results of the original SPIHT, Modified SPIHT and Modified SPIHT with DCT & (Run Length Encoding) RLE were made and the result shows that the last method gives good PSNR with relatively high CR).

Table of Contents

| Chapter One: General Introduction | | |
|--|---|----|
| 1.1 | Introduction | 1 |
| 1.2 | Image Compression | 2 |
| 1.3 | The Principles behind Compression | 2 |
| 1.4 | Literature Survey | 3 |
| 1.5 | Aim of Thesis | 8 |
| 1.6 | Thesis Layout | 8 |
| Chapter Two: Theoretical Background | | |
| 2.1 | Introduction | 10 |
| 2.2 | Image Redundancy | 11 |
| 2.3 | Color Space | 13 |
| 2.4 | BMP File | 18 |
| 2.5 | Image Compression Methods | 18 |
| 2.6 | Lossless Compression Methods | 19 |
| 2.6.1 | S-Shift Coding | 20 |
| 2.6.2 | Run Length Encoding (RLE) | 21 |
| 2.7 | Lossy Compression Methods | 21 |
| 2.7.1 | Quantization | 22 |
| 2.7.2 | Transform Coding (TC) | 24 |
| 2.8 | Wavelet Transform (WT) | 24 |
| 2.9 | Haar Wavelet Transform (HWT) | 27 |
| 2.9.1 | Forward Haar Wavelet Transform (FHWT) | 28 |
| 2.9.2 | Inverse Haar Wavelet Transform (IHWT) | 28 |
| 2.10 | Float wavelet Transform (Tap 9/7) | 29 |
| 2.11 | Wavelet Transform Characteristics | 30 |
| 2.12 | Discrete Cosine Transform (DCT) | 32 |
| 2.13 | Set Partitioning In Hierarchical Tree (SPIHT) | 33 |
| 2.13.1 | Set Partitioning Sorting Algorithm | 36 |
| 2.13.2 | Spatial Orientation Trees | 37 |
| 2.13.3 | SPIHT Coding | 39 |
| 2.13.4 | The Main Steps of the SPIHT Encoder | 40 |
| 2.14 | Fidelity Criteria | 42 |
| 2.15 | Performance Parameters | 44 |

| Chapter Three: An Image compression based on Modified SPIHT Algorithm | | |
|--|--|-----|
| 3.1 | Introduction | 45 |
| 3.2 | System Model | 46 |
| 3.2.1 | Compression Unit | 47 |
| 3.2.1.1 | Image Loading | 47 |
| 3.2.1.2 | Convert from RGB to $YCbCr$ | 48 |
| 3.2.1.3 | Down Sample | 49 |
| 3.2.1.4 | Wavelet Transform | 51 |
| 3.2.1.5 | Hierarchical Uniform Quantization | 55 |
| 3.2.1.6 | Discrete Cosine Transform (DCT) | 57 |
| 3.2.1.7 | Uniform Quantization of the DCT Coefficients | 58 |
| 3.2.1.8 | The modified SPIHT Coding | 59 |
| 3.2.1.9 | Mapping to Positive | 62 |
| 3.2.1.10 | Run Length Encoding (RLE) | 63 |
| 3.2.1.11 | S-Shift Optimizer | 64 |
| 3.2.1.12 | Shift Coding | 65 |
| 3.2.2 | Decompression Unit | 66 |
| 3.2.2.1 | Shift Decoding | 67 |
| 3.2.2.2 | Run Length Decoding | 68 |
| 3.2.2.3 | SPIHT Decoding | 68 |
| 3.2.2.4 | Dequantization | 69 |
| 3.2.2.5 | Dequantization of the DCT Coefficients | 71 |
| 3.2.2.6 | Inverse Discrete Cosine Transform | 72 |
| 3.2.2.7 | Inverse Wavelet Transform | 73 |
| 3.2.2.8 | Up Sampling | 76 |
| 3.2.2.9 | $YCbCr$ to RGB Transform | 77 |
| Chapter Four: Test and Results | | |
| 4.1 | Introduction | 78 |
| 4.2 | Results and Discussion | 79 |
| Chapter Five: Conclusions and Future Works | | |
| 5.1 | Conclusions | 103 |
| 5.2 | Future Works | 103 |
| References | | |



Chapter One

General Introduction

CHAPTER ONE

General Introduction

1.1 Introduction

As the amount of digital information was grown and increased, the need for efficient ways of sorting and transmission this information were increased as well. Data compression is concerned with the means of providing an efficient way to present information by using techniques to exploit different kinds of structures that may present in the data.

Data compression is the process of converting data files into smaller files to improve the efficiency of storage and transmission, (in other worlds, compression is the process of representing information in a compact form). As one of the enabling technologies of the multimedia revolution, data compression is a key to rapid progress being made in information technology. It would not be practical to put images, audio, and video data as there are on web sites without compression [Xia01].

In order to be useful, a compression algorithm has corresponding decompression algorithms that reproduce the original file from the compressed one. Many types of compression algorithms were develop, these algorithms classified into two broad types: *lossless algorithms* and *lossy algorithms*. A lossless algorithm reproduces the exact original file. A lossy algorithm, as its name implies some loss of data, which may be unacceptable in many applications. The use of these methods depends on the data used, for example in text any loss of information cause an error in the text that means a loss of small information cause an error. So the text compression requires the lossless method. There are also many situations where loss may be either unnoticeable or acceptable. In image compression, for example, the exact reconstructed value of each sample of

the image is not necessary. Depending on the quality required for the reconstructed image, varying amounts of information loss can be accepted [Xia01].

Efficient image compression solutions are becoming more critical with the recent growth of data intensive and multimedia-based web application.

1.2 Image Compression

Image compression involves reducing the size of image data files, while retaining necessary information [Umb98].

The progress of many system operations, such as downloading a file, may also be displayed graphically. Many applications provide a Graphical User Interface (GUI). Computer graphics is used in many areas in everyday life to convert many types of complex information to images. Thus, images are important, but they tend to be big, since modern hardware can display many colors, it is common to have a pixel represented internally as a 24-bit number, where the percentages of red, green, and blue occupy 8 bits each. Such a 24-bit pixel can specify one of $2^{24} \approx 16.78$ million colors. As a result, an image at a resolution of 512×512 that consists of such pixels occupies 786.432 bytes. At a resolution of 1024×1024 it becomes four times as big, requiring 3,145,728 bytes. This is why image compression is so important [Sal02].

1.3 The Principles behind Compression

A common characteristic of most images is that the neighboring pixels are correlated and therefore contain redundant information. The foremost task then is to find less correlated representation of the image. Two fundamental components of compression are redundancy and irrelevancy reductions:

- 1. Redundancy Reduction:** aims at removing duplication from the signal source (image/video).
- 2. Irrelevancy Reduction:** omits parts of the signal that will not be notice by the signal receiver, namely the Human Visual System (HVS).

In general, three types of redundancy can be identified:

- **Spatial Redundancy** or correlation between neighboring pixel values.
- **Spectral Redundancy** or correlation between different color planes or spectral bands.
- **Temporal Redundancy** or correlation between adjacent frames in a sequence of images (in video application).

Image compression research aims at reducing the number of bits needed to represent an image by removing the spatial and spectral redundancies as much as possible [Sah04].

1.4 Literature Survey:

- 1. Xiang 2001, "Image Compression by Wavelet Transform" [Xia01]:**

This thesis studies image compression with wavelet transform. As a necessary background, the basic concept of graphical image storage and currently used compression algorithms are discussed. The mathematical properties of several types of wavelets, including Haar, Daubechies, and biorthogonal spline wavelets are covered and the Embedded Zerotree Wavelet (EZW) coding algorithm is introduced. the EZW coding method is one of the more efficient quantization and coding methods for wavelet coefficients.

2. Thomas et al 2002, "SPIHT Image Compression on FPGAs" [Tho02] :

In this paper, they have demonstrated a viable image compression routine on a reconfigurable platform. They showed that by analyzing the range of data processed by each section of the algorithm, it is advantageous to create optimized memory structures as with their Variable Fixed Point work. Doing so minimizes memory usage and yields efficient data transfers. (I.e. each bit transferred between memory and the processor board directly impacts the final result). In addition their Fixed Order SPIHT work illustrates how by making slight adjustments to an existing algorithm, it is possible to dramatically increase the performance of a custom hardware implementation and simultaneously yield essentially identical results. With Fixed Order SPIHT the throughput of the system increases by roughly an order of magnitude while still matching the original algorithm's PSNR curve. Their SPIHT work is part of an ongoing development effort funded by National Aeronautics and Space Administration (NASA).

3. Abdul Hameed 2003, "Advanced Compression Techniques for Multimedia Applications" [Abd03]:

In this thesis, the lossless compression is applied for multimedia files (i.e. such as, image, audio and text). The implemented data compression algorithms are belonging to three main classes: Run Length Encoding (RLE) techniques, Statistical techniques, and Dictionary techniques. For each of the proposed approaches, an algorithm is constructed. These algorithms are tested by changing their parameters. A standard test files for given application were used in the tests. The Statistical Coder with order 3 model named Prediction with Partial string Matching (PPM) produced best

compression ratio that the well known software WinZip (V.8) reduction ratio to about 3% on average and over 8% for best case.

4. Kotteri 2004, "Optimal, Multiplierless Implementations of the Discrete Wavelet Transform for Image Compression Applications" [kot04]:

He designed and implemented image compression by using biorthogonal Tap9/7 Discrete Wavelet Transform (DWT), and applied uniform scalar quantization on wavelet coefficient. He utilized the fidelity measure (MSE and PSNR) to assess the quality of the compressed image to avoid wasting in computation he improved the efficiency of the filter bank.

5. Danyali 2004, "Fully Spatial and SNR Scalable, SPIHT-Based Image Coding for Transmission Over Heterogeneous Networks" [Dan04]:

This paper presents a fully scalable image coding scheme based on the Set Partitioning In Hierarchical Trees (SPIHT) algorithm. The proposed algorithm, called Fully Scalable SPIHT (FS-SPIHT), adds the spatial scalability feature to the SPIHT algorithm. It provides this new functionality without sacrificing other important features of the original SPIHT bitstream such as: compression efficiency, full embeddedness and rate scalability. The flexible output bitstream of the FS-SPIHT encoder which consists of a set of embedded parts related to different resolutions and quality levels can be easily adapted (reordered) to given bandwidth and resolution requirements by a simple parser without decoding the bitstream.

FS-SPIHT is a very good candidate for image communication over heterogeneous networks which require high degree of scalability from image coding systems.

6. Nikola 2005, "Modified SPIHT algorithm for wavelet packet image coding" [Nik05]:

This paper introduced a new implementation of Wavelet Packet (WP) decomposition which is combined with (SPIHT) compression scheme. It provided the analysis of the problems arising from the application of zerotree quantization based algorithms (such as SPIHT) to wavelet packet transform coefficients. The compression performance of WP-SPIHT has been compared to SPIHT both visually and in terms of PSNR. WP-SPIHT significantly outperforms SPIHT for textures. For natural images, which consist of both smooth and textured areas, the chosen cost function used in WP is not capable of estimating correctly the true cost for a case when the subband is encoded with SPIHT. For those images the PSNR performance of WP-SPIHT is usually slightly worse.

7. Marc 2005, "Silicon Implementation of the SPIHT Algorithm for Compression of ECG Records" [Mar05]:

In this paper, the first Very Large Scale Integration (VLSI) implementation using a Modified SPIHT algorithm (MSPIHT) was presented. MSPIHT compression ratios of up to 20:1 for Electro Cardio Graph (ECG) signals leads to acceptable results for visual inspection by medical doctors. Further compression may lead to false diagnosis by medical doctor's visual inspection due to vanishing details. The Application Specific Integrated Circuit (ASIC) runs typically 0.35 s to process a 512×512 image at the maximum clock frequency of 75MHz.

8. Khelifi 2005, "A Scalable SPIHT-Based Multispectral Image Compression Technique" [Khe05]:

In this paper, multispectral image compression based on the celebrated SPIHT algorithm had been addressed. In order to exploit the spectral redundancy within multi-band images, an efficient technique has been proposed. The key idea consisted in joining each two consecutive bands according to a new tree-structure using a virtual parent-descendant relationship. Furthermore, to overcome the worst cases where one of the direct descendants is significant while the other is not, the proposed algorithm, JST-SPIHT, uses four types of insignificant sets during the sorting pass. Simulation results, conducted at different bit-rates and carried out on two sample multispectral images show a highly better performance of the proposed technique when compared to the conventional 3D-SPIHT.

9. Susan 2007, "Image Compression Based on Fractal and wavelet Transform" [Sus07]:

In this thesis, both fractal and wavelet transform were assembled in hybrid compression system to compress image. This helped to overcome some of problems which faced the compression methods based on each method separately. Also the hybrid system will take the advantages of both methods.

10. Sara 2008, "Using Discrete Cosine Transform To Encode Approximation Wavelet subband" [Sar08]:

In this thesis, a combined transform coding scheme was proposed, this scheme utilizes both Wavelet transform and DCT, the first will be considered as the main core engine of the compression system, while the second transform will be used as secondary coding tool. The advantages of both transforms will be taken into consideration to encode the portioned regions of the image. The

requirement of low complexity is taken into consideration in designing the scheme of the proposed system.

1.5 Aim of Thesis

An image compression research aims to reduce the number of bits needed to represent an image by removing the spatial and spectral redundancies as much as possible. In this research, the modified SPIHT algorithm will be used to compress bitmap color images of type (RGB) after transformed them into ($YCbCr$) model. Other encoding methods such as (DCT & RLE) were used to enhance the compressed file and the quality of the reconstructed images.

1.6 Thesis Layout

Beside to this chapter the remaining part of thesis consists of the following four chapters:

- **Chapter two: *Theoretical Background***

This chapter discussed the image compression techniques in details, including Wavelet Transform, Discrete Cosine Transform and the SPIHT algorithm.

- **Chapter Three: *The Proposed Image Compression System***

This chapter includes in details the designed and implemented image compression models; all the developed algorithms that used in this research work are presented.

- **Chapter four: *Performance Measure and Test Result***

This chapter contains the result of the conducted tests on some samples of images that used as test material in this work. Used performance criteria are the fidelity measures (MSE, PSNR) beside the compression ratio (CR).

- **Chapter Five: *Conclusions and Future Work***

This chapter includes the derived conclusions and some suggestions for future works.



Chapter Two

Theoretical Background

CHAPTER TWO

Theoretical Background

2.1 Introduction

With the advance of the information age, the need for mass information storage and fast communication links was grown. Images are widely used in computer applications and storing these images in less memory leads to a direct reduction in storage cost and faster data transmission. These facts justify the efforts of academic and industrial research centers and personal on new better performance image compression algorithms.

Images are stored on computers as collections of bits representing pixels or points forming the image elements. Since the human eye can process large amounts of information, many pixels are required to store moderate quality images. Most data contains some amount of redundancy, which can sometimes be removed for storage and replaced for recovery, but this redundancy does not lead to high compression ratios. An image can be changed in many ways that are either not detectable by the Human Vision System (HVS), or do not contribute to the degradation of the image. The standard methods of image compression come in several forms. In this chapter some relevant concepts to image compression discipline given. Also some of the common image compression techniques are illustrated. And explains the theoretical concept of image compression using the (SPIHT) algorithm which is the subject of this work.

2.2 Image Redundancy

The common characteristic of most images is that the neighboring pixels contain redundant information [Sah04]. Digital image data compression can be performed by removing all redundancies that exist in an image data, so it takes up less storage space and require less bandwidth to be transmitted. There are many types of redundancies, among these are the following:

- 1. Interpixel Redundancy:** interpixel redundancy implies that the intensity value of a pixel can be predicated from its neighboring pixel, because the values of adjacent pixels tend to be highly correlated [Shi00]. It is the result of the fact that in most images the brightness levels do not change rapidly, but change gradually, so that the adjacent pixels tend to be relatively close to each other in value [Gon02]. This kind of redundancy can be removed by transforming the image into a state where the interpixel redundancy can be discovered and eliminated, and this kind of transformation process is called a mapping. Sometimes the interpixels redundancy is called geometric redundancy, spatial redundancy or interframe redundancy (for video and motion image).
- 2. Coding redundancy:** Gray levels are usually coded with equal-length binary codes, and coding redundancy may exist in such codes. For example, the 8 bit/pixel image allows 256 gray level values but sometimes the actual image may contain only 16 gray level values, then as a suboptimal coding only 4 bit/pixel is actually needed. Also, more efficient coding can be achieved if variable-length coding is employed. Variable-length coding assigns fewer bits to the gray levels with higher occurrence probabilities in an image. The average length required to

represent a pixel within a compressed image using variable-length coding method, is given by [Sal98]:

$$\bar{n} = \sum_{k=0}^{L-1} n(r_k)P(r_k) \dots\dots\dots (2.1)$$

Where, r_k is the gray level of an image.

$P(r_k)$ is the probability of occurrence of r_k .

$n(r_k)$ is the number of bits used to represent each value of r_k .

k is the number of gray levels.

3. **Psychovisual Redundancy:** Psychovisual redundancy originates from the characteristics of the Human Visual System (HVS). The human perception is not a constant pixel oriented mechanism. This implies that every area in an image is not processed with same amount of sensitivity, and the image areas which do not contribute to the valuable visual content can be removed without a major loss in quality for the human perceiver. The elimination of these redundancies can be considered as a sort of quantization, which is an irreversible process [Fri95]. For color images, this kind of redundancy can be used to reduce the size of the chromatic components, since the human eyes are less sensitive to the variation in chromaticity than the variation in light intensity [Shi00].
4. **Spectral Redundancy:** it is the correlation between different spectral bands [Shi00].
5. **Temporal Redundancy:** it is the correlation between different subsequence [Haf01].

2.3 Color Space

A wide range of colors can be created by the primary colors of pigment (Cyan, Magenta, Yellow) CMY. Those colors then define a specific color space. To create a three-dimensional representation of a color space, we can assign the amount of magenta color to the representations X axis, the amount of cyan to its Y axis, and the amount of yellow to its Z axis. The resulting 3-D space provides a unique position for every possible color that can be created by combining those three pigments. However, this is not the only possible color space, many color spaces can be represented as three-dimensional (X, Y, Z) values in this manner, but some have more, or fewer dimensions, and some cannot be represented in this way at all [Web1].

In the following some of the popular color spaces are given:

1. **CMY (Cyan, Magenta and Yellow):** it is a subtractive color mixing used in the printing process, because it describes what kind of inks need to be applied so the light reflected from the substrate and through the inks produces a given color. The sum of the three CMY color values produces black [web1].

The black produces by a CMY color system often falls short of being a true black. To produce a more accurate black in printed images, black is often added as a fourth colore component. This is known as the CMYK color system and is commonly used in printing industry [Law99].

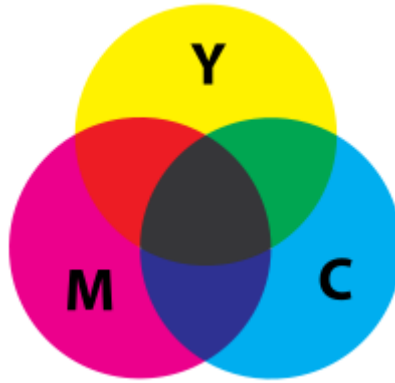


Figure (2.1) Subtractive color mixing

- 2. RGB (Red, Green and Blue):** it is an additive system in which varying amounts of the colors Red, Green and Blue are added to black to produce new colors, Graphics files using the RGB color system represent each pixel as a color triplet-three numerical values in the form (R, G and B), each representing the amount of Red, Green and Blue in the pixel, respectively [Mur07].

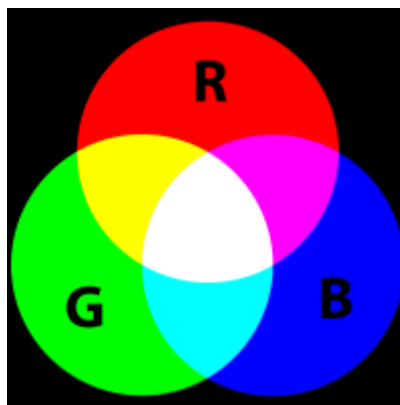


Figure (2.2) additive color mixing

- 3. HSV (Hue, Saturation and value):** also known as HSB (Hue, Saturation and Brightness) is often used by artists because it is often more natural to think about a color in terms of hue and saturation than in terms of additive or subtractive color components. HSV is a transformation of an RGB colorspace, and its components and

colorimetry are relative to the RGB colorspace from which it was derived [Web1].

4. **HSL (Hue, Saturation, Lightness/Luminance):** also known as HLS or HSI (hue, saturation, Intensity) is quite similar to HSV, with "lightness" replacing "brightness". The difference is that the brightness of a pure color is equal to the brightness of white, while the lightness of a pure color is equal to the lightness of a medium gray [Web1].
5. **YUV:** The YUV model defines a color space in terms of one luma and two chrominance components. The YUV color model is used in the (Phase Alternating Line) PAL, (National Television system Committee) NTSC, and SECAM composite color video standards.

YUV signals are created from an original RGB (red, green and blue) source. The weighted values of R, G, and B are added together to produce a single Y signal, representing the overall brightness, or luminance, of that spot. The U signal is then created by subtracting the Y from the blue signal of the original RGB, and then scaling; V is created by subtracting the Y from the red, and then scaling by a different factor. This can be accomplished easily with analog circuitry [Web2].

Mathematically, the analog version of YUV can be obtained from RGB with the following relationships:

$$Y = 0.299R + 0.587G + 0.114B \dots\dots\dots (2.2)$$

$$U = 0.436(B - Y) / (1 - 0.114) \dots\dots\dots (2.3)$$

$$V = 0.615(R - Y) / (1 - 0.299) \dots\dots\dots (2.4)$$

The inverse relationship, from YUV to RGB, is given by:-

$$R = Y + 1.13983V \dots\dots\dots(2.5)$$

$$G = Y - 0.39465U - 0.58060V \dots\dots\dots(2.6)$$

$$B = Y + 2.03211U \dots\dots\dots(2.7)$$

6. YIQ: This color space has been utilized in NTSC (National Television Systems Committee) TV systems for years. Note that NTSC is an analog composite color TV standard and is used in North America and Japan. The Y component is still the luminance. The two chrominance components are the linear transformation of the U and V components defined in the YUV model [Shi00]. Specifically,

$$I = -0.545U + 0.839V \dots\dots\dots(2.8)$$

$$Q = 0.839U + 0.545V \dots\dots\dots(2.9)$$

Substituting the U and V expressed in Equations 1.4 and 1.5 into the above two equations, we can express YIQ directly in terms of RGB. That is,

$$\begin{pmatrix} Y \\ I \\ Q \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \dots\dots\dots(2.10)$$

7. YD_bD_r: The YD_bD_r model is used in the SECAM (Sequential Couleur a Memoire) TV system. Note that SECAM is used in France, Russia, and some eastern European countries. The relationship between YD_bD_r and RGB appears below.

$$\begin{pmatrix} Y \\ Db \\ Dr \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ 0.450 & -0.883 & 1.333 \\ -1.333 & 1.116 & -0.217 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \dots\dots\dots(2.11)$$

That is,

$$Db = 3.059U \dots\dots\dots (2.12)$$

$$Dr = -2.169V \dots\dots\dots (2.13)$$

8. YCbCr: From the above, we can see that the U and V chrominance components are differences between the gamma-corrected color B and the luminance Y, and the gamma-correct R and the luminance Y, respectively. The chrominance component pairs I and Q, and Db and Dr are both linear transforms of U and V. Hence they are very closely related to each other. It is noted that U and V may be negative as well. In order to make chrominance components nonnegative, the Y, U, and V are scaled and shifted to produce the YCbCr model, which is used in the international coding standards JPEG and MPEG [Shi00], where Y is the luminous components while C_b and C_r provide the Color information:

$$\begin{pmatrix} Y \\ Cb \\ Cr \end{pmatrix} = \begin{pmatrix} 0.257 & 0.504 & 0.098 \\ -0.148 & -0.291 & 0.439 \\ 0.439 & -0.368 & -0.071 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \dots\dots\dots (2.14)$$

The inverse relationship, from YCbCr to RGB, is given by:-

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1 & 0.001 & 1.582 \\ 1 & 1.862 & -0.0002 \\ 1 & -0.188 & -0.469 \end{pmatrix} \begin{pmatrix} Y \\ Cb \\ Cr \end{pmatrix} \dots\dots\dots (2.15)$$

In this way %80 of the information will be in the Y sub band and %20 will be in the C_b, C_r sub bands, the goal of this preprocessing is to prepare the image for encoding process by eliminating any irrelevant information.

2.4 BMP File

Bitmap files are especially suited for the storage of real-world images; complex images can be rasterized in conjunction with video, scanning, and photographic equipment and stored in a bitmap format.

Advantages of bitmap files include the following:

1. Bitmap files may be easily created from existing pixel data stored in an array in memory.
2. Retrieving pixel data stored in a bitmap file may often be accomplished by using a set of coordinates that allows the data to be conceptualized as a grid.
3. Pixel values may be modified individually or as large groups.

Bitmap files, however do have drawbacks: they can be very large, particularly if the image contains a large number of colors. Data compression can shrink the size of pixel data, but the data must be expanded before it can be used, and this can slow down the reading process. Also, the more complex a bitmap image (large number of colors and minute detail), the less efficient the compression process will be [Mur07].

2.5 Image Compression Methods

Compression process takes an input X and generates a representation X_c that hopefully requires fewer storage sizes. While the reconstruction algorithm operates on the compressed representation X_c to generate the reconstruction Y .

Based on the difference between the original and reconstructed version, data compression schemes can be divided into two broad classes, see figure (2.3). The first is lossless compression, at which Y is identical to X . while the second is lossy compression, which generally provided much higher compression than lossless compression but makes Y not exactly as X [Add00].

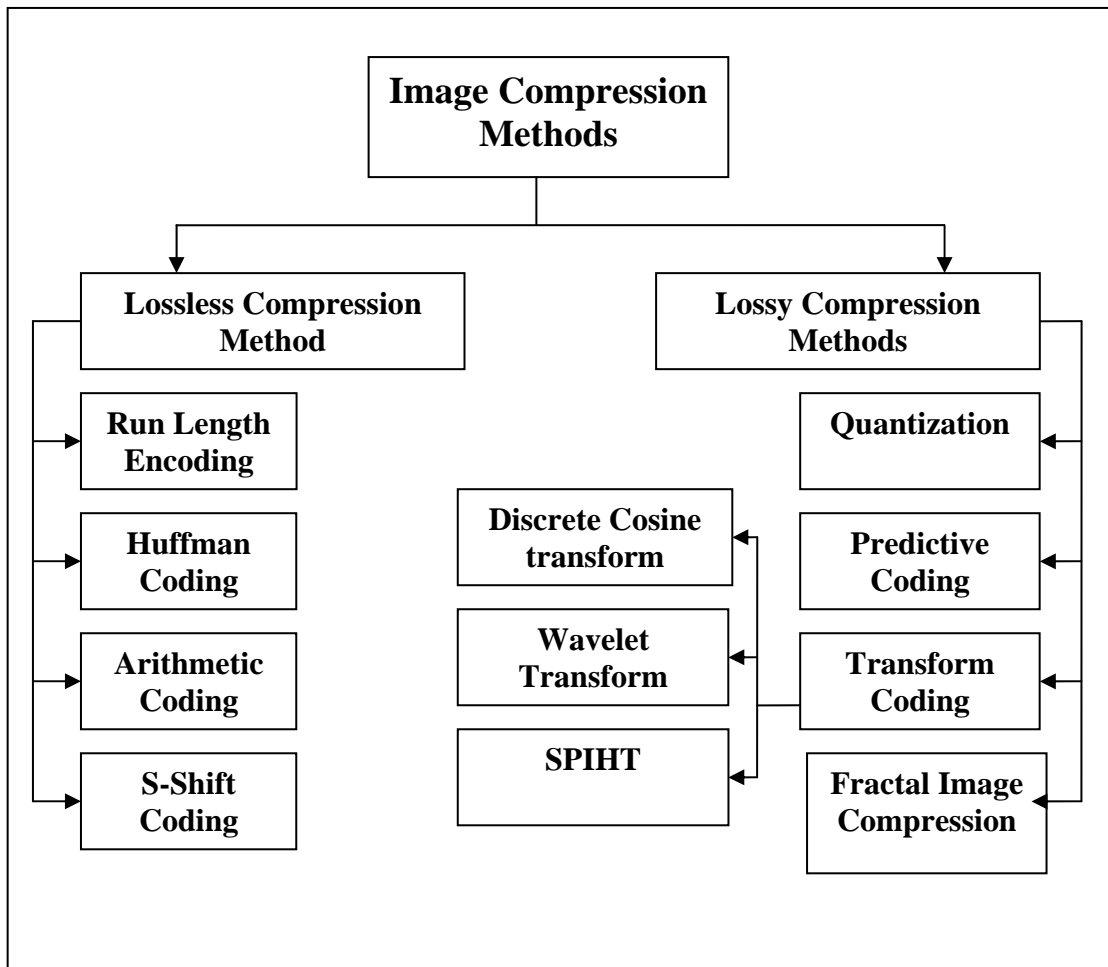


Figure (2.3) the most popular image compression methods

2.6 Lossless Compression Methods

Lossless compression method provide the guarantee that no pixel difference will occur between the original and decompressed image, in other words, lossless schemes result in reconstructed data that exactly matches the original. It is generally used applications that cannot allow any difference between the original and reconstructed data [Avc02].

The most popular lossless compression methods are:

1. Huffman Coding.
2. Arithmetic Coding.
3. S-Shift Coding.
4. Run Length coding

Below, a description of the methods that were used in this research.

2.6.1 S-Shift Coding

The idea of this method is to encode a set of numbers by codewords whose bit length is less than the bit length required to represent the maximum value of the set. While the numbers whose values are large may encode using sequence of codewords, each sequence may consist of many short-length codewords, the values of the codewords are determined according to the following equation [Gon00].

$$X = nW_m + W_r \dots\dots\dots (2.16)$$

Where,

X is the number to be coded.

n is the number of codewords that used to encode the number X.

W_m is the lowest value which cannot be coded by using a single codeword.

W_r is the value of the last word used to encode X.

The values of **W_m**, **W_r** and **n** are determined using the following equations:

$$W_m = 2^b - 1 \dots\dots\dots (2.17)$$

$$W_r = X \text{ mod } W_m \dots\dots\dots (2.18)$$

$$n = X \text{ div } W_m \dots\dots\dots (2.19)$$

Where **b** is the number of bits used to represent each single s-codeword.

The performance of shift coding is better when the histogram of the sequence of coded numbers is highly peaked. The performance of shift coding is better than Huffman and arithmetic coding when the histogram has long tails [Ibr04, Gon00].

2.6.2 Run Length Encoding (RLE)

It is simple, intuitive, and fairly effective compression scheme for bitmap graphics. Its main concept is to detect repeating pixels on each row of an image and output them as a pixel count and pixel value pair, instead of outputting the repeated pixels individually. RLE dose not do well for stipple patterns or scanned images, which do not have repeating pixels in rows the encoding for these types of images, may actually be larger after RLE. Despite this limitation, RLE is very good for other types of images, and is supported by the BMP, TIFF, as well as many others [Mur07].

RLE work by reducing the physical size of a repeating string of characters. This repeating string, called a run, is typically encoded into two bytes. The first byte represents the number of characters in the run and is called the run count. In practice, an encoded run may contain 1 to 128 or 256 characters; the run count usually contains as the number of characters minus one (a value in the range of 0 to 127 or 255). The secound byte is the value of the character in the run. This is in the range of 0 to 255, and is called the run value. Uncompressed, a character run of 15 A characters would normally required 15 byte to store:AAAAAAAAAAAAAAAAA. The same string after RLE would require only two bytes: 15A. The 15A code generated to represent the character string is called an RLE packet. Here, the first byte 15 is the run count and contains the number of repetitions. The second byte A is the run value and contains the actual repeated value in the run [Mur07].

2.7 Lossy Compression Methods

Lossy compression involves elimination of "less important" information with respect to the "goodness" of the reconstructed data in the process of compression.

The most popular lossy compression methods are:

1. Predictive Coding.
2. Quantization.
3. Transform Coding.
4. Fractal Image Compression.

The image compression techniques based on wavelet transform methods are lossy. This involves eliminating the wavelet transform coefficients which are "less important" in contributing to the image's appearance and keeping the rest. More specifically, the coefficients with higher magnitudes are more important than coefficients with lower values [Wan00].

Below, a description of the methods that were used in this research.

2.7.1 Quantization

The dictionary definition of term "quantization" is to restrict a variable quantity to discrete values rather than to a continuous set of values [Sal06]. Any analog quantity that is to be processed by a digital computer or digital system must be converted to an integer number proportional to its amplitude. The conversion process between analog samples and discrete-valued samples is called quantization [Pra01].

In the field of data compression, quantization is used in two ways:

1. if the data to be compressed is in the form of large numbers, quantization is used to convert it to small numbers, small numbers take less space than large ones, so quantization generates compression. On the other hand, small numbers generally contain less information than large ones, so quantization results in lossy compression.

2. if the data to be compressed is analoge (e.g., a voltage that changes with time) quantization is used to digitize it into small numbers. The smaller number is the better compression, but also the greater number is the loss of information. This aspect of quantization is used by several speech compression methods.

There are two types of quantization: scalar Quantization and Vector Quantization. In Scalar Quantization, each input pixel is treated separately in producing the output, while in Vector Quantization the input pixels are clubbed together in groups called vectors, and processed to give the output. This clubbing of data and treating them as a single unit increases the optimality of the vector quantizer, but at the cost of increased computational complexity [Sal02].

A quantizer can be specified by its input partitions and outputs levels (also called reproduction points). If the input range is divided into levels of equal spacing, then the quantizer is termed as Uniform Quantizer, and if not, it is termed as a Non-Uniform Quantizer. A uniform quantizer can be easily specified by its lower bound and the step size. Also, implementing a uniform quantizer is easier than a non-uniform quantizer.

The quantization error $(X-XQ)$ is used as a measure of the optimality of the quantizer and dequantizer [Sal02].

2.7.2 Transform Coding (TC)

Although the prediction-residual coding method can be made by two dimensional, it is very hard to fully exploit the two dimensional correlation using only prediction. A better way is to transform the pixels in the image domain into another domain, where the representation is more natural and therefore more compact. This representation should preferably be such that some coefficients give the bulk of the energy in the image, while others are very likely to be very small or zero. When the latter ones are small, it signifies that the correlation that was expected is present. To have this kind of representation there is a need to decorrelate the coefficients with a reversible transformation (at least almost reversible), where still maintaining the same amount of total energy in the basis coefficients. Various fast algorithms were developed to perform the necessary transformations like [Nis98][sal02]:

1. Fourier Transform.
2. Discrete Cosine Transform (DCT).
3. Wavelet Transform.
4. Discrete Wavelet Transform (DWT).
5. Set Partitioning In Hierarchical Tree (SPIHT).
6. Joined Photographic Expert Group (JPEG) 2000.

2.8 Wavelet Transform (WT)

During the last decade, the wavelet transform has gained the attention of many researchers in the field of image compression. In images, fine information content is generally found in the high frequencies whereas the coarse information content exists in the low frequencies. The multi-resolution capability of wavelet transform is used to decompose the image into multiple frequency bands.

The fundamental idea behind wavelets is to analyze the signal at different scales or resolutions, which is called multiresolution. Wavelets are functions used to localize a given signal in both space and scaling domains. The wavelet transform is suited for non stationary signals. (Like very brief signals and signals with interesting components at different scales) **[Hub95]**.

An image signal can be analyzing by passing it through an analysis filter bank. This analysis filter bank consists of a low pass and a high pass filter at each decomposition stage **[Gra95]**.

When a signal passes through these filters, it is split into two bands. The low pass filter, which corresponds to an averaging operation, extracts the coarse information of the signal. The high pass filter, which corresponds to a differencing operation, extracts the detail information of the signal. The output of the filtering operations is then decimated by two **[Hil94]**.

The two dimensional wavelet transform can be accomplished by performing two separate one-dimensional transforms, as depicted in Figure (2.4), first, the image is filtered along the x-dimension and decimated by two. Then, it is followed by filtering the sub-image along the y-dimension and decimated by two. Finally, the image data contents are split into four bands denoted by LL, HL, LH and HH after one-level decomposition **[Mul97]**.

Further decompositions can be achieved by acting upon the LL subband successively, and then the resultant image is split into multiple bands **[Sto94, Bur98]**.

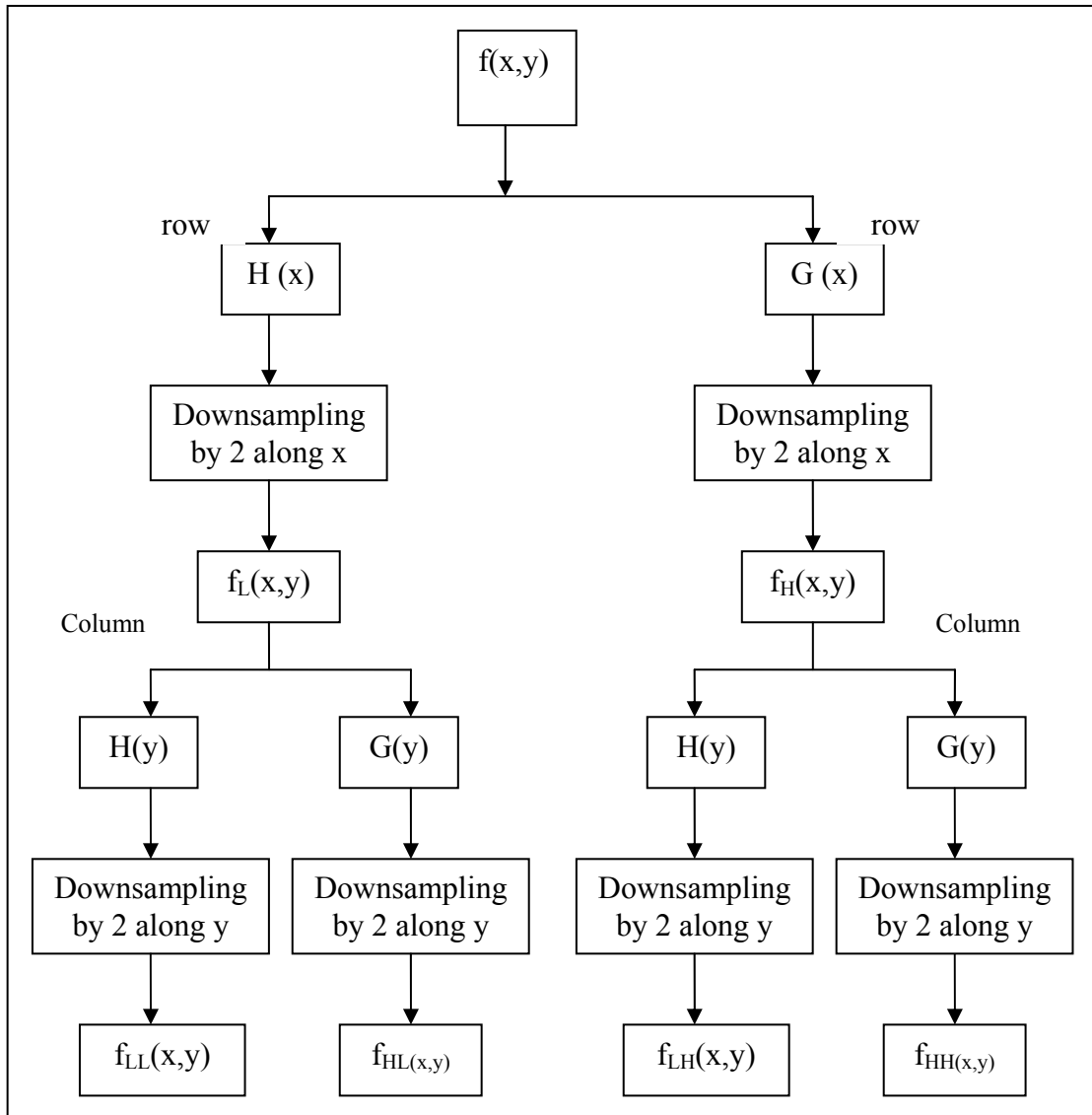


Figure (2.4) wavelet decomposition

In mathematical terms, the averaging operation (or low pass filtering) is the inner product between the signal and the scaling function whereas the differencing operation (or high pass filtering) is the inner product between the signal and the wavelet function [Cro01].

The reconstruction of the image can be carried out by the following procedure, first, the four subbands are up-sampled by a factor of two at the coarsest scale, and the subbands are filtered in each dimension. Then the sum of the four filtered subbands is determined to reach the low-low subband for the next finer scale. This process is repeated until the image is fully reconstructed, as depicted in Figure (2.5) [Val01].

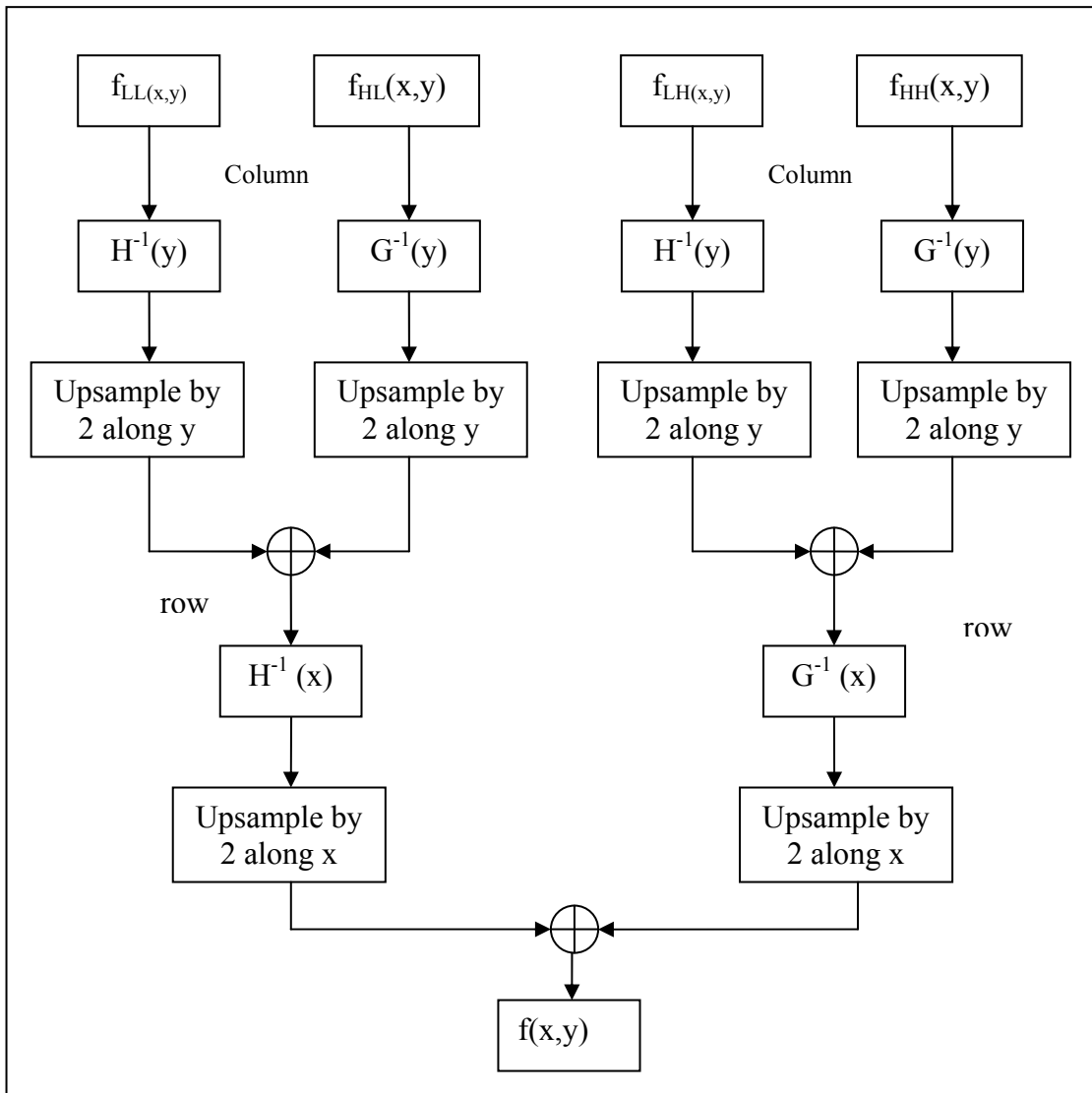


Figure (2.5) The two-dimensional inverse wavelet transform

2.9 Haar Wavelet Transform (HWT)

The oldest and most basic wavelet system had been constructed from the Haar basic function. The equation for forward Haar wavelet transform and inverse Haar wavelet transform are given in the following subsections.

2.9.1 Forward Haar Wavelet Transform (FHWT) [Jia03]

Given an input sequence (x_i) , $i=0 \dots N-1$, then FHWT produces L_i and H_i (where $i=0 \dots N/2-1$) by using the following transforms equations:

A. if N is even:

$$\left. \begin{aligned} L_i &= \frac{X_{2i} + X_{2i+1}}{\sqrt{2}}, i=0 \dots (N/2)-1 \\ H_i &= \frac{X_{2i} - X_{2i+1}}{\sqrt{2}}, i=0 \dots (N/2)-1 \end{aligned} \right\} \dots \dots \dots (2.20)$$

B. if N is odd

$$\left. \begin{aligned} L_i &= \frac{X_{2i} + X_{2i+1}}{\sqrt{2}}, i=0 \dots (N-1/2)-1 \\ H_i &= \frac{X_{2i} - X_{2i+1}}{\sqrt{2}}, i=0 \dots (N-1/2)-1 \\ L_{(N+1)/2} &= \sqrt{2} X_{(N-1)} \\ H_{(N+1)/2} &= 0 \end{aligned} \right\} \dots \dots \dots (2.21)$$

2.9.2 Inverse Haar Wavelet Transform (IHWT) [Jia03]

The inverse one-dimensional HWT is, simply, the inverse to the equations of FHWT; so, the IHWT equations are:

A. if N is even:

$$\left. \begin{aligned} X_{2i} &= \frac{L_i + H_i}{\sqrt{2}}, i=0 \dots (N/2)-1 \\ X_{2i+1} &= \frac{L_i - H_i}{\sqrt{2}}, i=0 \dots (N/2)-1 \end{aligned} \right\} \dots \dots \dots (2.22)$$

B. if N is odd

$$\left. \begin{aligned} X_{2i} &= \frac{L_i + H_i}{\sqrt{2}}, \quad i=0 \dots (N-1/2) \\ X_{2i+1} &= \frac{L_i - H_i}{\sqrt{2}}, \quad i=0 \dots (N-1/2) \\ X_{N-1} &= \sqrt{2} L_{(N+1)/2} \end{aligned} \right\} \dots \dots \dots (2.23)$$

2.10 Float wavelet Transform (Tap 9/7)

The biorthogonal filter (Tap 9/7) was chosen as the basic of the JPEG2000 lossy image compression standard for still images. The coefficients of this filter are given as floating-point numbers. The float filter primarily suited to high visual quality compression. [Mah05].

The (Tap 9/7) floating-point wavelet transform is computed by executing four "lifting" steps followed by two "scaling" steps on the extended pixel values P_k through P_m (a row of pixels in a tile denoted by P_k, P_{k+1}, \dots, P_m). Each step is performed over all the pixels in the tile before the next step starts, by using the following equations [Sal06]:-

$$C_{(2i+1)} = P_{(2i+1)} + \alpha [P_{(2i)} + P_{(2i+2)}], \quad k-3 \leq 2i+1 < m+3 \quad \dots \dots (2.24).$$

$$C_{(2i)} = P_{(2i)} + \beta [C_{(2i-1)} + C_{(2i+1)}], \quad k-2 \leq 2i < m+2 \quad \dots \dots (2.25).$$

$$C_{(2i+1)} = C_{(2i+1)} + \gamma [C_{(2i)} + C_{(2i+2)}], \quad k-1 \leq 2i+1 < m+1 \quad \dots \dots (2.26).$$

$$C_{(2i)} = C_{(2i)} + \delta [C_{(2i-1)} + C_{(2i+1)}], \quad k \leq 2i < m \quad \dots \dots (2.27).$$

$$C_{(2i+1)} = -K \times C_{(2i+1)}, \quad k \leq 2i+1 < m \quad \dots \dots (2.28).$$

$$C_{(2i)} = (1/K) \times C_{(2i)}, \quad k \leq 2i < m \quad \dots \dots (2.29).$$

Where, the five constants (wavelet filter coefficients) used by JPEG 2000 are given by $\alpha = -1.586134342$, $\beta = -0.052980118$, $\gamma = 0.882911075$, $\delta = 0.443506852$, and $K = 1.230174105$.

These one-dimensional wavelet transforms are applied L times, where L is a parameter (either user-controlled or set by the encoder), and are interleaved on rows and columns to form L levels (or resolutions) of subbands. Resolution $L - 1$ is the original image, and resolution 0 is the lowest-frequency subband, see Figure (2.6) [Sal06].

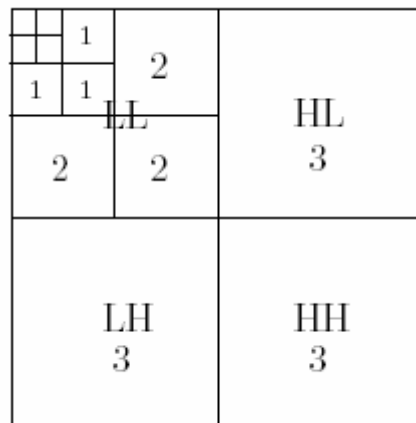


Figure (2.6) three levels decomposition of the Tap9/7 wavelet transform

2.11 Wavelet Transform Characteristics [Bur98] [Kha03]:

Wavelet transform have proven to be very efficient and effective in analyzing a very wide class of signals and phenomena. The reasons for that are;

1. The size of the wavelet expansion coefficients drop off rapidly with its indices for a large class of signals. This property is being called an unconditional basis and it is why wavelets are very effective in signal and image compression, denoising and detection.
2. The wavelet expansion allows a more accurate description and separation of signal characteristics. A wavelet expansion coefficient representation a component that is itself local and is easier to interpret. The wavelet expansion may allow a separation of components of a signal that overlap in both time and frequency.

3. Wavelets are adjustable and adaptable. Because there is not just one wavelet, they can be designed to fit individual applications. They are ideal for adaptive systems that adjust themselves to suit the signal.
4. The generation of wavelets and the calculation of the discrete wavelet transform are well matched to the digital computer. The defining equation for a wavelet uses no calculus. There are no derivatives integrals, just multiplication and additions operations that are basic to a digital computer.
5. Wavelet transform has a good energy compact, it is preserved across the transform, i.e. the sum of squares of wavelet coefficients is equal the sum of squares of the original image.
6. Wavelets can provide a good compression; it can perform better than JPEG2000, both in terms of Signal to Noise Ratio (SNR) and image quality. Thus show no blocking effect unlike JPEG2000.
7. The entire image is transformed and compressed as a single data object using wavelet transforms, rather than block by block. This allows for uniform distribution of compression error across the entire image and at all scales.
8. The wavelet transform methods have been shown to provide integrity at higher compression rates than other methods where integrity of data is important e.g., medical image and fingerprints, etc.
9. Multi resolution properties allow for progressive transmission and zooming, without extra storage.
10. It is a fast operation performance, in addition to symmetry: both the forward and inverse transform have the same complexity, in both compression and decompression phases.
11. Many image operations such as noise reduction and image scaling can be performed in wavelet transformed images.

2.12 Discrete Cosine Transform (DCT)

The DCT is a technique for converting a signal into elementary frequency components. It is a popular transform used by the JPEG image compression standard for lossy compression of images. Since it is used so frequently, DCT is often referred to in the literature as JPEG-DCT, which indicated that DCT is used in JPEG. JPEG-DCT is a transform coding method consists of four steps. The source image is first partitioned into sub-blocks of size 8×8 pixels in dimension. Then, each block is transformed from spatial domain to frequency domain using a 2-Dimension (2D) DCT basis function. The resulting frequency coefficients are quantized and finally output to a lossless entropy coder. DCT is an efficient image compression method since it can decorrelate pixels in the image (because the cosine bases are orthogonal) and compact most of the image energy into a few transformed coefficients. Moreover, DCT coefficients can be quantized according to some human visual characteristics. Therefore, the JPEG image file format is very efficient. This makes it very popular, especially in World Wide Web. However, in JPEG2000 the wavelet transform is used instead of DCT due to its better compression performance [Cab02, Tru99].

The forward DCT formula is given by [Sal02]:

$$C_{ij} = \frac{2}{\sqrt{mn}} C_i C_j \sum_{x=0}^{n-1} \sum_{y=0}^{m-1} P_{xy} \cos\left(\frac{(2y+1)j\pi}{2m}\right) \cos\left(\frac{(2x+1)i\pi}{2n}\right) \dots\dots\dots(2.30)$$

Where

$$C_f = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } f=0 \\ 1 & \text{otherwise} \end{cases} \dots\dots\dots(2.31)$$

C_{ij} represents the transform coefficients.

$0 \leq i < n$ and $0 \leq j < m$ are the indexes of the transform coefficients.

P_{xy} is the value of the pixel (x,y) .

n is the image width (number of columns).

m is the image height (number of rows).

To turn the image back to its original domain the inverse transform must be applied, the inverse DCT is given by:

$$P'_{xy} = \frac{2}{\sqrt{mn}} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} C_i C_j C_{ij} \cos\left(\frac{(2x+1)i\pi}{2n}\right) \cos\left(\frac{(2y+1)j\pi}{2m}\right) \dots\dots\dots (2.32)$$

Where

P' is the reconstructed image.

$0 \leq x < n$ and $0 \leq y < m$ are the image pixels coordinates.

C is the transformed image.

n, m is the number of pixels.

2.13 Set Partitioning In Hierarchical Trees (SPIHT)

Set Partitioning in Hierarchical Tree (SPIHT) is an image compression algorithm that exploits the inherent similarities across the subbands in a wavelet decomposition of an image [Sai96a].

SPIHT is a wavelet-based image compression coder. It first converts the image into its wavelet transform and then transmits information about the wavelet coefficients. The decoder uses the received Signal to reconstruct the wavelet and performs an inverse transform to recover the image. We selected SPIHT because SPIHT and its predecessor, the embedded zerotree wavelet coder, were significant breakthroughs in still image compression in that they offered significantly improved quality over vector quantization, JPEG, and wavelets combined with quantization,

while not requiring training and producing an embedded bit stream. SPIHT displays exceptional characteristics over several properties all at once including [Sai96b]:

- Good image quality with a high PSNR.
- Fast coding and decoding
- A fully progressive bit-stream
- Can be used for lossless compression
- May be combined with error protection
- Ability to code for exact bit rate or PSNR

SPIHT is a method of coding and decoding the wavelet transform of an image. By coding and transmitting information about the wavelet coefficients, it is possible for a decoder to perform an inverse transformation on the wavelet and reconstruct the original image. The entire wavelet transform does not need to be transmitted in order to recover the image. Instead, as the decoder receives more information about the original wavelet transform, the inverse-transformation will yield a better quality reconstruction (i.e. higher peak signal to noise ratio) of the original image. SPIHT generates excellent image quality and performance due to several properties of the coding algorithm. They are partial ordering by coefficient value, taking advantage of redundancies between different wavelet scales and transmitting data in bit plane order [Sai96b].

A Different wavelet filters produce different results depending on the image type, but it is currently not clear what is the best filter for any given image type. Regardless of the particular filter used, the image is decomposed into subbands, such that lower subbands correspond to higher image frequencies (they are the highpass levels) and higher subbands correspond to lower image frequencies (low pass levels), where most of the

image energy is concentrated (Figure 2.7.c). This is why we can expect the detail coefficients to get smaller as we move from high to low levels. Also, there are spatial similarities among the subbands (Figure 2.7.b). An image part, such as an edge, occupies the same spatial position in each subband. These features of the wavelet decomposition are exploited by the SPIHT method [Sai96a].

SPIHT is a coding method, so it can work with any wavelet transform; SPIHT was designed for optimal progressive transmission, as well as for compression. One of the important features of SPIHT is that at any point during the decoding of an image, the quality of the displayed image is the best that can be achieved for the number of bits input by the decoder up to that moment. Another important SPIHT feature is its use of embedded coding. This feature is defined as follows: If an (embedded coding) encoder produces two files, a large one of size M and a small one of size m , then the smaller file is identical to the first m bits of the larger file [Sal02].

The following example illustrates the meaning of this definition. Suppose that three users wait for you to send them a certain compressed image, but they need different image qualities. The first one needs the quality contained in a 10 Kb file. The image qualities required by the second and third users are contained in files of sizes 20 Kb and 50 Kb, respectively. Most lossy image compression methods would have to compress the same image three times, at different qualities, to generate three files with the right sizes. SPIHT produces one file, and then three chunks of lengths 10 Kb, 20 Kb, and 50 Kb, all starting at the beginning of that file can be sent to the three users, thereby satisfying their needs [Sal06].

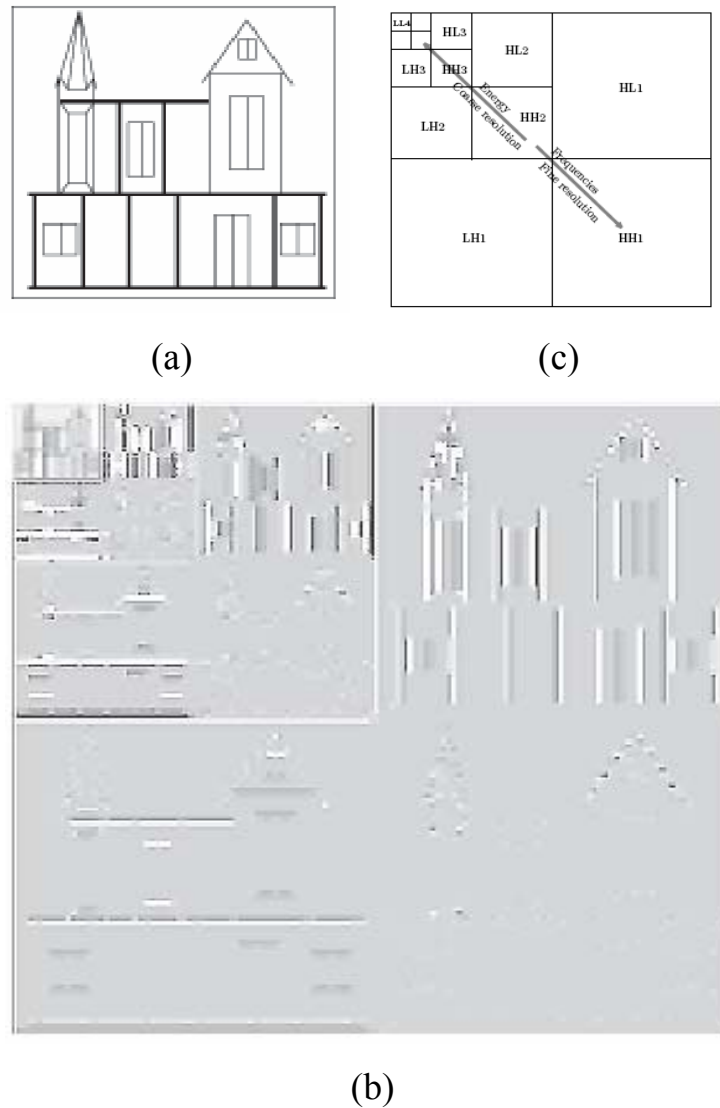


Figure (2.7) Subbands and Levels in Wavelet Decomposition

2.13.1 Set partitioning Sorting Algorithm

It's the algorithm that hierarchically divides coefficients into significant and insignificant, from the most significant bit to the least significant bit, by decreasing the threshold value at each hierarchical step for constructing a significance map. At each threshold value, the coding process consists of two passes: the sorting pass and the refinement pass-except for the first threshold that has only the sorting pass. Let $c_{(i,j)}$ represent the wavelet-transformed coefficients and m is an integer. The sorting pass involves selecting the coefficients such that $2^m \leq |c_{(i,j)}| \leq 2^{m+1}$, with m being decreased

at each pass. This process divides the coefficients into subsets and then tests each of these subsets for significant coefficients. The significance map constructed in the procedure is tree-encoded. The significant information is store in three ordered lists: List of Insignificant Pixels (LIP), List of Significant Pixels (LSP), and List of Insignificant Sets (LIS). At the end of each sorting pass, the LSP contains the coordinates of all significant coefficients with respect to the threshold at that step. The entries in the LIS can be one of two types: type A represents all its descendants; type B represents all its descendants from its grandchildren onward. The refinement pass involves transmitting the m^{th} most significant bit of all the coefficients with respect to the threshold, 2^{m+1} , the significance test can be summarized by [Shi00]:

$$S_m(T) = \begin{cases} 1, & \max_{(i,j) \in T} |c_{i,j}| \geq 2^m, \\ 0, & \text{otherwise.} \end{cases} \dots\dots\dots (2.33)$$

Where, T is a set of pixels.

2.13.2 Spatial Orientation Tree

The idea of a spatial orientation tree is based on the following observation. Normally, among the transformed coefficients most of the energy is concentrated in the low frequencies. For the wavelet transform, when we move from the highest to the lowest levels of the subband pyramid the energy usually decreases. It is also observed that there exists strong spatial self-similarity between subbands in the same spatial location such as in the zerotree case. Therefore, a spatial orientation tree structure has been proposed for the SPIHT algorithm. The spatial orientation tree

naturally defines the spatial relationship on the hierarchical pyramid as shown in (Figure 2.8) [Shi00].

During the coding, the wavelet-transformed coefficients are first organized into spatial orientation trees as in (Figure 2.8). In the spatial orientation tree, each pixel $c_{(i,j)}$ from the former set of subbands is seen as a root for the pixels $(2i, 2j)$, $(2i+1, 2j)$, $(2i, 2j+1)$, and $(2i+1, 2j+1)$ in the subbands of the current level. For a given n -level decomposition, this structure is used to link pixels of the adjacent subbands from level n until to level l . In the highest-level n , the pixels in the low-pass subband are linked to the pixels in the three high-pass subbands at the same level. In the subsequent levels, all the pixels of a subband are involved in the tree-forming process. Each pixel is linked to the pixels of the adjacent subband at the next lower level. The tree stops at the lowest level [Shi00].

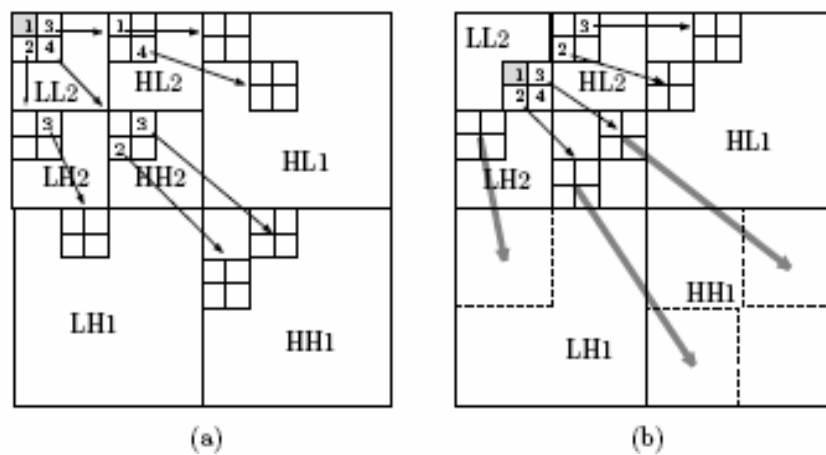


Figure (2.8) Spatial Orientation Trees in SPIHT [Sal98]

The terms offspring used for the four children of a node, and descendants for the children, grandchildren, and all their descendants.

The set partitioning sorting algorithm uses the following four sets of coordinates [Sal98]:

1. **O(i, j):** the set of coordinates of the four offspring of node (i, j). If node (i, j) is a leaf of a spatial orientation tree, then O(i, j) is empty.
2. **D(i, j):** the set of coordinates of the descendants of node (i, j).
3. **H(i, j):** the set of coordinates of the roots of all the spatial orientation trees (3/4 of the wavelet coefficients in the highest LL subband).
4. **L(i, j):** The difference set $D(i, j) - O(i, j)$. This set contains all the descendants of tree node (i, j), except its four offspring.

2.13.3 SPIHT Coding

The SPIHT coding algorithm uses three lists called list of significant pixels (LSP), list of insignificant pixels (LIP), and list of insignificant sets (LIS). These are lists of coordinates (i, j) that in the LIP and LSP represent individual coefficients, and in the LIS represent either the set D(i, j) (a type A entry) or the set L(i, j) (a type B entry).

The LIP contains coordinates of coefficients that were insignificant in the previous sorting pass. In the current pass they are tested, and those that test significant are moved to the LSP. In a similar way, sets in the LIS are tested in sequential order, and when a set is found to be significant, it is removed from the LIS and is partitioned.

The new subsets with more than one coefficient are placed back in the LIS, to be tested later, and the subsets with one element are tested and appended to the LIP or the LSP, depending on the results of the test. The refinement pass transmits the m^{th} most significant bit of the entries in the LSP [Sal02]. Figure (2.9) illustrates the sorting pass of SPIHT Algorithm.

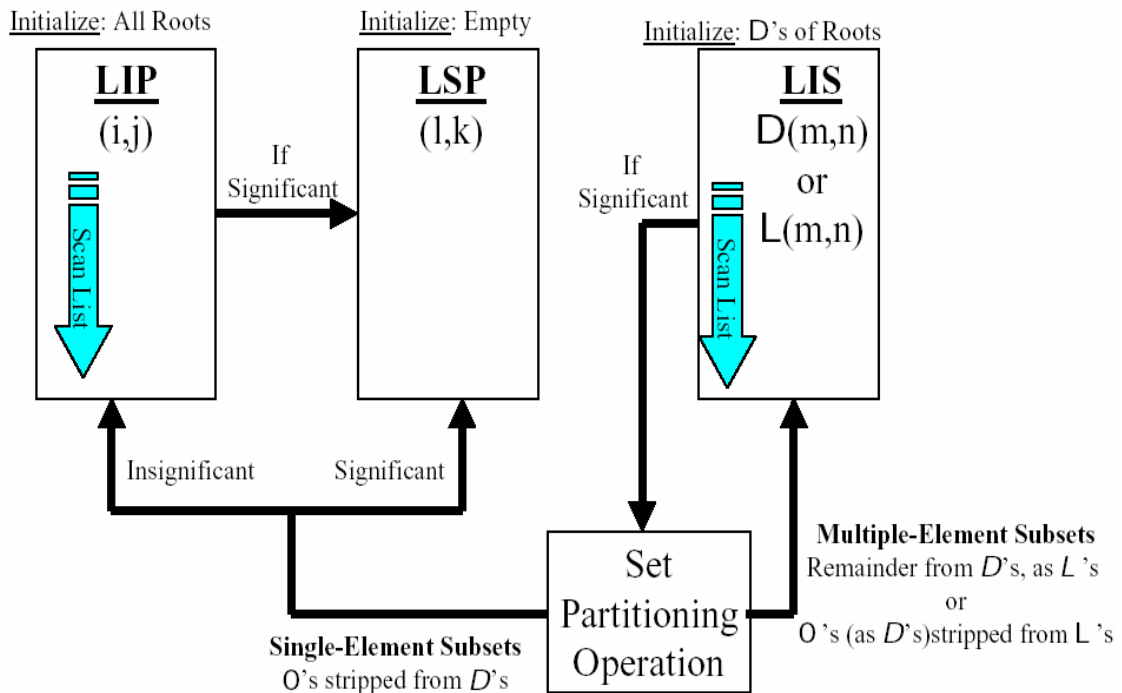


Figure (2.9) Sorting pass of SPIHT algorithm [Sai96a].

2.13.4 The Main Steps of the SPIHT Encoder:

Step (1) Initialization: Given an image to be compressed, perform its wavelet transform using any suitable wavelet filter, decompose it into transform coefficients $(C_{i,j})$, then find an integer $m = \lfloor \log_2(\max_{(i,j)} \{|C_{(i,j)}|\}) \rfloor$. Here $\lfloor \rfloor$ represent an operation of obtaining the largest integer less than $|c_{(i,j)}|$. The value of m is used for testing the significance of coefficients and constructing the significance map. The LIP is set as an empty list. The LIS is initialized to contain all the coefficients in the low-pass subbands that have descendants. These coefficients can be used as roots of spatial trees. All these coefficients are assigned to be of type A. The LIP is initialized to contain all the coefficients in the low-pass subbands [Sal06].

Step (2) Sorting pass: each entry of the LIP is tested for significance with respect to the threshold value 2^m . The significance map is transmitted in the following way. If it is significant, a “1” is transmitted, a sign bit of the coefficient is transmitted, and the coefficient coordinates are moved to the LSP. Otherwise, a “0” is transmitted. Then, each entry of the LIS is tested for finding the significant descendants. If there are none, a “0” is transmitted. If the entry has at least one significant descendant, then a “1” is transmitted and each of the immediate descendants are tested for significance. The significance map for the immediate descendants is transmitted in such a way that if it is significant, a “1” plus a sign bit are transmitted and the coefficient coordinates are appended to the LSP. If it is not significant, a “0” is transmitted and the coefficient coordinates are appended to the LIP. If the coefficient has more descendants, then it is moved to the end of the LIS as an entry of type B. If an entry in the LIS is of type B, then its descendants are tested for significance. If at least one of them is significant, then this entry is removed from the list, and its immediate descendants are appended to the end of the list of type A [Shi00].

Step (3) Refinement pass: the m^{th} most significant bit of the magnitude of each entry of the LSP is transmitted except those in the current sorting pass [Shi00].

Step (4) Iterate: m is decreased by 1 and the procedure is repeated from the sorting pass [Shi00].

The last iteration is normally performed for $m = 0$, but the encoder can stop earlier, in which case the least important image information (some of the least significant bits of all the wavelet coefficients) will not be transmitted. This is the natural lossy option of SPIHT. It is equivalent to scalar quantization, but it produces better results than what is usually achieved with scalar quantization, since the coefficients are transmitted in sorted order. An alternative is for the encoder to transmit the entire image (i.e., all the bits of all the wavelet coefficients) and the decoder can stop decoding when the reconstructed image reaches a certain quality. This quality can either be predetermined by the user or automatically determined by the decoder at run time [Sal06].

2.14 Fidelity Criteria

There are two types of the fidelity criteria:

1. **Objective Fidelity Criteria:** are borrowed from digital signal processing and information theory and provide us with equations that can be used to measure the amount of error in the reconstructed signal (image, sound or video).

The commonly used Objective Fidelity Criteria are the mean square error (MSE) and peak signal to noise ratio (PSNR):

The MSE is found by taking the summation of the square of the difference between the original and the reconstructed signal and finally divide it by the total number of samples as shown below:

$$\mathbf{MSE} = \frac{1}{size} \sum_{i=0}^{size-1} (R_i - O_i)^2 \dots\dots\dots (2.34)$$

Where

R= Reconstructed Image.

O= Original Image.

Size= number of image pixels.

The PSNR is based on the mean square error of the reconstructed image. The formula for PSNR is given as follow:

$$PSNR = 10 \log_{10} \left[\frac{(L-1)^2}{MSE} \right] \dots\dots\dots (2.35)$$

Where

L is the number of gray levels.

2. Subjective Fidelity Criteria

Subjective fidelity criteria depends on human evaluation, the evaluation can be classified into three categories [Gon02, Umb98]:

- i. Impairment test:** where the test subject scores the images in terms of how bad they are.
- ii. Quality test:** where the test subject rates the images in terms of how good they are.
- iii. Comparison test:** this test provides a relative measure; which is the easiest metric for most people to determine.

2.15 Performance Parameters

There are many ways to evaluate the compression methods, two of them are:

- i. Compression Ratio (CR):** it is a basic measure for the performance of any compression algorithm. It is the ratio of the original (uncompressed image file) to the compressed file, its denoted by:

$$CR = \frac{UncompressFileSize}{CompressedFileSiz} \dots\dots\dots (2.36)$$

- ii. Bit Rate (BR):** refers to the average number of bits required to represent the value of each image pixel, usually it is determined as the ratio between the size of compressed file and the size of original image file [Umb98]:

$$BR = \frac{compressionfilesize(inbits)}{imagefilesize(inpixels)} \dots\dots\dots (2.37)$$



Chapter Three

Chapter Three

*An Image compression
based on Modified
SPIHT Method*

CHAPTER THREE

An Image Compression Based on modified SPIHT Algorithm

3.1 Introduction

In this chapter, an image compression methods based on the modified SPIHT algorithm will be discuss and implement. The other techniques used in this work are the (Wavelet transform, DCT and RLE) methods. All of them show some advantages and disadvantages when they are applied on the image.

The implementation steps of the established image compression system are given. The data of the color components (R, G, and B) are transformed to (Y, C_b and C_r) components, to take the advantage of the existing spectral correlation and consequently to gain more compression. Also, the low spatial resolution characteristic of the human vision system to the chromatic components (C_b and C_r) is utilized to increase the compression ratio without making significant subjective distortions. Then each component will be decomposed by using the wavelet transform (Haar wavelet transform or Tap9/7 wavelet transform). In this way, most of the energy of an image was contained in the low-frequency bands, and most wavelet coefficients in the high-frequency bands have very low energy, so the quantization used to quantize many of these high-frequency wavelet coefficients to zero to reduce the number of bits needed to represent the coefficients of these bands, and the low-frequency wavelet coefficients are coded by using (DCT), and uniform quantization; then the modified SPIHT algorithm will be performed on the entire bands (low and high frequency). At the end, spatial coding steps like (Run Length Encoding (RLE), Shift Coding) were applied on List of Significant Pixels (LSP) to gain more

compression; and that led to good image quality with a high Peak Signal to Noise Ratio (PSNR), Compression Ratio (CR), fast coding and decoding.

3.2 System model

Digital image compression system consists of two units: the first unit called "Compression Unit", and the second one called "Decompression Unit", the first unit consists of many parts, as shown in figure (3.1).

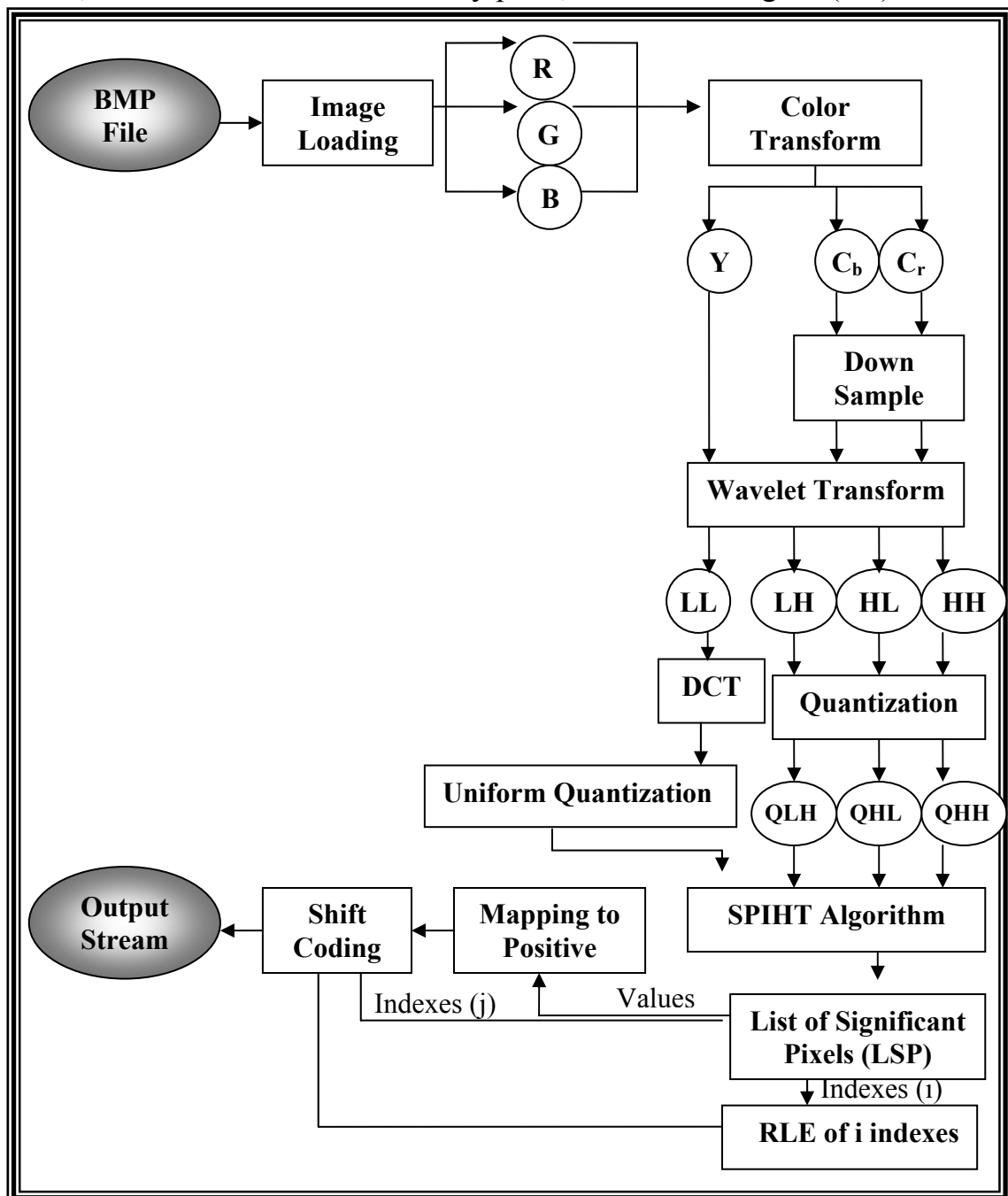


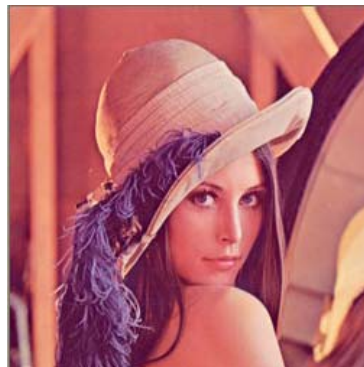
Figure (3.1) The Compression Unit of the proposed method

3.2.1 Compression Unit

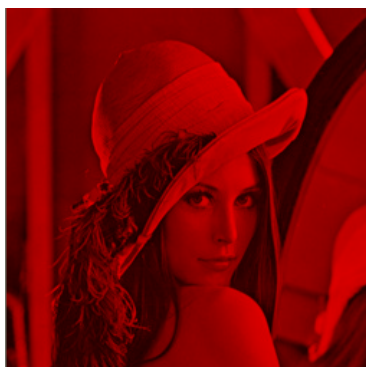
As shown in Figure (3.1), this unit consists of eleven operations which are all together responsible for reducing the data size of the desired color image, and generate compressed stream of data that represent the compressed image. In the following subsections, a functional description and implemented steps for each operation are given.

3.2.1.1 Image Loading

In this operation, the color image data is loaded and put it in one array of three records (R, G and B), with (H×W) size, where H denotes the height of the image, and W denotes its width. Figure (3.2) presents a typical RGB color image (256×256) with its three RGB color bands.



Original Image



Red



Green



Blue

Figure (3.2) Lena Image and its RGB components

3.2.1.2 Convert from RGB to YC_bC_r :

The following steps have been implemented on the input data (as image) given as two dimensional array ($H \times W$) to convert it from RGB to YC_bC_r space, by using equations (2.14).

Generally, YC_bC_r consist of a luminance component Y and two of the chrominance components C_b and C_r . The Y component consists of the luminance, black and white image information, while C_b represents the difference between R and Y, and C_r represents the difference between Y and B. in YC_bC_r space, most of the image information is in the Y components. This representation is used during JPEG compression. The JPEG compression grossly removes large portions of the C_b and C_r components without damaging the image. The JPEG algorithm uses compression to reduce the Y component since it has more effect on the quality of the compressed image.

Figure (3.3) shows the result of applying color transform operation on Lena image; Algorithm (3.1) shows the steps of implementing color transform algorithm.

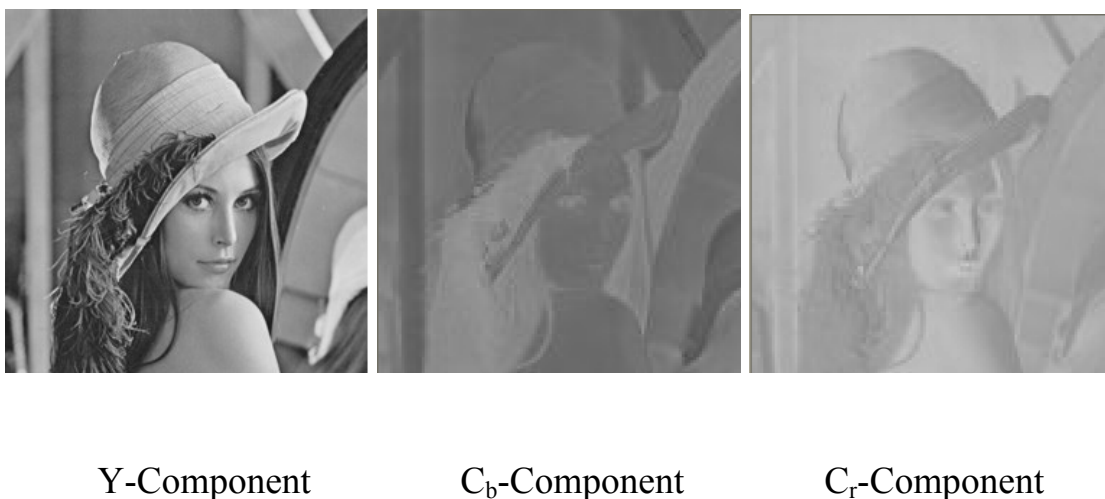


Figure (3.3) Lena Image and its YC_bC_r components

Algorithm (3.1) Convert from RGB to $Y C_b C_r$

Input: pic() = 2-D array of RGB image(W×H)

W=widthof image, H=high of image.

Output: YCC() = 2-D array of $Y C_b C_r$ image (W×H)

Method:

For each Column y (y=0,1, ..., H-1)

For each Row x (x=0,1, ..., W-1)

- **Compute Y band from pic(x,y)**

$$YCC.Y=0.2989 \times pic(x,y).R+0.5866 \times pic(x,y).G+0.1145 \times pic(x,y).B$$

- **Compute C_b band from pic(x,y)**

$$YCC.C_b=(-0.168 \times pic(x,y).R-0.33 \times pic(x,y).G+0.498 \times pic(x,y).B)+128$$

- **Compute C_r band from pic(x,y)**

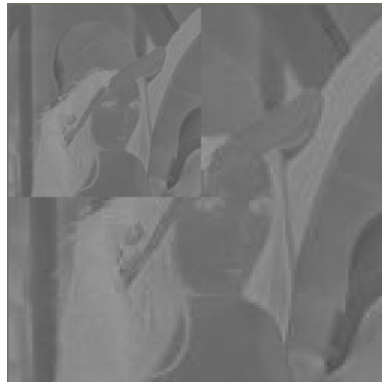
$$YCC.C_r=(0.498 \times pic(x,y).R-0.417 \times pic(x,y).G-0.081 \times pic(x,y).B)+128$$

End loop x

End loop y

3.2.1.3 Down Sample

In this operation, the C_b , C_r components have been down sampled by 2 to get an effective compression. The adopted down sampling method was the averaging method, where the average value of each (2×2) block is determined, and taken as a value represent that block in the down sampled image. Figure (3.4) shows the results of applying down sampling operation on the (C_b , C_r) color sub bands of Lena image; Algorithm (3.2) shows the steps of the implemented algorithm.



C_b -component & its down sample



C_r -component & its down sample

Figure (3.4) The Down Sample of C_b, C_r components of Lena image

Algorithm (3.2) Down Sampling

Input: $YCC()$ = 2-D array of $Y C_b C_r$ image ($W \times H$)

Output: $BndCb()$ = 2-D array($W/2 \times H/2$), $BndCr()$ = 2-D array($W/2 \times H/2$),

Method:

$W_m = W/2$: $H_m = H/2$: $y_1 = 0$: $y_2 = 1$

For each Column y ($y=0, 1, \dots, H_m-1$)

$x_1 = 0$: $x_2 = 1$

For each Row x ($x=0, 1, \dots, W_m-1$)

• **Down Sample the C_b Components**

$BndCb(x, y) = (YCC.Cb(x_1, y_1) + YCC.Cb(x_2, y_1) + YCC.Cb(x_1, y_2) + YCC.Cb(x_2, y_2)) / 4$

• **Down Sample the C_r Components**

$BndCr(x, y) = (YCC.Cr(x_1, y_1) + YCC.Cr(x_2, y_1) + YCC.Cr(x_1, y_2) + YCC.Cr(x_2, y_2)) / 4$

$x_1 = x_1 + 2$: $x_2 = x_2 + 2$

End loop x

$y_1 = y_1 + 2$: $y_2 = y_2 + 2$

End loop y

3.2.1.4 Wavelet Transform

The first step of encoding stage was applying the Haar wavelet transform on Y , C_b and C_r components separately by using equations (2.20, 2.21). The input for this algorithm was 2D array of size $(W \times H)$, and the output was the wavelet coefficients in 4-sub bands (LL, LH, HL and HH) which represented also in 2D array. In this way, the image data is decomposed into sub bands, each hold certain kind of image information; such that most of the image information energy is concentrated in the lowest frequency sub band (i.e., LL sub band). Algorithm (3.3) illustrates the implemented steps of the forward Haar wavelet transform, and then Tap9/7 wavelet transform was applied instead of Haar wavelet transform to find the best wavelet transform. At this step, Haar wavelet gives best PSNR than Tap9/7 but after applying the quantization; Tap 9/7 led to high (PSNR), so Tap9/7 wavelet transform was used at the reset of this research. Figure (3.5) shows the results of applying Tap9/7 algorithm on (Y , C_b and C_r) sub bands of Lena image. Algorithm (3.4 a, b) illustrates the steps of Tap9/7 wavelet transform,

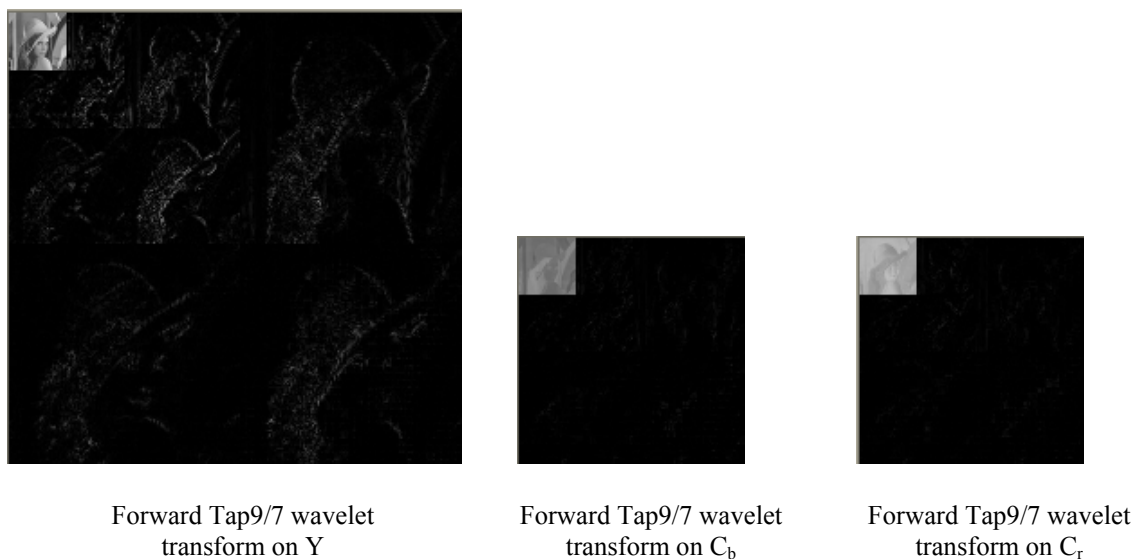


Figure (3.5) Forward Tap9/7 Wavelet transform applied on (Y, C_b and C_r) components of Lena image.

Algorithm (3.3) Haar Wavelet Transform

Input:

Bnd()= 2-D array (Y or C_b or Cr)

W=Width of Bnd : H=hight of Bnd

Nopass =the number of wavelet transform levels or pass.

Output:

BndW() = 2-D array consist at least four subbands (LL,LH,HL and HH).

Method:

For all I where $1 \leq I \leq \text{Nopass}$

Hh=H div 2 : Wh=W div 2

For each Column y (y=0,1....Hh-1) : y1 =2*y :y2 =y1+1

For each Row x (x=0,1....Wh-1) : x1 =2*x : x2 = x1+1

BndW(x,y)=(Bnd(x1,y1)+ Bnd(x2,y1)+ Bnd(x1,y2)+ Bnd(x2,y2))/2

BndW(x+Wh,y)=(Bnd(x1,y1)+ Bnd(x2,y1)- Bnd(x1,y2)-Bnd(x2,y2))/2

BndW(x,y+Hh)=(Bnd(x1,y1)- Bnd(x2,y1)+ Bnd(x1,y2)- Bnd(x2,y2))/2

BndW(x+Wh,y+Hh)=(Bnd(x1,y1)-Bnd(x2,y1)-Bnd(x1,y2)+ Bnd(x2,y2))/2

End loop x

End loop y

End loop I

Algorithm (3.4 a) Tap9/7 wavelet transform

Input:

Bnd()= 2-D array (Y or C_b or Cr)

W=Width of Bnd

H=hight of Bnd

Nopass =the number of wavelet transform levels or pass.

Output:

BndW() = 2-D array consist at least four subbands (LL, LH, HL and HH).



Method:

W1 = W: H1 = H

For all I where (1 ≤ I ≤ Nopass)

Wmm = W1 - 1: Hmm = H1 - 1

For each column (y = 0,1,..., Hmm)

For each row (x = 0,1,...,Wmm)

A(x) = Bnd(x,y)

End loop x

Tap97Forward W1, A(), B()algorithm (3.4b).

For each row (x = 0,1,...,Wmm)

Bnd(x,y) = B(x)

End loop x

End loop y

For each row (x = 0,1,...,Wmm)

For each column (y = 0,1,..., Hmm)

A(y) = Bnd(x, y)

End loop y

Tap97Forward H1, A(), B().....algorithm (3.4b).

For each column (y = 0,1,..., Hmm)

Bnd(x, y) = B(y)

End loop y

End loop x

W1 = W1 \ 2: H1 = H1 \ 2

End loop I

For each y (y=0,...,H)

For each x (x=0,..., W)

Bndw(x,y)=Bnd(x,y)

End loop x

End loop y

Algorithm (3.4 b) Tap9/7Forward

Input:

N= the length of Bnd (height or width).

A () = vector consist one row or column.

Output:

B() =vector consist the changed row or column.

Method:

$$\text{Neven} = (N + 1) \setminus 2: \text{Nodd} = N - \text{Neven}$$

$$\text{Nm} = N - 1: \text{Np1} = N + 1: \text{Np2} = N + 2$$

For all K where (K = 0, ... ,Nm)

$$C(K) = A(K)$$

End loop K

For all K where (K = 1, ..., 4)

$$C(-K) = C(K)$$

End loop K

For all K where (K = 1... 4)

$$C(\text{Nm} + K) = C(\text{Nm} - K)$$

End loop K

For all K where (K = -3... Np2) (Step 2)

$$C(K) = C(K) - 1.586134342 * (C(K - 1) + C(K + 1)) \dots \text{eq.}(2.24)$$

End loop K

For all K where (K = -2... Np1) (Step 2)

$$C(K) = C(K) - 0.05298011854 * (C(K - 1) + C(K + 1)) \dots \text{eq.}(2.25)$$

End loop K

For all K where (K = -1... N) (Step 2)

$$C(K) = C(K) + 0.8829110762 * (C(K - 1) + C(K + 1)) \dots \text{eq.}(2.26)$$

End loop K

For all K where (K = 0... Nm) (Step 2)

$$C(K) = C(K) + 0.4435068522 * (C(K - 1) + C(K + 1)) \dots \text{eq.}(2.27)$$

End loop K



```

aa = 1 / 1.230174105 : I = 0 : J = 1
For all K where (K = 0,..., Neven - 1)
    B(K) = C(I) * aa .....eq(2.29)
    I = I + 2
End loop K
For all K where (K = 0,...,Nodd - 1)
    B(K + Neven) = C(J) * 1.230174105 .....eq(2.28)
    J = J + 2
End loop K

```

3.2.1.5 Hierarchical Uniform Quantization

It refers to the process of approximating the continuous set of values in the image data with a finite (small) set of values.

The uniform quantization was used to code the transformed wavelet coefficients sub bands (LH, HL, and HH) to achieve better compression result. As well as that is used to reduce the number of bits needed to store the coefficients of these sub bands [Kum03].

In this work a quantization function was used to determined the quantization step (Q_{step}) for each coefficients $BndW(x,y)$ as follow:

$$BndQ(x,y) = \text{round} \frac{BndW(x,y)}{Q_{step}} \dots\dots\dots(3.1)$$

Where,

$BndW ()$ is the array of wavelet transform coefficients.

$BndQ ()$ is the quantization index array.

Q_{step} for LH and HL sub band is:

$$Q_{step} = Q\alpha^{n-1} \dots\dots\dots(3.2)$$

While the Q_{step} for HH sub band is:

$$Q_{step} = Q\beta\alpha^{n-1} \dots\dots\dots(3.3)$$

Where, n is the wavelet level number (i.e., the pass number) (Q, α, β) are quantization parameters (such that, $Q \geq 1, \alpha \leq 1, \beta \geq 1$). Algorithm (3.5) illustrates the steps of the applied uniform quantization method.

Algorithm (3.5) Hierarchical uniform Quantization

Input:

Bndw()= 2-D array of the wavelet transform coefficients (Y, C_b and C_r).

W=Width of BndW : H=hight of BndW

Nopass =the number of wavelet transform levels or pass.

Q = the initial quantization step for (LH,HL and HH).

α, β = the quality numbers.

Output:

BndQ() = 2-D array of the quantization indices.

Method:

We =W :He =H

For all i where $1 \leq i \leq$ Nopass

Wm=We div 2 : Hm=He div 2

Qstep = $Q * \alpha^{i-1}$: QstepHH = $Q * \alpha^{i-1} * \beta$

• **Quantiz LHi**

For all x,y where (y=0 ... Hm-1) and (x=Wm ... We-1)

BndQ(x,y) = round (BndW(x,y) / Qstep)

End loop x,y

• **Quantiz HLi.....**

For all x,y where (y=Hm ... He-1) and (x= 0 ... Wm-1)

BndQ(x,y) = round (BndW(x,y) / Qstep)

End loop x,y

• **Quantiz HHi**

For all x,y where (y=Hm ... He-1) and (x=Wm....We-1)

BndQ(x,y) = round (BndW(x,y) / QstepHH)

End loop x,y

We=Wm : He=Hm

End loop i

3.2.1.6 Discrete Cosine Transform (DCT)

By using the DCT transform described by equation (2.30), each (8*8) block of pixels in the low-low subband of the wavelet transformed image will be transformed from the original time domain to the frequency domain. The following algorithm illustrates the steps of the DCT transformation operation.

Algorithm (3.6) Discrete Cosine Transform (DCT)

Input:

Bndw()= 2-D array of the wavelet transform coefficients (Y, C_b and C_r).

W=Width of BndW : H=high of BndW

Nopass = the number of wavelet transform levels or pass.

BlkSize = is the block size.

Output:

C() = is the transformed block of an (8*8) array of DCT coefficients.

Method:

BlkSize =8: sum =0

C0 = 1/sqr (2) : C2 = 1/sqr (2*BlkSize)

C1 = 3.14159/(2*B lkSize)

For all I where 0 ≤ I ≤ BlkSize-1

For all J where 0 ≤ J ≤ BlkSize-1

For all Y where 0 ≤ Y ≤ BlkSize-1

For all X where 0 ≤ X ≤ BlkSize-1

W1 = (2*Y+1)*J*C1

W2 = (2*X+1)*I*C1

Sum = Sum + Bndw(X,Y)*Cos(W1)*Cos(W2).....eq(2.29)

If I =0 then Ci =C0 else Ci = 1

If J =0 then Cj = C0 else Cj =1

C(I,J) = C2*Ci*Cj*Sum

End loop X

End loop Y

End loop J

End loop I

3.2.1.7 Uniform Quantization of the DCT Coefficients

To reduce the data required to represent the image, the first step is the quantization of the image's DCT coefficients; while the second step is coding the quantized coefficients.

In this operation, the uniform quantization was applied on the AC-transform coefficient, using the equation (3.1), and the Q_{step} of the $(x,y)^{th}$ AC-coefficients, will be determined by the following equation:

$$Q_{step}(x,y) = Q_{low}(1+\gamma(x+y-1)) \dots\dots\dots(3.4)$$

Where,

Q_{low} is the lowest quantization step value for AC-coefficients.

γ is the incrimination rate.

(x,y) are the frequency indices, whose values are within the range $[0,\dots,Block\ length-1]$.

The way of applying the uniform quantization on DCT coefficients is illustrated in algorithm (3.7).

Algorithm (3.7) Uniform Quantization of the DCT Coefficients

Input:
 $C()$ = is the transformed block of an $(8*8)$ array of DCT coefficients.
 W =Width of BndW : H =height of BndW
 Q_{low} = the quantization step of AC-coefficients
 Q_{Dc} = the quantization step of DC-coefficient
 γ = the quantization parameter

Output:
 $Q_{Dc}()$ = the array of quantization of DCT coefficients

Method:
For all I where $0 \leq I \leq BlkSize-1$
 For all J where $0 \leq J \leq BlkSize-1$
 If $I=0$ and $J=0$ then
 $Q_{Dc}(0,0) = round(C(0,0)/Q_{Dc})$ ➔

```

Else
    Qstep = Qlow * (1+γ*(I+J))
    QDc(I,J) = round (C(I,J)/Qstep)
End if
End loop J
End loop I

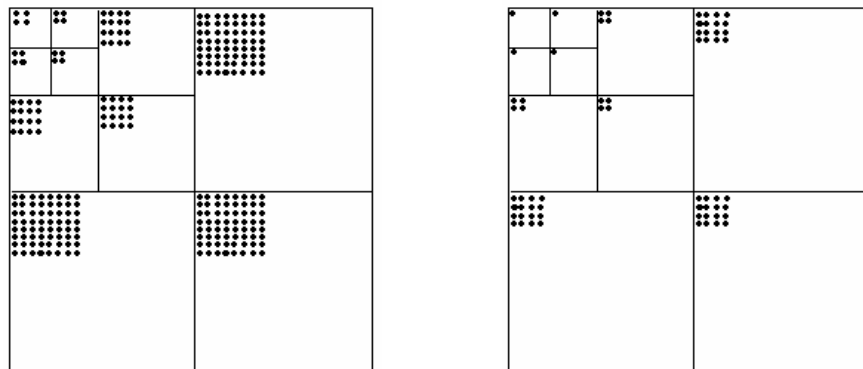
```

3.2.1.8 The modified SPIHT Coding

It is important to have the encoder and decoder test sets for significance in the same way. The coding algorithm therefore uses three lists called *List of Significant Pixels* (LSP), *List of Insignificant Pixels* (LIP), and *List of Insignificant Sets* (LIS). These are lists of coordinates (i, j) that in the LIP and LSP represent individual coefficients, and in the LIS represents either the set $D(i, j)$ (a type *A* entry) or the set $L(i, j)$ (a type *B* entry). The LIP contains coordinates of coefficients that were insignificant in the previous sorting pass. In the current pass they are tested, and those that test significant are moved to the LSP. In a similar way, sets in the LIS are tested in sequential order, and when a set is found to be significant, it is removed from the LIS and is partitioned. The new subsets with more than one coefficient are placed back in the LIS, to be tested later, and the subsets with one element are tested and appended to the LIP or the LSP, depending on the results of the test. The refinement pass transmits the n th most significant bit of the entries in the LSP [Sal02].

In the modified SPIHT coding, 4 unadjacent pixels are taken to build the spatial orientation tree. In this way, the number of roots will be increased and the number of levels will be decreased by 1 (LL will represent the roots only and its children will be in the LH, HL and HH) and building trees will be more efficient than before (in the original SPIHT

algorithm that was taking 4 adjacent pixels). Figure (3.6 a,b) illustrates the spatial orientation tree of the original and the proposed SPIHT algorithm. Algorithm (3.8) illustrates the steps of the modified SPIHT coding method.



a. Original SPIHT

b. Modified SPIHT

Figure (3.6) Spatial Orientation Trees in SPIHT

Algorithm (3.8) Modified SPIHT Coding

Input:

BndQ()= 2-D array of the wavelet transform coefficients (Y, C_b and C_r) after Quantization.

W=Width of BndQ : H=hight of BndQ

Output:

LSP() = 1-D array of record (v,i,j) represents the List of significant pixels (the value and its position).

LSPindex = length of LSP vector

Method:

• **Initialization :**

O(i,j): set of coordinates of all offspring of node (i,j)(children only)(unadjacent pixels).

D(i,j): set of coordinates of all descendants of node (i,j) (children, grandchildren, great-grand, etc.).

H (i,j): set of all tree roots (nodes in the highest pyramid level)(parents).

L (i,j): D(i,j) – O(i,j) (all descendents except the offspring)(grandchildren, great-grand, etc).

n = Log₂ (max |BndQ(i,j)|).

LIP = All elements in H ()

LSP = Empty



```

LIS = D's of Roots
LSPindex = 1
• Sorting Pass:
Do while (n>1) // sorting pass of LIP
for each coeff (i,j) in LIP
    Output Sn(i,j) // equation (2.18)
    If Sn(i,j)=1 then
        Move (i,j) to the LSP record : LSPindex = LSPindex + 1
    End if
End loop over LIP
For each set (i,j) in LIS // sorting pass of LIS
// Type D
    If type D then check Sn(D(i,j))
        If Sn(D(i,j))=1 then
            for (k=1... 4) ∈ O(i,j)
                output Sn(k,l)
                if Sn(k,l)=1, then add (k,l) to the LSP : LSPindex = LSPindex + 1
                if Sn(k,l)=0, then add (k,l) to the end of the LIP
            end for
        end if
// Type L
    Else
        check Sn( L(i,j))
        If Sn(L(i,j))=1 then add each (k=1 ... 4) ∈ O(i,j) to the end of the LIS as an
            entry of type D, and remove (i,j) from the LIS
        End if
End loop over LIS
• Refinement Pass:
    For each element (i,j) in LSP – except those just added above
        Output the nth most significant bit of coefficients.
    End loop over LSP
• Update
    Decrement n by 1
    Go to sorting pass
End loop n

```

3.2.1.9 Mapping to Positive

Since the produced numbers from the previous step are a combination of negative and positive integers, so for easily manipulating them a mapping to positive integers will be performed to convert the signed integer into positive integers:

$$X' = \begin{cases} 2X & \text{if } X \geq 0 \\ 2|X|+1 & \text{if } X < 0 \end{cases} \dots\dots\dots (3.5)$$

Where, X represents the signed integer value of the LSP. This type of mapping insures that all coefficients values are mapped to positive integers, and to keep the optimal number of bits needed to be used by shift coder as small as possible, see Algorithm (3.9).

Algorithm (3.9) Mapping to Positive

Input:
 Z()= 1-D array indicates the value of LSP .
 Zn = length of LSP.

Output:
 Positive LSP indices.

Method:
For all I (I = 1 to Zn)
 If Z(I) >=0 then
 Z(I) = 2 * Z(I)
 Else
 Z(I) = 2 * Abs(Z(I)) + 1
 End if
End loop I

3.2.1.10 Run Length Encoding (RLE)

RLE consists of the process of searching for repeated runs of symbols in an input stream, and replacing them by a run count. In this work, the indexes of LSP values will contain repeated values for all i indexes, so it is best to encode them by RLE to reduce the space needed to store the compressed file, see algorithm (3.10).

Algorithm (3.10) RLE

Input:

$Z()$ = 1-D array indicates the i indexes of LSP .

Z_n = length of LSP.

Output:

$A()$ = 1-D array indicates the encoded i indexes of LSP .

A_n = new length.

Method:

Temp = $Z(1)$: Counter = 1

$A(1)$ = $Z(1)$: $A_n = 2$

For all I (I = 2 to Z_n)

If $Z(I)$ = temp then

Counter = Counter + 1

Else

$A(A_n)$ = Counter : $A_n = A_n + 1$

$A(A_n)$ = $Z(I)$: $A_n = A_n + 1$

temp = $Z(I)$: Counter = 1

End if

End loop I

$A(A_n)$ = Counter

3.2.1.11 S-Shift Optimizer

The mechanism applied to compute the optimal length "N", in bits of the shift codeword is based on scanning all possible codeword lengths, starting from "1" bit and proceeding more till " N_{max} " bits; which represents the minimum number of bits required to represent the maximum coefficient value "Max" in the LSP parameters. This number " N_{max} " is considered as the length "in bits" of the second "auxiliary" codeword. The scan method was applied to test all possible values of bits that can be assigned to the first "shortest" codeword, so the length range of the first codeword is $[1, N_{max}]$.

Algorithm (3.11) shows the steps of computing the optimal length of the shift codeword.

Algorithm (3.11) S-Shift Optimizer

Input:

Z()= 1-D array indicates the value of LSP .

Zn = length of LSP.

Output:

Number of required bits (N, N_{max}).

Method:

//Find the maximum value of LSP

Max=Z(1)

For all I (I = 2 to Zn)

If Max < Z(I) then Max =Z(I)

End loop I

//Compute the number of bits required to represent the Max value.

$N_{max} = 1 : K = 1$

While Max >= K

$K = 2 * K + 1$

$N_{max} = N_{max} + 1$

End while



```

//Shift coding optimizer to compute the total number of required bits (MinBits).
MinBits = Nmax * Zn
For all N (1 ≤ N ≤ Nmax - 1)
    R = 2 ^ N - 1 : Sm = 0
    For all J (0 ≤ J ≤ Zn)
        If Z(J) < R then
            Sm = Sm + N
        Else
            Sm = Sm + (N + Nmax)
        End if
    End loop J
    If Sm < MinBits then MinBits = Sm
End loop N

```

3.2.1.12 Shift Coding


In this operation, the codeword produced by applying s-shift optimizer are sent to the compression bitstream (which represents the compressed data file), see algorithm (3.12).

Algorithm (3.12) Shift Coding

Input:
 Z()= 1-D array indicates the value of LSP .
 Zn = length of LSP.
 N = the length of the first shift codeword
 N_{max} = the largest number of bits required to represent the largest number.

Output:
 Compressed file.

Method:
 R = 2 ^ N - 1
For all I (1 ≤ I ≤ Zn)
If Z(I) < R then
Putword (Z(I), N) // Save the value of Z(I) as an integer has length N bits.



```

Else
    Putword (R, N) // Save the value of Max as an integer has length N bits.
    Putword (Z(I)-R, Nmax) // Save the value of (Z(I)-R) as an integer has length Nmax
    bits.
End if
End loop I
    
```

3.2.2 Decompression Unit

This unit consists of nine operations as shown in figure (3.7) which starts with loading the compressed data and ends with the reconstructed image. In the following subsections, the implemented steps for each operation are given.

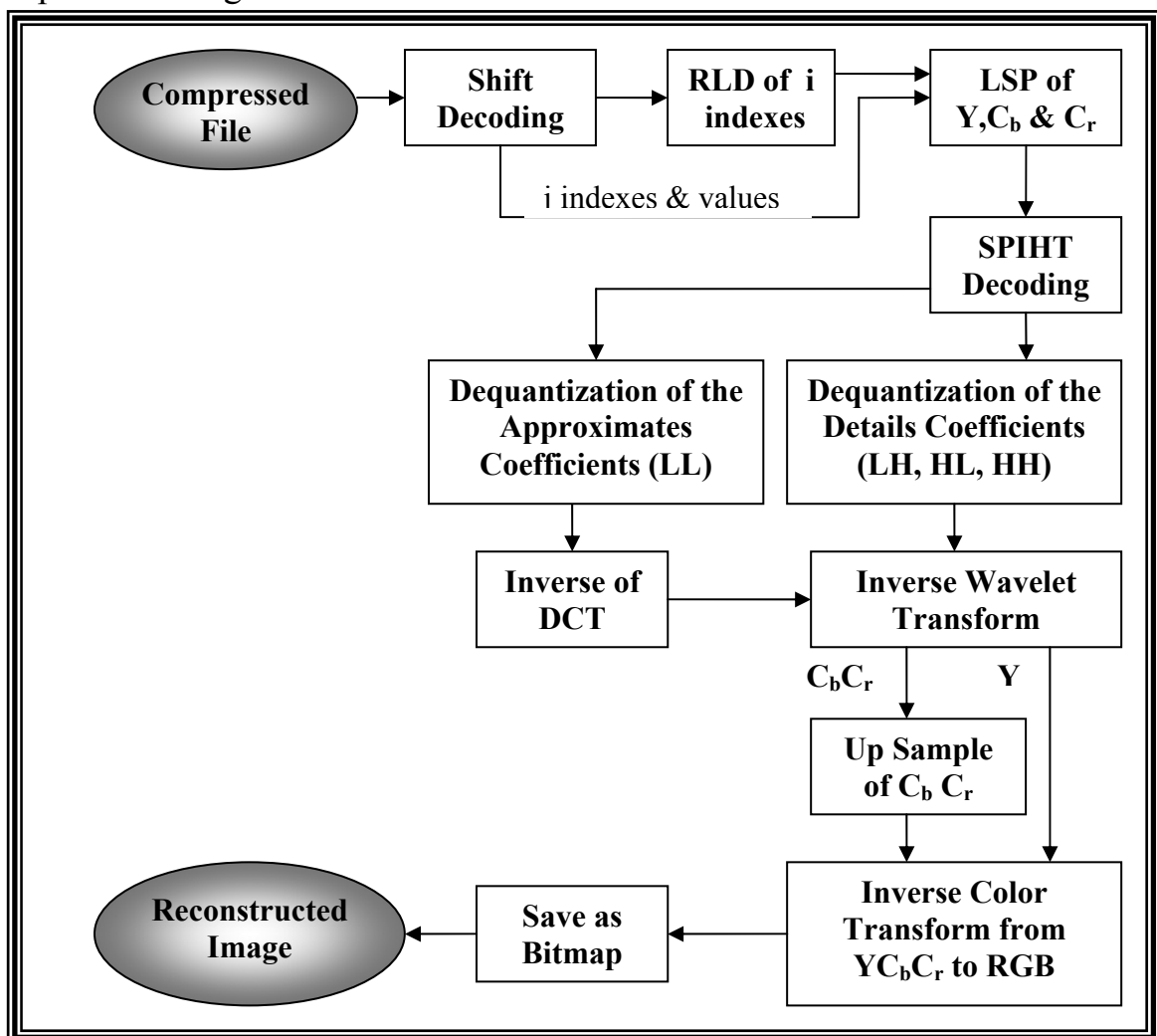


Figure (3.7) The Decompression Unit of the proposed system

3.2.2.1 Shift Decoding

Algorithm (3.13) illustrates the steps of the shift decoder.

Algorithm (3.13) Shift Decoding

Input:

N = the number of bits of the first shift codeword

N_{\max} = the number of bits of the second shift codeword

Output:

Z () = 1-D array of represents the (value or i or j) of list of significant pixels of the (Y, C_b , or C_r) components.

Z_n = length of Z

Method:

"" Shift Decoding

$R = 2^N - 1$

For all I (1 ≤ I ≤ Z_n)

E = Getword (N) // load N bits from the compression stream.

If $E < R$ then

$Z(I) = E$

Else

$Z(I) = E + \text{Getword}(N_{\max})$

End if

End loop I

"" Mapping the values of LSP from positive to actual values

For all I (1 ≤ I ≤ Z_n)

If $Z(I) < 0$ then

If $(Z(I) \text{ and } 1) = 0$ then

$LSP(I) = Z(I) \setminus 2$

Else

$LSP(I) = -(Z(I)+1) \setminus 2$

End if

Else

$LSP(I) = Z(I)$

End if

End loop I

3.2.2.2 Run Length Decoding

Algorithm (3.14) illustrates the steps that perform the run length decoding.

Algorithm (3.14) Run Length Decoding

Input:

Z()= 1-D array indicates the i indexes of LSP .

Zn = length of LSP.

Output:

A()=1-D array indicates the decoded i indexes of LSP .

An = new length.

Method:

An =1

For all I (I = 2 to Zn) step 2

Temp = Z(I)

Counter = Z(I+1)

For all J (J =1 to Counter)

A(An) = temp

An =An+1

End loop J

End loop I

An= An -1

3.2.2.3 SPIHT Decoding

In this operation, the two dimensional arrays for each component of (Y,C_b and C_r) will be reconstructed. Algorithm (3.15) illustrates how the steps of SPIHT decoding were implemented.

Algorithm (3.15) SPIHT Decoding

Input:

LSP () = 1-D array of record (v, i, j) represents the list of significant pixels of the (Y, C_b, or C_r) components.

LSPindex = length of LSP vector.

W=Width of reconstructed image: H=hight of reconstructed image

Output:

C() =2-D (W*H) array represents the (Y, C_b or C_r)components.

Method:

If (W=256) and (H=256) then

For all k (1 ≤ k ≤ LSPindex)

C (LSP (k).i , LSP(k).j) = LSP(k).v

End loop k

Else If (W=128) and (H=128) then

For all k (1 ≤ k ≤ LSPindex)

If (LSP (k).i ≤ 128) and (LSP (k).j ≤ 128) Then

C (LSP (k).i , LSP(k).j) = LSP(k).v

End loop k

Else If (W=64) and (H=64) then

For all k (1 ≤ k ≤ LSPindex)

If (LSP (k).i ≤ 64) and (LSP (k).j ≤ 64) Then

C (LSP (k).i , LSP(k).j) = LSP(k).v

End loop k

End if

3.2.2.4 DeQuantization

Algorithm (3.16) illustrates the steps used to perform the dequantization on the details coefficients (LH, HL and HH) of (Y, C_b and C_r) components, the dequantized sub bands can be calculated by multiplying the Qstep with the sub band matrix as the following equation:

$$C'(x,y) = Q_{\text{step}} \times C(x,y) \dots\dots\dots(3.6)$$

Where the Qstep explained in equations (3.2), (3.3).

Algorithm (3.16) DeQuantization

Input:

C() =2-D (W*H) array represents the (Y, C_b or C_r) components

W=Width of C

H=Height of C

Q = the initial quantization step for (LH,HL and HH).

α, β = the quantization parameters.

Output:

DC() = 2-D array of the dequant wavelet coefficients.

Method:

We =W :He =H

For all i where $1 \leq i \leq \text{Nopass}$

Wm=We div 2 : Hm=He div 2

Qstep = $Q * \alpha^{i-1}$: QstepHH = $Q * \alpha^{i-1} * \beta$

• ***Quantiz LHi.....***

For all x,y where (y=0 ... Hm-1) and (x=Wm ... We-1)

DC(x,y) =C(x,y) \times Qstep

End loop x,y

• ***Quantiz HLi.....***

For all x,y where (y=Hm ... He-1) and (x= 0 ...Wm-1)

DC(x,y) =C(x,y) \times Qstep

End loop x,y

• ***Quantiz HHi***

For all x,y where (y=Hm ... He-1) and (x=Wm....We-1)

DC(x,y) =C(x,y) \times QstepHH

End loop x,y

We=Wm : He=Hm

End loop i

3.2.2.5 DeQuantization of the DCT coefficients

The step of algorithm (3.17) illustrates the dequantization of DCT coefficients on LL-subband.

Algorithm (3.17) DeQuantization

Input:

$Q()$ = is the transformed block of an (8*8) array of quantization indices of DCT coefficients.

Q_{low} = is the lowest value of quantization step of AC-coefficients

Q_{Dc} = the quantization step of DC-coefficient

γ = the quantization parameter

Output:

$D()$ = the array of Dequantized DCT coefficients

Method:

For all y where $0 \leq y \leq \text{BlkSize}$

For all x where $0 \leq x \leq \text{BlkSize}$

For all I where $0 \leq I \leq \text{BlkSize}$

For all J where $0 \leq J \leq \text{BlkSize}$

If $I=0$ and $J=0$ then

$$D(I,J) = Q_{Dc} * Q(I,J)$$

Else

$$Qq = Q_{low} * (1 + \gamma * (I + J))$$

$$D(I,J) = Qq * Q(I,J)$$

End if

End loop J

End loop I

End loop x

End loop y

3.2.2.6 Inverse Discrete Cosine Transform (IDCT)

The inverse transform turns the quantized DCT coefficients to LL-subband coefficients. Algorithm (3.18) illustrates the steps to perform the inverse DCT.

Algorithm (3.18) IDCT

Input:

D() = dequantized DCT coefficients for LL subband.

Output:

B() = array of quantization indices for LL subband.

Method:

BlkSize = 8 :Sm =0

C0 = 1/sqr (2)

C2 = 1/sqr (2*BlkSize)

C1 = 3.14159/(2*B lkSize)

For all I where $0 \leq I \leq \text{BlkSize}-1$

For all J where $0 \leq J \leq \text{BlkSize}-1$

For all Y where $0 \leq Y \leq \text{BlkSize}-1$

For all X where $0 \leq X \leq \text{BlkSize}-1$

 W1 = (2*Y+1)*J*C1

 W2 = (2*X+1)*I*C1

 Sm = Sm + D(I,J)*Cos(W1)*Cos(W2)

 If I =0 then Ci =C0 else Ci = 1

 If J =0 then Cj = C0 else Cj =1

 B(X,Y) = C2*Ci*Cj*Sm

End loop X

End loop Y

End loop J

End loop I

3.2.2.7 Inverse Wavelet Transform

Algorithm (3.19) illustrates the steps of the inverse Haar wavelet transform, while algorithms (3.20 a,b) illustrate the steps of inverse Tap9/7 wavelet transform.

Algorithm (3.19) Inverse Haar Wavelet Transform

Input:

Dc()= 2-D array represent the dequantized band of (Y or C_b or Cr)

W=Width of Bnd : H=height of Bnd

Nopass =the number of wavelet transform levels or pass.

Output:

Bnd() = 2-D array represent one band of (Y,Cb,or Cr).

Method:

Hm =H-1 : Wm =W-1

H1 =H : W1 =W

For all I where 1 ≤ I ≤ Nopass -1

H1=H1 div 2 : W1=W1 div 2

End loop I

For all I where 1 ≤ I ≤ Nopass -1

Hh=H1 div 2 : Wh=W1 div 2

For each Column y (y=0,1....Hh-1)

y1 =2*y : y2 =y1+1: yy =y+Hh

For each Row x (x=0,1....Wh-1)

x1 =2*x : x2 = x1+1 :xx =x +Wh

Bnd(x1,y1) = (Dc(x,y)+ Dc(xx,y)+ Dc(x,yy)+Dc(xx,yy))/2

Bnd(x2,y1)= (Dc(x,y)+ Dc(xx,y)- Dc(x,yy)-Dc(xx,yy))/2

Bnd(x1,y2)= (Dc(x,y)- Dc(xx,y)+ Dc(x,yy)-Dc(xx,yy))/2

Bnd(x2,y2)= (Dc(x,y)- Dc(xx,y)- Dc(x,yy)+Dc(xx,yy))/2

End loop x

End loop y

W1=W1*2 : H1 =H1*2

End loop I

Algorithm (3.20 a) Inverse of Tap9/7wavelet transform

Input:

Dc()= 2-D array represent the dequantized band of (Y or C_b or Cr)

W=Width of Bnd : H=height of Bnd

Nopass =the number of wavelet transform levels or pass.

Output:

Bnd() = 2-D array represent one band of (Y,Cb,or Cr).

Method:

W_m = W - 1: H_m = H - 1

For all I where (I= NoPass To 1) (Step -1)

W_w = W : H_h = H

For all J where (J= 2 To I)

W_w = (W_w + 1) \ 2: H_h = (H_h + 1) \ 2

End loop J

W_m = W_w - 1: H_m = H_h - 1

For all Y where (Y = 0 To H_m)

For all X where (X = 0 To W_m)

A(X) = Dc(X, Y)

End loop X

Tap97Inverse W_w, A(), B()//algorithm (3.20.b)

For all X where (X = 0 To W_m)

Dc(X, Y) = B(X)

End loop X

End loop Y

For all X where (X = 0 To W_m)

For all Y where (Y = 0 To H_m)

A(Y) = Dc(X, Y)

End loop Y

Tap97Inverse H_h, A(), B()//algorithm (3.20.b)

For all Y where (Y = 0 To H_m)

Dc(X, Y) = B(Y)

End loop Y

End loop X

End loop I



```

For all Y where (Y = 0 To Hm)
For all X where (X = 0 To Wm)
    T = Dc(X, Y)
    If T < 0 Then T = 0 Else If T > 255 Then T = 255
    Bnd(X, Y) = T
End loop X
End loop Y

```

Algorithm (3.20 b) inverse Tap9/7

Input:

N= the height or width of b Bnd.

A()= vector consist one row or column.

Output:

B() =vector consist the changed row or column.

Method:

Neven = (N + 1) \ 2 : Nodd = N – Neven
Nm = N – 1 : Np1 = N + 1 : Np2 = N + 2
aa = 1 / 1.230174105 : I = 0 : J = 1

For all K where (K = 0, ..., Neven-1)

C(I) = 1.230174105* A(K)

I=I+2

End loop K

For all K where (K = 0, ..., Nodd-1)

C(J) = aa* A(Neven+k)

J=J+2

End loop K

For all K where (K = 1, ..., 4)

C(Nm + K) = C(Nm - K)

End loop K

For all K where (K = 1, ..., 4)

C(-K) = C(K)

End loop K

For all K where (K = -2, ..., Np2) (Step 2)

C(K) = C(K) – 0.4435068522 * (C(K - 1) + C(K + 1))

End loop K

For all K where (K = -1, ..., Np1) (Step 2)

C(K) = C(K) – 0.8829110762 * (C(K - 1) + C(K + 1))

End loop K


```

For all K where (K = 0,...,N) (Step 2)
    C(K) = C(K) + 0.05298011854 * (C(K - 1) + C(K + 1))
End loop K
For all K where (K = 1,...,Nm) (Step 2)
    C(K) = C(K) + 1.586134342 * (C(K - 1) + C(K + 1))
End loop K
For all K where (K = 0,..., Nm)
    B(K) = C(K)
End loop K

```

3.2.2.8 Up Sampling

This operation implies the up sampling of (C_b , C_r) color components. Algorithm (3.21) illustrates the steps of the up sampling.

Algorithm (3.21) Up Sampling

```

Input: BndCb()= 2-D array(W/2×H/2), BndCr()= 2-D array(W/2×H/2) .
Output: Cb()= 2-D array(W×H) represent the up sampled components
           Cr()=2-D array(W×H) represent the up sampled components
Method:
Wm= W \ 2-1 : Hm= H \ 2-1 : y1=0 : y2=1
For each Column y (y=0,1....Hm-1)
    x1=0 : x2=1
    For each Row x (x=0,1....Wm-1)
        Cb(x1,y1)=BndCb(x,y) : Cr(x1,y1)=BndCr(x,y)
        If x2 < W then
            Cb(x2,y1)=BndCb(x,y) : Cr(x2,y1)=BndCr(x,y)
        End if
        If y2 < H then
            Cb(x1,y2)=BndCb(x,y) : Cr(x1,y2)=BndCr(x,y)
            If x2 < W then Cb(x2,y2)=BndCb(x,y) : Cr(x2,y2)=BndCr(x,y)
        End if
    x1= x1+2 : x2= x2+2
    End loop x
    y1= y1+2 : y2= y2+2
End loop y

```

3.2.2.9 YC_bC_r to RGB Transform

After reconstructing (Y, C_b and C_r) components, they must be transformed to (R, G and B) spaces, so the output of this operation will be the reconstructed image that will be saved as a Bitmap file by using the equation in (2.15). Algorithm (3.22) illustrates the steps of the inverse color transform.

Algorithm (3.22) Inverse Color Transform

Input: YCC() = 2-D array of YC_bC_r image (W×H)

W=width of image, H=high of image.

Output: Pic() = 2-D array of RGB image(W×H)

Method:

For each Column y (y=0,1....H-1)

For each Row x (x=0,1....W-1)

- **Compute R band from YCC(x,y)**

$$\text{Pic}(x,y).R = \text{YCC}(x,y).Y + 1.4075 * (\text{YCC}(x,y).Cr - 128)$$

- **Compute G band from YCC(x,y)**

$$\text{Pic}(x,y).G = \text{YCC}(x,y).Y - 0.34656 * (\text{YCC}(x,y).Cb - 128) - 0.71711 * (\text{YCC}(x,y).Cr - 128)$$

- **Compute B band from YCC(x,y)**

$$\text{Pic}(x,y).B = \text{YCC}(x,y).Y + 1.77804 * (\text{YCC}(x,y).Cb - 128)$$

End loop x

End loop y



Chapter Four

Test and Results

CHAPTER FOUR

Test and Results

4.1 Introduction

This chapter, attempted to evaluate both objective and subjective methods for an acceptable degree of the reconstructed images of different compression tests. Some of the well known fidelity measures (MSE, PSNR, CR and BR) have been used to assess the quality of the reconstructed image.

At the first step, this chapter illustrates a comparison between the Haar and Tap9/7 wavelet transform test results, then it will illustrate the effect of the Quantization parameters on the overall system performance, and in the last step, it will illustrate a comparison between the test results of the original SPIHT, Modified SPIHT and Modified SPIHT with DCT & RLE.

The test images used in this work are some standard bit-map (BMP) images. These images as shown in figure (4.1) which are six 24-bit true color images 256×256 .



Figure (4.1) the Original Test Color Images

4.2 Results and Discussion

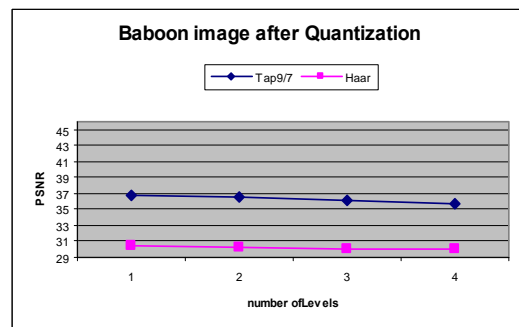
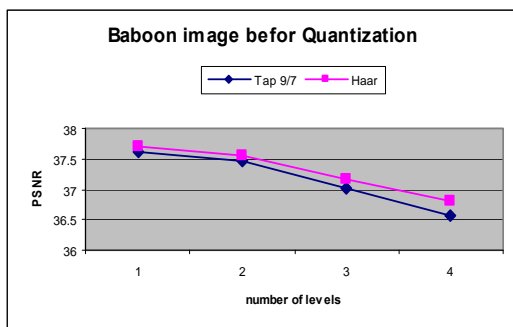
1. Results of (Haar and Tap9/7) wavelet transform: Table (4.1) illustrate a comparison between the (Haar and Tap9/7) wavelet transform, and shows the effect of the number of wavelet transform passes on the reconstructed images, also it illustrates the changes of the PSNR and MSE when the Nopass was varied between (1-4) for each image.

From table (4.1), it is clear that the PSNR is little better when using Haar wavelet transform rather than Tap9/7. Table (4.2) shows the effect of the quantization on each wavelet transform (Haar and Tap97), and shows that Tap9/7 give better PSNR than Haar wavelet transform on all the selected images after applying the uniform quantization. At this point, the project is continued using Tap97 instead of Haar wavelet transform.

| Table (4.1) The effect of Nopass parameters on the Haar & Tap9/7 Wavelet transform | | | | | |
|---|--------------|-------------|-------------|--------------|-------------|
| Image | Level | Haar | | Tap97 | |
| | | MSE | PSNR | MSE | PSNR |
| Baboon | 1 | 11.014 | 37.711 | 11.239 | 37.623 |
| | 2 | 11.395 | 37.563 | 11.673 | 37.458 |
| | 3 | 12.463 | 37.174 | 12.923 | 37.016 |
| | 4 | 13.55 | 36.811 | 14.312 | 36.573 |
| Girl | 1 | 2.45 | 44.238 | 2.413 | 44.304 |
| | 2 | 2.781 | 43.687 | 2.82 | 43.626 |
| | 3 | 3.504 | 42.684 | 3.818 | 42.311 |
| | 4 | 4.041 | 42.065 | 4.957 | 41.178 |
| Lena | 1 | 4.395 | 41.7 | 4.474 | 41.623 |
| | 2 | 4.746 | 41.366 | 4.9 | 41.228 |
| | 3 | 5.675 | 40.59 | 6.118 | 40.264 |
| | 4 | 6.707 | 39.865 | 7.454 | 39.406 |
| Baby face | 1 | 1.687 | 45.857 | 1.782 | 45.019 |
| | 2 | 2.013 | 45.09 | 2.178 | 44.75 |
| | 3 | 2.844 | 43.59 | 3.305 | 42.938 |
| | 4 | 3.785 | 42.349 | 4.587 | 41.514 |
| Parrots | 1 | 8.124 | 39.032 | 8.225 | 38.979 |
| | 2 | 8.468 | 38.852 | 8.64 | 38.765 |
| | 3 | 9.447 | 38.377 | 9.858 | 38.192 |
| | 4 | 10.43 | 37.947 | 11.177 | 37.647 |
| Beauty Girl | 1 | 2.05 | 45.011 | 2.143 | 44.819 |
| | 2 | 2.409 | 44.311 | 2.574 | 44.023 |
| | 3 | 3.193 | 43.088 | 3.558 | 42.618 |
| | 4 | 4.112 | 41.989 | 4.741 | 41.371 |

| Table (4.2) The effect of Quantization on Haar & Tap97 Wavelet transform | | | | | |
|--|-------|--------|--------|--------|--------|
| Image | Level | Haar | | Tap97 | |
| | | MSE | PSNR | MSE | PSNR |
| Baboon | 1 | 98.824 | 30.435 | 13.65 | 36.776 |
| | 2 | 61.498 | 30.242 | 14.452 | 36.531 |
| | 3 | 63.579 | 30.097 | 15.856 | 36.128 |
| | 4 | 65.053 | 29.998 | 17.218 | 35.7 |
| Girl | 1 | 38.428 | 32.284 | 3.917 | 42.2 |
| | 2 | 39.982 | 32.112 | 4.604 | 41.499 |
| | 3 | 40.185 | 32.09 | 5.934 | 40.396 |
| | 4 | 40.778 | 32.026 | 7.066 | 39.638 |
| Lena | 1 | 40.402 | 32.066 | 6.471 | 40.02 |
| | 2 | 41.423 | 31.958 | 7.135 | 39.596 |
| | 3 | 41.895 | 31.909 | 8.732 | 38.19 |
| | 4 | 43.049 | 31.791 | 9.981 | 38.139 |
| Baby face | 1 | 20.642 | 34.983 | 2.119 | 44.868 |
| | 2 | 21.468 | 34.812 | 2.598 | 43.983 |
| | 3 | 19.429 | 35.246 | 3.589 | 42.58 |
| | 4 | 20.378 | 35.039 | 4.855 | 41.268 |
| Parrots | 1 | 41.738 | 31.925 | 9.525 | 38.342 |
| | 2 | 42.74 | 31.822 | 10.12 | 38.075 |
| | 3 | 44.089 | 31.687 | 11.649 | 37.467 |
| | 4 | 45.505 | 31.55 | 13.002 | 36.99 |
| Beauty Girl | 1 | 22.119 | 31.684 | 3.373 | 42.85 |
| | 2 | 45.959 | 31.507 | 4.084 | 42.019 |
| | 3 | 45.305 | 31.509 | 5.131 | 41.028 |
| | 4 | 46.08 | 31.495 | 6.283 | 40.149 |

Figure (4.2) illustrates the difference between (Haar and Tap9/7) wavelet transform results before and after the quantization.



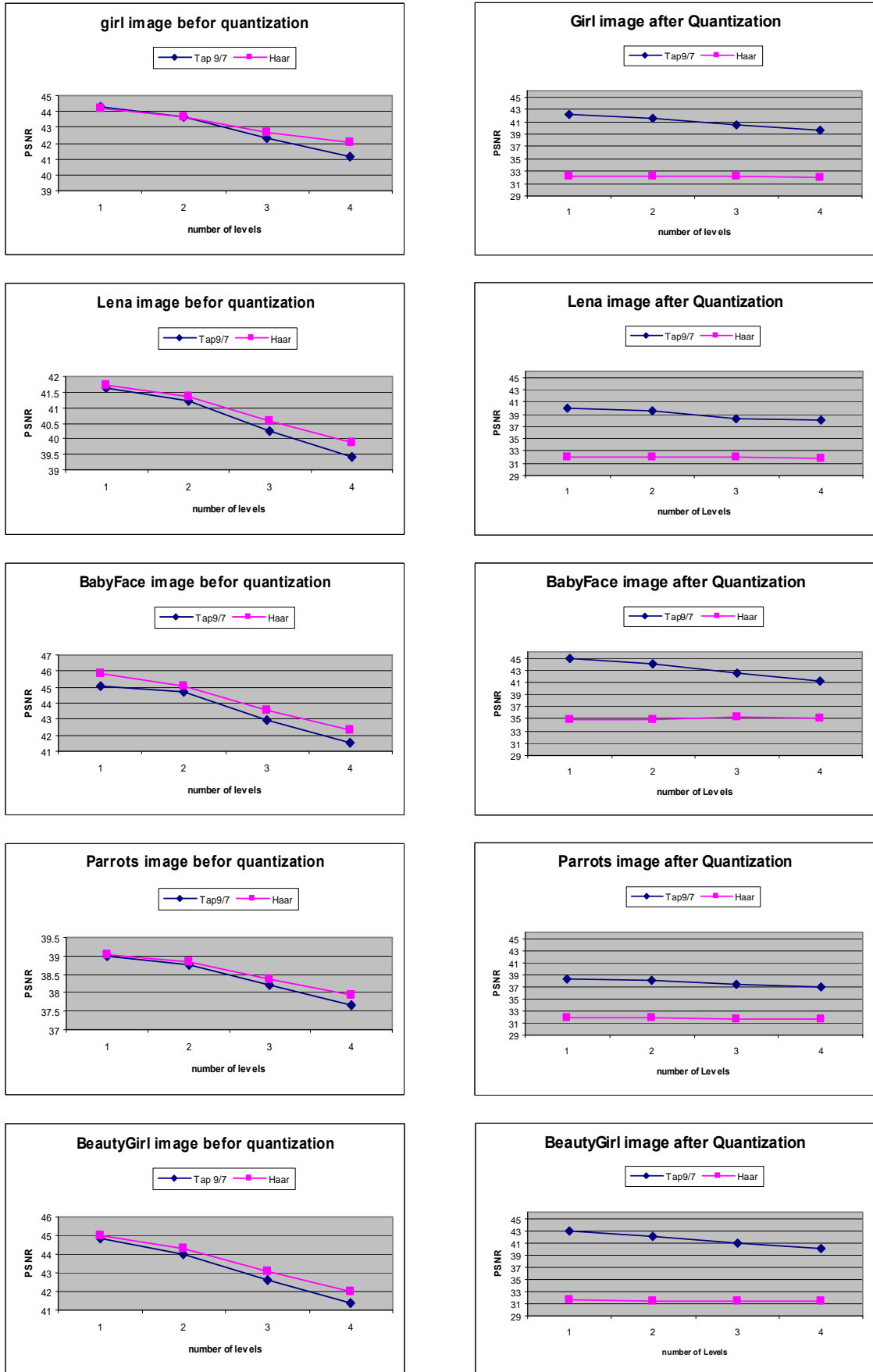


Figure (4.2) the difference between (Haar and Tap9/7) wavelet transform results before and after the quantization by using different Levels.

2. The effect and results of the quantization parameters:-

- Tables (4.3), (4.4), (4.5) and Figures (4.3), (4.4) and (4.5) illustrate the effect of the Hierarchical uniform quantization parameters (Q , β and α) on the compression performance after using equations (3.2 and 3.3) to quantize the (LH, HL and HH) subband of the wavelet transform.

| Table (4.3) The effect of Q | | | | | |
|---|------------|-------------|-----------|-----------|----------|
| When $\alpha=0.4$, $\beta=1.7$ | | | | | |
| Image | MSE | PSNR | CR | BR | Q |
| Baboon | 40.075 | 32.102 | 5.754 | 4.17 | 2 |
| | 51.806 | 30.986 | 9.041 | 2.654 | 4 |
| | 63.802 | 30.082 | 12.457 | 1.926 | 6 |
| | 76.713 | 29.282 | 16.181 | 1.483 | 8 |
| Girl | 32.51 | 33.01 | 9.172 | 2.616 | 2 |
| | 38.079 | 32.323 | 15.233 | 1.575 | 4 |
| | 43.769 | 31.719 | 20.636 | 1.162 | 6 |
| | 50.046 | 31.137 | 25.316 | 0.947 | 8 |
| Lena | 34.997 | 32.69 | 7.616 | 3.15 | 2 |
| | 42.807 | 31.815 | 11.982 | 2.002 | 4 |
| | 51.034 | 31.052 | 15.719 | 1.526 | 6 |
| | 60.238 | 30.332 | 19.366 | 1.239 | 8 |
| Baby face | 11.225 | 37.628 | 18.957 | 1.265 | 2 |
| | 13.383 | 36.865 | 32.379 | 0.741 | 4 |
| | 15.262 | 36.294 | 42.924 | 0.5 | 6 |
| | 16.989 | 35.828 | 50.155 | 0.478 | 8 |
| Parrots | 41.982 | 31.9 | 10.09 | 2.378 | 2 |
| | 48.454 | 31.277 | 16.653 | 1.441 | 4 |
| | 54.485 | 30.768 | 21.852 | 1 | 6 |
| | 60.251 | 30.331 | 26.393 | 0.969 | 8 |
| Beauty Girl | 23.707 | 34.381 | 7.756 | 3.094 | 2 |
| | 28.58 | 33.57 | 11.866 | 2.022 | 4 |
| | 35.815 | 32.59 | 15.927 | 1.506 | 6 |
| | 42.003 | 31.795 | 19.639 | 1.222 | 8 |

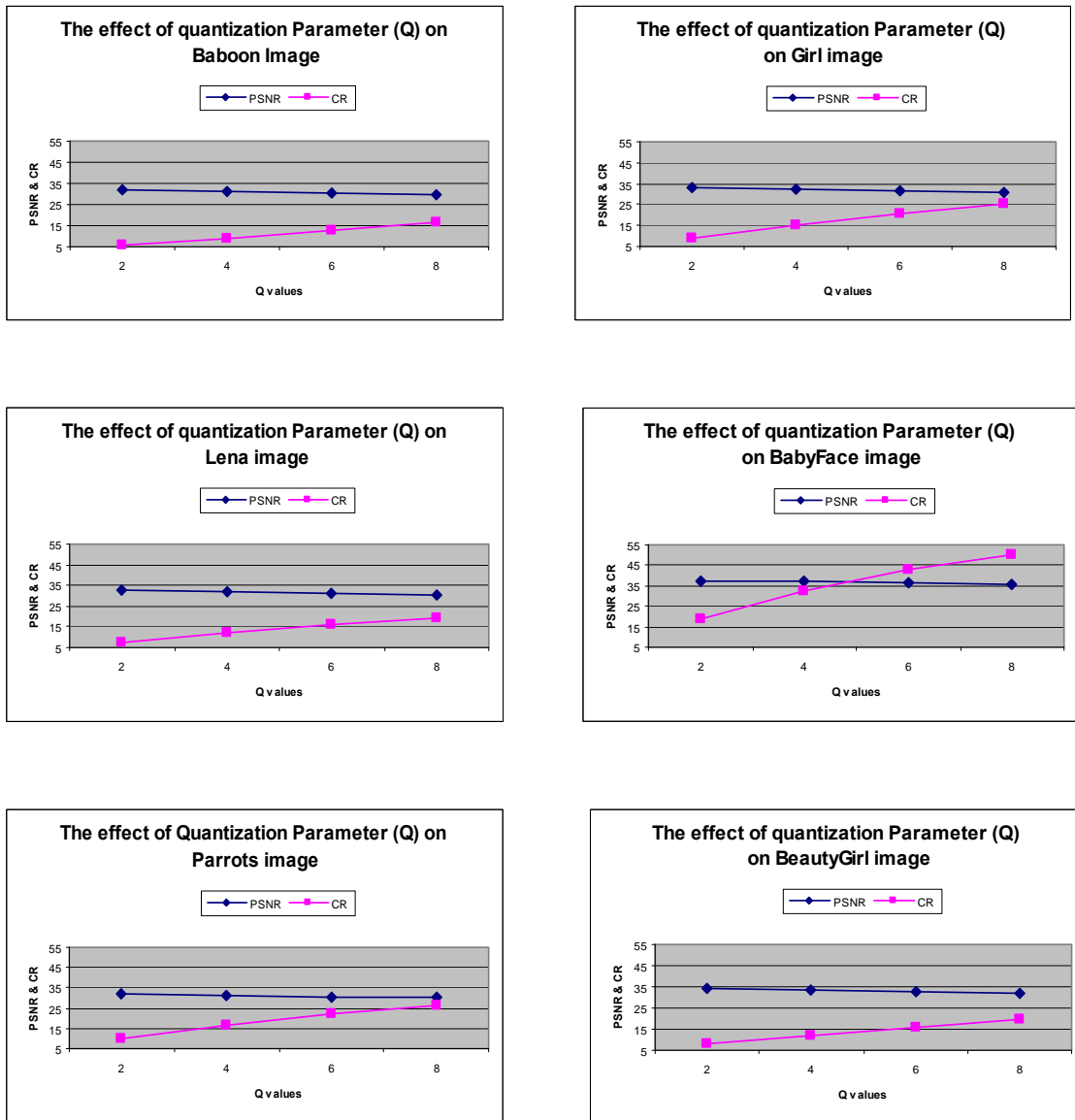


Figure (4.3) the effect of (Q) on compression performance

| Table (4.4) The effect of β When $Q=6, \alpha=0.4$ | | | | | |
|---|--------|--------|--------|-------|---------|
| Image | MSE | PSNR | CR | BR | β |
| Baboon | 62.246 | 30.189 | 11.646 | 2.06 | 1.2 |
| | 63.802 | 30.082 | 12.457 | 1.926 | 1.7 |
| | 64.606 | 30.028 | 12.809 | 1.873 | 2 |
| | 65.374 | 29.967 | 13.083 | 1.834 | 2.2 |
| Girl | 42.92 | 31.804 | 19.444 | 1.234 | 1.2 |
| | 43.769 | 31.719 | 20.636 | 1.162 | 1.7 |
| | 44.195 | 31.677 | 21.307 | 1.126 | 2 |
| | 44.673 | 31.63 | 21.713 | 1.105 | 2.2 |
| Lena | 49.956 | 31.144 | 14.939 | 1.606 | 1.2 |
| | 51.034 | 31.082 | 15.719 | 1.526 | 1.7 |
| | 51.746 | 30.991 | 16.066 | 1.493 | 2 |
| | 52.371 | 30.939 | 16.452 | 1.458 | 2.2 |
| Baby face | 15.127 | 36.333 | 41.338 | 1.58 | 1.2 |
| | 15.262 | 36.294 | 42.924 | 0.5 | 1.7 |
| | 15.334 | 36.274 | 43.778 | 0.548 | 2 |
| | 15.432 | 36.246 | 44.141 | 0.543 | 2.2 |
| Parrots | 53.929 | 30.812 | 21.061 | 10139 | 1.2 |
| | 54.485 | 30.768 | 21.852 | 1 | 1.7 |
| | 54.708 | 30.75 | 22.321 | 1.075 | 2 |
| | 54.827 | 30.74 | 22.663 | 1.058 | 2.2 |
| Beauty Girl | 34.846 | 32.709 | 14.951 | 1.605 | 1.2 |
| | 35.815 | 32.59 | 15.927 | 1.506 | 1.7 |
| | 36.361 | 32.524 | 16.359 | 1.467 | 2 |
| | 36.696 | 32.484 | 16.626 | 1.443 | 2.2 |

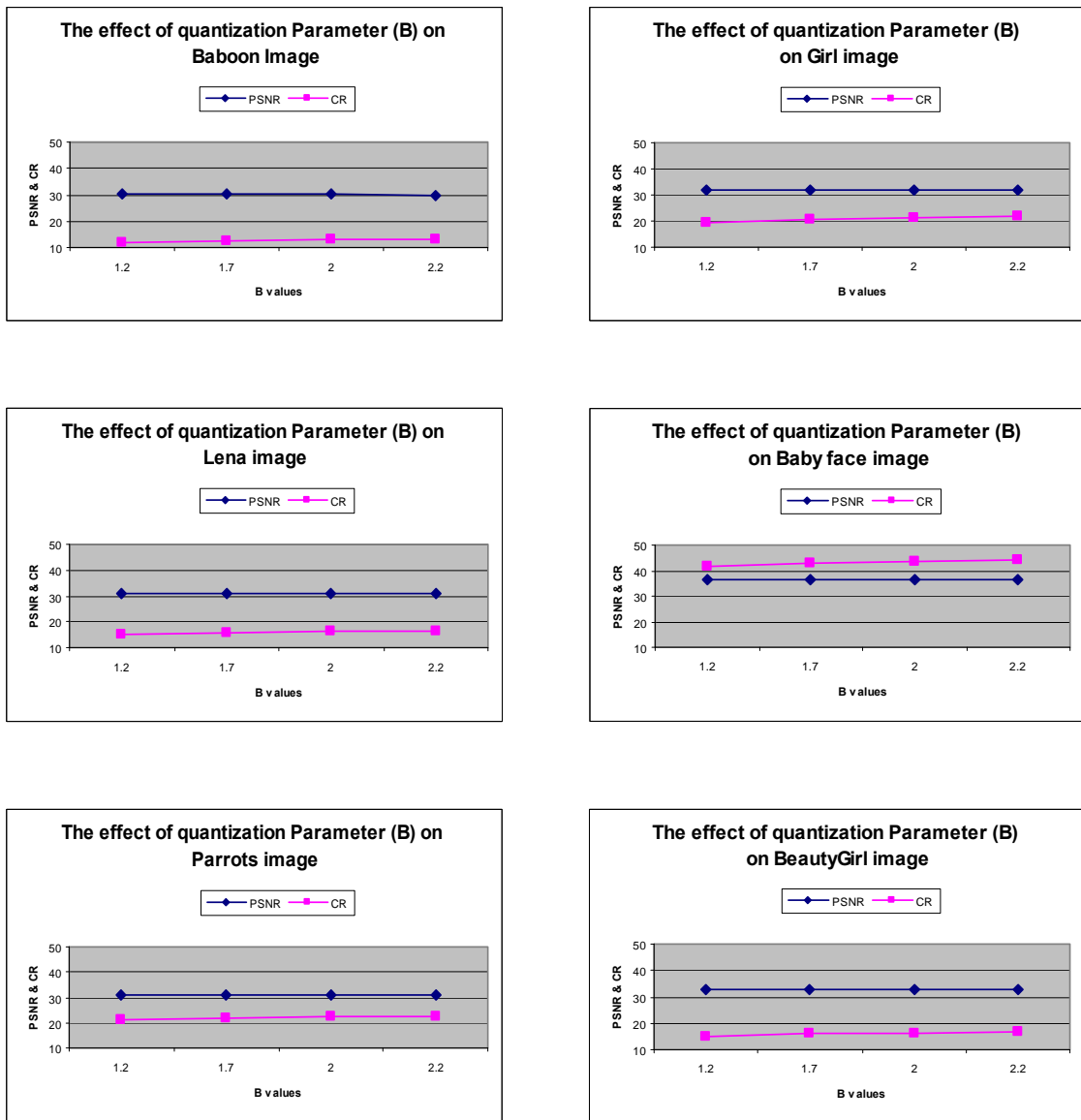


Figure (4.4) the effect of (β) on compression performance

| Table (4.5) The effect of α When $Q=6, \beta=1.7$ | | | | | |
|---|--------|--------|--------|-------|----------|
| Image | MSE | PSNR | CR | BR | α |
| Baboon | 48.541 | 31.269 | 7.359 | 3.26 | 0.2 |
| | 63.802 | 30.082 | 12.457 | 1.926 | 0.4 |
| | 84.087 | 28.883 | 19.483 | 1.231 | 0.6 |
| | 113.76 | 27.57 | 29.357 | 0.817 | 0.8 |
| Girl | 36.261 | 32.536 | 11.494 | 2.087 | 0.2 |
| | 43.769 | 31.719 | 20.636 | 1.162 | 0.4 |
| | 55.374 | 30.697 | 30.652 | 0.782 | 0.6 |
| | 72.249 | 29.542 | 41.61 | 0.576 | 0.8 |
| Lena | 40.029 | 32.106 | 9.45 | 2.539 | 0.2 |
| | 51.034 | 31.052 | 15.719 | 1.526 | 0.4 |
| | 64.919 | 30.007 | 22.213 | 1.08 | 0.6 |
| | 82.47 | 28.967 | 29.252 | 0.82 | 0.8 |
| Baby face | 12.519 | 37.154 | 19.119 | 1.255 | 0.2 |
| | 15.262 | 36.294 | 42.924 | 0.5 | 0.4 |
| | 18.631 | 35.428 | 58.409 | 0.41 | 0.6 |
| | 23.699 | 34.383 | 70.722 | 0.339 | 0.8 |
| Parrots | 46.661 | 31.441 | 12.281 | 1.954 | 0.2 |
| | 54.485 | 30.768 | 21.852 | 1 | 0.4 |
| | 64.49 | 30.018 | 31.292 | 0.766 | 0.6 |
| | 80.108 | 29.094 | 39.847 | 0.602 | 0.8 |
| Beauty Girl | 26.738 | 33.859 | 9.37 | 2.561 | 0.2 |
| | 35.815 | 32.59 | 15.927 | 1.506 | 0.4 |
| | 48.739 | 31.252 | 23.234 | 1.032 | 0.6 |
| | 69.526 | 29.709 | 32.13 | 0.746 | 0.8 |

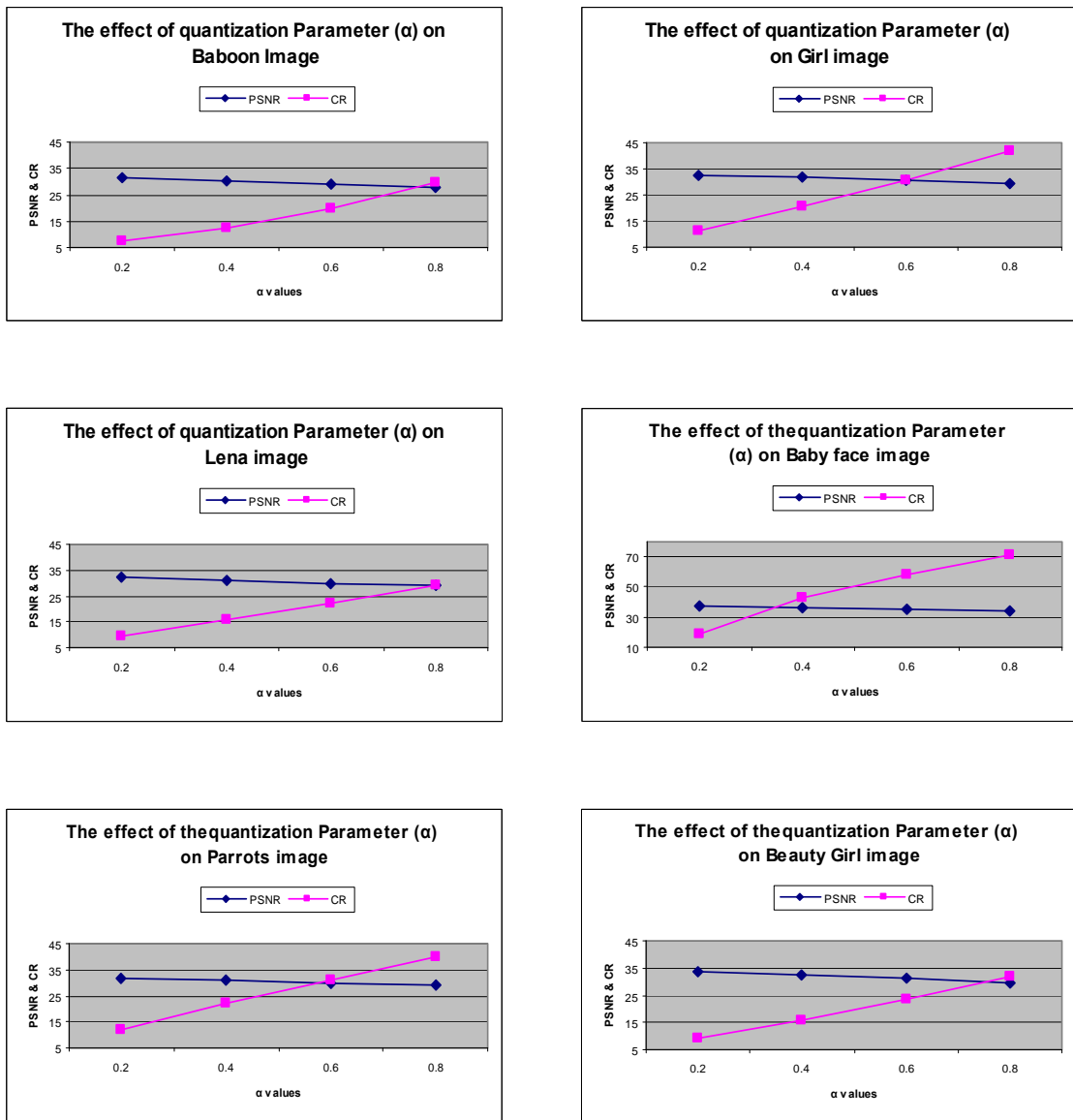


Figure (4.5) the effect of (α) on compression performance

From the above tables and figures it's clear that the Increase in quantization parameters (Q , β , and α), causes decrease in the image quality and increase in CR, and vice versa.

- Tables (4.6), (4.7), (4.8) and Figure (4.6), (4.7) and (4.8) illustrate the effect of the uniform quantization parameters (Q_{low} , Increment Rate (γ) and Q_{Dc}) that was applied on the LL subband of the wavelet transform using equation (3.4).

| Table (4.6) The effect of Q_{low} When BlkSize=8, $\gamma = 0.1$ and $Q_{Dc} = 4$ | | | | | |
|---|------------|-------------|-----------|-----------|-------------|
| <i>Image</i> | <i>MSE</i> | <i>PSNR</i> | <i>CR</i> | <i>BR</i> | <i>Qlow</i> |
| Baboon | 63.802 | 30.082 | 12.457 | 1.926 | 0.5 |
| | 70.459 | 29.651 | 13.01 | 1.844 | 1 |
| | 88.625 | 28.655 | 13.741 | 1.746 | 2 |
| | 113.198 | 27.592 | 14.181 | 1.692 | 3 |
| Girl | 43.769 | 31.719 | 20.636 | 1.162 | 0.5 |
| | 49.169 | 31.213 | 21.952 | 1.093 | 1 |
| | 62.776 | 30.152 | 23.416 | 1.024 | 2 |
| | 80.36 | 29.08 | 24.371 | 0.984 | 3 |
| Lena | 51.034 | 31.082 | 15.719 | 1.526 | 0.5 |
| | 56.807 | 30.586 | 16.471 | 1.457 | 1 |
| | 72.764 | 29.511 | 17.535 | 1.368 | 2 |
| | 92.296 | 28.478 | 18.379 | 1.305 | 3 |
| Baby face | 15.262 | 36.294 | 42.924 | 0.5 | 0.5 |
| | 18.571 | 35.442 | 48.07 | 0.449 | 1 |
| | 27.857 | 33.681 | 53.137 | 0.451 | 2 |
| | 39.141 | 32.204 | 56.65 | 0.423 | 3 |
| Parrots | 54.485 | 30.768 | 21.852 | 1 | 0.5 |
| | 62.198 | 30.193 | 23.545 | 1.019 | 1 |
| | 79.929 | 29.103 | 25.71 | 0.933 | 2 |
| | 102.926 | 28.002 | 27.182 | 0.882 | 3 |
| Beauty Girl | 35.815 | 32.59 | 15.927 | 1.506 | 0.5 |
| | 40.65 | 32.04 | 16.434 | 1.46 | 1 |
| | 51.873 | 30.981 | 17.181 | 1.396 | 2 |
| | 68.106 | 29.798 | 17.02 | 1.355 | 3 |

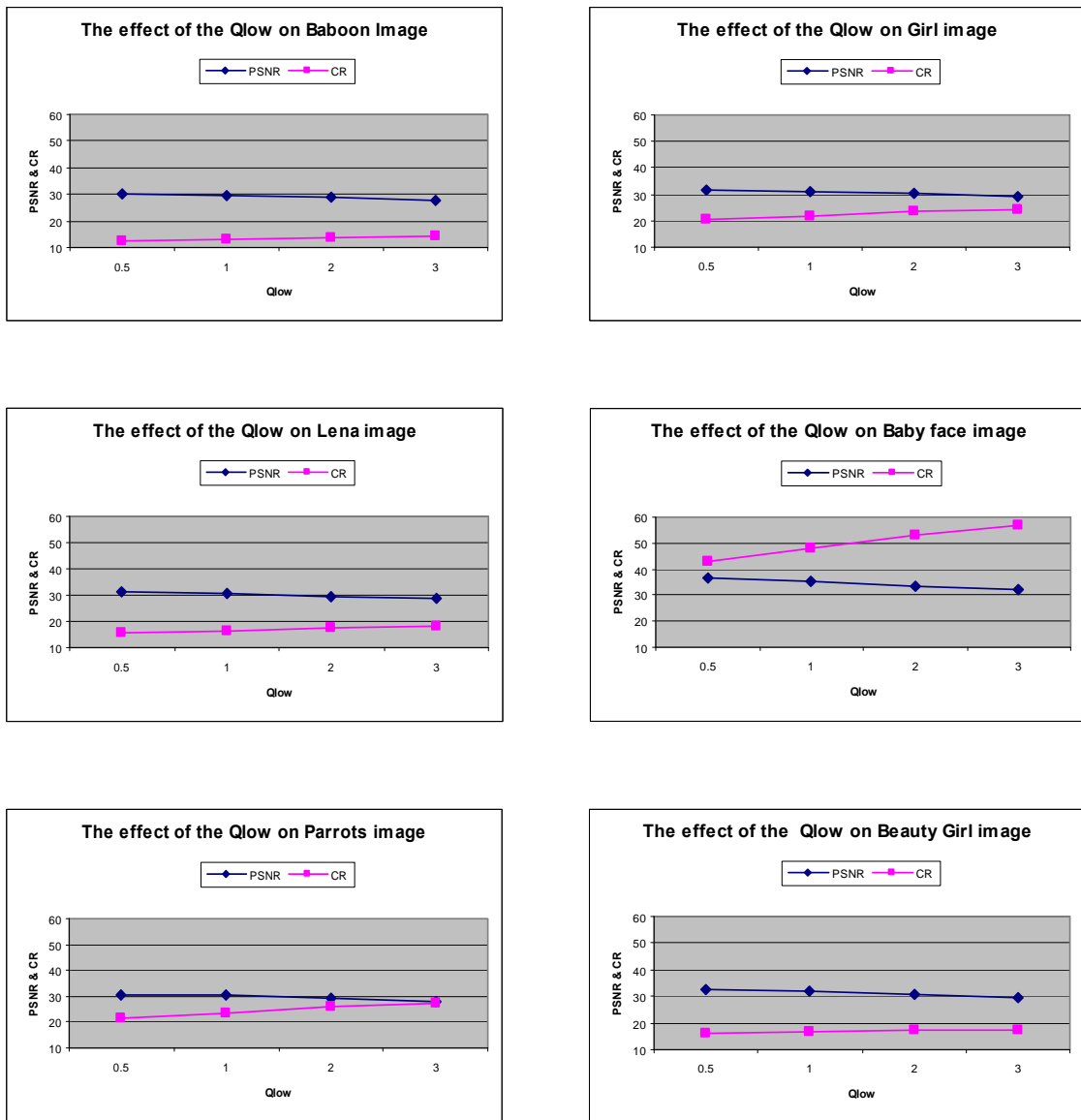


Figure (4.6) the effect of Q_{low} on compression performance

| Table (4.7) The effect of γ When BlkSize=8, Qlow=0.5 and Q _{Dc} =4 | | | | | |
|---|---------|--------|--------|-------|--------------|
| Image | MSE | PSNR | CR | BR | (γ) |
| Baboon | 63.802 | 30.082 | 12.457 | 1.926 | 0.1 |
| | 87.25 | 28.723 | 13.482 | 1.78 | 0.3 |
| | 114032 | 27.56 | 13.999 | 1.714 | 0.5 |
| | 133.101 | 26.888 | 14.262 | 1.682 | 0.7 |
| Girl | 43.769 | 31.719 | 20.636 | 1.162 | 0.1 |
| | 61.075 | 30.272 | 22.834 | 1.051 | 0.3 |
| | 9.505 | 29.126 | 23.924 | 1.003 | 0.5 |
| | 99.967 | 28.132 | 24.563 | 0.977 | 0.7 |
| Lena | 51.034 | 31.082 | 15.719 | 1.526 | 0.1 |
| | 69.299 | 29.723 | 17.129 | 1.401 | 0.3 |
| | 95.891 | 28.313 | 18.019 | 1.331 | 0.5 |
| | 123.571 | 27.211 | 18.547 | 1.293 | 0.7 |
| Baby face | 15.262 | 36.294 | 42.924 | 0.5 | 0.1 |
| | 23.734 | 34.376 | 49.875 | 0.481 | 0.3 |
| | 34.38 | 32.767 | 53.586 | 0.447 | 0.5 |
| | 44.114 | 31.684 | 55.902 | 0.429 | 0.7 |
| Parrots | 54.485 | 30.768 | 21.852 | 1 | 0.1 |
| | 76.287 | 29.306 | 24.786 | 0.968 | 0.3 |
| | 99.789 | 28.139 | 26.372 | 0.91 | 0.5 |
| | 121.461 | 27.286 | 27.382 | 0.876 | 0.7 |
| Beauty Girl | 35.815 | 32.59 | 15.927 | 1.506 | 0.1 |
| | 50.869 | 31.066 | 16.932 | 1.417 | 0.3 |
| | 69.823 | 29.69 | 17.526 | 1.369 | 0.5 |
| | 96.987 | 28.263 | 17.878 | 1.342 | 0.7 |

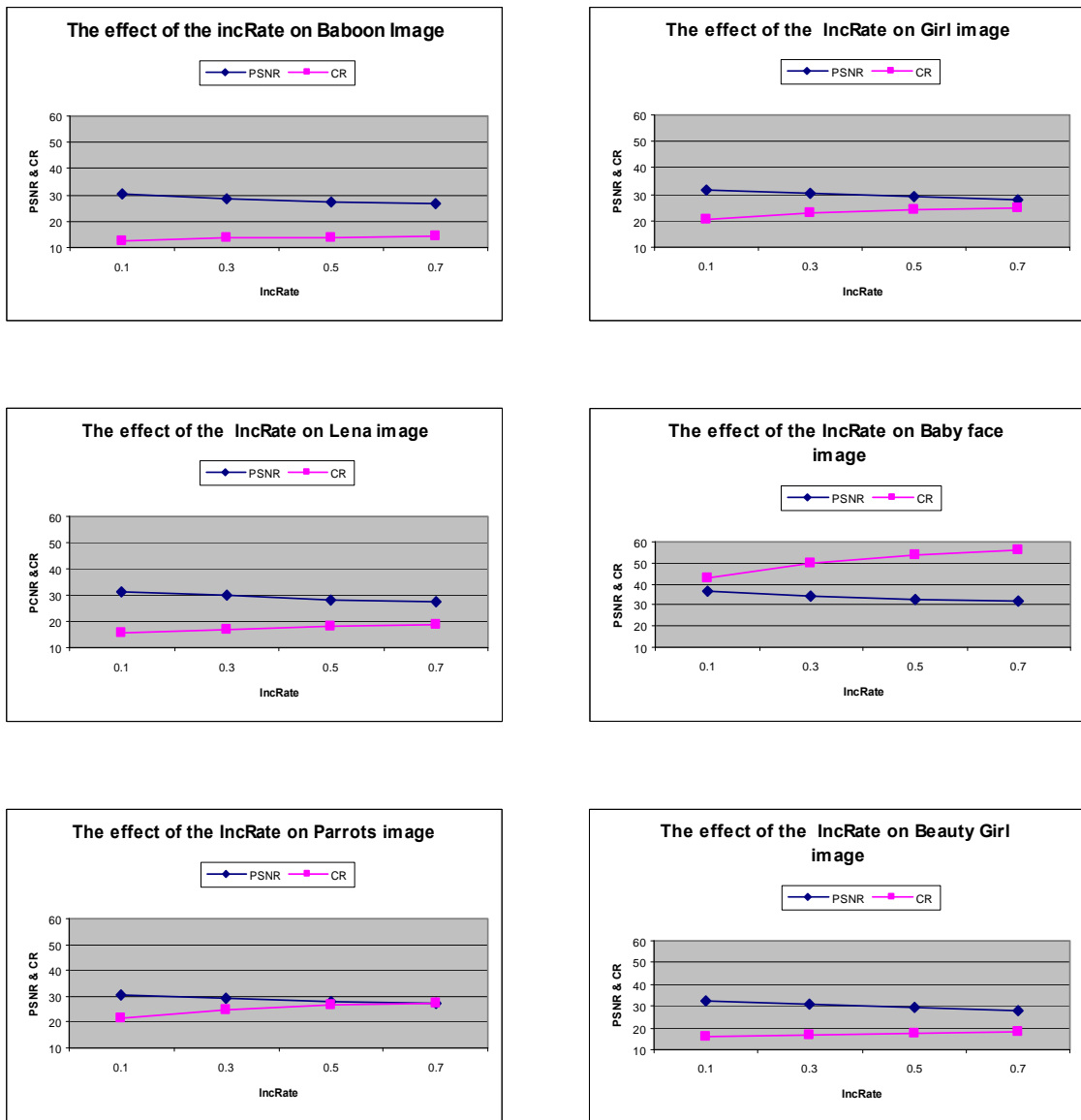


Figure (4.7) the effect of increment rate (γ) on compression performance

| Table (4.8) The effect of Q_{Dc} When BlkSize=8, $Q_{low}=0.5$ and $\gamma =0.1$ | | | | | |
|---|--------|--------|--------|-------|----------|
| Image | MSE | PSNR | CR | BR | Q_{Dc} |
| Baboon | 63.761 | 30.085 | 11.592 | 2.07 | 1 |
| | 63.71 | 30.088 | 12.093 | 1.984 | 2 |
| | 63.7 | 30.083 | 12.441 | 1.928 | 3 |
| | 63.802 | 30.082 | 12.457 | 1.926 | 4 |
| Girl | 43.766 | 31.719 | 19.895 | 1.206 | 1 |
| | 43.72 | 31.723 | 20.518 | 1.169 | 2 |
| | 43.767 | 31.719 | 20.615 | 1.164 | 3 |
| | 43.769 | 31.719 | 20.636 | 1.162 | 4 |
| Lena | 51.029 | 31.052 | 14.792 | 1.622 | 1 |
| | 51.004 | 31.054 | 15.369 | 1.561 | 2 |
| | 51.044 | 31.051 | 15.678 | 1.53 | 3 |
| | 51.034 | 31.052 | 15.19 | 1.526 | 4 |
| Baby face | 15.253 | 36.29 | 41.965 | 0.571 | 1 |
| | 15.249 | 36.298 | 42.509 | 0.564 | 2 |
| | 15.217 | 36.3 | 42.862 | 0.559 | 3 |
| | 15.262 | 36.294 | 42.924 | 0.5 | 4 |
| Parrots | 54.337 | 30.778 | 20.602 | 1.164 | 1 |
| | 54.372 | 30.777 | 21.377 | 1.122 | 2 |
| | 54.344 | 30.779 | 21.741 | 1.103 | 3 |
| | 54.485 | 30.768 | 21.852 | 1 | 4 |
| Beauty Girl | 35.79 | 32.594 | 15.012 | 1.598 | 1 |
| | 35.707 | 32.603 | 15.613 | 1.537 | 2 |
| | 35.818 | 32.589 | 15.685 | 1.53 | 3 |
| | 35.815 | 32.59 | 15.927 | 1.506 | 4 |

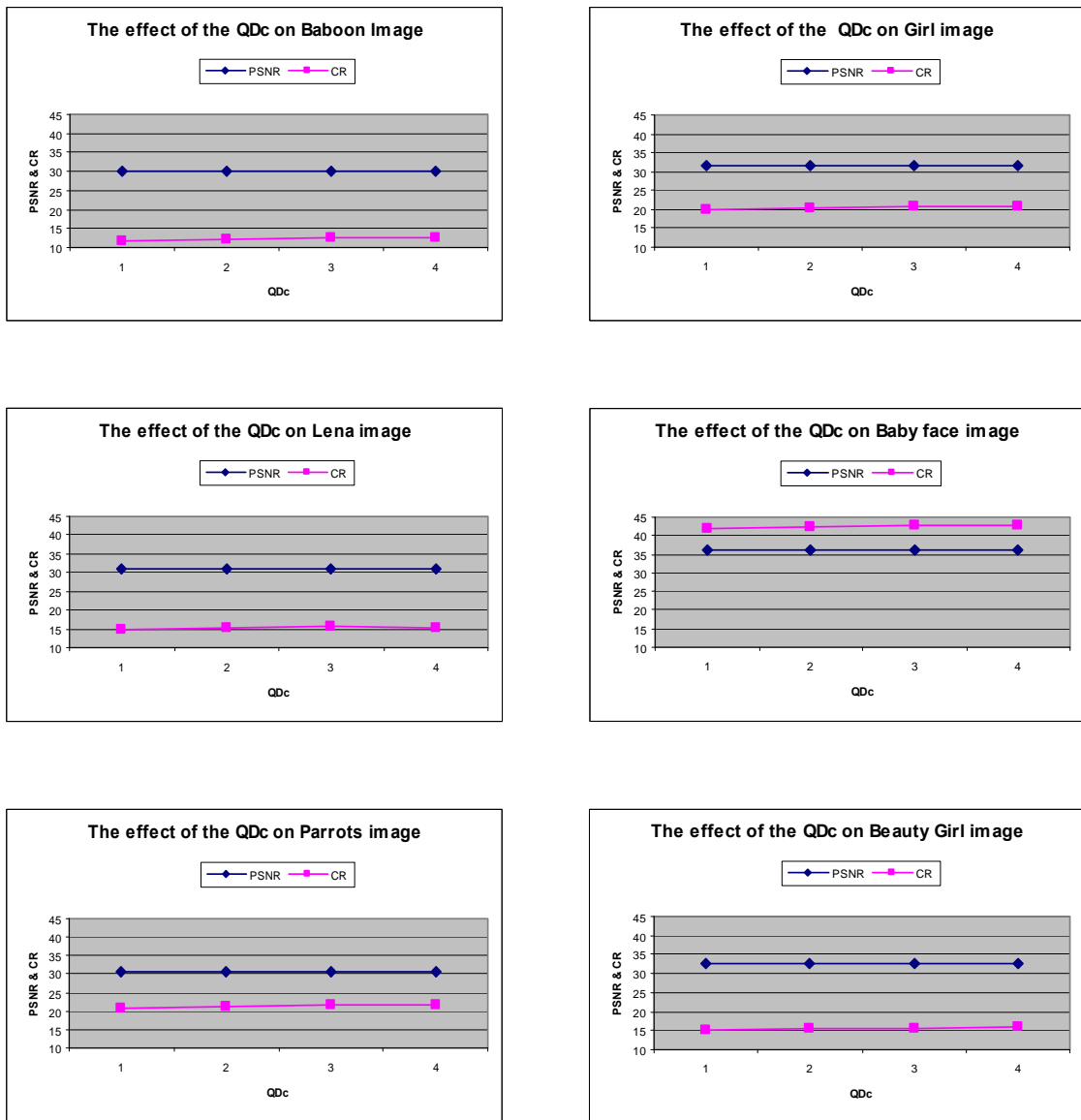


Figure (4.8) the effect of Q_{step} on compression performance

From the above tables and figures, it's clear that the increase in quantization parameters (Q_{low} , γ , and Q_{Dc}), causes decrease in the image quality and increase in CR, and vice versa.

- Table (4.9) illustrates the effects of the block length (Blksize) on compression performance parameters, when the value of (Blksize) was varied from 2 to 16 pixels, see Figure (4.9).

| Table (4.9) The effect of Blksize When $Q_{low}=0.5, Q_{DC}=4$ and $incr=0.1$ | | | | | |
|--|------------|-------------|-----------|-----------|----------------|
| <i>Image</i> | <i>MSE</i> | <i>PSNR</i> | <i>CR</i> | <i>BR</i> | <i>BlkSize</i> |
| Baboon | 59.283 | 30.401 | 11.486 | 2.089 | 2 |
| | 59.83 | 30.361 | 11.968 | 2.005 | 4 |
| | 63.802 | 30.082 | 12.457 | 1.926 | 8 |
| | 80.655 | 29.064 | 12.453 | 1.927 | 16 |
| Girl | 40.259 | 32.082 | 18.551 | 1.293 | 2 |
| | 40.007 | 32.109 | 19.613 | 1.223 | 4 |
| | 43.769 | 31.719 | 20.636 | 1.162 | 8 |
| | 55.023 | 30.725 | 21.034 | 1.14 | 16 |
| Lena | 46.406 | 31.465 | 14.226 | 1.687 | 2 |
| | 47.07 | 31.403 | 14.901 | 1.61 | 4 |
| | 51.034 | 31.052 | 15.719 | 1.526 | 8 |
| | 62.101 | 30.199 | 16.449 | 1.458 | 16 |
| Baby face | 12.892 | 37.027 | 32.872 | 0.73 | 2 |
| | 13.107 | 36.955 | 38.385 | 0.625 | 4 |
| | 15.262 | 36.294 | 42.924 | 0.5 | 8 |
| | 22.873 | 34.537 | 46.844 | 0.512 | 16 |
| Parrots | 50.016 | 31.139 | 19.279 | 1.244 | 2 |
| | 50.653 | 31.084 | 20.704 | 1.159 | 4 |
| | 54.485 | 30.768 | 21.852 | 1 | 8 |
| | 68.14 | 29.796 | 23.076 | 1.04 | 16 |
| Beauty Girl | 33.167 | 32.923 | 14.41 | 1.665 | 2 |
| | 33.364 | 32.897 | 15.302 | 1.568 | 4 |
| | 35.815 | 32.59 | 15.927 | 1.506 | 8 |
| | 43.136 | 31.782 | 15.826 | 1.516 | 16 |

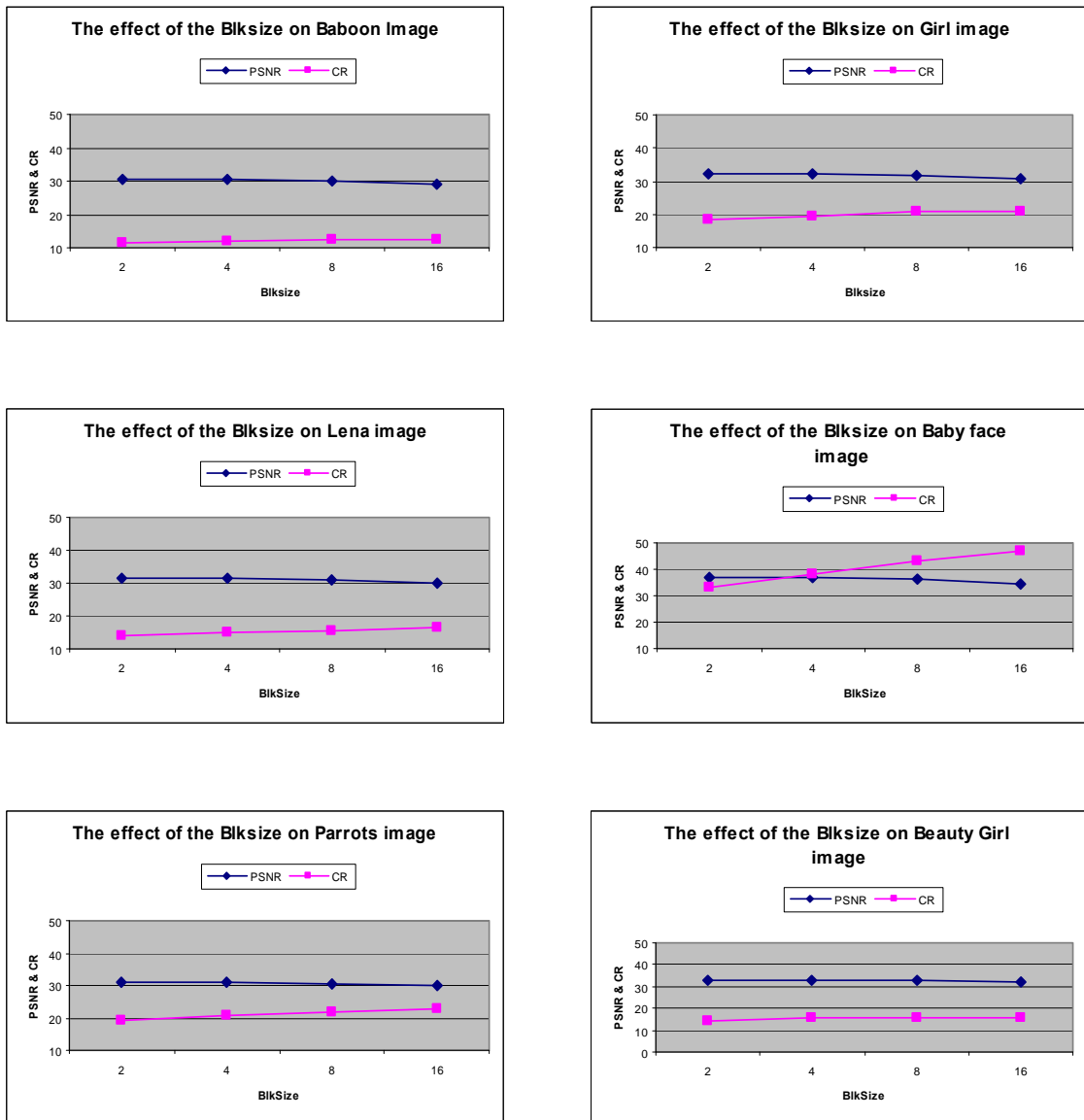


Figure (4.9) the effect of BlkSize on compression performance

From the above results it's clear that the increase in Low-Low block size causes increase in CR and decrease in image Quality.

From all these tables and figures the tested compression scheme parameters are chose to take the values described in table (4.10) as the best values for each parameter

| Table (4.10) The Best values of the relevant coding parameters | |
|---|-------------------|
| Parameter | Best Value |
| Number of passes | 3 |
| Q_{low} | 0.5 |
| γ | 0.1 |
| Q_{step} | 4 |
| Q | 6 |
| β | 1.7 |
| α | 0.4 |
| Block length | 8 |

3. Comparison between the test results of the original SPIHT, Modified SPIHT and Modified SPIHT with DCT & RLE:-

- The following Table illustrates comparison between the results of these methods for each image. Figure (4.10) and Figure (4.11) show the reconstructed images of each method.

| Table (4.11) comparison between the A-original SPIHT, B-modified SPIHT and C-modified SPIHT with DCT&RLE | | | | | | | |
|---|---------------|------------|-------------|-----------|-----------|-------------------|---------------------|
| <i>Image</i> | <i>Method</i> | <i>MSE</i> | <i>PSNR</i> | <i>CR</i> | <i>BR</i> | <i>Comp. Time</i> | <i>DComp . Time</i> |
| Baboon | A | 119.304 | 27.364 | 15.236 | 1.575 | 0.953 | 0.39 |
| | B | 56.698 | 30.595 | 9.508 | 2.524 | 2.062 | 0.39 |
| | C | 63.802 | 30.082 | 12.457 | 1.926 | 1.515 | 0.405 |
| Girl | A | 85.35 | 28.818 | 18.263 | 1.314 | 0.953 | 0.374 |
| | B | 37.745 | 32.362 | 13.315 | 1.802 | 1.734 | 0.39 |
| | C | 43.769 | 31.719 | 20.636 | 1.162 | 1.093 | 0.39 |
| Lena | A | 114.783 | 27.532 | 16.994 | 1.412 | 0.937 | 0.358 |
| | B | 43.967 | 31.699 | 11.138 | 2.154 | 1.89 | 0.374 |
| | C | 51.034 | 31.082 | 15.719 | 1.526 | 1.265 | 0.39 |
| Baby face | A | 265.403 | 23.891 | 22.658 | 1.059 | 0.905 | 0.358 |
| | B | 10.722 | 37.827 | 18.35 | 1.307 | 1.577 | 0.374 |
| | C | 15.262 | 36.294 | 42.924 | 0.5 | 0.78 | 0.358 |
| Parrots | A | 96.919 | 28.266 | 18.717 | 1.282 | 0.905 | 0.358 |
| | B | 47.396 | 31.373 | 13.86 | 1.731 | 1.686 | 0.375 |
| | C | 54.485 | 30.768 | 21.852 | 1 | 1.187 | 0.358 |
| Beauty Girl | A | 159.139 | 26.113 | 16.814 | 1.427 | 0.953 | 0.359 |
| | B | 30.925 | 33.227 | 10.913 | 2.199 | 1.89 | 0.39 |
| | C | 35.815 | 32.59 | 15.927 | 1.506 | 1.249 | 0.374 |



Figure (4.10) the differences between the results of the Original, Modified SPIHT and Modified SPIHT with DCT&RLE methods.

From these tables and charts, the following can be concluded:

1. Increasing the number of roots with decreasing the number of levels in the modified SPIHT algorithm led to increase number of pixels in the (LSP), i.e., it increases the Image quality and decreases CR.
2. The modified SPIHT algorithm increases PSNR with little increase in compress time and decoding time.
3. Using RLE and DCT with the modified algorithm cause increasing CR with preserving image quality as it, and decreasing in the compress time.

Figure (4.11) shows the reconstruction of all images after applying the Original SPIHT, Modified SPIHT and Modified SPIHT with DCT&RLE methods.



A

B

C



A

B

C



A

B

C



A

B

C



A

B

C



A

B

C

Figure (4.11) the reconstructed images of the A-Original, B-Modified SPIHT and C-Modified SPIHT with DCT&RLE methods

4. By using the modified SPIHT method, the image can be reconstructed in different sizes at the same time without needing to recompress the image in different size. This is done by using part of the same compressed file, so this property is more useful to be used on networks, see Figure (4.12).



**Image size=256*256
File size=192 Kb**



**Image size =128*128
File size=48 Kb**



**Image size =64*64
File size=12 Kb**

Figure (4.12) the reconstructed Parrots image in different sizes.



Chapter Five

Chapter

*Conclusions and
Future Works*

CHAPTER FIVE

Conclusions and Future works

5.1 Conclusions

Many test result were performed to study the effect of each method on the compression performance parameters. From the results that are mentioned in chapter four, the following conclusions could be summarized:

1. Tap9/7 wavelet transform are more efficient than Haar wavelet transform.
2. The quantization parameters mainly affect MSE, PSNR, CR and BR. It was found that the suitable quantization parameters values are ($Q_{low}=0.5$, $Q_{step}=4$, $incrate(\gamma)=0.1$, $Q=6$, $\beta=1.7$, $\alpha=0.4$, $Blksize=8$) they led to good PSNR (i.e., low distortion) and relatively high CR.
3. The increase of the roots in the modified SPIHT led to increase in the Compression time.
4. The time required to reconstruct the image is less than the time required to compress the image by this coding scheme.

5.2 Future Works

The following suggestions are introduced for future works:

1. Using the partitioned iterated function system (PIFS) or Huffman coding to compress the LL sub band.
2. Try another type of entropy coding (like, LZW coding) instead of shift coding.

3. Using another compression method on C_b and C_r instead of the SPIHT and DCT, such as the Vector Quantization.
4. The functionality of the proposed image compression could be extended from still image to video. With growing market of multimedia applications this subject needs greater considerations, because it is very fast to reconstruct the image.
5. Using different image formats and notice if it is effect on the results or not.



References

References

- [Aal96] Aalmoes, R., *"Video Compression Techniques Over Low-Bandwidth Line"*, M.Sc., thesis, twente University, 1996.
- [Abd03] Abdul Hameed T. Z., *"Advanced Compression Techniques for Multimedia Applications"*, M.Sc. thesis, Department of Computer Science, Saddam University, 2003.
- [Add00] Addl, J., *"Lossy and Lossless Image Compression"*, prentice Hall PTR, 2000.
- [Avc02] Avcbas, I., Memon, N, Sankur, B., and Sayood, K., *"A progressive lossless/Near-lossless Image Compression Algorithm"*, IEEE Signal Processing Letters, vol.9, no. 10, p.p.312-314, October-2002.
- [Bur98] Burrus, C., *"Introduction to Wavelets and Wavelet Transforms: A Prier"*, prentice Hall, New Jersey, 1998.
- [Cab02] Cabeen, K., *"Image Compression Using Distributed System"*, College of the Redwoods, 2002.
- [Cro01] Crosswinds, *"An Introduction to Image Compression"*, 2001 <http://www.crosswinds.net/~sskr/iagecp/index.ht>.
- [Dan04] Danyali H.^a, and Mertins A.^b, *" Fully Spatial and SNR Scalable, SPIHT-Based Image Coding for Transmission Over Heterogenous Networks"*, ^a Department of Electrical Engineering, University of Kurdistan, Sanandaj, Iran, ^b Signal Processing Group, Institute of Physics, University of Oldenburg, 26111 Oldenburg, Germany, Paper first received 2nd October 2003 and in revised form 16th May 2004.
- [Dat01] Data Compression References Center, *"RLE-Run Length Encoding"*, 2001, <http://www.rasip.fer.hr/research/copress/algorithms/fund/rl/index.html>.

- [Dun99] Dunn, "*Digital Color*", 1999, <http://davis.wpi.edu/~matt/courses/color>.
- [For98] Ford, A., Roberts, A., "*Color Space Conversions*", reports, 1998, <http://www.poynton.com/PDFs/coloureq.pdf>.
- [Fri95] Frigaard, C., "*Fast Fractal 2D/3D Image Compression*", report, Institute of Electronic System, Aalborg University, Laboratory of Image Analysis, 1995.
- [Gon00] Gonzales, R., C., and Woods, R., E., "*Digital Image processing*", Addison-Wesley Publishing Company, 2000.
- [Gon02] Gonzalez, R., Woods, R., "*Digital Image processing*", Pearson Education International, prentice hall, Inc, 2nd Edition, 2002.
- [Gra95] Graps, A., "*An Introduction to Wavelets*", IEEE Computational Sciences and Engineering, Vol. 2, P.P. 50-61, Summer 1995.
- [Haf01] Hafren, U., "*image and Video Compression*", publisher Linux Journal, 2001, <http://Ulli.linuxave.net/>.
- [Hil94] Hilton, Jawerth, B., Sengupta, A., "*Compressing Still and moving Image with Wavelets*", multimedia Systems, Vol. 2 and No. 3, 1994.
- [Hub95] Hubbard, b., "*The World According to Wavelets*", A K peters Wellesley, Massachusetts, 1995.
- [Ibr04] Ibraheem, N., I., "*Image Compression Using Wavelet Transform*", M.Sc. thesis, Bghdad University, 2004.
- [Jia03] Jainyun, X., "*Text Steganography using Wavelet Transform*" Dep. Copmuter science, new Mexico Teach, Socorro, USA, 2003.
- [Kha03] Khalifa, O., O., "*Fast Algorithm for VQ-based on Wavelet Coding System*", Electrical and Computer Department, University Malaysia, 2003,

http://sprg.massesey.ac.nz/ivcnz/Proceedings/IVCNZ_25.pdf.

- [Khe05] Khelifi F., Bouridane A., and Kurugollu F., "*A Scalable SPIHT-Based Multispectral Image Compression Technique*", School of Electronics, Electrical engineering and Computer Science Queen's University Belfast Northern Ireland, UK, 2005.
Email: fkhelifi01,a.bouridane,f.kurugollug@qub.ac.uk.
- [Kom94] Kominek, J., "*Still Image Compression an Issue of Quality*", Department of Computer Science, University of Waterloo, 1994,
<http://www.links.uwaterloo.ca:/pub/Fractals/papers/waterloo/kominek49b.xxx.ps.gz>
- [Kot04] Kotteri A. K., "*Optimal, Multiplierless Implementations of the Discrete Wavelet Transform for Image Compression Applications*", M.Sc. Thesis, Virginia Polytechnic Institute and state university, 2004.
- [Law99] H. Lawrence Rodrigues, "*Programming in Java Advanced Imageing*", Sun Microsystems Library,1999, Site: java.sun.com/products/java-media/jai/community/books/index.html
- [Mah05] mahmoud, G. A. H., "*Robust Watermarking system based on Wavelet Transform*", M.Sc. thesis, University of Technology, 2005.
- [Mar05] Marc S.W., David P., Tobias B., Stephan S., Philipp S., Norbert F., and Wolfgang F., "*Silicon Implementation of the SPIHT Algorithm for Compression of ECG Records*", Integrated Systems Laboratory (IIS) Swiss Federal Institute of Technology (ETH Zurich), Zurich, 2005. Email: mwegmuel@iis.ee.ethz.ch.

- [Mul97] Mulcahy, C., "*Image Compression Using the Haar Wavelet Transform*", Spelman College science & mathematics journal, Vol. 1, No. 1, pp. 22-31, 1997.
- [Mur07] Murry, D., James, V., Ryper and William, "*Run Length Encoding (RLE)*", Encyclopedia of Graphics File Formats book, 2ndEddition, 2007,
http://www.fileformat.info/mirror/egff/ch09_03.htm.
- [Nik05] Nikola S.^a, Sonja G.^b, Mislav G.^b, "*Modified SPIHT algorithm for wavelet packet image coding*", ^a Multimedia and Vision Lab, Department of Electronic Engineering, Queen Mary, University of London, London E1 4NS, UK, ^b Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3/XII, HR-10000 Zagreb, Croatia. Available online 10 August 2005.
[Site:www.sciencedirect.com](http://www.sciencedirect.com)
- [Nis98] Nister, D., "*embedded image Coding*", Licentiate of Engineering, Chalmers University of Technology, 1998.
- [Pra01] Pratt, W. K.; "*Digital Image Processing*"; 3rd Edition; John Wiley and Sons; New York; 2001.
- [Sah04] Saha S., "*Image Compression-from DCT to Wavelet: A Review*", the Math form internet mathematics library, Drexel School of Education, 2004. Site : <http://dret.net/biblio/reference/sah00>
- [Sai96a] Said, A., Pearlman, W.A., "*Anew fast and Efficient Image Code Based on Set Partitionning in Hirarchical Trees*" IEEE Transaction on Circuits and Systems for Video Technology, Vol.6,pp243-250,june 1996.
- [Sai96b] Said, A., Pearlman, W.A., "*SPIHT Image Compression: properties of the Method*" 1996,
<http://www.cipr.rpi.edu/research/SPIHT/siht1.html>.

- [Sal98] Salomon, D., *"Data Compression"*, second Edition, Springer, new York, 1998.
- [Sal02] Salomon, D., *"Data Compression" the Complete Reference*, Third edition, Addison Wesley Company, Northridge California, 2002.
- [Sal06] Salomon, D., *"Data Compression"*, the Complete Reference, Fourth edition, Lakeside, California, 2006.
- [San98] Sangwine, S., Horne, R., *"The Color Image Processing Handbook"*, Chapman & Hall, 1998.
- [sar08] sara A. Mahmood *"Using Discrete Cosine Transform To Encode Approximation Wavelet subband"*, M.Sc. thesis of Al-Nahrain University 2008.
- [Say00] Sayood, K., *"Introduction to Data Compression"*, second edition, Academic press, 2000.
- [Sch01] Schindler, M., *"Practical Huffman Coding"*, 2001, <http://www.compressconsult.com/Huffman>.
- [Shi00] Shi, Y., Sun, H., *"Image and Video Compression for Multimedia engineering: Fundamentals, Algorithms, and Standards"*, CRC, press, LLC, First Edition, 2000.
- [Sto94] Stollnitz, E., DeRose, T., Salesin, D., *"wavelets for Computer Graphics: A Primer"*, Tech. Report, Computer Science and Engineering Dep., Wessington Uni. Seattle, 1994.
- [Sus07] Susan S. Al-Barazanchi *"Image Compression Based on Fractal and wavelet Transform"*, M.Sc. thesis of Al-Nahrain University 2007.
- [Tho02] Thomas W. F., *"SPIHT Image Compression on FPGAs"* IBM Microelectronics Waltham, MA 02138 tom@tomfry.com, Scott Hauck Department of Electrical Engineering University of Washington Seattle, WA 98195 hauck@ee.washington.edu, 2002.

- [Tru99] Trulove, J., "*Multimedia Networking Handook*", 1999.
- [Umb98] Umbaugh, S.E., "*Computer vision and Image Processing: A practical approach using CVIP Tools*", prentice Hall PTR, Inc; USA, 1998.
- [Val01] Valens, C., "*A Really Friendly Guide to Wavelets*", 2001, <http://perso.wanadoo.fr/polyvalens/clemens/wavelets/wavelets.html>.
- [Wan00] Wang, A., "*Computer Basics*", Dep. Information and Computer Science, California Irvin, 2000, <http://www.ics.uci.edu/~dan/pubs/compressionBasics.html,albertAcs.tut.fi>, 2000.
- [Web1] "Color Space", Wikipedia, the free encyclopedia, 2009. Site:http://en.wikipedia.org/wiki/color_space
- [Web2] "YUV", Wikipedia, the free encyclopedia, 2009. Site:<http://en.wikipedia.org/wiki/YUV>
- [Xia01] Xiang, P., "*Image Compression by Wavelet Transform*", M.Sc. thesis, Qeartment of Computer and Information Sciences, East Tennessee State University, 2001.

الخلاصة

إن ضغط الصور هو احد تطبيقات ضغط البيانات على الصور الرقمية. في الواقع إن الهدف من ضغط الصور هو تقليل التكرار الموجود في بيانات الصورة لكي تكون قابلة للنقل وإرسال البيانات بشكل كفوء، هناك العديد من خوارزميات الضغط المتطورة، وهذا البحث يوضح عملية ضغط الصور باستخدام طريقة تقسيم المجموعة في الشجرة المرتبية.

في هذا البحث، تم تحويل الصورة من النظام اللوني احمر، اخضر، ازرق إلى نظام الشدة ومركبتي اللون، والهدف من هذا التحويل هو تهيئة الصورة لعملية التشفير وذلك بإزالة أي معلومات غير ضرورية. ثم تم تحويل مركبتي اللون إلى مرتبة اقل دقة بسبب افتقارهما إلى المعالم المكانية، بعد ذلك تبدأ عملية الضغط بتطبيق التحويل الموجي على كل حزمة لونية بصورة منفصلة. ثم يتم أولاً تشفير الحزمة الأولى للصورة باستعمال تحويل جيب التمام المتقطع والتكميم المنتظم أما المعاملات الأخرى للصورة فقد شغرت باستعمال المقياس الكمي الهرمي الموحد، وبعد ذلك طبقت طريقة تقسيم المجموعة في الشجرة المرتبية على كل حزمة بصورة منفصلة، وفي النهاية طبقت بعض طرق التشفير الأخرى مثل حساب طول الخطوة وتشفير الزحف على مجموعة النقاط المهمة للحصول على ضغط أكثر.

تم استخدام ست صور ملونة في هذا البحث لاختبار أداء النظام. في بادئ الأمر، أجريت مقارنة بين نتائج طريقتي التحويل الموجي (Haar, Tap9/7). وأظهرت النتائج إن (Tap9/7) يعطي نتائج أفضل بالنسبة إلى مقياس (نسبة قمة الإشارة إلى الضوضاء) لكل الصور المستخدمة، لذا فإن هذا العمل تم إكماله باستخدام طريقة (Tap9/7). وبعد ذلك أجريت العديد من الاختبارات لاختيار أفضل قيم لمتغيرات التكميم المنتظم. أخيراً، أجريت مقارنة بين نتائج كل من ضغط الصور باستخدام طريقة تقسيم المجموعة في الشجرة المرتبية الأصلية، وضغط الصور باستخدام طريقة تقسيم المجموعة في الشجرة المرتبية المعدلة وضغط الصور باستخدام طريقة تقسيم المجموعة في الشجرة المرتبية المعدلة مع تحويل جيب التمام المتقطع وحساب طول الخطوة، وقد أظهرت النتائج إن ضغط الصور باستخدام الطريقة الأخيرة يعطي قيم جيدة بالنسبة لمقياس (نسبة قمة الإشارة إلى الضوضاء) و(نسبة ضغط) عالية نسبياً.



جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة الكوفة
كلية العلوم

ضغط الصور الملونة باستخدام طريقة تقسيم المجموعة في الشجرة المرتبية المعدلة

رسالة

مقدمة الى كلية العلوم جامعة الكوفة كجزء من متطلبات
نيل درجة الماجستير في علوم الحاسبات

من قبل

خيرية سعيد عبد الجبار

(بكالوريوس جامعة الكوفة ٢٠٠٦)

باشرف

د. بان نديم ذنون