Abstract

Image compression involves reducing the size of image data files, while retaining necessary information. Compression is a necessary and essential method for creating image files with manageable and transmittable sizes. There have been many types of compression algorithms developed. This work exploits a hybrid image compression based on fractal coding and wavelet transform.

In the proposed method, the RGB image was transformed into YC_bC_r color transform, the goal of this transforming is to prepare the image for encoding process by eliminating any irrelevant information, then the compression technique is started by applying the Haar wavelet transform on the luminance component. The LL approximation subband is recompressed by applying the Partitioned Iterated Function System (PIFS) on it. The LL sub band is partitioned into non overlapped range blocks and overlapped domain blocks (the over lapping is according to the jump step value) using fixed size square blocks Partitioning. The matching technique is used to find the best domain block which satisfies the best map to the range block with minimum error. The detailed subbands (HL, LH and HH) are quantized using a uniform quantization. A new scheme of Run Length Encoding (RLE) is applied on the quantized sub bands. The resulted sub bands are coded again using S-Shift optimizer and S-Shift encoder.

The Chrominance components (Cb and Cr) are also compressed using Fractal coding with the same technique that used on the LL sub band.

الخلاصة

ضغط الصورة يتضمن تقليل حجم ملفات الصور مع الاحتفاظ بالمعلومات الضرورية. الضغط هو عملية ضرورية و اساسية لتكوين ملفات الصور ذات الحجوم القابلة للنقل . هنالك عدة طرق للضغط. هذا المشروع يستخدم ضغط الصور بالاعتماد على الضغط الكسوري و التحويل الموجي.

في هذا الشروع المقترح تم تحويل الصورة من النموذج اللوني وهو (احمر– اخضر– ازرق) الى مركبة الشدة و اللون، و الهدف من هذا التحويل لتحضير الصورة لعملية الترميز من خلال حذف اي معلومة غير متصلة بالموضوع، ثم تبدأ عملية الضغط من خلال تطبيق تحويل موجي نوع (هار) على مركبة الشدة. الحزمة التقريبية تم ضغطها مرة اخرى بأستخدام نظام دالة التقسيم المتكرر. الحزمة التقريبية تم تقسيمها الى كتل المجال غير المتداخلة و كتل المجال المقابل المتداخلة (و هذا التداخل يكون بحسب قيمة القفزة) باستخدام تقسيم ذا ابعاد ثابتة و متساوية. وقد استخدمت تقنية التكافؤ لايجاد افضل كتلة من المجال المقابل التي تكافئ كتلة من المجال باقل نسبة من الخطأ. اما بالنسبة لبقية الحزم الثانوية فقد حسبت بطريقة منتظمة وطبقت عليها طريقة جديدة بأستخدام ترميز طول التنفيذ ، وناتج هذة العملية تم ترميزه باستخدام مختزل الأس ومشفر الأس.

بالنسبة لبقية المركبات اللونية فقد ضغطت ايضا باستخدام الضغط الكسوري مع نفس التقنية المستخدمة بالنسبة للحزمة التقريبية.

Certification of the Examination Committee

We chairman and members of the examination committee, certify that we have studies this thesis "**Image Compression based on Fractal Coding and Wavelet Transform**" presented by the student **Susan Saadoon Al-Barazanchi** and examined her in its contents and that we have found it worthy to be accepted for the degree of Master of Science in Computer Science with very good degree.

Signature:

Name: **Dr. Abdul Monem S. Rahma**
Title : **Assistant Professor**
Date : / /**2007**
(Chairman)

Signature:

Name: **Dr. Bushra K. AL-Abudi**
Title : **Assistant Professor**
Date : / /**2007**
(Member)

Signature:

Name: **Dr. Taha S. Bashaga**
Title : **Lecturer**
Date : / /**2007**
(Member)

Signature:

Name: **Dr. Ban N. Al-Kallak**

Title : **Lecturer**

Date : / /**2007**

(Supervisor)

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

Signature:

Name: **Dr. LAITH ABDUL AZIZ AL-ANI**
Title : **Assist. Prof**.
Date : / /**2007**
(Dean of Collage of Science)

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 **Susan Saadoon Al-Barazanchi** as partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

Signature:
Name : **Dr. Ban N. Al-Kallak**
Title : **Lecturer**
Date : / / **2007**

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 : / / **2007**

Chapter Five
Conclusions and Future Work

5.1 Conclusions

In this work, an algorithm based on fractal coding and wavelet transform was proposed which led to the development of compression algorithms with high compression ratio.

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

1. **Block Size**: increasing the block size is proportional with the CR and inversely proportional to the PSNR, this is due to decreasing the number of the range and domain blocks will decrease the probability of finding the best domain block that match each range block.
2. **Jump step**: increasing the jump step value is proportional with the CR, and inversely proportional to the PSNR.
3. Increasing the number of bits needed to store the offset (NOBO) and scale (NOBS) parameters is proportional to the PSNR, and inversely promotional to the CR.
4. **The Quantization parameters "A", "B" and "R":** increasing the quantization parameters led to increasing the CR and decreasing the PSNR.
5. In order to get a good reconstructed image with best quality, neither the block size nor the jump step must be in a high values, such that it's better that the block size is not more than (4*4) block size for two wavelet levels, and (2*2) for three wavelet level . It's also not recommended that the jump step is more that 4 value, although it will increase the CR but it will decrease the PSNR and affect on the quality of the reconstructed image.
6. taking the block size (BS=8) and jump step (JS=4) for Cb and Cr components using Fractal coding will not have high affect on the quality

of the reconstructed image as it when applying on Y component, this is because that Cb and Cr just have the color information .

7. The IFS coefficients (scale and offset values) are highly affects on the compression ratio and PSNR values. They give better compression performance parameters when they are quantized with a high number of bits for both scale and offset parameters, such that NOBS=5, NOBO=7.

5.2 Future Work

1. Choosing another type of wavelet transform such as Tab97 that give a better compression performance.
2. Choosing another type of block partitioning instead of fixed block size partitioning, such as Quadtree partitioning, Horizontal-Vertical partitioning, Triangular partitioning.
3. Instead of a exhaustive search mechanism ,a Genetic Algorithm GA based on PIFS can be used to find the near optimal solution , it try to emulate biological evolutionary processes to solve optimization problems Instead of searching one point at a time .
4. Applying another type of compression method on Cb and Cr instead of the PIFS, such as Vector quantization.

Chapter Two
Theoretical Background

2.1 Introduction

Image compression has been pushed to the forefront of the image processing field. This is largely a result of the rapid growth in computer power, the corresponding growth in the multimedia market, and advent of the World Wide Web which makes the Internet easily accessible for everyone.

Compression algorithms development starts with applications to two-dimensional still images. Because video and television signals consist of consecutive frames of 2-D image data, the development of compression methods for 2-D still data is paramount importance. After these are developed, they are often extended to video. **[Umb98]**

This chapter explains the theoretical concept of image compression using a fractal and wavelet Transform, in addition to Run Length Encoding and S-Shift coding methods.

2.2 Compression methods

Compression method teaks in input X and generates a representation Xc (compressed output) that hopefully requires fewer bits. There is a reconstruction algorithm that operates on the compressed representation Xc to generate the reconstruction Y. Based on the requirement of reconstruction, data compression schemes can be divided into two broad classes (see figure (2.1)).

The first is lossless compression, in which Y is identical to X. The other is lossy compression, wich generally provide much higher compression than lossless compression but makes Y different from X**. [Ald00]**

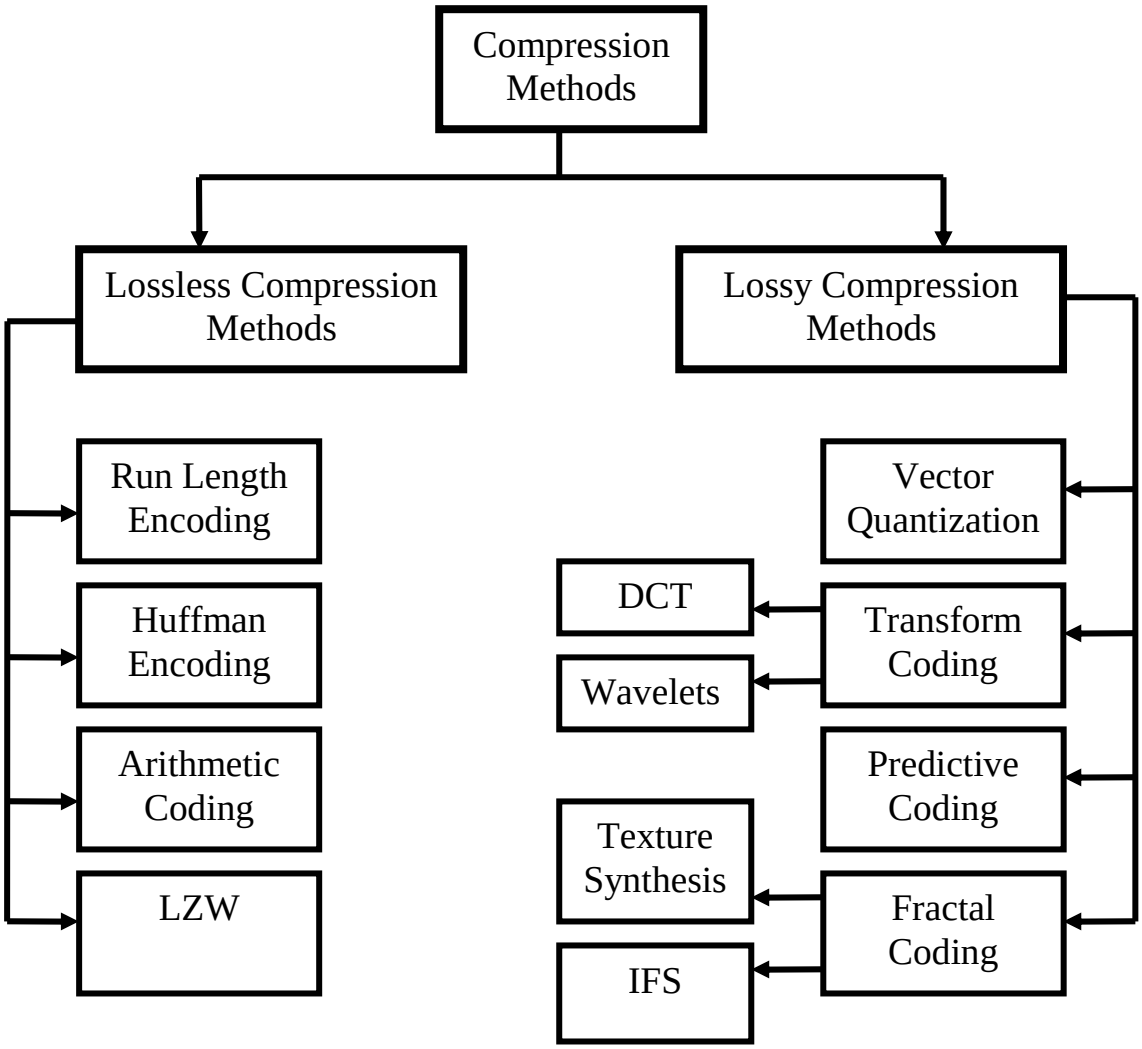


Figure (2.1): image compression methods [Ald00]

2.2.1 Lossless Compression Methods

The Lossless methods guaranty that the decompressed image is absolutely identical to the image before compression. This is the case when binary data such as executables, documents, etc. are compressed. They need to be exactly reproduced when decompressed. **[Kum03]**

2.2.1.1 Run Length Encoding (RLE)

It is Simple, intuitive, and fairly effective compression scheme for bitmapped 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 encoding does 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 encoding. 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 works 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 second byte is the value of the character in the run, which is in the range of 0 to 255, and is called the run value. Uncompressed, a character run of 15 A characters would normally require 15 bytes to store: AAAAAAAAAAAAAAA. The same string after RLE encoding 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]**

2.2.1.2 Huffman Encoding

Probably the best known coding method based on probability statistics is Huffman coding. D. A. Huffman published a paper in 1952 describing a method of creating a code table for a set of symbols given their probabilities**.** **[Nel98]**

The method starts by building a list of all the alphabet symbols in descending order of their probabilities. It then constructs a tree, with a symbol at every leaf, from the bottom up. This is done in steps, where at each step the two symbols with smallest probabilities are selected, added to the top of the partial tree, deleted from the list, and replaced with an auxiliary symbol representing both of them. When the list is reduced to just one auxiliary symbol (representing the entire alphabet), the tree is complete. The tree is then traversed to determine the codes of the symbols. [**Sal04**]

2.2.1.3 Arithmetic Coding

Arithmetic coding completely takes a stream of input symbols and replaces it with a single floating point output number. The longer (and more complex) message, the more bits are needed in the output number, until a practical method was found to implement this on computers with fixed sized registers. The output from an arithmetic coding process is a single number less than 1 and greater than or equal to 0. **[Nel98]**

As the message becomes longer, the interval needed to represent it becomes smaller and smaller, and the number of bits needed to specify that interval increases. Successive symbols in the message reduce this interval in accordance with the probability of that symbol. The more likely symbols reduce the range, and thus add fewer bits to the message. **[Lou06]**

The single number can be uniquely decoded to create the exact stream of symbols that went into its construction. In order to construct the output number, the symbols being encoded have a set probabilities assigned to them. **[Nel98]**

2.2.1.4 Lempel-Ziv-Welch (LZW)

It was developed by Terry Welch in 1984 as an improved version of the LZ78 dictionary coding algorithm developed by Abraham Lempel and Jacob Ziv.

Coding starts after the dictionary is initialized. The coder outputs the code of the longest dictionary entry that matches the characters at the current position in the input. It then forms a new entry by appending the input's next character to the current string, and gives this entry an unused code. These steps are repeated until there is no more input. The LZW decoder always maintains the same dictionary as the coder. It looks up each incoming code, outputs the corresponding entry, and then, by using the same rule as the coder, forms a new entry and gives it a code. **[Sly91]**

2.2.2 Lossy Compression Methods

In Lossy method, the result is less than optimal. This is because results have certain statistical properties which can be exploited by encoders specifically designed for them. Also, some of the finer details in the image can be sacrificed for the sake of saving a little more bandwidth or storage space. This also means that lossy compression techniques can be used in this area. **[Kum03]**

Lossy image coding techniques normally have three components:

- ***image modeling*** which defines such things as the transformation to be applied to the image
- ***parameter quantization*** whereby the data generated by the transformation is quantized to reduce the amount of information
- ***Encoding***, where a code is generated by associating appropriate code words to the raw data produced by quantizer.

Each of these components is in some part responsible of the compression. Image modeling is aimed at the exploitation of statistical characteristics of the image (i.e. high correlation, redundancy). Typical examples are transform coding methods, in which the data is represented in a different domain (for example, frequency in the case of the Fourier Transform (FT), the Discrete Cosine Transform (DCT), and so on), where a reduced number of coefficients contains most of the original information. In many cases this first phase does not result in any loss of information. The aim of quantization is to reduce the amount of data used to represent the information within the new domain. Encoding is usually error free. It optimizes the representation of the information and may introduce some error detection codes. **[Wil97]**

In the following sections, a review of the most important coding schemes for lossy compression is provided.

2.2.2.1 Vector Quantization (VQ)

Vector Quantization uses a codebook containing pixel patterns with corresponding indexes on each of them. The main idea of VQ is to represent arrays of pixels by an index in the codebook. In this way, compression is achieved because the size of the index is usually a small fraction of the block of pixels. **[Xia01]**

Here is an intuitive lossy method for image compression by vector quantization. Analyze a large number of different “training” images and find the *B* most common blocks. Build a codebook with these *B* blocks into both encoder and decoder. Each entry of the codebook is a block. To compress an image, scan it block by block, and for each block find the codebook entry that best matches it, and output a pointer to that entry. **[Sal04]**

2.2.2.2 Predictive Coding

Predictive coding has been used extensively in image compression. Predictive image coding algorithms are used primarily to exploit the correlation between adjacent pixels. They predict the value of a given pixel based on the values of the surrounding pixels. Due to the correlation property among adjacent pixels in image, the use of a predictor can reduce the amount of information bits to represent image. **[Xia01]**

The predictor will estimate the value of the input signal using the previous elements in the raw data. Then, the error signal, which is the difference between the actual input signal and the estimated signal, will be forwarded to the quantizor. At the quantizor, the error signal is quantized and forwarded to the receiver through the digital channel. At the receiver, the same operation will be performed. The predictor will estimate the value of the incoming signal and this value is added to the quantized error signal to reconstruct the actual signal plus a quantization error**. [Hus04]**

This type of lossy image compression technique is not as competitive as transform coding techniques used in modern lossy image compression, because predictive techniques have low compression ratios and worse reconstructed image quality than those of transform coding. **[Xia01]**

2.2.2.3 Transform Coding

The process of using a basis to resolve an image into a collection of weights is called a ***transform*. [Han99]**

A general transform coding scheme involves subdividing an $N \times N$ image into smaller $n \times n$ blocks and performing a unitary transform on each sub image. A unitary transform is a reversible linear transform whose kernel describes a set of complete, discrete basic functions. The goal of the transform is to correlate the original signal, and this correlation generally results in the signal energy being redistributed among only a small set of

transform coefficients. In this way, many coefficients may be discarded after quantization and prior to encoding.

Transform based compression is one of the most useful applications. Combined with other compression techniques, this technique allows the efficient transmission, storage, and display of images that otherwise would be impractical**. [Xia01]**

2.2.2.4 Fractal Coding

The birth of fractal geometry (or rebirth, rather) is usually traced to IBM Mathematician Benoit B. Mandelbrot and the 1977 publication of his seminal Book the Fractal Geometry of Nature. Later in the decade Michael Barnsley, a leading researcher from Georgia Tech, wrote the popular book Fractals Everywhere. He presents the mathematics of Iterated functions Systems (IFS), and proves a result known as the Collage Theorem. The Collage Theorem states what an Iterated Function System must be like in order to represent an image. **[Kom04]**

In Fractal image compression, an image is modeled as the unique fixed point of a contractive operator on the space of images. This type of image representation was first proposed by Barnsley who devised the first practical fractal coder. Fractal coding has since been a topic of active research because it has opened up a refreshing new view to image compression. It leads to visually pleasing results at high compression ratios, and it provides resolution independent image descriptions.

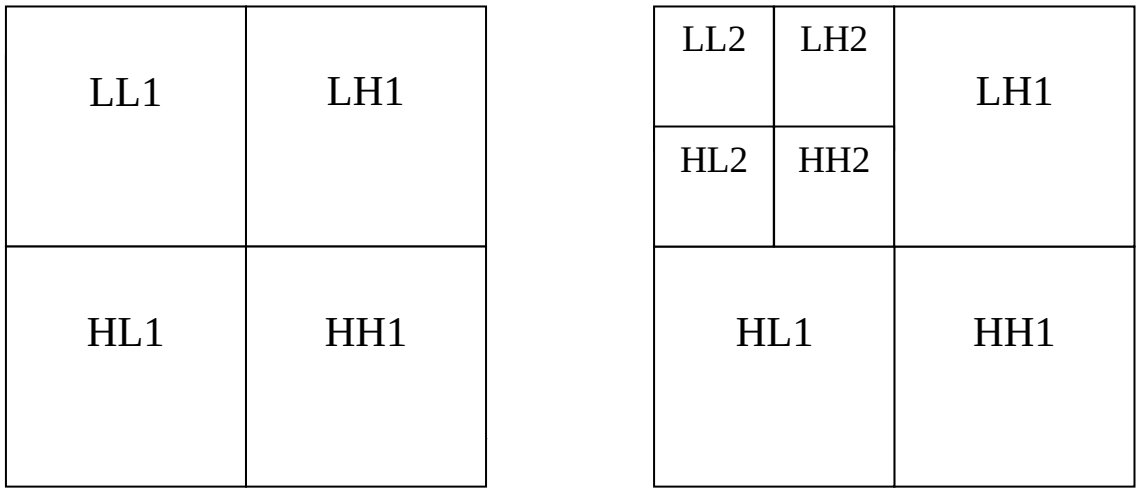
In fractal compression the image to be encoded is partitioned into image blocks called ranges. Each range is “coded” by a reference to some other part of the image and by some transformation parameters. These parameters describe how the referenced image part has to be adjusted with respect to contrast and brightness in order to give a good approximation to the range to be encoded. **[Har00]**

2.3 Wavelet Transform

The fundamental idea behind wavelets is to analyze the signal at different scales or resolutions, which is called multiresolution. Wavelets are a class of functions used to localize a given signal in both space and scaling domains. A family of wavelets can be constructed from a mother wavelet. Compared to Windowed Fourier analysis, a mother wavelet is stretched or compressed to change the size of the window. In this way, big wavelets give an approximate image of the signal, while smaller and smaller wavelets zoom in on details. Therefore, wavelets automatically adapt to both the high-frequency and the low-frequency components of a signal by different sizes of windows. Any small change in the wavelet representation produces a correspondingly small change in the original signal, which means local mistakes will not influence the entire transform**. [Xia01]**

Wavelet compression uses band pass filters to separate an image into images with low or high spatial frequencies. Low frequency images are those in which brightness change is gradual, for example, flat or rounded background areas. Such images appear soft and blurry. Higher frequency band images are crisp and sharp edged. To begin the decomposition, the image data are first partitioned into four sub band labeled as LL1, LH1, HL1, HH1, the goal of sub banding analysis is to transform the source image into alternative representation so that most of the energy is concentrated in lowest frequency sub band and in a few coefficients, to reduce the correlation and provide a useful data structure. To obtain the next level of decomposition, the LL1 sub band is further decomposed into the next level of four sub bands and the decomposition can be continued to as many levels as needed as shown in figure(2.2),the reconstruction is obtained by applying an inverse operation to that of the decomposition. Since that most of the image energy is concentrated in the lowest frequency sub band. Therefore, the quality of reconstruction of this sub band has great influence on quality of

the fully reconstructed image, for this reason this sub band has to be coded with relatively high fidelity. **[Kas02]**



(a) First level decomposition **(b) second level decomposition**

Figure (2.2), 2-Dimensional Wavelet Transform. [Kas02]

Where, L denotes a low band, H denotes a high band, and the subscript denote the number of the level.

Before talking about Haar wavelet transform, it is essential to describe the kinds of filters that are related to wavelet transform. **[Kap04]**

1. ***High pass filter***

   In digital signal processing (DSP) terms, the wavelet function is a high pass filter. A high pass filter allows the high frequency components of a signal through while suppressing the low frequency components. For example, the differences that are captured by the Haar wavelet function represent high frequency change between an odd and an even value.

2. ***Low pass filter***

   In digital signal processing (DSP) terms, the scaling function is a low pass filter. A low pass filter suppresses the high frequency components of a signal and allows the low frequency components through. The Haar

scaling function calculates the average of an even and an odd element, which results in a smoother, low pass signal.

3. Orthogonal (or Orthonormal) Transform

Orthogonal transforms are of interest because they can be used to re-express a time series in such a way that we can easily reconstruct the series from its transform. In a loose sense, the "information" in the transform is thus equivalent to the "information" in the original series; to put it in another way, the series and its transform can be considered to be two representations of the same mathematical entity.

2.3.1 The Haar Transform

The Haar Wavelet function consists of both: low pass and high pass filters. The high pass and low pass filters are called the decomposition filters because they break the image down or decompose the image into detailed and approximation coefficients, respectively. The approximation band (LL) is the result of applying low pass filter in vertical and horizontal directions, the (LH) band is the result of applying horizontal low pass filter and vertical high pass filter, while the (HL) band is the result of horizontal high pass filter and vertical low pass filter, the (HH) band is resulted from horizontal and vertical high pass filter. In this transform each (2×2) adjacent pixel are picked as group and passed simultaneously through four filters (LL, HL, LH, HH) to obtain the four wavelet coefficients, the basis of 4-filters could be derived as follows: **[Kas02]**

Low filter and high filter:

$$L=\frac{1}{\sqrt{2}}(1 \quad 1) \text{ ……..} \quad (2.1)$$

$$H=\frac{1}{\sqrt{2}}(1 \quad -1) \text{ ……} \quad (2.2)$$

Thus the horizontal low pass filter followed by the vertical low pass filter is equivalent to:

$$LL=\frac{1}{2}\begin{pmatrix}1\\1\end{pmatrix}\begin{pmatrix}1 & 1\end{pmatrix}=\frac{1}{2}\begin{pmatrix}1 & 1\\1 & 1\end{pmatrix} \ldots\ldots\ldots (2.3)$$

The horizontal high pass filter followed by vertical low pass filter is:

$$HL=\frac{1}{2}\begin{pmatrix}1\\-1\end{pmatrix}\begin{pmatrix}1 & 1\end{pmatrix}=\frac{1}{2}\begin{pmatrix}1 & 1\\-1 & -1\end{pmatrix} \ldots\ldots\ldots (2.4)$$

While the horizontal low pass filter followed by vertical high pass filter is equivalent to:

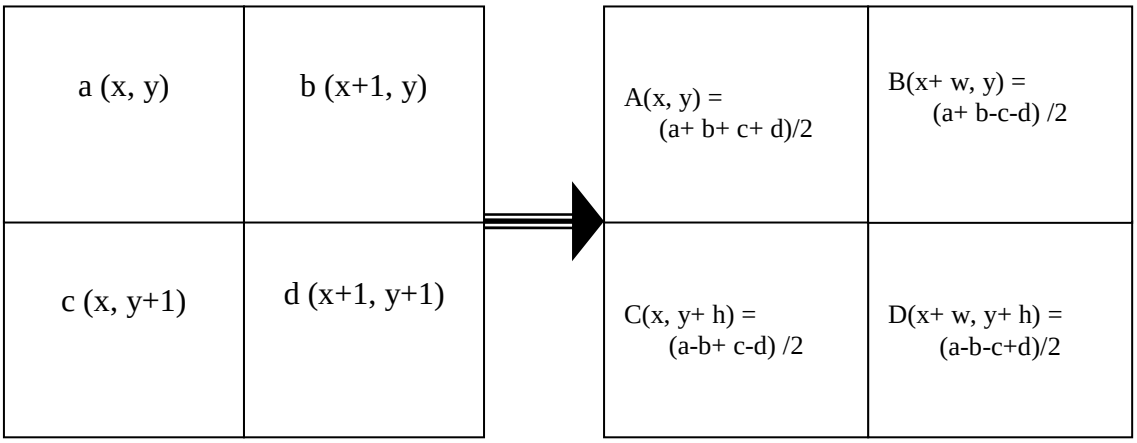
$$LH=\frac{1}{2}\begin{pmatrix}1\\1\end{pmatrix}\begin{pmatrix}1 & -1\end{pmatrix}=\frac{1}{2}\begin{pmatrix}1 & -1\\1 & -1\end{pmatrix} \ldots\ldots\ldots (2.5)$$

Finally the horizontal high pass filter followed by vertical high pass filter is:

$$HH=\frac{1}{2}\begin{pmatrix}1\\-1\end{pmatrix}\begin{pmatrix}1 & -1\end{pmatrix}=\frac{1}{2}\begin{pmatrix}1 & -1\\-1 & 1\end{pmatrix} \ldots\ldots\ldots (2.6)$$

2.3.2 Forward Haar Wavelet Transform

Forward Haar wavelet transform can be illustrated in figure(2.3),where a, b, c and d are the image pixel values, while A, B, C and D are the corresponding wavelet coefficients, w and h are half of the image width and height , respectively**.[Kas02]**



a. (2×2) **adjacent pixels** **b. (forward transform)**

Figure (2.3) the forward Haar wavelet transform [Kas02]

2.3.3 Inverse Haar Wavelet Transform

The output of forward Haar wavelet transform is the wavelet coefficients of the (LL, LH, HL and HH) bands. To reconstruct the image, the same four two dimensional filters have been used. Figure (2.4) illustrate the inverse Haar wavelet transform, where A, B, C and D are wavelet coefficients, while a, b, c are the reconstructed pixel values**. [Kas02]**

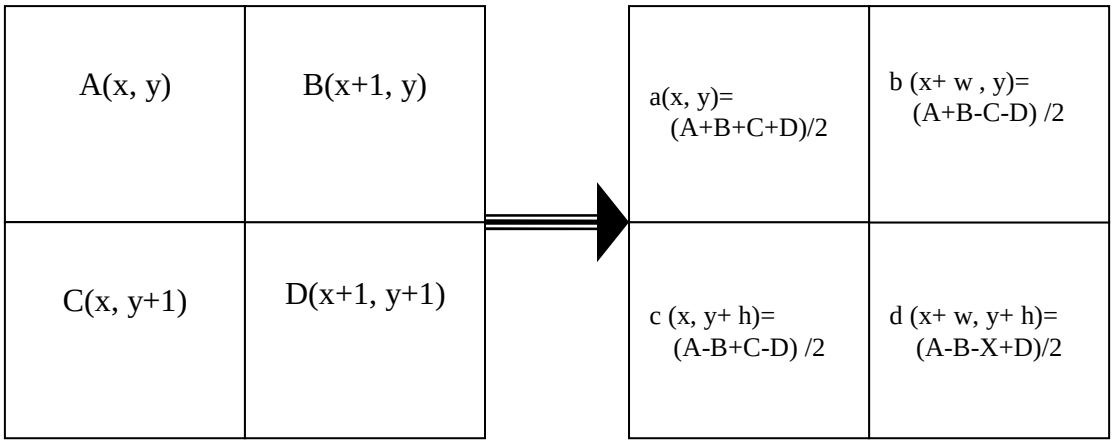


Figure (2.4) the inverse Haar wavelet transform [Kas02]

2.3.4 Why Wavelet Transform? [Zaf02]

Wavelet Transform has several advantages. Here are a number of these in regard to image compression and processing.

1. One of the main features of Wavelet Transform, which is important for data compression and image processing applications, is its good decorrelating behavior.
2. Wavelets are localized in both the space (time) and scale (frequency) domains. Hence they can easily detect local features in a signal.
3. Wavelets are based on multi-resolution analysis. Wavelet decomposition allows analyzing a signal at different resolution levels (scales), which results in superior objective and subjective performance.
4. Wavelets are smooth, which can be characterized by their number of vanishing moments. The higher the number of vanishing moments, the better smooth signals can be approximated with the wavelet basis.

2.4 Fractal definition

It is a rough or fragmented geometric shape that can be subdivided in parts, each of which is (at least approximately) a reduced/size copy of the whole. The term was coined by Mandelbrot and was derived from the Latin ***fractus*** meaning broken or fractured. **[Web1]**

Mathematically, a set of points whose fractal dimension exceeds its topological dimension**. [Bou91]**

Seven things about Fractal Image Compression:

1. It is a promising new technology, arguably superior to JPEG, but only with an argument.

2. It is a lossy compression method.

3. The fractals in Fractal Image Compression are Iterated Function Systems.

4. It is a form of Vector Quantization, one that employs a virtual codebook.

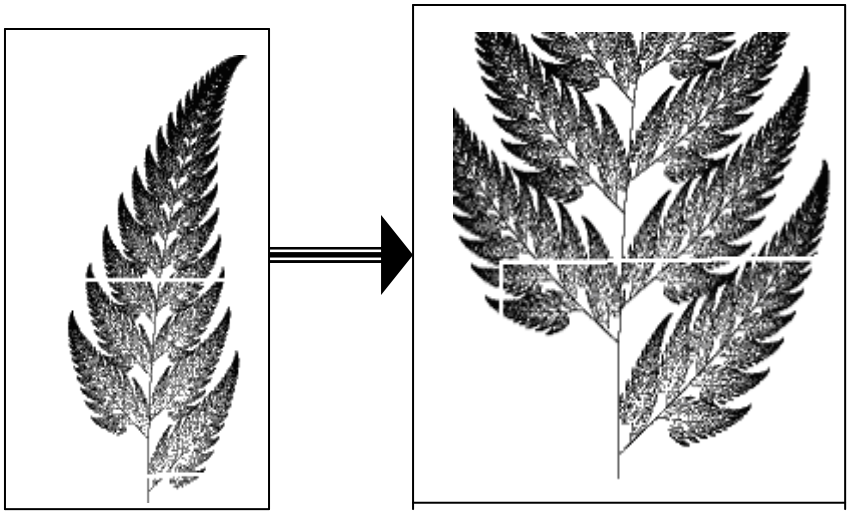
5. Resolution enhancement is a powerful feature but is not some magical way of achieving 1000:1 compression.

6. Compression is slow, decompression is fast

2.5 Self-similarity

An important (defining) property of a fractal is **self-similarity**, which refers to an infinite nesting of structure on all scales. Strict self- similarity refers to a characteristic of a form exhibited when a substructure resembles a superstructure in the same form.

Let's take a look at a common plant, the fern. The fern is typical of many plants in that it exhibits self similarity. A fern consists of a leaf, which is made up from many similar, but smaller leaves, each of which, in turn, is made from even smaller leaves. The closer we look the more detail we see, figure (2.5). This is a standard fern we will see the overall theme of repeating leaves. Each smaller leaf looks similar to the larger leaf.



Looking a little closer, you can see that those small leaves are made up from even smaller leaves.

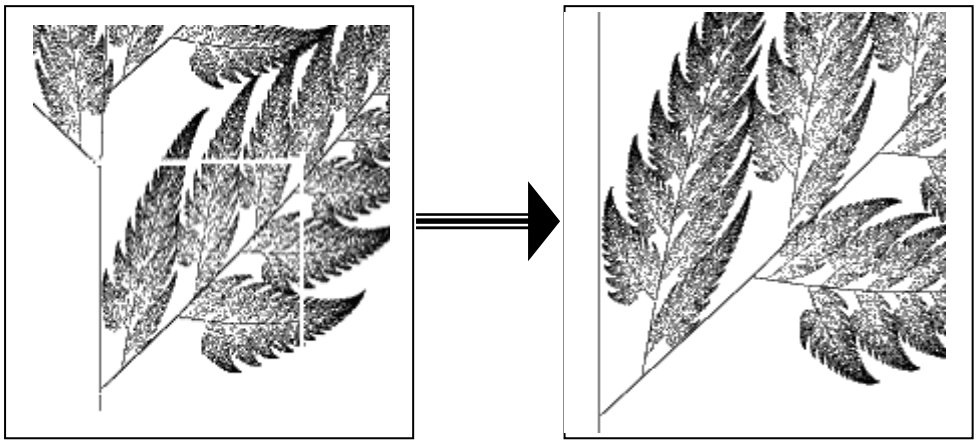


Figure (2.5) self similarity in fern [Man83]

No matter how close you look, more detail is always apparent, the same tends to be true for all fractals. A very simple algorithm can explain an infinitely complex object. **[Man83]**

2.6 Some kinds of fractals

There are many different kinds of fractal; this section gives an introduction to some kinds, like the L-system, chaotic, with the Mendelboart being an example, the "true" Mathematical Fractals, and IFS.

2.6.1 L-system

L-systems are a mathematical formalism proposed by the biologist Lindenmayer as a foundation for an axiomatic theory of biological development. More recently, L-systems have found several applications in computer graphics the basic idea is to define complex objects by successively replacing parts of a simple object using a set of rewriting rules or productions. The rewriting can be carried out recursively. **[Och98]**

2.6.2 Chaotic Fractals

It is non-linear fractals. This fractal type is connected with the theory of chaos, and its elements are obtained by a simple mathematical equation. For visualizing them, each point on the paper or screen is related to a certain

number e.g. in the case of the “Mandelbrot set” this is a complex number. This number is then iterated, that means it is used in a formula and the new number resulting from that is then again used in the same formula, which leads to the next iteration. This sequence of operations is "similar" to the work of the "copy-machine" of linear fractals - with regard to insertion. The insertion is repeated until the numerical values approach infinity, converge or fluctuate between several numbers. Depending on the result, the original point may be colored differently. **[Lor03]**

2.6.3 The “true” Mathematical Fractals

The development of this kind of fractals consists of simple rules a starting image, the so-called initiator, is replaced by another image, the so called generator. But nevertheless they are very complex and always strictly self similar: it does not matter which part we analyze, it always looks exactly like a scaled down copy of the whole set. The tools to create such fractals are called iteration and feedback: Iteration means that the procedure is repeated based on the result of the previous step. **[Lor03]**

2.6.4 Iterated Function System (IFS)

An iterated function system is a collection of affine transforms, it can be of any number of dimensions, but are commonly computed and drawn in two Dimensions. An IFS fractal is a solution to a recursive set equation. The fractal is made up of the union of several copies of itself, each copy being transformed by a function hence (functions system). The functions are normally contractive which means they bring points closer together and make shapes smaller. Hence the shape of an IFS fractal is made up of several possibly-overlapping smaller copies of itself, each of which is also made up of copies of itself, infinitum. This is the source of its self-similar fractal nature. **[And00]**

2.7 Partitioned Iterated Function System (PIFS)

In general, the theory of fractal compression is based on the contraction mapping in the mathematics of metric spaces. The Partitioned Iterated Function System (PIFS), which is essentially a set of contraction mappings, is formed by analyzing the image. Those mappings can exploit the redundancy that is commonly present in most images. This redundancy is related to the similarity of an image with itself. Fractal encoding techniques rely on the concept that parts of an image are very similar to other parts of the same image. More explicitly, the assumption is that one part of an image can be closely described by a scaled down copy of some other part of the image that has been translated and/or rotated according to a particular transformation. In preparation for encoding, an image is broken up into a set of non-overlapping "range" cells (i.e. image sub regions). The concept behind fractal encoding is that a set of "domain" cells can be mathematically transformed in some way to closely resemble every range cell of the image being compressed. This is pictorially illustrated in Figure (2.6).

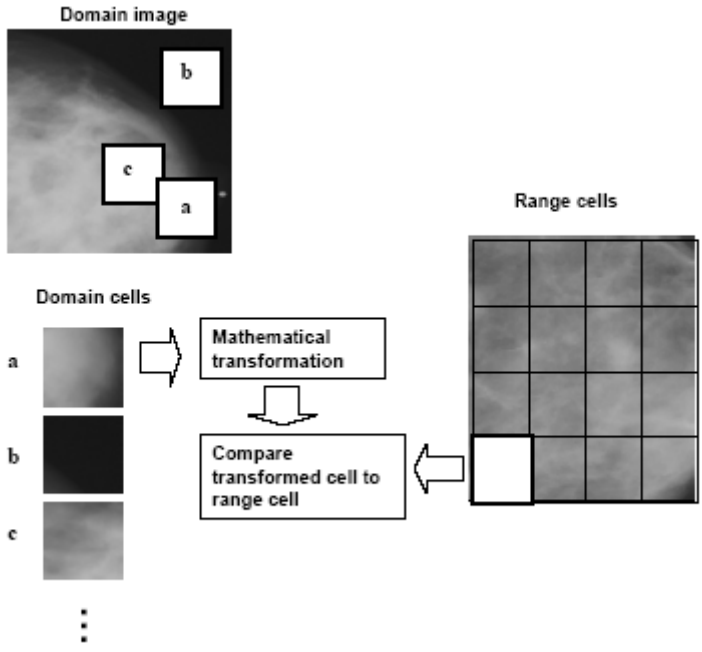


Figure (2.6) the basis of fractal encoding is the comparison of a set of domain cells transformed to mach the range cell [Fer02]

The image to be encoded is broken into a number of sub images or range blocks that cover an image completely in a non-overlapping manner. The domain blocks may arise from a similarly broken down image, or may consist of a number of unrelated sub images, or may be created by using overlapping regions of an image. Each of these domain cells will undergo a number of transformations such as rotations, shrinkage, or intensity modifications to more closely match a given range image. **[Fer02]**

Before the matching stage, domain blocks are transformed as follows:

- Down sampling each domain block.
- Geometric transformations, eight isometrics are considered, using equations (2.12.., 2.2.19)
- Scale and shift of luminance value that are computed according to the minimum mean squared error value. **[Rod95]**

Each range image has associated affine transformation

$$W_K \begin{bmatrix} u \\ v \\ z \end{bmatrix} = \begin{bmatrix} a_K & b_K & 0 \\ c_K & d_K & 0 \\ 0 & 0 & S_K \end{bmatrix} \begin{bmatrix} u \\ v \\ z \end{bmatrix} + \begin{bmatrix} e_K \\ f_K \\ O_K \end{bmatrix} \quad \ldots\ldots (2.7)$$

Where a_K, b_K, c_K, d_K, e_K, f_K represent the geometric transformation and S_K (scale), O_K (offset) are the affine transform coefficients. *u*, *v* are the pixel coordinates and *z* the gray level.

Where S can be computed as follow:

$$S = \frac{n\sum_{i=1}^{n} R_i D_i - \sum_{i=1}^{n} R_i \sum_{i=1}^{n} D_i}{n\sum_{i=1}^{n} D_i^{\,2} - \left(\sum_{i=1}^{n} D_i\right)^2} \qquad \ldots\ldots\ldots\ldots\ldots\ldots (2.8)$$

And O can be computed as follow:

$$O = \frac{\sum_{i=0}^{n-1} D_i^2 \sum_{i=0}^{n-1} R_i - \sum_{i=0}^{n-1} D_i \sum_{I=0}^{n-1} R_i D_i}{n\sum_{I=0}^{n-1} D_i^2 - \left(\sum_{i=0}^{n-1} D_i\right)^2} \quad \ldots\ldots\ldots\ldots (2.9)$$

So the transformations are applied to the domain cells to find the best map, in terms of minimizing an error metric, onto the range cell**. [Fis95]**

If the minimum error between the range cell and the closest domain cell is below the designated similarity threshold, the region is considered mapped and the location of the range cell, the index of the domain that maps to it, and the transformation parameters are recorded. If the error associated with the best match is greater than the similarity threshold, this range cell could be marked as not being matched or representing an anomalous region. **[Fer02]**

The error between the range block and the closest domain block is measured with sum of square error function (α^2): **[Fis95]**

$$\alpha^2 = \sum_{i=1}^{n} R^2{}_i + S^2 \sum_{i=1}^{n} D_i^{\,2} + O^2 n - 2S\sum_{i=1}^{n} R_i D_i - 2O\sum_{i=1}^{n} R_i + 2SO\sum_{i=1}^{n} D_i \;..\; (2.10)$$

Where:

- ***n*** is the number of pixels in each block.
- R_i Is the range block.
- D_i Is the transformed domain block.

In order to obtain high compression ratios, only a small number of blocks are allowed. Thus, the key point in fractal compression is to partition the image into a small number of blocks that are similar to other image parts under certain transformations. Many partitioning methods have been proposed for fractal image compression; among the most widely known and most successful are

Fisher's quad tree scheme, Fisher's and Menlove's horizontal–vertical Partitions, Fixed size square partitioning and Triangular Partitioning. **[Har00]**

2.8 Affine Transform

Applying an affine transformation to a uniformly distorted image can correct for a range of perspective distortions by transforming the measurements from the ideal coordinates to those actually used.

An affine transformation is an important class of linear 2-D geometric transformations which maps variables (*e.g.* pixel intensity values located at position (X_1, Y_1) in an input image) into new variables (*e.g.* (X_2, Y_2) in an output image) by applying a linear combination of translation, rotation, scaling operations**. [Fis94]**

2.9 Fixed size square blocks Partitioning

The simplest possible range partition consists of the fixed size square blocks. This type of block partition is successful in transform coding of individual image blocks since an adaptive quantization mechanism is able to compensate for the varying "activity" levels of different blocks, allocating few bits to blocks with little detail and many to detailed blocks. Fractal coding based on the standard block transform, in contrast, is not capable of such adaptation, representing a significant disadvantage of this type of block partition for fractal coding. This deficiency may be addressed by introducing adaptively to the available block transforms, but the usual solution is to introduce an adaptive partition with large blocks in low detail regions and small blocks where there is significant detail. There is, of course, a trade off between the lower distortion expected by adapting the partition to the image content, and the additional bits required to specify the partition details. **[Woh99]**

2.10 Fidelity Criteria

Fidelity means "how accurately the signal recovered in the received represents the one sampled in the sender".

Quality means "a composition of the fidelity and the delay used to replay the samples". **[Saf02]**

Methods for image quality evaluation can be classified as objective and subjective measures:

- **Objective measure:** By objective measures some statistical indices are calculated to indicate the reconstructed image quality. it includes :

1. **RMSE**: is found by taking the square root of the error squared divided by the total number of pixels in the image ("mean").

   The total error in N*M decompressed image can be defined as:

$$Totalerror = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \left[\left(f(i,j) - f'(i,j) \right) \right] \ldots (2.11)$$

   The RMSE is computed s follow:

$$RMSE = \sqrt{\frac{1}{N \times M}} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \left[\left(f(i,j) - f`(i,j) \right)^2 \right] \ldots (2.12)$$

   Where f (i, j) is the original image data and f '(i, j) is the reconstructed image value. **[Umb98]**

2. **PSNR**: Another quantitative measure is the peak signal-to-noise ratio (PSNR), based on the root mean square error of the reconstructed image. The formula for PSNR is given as follow:

$$PSNR = 10\log\left(\frac{(255)^2}{MSE}\right) \ldots. (2.13)$$

- **Subjective measure:** subjective measure viewers read images directly to determine their quality. Subjective evaluation by viewers is still a method commonly used in measuring image quality. The subjective test emphatically examines fidelity and at the same time considers image intelligibility. When taking subjective test, viewer's focus on the difference between reconstructed image and the original image, they notice such details where information loss cannot be accepted. **[Saf02]**

2.11 Performance Parameters

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

- **Compression Ratio** : is the ratio of the original uncompressed image file and the compressed file , its denoted by:

$$Compressio\ nRatio = \frac{Uncompress\ FileSize}{Compressed\ FileSize} = \frac{Size_U}{Size_C} \ \ldots (2.14)$$

And it is often written as $Size_U : Size_C$

This is called a "10:1 compression "or "10 times compression", or it can be stated as "compressing the image to 1/10 its original size. **[Umb98]**

Chapter One
General Introduction

1.1 Introduction

With the growth of the digital technology, the information can be transmitted, manipulated and stored in digital format. Obtaining and modifying digital information is much easier than before. So there is a strong need for efficient ways of transmission this information. Compression is an outstanding solution to this problem. **[Qia03]**

Data compression "***is the process of converting data files into smaller files for efficiency of storage and transmission"***. As one of the enabling technologies of the multimedia revolution, it is a key to rapid progress being made in information technology. Data compression treats information in digital form that is, as binary numbers represented by bytes of data with very large data sets. Large image files remain a major bottleneck in a distributed environment. Although increasing the bandwidth is a possible solution, the relatively high cost makes this less attractive. Therefore, compression is a necessary and essential method for creating image files with manageable and transmittable sizes. In order to be useful, a compression algorithm has a corresponding decompression algorithm that, given the compressed file, reproduces the original file. There have been many types of compression algorithms developed. These algorithms fall into two broad types, lossless algorithms and lossy algorithms. A lossless algorithm reproduces the original exactly. A lossy algorithm, as its name implies, loses some data. Data loss may be unacceptable in many applications. For example, text compression must be lossless because a very small difference can result in statements with totally different meanings. There are also many situations where loss may be either unnoticeable or acceptable. In image compression, for example, the exact reconstructed value of each sample of the image is not necessary. Depending on the quality required of the reconstructed image, varying amounts of loss of information can be accepted. **[Xia01]**

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 (Graphic User Interface). 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 $24 \approx 16.78$ 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**. [Sal04]**

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 reduction:

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 noticed 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 applications).

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 Color Models

There are some color space models including:

1.4.1 Red, Green, Blue (RGB)

RGB is perhaps the most widely used color system in image formats. 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. For 24-bit color, the triplet (0, 0, 0) normally represents black, and the triplet (255,255,255) represents white. When the three RGB values are set to the same value for example, (63, 63, 63) or (191,191,191) the resulting color is a shade of gray. **[Mur07]**

1.4.2 YUV

The YUV color space is used by the PAL (Phase Alternation Line), NTSC (National Television System Committee)**,** and SECAM (Sequential Color with Memory) composite color video standard, the Black and White System uses only the luminance (Y) information; color information (U and V) was added in such a way that a black and white receiver decode the additional color information to display the color picture.

The conversion from RGB to YUV color model can be accomplished using the following equations**: [San98]**

$$Y = 0.299 * R + 0.587 * G + 0.144 * B \quad (1.1)$$

$$U = -0.147 * R - 0.289 * G + 0.436 * B \quad (1.2)$$

$$V = 0.615 * R - 0.515 * G - 0.1 * B \quad (1.3)$$

The conversion from YUV to RGB color model can be accomplished using the following equations**:**

$$R = Y + 1.14 * V \quad (1.4)$$

$$G = Y - 0.395 * U - 0.581 * V \quad (1.5)$$

$$B = Y + 2.031 * U \quad (1.6)$$

1.4.3 YC_bC_r

The YC_bC_r color space was developed as part of ITU-R BT.601 during the development of the world wide digital component video standard. YC_bC_r is a scale down and offset version of YUV color space .Y is defined to have nominal 8-bit range of (16-235), C_b and C_r are defined to have a nominal range of (16-240). YC_bC_r Used in image compression (e.g. JPEG format). **[Itu02]**

Conversions the image from RGB to YC_bC_r color model can be accomplished with the following equations. [**Shi00]**

$$Y = (0.257 * R + 0.504 * G + 0.098 * B) + 16 \quad (1.7)$$

$$C_b = (-0.148 * R - 0.291 * G + 0.439 * B) + 128 \quad (1.8)$$

C_r = (0.439 *R - 0.368* G - 0.071 *B) + 128 ……. (1.9)

Conversions the image from YC_bC_r color model to *RGB* can be accomplished with the following equations*:*

R = (1.164* Y + 0.000* C_b + 1.596* C_r) - 16 ……... (1.10)

G= (1.164*Y - 0.392 * C_b - 0.813* C_r) - 128 ……..…... (1.11)

B = (1.164* Y + 2.017* C_b - 0.000* C_r) - 128 ……..…... (1.12)

1.4.4 YIQ

It is NTSC Transmission Color Coordinate System. In the development of the color television system in the United States, NTSC formulated a color coordinate system for transmission composed of three values, Y, I, Q. The Y value, called luma, is proportional to the gamma-corrected luminance of a color. The other two components, I and Q, called chroma, jointly describe the hue and saturation attributes of an image. The reasons for transmitting the YIQ components rather than the gamma-corrected components directly from a color camera were two fold: The Y signal alone could be used with existing monochrome receivers to display monochrome images; and it was found possible to limit the spatial bandwidth of I and Q signals without noticeable image degradation. As a result of the latter property, a clever analog modulation scheme was developed such that the bandwidth of a color television carrier could be restricted to the same bandwidth as a monochrome carrier.

The YIQ transformations are given by: **[Pra01]**

Y = 0.298 *R + 0.586* G + 0.114 *B…….. (1.13)

I= 0.595 * R - 0.274 *G - 0.439 *B……..……...….. (1.14)

Q = 0.211* R - 0.522 * G +0.311* B…….…..……. (1.15)

$R = 1.000 * Y + 0.956 * I + 0.620 * Q$……..…….....….. (1.16)

$G = 1.000 * Y - 0.271 * I - 0.648 * Q$ ……..….. (1.17)

$B = 1.000 * Y + 1.105 * I + 1.702 * Q$ …….……...……. (1.18)

1.4.5 Cyan, Magenta, Yellow, Black (CMYK)

The color printing and color photographic processes are based on a subtractive color representation. In color printing, the linear RGB color components are transformed to cyan (C), magenta (M), and yellow (*Y*) inks, which are overlaid at each pixel on a, usually, white paper. The simplest transformation relationship is:

C=1.0-R…… (1.19)

M=1.0-G…….…… (1.20)

Y=1.0-B……. (1.21)

Where, the linear RGB components values are over [0.0, 1.0].

R=1.0-C…… (1.22)

G=10-M…… (1.23)

B=10-Y…… (1.24)

In high-quality printing systems, the RGB-to-CMY transformations, which are usually proprietary, involve color component cross-coupling and point nonlinearities.

To achieve dark black printing without using excessive amounts of CMY inks, it is common to add a fourth component, a black ink, called the key (K*)* or black component. The black component is set proportional to the smallest of the CMY components as computed using the following transformation:

C=1.0-R-uK_b …… (1.25)

M=1.0-G-uK_b …… (1.26)

$$Y=1.0-B-uK_b \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (1.27)$$

$$K=bK_b \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (1.28)$$

Where $K_b = MIN\{1.0-R, 1.0-G, 1.0-B\}$

And $0.0 \le u \le 1.0$ is the undercolor removal factor and $0.0 \le b \le 1.0$ is the blackness factor. **[Pra01]**

1.4.6 Hue, Saturation , Value (HSV)

The HSV (Hue, Saturation, and Value) model, also known as HSB (Hue, Saturation, and Brightness) or HSI is (Hue, saturation and intensity).

It can be conceived as a property of the surface reflecting or transmitting the light. For example, a blue car reflects blue hue. Moreover it is also an attribute of the human perception. The hue which is essentially the chromatic component of our perception may again be considered as weak hue or strong hue. The colorfulness of a color is described by the saturation component. For example, the color from a single monochromatic source of light, which produces colors of a single wavelength only, is highly saturated, while the colors comprising hues of different wavelengths have little chromic and have less saturation. The gray colors do not have any hues and hence they are zero saturation or unsaturated. Saturation is thus a measure of colorfulness or whiteness in the color perceived. The lightness (L) or intensity (I) or value (V) essentially provides a measure of the brightness of colors. This gives a measure of how much light is reflected from the object or how much light is emitted from a region. It is proportional to the electromagnetic energy radiated by the object. The luminosity (or intensity) essentially helps human eye to perceive color. **[Ach05**]

1.5 BMP file

Bitmap files are especially suited for the storage of real-world images; complex images can be rasterzed 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]**

1.6 Literature Survey

1. ***Roche S. and Dugelay J-L***, **[Roc95]:**

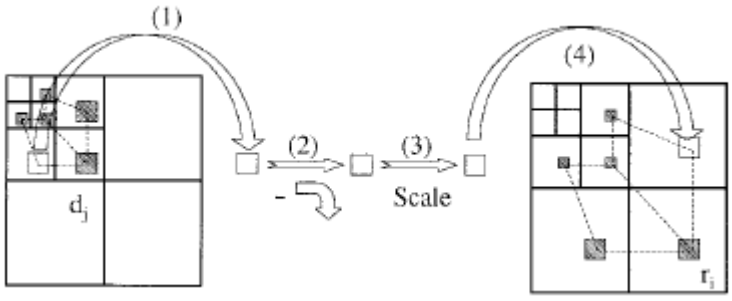
   In this research, the Iterated Function System (IFS) was studied focusing on two aspects, the first one concerned the definition of the contractive constraint during the coding stage in order to ensure the convergence of the iterative decoding process, this aspect is important in the estimation of the optimum IFS code of an image, the second one concerned the choice of the initial image for starting the decoding stage, so the optimal choice for the initial image in the decoding process was considered in order to increase the decoding speed.

2. *Zaki T.* ,[Zak97]:

In this research, the lossless data compression was applied for multi media files (i.e. such as image, audio, text). The implemented data compression algorithms are divided into three main classes: Run Length Encoding (RLE) techniques, statistical techniques, and dictionary techniques. RLE depends on removing the repetition of consecutive symbols. A slight modification is proposed of repetitions used in standard RLE.

3. *Li J. and Jay C. K.*, [Lij99]:

In this research, a hybrid wavelet-fractal coder (WFC) for image compression was proposed. The WFC uses the fractal contractive mapping to predict the wavelet coefficients of the higher resolution from those of the lower resolution and then encode the prediction residue with a bitplane wavelet coder. The fractal prediction was adaptively applied only to regions where the rate saving offered by fractal prediction justifies its overhead. A rate-distortion criterion was derived to evaluate the fractal rate saving and used to select the optimal fractal parameter set for WFC.



The performance of the proposed WFC has been compared with several typical fractal and wavelet coders. They include the block based fractal coder with biorthogonal wavelet and zerotree coding. The test images used in the experiment is the Lena of size 512*512. The performance of block based fractal coder is not good even compared with JPEG. Although it outperforms JPEG at low bit rates, but it can hardly compete with the state of the art wavelet coders such as EZW by using the biorthogonal wavelet and zerotree.

4. ***Al-Dulaimy A. A.***, **[Ald00]**:

In this research A development of Fractal Image Compression (FIC) was introduces. The main scheme of fractal compression was implemented, which lead to a good compression performance with reduction in coding time. A speeding up operation based on a new mathematical approach for determining the IFS codes between the range and domain blocks was introduced.

5. ***Khalifa O. and Dlay S.*** **, [Kha00]**:

A fractal coding scheme in wavelet transform domain was presented in this research. The combination and links between these techniques are investigated. They analyzed the capability of fractal coders to predict wavelet coefficients where higher frequency sub bands of higher levels coefficients of wavelet pyramid are described by filtering, decimating and scaling the coefficients from lower levels higher frequency bands. This work explained the improved fractal coders in wavelet domain and proposes a new mixed fractal wavelet compression scheme. It had a significant advantage of simplifying decoding by avoiding iterations and a time saving as compared to exhaustive searching. Also, it provided new thought into the concept of fractal-wavelet coding.

This research shows that the proposed method is highly efficient and fractal coding in wavelet domain exploit repetition of patterns at different scales. It significantly reduces the block artifacts and permits reconstruction in finite number of iterations, thus it reduces the number of domains to be searched and the number of computations. Using large range blocks ensure high compression ratio to be large as possible. Although it result in a high compression ratio but it has a draw back in peak signal to noise ratio, such that for every tested image, the PSNR was not more than 30, i.e. it produced a good image compression ratio with high degradation in mean square error.

6. ***Kassim B.,*** **[Kas02**]:

In this research, a new color image scheme based on truncation coding methods were proposed, a multilevel block truncation codes based on wavelet transform were applied, the vertical and horizontal Haar filters were composed to construct four 2-dimensional filters, such filters were applied directly to image to speed up the implementation of Haar wavelet transform. Also other scheme was introduced such as: Quadtree and adaptive HV partitioning.

7. ***Engel D.,*** **[Eng03]:**

In this research, the approaches of object based and adaptive image compression using wavelets were presented. After discussing the basic principles of the wavelet transform and Zero tree encoding in general, a recent propositions of how to adapt these techniques to arbitrarily shaped objects and a comparison of different methods and implementations were presented. This work showed how object based adaptively can be achieved and how well it performed as compare to other methods in an object based framework. It also presented an image coding framework that use wavelet transformation in order to increase compression performance.

1.7 Aim of thesis

Both wavelet transforms and fractal were utilized individually to compress images. In this research, both methods will be assembled in hybrid compression system to compress images. This will help to overcome some of problems which faced each method separately. Also the hybrid system will take the advantage of both methods.

1.8 Thesis Layout

In addition to chapter one that gives a general introduction to the compression process, this thesis includes other four chapters:

- **Chapter 2:** ***Theoretical Background***

  Describe the basic backgrounds of the image compression methods including the lossy and lossless methods, And to penetrate deeply into the conceptual reviews about wavelet transform, fractal encoding and Run Length Encoding (RLE) methods.

- **Chapter 3:** ***The Proposed Image Compression Methods***

  Provides the proposed image compression system design, the implemented algorithms including: Haar wavelet transform, Fractal image coding and other algorithms such as: Color transform, Quantization, RLE and S-Shift coding with details for each one.

- **Chapter 4:** ***Performance Measures and Test Result***

  Describe the performance measures and gives the results of some tests applied on some samples.

- **Chapter 5:** ***Conclusions and Future Work***

  Provide the conclusion about this thesis and some suggestion scheme for the future work.

Chapter three
The Proposed Image Compression Methods

3.1 Introduction

Recently, in order to achieve satisfactory image and video quality and fast transmission in sometimes very low bandwidth channels, the demand for high compression ratios and fast speed in the coding and decoding procedure has been increased. An algorithm for very high compression of images is proposed in this chapter. First of all the image is decomposed through a wavelet transform. Then the low frequency part of the image is coded by using a fractal coding technique, the other parts of wavelet transform is coded by using RLE method. That led to good image compression performance with respect to peak signal to noise ratio and compression ratio.

Wavelets are mathematical functions that provide very good quality compression at very high ratios because of their ability to decompose signals into different scales or resolutions. The Haar wavelet transform splits an image into a low-resolution version of the image and a series of images that contain the finer details of the image. Because of this characteristic is well suited for applications where scalability and/or intelligent signal degradation is required (low bit rate image transmission trough a low bandwidth network for example). Though not a worldwide approved standard at this moment over the last years more and more organizations and groups are focusing towards the inclusion of wavelet transforms in new image coding schemes**. [And00]**

Fractal compression of images attempts to exploit self-similarity in images. Recent results show that performing the fractal coding algorithm in the wavelet domain gives better compression results compared to classic compression schemes as well as simple fractal coding schemes because it diminishes the blocking effect and leads to much more efficient and high quantization of the coding coefficients. However, the simplicity in the codebook generation and the

long encoding times lead to rather inefficient implementations compared to wavelet methods. **[And00]**

3.2 Compression System

The original image will be colored image which can be represented as three band monochrome image data, Typically, colored image are represented by using 3-principal colors (Red, Green and Blue) for that it's called RGB image **[Umb98].**

The encoding stage involves the following steps that are summarized as following:

1. The first step of the encoding system is the preprocessing stage that involved transforming the (RGB) to (YC_bC_r) color model. The goal of preprocessing is to prepare the image for encoding process by eliminating any irrelevant information.
2. Applying Haar Wavelet Transform on Y component.
3. The approximation subband, which is LL, will be compressed by using fractal encoding. This is due to that the LL band contains most of image information. another reason is that the fractal coding algorithm gives better compression results compared to classic compression schemes as well as simple fractal coding schemes because it minimize the blocking effect and leads to much more efficient and high quantization of the coding coefficients.
4. Applying the uniform quantization on the details subband (i.e. LH, HL and HH) bands to reduce the number of bits needed to represent the coefficients of these bands.
5. Applying the Mapping Process on the quantized parameters of the wavelet sub bands (LH, HL and HH) and encoded them by applying RLE algorithm.

6. The final encoding step was applying the S-shift optimizer and S-shift encoder then saving the results.
7. Also applying Fractal coding algorithm to C_b and C_r components by using the same fractal coding steps that applied on LL sub band with a difference in domain creation module, then saving the result as final step.

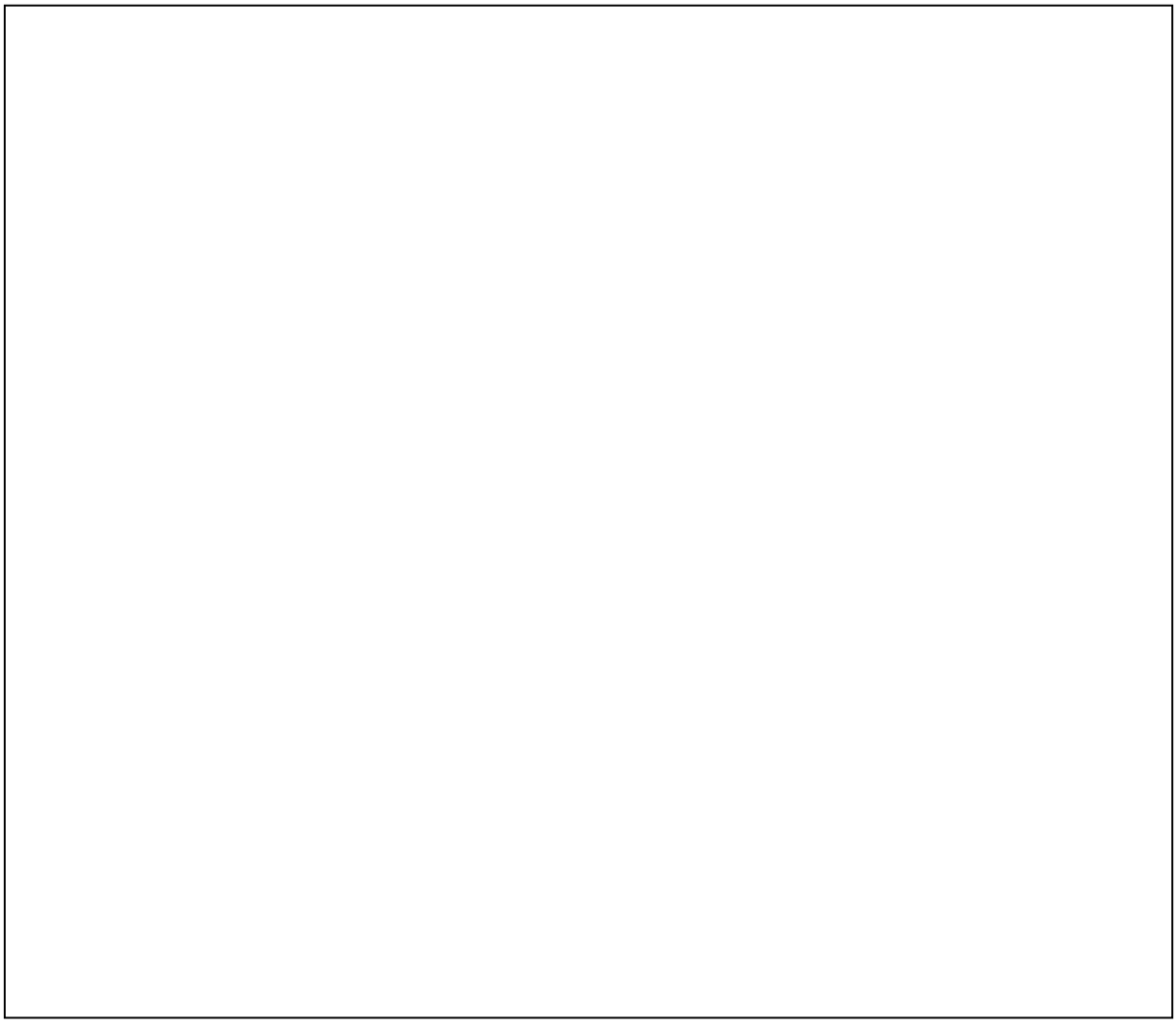


Figure (3.1) Block diagram of the proposed compressing method

3.2.1 Color Transform:

The color signal can be seen as a summation of light intensities of three primary wavelength bands. Given 24 bits/pixel RGB, we can find the Y , C_b and C_r values using equations (1.7…1.9). Algorithm (3.1) shows the steps of the implemented algorithm.

Algorithm (3.1) the Color Transform

- **Input:** pic( ) = *RGB* Image, (*W*×*H*)
- **Output:** pic2( ) = *YCbCr* Image (*W*×*H*)
- **Method:**

For each Column y (y=0, 1… h-1)

For each row x (x=0, 1… w-1)

- Compute Y component from pic(x,y)

Y= (0.257 * pic(x, y).R + 0.504 * pic(x, y).G + 0.098 *pic(x, y).B) + 16

- Compute Cb band from pic(x,y)

C_b = (-0.148 * pic(x, y).R - 0.291 * pic(x, y).G + 0.439 *pic(x, y).B) + 128

- Compute Cr band from pic(x,y)

C_r = (0.439 * pic(x, y).R - 0.368 * pic(x, y).G - 0.071 *pic(x, y).B) + 128

End loop x

End loop y

3.2.2 Forward Haar Wavelet Transform

In the proposed work, the first step of encoding stage was applying the Haar Wavelet Transform on Y component. The Haar Wavelet function consists of both: low pass and high pass filters. The input for this algorithm was the Y component which is 2D array $(W \times H)$, and the out put was the wavelet coefficients in 4-sub bands (LL, LH, HL, and HH) which represented also in 2D array, See Algorithm (3.2).

Algorithm (3.2) the Forward Haar Wavelet Transform

- **Input:** pic2( )= YC_bC_r Image $(W \times H)$

  Wlevel= level of wavelet transform
- **Output:** Wimage( ) = 2D array, wavelet coefficients $(W \times H)$
- **Method:**

  Set pic2 ( ) to wimage ( )

  Set wtemp = W

  Set htemp = H

  For each wavelet level I (I = 1… wlevel)

  - Set w2 = wtemp / 2
  - Set h2 = htemp / 2

  For each Column y (y=0, 1… h2-1)

  - Set yy = 2 * y
  - Set Yp = yy + 1

  For each row x (x=0, 1… w2-1)

  - Set xx = 2 * x
  - Set Xp = xx + 1

- Compute tempimage for (x, y) coordinate

  tempimage = (wimage (xx, yy) + wimage (Xp, yy) + wimage (xx,Yp) + wimage (Xp, Yp)) /4

- Compute tempimage for (x + w2, y) coordinates

  tempimage =(wimage (xx, yy) + wimage (Xp, yy) – wimage (xx,Yp)- wimage (Xp, Yp)) / 4

- Compute tempimage for (x, y + h2) coordinates

  tempimage= (wimage (xx, yy) - wimage (Xp, yy) + wimage (xx, Yp)- wimage (Xp, Yp)) / 4

- Compute tempimage for(x + w2, y + h2) coordinates

  tempimage= (wimage (xx, yy) - wimage (Xp, yy) – wimage (xx,Yp) + wimage (Xp, Yp)) / 4

End loop x

End loop y

For each Column y (y=0…h - 1)

For each Row x (x= 0…w - 1)

- Set wimage(x, y) = tempimage(x, y)

End loop x

End loop y

- Set wtemp = w2
- Set htemp = h2

End loop I

3.2.3 Fractal Encoding

The first step in fractal encoding is to partition the image into a set of non-overlapping regions referred to as **Range** Blocks (R_1…., R_n) where n is the number of range block (see algorithm (3.3)), and a set of overlapping regions referred to as **Domain** blocks (D_1…., D_m)where m is the number of domain blocks (see algorithm (3.4), algorithm (3.5)), both of domain and range blocks are of the same size (B_S**)**. After partitioning stage, a matching technique will be start such that, for each range block search the best domain block which satisfy the best map to this range block with minimum error(see algorithm (3.6)). The results will be (S, O, Sym, X and Y coordinates of Di), which called IFS coefficients, see figure (3.2).

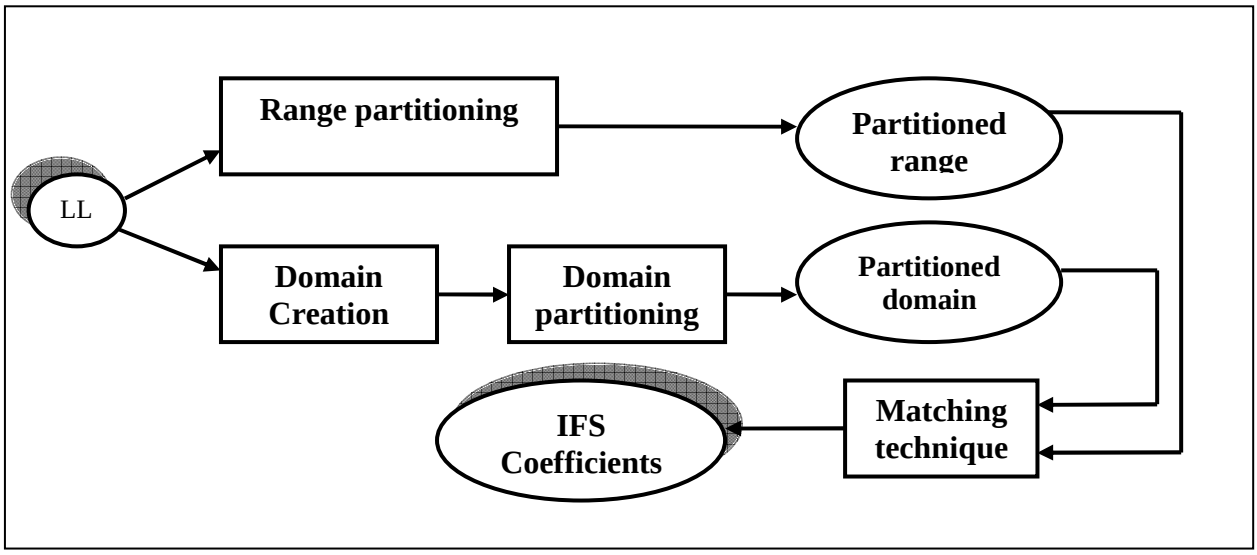


Figure (3.2) Fractal Encoding stages

3.2.3.1 Range partitioning

In this research, a fixed size partitioning was applied because it requires less computational time than other partitioning methods. This was done by choosing the block size as an input, so the image will be divided into non overlapped Range blocks each of them is of size $B_S \times B_S$. The choice of the block size is affect on the quality of the reconstructed image.

If the block size was big then the time consuming will be reduced while the quality of the reconstructed image will be decrease (i.e. the largest image region leads the minimum number of partitions). But if the block size was small then the time consuming will be increased and the quality will be increased because the image quality may be improved by more searching on overall the image (i.e. the smallest image region leads to determine the maximum number of partitions). In this research the block size takes value 2, 4 and 8. The LL sub band diminutions are (Widd × Hgtt)

Algorithm (3.3) Range Partitioning

- **Input :** LL( ) = LL sub band

  B_S =block size

- **Output:** Rinfo ( ) = 1D array of range blocks coordinates of size R_n

  R_n =number of range blocks

- **Method**:

  Set ys = 0

  Set R_n=0

  Set Widd =w / 2^{Wleve}

  Set Hgtt = h / 2^{Wleve}

➡

While (ys + B_S -1) < widd

- Set xs = 0

While (xs + B_S -1) < hgtt

- Set Rinfo of x coordinate to xs
- Set Rinfo of y coordinate to ys
- Increment R_n by "1"
- Increment xs by "B_S"

End Loop xs

- Increment ys by "B_S"

End loop ys

3.2.3.2 Domain Creation

In this module ,a two dimensional array was created called (domain), the domain matrix has different width and height since its size was reduced to %25 from original image size, it is done by replacing each four pixel (2×2) by one pixel which is the average of these four pixels. The new image will be of size (Wh × Hh) i.e.:-

$$Wh = (Widd / 2) \quad \text{(3.1)}$$

And

$$Hh = (Hgtt / 2) \quad \text{(3.2)}$$

Algorithm (3.4) Domain Creation

- **Input:** LL( ) = LL sub band
- **Output**: domain( ) =down sampled (Wh × Hh)
- **Method:**

Set ys = 0

Set dy = 0

Set sum = 0

While ys < Hh

- Set xs = 0
- set dx = 0

While xs < Wh

- Compute the sum for every four pixels:

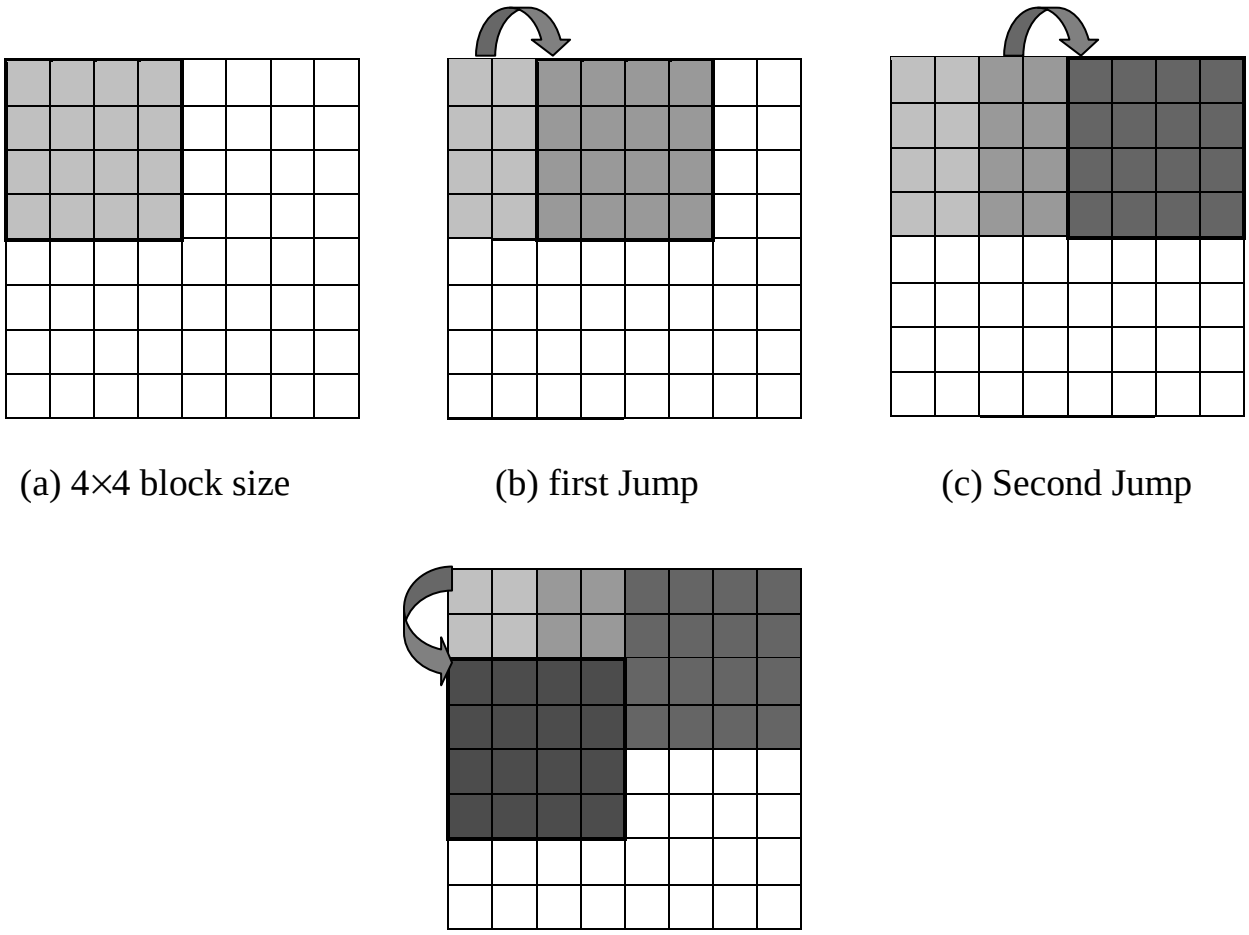
  Sum= LL (xs, ys) + LL (xs + 1, ys) + LL (xs, ys + 1) + LL (xs +1, ys + 1)
- Set domain (dx, dy) = sum / 4
- Increment dx by "1"
- Increment xs by "2"

End loop xs

3.2.3.3 Domain Partitioning

In this module the fixed size partitioning also used to partition the domain space into **domain blocks,** with the same size of range blocks that's $B_S \times B_S$, but in overlapping way. The overlapping blocks lead too many possible domain blocks, which lead to obtain a good approximation. The jump step indicates the number of pixel to be start from the previous one, this is include the X-coordinate and Y-coordinate. For example, if the image size was 8×8, and the

B_S=4 and the jump step =2, then the overlapping blocks will be horizontally then vertically partitioned, See figure (3.3).



(a) 4×4 block size (b) first Jump (c) Second Jump

(d) Third jump

Figure (3.3) domain partitioning for block size =4, jump step =2

The jump step must be greater than zero and less than or equal to the block size. If the jump step was greater than block size, then this will lead to non overlapping blocks.

If the jump step was small, then the domain blocks will be increased and that leads to a good approximations and high quality. But it will lead to high encoding time, since searching a large number of domain blocks is time consuming. Algorithm (3.5) shows the steps of domain partitioning.

Algorithm (3.5) Domain Partitioning

- **Input:** domain( ) = 2D array (Wh × Hh)

  JStep = Jump step

- **Output:** Dinfo( ) = 1D array of domain blocks coordinates of size D_n

  D_n =number of domain blocks

- **Method:**

  Set Ys =0

  Set D_n=0

  While (ys + B_S-1) < Hh

  - Set xs = 0

    While (xs + B_S-1) < Wh

    - Set Dinfo of x coordinate to xs
    - Set Dinfo of y coordinate to ys
    - Increment D_n by "1"
    - Increment xs by " JStep"

    End Loop xs

  - Increment ys by " JStep"

  End Loop ys

3.2.3.4 Matching Technique

The image to be encoded was partitioned into non overlapping range blocks "R" and overlapped domain blocks "D". The encoder then finds a best domain block D of the same image for every range block. In other words, trying to find a part of the image that looks similar to R_i. Such that a transformation of D_j is a

good approximation of the range block R_i, then compute the scale and offset values to find the error between the transformed domain block and the range block.

If the best match between the range block and the transformed domain block still has an error measurement that is greater than the similarity threshold, the algorithm will continue with other domain cell and re-evaluation continues similarly with each of these blocks. Until finding the best domain block with minimum error. Algorithm (3.6) shows the steps of matching technique, the (original) image referred to the LL sub band, C_b or C_r components.

Algorithm (3.6) Matching Technique

- **Input:** Rinfo ( ) = 1D array of range blocks coordinates
  Dinfo ( ) = 1D array of domain blocks coordinates
  Original ( ) = 2D array original pic
  R_n = number of Range blocks
  D_n = number of Domain blocks
  B_S = block size
- **Output:** S, O, X, Y, Sym =IFS Coefficients for each range block
- **Method:**

For Each Range block R_i (i=0, 1… R_n)

- Get range block coordinates from Rinfo( )
- Get range block from original
- Set the range block rangetemp ( ) =original ( )

For Each Domain block D_j (j=0, 1… D_n)

- Get domain block coordinates from Dinfo( )
- Set domain block domaintemp ( ) =Domain ( )

For Sym=0 to 7

- Set D as Symmetric affine transform of domaintemp using equations (3.3 ... 3.10).
- Compute the "**S**" value of (D, rangetemp) using equation (2.8).
- Compute the "**S quantized**" using equation (3.11).
- Compute the "**O**" value of (D, rangetemp) using equation (2.9).
- Compute the "**O quantized**" using equation (3.12).
- Compute the Error value between (D, rangetemp) using equation (2.10).
- If the Error < Minimum Error then
  - Set Minimum Error = Error
  - Save S, O, Sym, X, Y coordinates for the transformed Domain block position.

End if

End loop Sym

End loop j

End loop i

3.2.3.5 Affine Transform

The Affine transformation is a composition of identity, reflection, rotation, with three different degrees. **[Fis95]**

1. Identity case:

D(x, y) =domain(x, y)…….……..……... (3.3)

2. Rotation +90:

   D(x, y) =domain(y, B_S-x)…….……..…….. (3.4)

3. Rotation +180:

   D(x, y) =domain (B_S-x, B_S-y)…….……..…… (3.5)

4. Rotation +270:

   D(x, y) =domain (B_S-y, x)……..… (3.6)

5. Reflection case:

   D(x, y) = domain (B_S-x, y)……. (3.7)

6. Reflection and Rotation -90:

   D(x, y) = domain (B_S-y, B_S-x)……. (3.8)

7. Reflection and Rotation -180:

   D(x, y) =domain (x, B_S-y)…… (3.9)

8. Reflection and Rotation -270:

   D(x, y) =domain (y, x)……..……..……. (3.10)

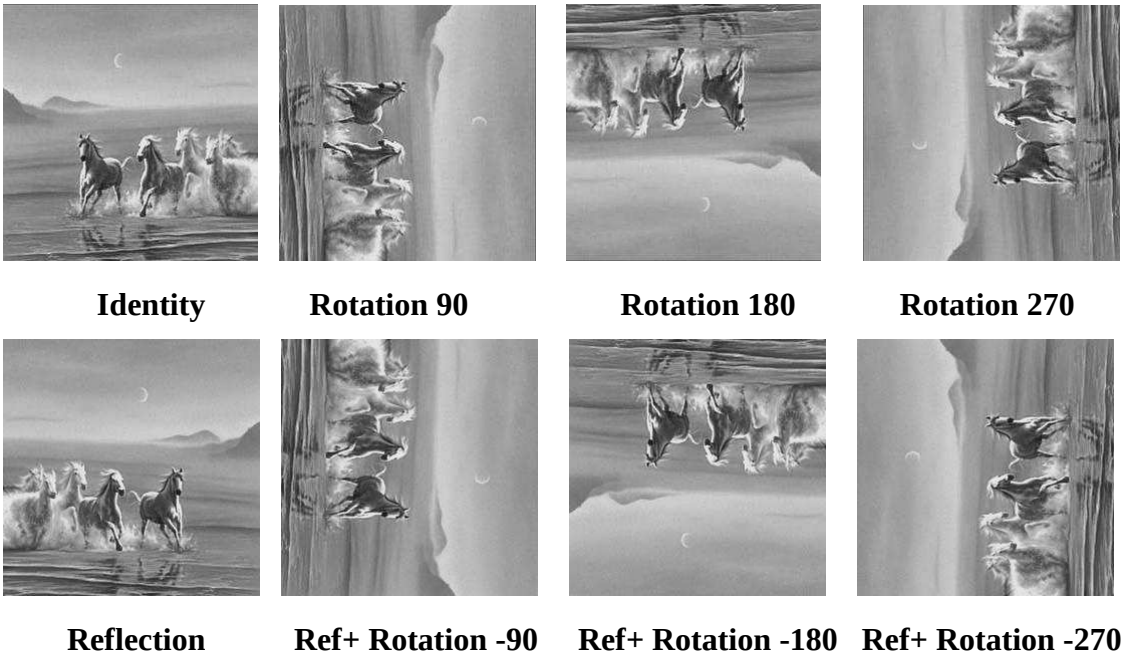


Figure (3.4) The eight isometrics transform

3.2.3.6 Fractal Quantization

After getting the IFS coefficients for each range block, these coefficients are quantized by a method which tends to suppress higher-frequency elements and reduce the number of bits required for each coefficient (i.e. S, O). So uniform quantization was performed to quantize the scale and offset coordinates. the scale coefficient was quantized using following equation:

$$Sq = round\left(\frac{2^{NOBS-1}-1}{MaxS \times 2}\right) \times S \quad \text{…….. (3.11)}$$

While the Offset coefficient was quantized using following equation:

$$Oq = round\left(\frac{O - MinO}{MaxO - MinO} \times \left(2^{NOBO} - 1\right)\right) \quad \text{…..… (3.12)}$$

Where:

- **S** is the Scale value
- **Sq** is the quantized Scale value
- **O** is the Offset value
- **Oq** is the quantized Offset value
- **NOBO** is the number of bits assigned to Offset
- **NOBS** is the number of bits assigned to Scale
- **MaxS** is the maximum value for the Scale coefficient
- **MaxO** is the maximum value for the Offset coefficient
- **MinO** is the minimum value for the Offset coefficient

3.2.4 Fractal Encoding for C_b and C_r

The same fractal matching technique was used for both C_b and C_r but with difference in domain creation module; Such that instead of down sampling each

four pixels (i.e. each 2 adjacent pixels) the domain partitioned for each 16 pixels (i.e. each 4 adjacent pixels) to one pixel whose value is the average of these 16 pixels, this was due to that C_b and C_r components have less information than Y component.

$$sum = \sum_{j=ys}^{ys+4} \sum_{i=xs}^{xs+4} a(i, j) \quad \ldots\ldots\ldots\ldots\ldots\ldots (3.13)$$

Algorithm (3.7) Domain Creation for Cb, CR

- **Input:** a( ) = (W× H) // (a) is either C_b or C_r
- **Output**: domain ( ) = down sampled (Wh × Hh)
- **Method:**

Set ys = 0, dy = 0, sum = 0

While ys < Hh

- Set xs = 0
- set dx = 0

While xs < Wh

- Compute the sum for every 16 pixels using equation (3.13)
- Set domain (dx, dy) = sum / 16
- Increment dx by "1"
- Increment xs by "4"

End Loop xs

- Increment dy by " 1"
- Increment ys by " 4"

End Loop ys

3.2.5 Quantization

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

- **Uniform Quantization:** if the input range is divided into levels of equal spacing. It can be specified by its lower bound and the step size.
- **Non Uniform Quantization**: else where.

The Uniform Quantization was used to code the transformed wavelet coefficients sub bands (LH, HL and HH) to achieve better compression result. It was also applied to reduce the number of bits need to store these sub bands coefficients**. [Kum03]**

In this research a quantization function was used to determine the quantization step "**Qstep"** for each coefficients c(x, y) as follow:

$$I(x, y) = round\left(\frac{c(x, y)}{Qstep}\right) \quad \text{..... (3.14)}$$

Where the **Qstep** for **LH** and **HL** sub bands is:

$$Qstep = A \times R^{(n-1)} \quad \text{..... (3.15)}$$

While the **Qstep** for **HH** sub band is:

$$Qstep = A \times R^{(n-1)} \times B \quad \text{..... (3.16)}$$

Where **n** is the number of wavelet levels.

3.2.6 Mapping Process

Before the mapping process, a transformation from two dimensional arrays to one dimensional vector was applied on the detailed subbands (i.e. LH, HL and HH), see figure (3.5). In mapping process each value will be mapped to be positive, the following mapping equation had been used to convert the signed integer into positive integers:

$$X' = \begin{cases} 2X & if x \geq 0 \\ 2|X| + 1 & if x < 0 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.17)$$

Where, X represents the signed integer value of the quantization index. This type of the mapping insure that all coefficients values are mapped to positive integers, and to keep the optimal number of bits needed to be used by RLE and shift coder as small as possible , so that the histogram of the coded coefficients is highly peaked into zero.

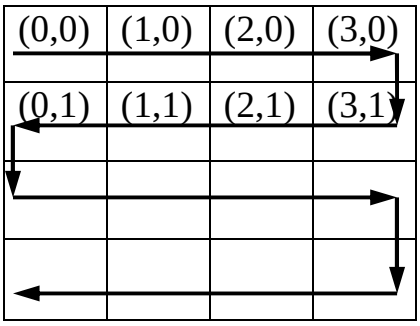


Figure (3.5) convert 2-D array to 1-D vector

Algorithm (3.8) Mapping Process

- **Input:** a( ) = 2D array // (a) indicates the (LH, HL and HH)
  (x1, x2, y1, y2)= the sub band coordinates
- **Output**: z( ) =1D array indicates the 1-Dvector
  Z_n = length of the vector
- **Method:**

Set $Z_n = 0$

Flag= 0

For each Column y (y=y1... y2)

- If flag = 0 Then
  Xs=x1, xe = x2, stp =1, flag=1
  Else
  Xs=x2, xe = x1, stp =-1, flag=0
  End if
- For each row x (x=xs…xe step stp)
  - If (a(x, y)) >= 0 Then
    Z (Z_n) =2*a(x,y)
    Else
    Z (Z_n) = 2 * (Abs (a(x, y))) + 1
    End If
  - Increment Zn by "1"

End loop x

End loop y

3.2.7 RLE

RLE consists of the process of searching for repeated runs of zeros in an input stream, and replacing them by a run count, such that, it first check the start type, i.e., the first value in the input stream, if it was zero then evaluate the start type to zero followed by its count, if it was non zero then evaluate the start type to one followed by the non zero number. The algorithm continue searching for each adjacent zeros and replacing them with its count. See figure bellow:

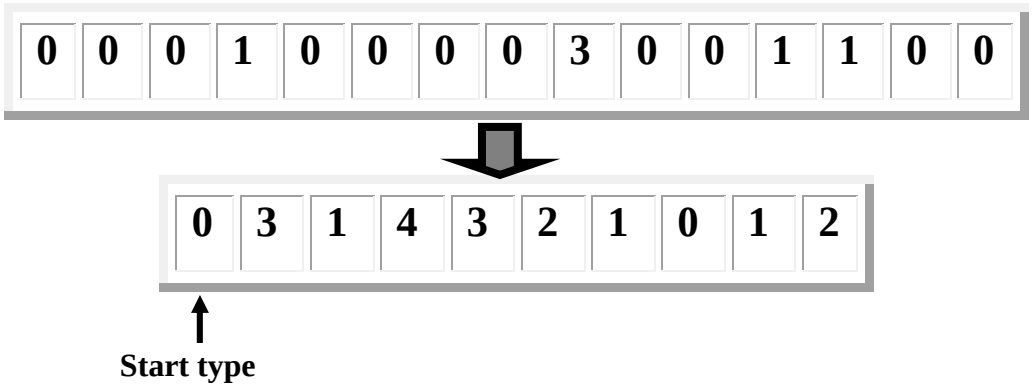


Figure (3.6) Examples of the RLE

The out put will be:

1. Bit "0" (because the first number is zero).
2. Word "3" (three zeros).
3. Word "1" (the non zero number is one).
4. Word "4" (because there are four zeros after number one).
5. Word "3" (the non zero number is three).
6. Word "2" (because there are two zeros after number three).
7. Word "1" (the non zero number is one).
8. Word "0" (because there is no zeros after number one).
9. Word "1" (the non zero number is one).
10. Word "2" (because there are two zeros after number one).

Algorithm (3.9) RLE

- **Input:** z( ) =1D array indicates the Zigzag vector
  Zn= length of the zigzag vector
- **Output**: enc( ) =1D array of encoded vector
  encl= length of the encoded vector
- **Method:**

Set encl = 0
If (z(0) = 0) Then
enc (0) = 0: typ = 0: counter = 1: encl = 1
Else
enc (0) = 1: enc(1) = z(0): typ = 1: encl = 2
End If
For each I (I = 1 ...Zn - 1)

- If (z(I) = 0) Then
  If typ = 0 Then
  - Increment counter by "1"

  Else
  - Set counter = 1
  - Set typ = 0

  End If
- Else
  If typ = 0 Then
  - Set enc (encl) = counter
  - Increment encl by "1"
  - Set enc (encl) = z (I)
  - Increment encl by "1"
  - Set typ = 1

Else

- Set enc (encl) = 0
- Increment encl by "1"
- Set enc(encl) = z(I)
- Increment encl by "1"

End If

End If

- If (I = Zn - 1) Then

If (z (I) = 0) Then

- Set enc (encl) = counter
- Increment encl "1"

End If

End loop I

3.2.8 S-Shift Optimizer

The mechanism applied to compute the optimal length "b", in bits of the shift codewords is based on scanning all possible lengths, starting from "2" bits and proceeding more till the numbers "k"; which represent the minimum number of bits required to represent the maximum coefficient value "L" in the set of RLE parameters. This number "k" is considered as the length "in bits" of the second "auxiliary" codeword. The scan method was applied to test all possible value of bits that can be assigned to the first "shortest" codeword, so the length range of the first codeword is [2, k].

The scanning mechanism implies iteration over all possible numbers of bits "b" that can be assigned to the first codeword; for each possible number of bits "b" the total number of bits "T" required to encode the encoded coefficients is determined by the following equation:

$$K = \lceil Log_2(L) \rceil \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(3. 18)}$$

$$T = b\sum_{i=0}^{L} His\,(i) + K \sum_{i=2^K-1}^{L} His\,(i) \ldots\ldots\ldots \text{(3. 19)}$$

Then the value of "b" which leads to the lowest determined value of "T" is considered as the optimal length of the first codeword. Algorithm (3, 10) shows the steps to compute the optimal length of the shift codeword.

Algorithm (3.10) S-Shift optimizer

- **Input:** enc( ) =1D array of encoded vector
  Encl = length of the encoded vector
- **Output**: maxnobits, optbits = number of the required bits
- **Method:**
  - Find the maximum value
    - Set max = enc (s1) // for odd s1=1, even s1=2
    - For each I (I = s2 … encl - 1 Step 2) // for odd s2=3, even s2=4
      If max < enc (I) Then max = enc (I)
      End loop I
  - Compute the number of bits required to represent the MAX number
    - maxnobits = Log(Max0) / Log(2)
    - If (2 ^ maxnobits) < max Then
      Increment maxnobits by "1"

- Compute the histogram for the encoded vector
  - For each I (I = 1…encl - 1)

    Set x = enc (I)

    Set His (x) = His(x) + 1

    End loop I

- Shift coding optimizer to compute the number of required bits (maxnobits, optbits)
  - Set maxtotal = maxnobits *encl
  - Set optbits = maxnobits
  - For each I (I= 2 …maxnobits – 1)

    Set range = 2 ^ I - 1

    Set totbits = 0

    For each j (j = 0… max)

    If j < range Then

    Set totbits = totbits + I * His (j)

    Else

    totbits = totbits + (I + maxnobits) * his(j)

    End If

    End loop j

    If totbits < maxtotal Then

    maxtotal = totbits

    optbits = I

    End If

    End loop I

3.2.9 S-Shift Coding

The S-Shift Coding process is illustrated in algorithm (3.11), the algorithm steps is in general, i.e. for odd and even indexes.

Algorithm (3.11) S-Shift Coding

- **Input:** enc ( ) = 1D array of encoded RL vector.
  Encl = length of the encoded vector.
  maxnobits, optbits = number of the required bits
- **Output**: S ( )= 1D array of encoded S-Shift
  totallall = total number of bits
- **Method:**

Set max = (2 ^ optbits) – 1
For I (I = 1 …encl – 1)
- If enc(I) < max Then
  - Set S(I) = enc (I) as an integer has a length (optbits) bits
  - Set totallall = totallall + optbits

Else
  - Set S(I)= (enc (I)-max) as an integer has length (maxnobits) bits
  - Set totallall = totallall + maxnobits

End If
End loop I

3.3 Decompression System

The Summarized Decompression stages can be illustrated in figure (3.7)

Figure (3.7) diagram of the decompression method

3.3.1 S-Shift Decoding

The S-Shift decoding process is illustrated in algorithm (3.12), the algorithm steps is in general, i.e. for odd and even indexes.

Algorithm (3.12) S-Shift Decoding

- **Input:** S ( ) = 1D array of encoded S-Shift.
  Encl = length of the encoded vector.
  maxnobits, optbits = number of the required bits
- **Output**: SD ( )= 1D array of Decoded vector
- **Method:**

Set max = (2 ^ optbits) – 1
For I (I = 1 …encl – 1)
S ( ) =get bits (optbits)
- If S(I) < max Then
  - Set SD(I) = enc (I)

Else
  - Set SD(I)= (enc (I)+max) as an integer has length maxnobits bits

End If
End loop I

3.3.2 Run Length Decoding

The Run Length decoding steps are illustrated in the following algorithm:

Algorithm (3.13) Run Length Decoding

- **Input:** SD( ) =1D array 1D array of Decoded S-Shift vector
- **Output**: RD( ) =1D array of decoded RL
  Decl= length of decoded vector
- **Method:**

If (SD (0) = 0) Then Set typ = 0
Else Set typ = 1
End If
Set decl = 0, set I = 1
While I <= (encl - 1)
If (SD (I) = 0) Then
- Increment I by "1"
- Set RD (decl) = SD (I)
- Increment decl by "1"

Else
If typ = 0 Then
- Set Counter = Sd(I)
- For j = 0 to counter - 1
  - Set RD (decl) = 0
  - Increment decl by "1"

End loop j
- Set typ = 1

Else
- Set RD (decl) = SD (I)
- Increment decl by "1"
- Set typ = 0

End If

End If

Increment I by "1"

End loop I

3.3.3 Inverse mapping Process

The inverse mapping process steps are illustrated as follow:

Algorithm (3.14) Inverse Mapping Processes

- **Input:** RD( ) = 1D array // RD( ) indicates the (LH, HL and HH)
  (x1, x2, y1, y2)= the sub band coordinates
- **Output**: DZ( ) =2D array represent the reconstructed sub bands (LH, HL, and HH) From RLE decoding
- **Method:**

Set Flag= 0

Set I=1

For each Column y (y=y1... y)

- If flag = 0 Then

  Xs=x1, xe = x2, stp =1, flag=1

  Else

  Xs=x2, xe = x1, stp =-1, flag=0

  End if
- For each row x (x=xs…xe step stp)
  - If (RD(I)) mod 2= 0) Then

    DZ(x, y) = RD (I)/2

    Else

    DZ(x, y) = - ((RD (I)/ 2) + 1)
  - Increment I by "1"

End loop x, End loop y

3.3.4 Dequantization

As mentioned in section (3.2.5), a uniform quantization was applied on the sub bands (LH, HL and HH).The dequantized sub bands can be calculated by multiplying the **Qstep** with the sub band matrix.

$$C'(x, y) = Qstep \times I(x, y) \text{ .....} \quad (3.20)$$

Where the **Qstep** explained in equations (3.15) and (3.16)

3.3.5 Fractal Decoding

The fractal decoding steps include:

A. Dequantization

The reconstructed Scale coefficient was produced using the following equation:

$$Sdq = Sq \times MaxS \frac{2}{\left(2^{NOBS} - 1\right)} \quad \text{..... (3.21)}$$

While the reconstructed offset coefficient was produces using the following equation:

$$Odq = Oq \times \frac{MaxO - MinO}{2^{NOBO} - 1} + MinO \quad \text{..... (3.22)}$$

Where:

- **Sq** is the Quantized Scale value
- **Sdq** is the Dequantized Scale value
- **Oq** is the Quantized Offset value
- **Odq** is the Dequantized Offset value
- **NOBO** is the number of bits assigned to Offset

- **NOBS** is the number of bits assigned to Scale
- **MaxS** is the maximum value for the Scale coefficient
- **MaxO** is the maximum value for the Offset coefficient
- **MinO** is the minimum value for the Offset coefficient

B. Decoding steps

The first step in Fractal decoding was to generate an arbitrary array which has the same size of domain, and initialize it to zero values, then for each Range block, get the IFS coefficients (coordinates x, y) of the matched domain block to get this block from the initialized domain matrix, the rest of IFS coefficients (S, O, Sym) were applied on the domain block to get the Reconstructed Range block using the following equation:

$$Rf(x,y) = Sym(Dom(x,y)) \times Sdq + Odq \quad \text{..... (3.23)}$$

Where:

- **Sdq** is the Dequantized Scale value
- **Odq** is the Dequantized Offset value
- **Sym** is the symmetry case

these steps were applied on each Range block to get the reconstructed image which has the same size of the original image. The result was down sampled to create the new domain. This process was applied in an iterative manner repeats the affine reconstruction again until getting the final reconstructed image.

Algorithm (3.15) Fractal Decoding

- **Input:** S, O, X, Y, Sym =IFS Coefficients for each range block

  R_n = number of Range blocks

- **Output:** Rf ( )=2D array represented the Reconstructed image
- **Method:**

For each column y (y=0, 1…Hh-1)

For each row x(x=0, 1…Wh-1)

- Set domain(x, y) =0

End loop x

End loop y

For each Iteration (Iteration=0, 1… 10)

- For each Range block R_i (i=0, 1… R_n)
  - Get IFS Coefficients (Sym, S, O, X and y)
  - Compute the **"Sdq"** using equation (3.21).
  - Compute the **"Odq"** using equation (3.22).
  - Get Domain block "**Dom**" from "**domain**" according to X and Y coordinates from IFS Coefficients
  - Applying the affine transformation on "**Dom**" according to the "Sym" case
  - Compute the Reconstructed Image Rf(x, y) using equation(3.23)

End loop R_i

- Compute the domain by down sampling the Reconstructed range block Rf (x,y)

End loop Iteration

3.3.6 Inverse Haar Wavelet Transform

The inverse haar wavelet transform steps are as follow:

Algorithm (3.16) the Inverse Haar Wavelet Transform

- **Input:** Rf( )= reconstructed image from Fractal decoding
  Dz( ) = represent the reconstructed sub bands (LH, HL, and HH) From RLE decoding
  Wlevel= level of wavelet transform
- **Output:** Rwave( ) = the reconstructed image
- **Method:**

Assign the sub bands arrays (Rf, Dz) to one matrix (Rwave)

Set wtemp = w

Set htemp = h

For each wavelet level j (j = 1… wlevel)

- Set wtemp = wtemp / 2
- Set htemp = htemp / 2

End loop j

For each wavelet level (i = 1… wlevel)

- Set W2 = wtemp * 2
- Set H2 = htemp * 2

For each Column y (y=0, 1… htemp-1)

- Set yy = 2 * y
- Set Yp = yy + 1

For each Row x (x=0, 1… wtemp-1)

- Set xx = 2 * x
- Set Xp = xx + 1
- Compute tempimage for (xx, yy) coordinate

tempimage = (Rwave (x, y) + Rwave (x + wtemp, y)+
Rwave (x, y + htemp) + Rwave (x + wtemp, y + htemp))

- Compute tempimage for (Xp, yy) coordinate
  tempimage = (Rwave (x, y) + Rwave (x + wtemp, y) -
  Rwave (x, y + htemp) - Rwave (x + wtemp, y + htemp))

- Compute tempimage for (xx, yp) coordinate
  tempimage = (Rwave (x, y) - Rwave (x + wtemp, y) +
  Rwave (x, y + htemp) - Rwave (x + wtemp, y + htemp))

- Compute tempimage for (xx, yp) coordinate
  tempimage = (Rwave (x, y) - Rwave (x + wtemp, y) -
  Rwave (x, y + htemp) + Rwave (x + wtemp, y + htemp))

End loop x

End loop y

For each Column y (y=0, 1… h2-1)

For each Row x (x=0, 1… w2-1)

- Set Rwave (x,y) =tempimage (x,y)

End loop x

End loop y

- Set wtemp = W2
- Set htemp = H2

End loop j

3.3.7 Inverse Color Transform

The YC_bC_r color model was transformed to RGB using equations (1.10…1.12). The inverse color transform steps are as follow:

Algorithm (3.17) the Color Detransform

- **Input :** pic3( ) = the reconstructed image // YCbCr
- **Output**: Recimage( ) = RGB Image $(W \times H)$
- **Method:**

For each Column y (y=0, 1… h-1)

For each Row x (x=0, 1… w-1)

- Compute R color from pic3(x, y)

R= 1.164 * (pic3(x, y).y - 16) + 0 * (pic3(x, y).C_b - 128) + 1.596 * (pic3(x, y).C_r - 128)

- Compute G color from pic3(x, y)

G= 1.164 * (pic3(x, y).y - 16) - 0.392 * (pic3(x, y). C_b - 128) - 0.813 * (pic3(x, y).C_r - 128)

- Compute B color from pic3(x, y)

B= 1.164 * (yuv(x, y).y - 16) + 2.017 * (yuv(x, y). C_b - 128) -128

End loop x

End loop y

Table (4.6): The effect of Block Size on the Reconstructed Hakam Image

BS of Y	BS of Cb, Cr	Y Component			Cb Component			Cr Component			RGB		
		MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR
2	2	5.158	41.005	8.067	0.837	48.899	1.454	0.925	48.468	1.454	8.993	38.591	2.001
4	4	57.154	30.560	8.565	1.452	46.511	5.817	1.694	45.841	5.817	80.652	29.064	6.514
2	8	5.158	41.005	8.067	2.526	44.105	23.269	3.6015	42.565	23.269	13.924	36.692	14.292

Table (4.7): The effect of Block Size on the Reconstructed Girl Image

BS of Y	BS of Cb, Cr	Y Component			Cb Component			Cr Component			RGB		
		MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR
2	2	7.995	39.102	4.385	0.980	48.218	1.453	1.534	46.272	1.451	13.346	36.877	1.869
4	4	86.916	28.739	4.530	2.494	44.160	5.804	4.838	41.283	5.779	99.883	27.343	5.300
2	8	7.995	39.102	4.385	6.051	40.312	23.161	13.951	36.684	22.934	30.890	33.232	9.530

Table (4.8): The effect of Block Size on the Reconstructed Horses Image

BS of Y	BS of Cb, Cr	Y Component			Cb Component			Cr Component			RGB		
		MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR
2	2	5.989	40.357	6.445	0.870	48.731	1.454	0.803	49.080	1.454	9.962	38.147	1.960
4	4	37.640	32.374	6.751	2.049	45.014	5.817	1.783	45.617	5.817	55.474	30.689	6.098
2	8	5.989	40.357	6.445	5.013	41.129	23.269	4.282	41.814	23.269	18.815	35.385	12.443

4.2.2 Block size tests

In this test, different block sizes were implemented on three different images. From the wavelet level tests that mentioned before, third level wavelet transform gave better performance parameters i.e., MSE, PSNR, Compression Ratio, which seems to be acceptable. The input parameters were taken as follow (see table (4.٥)).

Table (4.5): Input Parameters for different Block sizes

WL	JS	NOBS	NOBO	A	B	R
3	2	5	7	6	1.7	0.4

From tables (4.6…4.8) we can see that the increasing in block size affects on increasing of the MSE so the PSNR will be decreased; this is undesirable for compression performance parameter, but the CR will be increased. So the first test with (BS=2) gives better performance parameters with better image quality for Y component, while for Cb and Cr components the (BS=8) gives better compression ratio.

Wavelet level=3

Block size for Y component=2

Block size for Cb and Cr components=8

Jump step=2

NOBO=7, NOBS=5

A=6, B=1.7, R=0.4

PSNR=36.692

CR=14.292

Figure (4.2) the reconstructed RGB Hakam image

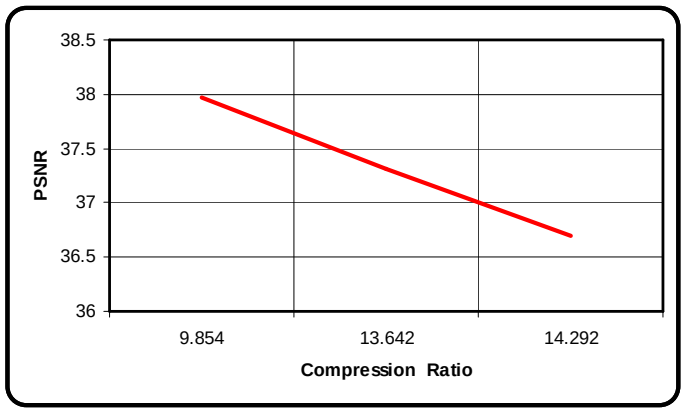


Figure (4.3) Compression ratio versus PSNR of suggested method for Hakam image

Wavelet level=3

Block size for Y component=2

Block size for Cb and Cr components=8

Jump step=2

NOBO=7, NOBS=5

A=6, B=1.7, R=0.4

PSNR=33.232

CR=9.530

Figure (4.4) the reconstructed RGB Girl image

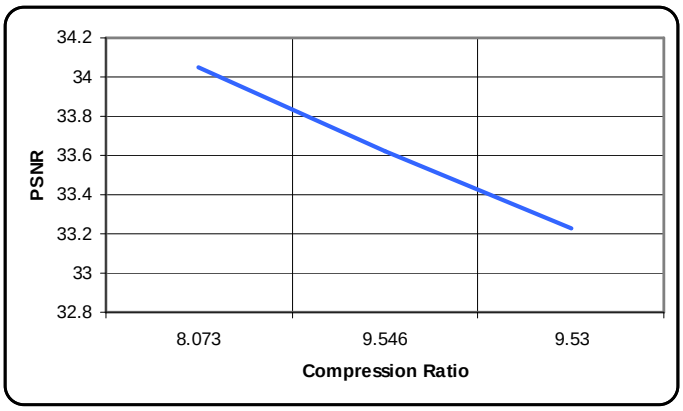


Figure (4.5) Compression ratio versus PSNR of suggested method for Girl image

Wavelet level=3

Block size for Y component=2

Block size for Cb and Cr components=8

Jump step=2

NOBO=7, NOBS=5

A=6, B=1.7, R=0.4

PSNR=35.385

CR=12.443

Figure (4.6) the reconstructed RGB Horses image

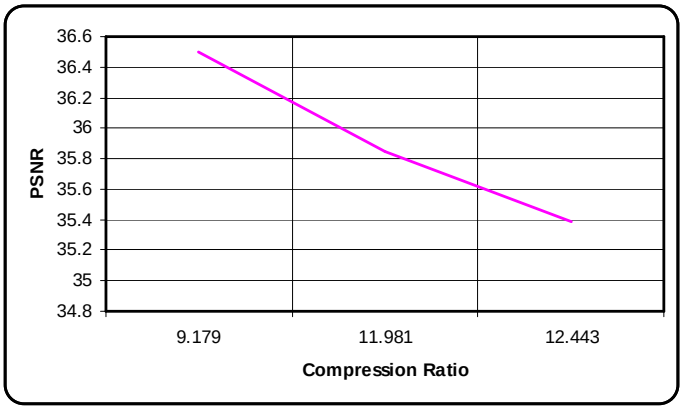


Figure (4.7) Compression ratio versus PSNR of suggested method for Horses image

Table (4.14): The Fractal Quantization test on the Reconstructed Hakam Image

NOBS	NOBO	Y Component			Cb Component			Cr Component			RGB		
		MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR
5	6	6.659	39.896	8.099	3.466	42.731	24.377	5.083	41.068	24.357	18.624	35.429	14.595
5	7	5.158	41.005	8.067	2.526	44.105	23.269	3.6015	42.565	23.269	13.924	36.692	14.292
6	8	4.284	41.811	8.003	2.039	45.034	21.330	2.997	43.362	21.330	11.529	37.512	13.717

Table (4.15): The Fractal Quantization test on the Reconstructed Girl Image

NOBS	NOBO	Y Component			Cb Component			Cr Component			RGB		
		MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR
5	6	10.093	38.090	4.394	7.129	39.600	24.222	14.911	36.395	23.799	36.002	32.567	9.650
5	7	7.995	39.102	4.385	6.051	40.312	23.161	13.951	36.684	22.934	30.890	33.232	9.530
6	8	7.099	39.618	4.367	5.619	40.634	21.233	13.484	36.832	21.070	29.028	33.502	9.272

Table (4.16): The Fractal Quantization test on the Reconstructed Horses Image

NOBS	NOBO	Y Component			Cb Component			Cr Component			RGB		
		MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR
5	6	7.062	39.641	6.464	5.933	40.397	24.296	5.266	40.915	24.336	22.544	34.600	12.661
5	7	5.989	40.357	6.445	5.013	41.129	23.269	4.282	41.814	23.269	18.815	35.385	12.443
6	8	5.093	41.060	6.404	4.547	41.553	21.330	3.804	42.327	21.330	16.515	35.951	12.005

4.2.4 Fractal Quantization tests

In these tests, different NOBO and NOBS values were taken. The input parameters are as follow:

Table (4.13) Input parameters for different NOBS and NOBO values

WL	BS	JS	A	B	R
3	**2×2**	**2**	**6**	**1.7**	**0.4**

From tables (4.14…4.16) we can see that the increasing in NOBO and NOBS value affects on decreasing of the MSE so the PSNR will be increased; this is desirable for compression performance parameter, but the CR will be decreased. When NOBS is = 5, NOBO =7 it gives better performance parameters.

Table (4.10): The effect of Jump step on the Reconstructed Hakam Image

JS	Y Component			Cb Component			Cr Component			RGB		
	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR
1	4.402	41.693	8.004	2.526	44.105	23.269	3.6015	42.565	23.269	12.851	37.041	14.226
2	5.158	41.005	8.067	2.526	44.105	23.269	3.6015	42.565	23.269	13.924	36.692	14.292
4	6.471	40.020	8.131	2.526	44.105	23.269	3.6015	42.565	23.269	15.566	36.209	14.359

Table (4.11): The effect of Jump step on the Reconstructed Girl Image

JS	Y Component			Cb Component			Cr Component			RGB		
	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR
1	6.439	40.042	4.367	6.051	40.312	23.161	13.951	36.684	22.934	28.690	33.553	9.501
2	7.995	39.102	4.385	6.051	40.312	23.161	13.951	36.684	22.934	30.890	33.232	9.530
4	14.846	36.414	4.404	6.051	40.312	23.161	13.951	36.684	22.934	39.909	32.120	9.560

Table (4.12): The effect of Jump step on the Reconstructed Horses Image

JS	Y Component			Cb Component			Cr Component			RGB		
	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR
1	4.907	41.222	6.405	5.013	41.129	23.269	4.282	41.814	23.269	17.326	35.743	12.392
2	5.989	40.357	6.445	5.013	41.129	23.269	4.282	41.814	23.269	18.815	35.385	12.443
4	8.446	38.863	6.486	5.013	41.129	23.269	4.282	41.814	23.269	22.130	34.680	12.493

4.2.3 Jump Step tests

In this test, different Jump step values were implemented on three different images. From the block size tests that mentioned before, we will take (BS=2) which gave better performance parameters i.e., MSE, PSNR, CR, and seems to be acceptable. The other input parameters were taken as follow (see table (4.9)).

Table (4.9): Input Parameters for different Jump Step values

WL	BS	NOBS	NOBO	A	B	R
3	2×2	5	7	6	1.7	0.4

From tables (4.10…4.12) we can see that the increasing in Jump step value had a little affect on increasing the MSE and decreasing the PSNR; this also had a little affect on CR which will be increased when increasing the jump step value.

Table (4.2): The effect of Wavelet levels on the Reconstructed Hakam Image

WL	Y Component			Cb Component			Cr Component			RGB		
	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR
1	2.517	44.121	4.577	2.526	44.105	23.269	3.6015	42.565	23.269	10.382	37.967	9.854
2	3.790	42.343	7.465	2.526	44.105	23.269	3.6015	42.565	23.269	12.067	37.314	13.642
3	5.158	41.005	8.067	2.526	44.105	23.269	3.6015	42.565	23.269	13.924	36.692	14.292

Table (4.3): The effect of Wavelet levels on the Reconstructed Girl Image

WL	Y Component			Cb Component			Cr Component			RGB		
	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR
1	3.987	42.123	3.511	6.051	40.312	23.161	13.951	36.684	22.934	25.571	34.053	8.073
2	6.029	40.328	4.396	6.051	40.312	23.161	13.951	36.684	22.934	28.219	33.625	9.546
3	7.995	39.102	4.385	6.051	40.312	23.161	13.951	36.684	22.934	30.890	33.232	9.530

Table (4.4): The effect of Wavelet levels on the Reconstructed Horses Image

WL	Y Component			Cb Component			Cr Component			RGB		
	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR
1	2.857	43.571	4.151	5.013	41.129	23.269	4.282	41.814	23.269	14.554	36.500	9.179
2	4.584	41.517	6.081	5.013	41.129	23.269	4.282	41.814	23.269	16.901	35.851	11.981
3	5.989	40.357	6.445	5.013	41.129	23.269	4.282	41.814	23.269	18.815	35.385	12.443

Chapter Four
Performance Measures and Test Result

4.1 Introduction

Many aspects have been used to study visual perception. It has been shown that the human visual perception system is sensitive to changes in luminance rather than the absolute luminance values themselves, and that perception is most sensitive to mid-frequencies and less sensitive to high frequencies in the image. This chapter, attempted to evaluate both objective and subjective methods for an acceptable degree of the reconstructed images for different compression tests.

4.2 Tests and Results

To evaluate the performance of the proposed compression methods, a picture was taken in size (256×256). It is of 24 bits/pixel RGB signal, and it was transformed to YC_bC_r signal.

Many tests were accomplished to find the best input parameters that give the best compression performance results.

The testing consists of four main columns; the first one is that Wavelet, PIFS, Uniform Quantization, RLE and S-Shift coder were applied on Y component. The second and the third columns applied only PIFS on Cb and Cr components, the last is the RGB signal; this column has the average performance parameters.

Each column has its performance parameters, which include:

- "CR" Compression Ratio.
- "PSNR" Peak Signal to Noise Ratio.
- "MSE" Mean Square Error.

The input parameters are:

- "WL" Wavelet Level.
- "BS" Block Size.
- "JS" Jump Step.
- "NOBO" Number of Bits need for the Offset.
- "NOBS" Number of Bits need for the Scale.
- "A","B" and "R" are the Quantization Parameters.

(a) Original Hakam image

(b) Original Girl Image

(c) Original Horses image

Figure (4.1) the original RGB images (uncompressed)

4.2.1 Wavelet Level tests

The first test the Wavelet level test and it is applied on three pictures; Hakam horses and girl.

In this test, the wavelet transform levels were taken (WL=1, 2 and 3) .The effect of these levels can be notice on three input images .the other parameters were fixed and illustrated in table (4.1), for Y component the (Bs=2 and JS=2) while Cb and Cr components the (Bs=8 and JS=4).

Table (4.1): the input parameters for different Wavelet levels

BS	JS	NOBS	NOBO	A	B	R
2×2	2	5	7	6	1.7	0.4

From tables (4.2…4.4) we can see that the increasing in wavelet levels affects on increasing of the MSE so the PSNR will be decreased; this is undesirable for compression performance parameter, but the CR will be increased. So the third level gives better performance parameters with better image quality.

Table (4.18): The effect of Quantization parameter "A" on the Reconstructed Hakam Image

A	Y Component			Cb Component			Cr Component			RGB		
	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR
2	3.572	42.601	5.054	2.526	44.105	23.269	3.601	42.565	23.269	11.818	37.405	10.570
4	5.158	41.005	8.067	2.526	44.105	23.269	3.601	42.565	23.269	13.924	36.692	14.292
6	6.181	40.220	10.711	2.526	44.105	23.269	3.601	42.565	23.269	15.236	36.301	16.731

Table (4.19): The effect of Quantization parameter "A" on the Reconstructed Girl Image

A	Y Component			Cb Component			Cr Component			RGB		
	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR
2	5.185	40.982	2.953	6.051	40.312	23.161	13.951	36.684	22.934	27.200	33.784	7.053
4	7.995	39.102	4.385	6.051	40.312	23.161	13.951	36.684	22.934	30.890	33.232	9.530
6	10.342	37.984	5.957	6.051	40.312	23.161	13.951	36.684	22.934	34.128	32.799	11.782

Table (4.20): The effect of Quantization parameter "A" on the Reconstructed Horses Image

A	Y Component			Cb Component			Cr Component			RGB		
	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR
2	4.198	41.899	4.351	5.013	41.129	23.269	4.282	41.814	23.269	16.397	35.983	9.501
4	5.989	40.357	6.445	5.013	41.129	23.269	4.282	41.814	23.269	18.815	35.385	12.443
6	7.221	39.544	8.687	5.013	41.129	23.269	4.282	41.814	23.269	20.207	35.075	14.921

4.2.5 Quantization tests

In these tests, different values were taken for each quantization parameters (A, B and R).

1. **"A" tests : The input parameters are as follow:**

Table (4.17) Input parameters for different "A" parameter values

WL	BS	JS	NOBS	NOBO	B	R
٣	٢×٢	2	5	7	1.7	0.4

From tables (4.18…4.20) we can see that the increasing in "A" value affects on increasing of the MSE so the PSNR will be decreased; but it still with acceptable range, at the same time the CR will be increased, so we can see that when (A=6) it gives better compression performance parameters.

Table (4.22): The effect of Quantization parameter "B" on the Reconstructed Hakam Image

B	Y Component			Cb Component			Cr Component			RGB		
	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR
1.5	5.088	41.064	8.048	2.526	44.105	23.269	3.601	42.565	23.269	13.830	36.722	14.272
1.7	5.158	41.005	8.067	2.526	44.105	23.269	3.601	42.565	23.269	13.924	36.692	14.292
2	5.257	40.923	8.116	2.526	44.105	23.269	3.601	42.565	23.269	14.059	36.651	14.343

Table (4.23): The effect of Quantization parameter "B" on the Reconstructed Girl Image

B	Y Component			Cb Component			Cr Component			RGB		
	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR
1.5	7.772	39.225	4.382	6.051	40.312	23.161	13.951	36.684	22.934	30.618	33.270	9.525
1.7	7.995	39.102	4.385	6.051	40.312	23.161	13.951	36.684	22.934	30.890	33.232	9.530
2	8.355	38.911	4.486	6.051	40.312	23.161	13.951	36.684	22.934	31.359	33.167	9.688

Table (4.24): The effect of Quantization parameter "B" on the Reconstructed Horses Image

B	Y Component			Cb Component			Cr Component			RGB		
	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR
1.5	5.880	40.436	6.430	5.013	41.129	23.269	4.282	41.814	23.269	18.672	35.418	12.424
1.7	5.989	40.357	6.445	5.013	41.129	23.269	4.282	41.814	23.269	18.815	35.385	12.443
2	6.128	40.257	6.549	5.013	41.129	23.269	4.282	41.814	23.269	19.001	35.342	12.572

2. "B" tests : The input parameters are as follow:

Table (4.21) Input parameters for different "B" parameter values

WL	BS	JS	NOBS	NOBO	A	R
3	2×2	2	5	7	6	0.4

From tables (4.22…4.24) we can see that the increasing in "B" value affects on increasing of the MSE so the PSNR will be decreased; at the same time the CR will be increased until it reach value which does not affects on the performance parameters, unlike the "A" parameters that has a big affects on the performance parameters. It is obvious when (B=1.7) it gives better compression performance parameters.

Table (4.26): The effect of Quantization parameter "R" on the Reconstructed Hakam Image

R	Y Component			Cb Component			Cr Component			RGB		
	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR
0.4	5.158	41.005	8.067	2.526	44.105	23.269	3.601	42.565	23.269	13.924	36.692	14.292
0.5	5.362	40.837	9.341	2.526	44.105	23.269	3.601	42.565	23.269	14.127	36.630	15.544
0.7	6.541	39.974	10.210	2.526	44.105	23.269	3.601	42.565	23.269	15.833	36.134	16.313

Table (4.27): The effect of Quantization parameter "R" on the Reconstructed Girl Image

R	Y Component			Cb Component			Cr Component			RGB		
	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR
0.4	7.995	39.102	4.385	6.051	40.312	23.161	13.951	36.684	22.934	30.890	33.232	9.530
0.5	8.324	38.927	4.812	6.051	40.312	23.161	13.951	36.684	22.934	31.381	33.164	10.184
0.7	10.120	38.078	5.214	6.051	40.312	23.161	13.951	36.684	22.934	33.643	32.861	10.770

Table (4.28): The effect of Quantization parameter "R" on the Reconstructed Horses Image

R	Y Component			Cb Component			Cr Component			RGB		
	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR	MSE	PSNR	CR
0.4	5.989	40.357	6.445	5.013	41.129	23.269	4.282	41.814	23.269	18.815	35.385	12.443
0.5	6.106	40.272	7.383	5.013	41.129	23.269	4.282	41.814	23.269	18.804	35.388	13.550
0.7	7.392	39.442	7.985	5.013	41.129	23.269	4.282	41.814	23.269	20.977	34.913	14.206

3. "R" tests : The input parameters are as follow:

Table (4.25) Input parameters for different "R" parameter values

WL	BS	JS	NOBS	NOBO	A	B
3	2×2	2	5	7	6	1.7

From tables (4.26…4.28) we can see that the when "R"=0.4 it gives better compression performance parameters.

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

Image Compression Based on Fractal Coding and Wavelet Transform

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

By

Susan Saadoon Al-Barazanchi

(B.Sc. 2004)

Supervised By

Dr. Ban N. Al-Kallak

2007 **1428**

September **Ramadan**

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة النهرين
كلية العلوم

ضغط الصورة بالاعتماد على الضغط الكسوري و التحويل الموجي

رساله
مقدمه الى قسم علوم الحاسبات في جامعة النهرين كجزء
من متطلبات نيل درجة الماجستير في علوم الحاسبات

من قبل

سوزان سعدون البرزنجي

(بكالوريوس جامعة النهرين ٢٠٠٤)

أشراف

د. بان نديم الكلاك

٢٠٠٧ ١٤٢٨

أيلول رمضان

Appendix
The BMP File Format

The BMP file structure is very simple and is shown in figure (1):

File Header	Image Header	Color Table	Pixel Data

Figure (1): BMP File Format

• File Header

Every window BMP begins with a **BITMAPFILEHEADER** structure whose layout is shown in table (1). The main function of this table is to serve as the signature that identifies that file format.

Table (1): Bit Map File header structure

Field Name	Size in Byte	Description
bfType	2	Contains the characters "BM" that identify the file type
bfSize	4	File size
BfReserved1	2	unused
BfReserved2	2	unused
bfOffbits	4	Offset to start of pixel data

Three checks can be made to ensure that the file you are reading is in fact a BMP file:

1. The first two bytes of the file must contain the ASCII characters "B" followed by "M".

2. If you are using a file system where you can determine the exact file size in bytes, you can compare the file size with value in the bfSize field.
3. The bfReserved1 and bfReserved2 fields must be zero.

The file header also specifies the location of the pixel data in the file. When decoding a BMP file you must use the bfOffbits to determine the offset from the begging of the file to where the pixel data starts. Most applications place the pixel data immediately following the **BIEMAPINFOHEADER** structure or palette, if it is present. However, some applications place filter bytes between these structures and pixel data so you must use the bfOffbits to determine the number of bytes from the **BITMAPFILEHEADER** structure to the pixel data.

• Image Header

The image header immediately follows the **BITMAPFILEHEADER** structure .it comes I two distinct formats, defined by the **BITMAPINFOHEADER** and **BITMAPPCOREHEADER** structures.

BITMAPPCOREHEADER represents the OS/2 BMP format and **BITMAPINFOHEADER** is the much more common windows format. Unfortunately, the re is no version field in the BMP definitions. The only way to determine the type of image structure used in particular file is to examine the structure's size field, which is the first 4 bytes of both structure types. The size of the **BITMAPPCOREHEADER** structure is 12 bytes; the size of the **BITMAPINFOHEADER**, at least 40 bytes.

The layout of **BITMAPINFOHEADER** is shown in table (2).thios structure gives the dimensions and bit depth of the image and tells if the image is compressed .window 95 supports a BMP format that uses an enlarged version of this header. Few applications create BMP files using this format; however; a decoder should be implemented so that it knows that header sizes can be larger than 40 bytes. The image height is an unsigned value. A negative value for the biHeight field specifies that the pixel data is ordered from the top down rather

than the normal bottom up. Image with a negative biHeight value may not be compressed.

Table (2): Bit Map Info Header structure

Field Name	Size in Byte	Description
biSize	4	Header size must be at least 40
biWidth	4	Image width
biHeight	4	Image height
biplanes	2	Must be 1
biBitCount	2	B?it per pixel
biCompression	4	Compression Type: BI_RGB=0,BI_RLE8=1,BI_RLE4=2 or BI_BIFIELDS=3
biSizeImage	4	Image size: maybe zero if not compressed
bixPelPerMeter	4	Preferred resolution in pixels per Meter
biyPelsPerMeter	4	Preferred resolution in pixels per Meter
biClrUsed	4	Number of entries in the color map that are actually used
biClrImportant	4	Number of significant colors

The **BITMAPPCOREHEADER** structure is the other image header format. Its layout is shown in table (3).

Table (3): Bit Map Core Header structure

Field Name	Size in Byte	Description
bcSize	4	Header size must be 12
bcWidth	2	Image Width
bcHeight	2	Image Height
bcPlanes	2	Must be 1
bcBitCount	2	Bit Count:1,4,8 or 24

Notice that it has fewer fields and that all have analogous fields in the **BITMAPINFOHEADER** structure. If the file uses **BITMAPPCOREHEADER** rather than **BITMAPINFOHEADER**, the pixel data can not be compressed.

• Color Palette

The color palette immediately follows the file header and can be in one of three formats. The first two are used to map pixel data to RGB color values when the bit count is 1, 4, or 8 (biBitCount or bcBitCount fields). For BMP files in the windows format, the palette consists of an array of 2 bitcount RGBQUAD structures, see table (4) .BMP file in OS/2 format use an array of RGBTRPLE structures, see table (5).

Table (4): RGBQUAD structure

Field Name	Size in Byte	Description
rgbBlue	1	Blue color value
rgbGreen	1	Green color value
rgbRed	1	Red color value
rgbReserved	1	Must be zero

Table (5): RGBTRIPLE structure

Field Name	Size in Byte	Description
rgbBlue	1	Blue color value
rgbGreen	1	Green color value
rgbRed	1	Red color value

References

Appendix

Chapter One

General Introduction

Chapter Two

Theoretical Background

Chapter Three

The Proposed Image Compression Methods

Chapter Four

Performance Measures and Test Results

Chapter Five

Conclusions And Future Work

بسم الله الرحمن الرحيم

((وإن ليسَ للإ نسان إلا ما سَعى وإنَ سَعيهٌ سوف يُرى ثم يُجزاه الجزاءَ الأوفى))

صَدقَ اللُّه العَظيم

القرآن الكريم –سورة النجم ، اية ٣٩-٤٠

Dedication

To my Parents ..

My sisters ..

Brothr..

and

My Friends

Acknowledgment

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

My great thanks to my supervisor Dr.Ban N. Al-Kallak for her guidance, supervision and efforts during this work.

Grateful thanks 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, sisters Vian and Mayada, my brother Mohammed for their help and patience during this work.

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

List of Abbreviations

BMP	Bit Map
CMY	Cyan-Magenta and Yellow
CR	Compression Ratio
GUI	Graphic User Interface
HH	High High
HL	High Low
HSV	Hue-Saturation-Value
HVS	Human Visual System
IFS	Iterated Function System
LH	Low High
LL	Low Low
RMSE	Root Mean Square Error
LZW	Lempel-Ziv-Welsh
PIFS	Partitioned Iterated Function System
PSNR	Peak Signal To Noise Ratio
RGB	Red-Green and Blue
RLE	Run Length Encoding
VQ	Vector Quantization
YUV	Y: Luminance Component U,V: Chrominance Components

المعلومات الشخصية

الاسم الثلاثي: سوزان سعدون بهلول البرزنجي

عنوان السكن: بغداد \ حي القاهرة \ محله ٣٠٧\ زقاق ١٨\ دار ٨٥

رقم الهاتف: موبايل : ٠٠٩٦٤٧٧٠٢٧٠٨٥٨٦

العنوان البريدي : suzan_albar@yahoo.com

تاريخ المناقشة : ٢٢\١٠\٢٠٠٧

أسم الرسالة:

Image Compression Based on Fractal Coding and Wavelet Transform

أسم المشرف: المدرس -الدكتورة بان نديم الكلاك

References

- **[Ach05]** Acharya T., Ray A. K., "Image Processing-Principles and Applications", Library of Congress Cataloging-in-Publication Data, A John Willy & Sons, INC., Publication, 2005.

- **[Ald00]** Al-Dulaimy A. A., "Fractal Image Compression with Fasting Approaches", M.Sc. Thesis, College of Science, Computer science department, Al-Nahrain University, 2000.

- **[And00]** Andreopoulos I., Karayiannis Y. A. , Stouraitis T., "A Hybrid Image Compression Algorithm Based On Fractal Coding and Wavelet Transform", Department of Electrical and Computer Engineering, University of Patras, Greece, ISCAS 2000 - IEEE International Symposium on Circuits and Systems, May 28-31, 2000, Geneva, Switzerland,3 2000.
  Site: *http:// citeseer.ist.psu.edu/458102.pdf*

- **[Bou91]** Bourke P.," An Introduction to Fractals", University of Western Australia, Math Forum Internet Mathematics Library, 1991.
  E-mail:*paul.bourke@uwa.edu.au*
  Site: *http:// astronomy.swin.edu.au/~pbourke/fractals/fracintro*

- **[Eng03]** Engel D., "Adaptive Object based Image Compression with Wavelet Methods", Department of Scientific Computing, University of Salzburg, In Proceedings of the 23rd International Picture Coding Symposium pp. 283–288, 2003.
  Site: *http://www.cosy.sbg.ac.at/~dengel/publications/dipl.pdf*

- **[Fer02]** Ferrell R., Gleason S., Tobin K., "Application of Fractal Encoding Techniques for Image Segmentation", Oak Ridge National Laboratory, Oak Ridge, TN, USA, 2002.
  Site: *http:// ww.ornl.gov/~webworks/cppr/y2001/pres/116616.pdf*

- **[Fis94]** Fisher B., Perkins S., Walker A., Erik W. ," Affine Transformation", Department of Artificial Intelligence, University of Edinburgh, UK, 1994.
  Site: *http:// www.cee.hw.ac.uk/hipr/html/affine.html*

- **[Fis95]** Fisher Y.," Fractal Image Compression", Library of Congress Cataloging-in-Publication Data, Springer-Veriag New York, Inc, 1995.

- **[Han99]** Hankerson D., Harris G.," Transform Methods and Image Compression", Linux Journal, on Friday 1999.
  URL: *http://www.linuxjournal.com/article/2567*
  Site: *http:// www.linuxjournal.com/node/2567/print*

- **[Har00]** Hartenstein H., "Region-Based Fractal Image Compression", Matthias Ruhl, and Dietmar Saupe, IEEE Transactions on Image Processing, vol. 9, no. 7, July 2000.
  Site: *http:// theory.lcs.mit.edu/~ruhl/papers/2000-toip.pdf*

- **[Hus04]** Hussain A.," Image Compression Using Neural Networks", Master project, School Of Computing and Mathematical Sciences, Liverpool John Moores University, 2004.
  Site: *http:// www2.umist.ac.uk/csc/issl/research/abir*

- **[Itu02**] International Telecommunications Union ITU-R BT.790-5,"Parameter Values for HDTV Standards for Production and International Program Exchange", 2002.

- **[Kap04]** Kaplan I., "Applying the Haar Wavelet Transform to Time Series Information ", IEEE, 2004.
  Site: *http:// www.bearcave.com/misl/misl_tech/wavelets/haar.pdf*

- **[Kas02]** Kassim B., "Color Image Data Compression Using Multilevel Block Truncation Coding Technique", Ph.D. thesis, College of Science, Baghdad University, 2002.

- **[kha00]** Khalifa O., Dlay S., "Fractal Wavelets Image Data Compression", International IMACS/IEEE International Multiconference on Circuits, Systems, Communications and Computers, Athens, pp5111-5114, 2000.
  E-mail: *O.O.Khalifa@ncl.ac.uk, S.S.Dlay@ncl.ac.uk*
  Site: *http:// www.wseas.com/2000and1999.xls*

- **[Kom04]** Kominek J.," Introduction to Fractal compression", the Association for Computing Machinery (ACM) digital library, Springer-Verlag New York, Inc, V.5, PP. 77, 2004.
  Site: *kominek@links.uwaterloo.ca*

- **[Kum03]** Kumar S.," An Introduction to Image Compression", Kluwer Academic Publishers, Hingham, MA, USA, Volume 11, Pages: 155 - 191, January, 2005.

- **[Lij99]** Li J. , Jay C. K., "Image Compression with a Hybrid Wavelet-Fractal Coder", IEEE Transactions on image processing, vol. 8, no. 6, June 1999.
  Site: *http://research.microsoft.com/users/jinl/paper_1999/ip99jun_fwt.pdf*

- **[Lor03]** Lorenz W. E.," Fractals and Fractal Architecture-Chaotic Fractals", department of computer aided planning and architecture, Vienna university of technology, 2003.

Site: *http:// www.iemar.tuwien.ac.at/modul23/Fractals/subpages/05Architecture.html*

- **[Lou06]** Loup J., "Comp. Compression Frequently Asked Questions (part 2/3)", Newsgroups (comp.compression) and (comp.compression.research), FAQ part (2/3), 2006.
  E-mail:*jloup@gzip.OmitThis.org*
  Site: *http:// www.faqs.org/faqs/compression-faq/part2*

- **[Man83]** Mandelbrot B. B.," Fractals and the Fractal Dimension", Walfarm Math World, Math Resources**,** 1983**.**
  Site*: http://mathworld.wolfram.com/Fractal.html*

- **[Mur07]** Murry M**.**, James D., Ryper V., William," Run Length Encoding (RLE)", Encyclopedia of Graphics File Formats book, 2ndEdition, 2007.
  Site: *http://www.fileformat.info/mirror/egff/ch09_03.htm*

- **[Nel98]** Nelson M.," Arithmetic Coding + Statistical Modeling = Data Compression", Dr. Dobb's Journal, the world of the software development, Part one, February 1998.
  Site: *http:// www.dogma.net/markn/articles/arith/part1.htm*

- **[Och98]** Ochoa G.," An Introduction to Lindenmayer Systems", School of Cognitive and Computing Sciences, The University of Sussex, 1998.
  Site: *http:// www.biologie.uni-hamburg.de/b-online/e28_3/lsys.html*

- **[Pra01]** Pratt **W. K.**, "Digital Image Processing", A Wiley-Inter science Publication , John Wiley & Sons, INC., third edition ,2001.

- **[Qia03]** Qiao Y., Lu Z. , Sun S.," Multipurpose Image Watermarking Based on Wavelet Transform", the 5th International Symposium on Test and Measurement, Shenzhen, China, 2003.
  Site: *http://lmb.informatik.uni-freiburg.de/people/zmlu/publications_en.htm*

- **[Roc95]** Roche S., Dugelay J., "Improvement in I.F.S Formulation for its Use in Still Image Coding", Institute Erecome, Multimedia Communications departement., 1995.
  E-mail: *dugelay@eurecome.fr*
  Site: URL: *http://www.cica.fr/~image*

- **[Sah04]** Saha S.," Image Compression - from DCT to Wavelets: A Review", the Math forum internet mathematics library, Drexel School of Education 2004.
  Site: *http://dret.net/biblio/reference/sah00*

- **[Sal04]** Salmon D., "Data Compression the complete reference", Department of Computer Science, California State University, USA, DeLaMera Library third edition, 2004.
  Site: *www.delamare.unr.edu/newbooks/books_404.html*

- **[San98]** Sangwine S. J., Horne R. E. N., "The color Image Processing Handbook", Chapman and Hall publishing company, London, 1998.

- **[Sly91]** Slyz M., "Image Compression Using a ZIV-LEMPEL Type Coder", Master Thesis, Albert Nerken School of Engineering, 1991.
  Site*: http://citeseer.ist.psu.edu/slyz91image.html*

- **[Shi00]** Shi Y. , Sun H., "Image and Video Compression for Multimedia Engineering", International standard book, library of Congress Cataloging, USA, 2000.

- **[Saf02]** Saffor A., Rahman bin Ramli A., Hoong Ng K., Dowsett D., "Objective and Subjective Evaluation of Compressed Computed Tomography (CT) Images" , The Internet Journal of Radiology, 2002.
  Site: *http://www.ispub.com/ostia/index.php?xmlFilePath=index.xml*

- **[Umb98]** Umbaugh Scott E., "Computer Vision and Image processing", Prentic-hall, Inc., USA, 1998.

- **[Vij06]** Vijay Shah P., Nicholas H., Surya Durbha S., Roger K. L.," Data Transformation for Primitive Feature Extraction in Image Information Mining", Department of Electrical and Computer Engineering, Mississippi State University, 2006.
  Sites: *vps1@ece.msstate.edu*
  suryad@gri.msstate.edu
  http:// earth.esa.int/rtd/Events/ESA-EUSC_2006/Poster/Ar61_Younan.pdf

- **[Web1]** "Fractal", Wikipedia, the free encyclopedia, 2006.
  Site*: http:// en.wikipedia.org/wiki/Fractal*

- **[Wil97]** Wilson N., "Image Compression Teqnichues", Linktionary Encyclopedia of Networking Defined and Hyperlink, second edition, 1997.
  Site: *http://www.linktionary.com/eon2.html*

- **[Woh99]** Wohlberg B., Jager G.,"A Review of the Fractal Image Coding Literature", IEEE Transactions on Image Processing, vol. 8, no. 12, pp. 1716-1729, December 1999.

Site:
http://math.lanl.gov/~brendt/Publications/Docs/wohlberg-1999-review.pdf

- **[Xia01]** Xiao P.," Image Compression by Wavelet Transform", M.Sc. Thesis, Computer and Information Sciences, East Tennessee State University, 2001.
  Site*: worldcat.org/wcpa/ow/34a283c97a65f3bba19afeb4da09e526.pdf*

- **[Zaf02]** Zafarifar B.," Micro codable Discrete Wavelet Transform", Computer Engineering Laboratory, Faculty of Information Technology and Systems , Delft University of Technology The Netherlands, 2002.
  Site*: http://einstein.et.tudelft.nl/~george/students/micro_codable.pdf*

- **[Zak97]** Zaki T., " Advanced Compression Techniques For Multimedia Applications", M.Sc. thesis ,College of science , AL-Nahrain University , 1997

Table of Contents

```
[.ShellClassInfo]
LocalizedResourceName=@%SystemRoot%\system32\shell32.dll,-21815
```