Lossless Image Compression
Using Prediction Coding and LZW Scheme

A Dissertation
Submitted to College of Science of Baghdad University in Partial Fulfillment of the Requirements for the Degree of High Diploma of Science in Computer Science

By
Huda Mahmood Hassoon
(B.Sc. 2008)

Supervisor
Dr. Loay E. George

November 2011 Thee-AL-Hejaah 1432
Supervisor Certification

I certify that this Dissertation was prepared under our supervision at the Department of Computer Science/ College of Science/ Baghdad University, by Huda Mahmood Hassoon as partial fulfillment of the requirements for the degree of High Diploma of Science in Computer Science.

Signature:
Name : Dr. Loay E. George
Title : Assist. Prof.
Date  : 1 / 11 / 2011

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

Signature:
Name : Dr. Loay E. George
Title : Head of Computer Science Department, College of Science University of Baghdad.
Date  : 1 / 11 / 2011
Certification of the Examination Committee

We chairman and members of the examination committee, certify that we have studied this dissertation entitled "Lossless Image Compression Using Prediction and LZW Scheme" presented by the student Huda Mahmood Hassoon. We examined her in its contents and found it is worthy to be accepted for the degree of High Diploma of Science in Computer Science.

Signature:
Name: Dr. Faisel Ghazee Mohammed
Title: Assist. Prof.
Date: 30 / 11 /2011

(Chairman)

Signature:
Name: Dr. Bara’a Ali Attea
Title: Assist. Prof
Date: 30 / 11 /2011

(Member)

Signature:
Name: Dr. Loay E. George
Title: Assist. Prof
Date: 30 / 11 /2011

(Supervisor)

Approved by the Dean of the College of Science, Baghdad University.

Signature:
Name: Dr. Saleh Mehdi Ali
Title: Prof
Date: 30 / 11 /2011

(Dean of College of Science)
DEDICATED TO

My Parents ...
My Sisters ...
My Brother ...

To everyone
Taught me a letter

Huda
Acknowledgment

First of all great thanks are due to Allah who helped me and gave me the ability to achieve this research from first to last step.

I would like to express my deep gratitude and sincere thanks to my supervisor Dr. Loay E. George for guidance, assistance and encouragement during the course of this project.

Grateful thanks for the Head of Department of Computer Science of Baghdad University for the continuous support during the period of my studies.

Deep gratitude and special thanks to my family: my parent, brother, and sisters for their encouragements and supporting to succeed in doing this work.

Special thanks to the staff of Baghdad University for their general support and help. Special thanks go to all people who provided me with any kind of help all my friends for giving me advises.
Abstract

One of the most important problems in multimedia applications is the storage and transmission of image, video and audio data. This made the field of developing image compression methods necessary and vital. Different image compression techniques were proposed to achieve high compression ratios and high image qualities in low computation time. Among the introduced compression schemes are predictive coding and dictionary based methods (like, LZW). Predictive coding is based on pruning the existing spatial redundancy in the image. while LZW is the most commonly used as lossless compression scheme to prune the existing statistical redundancy in the data set.

In this project a combined compression scheme is introduced to compress color images. Each color component (i.e., red, green and blue) is subjected to delta coding (i.e., first order predictor) in order to prune the existing local spatial redundancy between adjacent pixels. Then, the output is passed through LZW coding to compress the data according to statistical bases. In order to get lossless compression no quantization is applied on the output of predictive coding.

The conducted tests on the introduced system indicated that the attained compression ratio is better than that obtained by directly applying LZW on the image.
# List of Abbreviation

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-D</td>
<td>One – Dimensional</td>
</tr>
<tr>
<td>2-D</td>
<td>Two – Dimensional</td>
</tr>
<tr>
<td>ASCII</td>
<td>American Standard Code for Information Interchange</td>
</tr>
<tr>
<td>BMP</td>
<td>Bit Map Picture</td>
</tr>
<tr>
<td>BPP</td>
<td>Bit Per Pixel</td>
</tr>
<tr>
<td>BR</td>
<td>Bit Rate</td>
</tr>
<tr>
<td>CF</td>
<td>Compression Factor</td>
</tr>
<tr>
<td>CR</td>
<td>Compression Ratio</td>
</tr>
<tr>
<td>DCT</td>
<td>Discrete Cosine Transform</td>
</tr>
<tr>
<td>DPC</td>
<td>Differential Predictive Coding</td>
</tr>
<tr>
<td>DPCM</td>
<td>Differential Pulse Code Modulation</td>
</tr>
<tr>
<td>DWT</td>
<td>Discrete Wavelet Transform</td>
</tr>
<tr>
<td>GIF</td>
<td>Graphic Interchange Format</td>
</tr>
<tr>
<td>FIC</td>
<td>Fractal Image Compression</td>
</tr>
<tr>
<td>HVS</td>
<td>Human Visual System</td>
</tr>
<tr>
<td>JPEG</td>
<td>Joint Photographic Expert</td>
</tr>
<tr>
<td>LZW</td>
<td>Lempel – Ziv – Welch</td>
</tr>
<tr>
<td>PIFS</td>
<td>Partitioned Iterated Function System</td>
</tr>
<tr>
<td>RGB</td>
<td>Red-Green-Blue</td>
</tr>
<tr>
<td>RLC</td>
<td>Run - Length Coding</td>
</tr>
<tr>
<td>RLE</td>
<td>Run - Length Encoding</td>
</tr>
<tr>
<td>VQ</td>
<td>Vector Quantization</td>
</tr>
</tbody>
</table>
List of Contents

Chapter One: General Introduction 1-6
1.1 Introduction................................................................. 1
1.2 The Principles Behind Compression ......................... 2
1.3 Image Compression Technique................................. 4
1.4 Literature Survey.......................................................... 5
1.5 Aim of Dissertation....................................................... 6
1.6 Dissertation Layout......................................................... 6

Chapter Two: Compression Techniques Background 7-23
2.1 Introduction................................................................. 7
2.2 Image Compression......................................................... 7
2.3 Data Redundancy............................................................. 9
2.3.1 Coding Redundancy .................................................... 9
2.3.2 Interpixel Redundancy ............................................... 9
2.3.3 Psychovisual Redundancy ........................................... 9
2.4 Image Compression Methods.......................................... 10
2.4.1 Lossless Compression Methods................................. 11
2.4.2 Lossy Compression Techniques..................................... 20

Chapter Three: The System Implementation 24-36
3.1 Introduction................................................................. 24
3.2 The System Model.......................................................... 24
3.3 System Implementation.................................................... 25
3.3.1 Load BMP Image....................................................... 26
3.3.2 Image data compression............................................ 28
3.3.3 Image data decompression....................................... 30
3.3.4 Save bitmap file....................................................... 33
3.5 Performance Test Result............................................... 35
3.6 Image Test Material....................................................... 35

Chapter Four: Conclusion and Suggestions for Feature Work 37
4.1 Conclusion........................................................................................................... 37
4.2 Suggestions for Feature Work................................................................. 37

References ........................................................................................................... 38-40
List of Figure

Figure (2.1) The most popular image compression 11
Figure (2.2) The LZW compression algorithm 15
Figure (2.3) The LZW decompression algorithm 17
Figure (2.4) A lossless predictive coding model 20
Figure (2.5) Block diagram of predictive coding method 22
Figure (3.1) The system model 25
Figure (3.2) View the bitmap color image 34
Figure (3.3) Winrar archive to bitmap image 34
Figure (3.4) The bitmap image used as test image 36

List of Table

Table (2.1) Lossless vs. Lossy compression 10
Table (2.2) An example of the LZW compression process 16
Table (2.3) An example of the LZW decompression process 18
Table (3.1) The results of the introduced compression method 36
Chapter One

General Introduction
1.1 Introduction

A number of methods have been presented over the years to perform image compression. They all have one common goal to alter the representation of information contained in an image so that it can be represented sufficiently well with less information. Regardless of the details of each image compression method, the methods can be classified into two general categories: lossy or lossless. For methods in the first category, some information from the original image is lost, even if only a small amount. Conversely, lossless compression methods provide a perfect reproduction of the original image [Ham06].

Still image compression is an important issue in internet, mobile communications, digital library, digital photography, multimedia, teleconferencing applications, etc. It would be obvious that the present and future applications in multimedia and other areas would focus on the problem of optimizing storage space and transmission bandwidth. In many of these applications, big savings in terms of bit rate can be achieved by a tolerable loss of quality [Ham06].

Data compression "is the process of converting data files into smaller files for efficiency of storage and transmission". It is one of the enabling technologies of the multimedia revolution, and considered as a key to rapid progress being made in information technology.

Data compression treats information in digital form as binary sequence holds spatial and statistical redundancy. The relatively high cost of storage and transmission makes data compression worthy. compression is considered necessary and essential key for creating image files with manageable and transmittable sizes [Alb07].
Image compression methods are based on either redundancy reduction or irrelevancy reduction; while most compression methods exploit both. The parts of a coder that process redundancy and irrelevancy are separate in some methods; while in other methods they cannot be easily separated [Gha04].

1.2 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 is, then, to find less correlated representation of the image.

The two fundamental components of compression are redundancy and irrelevancy reduction:

1. Redundancy reduction: aims to duplication removal 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, four types of redundancy can be identified:

1. *Spatial Redundancy* or correlation between neighboring pixel values.
2. *Spectral Redundancy* or correlation between different color planes or spectral bands.
3. *Temporal Redundancy* or correlation between adjacent frames in a sequence of images (in video applications).
4. *Physic logical Redundancy* or correlation between volumabilty in recognizing some of the information region [Alb07].

Any image compression research aims to reduce the number of bits needed to represent an image by removing the spatial and spectral redundancies as much as possible [Sah04].
Digital images have broad application areas, such as: Internet browsing, TV transmission, video conferencing, transmission of remotely sensed images and printing. The vast amount of data kept to represent a digital data image restricts these applications.

The basic goal of image compression is to reduce the bit rate of an image to minimize the communication channel capacity or digital storage memory requirements; while maintaining necessary fidelity in the image, or, equivalently, to obtain the best possible fidelity for a given bit rate. The bit rate is measured in bits per pixel (bpp). The raw (uncompressed) bit rate is typically 8 bits per pixel for a gray-images and 24 bits per pixel for a color image (with three 8-bit components)[Gha04].

Almost all methods of image compression are based on two fundamental principles[Gha04]:

1. The first principle is to exploit the properties of the signal sources, i.e., the statistical property, and to remove redundancy from the signal. This approach is called redundancy reduction. Almost all sampled signals in coding are redundant because sampling typically tends to preserve some degree of inter sample correlation. This redundancy is reflected in the form of a non flat power spectrum. Greater degrees of non flatness lead to greater gains from redundancy removal. These gains are also referred to as prediction gains or transform coding gains, depending on whether the redundancy is processed in the time domain or frequency (or transform) domain.

2. The second principle is to exploit the properties of the signal receiver (usually the human visual system) and to remove parts or details of the signal that will not be noticed by the receiver. This approach is called irrelevancy reduction. The idea is to quantize the sample or transform coefficients just finely enough to leave an imperceptibly distorted result, even though the quantized quantity is not
mathematically zero. If the available bit rate is not sufficient to realize this kind of perceptual transparency, the intent is to minimize the perceptibility of the distortion.

1.3 Image Compression Techniques

In general, image compression techniques can be broadly classified into [Lee11]:

1. Lossless compression
2. Lossy compression

Lossless compression, also called noiseless coding [Gha04], allows exact reconstruction of each individual pixel value. This method is sometimes referred to as image coding, rather than image compression [Sob00]. Lossless compression methods are often based on redundancy reduction. With lossless compression, the compressed data can be exactly restored so as to be identical to the original [Gha04]. Lossless compression techniques guaranty full reconstruction of the original data without incurring any distortion in the process. Lossless Image coding is also important in applications where no information loss is allowed during compression [Abh02].

In lossy compression a perfect reconstruction of the image is sacrificed by the elimination of some amount of redundancies in the image to achieve higher compression ratio. However, no visible loss of information is perceived under normal viewing conditions [Cho01]. 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].
1.4 Literature Survey

There are many related work on data compression, we will include some of them:

1. Fleh S. K. [Fle05], Discusses two hybrid systems have been designed and implemented. The first one consists of one dimensional DCT the differential pulse code modulation (DPCM) and entropy coding (Huffman code) the difference signal. The DCT is applied row-wise and the DPCM is applied column-wise to transform image, finally the difference signal is Huffman code. The second hybrid system uses an expert system called the learning automata LA to code the difference signal obtained from hybrid DCT/DPCM system.

2. Zaki, [Zak97], studied the application of lossless data compression on multimedia 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. He proposed a slight modification for handling the existing repetitions that used in standard RLE.

3. Hammudy H. H. [Ham06], this thesis attempts to give recipe for selecting two proposed image compression algorithm based on wavelet and multiwavelet approaches as well as to make comparison of these approaches on gray–scale image. Image compression using wavelet transform was first achieved using Daubchies 4 basis function. It is applied to each $8 \times 8$ block of the image.

4. Attea B.A [Att96], this work deals with securing important image documents. Image ciphering is performed through image transformation and scrambling. Three types of transforms are considered. These are Fast Fourier Transform, Fast Hartley
Transform, and Fast Walsh Transform. The image to be ciphered is partitioned into a random number of random sized blocks, each is transformed randomly by one of the three transforms, then image pixels are scattered both locally within new blocks, and globally within the whole image. Ciphering images are stored in guarded GIF or 24-bit PCX format. Guarding GIF file format consist of LZW Compression (Decoder and Encoder Algorithm).

1.5 Aim of Dissertation
In this project, a first order linear predictor is used to reduce the spatial existing redundancy in color image data. Then the outputs of linear prediction stage will be fed to LZW to get more effective compression gain.

1.6 Dissertation Layout
In addition to chapter one that gives a general introduction to the compression process, this dissertation includes other three chapters:

**Chapter Two: “Compression Techniques Background”**
Describe the basic backgrounds of the image compression methods including the lossless methods. And, to penetrate deeply into the conceptual reviews about predictors method and LZW.

**Chapter Three: “The System Implementation”**
It presents the experimental results obtained from applying the studied method on some selected test image to evaluate the system performance.

**Chapter Four: “Conclusions and Future Work”**
Provides some derived conclusions about this work and some suggestions for the future work.
الخلاصة

تعتبر مشاكل خزن الصور الرقمية ونقلها من أهم المشاكل في تطبيقات الحاسوب وهذا
م يجعل تطوير طرق ضغط هذه الصور يكتسب أهمية خاصة. استحدثت طرق عديدة باستخدام
تقنيات مختلفة و الغرض منها تحقيق نسب ضغط عالية مع المحافظة على جودة الصور
المضغوطة، وأخذ بنظر الاعتبار انزال الضغط بأقل وقت ممكن. و إحدى التقنيات المستخدمة
هي طريقة الضغط بدون فقدان البيانات هي إحدى هذه الطرق وبالنتيجة فإن هذا سوف يقود إلى
تسريع مهمة البحث.

( predictive schema )
إحدى طرق الضغط بدون فقدان البيانات هي التنبؤ و تعتبر من التقنيات المستحدثة.
طريقة الضغط بدون فقدان المعلومات باستخدام التنبؤ تجعل البيانات أكثر ترتيبا و تعمل
على إزالة الفائض المكاني في الصورة وبعد ذلك نستخدم طريقة ( LZW ) التي عادة تستخدم
في ضغط الصور بدون فقدان المعلومات بإضافتها على مجموعة البيانات الناتجة من طريقة
التنبؤ.

في هذا البحث تم دمج طريقة التنبؤ وطريقة ( LZW ) لإنتاج صور ملونة مضغوطة وهذه
المكونات هي الأحمر والأزرق والأخضر لتعرض ال ( Delta Coding ) مثل طريقة التنبؤ
من الدرجة الأولى لترتيب الفائض في الحيز المكاني بين النقاط المجاورة. وبعد ذلك البيانات
المضغوطة الناتجة تمر خلال ( LZW ) لإعطاء ضغط صور بدون فقدان و بدون عملية تكميم في حالة عرض نتائج تطبيق التنبؤ.

نتيجة التجارب أشارت إلى أن نسبة الضغط أفضل في حالة دمج طريقة التنبؤ وطريقة
( LZW ) من استخدام طريقة فقط.
Chapter Two

Compression Techniques
Background
Compression Techniques

Background

2.1 Introduction

Normally, general data compression schemes does not take into account the type of data, which is being compressed, because it should be lossless. It can be applied to computer data files, documents, images, and so on.

In this chapter, we present some common lossless compression algorithms, Since many compression algorithms are applied to graphical images, and this work is concerned with compressing images; so the basic concepts of digital image storage are also discussed.

2.2 Image Compression

Image compression addresses the problem of reducing the amount of data required to represent a digital image [Lee11]. The underlying basis of the reduction process is “redundancy removal”. This amounts to transforming a 2-D pixel array into a statistically uncorrelated data set. The transformation is applied prior to storage or transmission of the image. At some later time, the compressed image is decompressed to reconstruct the original image or an approximation of it [Hor00]. A compression algorithm can be evaluated in a number of different ways. We could measure the relative complexity of the algorithm, the memory required to implement the algorithm, how fast the algorithm is performed on a given machine, the amount of compression, and how closely the reconstruction resembles the original [Abh02].

A very logical way of measuring how well a compression algorithm compresses a given set of data is to look at the ratio of the number of bits required to represent the data before compression to the number of bits
required to represent the data after compression. This ratio is called the compression ratio. Suppose an image made up of a square array of 256x256 pixels and each pixel require 8 bits to represented its color, then its storage requires 65,536 bytes. If this image is compressed and the compressed version requires 16,384 bytes, then we can would say that the attained compression ratio is 4:1 [Abh02].

In order to be useful, a compression algorithm has a corresponding decompression algorithm that gives the compressed file, in other words, reproduces the original file. Image compression involves reducing the size of image data files, while retaining its necessary information. The pre-reduced file is called the compressed file and is used to reconstruct the image, resulting in the decompressed image. The original image, before any compression was performed, is called uncompressed image file. The ratio of the original image (uncompressed image) and the compressed file is referred to as the compression ratio (CR):

\[
CR = \frac{\text{Uncompress image size (in bits)}}{\text{Compress file size in (bits)}} \tag{2.1}
\]

In some cases, the Compression Factor (CF) is more convenient to use, and defined as reciprocal of compression ratio, that is:

\[
CF = \frac{\text{Compress image size (in bits)}}{\text{Uncompress file size in (bits)}} \tag{2.2}
\]

Another way of reporting compression performance is to provide the average number of bits required to represent a single sample. This is generally referred to as the Bit Rate. For example, if we assume 8 bits per pixel, the average number of bits per pixel in the compressed representation is 2. Thus, the rate is 2 bits per pixel [Abh02]. The Bit Rate (BR) and the compression ratio are simply related as follows:
Compression algorithm are developed by taking advantage of the redundancy that is inherent in image data.

2.3 Data Redundancy

In digital image compression, three basic data redundancies can be identified and exploited: Coding redundancy, interpixel redundancy and psychovisual redundancy. Data compression is achieved when one or more of these redundancies are reduced or eliminated [Ali01, Hor00].

2.3.1 Coding Redundancy

It occurs when the data used to represent the image are not utilized in an optimal manner. For example, if we have an 8 bits/pixel image that allow 256 gray-level values, but the actual image contains only 16 gray-level values, this is a sub optimal coding because only 4 bits/pixel are actually needed.

2.3.2 Interpixel Redundancy

It occurs because of adjacent pixels tend to be highly correlated. This is a result of the fact that in most images the brightness do not change rapidly, it changes gradually, so that adjacent pixel values tend to be relatively close to each other in value.

2.3.3 Psychovisual Redundancy

It refers to the fact that some information is more important to the human visual system (HVS) than other types of information, it can eliminated without significantly impairing the quality of image perception. Eliminating these types of psychovisual redundancy leads to data compression. This redundancy can be removed by accomplishing some
Chapter Two

Compression Methods

kinds of compression techniques; such as Run Length Encoding, Huffman Coding, Predictive Coding, and Transform Coding [Ali01, Hor00].

Table (2.1) Lossless vs. Lossy compression

<table>
<thead>
<tr>
<th>Lossless</th>
<th>Lossy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preserves image quality perfectly</td>
<td>Degrades image quality</td>
</tr>
<tr>
<td>Provides low compression ratio</td>
<td>Provides a significant amount of Compression ratio.</td>
</tr>
<tr>
<td>Exploits coding redundancy or interpixel redundancy</td>
<td>Exploits the psychovisual Redundancy</td>
</tr>
<tr>
<td>Lossless operation is reversible</td>
<td>Lossy operation is irreversible</td>
</tr>
<tr>
<td>Does not contain quantization stage</td>
<td>Contains quantization stage</td>
</tr>
<tr>
<td>Huffman coding, arithmetic coding, and Lempel-Ziv algorithm are examples of lossless techniques</td>
<td>Vector quantization, fractal compression, DCT, and wavelet are examples of lossy techniques</td>
</tr>
<tr>
<td>Used in applications such as archiving of medical or business documents</td>
<td>Used in applications such as broadcast TV, video, and facsimile transmission</td>
</tr>
</tbody>
</table>

2.4 Image Compression Methods

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

Based on the difference between original (x) and reconstructed version (y), data compression schemes can be divided into two broad classes, (see Figure 2.1). The first class is called lossless compression, at which \( Y \) is identical to \( X \), while other class is called lossy compression,
which generally provides much higher compression than lossless compression but makes Y not exactly as X [Add00].

![Diagram of Compression Methods]

Fig (2.1) The most popular image compression

### 2.4.1 Lossless Compression Methods

Lossless compression techniques provide the guaranty that any pixel in the decompressed image is exactly the same as in the original one. (i.e lossless schemes result in reconstructed data that exactly matches the original). It is generally used for applications that cannot allow any difference between the original and reconstructed data. The most popular lossless compression methods are: Run length encoding, LZW, Arithmetic
coding, Huffman coding, S-shift coding. The most well known compression methods are:

A. Run-Length Encoding (RLE)

Run length encoding (RLE) is based on the idea of encoding a consecutive occurrence of the same symbol. This is achieved by replacing a series of repeated symbols with a count and the symbol. RLE finds runs of repeated symbols in the input stream and replaces them with a two-bytes code. The code consists of repeated symbols, and the code of a count byte. For instance, the string ‘AAAAAABBBBCCCCC’ could be more efficiently represented as ‘A6 B4 C5’; that saves six bytes. The following example illustrates the use of RLE to encode an input data with frequent runs of zeros [Mar01]:

Original data: 12000131415000000900

Encoded data:

| 1 | 2 | 0 | 3 | 13 | 14 | 15 | 0 | 6 | 9 | 0 | 2 |

If the data contains a large number of consecutive zeros, the data size can be greatly reduced using RLE. On the other hand, if the content of the data is random, then this encoding technique might increase the data size.

B. Huffman Coding

It is the most popular technique for removing coding redundancy. When the symbols of an information source are coded individually, Huffman coding yields the smallest possible number of code symbols per source symbols. This method is started with a list of the probabilities of the image data elements. Then, take the two least probable elements and
make them two nodes with branches (labeled “0” and “1”) to a common node which represents a new element. The new element has a probability, which is the sum of the two probabilities of the merged elements. The procedure is repeated until the list contains only one element [Alk04].

C. Lempel-Ziv-Welch (LZW)

Around 1977, Abraham Lempel and Jacob Ziv developed the Lempel-Ziv class of adaptive dictionary data compression techniques. Also known as LZ-77 coding. Now there are many popular variants of LZ compression techniques; the original Lempel-Ziv-Welsh (LZW) algorithm (also known LZ-78) builds a dictionary of frequently used groups of characters (or 8-bit binary values). Before the file is decoded, the compression dictionary must be sent (when transmitting data) or stored (if data is being stored). This method is good for compressing text files because text files contain ASCII characters (which are stored as 8-bit binary values) but not so good for real image files, which may have repeating patterns of binary digits that might not be multiples of 8 bits. This is a popular variant of LZ78, developed by Terry Welch in 1984. Its main feature is eliminating the second field of a token. An LZW token consists of just a pointer to the dictionary. The dictionary is an array of variable-size string. The LZW method starts by initializing the dictionary to all the symbols in the alphabet. In the common case of 8-bit symbols, the first 256 entries of the dictionary (entries 0 through 255) are occupied before any data is input. Because the dictionary is initialized, the next input character will always be found in the dictionary. This is why an LZW token can consist of just a pointer and does not have to contain a character code as in LZ78.

The principle of LZW is that the encoder inputs symbols one by one and accumulates them in a string I. After each symbol is input and is concatenated to I, the dictionary is search for string I. As long as I is
found in the dictionary, the process continues. At a certain point, adding
the next symbol \( x \) causes the search to fail; string I is in the dictionary but
string \( Ix \) (symbol \( x \) concatenated to I) is not.

At this point the encoder

1. outputs the dictionary pointer that points to string I
2. saves string \( Ix \) (which is now called a phrase) in the next
   available dictionary entry, and
3. initializes string I to symbols \( x \).

To illustrate this process, we will use the text string “sir sid easily
teases sea sick seals”.

The steps of algorithm are as follows:

0. Initialize entries 0-255 of the dictionary to all 256 8-bit bytes.
1. The first symbol s is input and found in the dictionary (in entry 115,
since this is the ASCII code of s). The next symbol I is input, but si is
not found in the dictionary.

The encoder performs the following

1. Outputs 115
2. saves string si in the next available dictionary entry (entry 256)
3. initializes I to the symbol i.

2. The r of sir is input, but string ir is not in the dictionary. The encoder
(1) outputs 105 (the ASCII code of i), (2) saves string ir in the next
available dictionary entry (entry 257), and (3) initializes I to the symbol r.

Since the first 256 entries of the dictionary are occupied right from the
start, pointers to the dictionary have to be longer than 8 bits. A simple
implementation would typically use 16-bit pointers, which allows for a
64K-entry dictionary (where \( 64K=2^{16}=65,536 \)). Such a dictionary will,
of course, fill up very quickly in all but the smallest compression jobs.

Another interesting fact about LZW is that strings in the dictionary get
only one character longer at a time. It therefore takes a long time to get
long strings in the dictionary, and thus a chance to achieve really good compression. We can say that LZW adapts slowly to its input data [Abb05].

For instance of 12 bit codes, the codes from 0 to 255 are individual bytes and codes 256 to 4095 are assigned to substrings.

1. **Compression:** A simple form of the LZW compression algorithm is shown in Figure (2.2). The algorithm shows that the LZW outputs codes for known strings and adds a new string to the string table when a new code is output [Lee10].

```
**Pseudo Code List(2.1) Routine LZW_COMPRESS**

STRING = get input character
WHILE there are still input characters DO
    CHARACTER = get input character
    IF STRING + CHARACTER is in the string table then
        STRING = STRING + character
    ELSE
        output the code for STRING
        add STRING + CHARACTER to the string table
        STRING = CHARACTER
    END of IF
END of WHILE
output the code for STRING
```

Fig. (2.2) The LZW compression algorithm

An example of the compression process with input string “/WED/WE/WEE/WEB/WET” is demonstrated in Table (2.2). The algorithm first runs through the while loop to check if the string “/W” is
in the table. Obviously it is not at the initial stage so the algorithm output “/” and add the string “/W” into the string table. The newly added string is assigned to code 256, right after the defined codes from 0 to 255. The third letter “E” is then read and the second string code “WE” is added to the string table while the letter “W” is output. The similar process is repeated until the repeated characters “/” and “W” are read. These characters match the number 256 string already defined in the string table; therefore, the code 256 is output and a new string “/WE” is added into the string table.

Table (2.2) An example of the LZW compression process

<table>
<thead>
<tr>
<th>New String</th>
<th>New Value</th>
<th>Code</th>
<th>Code Output</th>
<th>Character Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>/W</td>
<td>256</td>
<td>/</td>
<td>/W</td>
<td></td>
</tr>
<tr>
<td>WE</td>
<td>257</td>
<td>W</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>ED</td>
<td>258</td>
<td>E</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>D/</td>
<td>259</td>
<td>D</td>
<td>/</td>
<td></td>
</tr>
<tr>
<td>/WE</td>
<td>260</td>
<td>256</td>
<td>WE</td>
<td></td>
</tr>
<tr>
<td>E/</td>
<td>261</td>
<td>E</td>
<td>/</td>
<td></td>
</tr>
<tr>
<td>/WEE</td>
<td>262</td>
<td>260</td>
<td>WEE</td>
<td></td>
</tr>
<tr>
<td>E/W</td>
<td>263</td>
<td>261</td>
<td>/W</td>
<td></td>
</tr>
<tr>
<td>WEB</td>
<td>264</td>
<td>257</td>
<td>EB</td>
<td></td>
</tr>
<tr>
<td>B/</td>
<td>265</td>
<td>B</td>
<td>/</td>
<td></td>
</tr>
<tr>
<td>/WET</td>
<td>266</td>
<td>260</td>
<td>WET</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
<td></td>
<td>EOF</td>
<td></td>
</tr>
</tbody>
</table>

2. **Decompression:** The decompression algorithm requires the stream of codes output from the compression algorithm in order to reconstruct the original input. The LZW algorithm is considered efficient because
a large string table created in the compression process does not need to be passed to the decompression side. The decompression algorithm shown in Figure (2.3) allows us to recreate the exact string table that was built in the compression process.

![Fig. (2.3) The LZW decompression algorithm](https://example.com/image)

**Pseudo Code List(2.2) Routine LZW DECOMPRESSION**

```
Read OLD_CODE
Output OLD_CODE
WHILE there are still input characters DO
    Read NEW_CODE
    STRING = get translation of NEW_CODE
    Output STRING
    CHARACTER = first character in STRING
    Add OLD_CODE + CHARACTER to the translation table
    OLD_CODE = NEW_CODE
END of WHILE
```

Every time when the decompression algorithm reads in a new code, it adds a new string to the string table. An example of decompression process is demonstrated in Table (2.3). One can easy notice that the string table is identical to the one built during the compression process and the output stream is the same as the input string from the compression algorithm.
Table (2.3) An example of the LZW decompression process

<table>
<thead>
<tr>
<th>New table entry</th>
<th>CHARAC TER</th>
<th>STRING/ Output</th>
<th>OLD_COD</th>
<th>Input/ NEW_CO DE</th>
</tr>
</thead>
<tbody>
<tr>
<td>256 = /W</td>
<td>W</td>
<td>W</td>
<td>/</td>
<td>W</td>
</tr>
<tr>
<td>257 = WE</td>
<td>E</td>
<td>E</td>
<td>W</td>
<td>E</td>
</tr>
<tr>
<td>258 = ED</td>
<td>D</td>
<td>D</td>
<td>E</td>
<td>D</td>
</tr>
<tr>
<td>259 = D</td>
<td>/</td>
<td>/W</td>
<td>D</td>
<td>256</td>
</tr>
<tr>
<td>260 = /WE</td>
<td>E</td>
<td>E</td>
<td>256</td>
<td>E</td>
</tr>
<tr>
<td>261 = E/</td>
<td>/</td>
<td>/WE</td>
<td>E</td>
<td>260</td>
</tr>
<tr>
<td>262 = /WEE</td>
<td>E</td>
<td>E/</td>
<td>260</td>
<td>261</td>
</tr>
<tr>
<td>263 = E/W</td>
<td>W</td>
<td>WE</td>
<td>261</td>
<td>257</td>
</tr>
<tr>
<td>264 = WEB</td>
<td>B</td>
<td>B</td>
<td>257</td>
<td>B</td>
</tr>
<tr>
<td>265 = B</td>
<td>/</td>
<td>/WE</td>
<td>B</td>
<td>260</td>
</tr>
<tr>
<td>/WET</td>
<td>T</td>
<td>T</td>
<td>260</td>
<td>T</td>
</tr>
</tbody>
</table>

A horizontal difference predictor is applied before LZW to further improve compression ratio. A difference predictor allows LZW to compact the data more compactly due to the fact that many continuous-tone images do not have much variation between the neighboring pixels. This means that many of the differences should be 0, 1, or -1. The combination of LZW coding with horizontal differencing is lossless [Lee10].

**D. Arithmetic Coding**

Arithmetic coding, and its derivative technique, (i.e. Q-coding) is used to overcome some of the limitations of Huffman codes. It is a non-
block coding scheme, in that a single codeword is used to represent a sequence of input symbols; in contrast to Huffman coding where a source symbol block corresponds to a codeword block. Instead, it uses the real numbers to represent a sequence of symbols by recursively subdividing the interval between 0 and 1 to specify each successive symbol. The limitation of this technique is the precision required in performing the calculations and arriving at the code word which will represent the entire sequence correctly.

E. Lossless predictive Coding

Lossless predictive coding, is based on eliminating the interpixel redundancies of closely spaced pixels by extracting and coding only. The new information of a pixel is defined as the difference between the actual and predicted value of that pixel [Gon02].

Figure (2.4) shows the basic component of a lossless predictive coding system. The system consist of an encoder and a decoder, each containing an identical predictor. As each successive pixel of the input image, denoted $f_n$, is introduced to the encoder, the predictor generates the anticipated value of that pixel based on some number of past inputs. The output of the predictor is then rounded to the nearest integer, denoted $\hat{f}_n$, and used to form the difference or prediction error [Gon02].

$$e_n = f_n - \hat{f}_n \tag{2.4}$$
2.4.2 Lossy Compression Techniques

Lossy compression techniques involve some loss of information, and data cannot be recovered or reconstructed exactly. In some applications, exact reconstruction is not necessary. For example, it is acceptable that a reconstructed video signal be different from the original as long as the differences do not result in annoying artifacts. However, we can generally obtain higher compression ratios than is possible with lossless compression [Cho01]. The most well known lossy compression methods are:
A. 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. 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 inferior compression ratios and worse reconstructed image quality than those of transform coding. 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 used to represent the image [Xia01]. The differential code is also called differential pulse code modulation (DPCM) or known predictive coding, by utilizing spatial and temporal interpixel correlation. Differential predictive coding works by predicting the next pixel value based on the previous values and encoding the differences between the predicted value and the actual value [Uma06]. In the simplest case of linear prediction, the predicted value is a neighboring pixel value stored in the predictor. More complicated linear predictors calculate the predicted value as a linear combination of the value of the neighboring pixels. The decoder must receive initial value to start its predictor. This technique takes the advantage of the fact that the adjacent pixels are highly correlated, which means the difference is typically small. Because this difference is small, it will take only a small number of bits to represent it. This allows us to further reduce the data and optimize visual results, and then this difference is quantized and then can be coded.

A block diagram of this process is shown in Figure (2.5), where we can see that the predictor must be in feedback loop so that it matches the decompression system. The system must be initialized by retaining the first value (s) without any compression in order to calculate the first
predication. From the block diagram, we have the following: \( \tilde{I} \) is the predicted next pixel value, \( \hat{I} \) is the reconstructed pixel value, \( e = \hat{I} - \tilde{I} \) = error, and \( \hat{e} \) is the quantized error.

The prediction equation is typically a function of the previous pixel(s) and can also include global or application-specific information. The theoretically optimum predictor, using only the previous value and based on minimizing mean-squared error between the original and the decompressed image, is given by:

\[
\tilde{I}(r,c + 1) = p\tilde{I}(r,c) - (1 - p) I(r,c), \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ ld
B. Vector Quantization

The basic idea of Vector Quantization (VQ) is to compress a group of pixels (vector) together instead of a single value at a time, i.e., use a code to represent the vector [Gha04].

C. Transform Coding

A transform-based compression method generally implies a quantization step. Transform based compression is one of the most useful applications. It can be combined with other compression techniques in order to achieve efficient compression gain.

D. Fractal Compression:

The application of fractals in image compression started with M.F. Barnsley and A. Jacquin. Fractal image compression is a process to find a small set of mathematical equations that can describe the image. By ending the parameters of these equations to the decoder, we can reconstruct the original image. In general, the theory of fractal compression is based on the contraction mapping theorem in the mathematics of metric spaces [Alk04].
Chapter Three

The System Implementation
The System Implementation

3.1 Introduction

This Chapter is dedicated to present the layout of the proposed lossless compression method. Also, the taken steps to implement the method are clarified. The proposed system consist of two different stages; the first is predictive coding to get compression by exploiting the existence of local redundancy between adjacent pixels, and the second stage is the LZW coding to get further compression gain by exploiting the existing statistical redundancy.

3.2 The System Model

The general structure of the proposed system is illustrated in Figure (3.1). It is consist of two basic modules: Compression and Decompression modules. The input to the compress module is BMP (Bitmap) image file. The data of this image is passed through five stages, and subjected to various operations to produce the compressed file. This compressed file could be passed through the decoding stages, and subjected to sequence of operations, to reconstruct the bitmap image. Each module (i.e. compressing and decompression) implies several stages, working systematically to lead to the final result.

As shown in Figure (3.1a) the main stages of the compression model are started from loading image data, and passed through predictive coding, and LZW coding. The output of the last stage are saved in compressed file.
Chapter Three                                               The Proposed System Implementation

3.3 System Implementation

In this section, the implemented step in the established prototype program are clarified. the established program was written using Visual Basic (ver.6) language. The main form of the program contains the following controls:

Fig. (3.1) the system model

3.3 System Implementation

In this section, the implemented step in the established prototype program are clarified. the established program was written using Visual Basic (ver.6) language. The main form of the program contains the following controls:
Menu Items:

1. Files (Load & Save)
   b. Save Bitmap: save the process Image in to chosen file.

2. Coding (Compress & Decompress).
   a. Compress: Compress original image file.
   b. Decompress: Decompress the compress image file.

3.3.1 Load BMP Image

The input to this system is a BMP image file; in the established system, the BMP image file was used as an input to the system. The considered color resolution of the images is 24 or 8 bit/pixel. The image data is loaded and used to fill-up the Red, Green, and Blue arrays, each array is considered as one primary color. Code list (3.1) presents the steps of the bitmap loading file.
**Code List (3.1): Load the Image file**

**Input:**

ImgFileName // The name of image file

**Output:**

Wid, Hgt // image width and height

Red(0 to Wid-1, 0 to Hgt-1) // Red component of image

Green(0 to Wid-1, 0 to Hgt-1) // Green component of image

Blue(0 to Wid-1, 0 to Hgt-1) // Blue component of image

**Begin**

**Step 1:** Get from ImgFileName the BMPH // BMPH is the BMP Header

Get image's width and height values from its header

Set Wid ← BMPH.Wid

Set Hgt ← BMPH.Hgt

**Step 2:** Check image pixel resolution

If BMPH.BitPlan ← 24 Then

Set DataSize ← BMPH.FileSize - BMPHSize

Get ImgFileName, Img(DataSize-1) // Img contains the image's data

Set I ← 0

For all X, Y Do {where 0 <= X <= Wid-1, 0 <= Y <= Hgt-1}

Set Red(X, Y) ← Img(I)

Set Green(X, Y) ← Img(I+1)

Set Blue(X, Y) ← Img(I+2)

Increment I by 3

End For

Else if BMPH.BitPlan = 8 Then

Set NoColor ← (BMPH.OffsetPosition -54) div 4

Get ImgFileName, RGBrecord(NoColor-1) // RGBrecord contains 4 cells: Red, Green, Blue, and A is a reserved byte for BMP images of 8 bit/pixel resolution

Continue
3.3.2 Image Data Compression

As a next stage the three stage of image data compression process are performed, these stages are:

1. Delta Coding.
2. Data Realignment from 2D to 1D array.
3. LZW Compression are applied.

Code list (3.2) presents the implemented steps to compress the red, green, and blue arrays of the loaded image data.

Code List (3.2): Compress the Image file

Input:

    ImgFileName // The name of image file

Output:

    One –Dimensional array (0 to Wid* Hgt-1) // component of Red, Green, blue
**Begin**

**Step1:** Get from ImgFileName the BMPH // BMPH is the BMP Header

Get image's width and height values from its header

Set \( Wm \leftarrow Wd - 1 \)
Set \( Hgt \leftarrow Hgt - 1 \)
Set \( L \leftarrow Wd \times Hgt - 1 \)
Loop \( I \leftarrow 1 \) To \( 3 \)
Set \( N \leftarrow -1 \)
Loop \( Y \leftarrow 0 \) To \( HM \)

**Step2:** Check \( IF(Y \ And \ 1) \leftarrow 0 \)

Set \( X \leftarrow 0 \)
Set \( Xs \leftarrow WM \)
Set \( STP \leftarrow 1 \)
End IF

Else

Set \( Xs \leftarrow 0 \)
Set \( STP \leftarrow -1 \)
End IF

**Step3:** Case 1 of

Case 1:

Loop \( X \leftarrow Xs \) To \( Xe \) Step \( STP \)
Increment \( N \) by 1
Store \( Red(X,Y)IN \) \( A(N) \)
End LoopX

Case 2:

Loop \( X \leftarrow Xs \) To \( Xe \) Step \( STP \)
Increment \( N \) by 1
Store \( Green(X,Y)IN \) \( A(N) \)
End LoopX

Case 3:

Loop \( X \leftarrow Xs \) To \( Xe \) Step \( STP \)
Increment \( N \) by 1
Store \( Blue(X,Y)IN \) \( A(N) \)
End LoopX

End Case
End LoopY
3.3.3 Image Data Decompression

In this module, the stages:

1. LZW decoding.
2. Data Rearrangement from 1D to 2D array.
3. Delta Decoding are applied to reconstruct the red, green and blue arrays of the reconstructed color image.

Code list (3.3) presents the taken steps to implement the decoding process.
**Code List (3.3): Decompress the Image file**

**Input:**

One –Dimensional array (0 to Wd* Hgt-1)// component of Red, Green, blue

**Output:**

ImgFileName// The name of image file

**Begin**

**Step1:** Get from ImgFileName theBMPH // BMPH is the BMP Header

Get image's width and height values from its header

Set Wm ← Wd-1
Set Hgt ← Hgt-1
Set L ← Wd*Hgt-1
Loop I ← 1 To 3
Set J j ← 1
Loop M ← 1 ToL
Set Aa ← 0
While SetB(Jj) ← 255
Set Aa ←Aa+255: Set Jj ←Jj+1
Wend
SetA(M) ←Aa+B(Jj): SetJf ←Jj+1
EndLoopM
LoopM ←1 ToL
**Step2:** Check IF(A(M) And 1) ← 0 Then SetA(M) ←A(M)\2

Else

SetA(M) ←((A(M)+1)/2)
SetA(M) ←A(M)+A(M-1)
EndLoopM

SetN ←1
LoopY ← 0 ToHM

Continue
**Step 3:** Check $IF(Y\text{ And } 1) \rightarrow 0$

Set $Xs \leftarrow 0$

Set $Xe \leftarrow WM$

$STP \leftarrow 1$

Else

Set $Xs \leftarrow WM$

Set $Xe \leftarrow 0$

$STP \leftarrow -1$

End IF

**Step 4:** Case 1 of

Case 1:

Loop $Xs \leftarrow Xe \text{ To } Xe \text{ Step } STP$

Increment $N$ by 1

Store $A(N)$ IN Red($X,Y$)

End Loop $X$

Case 2:

Loop $X \leftarrow Xs \text{ To } Xe \text{ Step } STP$

Increment $N$ by 1

Store Green $A(N)$ IN ($X,Y$)

End Loop $X$

Case 3:

Loop $X \leftarrow Xs \text{ To } Xe \text{ Step } STP$

Increment $N$ by 1

Store Blue $A(N)$ IN ($X,Y$)

End Loop $X$

End Case

End Loop $Y$

End Loop $I$

**Step 5:** End.
3.3.4 Save Bitmap file

As a last stage in the decoding process is saving the reconstructed image color bands as a bitmap file.

Code list (3.4) shows the steps of saving file.

```
**Code List (3.4): Save the Image file**

**Input:**
- One –Dimensional array (0 to Wid* Hgt-1)// component of Red, Green, blue

**Output:**
- ImgFileName// The name of image file

**Begin**

**Step1:** Get from ImgFileName theBMPH // BMPH is the BMP Header
  - Get image's width and height values from its header
  - Set Wm ← Wid-1
  - Set Hgt ← Hgt-1
  - SetLoopY ← HM To0 Step-1
  - SetXx ← 0
  - SetLoopX ← 0ToWM
  - SetA(Xx) ← Blue(X,Y)
  - SetA(Xx+1) ← Green(X,Y)
  - SetA(Xx+2) ← Red(X,Y)
  - IncrementXx by3
  - EndLoopX
  - EndLoopY

**Step2:** End
```
Chapter Three                                               The Proposed System Implementation

Fig. (3.2) view the bitmap color image

After the selection of bitmap color image file, then the program will load the bitmap color image file, then present this image. As a last step the user can apply the compression method by making a click on the menu option “Coding”, then select compress, the program will apply the compress color image and present the winrar archive file.

Fig. (3.3) winrar archive to bitmap color image
3.5 Performance Test Results

This section is devoted to present the results of the conducted tests to study the compression performance of the suggested image compression schemes. Some of the fidelity measures (i.e. CR, BR) have been used to assess the quality of the reconstructed image.

3.6 Image Test Material

Six bitmap images have been taken as test samples, each image consist of the same number of pixels (i.e., 352×288), and they have color resolution (24bpp), and size (297KB). Figure (3.4) shows these two images.

a. Lena  
b. Shape final  
c. two bird  
d. town
Fig. (3.4) the bitmap images used as test samples

Table (3.1) The result of compression ratio file on images.

<table>
<thead>
<tr>
<th>Image Name</th>
<th>Original Size</th>
<th>Original File Size (bytes)</th>
<th>Image Compressed</th>
<th>Bit Rate</th>
<th>Compression Ratio</th>
<th>Compression File Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>297 KB</td>
<td>304182</td>
<td>178 KB</td>
<td>2.148</td>
<td>1.396</td>
<td>217781</td>
</tr>
<tr>
<td>Final Shap</td>
<td>297 KB</td>
<td>304182</td>
<td>189 KB</td>
<td>2.218</td>
<td>1.352</td>
<td>224901</td>
</tr>
<tr>
<td>Two bird</td>
<td>269 KB</td>
<td>275670</td>
<td>212KB</td>
<td>2.366</td>
<td>1.268</td>
<td>217377</td>
</tr>
<tr>
<td>Big town</td>
<td>272KB</td>
<td>278838</td>
<td>204KB</td>
<td>2.257</td>
<td>1.329</td>
<td>209749</td>
</tr>
<tr>
<td>Baghdad city</td>
<td>254KB</td>
<td>260982</td>
<td>212KB</td>
<td>2.138</td>
<td>1.403</td>
<td>185967</td>
</tr>
<tr>
<td>Big city</td>
<td>297 KB</td>
<td>304182</td>
<td>204KB</td>
<td>2.245</td>
<td>1.336</td>
<td>227594</td>
</tr>
</tbody>
</table>
Conclusions and Future Work Suggestion
Conclusions and Future Work

4.1 Conclusions

From the test results presented in previous chapter, some remarks related to the behavior and performance of the predictive and LZW schemes are stimulated. Among these remarks are the followings:

1. The attained compression becomes better when the system is applied on paint images.
2. Due to the use of predictive coding step before LZW coding, the required encoding time is increased by %10 relative to time required to apply LZW alone on the same image file.
3. The difference between the time required for compression process and the time required for decompression process is insignificant. So, the proposed encoding scheme could be considered time symmetric.

3.2 Suggestions for Future Work

1. The Compression method could be extended to cover the video input (MPEG).
2. Transformation coding, as wavelet based, could be applied.
3. Applying some standard coding (like, arithmetic coding, and progressive coding instead of LZW.
4. In case of applying quantization e output of predictive coding(i.e., before applying LZW), the compression scheme will become lossy and more compression ratio could be attained.
References
References


[Hor00] Horan, S, " Data Compression Statistic and Implications " New Mexico State University, 2000.


References


