DCT Coefficients Compression Using Embedded Zerotree Algorithm

A Thesis Submitted to the Council of The Science College of Babylon University in Partial Fulfillment of the Requirements

of the Degree of M.SC. Science

In Computer Science

By

Asa'ad N. Hashim Al-Sherifi

May – ٢٠٠٦ **Rebi el Thani-١٤٢٧**

ضغط معاملات DCT بأستخدام خوارزمية Embedded Zerotree

رســـالة

مقدمة الى مجلس كلية العلوم - جامعة بابل

وهي جزء من متطلبات نيل درجة ماجستير في علوم الحاسبات

من

أسعد نوري هاشم الشريفي

آيار- ٢٠٠٦   ربيع الثاني-١٤٢٧

بِسمِ اللهِ الرَحّمنِ الرحّيم

﴿وَقـُلِ اْعمَلـوُا فسَـيَرىَ اللهُ عَمَلــكُمْ وَرَسُــوُلهُ وَالمُؤْمِنُــونَ
وَسَـتُرَدُّونَ إلــى عَـالِم اْلغيْـبِ وَالشَّـهَادَةِ فَيُنَبــِّئُكُمْ بـِمَا كُنـتُمْ
تَعْمَلـُون﴾

صدَقَ اللهُ العَلِيُ العَظِيمْ

سُوَرْةُ التوْبَةْ (١٠٥)

Acknowledgment

I am indebted to **MY GOD** for helping to present this work.

I would like to express my deep gratitude and appreciation to my supervisor **Dr. Tawfiq A. Al-Asadi** for the advice, support and encouragement given during the course of this work. Their suggestions, guidance and moral support in difficult times have been essential for completing this work.

My deep appreciation goes to the Department of Computer Science, **head** of the department and the **members** of the staff.

Special thankes go to the **staff** of Engineering College in Kufa University for their continuous support and encouragement during the period of my study

I also wish to thank my **friends** for their cooperation . And for **all the persons** who helped me to finsh this research.

Supervisor Certification

I certify that this thesis was prepared under my supervision at the department of computer science/college of science / Babylon University, by **Azher R. H. Witwit** as partial fulfillment of the requirements for the degree of Master of Science in computer science.

Signature :
Name : **Dr. Tawfiq A. Abbas Al-Asadi**
Title : **Assistant Professor**
Date : / / ٢٠٠٦

In view of the available recommendation , I forward this theses for debate by the examination committee.

Signature :
Name :
Title : **Head of Department of computer science.**
Date : / / ٢٠٠٦.

Certification of the Examination Committee

We chairman and members of the examination committee, certify that we have studied the thesis entitled **(A HYBRID ALGORITHM FOR DATA COMPRESSION USING HUFFMAN & GENETIC ALGORITHM)** presented by the student **Azher R.H. Witwit** and examined him in its contents and in what is related to it, and we have found it worthy to be accepted for the degree of Master of Science in computer science with degree.

Signature :
Name :
Title :
Date : / / ٢٠٠٦.

(chairman)

Signature :
Name :
Title :
Date : / / ٢٠٠٦.

(member)

Signature :
Name :
Title :
Date : / / ٢٠٠٦.
(member)

Signature :
Name :
Title :
Date : / / ٢٠٠٦.
(supervisor)

Signature :
Name :
Title :
Date : / / ٢٠٠٦.
(Dean of college of science-Babylon university)

Certification of the Examination Committee

We chairman and members of the examination committee, certify that we have studied the thesis entitled (**DCT Coefficients Compression Using Embedded Zerotree Algorithm**) presented by the student **Asa'ad N. Hashim** and examined him in its content and in what is related to it, and we have found it worthy to accepted for the degree of **Master of Science** with (**Very Good**) degree.

Signature:

Name : **Jassim T. Sarsouh**

Title : **Professor Assistant**

Date : **/ /٢٠٠٦**

(Chairman)

Signature:

Name : **Dr. Abbas M. Albakry**

Title : **Professor Assistant**

Date : **/ /٢٠٠٦**

(Member)

Signature:

Name : **Israa H. Ali**

Title : **Professor Assistant**

Date : **/ /٢٠٠٦**

(Member)

Signature:

Name : **Dr. Tawfiq A. Abbas**

Title : **Professor Assistant**

Date : **/ /٢٠٠٦**

(Supervisor)

Signature:

Name : **Dr. Oda Mizi'l Yasser Alzamely**

Title : **Professor**

Date : **/ /٢٠٠٦**

(Dean of College of Science – Babylon University)

Supervisor Certification

I certify that this thesis was prepared under my supervision at the Department of Computer Science/College of Science/Babylon University, by **Asa'ad N. Hashim** as partial fulfillment of requirements for the degree of **Master of Science in Computer Science**.

Signature:

Name : **Dr. Tawfiq A. Abbas**

Title : **Professor Assistant**

Date : / /٢٠٠٦

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

Signature:

Name : **Dr. Abbas M. Albakry**

Title : **Head of Computer Science Department**

Date : / /٢٠٠٦

LIST OF ABBRIVIATION

VQ	Vector Quantization
BPP	Bits Per Pixel
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DWT	Discrete Wavelet Transform
EZW	Embedded Zerotree Wavelet
HDCT	Hierarchical Discrete Cosine Transform
HDTV	High Definition Television
JPEQ	Joint Photographic Experts Group
MSE	Mean Square Error
PSNR	Peak Signal to Noise Ratio
RGB	Red, Green, Blue
RLE	Run Length Encoding
SAQ	Successive Approximation Quantization
SNR	Signal to Noise Ratio

SPIHT	Set Partitioning In Hierarchical Trees
WFA	Weighted Finite Automation

LIST OF CONTENTS

Subject	Page No.
Chapter One: General Introduction	
١.١ Introduction	١
١.٢ Compression System Model	٣
١.٣ Image Compression	٥
١.٣.١ Distortion Measures	٥
١.٣.٢ Transform	٦
١.٣.٣ Compression Measures	٧
١.٣.٤ Entropy Coding	٧
١.٣.٥ The Histogram	٨
١.٣.٦ Different Classes Compression Techniques	٨
١.٤ Color Spaces	٩
١.٥ Literature Review	١١
١.٦ Aim of Thesis	١٥
١.٧ Thesis Layout	١٥
Chapter Two: Image Compression Techniques	
٢.١ Introduction	١٦
٢.٢ Lossless Coding Techniques	١٦
٢.٢.١ Run Length Encoding(RLE)	١٧
٢.٢.٢ Quadtree Compression	١٧
٢.٢.٣ Weighted Finite Automation(WFA)	١٩
٢.٢.٤ Huffman Coding	٢١
٢.٢.٥ Arithmetic Coding	٢٣
٢.٣ Lossy Coding Techniques	٢٧
٢.٣.١ Vectors Quantization	٢٨
٢.٣.٢ Predictive Coding	٢٩
٢.٣.٣ Transform Based Image Compression	٢٩
Chapter Three: DCT and EZW	
٣.١ Introduction	٣٣

٣.٢ Discrete Cosine Transform(DCT)	٣٣
٣.٢.١ Two-Dimensional DCT Equations	٣٣
٣.٢.٢ Doing The DCT on an ٨*٨ Block	٣٤
٣.٢.٣ Quantization	٣٦
٣.٢.٤ Encoding	٣٧
٣.٢.٥ Decompression	٣٨
٣.٣ Embedded Zerotree Wavelet(EZW)	٣٩
٣.٣.١ Introduction	٣٩
٣.٣.٢ The Zerotree Structure	٤٠
٣.٣.٣ The Concepts of EZW	٤٣
٣.٣.٤ EZW Algorithm	٤٥
٣.٣.٥ EZW an Example	٤٦
٣.٣.٦ Decoding The Bitstream Generated by EZW	٥٤
Chapter Four: The Proposed Compression System	
٤.١ Introduction	٥٦
٤.٢ The Proposed Compression Algorithm Steps	٥٦
٤.٣ The Proposed Decompression Algorithm Steps	٦٦
٤.٤ Characters A Proposed Compression System over EZW	٧١
Chapter Five: Results, Conclusions and Future Works	
٥.١ The Experimental Results	٧٣
٥.١.١ Results of The Proposed Compression System	٧٣
٥.١.٢ Comparison The of The Proposed System Results with other Systems	٧٧
٥.٢ Conclusions	٧٧
٥.٣ Methods form Incrementing Compression Ratio for proposed Algorithm	٧٧
٥.٤ Future Works	٧٨
References	٧٩

LIST OF FIGURES

Fig. No.	Description	Page No.
١.١	Compression System Model	٤
١.٢	The Compressor	٥
١.٣	The Decompressor	٥

١.٤	The RGB Space	١٠
١.٥	Example of yuv Space	١٠
٢.١	The Tree Construed Using The Morton Order	١٨
٢.٢	Quadrant Numbering	٢٠
٢.٣	Huffman Code	٢٢
٢.٤	Diagram Showing The Limits of an Arithmetic coder	٢٤
٣.١	Zig-Zag Sequence of Binary Encoder	٣٨
٣.٢	The Relations Between Wavelet Coefficients in Different Subbands as Quad-Tree	٤١
٣.٣	An Example Data together with Two Scan Orders	٤٢
٣.٤	Typical Flows of Data of Image Encoder	٤٣
٣.٥	An ٨*٨ sample Image	٤٧
٣.٦	Step١ of Dominant Pass	٤٨
٣.٧	Step٢ of Dominant Pass	٤٨
٣.٨	Step٣ of Dominant Pass	٤٩
٣.٩	Step٤ of Dominant Pass	٤٩
٣.١٠	Step٥ of Dominant Pass	٤٩
٣.١١	Step٦ of Dominant Pass	٥٠
٣.١٢	Step٧ of Dominant Pass	٥٠
٣.١٣	Step٨ of Dominant Pass	٥١
٣.١٤	Step٩ of Dominant Pass	٥١
٤.١	Basic Components of The proposed Compression System	٥٦
٤.٢	Block Diagram for Basic Stages of The proposed Algorithm	٥٧
٤.٣	Flowchart of Embedded Zerotree Algorithm	٦١
٤.٤	An Example Illustrate Special Case	٦٢
٤.٥	Flowchart of Dominant Pass	٦٤
٤.٦	Flowchart of Subordinate Pass	٦٥
٤.٧	Flowchart of Decompression Algorithm	٦٧
٤.٨	Flowchart of Inverse Embedded Zerotree Algorithm	٧٠
٥.١	Original, Reconstructed of The Test Images	٧٤

٥.٢	Original, Reconstructed Images of Comparison between The Proposed System Results and RLE Results	٧٤

LIST OF TABLES

Table	Title	Page
٢.١	Showing Probability of Four Symbols	٢٤
٢.٢	Given Portion of ٠-١ Range for Each symbol	٢٥
٢.٣	Given Results for Applying Arithmetic Encoding	٢٦
٢.٤	Given Results for Applying Arithmetic Decoding	٢٧
٣.١	Quantization Table with Quality Level of ٥٠	٣٦
٣.٢	First Dominant Pass	٥٢
٣.٣	First Subordinate Pass	٥٢
٥.١	Results of The Proposed Compression System	٧٥
٥.٢	Comparison Proposed Compression System with RLE Algorithm	٧٥

Subject	Page No.
Chapter One: General Introduction	
١.١ Introduction	١
١.٢ Compression System Model	٣
١.٣ Image Compression	٥

١.٣.١ Distortion Measures	٥
١.٣.٢ Transform	٦
١.٣.٣ Compression Measures	٧
١.٣.٤ Entropy Coding	٧
١.٣.٥ The Histogram	٨
١.٣.٦ Different Classes Compression Techniques	٨
١.٤ Color Spaces	٩
١.٥ Literature Review	١١
١.٦ Aim of Thesis	١٥
١.٧ Thesis Layout	١٥
Chapter Two: Image Compression Techniques	
٢.١ Introduction	١٦
٢.٢ Lossless Coding Techniques	١٦
٢.٢.١ Ren Length Encoding(RLE)	١٧
٢.٢.٢ Quadtree Compression	١٧
٢.٢.٣ Weighted Finite Automation(WFA)	١٩
٢.٢.٤ Huffman Coding	٢١
٢.٢.٥ Arithmetic Coding	٢٣
٢.٣ Lossy Coding Techniques	٢٧
٢.٤.١ Vecter Quantization	٢٨
٢.٤.٢ Predictive Coding	٢٩
٢.٤.٣ Transform Based Image Compression	٢٩
Chapter Three: DCT and EZW	
٣.١ Introduction	٣٣
٣.٢ Discrete Cosine Transform(DCT)	٣٣
٣.٢.١ Two-Dimensional DCT Equations	٣٣
٣.٢.٢ Doing The DCT on an ٨*٨ Block	٣٤
٣.٢.٣ Quantization	٣٦
٣.٢.٤ Encoding	٣٧
٣.٢.٥ Decompression	٣٨
٣.٣ Embedded Zerotree Wavelet(EZW)	٣٩
٣.٣.١ Introduction	٣٩
٣.٣.٢ The Zerotree Structure	٤٠
٣.٣.٣ The Concepts of EZW	٤٣
٣.٣.٤ EZW Algorithm	٤٥
٣.٣.٥ EZW an Example	٤٦
٣.٣.٦ Decoding The Bitstream Generated by EZW	٥٤
Chapter Four: The Proposed Compression System	
٤.١ Introduction	٥٦
٤.٢ The Proposed Compression Algorithm Steps	٥٦
٤.٣ The Proposed Decompression Algorithm Steps	٦٦

٤.٤ Characters A Proposed Compression System over EZW	٧١
Chapter Five: Results, Conclusions and Future Works	
٥.١ The Experimental Results	٧٣
٥.١.١ Results of Proposed Compression System	٧٣
٥.١.٢ Comparison Proposed System Results with other Systems	٧٧
٥.٢ Conclusions	٧٧
٥.٣ Methods form Incrementing Compression Ratio for proposed Algorithm	٧٧
٥.٤ Future Works	٧٨

Abstract

The goal of compression algorithms is to gain best compression ratio with acceptable visual quality, the proposed compression system presented a new approach to satisfy this goal , the results depend on many parameters such as the details of image and size of image.

Embedded Zerotree Algorithm works efficiently because of the following hypothesis:- " **If a DCT coefficient C at a coarse scale is insignificant with respect to a given threshold T, i.e. |C|<T then all DCT coefficients of the same orientation at finer scales are also likely to be insignificant with respect to T"**.

The system consist of four main steps which are :

Color space in which the data of image are converted into another mathematical space is called RGB to simplify the following procedures and the RGB considered as the standard color space which used in the displays, then this representation will converted into another color space is called YUV which consist of two basic components Y represent the luminance and UV represent the chrominance. In the second step for each component coming from color space, the suggested system divide the data into a number of blocks to

avoided the artifacts resulted from apply DCT and quantization. The coefficients are coded using embedded zerotree algorithm in the third steps. Finally the sequence of symbols will be encoded by Huffman algorithm.

The Proposed compression system is lossy since there are many steps oblige the image data to be lossy. The system achieve competitive compression ratio with a good PSNR. The primary purpose of this system is not only to compress image, but also highlight many relation subjects such as transform and compression methods.

الخلاصة:

الهدف من خوارزميات الضغط هو الحصول على أفضل نسبة ضغط مع مواصفات مظهرية مقبولة, النظام المقترح قدم اتجاهاً جديداً لتحقيق هذا الهدف, نتائج الضغط تعتمد على عدة عوامل مثل تفاصيل ألصورة و حجمها . خوارزمية Embedded Zerotree تعمل بكفاءة بسبب الافتراض التالي:

" لو افترضنا إن T عتبة معطاة, و لو افترضنا إن C هي القيمة المطلقة لمعامل DCT, وليكن C في البداية غير مهم بالمقارنة مع العتبة T. فإذا كان |C|<T فان جميع المعاملات التي بنفس الاتجاه إلى النهاية تكون أيضا غير مهمة بالمقارنة مع العتبة T"

النظام المقترح يتألف من أربعة خطوات أساسية وهي:

الفضاء اللوني و فيه يتم نقل بيانات الصورة إلى فضاء رياضي جديد هو RGB لتبسيط الإجراءات التالية ثم يحول هذا التمثيل أيضا إلى فضاء لوني آخر يسمى YUV والذي ينتج مكونين رئيسين هما Y والذي يمثل تباين الصورة luminance و UV و الذي يمثل القيم اللونية Chrominance , في الخطوة التالية سيقوم النظام المقترح بتقسيم البيانات لكل مكون قادم من الفضاء اللوني لتقليل التأثيرات الناتجة من تطبيق DCT و المعاملات سوف تشفر باستخدام خوارزمية Embedded Zerotree في الخطوة الثالثة. أخيرا السلسلة المؤلفة من الرموز سوف تشفر باستخدام خوارزمية Huffman .

النظام المقترح من ألانظمة ألفاقدة لجزء من بيانات ألصوره لأنه توجد عدد من الخطوات التي تجبر ألصورة على فقد جزء من بياناتها. النظام المقترح ينجز نسبة ضغط منافسةالانظمة الأخرى مع نسبة خطأ قليلة والغرض الأساسي من النظام ليس فقط ضغط ألصورة بل أيضا التركيز على بعض المواضيع المتعلقة مثل التحويلات و طرق الضغط.

DEDICATION

To

My Mother,

My Father,

My Family,

My Friends and

My Department's Staff

٢.١ Introduction

One of the important aspects of image storage is its efficient compression, in a distributed environment large image files remain a major bottleneck within systems. Compression is an important component of the solutions available for creating file sizes of manageable and transmittable dimensions. The aim of this chapter is to introduce the techniques used in image compression. Although some of these techniques are well known, it is important to consider how they are used in image compression.

Two categories of data compression algorithm can be distinguished**:** **lossless** and **lossy.** Lossy techniques cause image quality degradation in each compression/decompression step. Careful consideration of the human visual perception ensures that the degradation is often unrecognizable, though this depends on the selected compression ratio. In general, lossy techniques provide far greater compression ratios than lossless techniques[١٥].

٢.٢ Lossless Coding Techniques

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

Some compression methods can be made more effective by adding a ١D or ٢D delta coding to the process of compression. These deltas make more effectively use of run length encoding, have (statistically) higher maxima in code tables (leading to better results in Huffman and general entropy coding), and build greater equal value areas usable for area coding[١٥].

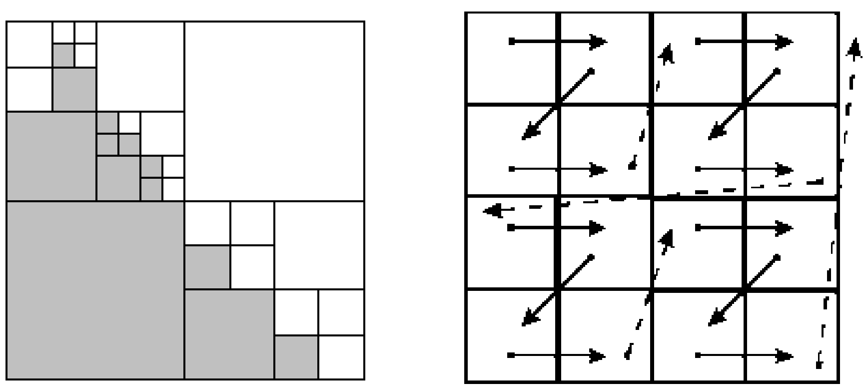
٢.٢.١ Run Length Encoding(RLE)

Run length encoding, sometimes called recurrence coding, is one of the simplest data compression algorithms. It is effective for data sets that are comprised of long sequences of a single repeated character. For instance, text files with large runs of spaces or tabs may compress well with this algorithm. RLE finds runs of repeated characters in the input stream and replaces them with a three-byte code. The code consists of a flag character, a count byte, and the repeated characters. For instance, the string "AAAAAABBBBCCCCC" could be more efficiently represented as "A٦B٤C٥". That saves us six bytes. Of course, since it does not make sense to represent runs less than three characters in length with a code, none is used. Thus "AAAAAABBCCCDDDD" might be represented as "A٦B٢C٣D٤". The flag byte is called a sentinel byte[٩].

٢.٢.٢ Quadtree Compression

This is another method with an application in bitmap graphics (particularly black and white). By dividing and sub-dividing quadrants within a picture, a quad-tree can be generated that identifies differences and similarities with a quadrant. Consider the following diagram a ١٦ x ١٦ pixel sample taken from a larger picture. The whole picture is divided into four quadrants. Each sub-quadrant is further divided into four quadrants, and so on until the smallest quadrant represents the smallest piece of data (dependent on requirements), as shown in Figure (٢.١).

A quadtree can now be constructed by starting from the top-left corner of the overall picture and using, for example, the Morton order described below. Each quadrant is represented in the quadtree as a node with four leaves, each representing one of the four sub-quadrants. That is, traverse the data-elements quadrant by quadrant storing the value of the smallest data elements as leaves and a 'combination of the leaf values' with their parent node, until every element has been include.



Example Morton Order

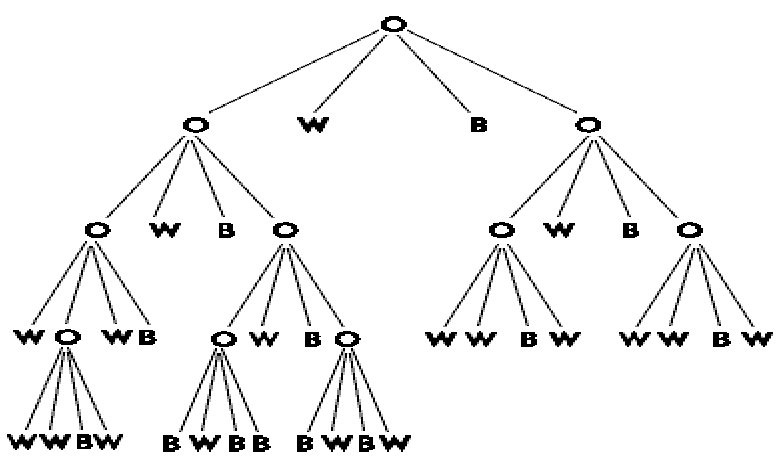


Figure (٢.١): Tree Constructed Using the Morton Order

Traversing the above quad-tree in Morton order gives the following representation of the above picture:

OOOWOWWBWWBWBOOBWBBWBOBWBWWBOOWWBWWBOW WBW.

Where 'O' represents a node, 'B' is the black area, and 'W' is the white area. As there are only three symbols used, each could be uniquely represented by two bits giving a total of ٨٢ bits [١٦].

٢.٢.٣ Weighted Finite Automata (WFA)

Finite-state automata(or finite-state machine) starts with bi-level (monochromatic) image and creates finite-state automation that completely describes the image. The automation is then written on the compressed stream and it becomes the compressed image. WFA is a lossless method but it is easy to add a lossy option where a user-controlled parameter indicates the amount of loss permitted .The method is based on two principles.

١. Any quadrant, subquadrant, and pixel in the image can be represented by string of the digits ٠, ١, ٢ and ٣. The longer the string, the smaller the image area is represented.

٢. Images used in practice have a certain amount of self-similarity, i.e., it is possible many times to find part of the image that looks the same as another a part, except for size, or is at least very similar to it. Sometimes part of an image has to be rotated or reflected, and this feature is also used by the method.

Assume that the quadrant numbering of Figure (٢.٢.a) is extended recursively to subquadrants. Figure (٢.٢.b) shows how each of the ١٦ subquadrants produced from the ٤ original ones are identified by a ٢-digit string of the digits ٠, ١, ٢, and ٣. After another subdivision, each of the resulting subquadrants is identified by a ٣-digit string, and so on.

١	٣
٠	٢

(a)

١١	١٣	٣١	٣٣
١٠	١٢	٣٠	٣٢
٠١	٠٣	٢١	٢٣
٠٠	٠٢	٢٠	٢٢

(b)

Figure (٢.٢): Quadrant Numbering

If the image size is $(٢^{n} * ٢^{n})$ then a single pixel is represented by a string of n digits, and a string of K digits represents a subquadrant of size $٢^{n-k} * ٢^{n-k}$ pixels. Once this is grasped, it is easy to see how an image can be represented by finite state automata. This is based on three rules:

١-Each state of the automata represents a part of the image. State ٠ is the entire image; other states represent quadrants or subquadrants of various sizes.

٢-Given a state i that represents a part of the image, it is divided into four quadrants. If, e.g., quadrant ٢ of i is identical to the image part represented by state j, then an arc is drawn from state i to state j, and is labeled ٢ (the label is the “weight” of the arc, hence, the name “Weighted Finite Automata” or (WFA). There is no need to worry about parts that are totally white. If quadrant ١ of state i, e.g., is completely white, there is no need to find an identical state and to have an arc with weight ١ Coming out if i. when the automata is used to reconstruct the image (i.e., when the compressed stream is decoded) any missing arcs are assumed to point to white subquadrants of the image[١٧].

٢.٢.٤ Huffman Coding

This is a commonly used method for data compression. It serves as the basis for several popular programs. Some of them use just the Huffman method while others use it as one step in a multi-step compression process. The Huffman method is somewhat similar to the Shanon-Fano method. 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.

Example:

Given five symbols with probabilities as shown in Figure (٢.٣.a), they are paired in the following order:

١-$a_{٤}$ is combined with $a_{٥}$ and both are replaced by the combined symbol $a_{٤٥}$ whose probability is ٠.٢.

٢-There are now four symbols left $a_{١}$, with probability ٠.٤, and $a_{٢}$, $a_{٣}$ and $a_{٤٥}$, with probabilities ٠.٢ each. We arbitrarily select $a_{٣}$ and $a_{٤٥}$, combine them and replace with auxiliary symbol $a_{٣٤٥}$ whose probability is ٠.٤.

٣-Three symbols are now left, $a_{١}$, $a_{٢}$ and $a_{٣٤٥}$, with probabilities ٠.٤, ٠.٢ and

٠.٤, respectively. We arbitrarily select $a_{٢}$ and $a_{٣٤٥}$, Combine them and replaced them with the auxiliary symbol $a_{٢٣٤٥}$ whose probability is ٠.٦. Finally, we combine the two remaining symbols, $a_{١}$ and $a_{٢٣٤٥}$, and replace them with $a_{١٢٣٤٥}$ with probability ١. The tree is now complete. To assign the codes, we arbitrarily assign a bit of ١ to the top edge, and a bit of ٠ to the bottom edge, of every pair of edges. This results in the codes ١, ١٠, ١٠٠, ١٠٠٠, and ٠٠٠٠. The assignments of bits to the edges are arbitrary but this must be consistent. The Huffman code is not unique. Some of the steps above were selected arbitrarily, since there were more than two symbols with smallest probabilities. Figure (٢.٣.b) shows how the same five symbols can be combined differently to obtain a different Huffman code.

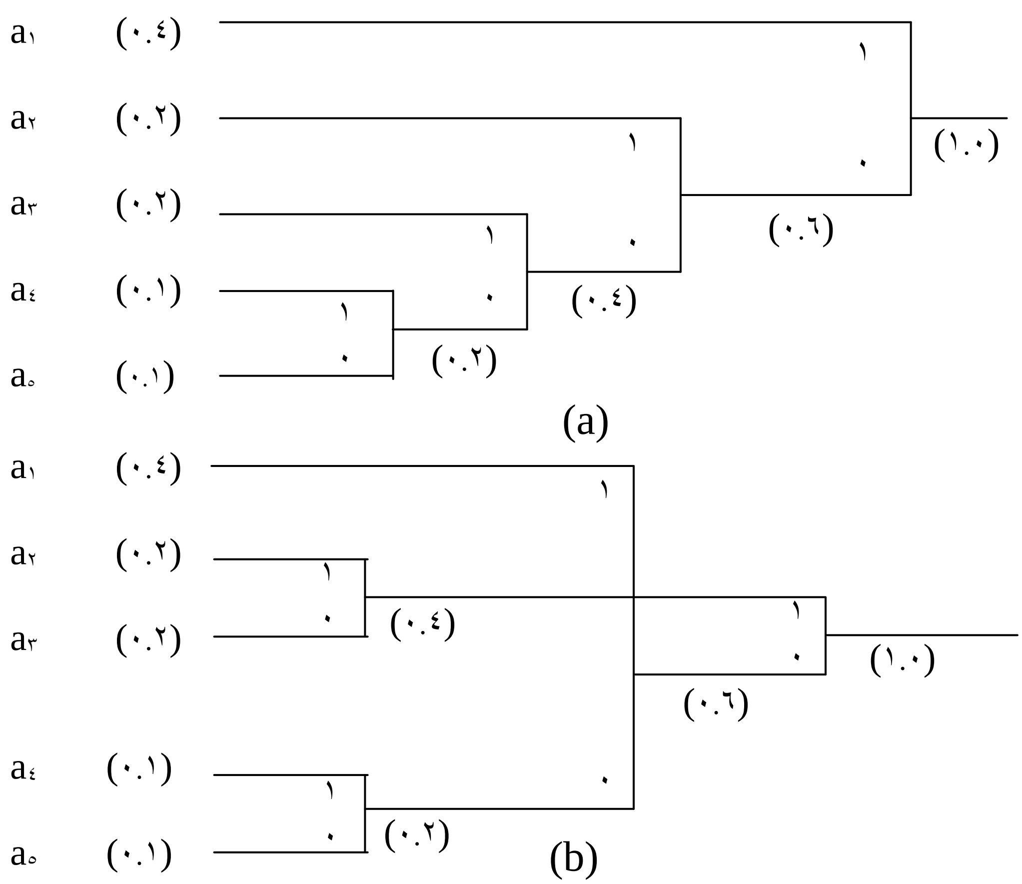


Figure (٢.٣): Huffman Code

Before starting the compression of a data stream, the compressor (encoder) has to determine the codes. It based on the probabilities (or frequencies of occurrence) of the symbols. The probabilities of frequencies have to appear on the compressed stream, so that any Huffman decompressor (decoder) will be able to decompress the stream. This is easy since the frequencies are integers and the probabilities can be written as scaled integers. It normally adds just a few hundred bytes to the compressed stream. The decoder must know what is at the start, read it, and construct the Huffman tree for the alphabet. Then it can be read and decode the rest of the stream. The algorithm of decoding is simple. Start at the root and read the first bit of the compressed stream. If it is zero follow the bottom edge, if it is one, follow the top edge. Read the next bit and move another edge toward the leaves of the tree.

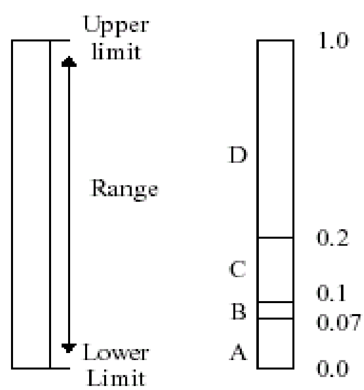
When the decoder gets to a leaf, it finds the original, uncompressed code of the symbol and that code is emitted by the decoder. The process starts again at the root with the next bit[٦].

٢.٢.٥ Arithmetic Coding

An arithmetic coder takes an upper and lower limit, and defines a *range* between these *upper* and *lower* limits to be equivalent to a symbol with the probability of ١.٠ . Symbols are encoded by modifying the *range* of the arithmetic coder and sending symbols to reconstruct this range information at the decoder. The operation of an arithmetic coder can be demonstrated by using the data in table (٢.١). The data can be represented as probabilities in the arithmetic coder as shown in figure (٢.٤.b) . It can be seen that the symbol probabilities stack to form a continuous range of probabilities between ٠.٠ and ١.٠. This give a range of probabilities that represent each symbol.

Table (٢.١): Table showing Probability of Four Symbols

Symbol	Probability
A	٠.٠٧
B	٠.٠٣
C	٠.١
D	٠.٨



(a) (b)

Figure (٢.٤): Block Diagrams Showing the Limits of an Arithmetic Coder

The *upper* limit of the arithmetic coder is initially set to a value which corresponds to a probability of ١.٠ while The *lower* limit is set to zero. The only problem with this assumption is that computers work with fixed length numbers, commonly ٤ to ٨ bytes long, so it is impossible to represent the infinite precision The operation of encoding a symbol by the coder requires the *range* to be reduced in the following way:

$$\text{Range} = \text{Upper Limit} - \text{Lower Limit} \qquad \text{...(٢.١)}$$

$$\text{New Upper Limit} = \text{Lower Limit} + \text{Range} * \text{P HIGH } (\textit{Smbol}) \qquad \text{...(٢.٢)}$$

$$\text{New lower Limit} = \text{Lower Limit} + \text{Range} * \text{P LOW } (\textit{Symbol}) \qquad \text{...(٢.٣)}$$

where $P_{HIGH}(Symbol)$ is the higher probability range of the symbol, and $P_{LOW}(Symbol)$ is the lower probability range of the symbol.

The method described above works, but it can reach a state where the coder is effectively ‘stuck’. The upper and lower limits can come so close together that the range can no longer be altered and no bits can be shifted out to increase the precision. This problem is known as underflow[٥] .

Example:

Encoding the random message "BILL GATES", giving probability distribution that looks like Table (٢.٢) .

Once the character probabilities are known, the individual symbols need to be assigned a range along a "probability line", which is normally ٠ to ١. It doesn't matter which characters are assigned which segment of the range, as long as it is done in the same manner by both the encoder and the decoder. The nine character symbol set use here would look like Table (٢.٢).

Table (٢.٢): Table Giving Portion of The ٠-١ Range for Each Character

Character	Probability	Range
SPACE	١/١٠	٠.٠٠ - ٠.١٠
A	١/١٠	٠.١٠ - ٠.٢٠
B	١/١٠	٠.٢٠ - ٠.٣٠
E	١/١٠	٠.٣٠ - ٠.٤٠
G	١/١٠	٠.٤٠ - ٠.٥٠
I	١/١٠	٠.٥٠ - ٠.٦٠
L	٢/١٠	٠.٦٠ - ٠.٨٠
S	١/١٠	٠.٨٠ - ٠.٩٠
T	١/١٠	٠.٩٠ - ١.٠٠

Each character is assigned the portion of the ٠-١ range that corresponds to its probability of appearance. Following the arithmetic algorithm through chosen message gives Table (٢.٣).

Table (٢.٣):Table Giving Results for Applying Arithmetic Encoding

New Character	Low value	High Value
	٠.٠	١.٠
B	٠.٢	٠.٣
I	٠.٢٥	٠.٢٦
L	٠.٢٥٦	٠.٢٥٨
L	٠.٢٥٧٢	٠.٢٥٧٦
SPACE	٠.٢٥٧٢٠	٠.٢٥٧٢٤
G	٠.٢٥٧٢١٦	٠.٢٥٧٢٢٠
A	٠.٢٥٧٢١٦٤	٠.٢٥٧٢١٦٨
T	٠.٢٥٧٢١٦٧٦	٠.٢٥٧٢١٦٨
E	٠.٢٥٧٢١٦٧٧٢	٠.٢٥٧٢١٦٧٧٦
S	٠.٢٥٧٢١٦٧٧٥٢	٠.٢٥٧٢١٦٧٧٥٦

So the final low value, ٠.٢٥٧٢١٦٧٧٥٢ will uniquely encode the message "BILL GATES" using our present encoding scheme. Given this encoding scheme, it is relatively easy to see how the decoding process will operate. We find the first symbol in the message by seeing which symbol owns the code space that our encoded message falls in. Since the number ٠.٢٥٧٢١٦٧٧٥٢ falls between ٠.٢ and ٠.٣, we know that the first character must be "B". We then need to remove the "B" from the encoded number. Since we know the low and high ranges of B, we can remove their effects by reversing the process that put them in. First, we subtract the low value of B from the number, giving ٠.٠٥٧٢١٦٧٧٥٢. Then we divide by the range of B, which is ٠.١. This gives a value of ٠.٥٧٢١٦٧٧٥٢. We can then calculate where that lands, which is in the range of the next letter, "I". The decoding algorithm for the "BILL GATES" message will proceed something like Table (٢.٤).

Table (٢.٤): Table Giving Results for Applying Arithmetic Decoding

Encoded Number	Output Symbol	Low	High	Range
٠.٢٥٧٢١٦٧٧٥٢	B	٠.٢	٠.٣	٠.١
٠.٥٧٢١٦٧٧٥٢	I	٠.٥	٠.٦	٠.١
٠.٧٢١٦٧٧٥٢	L	٠.٦	٠.٨	٠.٢
٠.٦٠٨٣٨٧٦	L	٠.٦	٠.٨	٠.٢
٠.٠٤١٩٣٨	SPACE	٠.٠	٠.١	٠.١
٠.٤١٩٣٨	G	٠.٤	٠.٥	٠.١
٠.١٩٣٨	A	٠.٢	٠.٣	٠.١
٠.٩٣٨	T	٠.٩	١.٠	٠.١
٠.٣٨	E	٠.٣	٠.٤	٠.١
٠.٨	S	٠.٨	٠.٩	
٠				

In summary, the encoding process is simply one of narrowing the range of possible numbers with every new symbol. The new range is proportional to the predefined probability attached to that symbol. Decoding is the inverse procedure, where the range is expanded in proportion to the probability of each symbol as it is extracted[١٨].

٢.٣ Lossy Coding Techniques

In most of applications we have no need in the exact restoration of stored image. This fact can help to make the storage more effective, and this way we get to lossy compression methods. Lossy image coding techniques normally have three components:

- *image modelling* which defines such things as the transformation to be applied to the image .
- *parameter quantisation* whereby the data generated by the transformation is quantised to reduce the amount of information .
- *encoding*, where a code is generated by associating appropriate codewords to the raw data produced by the quantiser.

Each of these operations is in some part responsible of the compression. Image modelling 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), Wavelet, and so on), where a reduced number of coefficients contains most of the original information. The aim of quantization is to reduce the amount of data used to represent the information within the new domain. Quantization is in most cases not a reversible operation; therefore, it belongs to the so called 'lossy' methods. Encoding is usually error free. It optimizes the representation of the information (helping, sometimes, to further reduce the bit rate), and may introduce some error detection codes[١٥].

٢.٣.١ Vector Quantization(VQ)

Vector quantization is a hybrid of statistical analysis and pattern recognition with a large number of well established techniques. This method first divides an image into blocks which are then serialized into large-dimensional vectors. Each vector is treated as a sample in a high-dimensional space and this space is partitioned into subspaces, called classes. For each class of vectors, a prototype is created that closely resembles the vectors which belong to the class. Vector quantization shifts the partition boundaries by calculating a distance measure. Errors between in-class vectors and the corresponding prototypes over all classes are minimized to provide optimum prototype placement. High levels of compression are achieved by coding every vector that belongs to a given class by a reference to the prototype vector. The level of lossiness is then dependent on the distance measure and the number of prototype vectors[١٩].

٢.٣.٢ 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. 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.

Linear predictive coding is a simple, special case of predictive coding in which the model simply take an average of the neighboring values[٩] .

٢.٣.٣ Transform Based Image Compression

The basic encoding method for transform based compression works as follows:

١- Image transform: Divide the source image into blocks and apply the transformations to the blocks.

٢- Parameter quantization: The data generated by the transformation are quantized to reduce the amount of information. This step represents the information within the new domain by reducing the amount of data.

Quantization is in most cases not a reversible operation because of its lossy property.

٣- Encoding : Encode the results of the quantization. This last step can be error free by using any lossless method such as Run Length Encoding or Huffman coding. It can also be lossy if it optimizes the representation of the information to further reduce the bit rate.

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.

DCT-Based Transform Coding: It is a popular transform used by the JPEG (Joint Photographic Experts Group) image compression standard for lossy compression of images. Since it is used so frequently, DCT is often referred to in the literature as JPEG-DCT, DCT used in JPEG. JPEG-DCT is a transform coding method comprising four steps. The source image is first partitioned into sub-blocks of size ٨x٨ pixels in dimension. Then each block is transformed from spatial domain to frequency domain using a ٢-D DCT basis function. The resulting frequency coefficients are quantized and finally output to a lossless entropy coder. DCT is an efficient image compression method since it can decorrelate pixels in the image (since the cosine basis is orthogonal) and compact most image energy to a few transformed coefficients. Moreover, DCT coefficients can be lossily quantized according to some human visual characteristics. Therefore, the JPEG image file format is very efficient. This makes it very popular, especially in the World Wide Web. However, JPEG may be replaced by wavelet-based image compression algorithms, which have better compression performance.

Wavelets Transform: Is a transform 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. The wavelet transform is suited for nonstationary signals, such as very brief signals and signals with interesting components at different scales. Wavelets are functions generated from one single function ̥́, which is called mother wavelet, by dilations and translations.

$$\psi_{a,b}(x) = |a|^{-1/2}\,\psi\left(\frac{x-b}{a}\right) \qquad \ldots(٢.٤)$$

where ψ must satisfy $\int \psi(x)dx = 0$

The basic idea of wavelet transform is to represent any arbitrary function *f* as a decomposition of the wavelet basis or write *f* as an integral over *a* and *b* of $\psi_{a,b}$

let $a = a_0^m, b = nb_0 a_0^m$, *with* $m, n, \in$ int*egers*, *and* $a_0 > 1, b_0 > 0$ *fixed* then the wavelet decomposition is

$$f = \sum c_{m,n}(f)\psi_{m,n} \qquad \ldots(٢.٥)$$

let $a_0 = 2, b_0 = 1$, we have an orthonomal basis, so that

$$c_{m,n}(f) = < \psi_{m,n}, f > = \int \psi_{m,n}(x) f(x) dx \qquad \ldots(٢.٦)$$

In image compression, we are dealing with sampled data that are discrete in time. We would like to have discrete representation of time and frequency, which is called the discrete wavelet transform (DWT). Before we start the DWT, we need to study another concept, multiresolution analysis[٩].

١.١ Introduction

Image compression is very important in many application, especially for progressive transmission, image browsing, Internet and multimedia applications. The goal of compression is to obtain an image representation while reducing as much as possible the amount of memory needed to encode the image[١].

Image compression is possible because images in general, are highly coherent(nonrandom),which means that there is redundant information. Visual data like other meaningful data, are usually structured, and this structure means that data over different parts of an image are interrelated. For example, consider an image in matrix format, if we take an arbitrary pixel, its color will likely be close to that of neighboring pixels, since they are more likely than not to belong to the same object. In any case, there are usually some redundant data because of the image structure. Image compression methods try to eliminate some of this redundancy to produce a more compact code that preserves the essential information contained in the image[٢].

The task of compression consists of two components, an *encoding* algorithm that takes a file and generates a “compressed” representation (hopefully with fewer bits), and a *decoding* algorithm that reconstructs the original file or some approximation of it from the compressed representation. These two components are typically intricately tied together since they both have to understand the shared compressed representation. We distinguish between *lossless algorithms*, which can reconstruct the original file exactly from the compressed file, and *lossy algorithms*, which can only reconstruct an approximation of the original file. Lossless algorithms are typically used for text, and lossy for images and sound where a little bit of loss in resolution is often undetectable, or at least acceptable. Lossy is used in an abstract sense, however, and does not mean random lost pixels, but instead means loss of a quantity such as a frequency component, or perhaps loss of noise. For example, one might think that lossy text compression would be unacceptable because they are imagining missing or switched characters. Consider instead a system that reworded sentences into a more standard form, or replaced words with synonyms so that the file can be better compressed. Technically the compression would be lossy since the text has changed, but the "meaning" and clarity of the message might be fully maintained, or even improved.

When discussing compression algorithms it is important to make a distinction between two components: the model and the coder. The *model* component somehow captures the probability distribution of the files by knowing or discovering something about the structure of the input. The *coder* component then takes advantage of the probability biases generated in the model to generate codes. It does this by effectively lengthening low probability files and shortening high-probability files. A model, for example, might have a generic "understanding" of human faces knowing that some "faces" are more likely than others. The coder would then be able to send shorter messages for objects that look like faces.

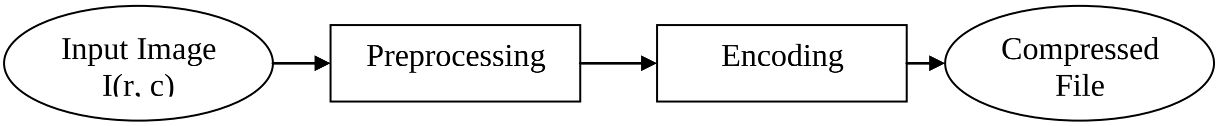
Main question about compression algorithms is how does one judge the quality of one versus another. In the case of lossless compression there are several criteria such that the time to compress, the time to reconstruct, the size of the compressed files, in the case of lossy compression the judgment is further complicated since we also have to worry about how good the lossy approximation is. There are typically tradeoffs between the amount of compression, the runtime, and the quality of the reconstruction. Depending on your application one might be more important than another and one would want to pick your algorithm appropriately[٣].

١.٢ Compression System Model

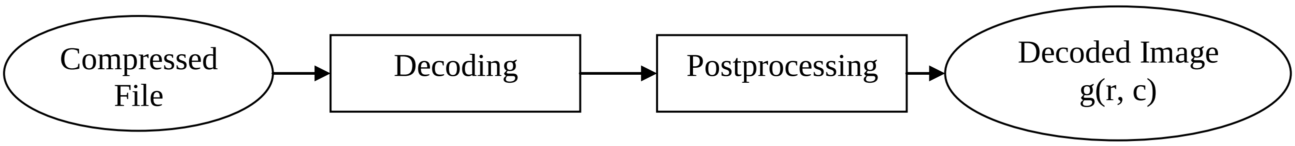
The compression system model consists of two parts: the compressor and the decompressor. The compressor consists of a preprocessing stage and encoding stage, whereas the decompressor consists of a decoding stage followed by a postprocessing stage Figure(١.١). Before encoding, preprocessing is performed to prepare the image for the encoding process, and consists of a number of operations that are application specific. After the compressed file has been decoded, postprocessing can be performed to eliminate some of the potentially undesirable artifacts brought about by the compression process. Often, many practical compression algorithms are a combination of a number of different individual compression techniques.

The compressor can be further broken down into two stages as illustrated in Figure (١.٢). The first stage in preprocessing is data reduction. Here, the image data can be reduced by gray-level and/or spatial quantization, or they can undergo any desired image enhancement (for example, noise removal) process. The second step in preprocessing is the mapping process, which maps the original image data into another mathematical space where it is easier to compress the data. Next, as part of the encoding process, is the quantization stage, which takes the potentially continuous data from the mapping stage and puts it in discrete form. The final stage of encoding involves coding the resulting data, which maps the discrete data from the quantizer onto a code in an optimal manner. A compression algorithm may consist of all the stages, or it may consist of only one or two of the stages. The decomposer can be further broken down into the stages shown in Figure (١.٣). Here the decoding process is divided into two stages. The first, the decoding stage, takes the compressed file and reverses the original coding by mapping the codes to the original, quantized values. Next, these values are processed by a stage that performs an inverse mapping to reverse the original mapping process. Finally, the image may be processes to enhance the final image.

In some cases this may be done to reverse any preprocessing, for example, enlarging an image that was shrunk in the data reduction process. In other cases the enhancement may simply enhance the image to ameliorate any artifacts from the compression process itself [٤].



(a) Compression



(b) Decompression

Figure (١.١): Compression System Model

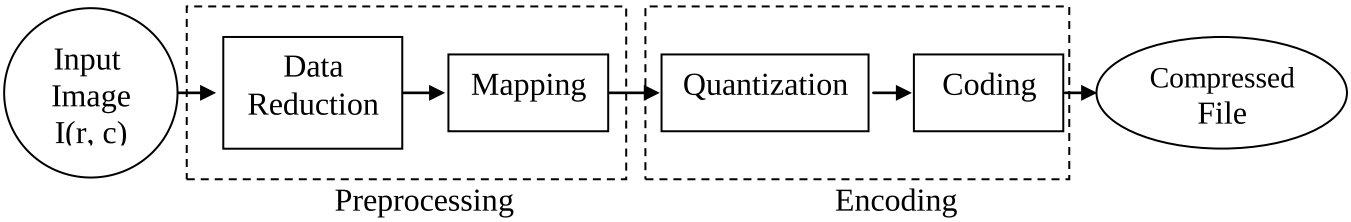


Figure (١.٢): The Compressor

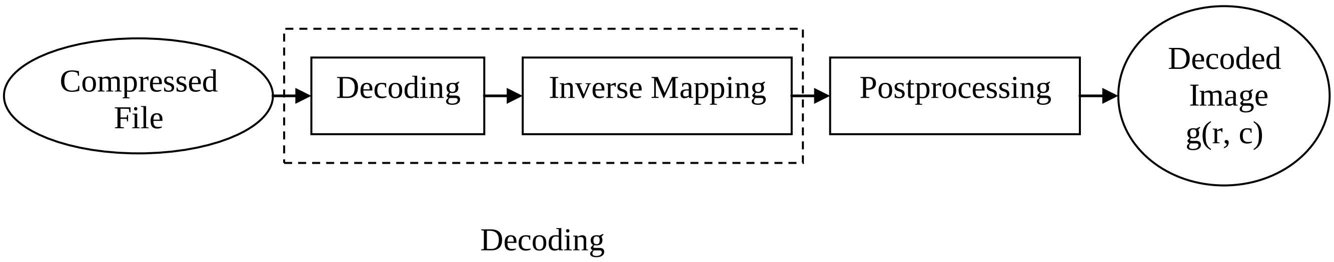


Figure (١.٣): The Decomposor

١.٣ Image Compression

Before images can be compressed effectively it is important to understand their properties. In this section the types of process that are applied to still images will be explained and examined. There are a number of processes that are always applied to an image before still image compression occurs. There are other processes that are used as part of the image compression.

١.٣.١ Distortion Measures

The distortion or error caused in the recovered image by the image compression process can be measured in several ways. The standard distortions measures are Root Mean Square Error (RMSE), Peak Signal to Noise Ratio (PSNR) and Signal to Noise Ratio (SNR). These

They work by comparing the squared error (power) between the original and the recovered digital images:

$$RMSE = \sqrt{\frac{1}{N^2}\sum_{r=0}^{N-1}\sum_{c=0}^{N-1}[g(r,c) - I(r,c)]^2} \qquad \ldots(١.١)$$

$$\text{SNR} = \sqrt{\frac{\sum_{r=0}^{N-1}\sum_{c=0}^{N-1}[g(r,c)]^2}{\sum_{r=0}^{N-1}\sum_{c=0}^{N-1}[g(r,c) - I(r,c)]^2}} \qquad \ldots(١.٢)$$

$$\text{PSNR} = 10\log_{10}\frac{(L-1)^2}{\frac{1}{N^2}\sum_{r=0}^{N-1}\sum_{c=0}^{N-1}[g(r,c) - I(r,c)]^2} \qquad \ldots(١.٣)$$

where L is the number of gray levels(e.g., for ٨ bits L=٢٥٦).

I(r,c) : the original image, g(r,c) : the decompressed image

r,c : row and column

None of the main methods for measuring image distortion takes into account, how good the recovered image looks to the human visual system. This is an area called psyco-visual image analysis, and it is an area of research which has a large scope. Unfortunately little progress has been made into an automated method for calculating a psyco-visual distortion measure[٥].

١.٣.٢ Transform

The transform is the defining part of an image compressor and image compressors are usually placed into broad groups based on which transform they use. The image transform should de-correlate the image, so that the image data is in a more compact form, in the new transform domain. Transforms generally come in pairs of forward and inverse transforms. If both the forward and inverse transforms are applied without compression, then the transform is either perfectly reconstructing (lossless), or the image information is quantised and lost after the transform stage (lossy).

A lossless transform does not further complicate an image compressor since it makes no decisions about which parts of the image data are useful (so all these decisions can be made by the compressor algorithm). However a lossy transform can often produce more compression or allow the transform algorithm to run faster, both of which may be beneficial. The Discrete Cosine Transform (DCT) and the Wavelet Transform are examples transforms that are used in image compression[٥] .

١.٣.٣ Compression Measures

The compression of images is usually measured in two ways:

١. Compression ratio. The size of the original image is compared to the size of the compressed image.

٢. Bits Per Pixel(BPP). This is the number of bits necessary to describe one pixel of the image. This is generally an average over the whole image[٤].

١.٣.٤ Entropy Coding

We can define the entropy of a signal symbol a_i as $-P_i \log_2 P_i$. This is the smallest number of bits needed, on the average, to represent the symbol. The amount of information contained in one, base-n symbol is:

$$H = -\sum_{i=0}^{n-1} P_i \log_2 P_i \qquad \ldots(١.٤)$$

This quantity is called the entropy of the data being transited. The entropy of the data depends on the individual probabilities P_i, and is smallest when all n probabilities are equal[٦].

١.٣.٥ The Histogram

The histogram of an image is a plot of the gray-level values versus the number of pixels at that value. The shape of the histogram provides us with information about the nature of the image, or subimage if we are considering an object within the image. The sum of all of the values in

$$N = \sum_{i=0}^{M-1} H_i \qquad \ldots(١.٥)$$

The histogram must be equal to the number of points in the signal: where H_i is the histogram, N is the number of points in the signal, and M is the number of points in the histogram[٦].

١.٣.٦ Different Classes of Compression Techniques

Two ways of classifying compression techniques are mentioned here.

a- Lossless and **Lossy Compression:** In lossless compression schemes, the reconstructed image, after compression, is numerically identical to the original image. However lossless compression can only a achieve a modest amount of compression. An image reconstructed following lossy compression contains degradation relative to the original. Often this is because the compression scheme completely discards redundant information. However, lossy schemes are capable of achieving much higher compression. Under normal viewing conditions, no visible loss is perceived (visually lossless).

b- Predictive and **Transform Coding:** In predictive coding, information already sent or available is used to predict future values, and the difference is coded. Since this is done in the image or spatial domain, it is relatively simple to implement and is readily adapted to local image characteristics. Differential Pulse Code Modulation (DPCM) is one particular example of predictive coding. Transform coding, on the other hand, first transforms the image from its spatial domain representation to a different type of representation using some well-known transform and then codes the transformed values (coefficients). This method provides greater data compression compared to predictive methods, although at the expense of greater computation[٧].

١.٤ Color Spaces

A color space is a mathematical representation of a set of colors. The three most popular color models are RGB(used in computer graphics); YIQ, YUV, or YCbCr(used in video systems); and CMYK(used in color printing). All of the color spaces can be derived from the RGB information supplied by devices such as cameras and scanners. The red, green, and blue (RGB) color space is widely used throughout computer graphics. Red, green, and blue are three primary additive colors(individual components are added together to form a desired color) and represented by a three-dimensional Cartesian coordinate system. The RGB color space is the most prevalent choice for computer graphics because color displays use red, green, and blue to create the desired color. Therefore, the choice of the RGB color space simplifies the architecture and design of the system[٨].

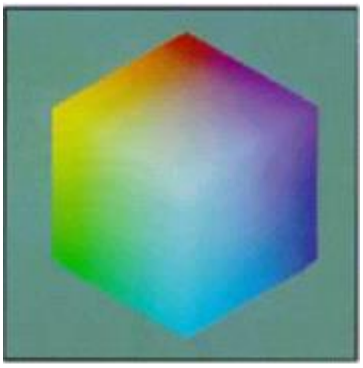


Figure (١.٤): The RGB Space

- **YUV Space**

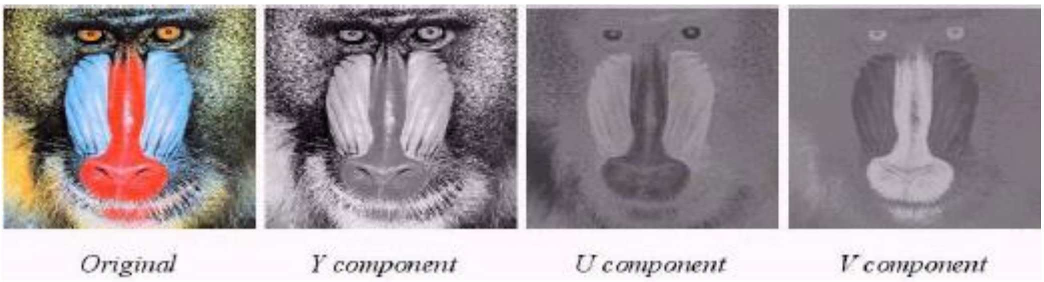
YUV was originally used for PAL (European standard) analog video as shown in Figure (١.٥). To convert from RGB to YUV spaces, the following equations can be used:

$$Y = 0.299\,R + 0.587\,G + 0.114\,B \qquad \ldots(١.٦)$$

$$U = 0.492\,(B - Y) \qquad \ldots(١.٧)$$

$$V = 0.877\,(R - Y) \qquad \ldots(١.٨)$$

Any errors in the resolution of the luminance (Y) are more important than the errors in the chrominance (U,V) values. The luminance information can be coded using higher bandwidth than the chrominance information[٩] .



Figure(١.٥): Example of YUV Space

YIQ Space

YIQ is used in the U.S. television standard, NTSC (National Television System Committee). It is similar to YUV, except that its color space is rotated ٣٣ degrees clockwise, so that I is the orange-blue axis, and Q is the purple-green axis. The equations to convert from RGB to YIQ are[٩] :

$$Y = 0.299\,R + 0.587\,G + 0.114\,B \qquad \ldots(١.٩)$$

$$I = 0.74(R - Y) - 0.27(B - Y) = 0.596\,R - 0.275\,G - 0.321\,B \qquad \ldots(١.١٠)$$

$$Q = 0.48(R - Y) + 0.41(B - Y) = 0.212\,R - 0.523\,G + 0.311\,B \qquad \ldots(١.١١)$$

- **YCrCb Space**

YCrCb is a subset of YUV that scales and shifts the chrominance values into the range of ٠ to ١. The linear transform from RGB to YCrCb generates one luminance space Y and two chrominance (Cr and Cb) spaces[٩] .

$$Y = 0.299\,R + 0.587\,G + 0.114\,B \qquad \ldots(١.١٢)$$

$$Cr = ((B - Y)/2) + 0.5 \qquad \ldots(١.١٣)$$

$$Cb = ((R - Y)/1.6) + 0.5 \qquad \ldots(١.١٤)$$

١.٥ Literature Survey

Many researches worked with image compression area using Embedded coding as a method for quantization or encoding coefficients results from one of many transform techniques, but we point to the ones closes to our search:

Jermo M Shapiro(١٩٩٣)[١٠], introduce effective and computationally simple technique for image compression called (EZW), The embedded wavelet algorithm(EZW) is a simple, yet remarkably effective, image compression algorithm, having the property that the bits in the bit stream are generated in order of importance, yielding a fully embedded code. The embedded code presents a sequence of binary decisions that distinguish an image from the "null" image. Using an embedded coding algorithm, an encoder can terminate the encoding at any point thereby allowing a target rate or target distortion metric to be met exactly. Also given a bit stream, the decoder can cease decoding at any point in the bit stream. The EZW algorithm is based on four key concepts:

١. A discrete wavelet transform.

٢. Prediction of absence significant information.

٣. Entropy-coded successive-approximation quantization.

٤. Universal lossless data compression .

Olivier Egger and others(١٩٩٥)[١١], proposed a new approach can be seen as a generalization of the Embedded Zerotree Wavelet (EZW) algorithm from rectangular images to arbitrarily shaped regions. In the case of compressing rectangular pictures this operation is performed using appropriate block transforms such as DCT or subband/wavelet transforms. Various approaches have been proposed to generalize the block-based techniques to arbitrary shaped regions. These techniques are either computationally very expensive and/or do not perform a full decorrelation of neighboring pixels in a given region . Special care has been dedicated to retain all the advantages of the EZW algorithm for arbitrarily shaped regions without any compromise. As for the original EZW algorithm the proposed technique is based on three basic blocks, namely the transformation, the zerotree prediction and the Successive Approximation Quantization (SAQ).

Amir Said and others(١٩٩٦)[١٢], offered an alternative explanation of the principles of J.M. Shapiro, so that the reasons for its excellent performance can be better understood. These principles are partial ordering by magnitude with a set partitioning sorting algorithm, ordered bit plane transmission, and exploitation of self-similarity across different scales of an image wavelet transform. Moreover, search presented a new and different implementation, based on set partitioning in hierarchical trees (SPIHT). In this paper, Amir Said again explain that the EZW technique is based on three concepts:

١. Partial ordering of the transformed image elements by magnitude, with transmission of order by a subset partitioning algorithm that is duplicated at the decoder.

٢. Ordered bit plane transmission of refinement bits.

٣. Exploitation of the self-similarity of the image wavelet transform across different scales. As to be explained, the partial ordering is a result of comparison of transform element (coefficient) magnitudes to a set of octavely decreasing thresholds. Amir Said explain that an element is significant or insignificant with respect to a given threshold, depending on whether or not it exceeds that threshold.

Amir Averbuch and others(١٩٩٨)[١٣], presented a fast and modified version of the embedded zero-tree wavelet (EZW) coding algorithm that is based on for low bit-rate still image compression applications. This method presented the trade-off between the image compression algorithm speed and the reconstructed image quality measured in terms of PSNR. The fast algorithm based on three different techniques :

١. Geometric wavelet decomposition. This technique speeds-up the multi-resolution wavelet decomposition significantly without any effect on the reconstructed image quality.

٢. Modified and reduced version of the EZW algorithm for zero-tree coefficient classification. Paper describe a very efficient data structure in which the zero-tree is scanned. In addition, a shorter scan in each quantization iteration is performed.

٣. A combination of exact model arithmetic coding and binary coding. This technique affects the obtained image quality because it is sub-optimal in comparison to the adaptive model arithmetic coding .

Each component of the algorithm contributes to the overall speedup.

Debin Zhao and others(١٩٩٨)[١٤], illustrate that the zerotree quantizer developed originally for wavelet compression can be effectively applied to Discrete Cosine Transform (DCT) in a hierarchical way. In this Hierarchical DCT (HDCT), the input image is partitioned into a number of ٨* ٨ blocks and a first level DCT is used to each of these blocks individually. As DC coefficients of DCT neighboring blocks are highly correlated and particularly pronounced to obtain the compression results at low bit rates, another level DCT is applied to only DC coefficients re-organized as ٨* ٨ blocks. This procedure is repeated until the last step is reached. All the HDCT coefficients within a DCT block are then rearranged into a subband structure in which the zerotree quantizer can be employed. The proposed algorithm yields a fully embedded, low-complexity coder with competitive PSNR performance.

١.٦ Aim of Thesis

The aim of the thesis is to present a new approach for image compression, the proposed system comprise four main stages:

- Convert the data of image into suitable color space .
- Apply Discrete Cosine Transform (DCT) on blocks (٨*٨) of data of acquired image.
- Apply Embedded Zerotree Algorithm on the DCT coefficients .
- And using Huffman Encoding for compress the symbols of previous step. The proposed system produces high compression ratio with minimum distortion.

١.٧ Thesis Layout

The thesis organized into five chapters:

- Chapter One: General Introduction.

  This chapter gives general introduction to the image compression.
- Chapter Two: Image Compression Techniques.

  This chapter describes the methods which used for compression.
- Chapter Three: DCT and EZW.

  This chapter describe in details Discrete Cosine Transform and Embedded Zerotree Wavelet.
- Chapter Four: The Proposed Image Compression System.

  This chapter presents the suggested compression system and discussion to the results .
- Chapter Five: Results, Conclusions and Suggestions for developing system in the future.

٣.١ Introduction

The rapid growth of digital imaging applications, including desktop publishing, multimedia, teleconferencing, and high-definition television (HDTV) has increased the need for effective and standardized image compression techniques, so most important of these standard and closes to the proposed compression system are Discrete Cosine Transform and Embedded Zerotree Wavelet.

٣.٢ Discrete Cosine Transform(DCT)

Among the emerging of compression techniques standards are JPEG, for compression of still images; MPEG, for compression of motion video (also known as Px٦٤), for compression of video telephony and teleconferencing. All three of these standards employ a basic technique known as the discrete cosine transform (DCT), the DCT is a close relative of the discrete Fourier transform (DFT)[٢٠].

٣.٢.١ The Two-Dimensional DCT Equations

The DCT equation computes the i, jth entry of the DCT

$$D_{ij} = \frac{1}{\sqrt{2n}} C_i C_j \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} P_{xy} \, Cos\left(\frac{(2x+1)i\pi}{2n}\right) Cos\left(\frac{(2y+1)j\pi}{2n}\right) \qquad \ldots(3.1)$$

where

$$C_f = \begin{cases} \frac{1}{\sqrt{2}} & f = 0 \\ 1, & f > 0 \end{cases} \qquad \text{and } 0 \le i, \ j \le n-1$$

p_{xy} is the x,yth element of the image represented by the matrix p. N is the size of the block that the DCT is done on. The equation calculates one entry(i,jth) of the transformed image from the pixel values of the original mage matrix. For the standard ٨*٨ block that JPEC compression uses, N equals ٨ and x and y rang from ٠ to ٧. Therefore D(i,j,) as shown in Equation (٣.٢).

$$D_{ij} = \frac{1}{4} C_i C_j \sum_{x=0}^{7}\sum_{y=0}^{7} P_{xy} \, Cos\left(\frac{(2x+1)i\pi}{16}\right) Cos\left(\frac{(2y+1)j\pi}{16}\right) \qquad ...(3.2)$$

The decoder works by using the Inverse Discrete Cosine Transform equation (IDCT) as shown in Equation (٣.٣)[٢١].

$$P_{xy} = \frac{1}{4} \sum_{x=0}^{7}\sum_{y=0}^{7} C_i C_j D_{ij} Cos\left(\frac{(2x+1)i\pi}{16}\right) Cos\left(\frac{(2y+1)j\pi}{16}\right) \qquad ...(3.3)$$

Where

$$C_f = \begin{cases} \frac{1}{\sqrt{2}} & f = 0 \\ 1, & f > 0 \end{cases}$$

٣.٢.٢ Doing the DCT on an ٨*٨ Block

Before beginning, it should be noted that the pixel values of a black-and-white image rang from ٠ to ٢٥٥, where pure black is represented by ٠, and pure white by ٢٥٥. Since an image comprises hundreds or even thousands of ٨*٨ blocks of pixels, the following description of what happens to one ٨*٨ block is a microcosm of the JPEG process; what is done to one block of image pixels is done to all of them, in the order earlier specified.

Now, let's start with a block of image-pixel values. This particular block was chosen from the very upper-left-hand corner of an image.

$$\text{Original} = \begin{bmatrix} 154 & 123 & 123 & 123 & 123 & 123 & 123 & 136 \\ 192 & 180 & 136 & 154 & 154 & 154 & 136 & 110 \\ 254 & 198 & 154 & 154 & 180 & 154 & 23 & 123 \\ 239 & 180 & 136 & 180 & 180 & 166 & 123 & 123 \\ 180 & 54 & 136 & 167 & 166 & 149 & 136 & 136 \\ 128 & 136 & 123 & 136 & 154 & 180 & 198 & 54 \\ 123 & 105 & 110 & 149 & 136 & 136 & 180 & 166 \\ 110 & 136 & 123 & 123 & 123 & 136 & 154 & 136 \end{bmatrix}$$

Now, the DCT is applied on the above matrix by using the equation(٣.٢). This yields the following matrix.

$$D = \begin{bmatrix} 1186 & 40 & 20 & 71 & 30 & 11 & -18 & -12 \\ 29 & 108 & 10 & 32 & 27 & -15 & 18 & -2 \\ -93 & -60 & 12 & -43 & -31 & 6 & -3 & 7 \\ -39 & -83 & -5 & -22 & -13 & 15 & -1 & 3 \\ -30 & 18 & -5 & -12 & 14 & -5 & 11 & -6 \\ -1 & -11 & 12 & 0 & 28 & 12 & 8 & 3 \\ 5 & -2 & 12 & 6 & -18 & -12 & 7 & 12 \\ -10 & 11 & 7 & -16 & 21 & 0 & 6 & 10 \end{bmatrix}$$

This block matrix now consists of ٦٤ DCT coefficients, D_{ij}, where I and j rang from ٠ to ٧. The top-left coefficient, $D_{٠٠}$, correlates to the low frequencies of the original image block . As we move away from $D_{٠٠}$ in all directions, the DCT coefficients correlate to higher and higher frequencies of the image block, where $D_{٧٧}$ corresponds to the highest frequency. It is important to note that the human eye is most sensitive to low frequencies, and results from the quantization step will reflect this fact[٢١].

٣.٢.٣ Quantization

The ٨*٨ block of DCT coefficients is now ready for compression by quantization. A remarkable and highly useful feature of JPEG process is that in this step, varying levels of image compression and quality are obtainable through selection of specific quantization matrices. This enables the user to decide on quality levels ranging from ١ to ١٠٠, where ١ gives the poorest image quality and highest compression, while ١٠٠ gives the best quality and lowest compression. As result, the quality/compression ratio can be tailored to suit different needs.

Subjective experiments involving the human visual system have resulted in the JPEG standard quantization matrix. With a quality level of ٥٠, this matrix renders both high compression and excellent decompressed image quality.

Table(٣.١): Quantization Table with Quality Level of ٥٠

$$Q_{50} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Quantization is achieved by dividing each element in the transformed image matrix D by the corresponding element in the quantization matrix, and then rounded to the nearest integer. For the following step, quantization matrix Q٥٠ is used.

$$C_{i,j} = \text{round}\left(\frac{D_{i,j}}{Q_{i,j}}\right) \qquad ...(٣.٤)$$

$$C = \begin{bmatrix} 74 & 4 & 2 & 5 & 1 & 0 & 0 & 0 \\ 3 & 9 & 1 & 2 & 1 & 0 & 0 & 0 \\ -7 & -5 & 1 & -2 & -1 & 0 & 0 & 0 \\ -3 & -5 & 0 & -1 & 0 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Recall that the coefficients situated near the upper-left corner correspond to the lower frequencies-to which the human eye is most sensitive-of the image block. In addition , the zeros represent the less important, higher frequencies that have been discarded, giving rise to the lossy part of compression. As mentioned earlier, only the remaining nonzero coefficients will be used to reconstruct the image. It is also interesting to note the effect of different quantization matrices[٢١].

٣.٢.٤ Encoding

After quantization, it is not unusual for more than half of the DCT coefficients to equal zero. JPEG incorporates run-length coding to take advantage of this. For each non-zero DCT coefficient, JPEG records the number of zeros that preceded the number, the number of bits needed to represent the number's amplitude, and the amplitude itself. To consolidate the runs of zeros, JPEG processes DCT coefficients in the zigzag pattern shown in figure (٣.١).

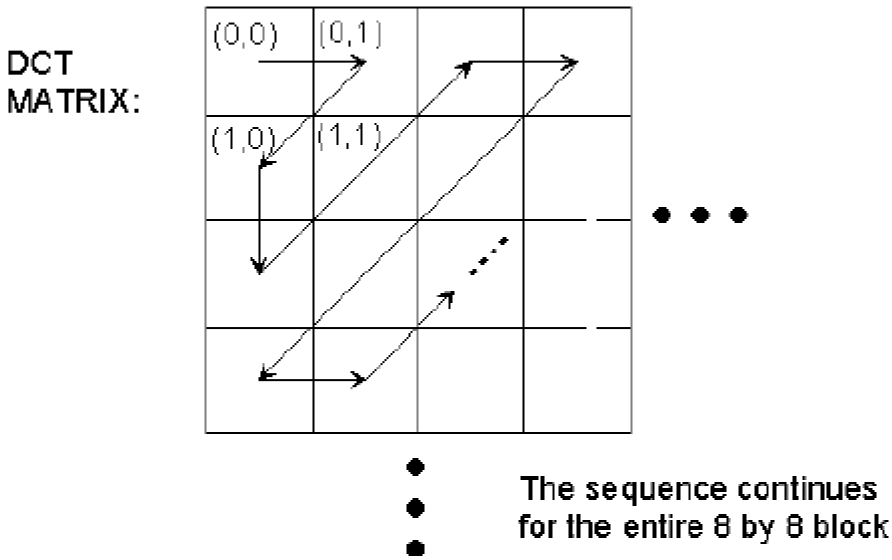


Figure (٣.١): Zig-Zag Sequence for Binary Encoding

The number of previous zeros and the bits needed for the current number's amplitude form a pair. Each pair has its own code word, assigned through a variable length code (for example Huffman, Shannon-Fano or Arithmetic coding). JPEG outputs the code word of the pair, and then the codeword for the coefficient's amplitude (also from a variable length code). After each block, JPEG writes a unique end-of-block sequence to the output stream, and moves to the next block. When finished with all blocks, JPEG writes the end-of-file marker[٢٢].

٣.٢.٥ Decompression

Reconstruction of image begins by decoding the bit stream representing the compressed matrix C. Each element of matrix C is then multiplied by the corresponding element of the quantization matrix originally used.

$$R_{i,j} = Q_{i,j} * C_{i,j} \qquad \ldots (٣.٥)$$

The IDCT equation (٣.٣) is next applied to matrix R, which is rounded to the nearest integer, giving us the decompressed of original ٨*٨ image block [٢١].

$$Decompressed = \begin{bmatrix} 149 & 133 & 119 & 116 & 121 & 125 & 127 & 127 \\ 203 & 168 & 140 & 143 & 155 & 150 & 135 & 124 \\ 252 & 195 & 155 & 166 & 182 & 165 & 130 & 110 \\ 244 & 184 & 148 & 166 & 183 & 160 & 124 & 107 \\ 187 & 149 & 132 & 154 & 172 & 159 & 140 & 136 \\ 131 & 122 & 124 & 142 & 159 & 165 & 167 & 171 \\ 109 & 119 & 125 & 127 & 139 & 158 & 167 & 165 \\ 110 & 126 & 127 & 113 & 118 & 140 & 146 & 134 \end{bmatrix}$$

٣.٣ Embedded Zerotree Wavelet(EZW)

When searching through wavelet literature for image compression schemes it is almost impossible not to note Shapiro's *Embedded Zerotree Wavelet* encoder or *EZW* encoder. An EZW encoder is an encoder specially designed to use with *wavelet transforms*, which explains why it has the word wavelet in its name. The EZW encoder was originally designed to operate on images (٢D-signals) but it can also be used on other dimensional signals.

The EZW encoder is based on *progressive encoding* to compress an image into a bit stream with increasing accuracy. This means that when more bits are added to the stream, the decoded image will contain more detail, a property similar to *JPEG* encoded images. It is also similar to the representation of a number like every digit we add increases the accuracy of the number, but we can stop at any accuracy we like. Progressive encoding is also known as *embedded encoding*, which explains the E in EZW. This leaves us with the Z. This letter is a bit more complicated to explain, but it will explained in the next section. Coding an image using the EZW scheme, together with some optimizations results in a remarkably effective image compressor with the property that the compressed data stream can have *any* bit rate desired. *Any* bit rate is only possible if there is information loss somewhere so that the compressor is *lossy*[٢٣].

٣.٣.٢ The Zerotree Structure

The EZW encoder is based on two important observations:

١. Natural images in general have a low pass spectrum. When an image is wavelet transformed the energy in the subbands decreases as the scale decreases (low scale means high resolution), so the wavelet coefficients will, on average, be smaller in the higher subbands than in the lower subbands. This shows that progressive encoding is a very natural choice for compressing wavelet transformed images, since the higher subbands only add detail.

٢. Large wavelet coefficients are more important than small wavelet coefficients.

These two observations are exploited by encoding the wavelet coefficients in decreasing order, in several passes. For every pass a threshold is chosen against which all the wavelet coefficients are measured. If a wavelet coefficient is larger than the threshold it is encoded and removed from the image, if it is smaller it is left for the next pass. When all the wavelet coefficients have been visited the threshold is lowered and the image is scanned again to add more detail to the already encoded image. This process is repeated until all the wavelet coefficients have been encoded completely or another criterion has been satisfied (maximum bit rate for instance). The trick is now to use the dependency between the wavelet coefficients across different scales to efficiently encode large parts of the image which are below the current threshold. It is here where the *zerotree* enters. If we want to compress the transformed signal we have to code not only the coefficient values, but also their position. After wavelet transforming an image we can represent it using trees because of the subsampling that is performed in the transform. A coefficient in a low subband can be thought of as having four descendants in the next higher subband as shown in Figure(٣.٢). The four descendants each also have four descendants in the next higher subband and we see a *quad-tree* emerge: every root has four leafs.

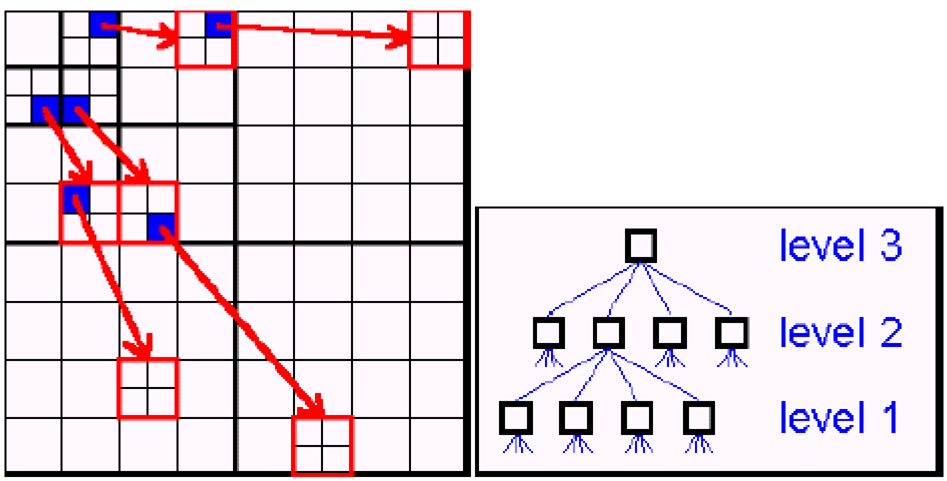
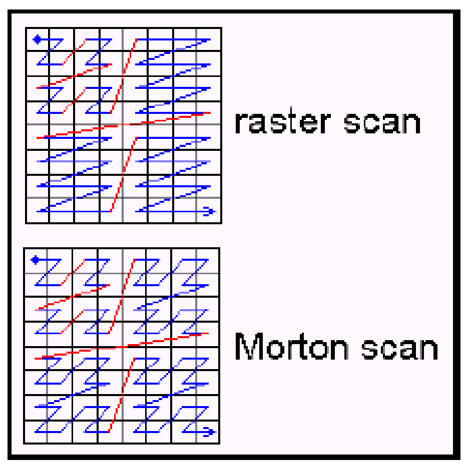


Figure (٣.٢): The Relations between Wavelet Coefficients in Different Subbands as Quad-Tree.

A zerotree is a quad-tree, the tree is coded with a single symbol and reconstructed by the decoder as a quadtree filled with zeroes. To clutter this definition we have to add that the root has to be smaller than the threshold against which the wavelet coefficients are currently being measured. The EZW encoder exploits the zerotree based on the observation that wavelet coefficients decrease with scale. It assumes that there will be a very high probability that all the coefficients in a quad tree will be smaller than a certain threshold if the root is smaller than this threshold. If this is the case then the whole tree can be coded with a single zerotree symbol.

One important thing is the coefficient positions. Indeed, without this information the decoder will not be able to reconstruct the encoded signal. EZW encoding uses a predefined scan order to encode the position of the wavelet coefficients as shown in Figure(٣.٣). Through the use of zerotrees many positions are encoded implicitly. Several scan orders are possible, as long as the lower subbands are completely scanned before going on to the higher subbands. In (Shapiro[٩]) a raster scan order is used. The scan order seems to be of some influence of the final compression result[٢٣].

١٢٥	-٧٨	٩٩	-٥	٧	١٣	-١٢	٧
١٥	٩	١	-٣	-٣	٤	٦	-١
١١	١٠	٤	٨	٥	-٧	٣	٩
٨	-٦	٥	-١٤	٤	-٢	٣	-٢
-٥	٩	-١	٩٥	٤	٦	-٢	٢
٣	٠	-٣	٢	٣	-٢	٠	٤
٢	-٣	٦	-٤	٣	٦	٣	٦
٥	١١	٥	٦	٠	٣	-٤	٤



Figure(٣.٣): An Example Data together with Two Scan Orders.

٣.٣.٣ The Concepts of EZW

The EZW image encoder follows the typical flow of data as shown in the Figure(٣.٤), and has three basic steps:

١-Transformation

٢-Quantization and

٣-Compression.

Transformation: EZW uses the Discrete Wavelet Transform (DWT) to transform the original image. In order to perform the DWT, the image has to be a square image, and its row/column size must be an integer power of ٢. So, technically, the EZW is applicable to the square images of sizes in integer powers of ٢ (for example, image sizes like ١٢٨ x ١٢٨). This transformation is theoretically lossless, although this may not always be the case. The purpose of the transformation is to generate decorrelated coefficients, which means it removes all the dependencies between samples.

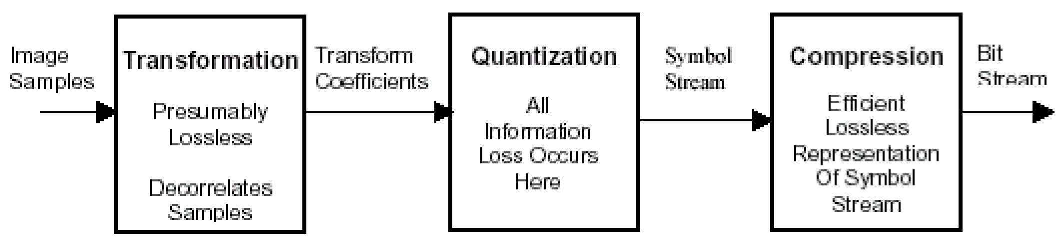


Figure (٣.٤): Typical Flows of Data of Image Encoder

Quantization: This step involves the quantization of transformed coefficients. Thus, the entropy of the resulting distribution of the bin indexes is small enough that the symbols can be entropy coded at some low target bit rate. Quantizers are symmetrically read. Assuming the central index is zero, which treats positive or negative indexes alike, all quantizers are set to be symmetric. The main advantage of symmetry is that it saves the bits needed to represent the symbols since encoding of a non-zero coefficient requires at least one bit per sign. An entropy code can be designed using the probabilities of the bin indices as the fraction of coefficients in which the only absolute values of bin indexes are involved.

The EZW uses Successive Approximation Quantization (SAQ). Successive approximation quantization serves two purposes in the EZW algorithm. First, it is used as a method to generate a large number of zero-trees, which is good since zero-trees are easily coded. Second, successive approximation quantization is used to sort the bit order of coded bits so that the most significant bits are sent first. EZW implements successive approximation quantization through a multipass scanning of the wavelet coefficients using successively decreasing thresholds $T_0, T_1, T_2....$The initial threshold is set to the value of $T_0 = 2^{\lfloor \log_2 (max\_coeff) \rfloor}$ where max_coeff is the largest wavelet coefficient. Each scan of wavelet coefficients is divided into two passes: dominant and subordinate. The dominant pass establishes a significance map of the coefficients relative to the current threshold T_i . Thus, coefficients which are significant on the first dominant pass are known to lie in the interval$[T_0, 2T_0)$, and can be represented with the reconstruction value of$(3T_0/2)$. The dominant pass essentially establishes the most significant bit of binary representation of the wavelet coefficient, with the binary weights being relative to the thresholds T_i. the positions of the significant and the insignificant coefficients are indicated in significance maps.

<u>Compression</u>: The concept of a zerotree data structure is applied in the compression process of the significance map. Each wavelet coefficient is compared with the threshold, Ti, to determine its significance. In addition to encoding the significance map, further encoding of significant coefficients is done using signs. All the significant coefficients are encoded into only four signs, zerotree root(ZT), isolated zero(IZ), positive significant(P), and negative significant(N). Encoding into symbols makes embedded coding handy.

EZW follows adaptive arithmetic coding for compression. The main advantage of arithmetic coding in this algorithm is that it contains a maximum of four symbols at any time. For instance, the encoder contains two symbols for subordinate passes, three symbols for dominant passes with no zerotree symbol and four symbols for dominant passes with zerotree symbol (the terms dominant pass and subordinate pass will be explained later). Because the maximum number of symbols is set to four, the occurrence of the possible symbols can be measured with less effort. This advantage lets the algorithm use a short memory to learn quickly and constantly changing symbol probabilities[٢٤].

٣.٣.٤ EZW Algorithm

The EZW coding algorithm can be summarized as follows.

١-Initialization: Place all wavelet coefficients on the dominant list. Set the initial threshold to

$$T_0 = 2^{\lfloor \log_2 (max\_coeff) \rfloor} \quad ...(٣.٦)$$

٢- Dominant Pass: Scan the coefficients on the dominant list using the current threshold T_i and subband ordering shown in Figure(٣.٣). Assign each coefficient one of four symbols:

- ✓ Positive significant (ps): Meaning that the coefficient is significant relative to the current threshold T_i and positive.
- ✓ Negative significant (ns): Meaning that the coefficient is significant relative to the current threshold T_i and negative.
- ✓ Isolated Zero (iz): Meaning the coefficient is insignificant relative to the threshold T_i and one or more of its descendants are significant.
- ✓ Zero-Tree Root (ztr): Meaning the current coefficient and all of its descendants are insignificant relative to the current threshold T_i.

Any coefficient that is the descendant of a coefficient that has already been coded as a zero-tree root is not coded, since the decoder can deduce that it has a zero value. Coefficients found to be significant are moved to the subordinate list and their values in the original wavelet map are set to zero. The resulting symbol sequence is entropy coded.

٣- Subordinate Pass: Output a ١ or a ٠ for all coefficients on the subordinate list depending on whether the coefficient is in the upper or lower half of the quantization interval.

٤- Loop: Reduce the current threshold by two, $T_i = T_i$ / ٢. Repeat the Steps (٢) through (٤) until the target fidelity or bit rate is achieved[٢٥].

٣.٣.٥ EZW an Example

The example of a ٢-scale wavelet transformation of an ٨x٨ image is used to explain the algorithm. The image values are shown in Figure (٣.٥). The initial threshold (T٠) is determined according to the equation of Thresholding, $T_0=2^{|\log_2 (max\_coeff)|}$, and so the first step is to find the maximum image value, seen to be ١٢٥ in Figure (٣.٥). Then the initial threshold can be set to ٦٤. The Dominant List is actually the same as the image, which is an ٨x٨ array of pixel values. The Subordinate List is a one-dimensional row matrix, initially a null matrix. The first Dominant Pass is then conducted using the initial threshold ٦٤ and is explained as follows.

١٢٥	-٧٨	٩٩	-٥	٧	١٣	-١٢	٧
١٥	٩	١	-٣	-٣	٤	٦	-١
١١	١٠	٤	٨	٥	-٧	٣	٩
٨	-٦	٥	-١٤	٤	-٢	٣	-٢
-٥	٩	-١	٩٥	٤	٦	-٢	٢
٣	٠	-٣	٢	٣	-٢	٠	٤
٢	-٣	٦	-٤	٣	٦	٣	٦
٥	١١	٥	٦	٠	٣	-٤	٤

Figure (٣.٥): An ٨x٨ Sample Image

١- The first coefficient, ١٢٥, which is in level ٣, subband LL٣, is greater than the threshold ٦٤ and is positive. Therefore, a positive symbol **P** is coded as illustrated in Figure (٣.٦).

٢- The scanning order of the coefficients is –٧٨, ١٥ and ٩, which belong to the ٣rd level subbands HL٣, LH٣ and HH٣, respectively. Compared to the threshold ٦٤, -٧٨ is greater but negative, and so **N** is coded as illustrated in as illustrated in Figure(٣.٧).

٣- The coefficient ١٥ is insignificant compared with coefficient ٦٤. The ٢nd level subband LH٢ coefficients {١١, ١٠, ٨, -٦) are also insignificant compared to coefficient ٦٤. However, the ١st level subband LH١ has a significant coefficient ٩٥, and so the root of the zerotree ١٥ is coded as insignificant zero, **IZ** as illustrated in Figure (٣.٨).

٤- The coefficient ٩ is less than ٦٤, and the next finer subband coefficients {٤, ٨, ٥, -١٤} and {٤, ٦,٣,-٢, ... , ٣,٦, -٤, ٤} are also insignificant. Therefore, ٩, the root of the zerotree is coded as **ZT** as illustrated in Figure (٣.٩).

٥- The scanning of the coefficients follows the order ٩٩, -٥, ١, -٣ then ١١, ١٠, ٨, -٦ and ٤, ٨, ٥, -١٤. These coefficients's location can be represented as the second level subbands, which are HL٢, LH٢ and HH٢ as illustrated in Figure (٣.١٠).

١٢٥	-٧٨	٩٩	-٥	٧	١٣	-١٢	٧
١٥	٩	١	-٣	-٣	٤	٦	-١
١١	١٠	٤	٨	٥	-٧	٣	٩
٨	-٦	٥	-١٤	٤	-٢	٣	-٢
-٥	٩	-١	٩٥	٤	٦	-٢	٢
٣	٠	-٣	٢	٣	-٢	٠	٤
٢	-٣	٦	-٤	٣	٦	٣	٦
٥	١١	٥	٦	٠	٣	-٤	٤

Figure(٣.٦): Step١ of Dominant Pass

١٢٥	-٧٨	٩٩	-٥	٧	١٣	-١٢	٧
١٥	٩	١	-٣	-٣	٤	٦	-١
١١	١٠	٤	٨	٥	-٧	٣	٩
٨	-٦	٥	-١٤	٤	-٢	٣	-٢
-٥	٩	-١	٩٥	٤	٦	-٢	٢
٣	٠	-٣	٢	٣	-٢	٠	٤
٢	-٣	٦	-٤	٣	٦	٣	٦
٥	١١	٥	٦	٠	٣	-٤	٤

Figure(٣.٧): Step٢ of Dominant Pass

١٢٥	-٧٨	٩٩	-٥	٧	١٣	-١٢	٧
١٥	٩	١	-٣	-٣	٤	٦	-١
١١	١٠	٤	٨	٥	-٧	٣	٩
٨	-٦	٥	-١٤	٤	-٢	٣	-٢
-٥	٩	-١	٩٥	٤	٦	-٢	٢
٣	٠	-٣	٢	٣	-٢	٠	٤
٢	-٣	٦	-٤	٣	٦	٣	٦
٥	١١	٥	٦	٠	٣	-٤	٤

Figure(٣.٨): Step٣ of Dominant Pass

١٢٥	-٧٨	٩٩	-٥	٧	١٣	-١٢	٧
١٥	٩	١	-٣	-٣	٤	٦	-١
١١	١٠	٤	٨	٥	-٧	٣	٩
٨	-٦	٥	-١٤	٤	-٢	٣	-٢
-٥	٩	-١	٩٥	٤	٦	-٢	٢
٣	٠	-٣	٢	٣	-٢	٠	٤
٢	-٣	٦	-٤	٣	٦	٣	٦
٥	١١	٥	٦	٠	٣	-٤	٤

Figure(٣.٩): Step٤ of Dominant Pass

١٢٥	-٧٨	٩٩	-٥	٧	١٣	-١٢	٧
١٥	٩	١	-٣	-٣	٤	٦	-١
١١	١٠	٤	٨	٥	-٧	٣	٩
٨	-٦	٥	-١٤	٤	-٢	٣	-٢
-٥	٩	-١	٩٥	٤	٦	-٢	٢
٣	٠	-٣	٢	٣	-٢	٠	٤
٢	-٣	٦	-٤	٣	٦	٣	٦
٥	١١	٥	٦	٠	٣	-٤	٤

Figure(٣.١٠): Step٥ of Dominant Pass

٦- Since coefficient ٩٩ is greater than ٦٤ and positive, it is coded as **P**. The next three coefficients, -٥, ١, and –٣, are coded as **ZT,** because they are not descendants of any other zerotree root and their own descendants are insignificant compared to coefficient ٦٤. The preceding statement can also be illustrated as follows: because -٧٨, the parent coefficient of ٩٩,– ٥, ١ and –٣, was significant and ١٥, the parent coefficient of ١١, was coded as isolated zero as illustrated in Figure(٣.١١), also Figure (٣.٧) and (٣.٨).

٧- The coefficient ١٠ is less than ٦٤, but it has a significant descendant ٩٥ in the next generation. So, it is coded as isolated zero **IZ**. ٨ and –٦ are coded as **ZT**s (zerotree) as explained in the previous step as shown in Figure (٣.١٢).

١٢٥	-٧٨	٩٩	-٥	٧	١٣	-١٢	٧
١٥	٩	١	-٣	-٣	٤	٦	-١
١١	١٠	٤	٨	٥	-٧	٣	٩
٨	-٦	٥	-١٤	٤	-٢	٣	-٢
-٥	٩	-١	٩٥	٤	٦	-٢	٢
٣	٠	-٣	٢	٣	-٢	٠	٤
٢	-٣	٦	-٤	٣	٦	٣	٦
٥	١١	٥	٦	٠	٣	-٤	٤

Figure(٣.١١): Step٦ of Dominant Pass

١٢٥	-٧٨	٩٩	-٥	٧	١٣	-١٢	٧
١٥	٩	١	-٣	-٣	٤	٦	-١
١١	١٠	٤	٨	٥	-٧	٣	٩
٨	-٦	٥	-١٤	٤	-٢	٣	-٢
-٥	٩	-١	٩٥	٤	٦	-٢	٢
٣	٠	-٣	٢	٣	-٢	٠	٤
٢	-٣	٦	-٤	٣	٦	٣	٦
٥	١١	٥	٦	٠	٣	-٤	٤

Figure(٣.١٢): Step٧ of Dominant Pass

٨- The next coefficients to be coded are ٤, ٨, ٥ and –١٤. All four are insignificant compared to ٦٤ and they are in the non-root part of the zerotree, as their ancestor ٩ is the root of the zerotree. Therefore, all four coefficients are left un-encoded as illustrated in Figure (٣.١٣).

٩- Until now, all the coefficients, except the coefficients in the subbands LH١,HL١,HH١, are encoded. Usually, most of the higher frequency subband coefficients are not encoded at higher threshold values, as they are often descendants of roots of zerotree or usually insignificant. For the level-١ subbands (HL١, LH١ and HH١), the encoder uses only ٣ symbols (P, N, Z), because these subband coefficients do not have any descendents and cannot be the roots of a zerotree. The final significance map is shown in Figure (٣.١٤).

During the first Dominant Pass of reconstruction, if the decoder sees a symbol **P** and already knows the initial threshold value to be ٦٤, the decoder outputs ٩٦, the midpoint of the range [٦٤, ١٢٨], as the reconstructed value as shown in Table (٣.٢). However, the actual value of the coefficient in the original image is ١٢٥. The difference between the original and the reconstructed coefficient is higher, and so EZW uses another pass i.e. Subordinate Pass to refine the encoding information of the already found significant coefficients. Subordinate Pass is performed immediately after each Dominant Pass.

١٢٥	-٧٨	٩٩	-٥	٧	١٣	-١٢	٧
١٥	٩	١	-٣	-٣	٤	٦	-١
١١	١٠	٤	٨	٥	-٧	٣	٩
٨	-٦	٥	-١٤	٤	-٢	٣	-٢
-٥	٩	-١	٩٥	٤	٦	-٢	٢
٣	٠	-٣	٢	٣	-٢	٠	٤
٢	-٣	٦	-٤	٣	٦	٣	٦
٥	١١	٥	٦	٠	٣	-٤	٤

Figure(٣.١٣): Step٨ of Dominant Pass

P	N	P	ZT	Z	Z	*	*
IZ	ZT	ZT	ZT	Z	Z	*	*
ZT	IZ	*	*	*	*	*	*
ZT	ZT	*	*	*	*	*	*
*	*	Z	P	*	*	*	*
*	*	Z	Z	*	*	*	*
*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*

Figure(٣.١٤): Step٩ of Dominant Pass

Table (٣.٢): First Dominant Pass

Scanning Order	Subband	Coefficients Value	Symbol	Reconstruction Value
١	LL٣	١٢٥	P	٩٦
٢	HL٣	-٧٨	N	-٩٦
٣	LH٣	١٥	IZ	٠
٤	HH٣	٩	ZT	٠
٥	HL٢	٩٩	P	٩٦
٦	HL٢	-٥	ZT	٠
٧	HL٢	١	ZT	٠
٨	HL٢	-٣	ZT	٠
٩	LH٢	١١	ZT	٠
١٠	LH٢	١٠	IZ	٠
١١	LH٢	٨	ZT	٠
١٢	LH٢	-٦	ZT	٠
١٣	HL١	٧	Z	٠
١٤	HL١	١٣	Z	٠
١٥	HL١	-٣	Z	٠
١٦	HL١	٤	Z	٠
١٧	LH١	-١	Z	٠
١٨	LH١	٩٥	P	٩٦
١٩	LH١	-٣	Z	٠
٢٠	LH١	٢	Z	٠

Table(٣.٣): First Subordinate Pass

Coefficient Absolute Value	Arithmetic Coding of the Symbol	Reconstruction Value
١٢٥	١	١١٢
-٧٨	٠	٨٠
٩٩	١	١١٢
٩٥	٠	٨٠

The following comments explain the first subordinate pass as illustrated in Table (٣.٣).

١- During the first dominant pass, a subordinate list is created containing only significant coefficients, which are encoded either as P or as N. Thus, for the above example, the subordinate list is {١٢٥, -٧٨, ٩٩, ٩٥}.

٢- Two intervals, upper and lower, exist for each subordinate pass, depending on the threshold value. For threshold ٦٤, the upper interval is defined between [٩٦, ١٢٨], and the lower interval between [٦٤, ٩٦].

٣- The first coefficient of the subordinate list, ١٢٥, belongs to the upper level, and so it is encoded as **H**. The reconstruction value is the center of the upper interval, or ١١٢.

٤- The next coefficient is ٧٨, which is placed in the lower interval and encoded as **L**. The reconstruction value is the center of the lower interval, or ٨٠.

٥- The third entry, ٩٩, is encoded as **H** and has a reconstruction value of ١١٢. Finally, the last entry, ٩٥, is encoded as **L**, and its reconstruction value is set to ٨٠.

Notice that the reconstruction value after the subordinate pass of the coefficient ٩٥ is changed from ٩٦ to ٨٠, which results in an increase of reconstruction error from ١ to ١٥. However, the uncertainty interval is decreased from ٣٢ (٦٤<٩٦<١٢٨) to ١٦ (٦٤<٨٠<٩٦), which will ensure overall improvement of reconstruction error. The subordinate list from the first subordinate pass is carried over to the second subordinate pass, and the significant values generated are placed next to the previously found significant values. In addition, the subordinate list's coefficients are reordered based on decreasing order of the reconstruction values. Initially, the subordinate list follows the same order as the scanning order, which is {١٢٥, ٧٨, ٩٩, ٩٥}, and the reconstruction values of the respective coefficients are given as {١١٢, ٨٠, ١١٢, ٨٠}. After the first subordinate pass, the reconstructed values, in descending order, are {١١٢, ١١٢, ٨٠, ٨٠}. They make coefficient ٩٩ precede coefficient ٧٨, and so, the new order for future subordinate passes is {١٢٥, ٩٩, ٧٨, ٩٥}. Notice that coefficient ٩٩ still precedes coefficient ٧٨, which is smaller, because the decoder considers both the coefficients alike since their reconstructed values are same[٢٤].

٣.٣.٦ Decoding the Bitstream Generated by EZW

The decoding process follows the same steps as the encoder. The decoder also uses two passes, Dominant Pass and Subordinate Pass, during the reconstruction process, similar to the dominant pass and subordinate pass of the encoder. In the beginning, the image to be reconstructed is initialized to all zeros, and then the encoded bitstream is passed through the dominant pass.

The bitstream comes with a header that contains the essential information needed for the decoder, such as initial threshold value, image dimensions, and the number of levels used for the DWT decomposition. As the bitstream is in binary ١'s and ٠'s, the decoder reads two bits from the bitstream and turns them into symbols. If the decoder finds a positive or negative symbol, it places the reconstruction value of that particular pass, which is ٣/٢ times the threshold value, at the corresponding location. If the symbol is zerotree root (ZT) or insignificant zero (IZ), the corresponding coefficient positions are filled with the suitable values.

The Dominant Pass ends after scanning all coefficients of the image. The scanning order must be same for both encoder and decoder. Subordinate

Pass II reads one bit from the bitstream. As discussed in earlier sections, if the bit is “١”, then the corresponding coefficient’s reconstruction value is reorganized to a higher value; if “٠”, then to a lower value. The term “embedded” is justified in the decoding process, as the decoder can stop at any time. The quality of the reconstructed image is directly proportional to the number of bits decoded. If the image is encoded until the least value is recognized, and if the bitstream is decoded until the last bit, then the resulting reconstructed image will be closes to same as the original image[٢٤].

٤.١ Introduction

This chapter presents practical implementation to general embedded bit streams for Discrete Cosine Transform(DCT) coefficients according to their importance. Two algorithms were suggested, the first one for compression and the other for decompression as shown in Figure (٤.١).

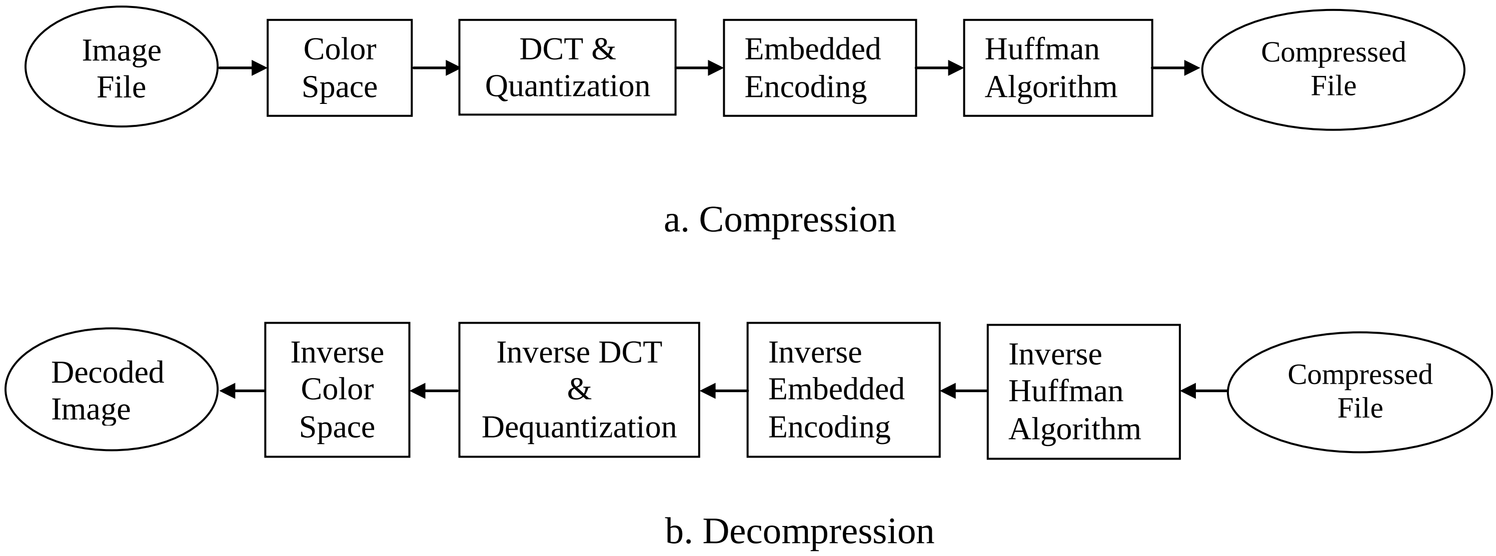
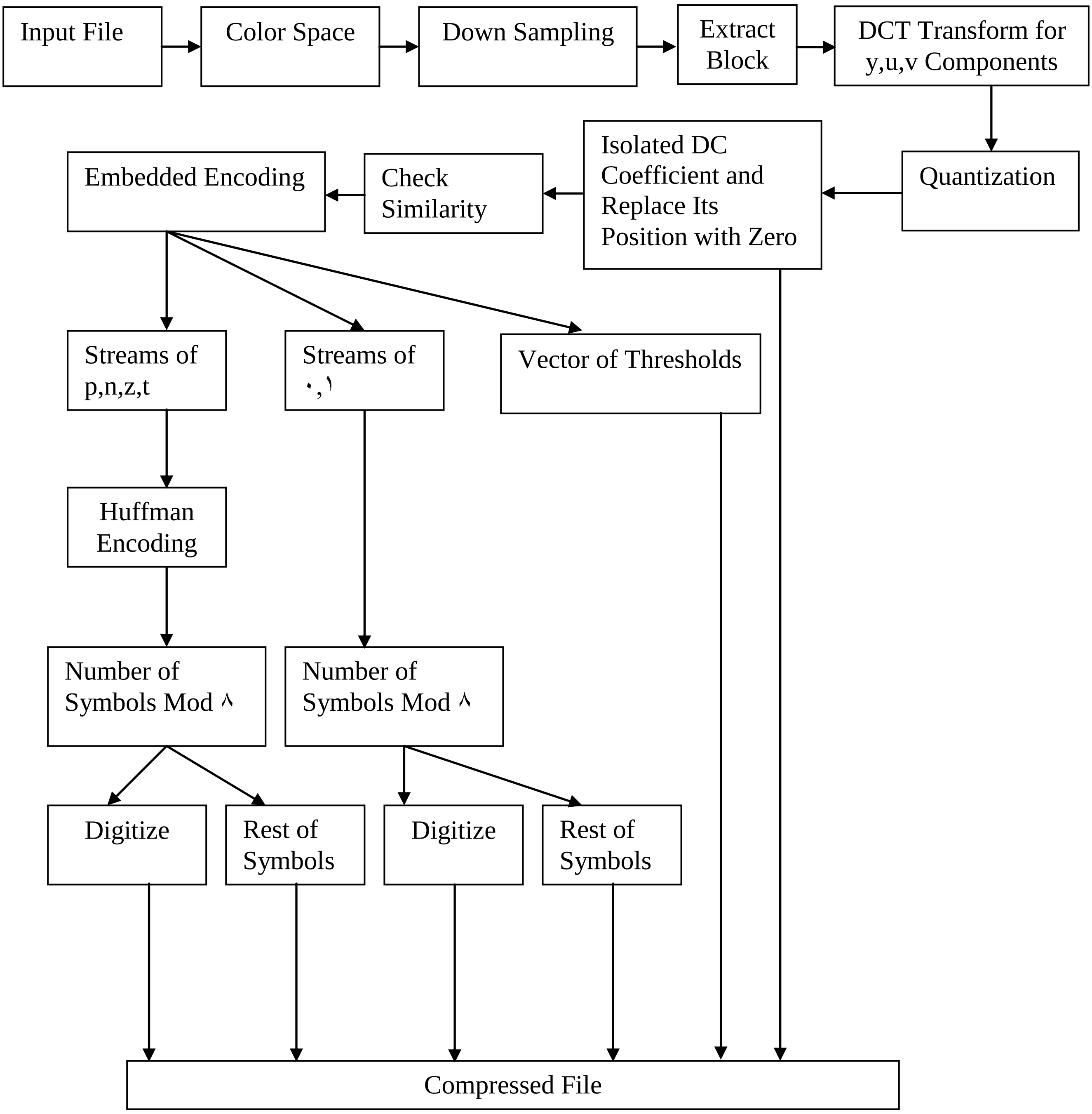


Figure (٤.١): Basic Components of The Proposed Compression System

٤.٢ The Proposed Compression ِAlgorithm Steps

The stages of the suggested compression algorithm is illustrated in Figure (٤.٢) which represents block diagram for the Basic stages as well as the inputs, outputs , and substages embedded in system.



Figure(٤.٢): Block Diagram for Basic Stages of The Proposed Algorithm

<u>Step One</u> : Read BMP(٢٤ bits) image

BMP image file usually consists of two part, first part contains general information about image and called Header, while the other part called Data and contains the values of color image.

<u>Step Two</u>: Color Space and Dawn Sampling

Transform color image into a suitable color space, generally RGB transform into a luminance/chrominance color space (YUV), while the luminance component is grayscale and the other two axes are color information. The reason for this stage is that minimize the information in the chrominance component, the human eye is not as sensitive to high-frequency chrominance information as it is to high frequency luminance.

Dawn Sampling is optional substage performed by averaging together groups of values, the luminance components is left at full resolution, while the chrominance components are reduced to "٤٢٢" sampling (there are another samples), in numerical terms it is lossy.

<u>Step Three</u>: Dividing Components

Group the values for each component in to ٨*٨ blocks, The ٨*٨ block is the basis of the JPEG standard, as well as several others standards such as MPEG-١, the reason for this are simplify the arithmetic operation and reduce the artifacts effects results from apply DCT algorithm later, although the compression ratio with entire image produce better results.

<u>Step Four</u>: Apply DCT And Quantization

Transform each ٨*٨ block through a discrete Cosine Transform(DCT), the result is ٨*٨ block, the first element at ٠,٠ position is called the DC coefficient and the remaining ٦٣ coefficients in each block are called AC coefficients, round the AC's to integers, the DC represent the average of all values in each block.

The quantization is the process of reducing the number of possible values of block, in other hand reduce the data required to represent the image. Simple example of quantization is the rounding of real values into integers, other scheme for quantization using table construct depend on one parameter R supplied by user, a simple expression such as

$$Q_{ij} = 1 + (i + j) * R \qquad \ldots(٤.١)$$

satisfy required table. In the proposed system the quantization accomplished by using Default tables were the elements in the table generally grow as we move from the upper left corner to the bottom right one as in Table(٣.١).

<u>Step Five</u>: Isolate DC Coefficient

This step is accomplished by isolate DC coefficient of each block and replace it position with zero(stars), the isolated DC's of blocks are stored in individual array and sent to the compressed file without any additional process, the reason for this step is to avoided problems arise from very large value of DC compared with AC's, usually Embedded Zerotree Algorithm gives better results with blocks have closes values.

<u>Step Six</u>: Check Similarity

The values of each block are tested, if all values are same, the block skips the Embedded Zerotree Algorithm and one value taken from block and stored in individual array. In this case the system store two parameters for reconstructing current block, the DC coefficient represent first element of block and one value using for reconstructing all AC's coefficients.

<u>Step Seven</u>: Embedded Zerotree Algorithm

Is a simple, yet remarkable effective image compression algorithm, more details will be presented later.

Step Eight: Huffman Algorithm

Its very known algorithm. The reason for using this algorithm is to the small alphabet of the proposed algorithm with high frequency for specific symbol (usually 't' which refer to insignificant value).

Step Nine: Digitize

Group each eight bits and convert its into corresponding decimal number. Moreover the rest bits place in individual array and stored without any additional operation(rest bits result from apply MOD function between the whole number of bits and ٨).

Embedded Zerotree Algorithm

Embedded Zerotree Algorithm presented by J. Shapiro(١٩٩٣), consist of four main steps which are: threshold, dominant, subordinate, and loop. To make this algorithm appropriate to work with DCT coefficients which completely defers from Wavelet coefficients, the proposed system includes another steps to satisfy this goal. Figure(٤.٣) represents flowchart contain the main steps of Embedded Coding.

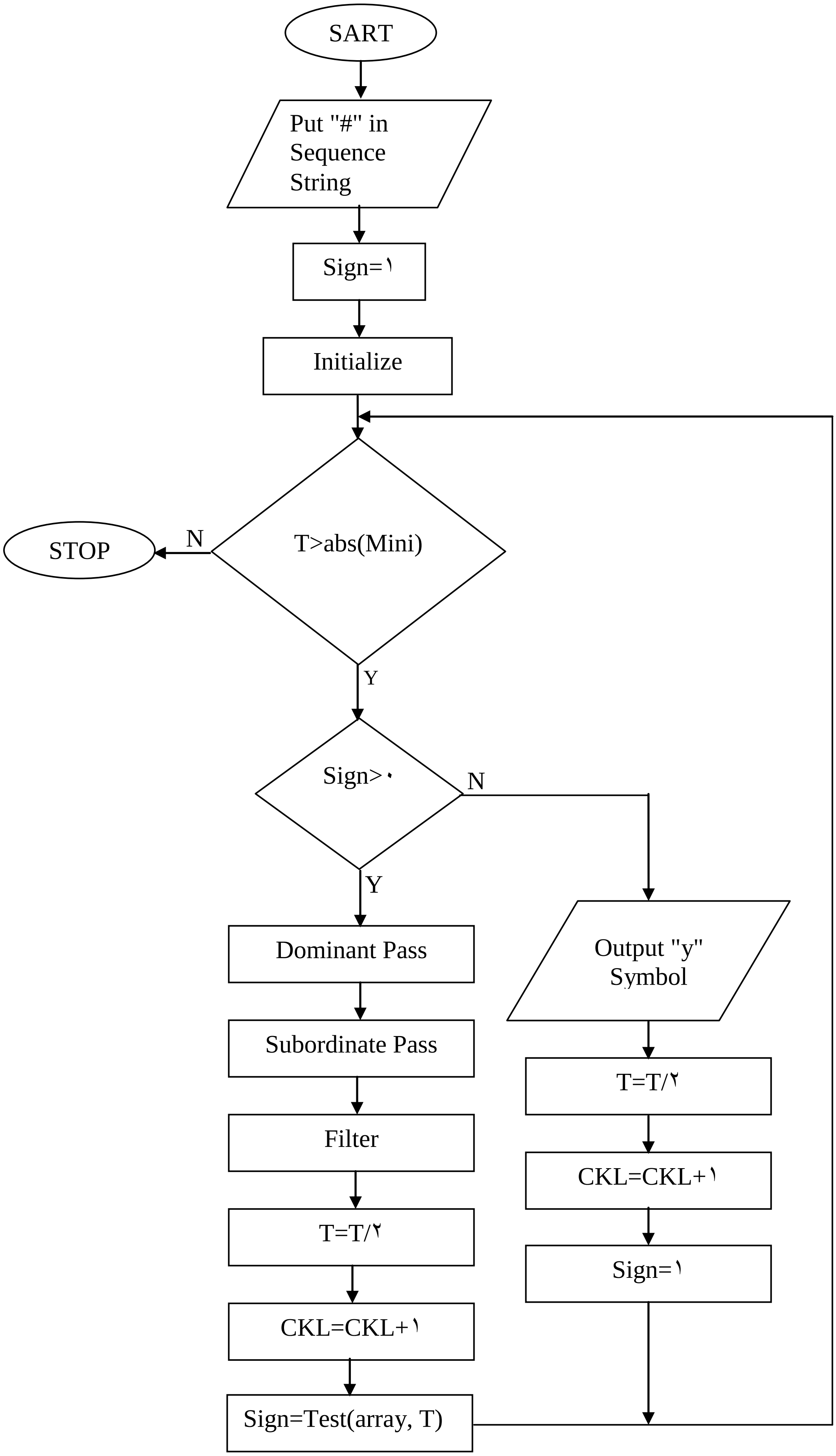
The used alphabetic contain eight symbols "p, n, z, t, #, @, -, y " , while first four symbols represent the original alphabetic used by Shapiro, other symbols added by proposed algorithm.

Where '#' represent an new block of coefficients,

'@' refers to similar block(all coefficients have same value),

'-' refers to a new row (in another word new cycle),

'y' refers to special case appear when the returned value(sign) from Test function equaled to zero.



Figure(٤.٣): Flowchart of The Embedded Zerotree Algorithm

Initialize

For each block a new threshold is specified, the threshold determines which of Coefficients should be classified as significant and which of coefficients should be classified as insignificant, significance concept means that the coefficient greater than the current threshold while insignificance concept refers to the coefficient less than the current threshold. There are many methods for finding the threshold for each block, most of these methods based on the maximum values of block, some of implementation, the initial threshold is set to be half of the maximum absolute value, in the suggested system, threshold obtained by apply equation(٣.٦).

Test Function

This function test all coefficients if there are at least one absolute value greater than Mini(Mini absolute smallest number of block), in the same time this coefficient is smaller than current threshold. To explain this case take this example array.

١٣	٩	١	٠	٠	٠	٠	٠
٣	-٢	٠	٠	٠	٠	٠	٠
١	٠	٠	٠	٠	٠	٠	٠
٠	٠	٠	٠	٠	٠	٠	٠
٠	٠	٠	٠	٠	٠	٠	٠
١	٠	٠	٠	٠	٠	٠	٠
٠	٠	٠	٠	٠	٠	٠	٠
٠	٠	٠	٠	٠	٠	٠	٠

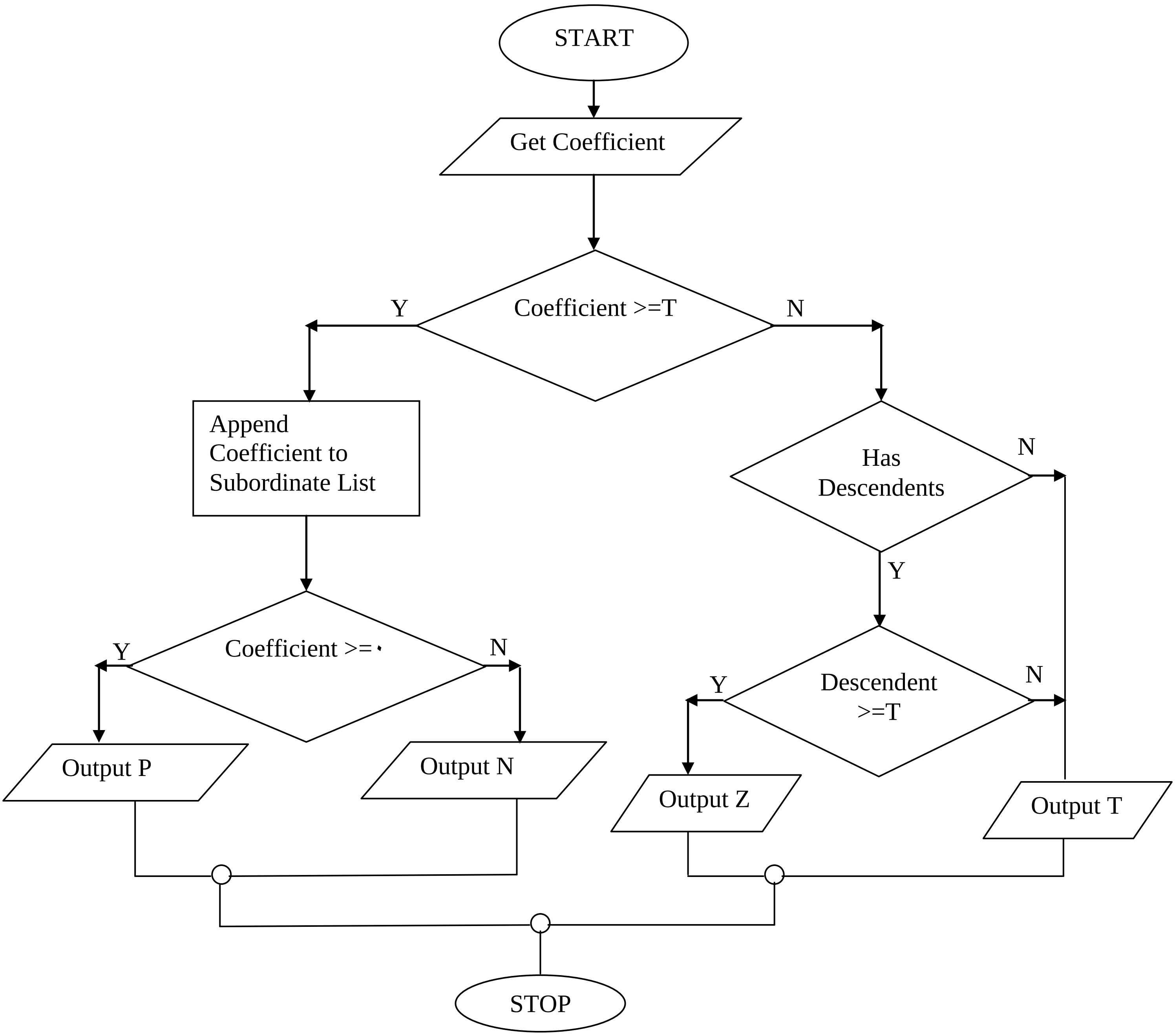
Figure(٤.٤): An Example Illustrate Special Case

In this case the absolute maximum coefficient is ١٣ and by applying the equation (٣.٦) for finding the threshold value, the result is ٨, so ٨ represent the initial value. In the first phase there two coefficient greater than current threshold (٩,١٣) and will be extracted and replaced there positions with zero. In the second phase or cycle the initial threshold halved and the current threshold equaled to ٤, also there are many absolute value greater than Mini, but smaller than current threshold, to process this drop, "Test" function is added to specify special character 'y' placed in individual row of result array .

Dominant Pass

Dominant pass as illustrated in Figure (٤.٥) checks all trees for significant values with respect to a certain threshold. If the coefficient is larger than the threshold, a positive (p) is coded. If the coefficient is negative, but it's a absolute value is larger than the current threshold, a negative(n) is coded . If the coefficient and his children smaller than the current threshold, a zerotree (t) is coded . If coefficient smaller than threshold, but has children (at least one) greater than current threshold, an isolated(z) is coded. The coefficients coded as positive (p) or negative (n) consider significant.

Finally, all the coefficients that are in absolute value larger than the current threshold are extracted and placed on the subordinate list.

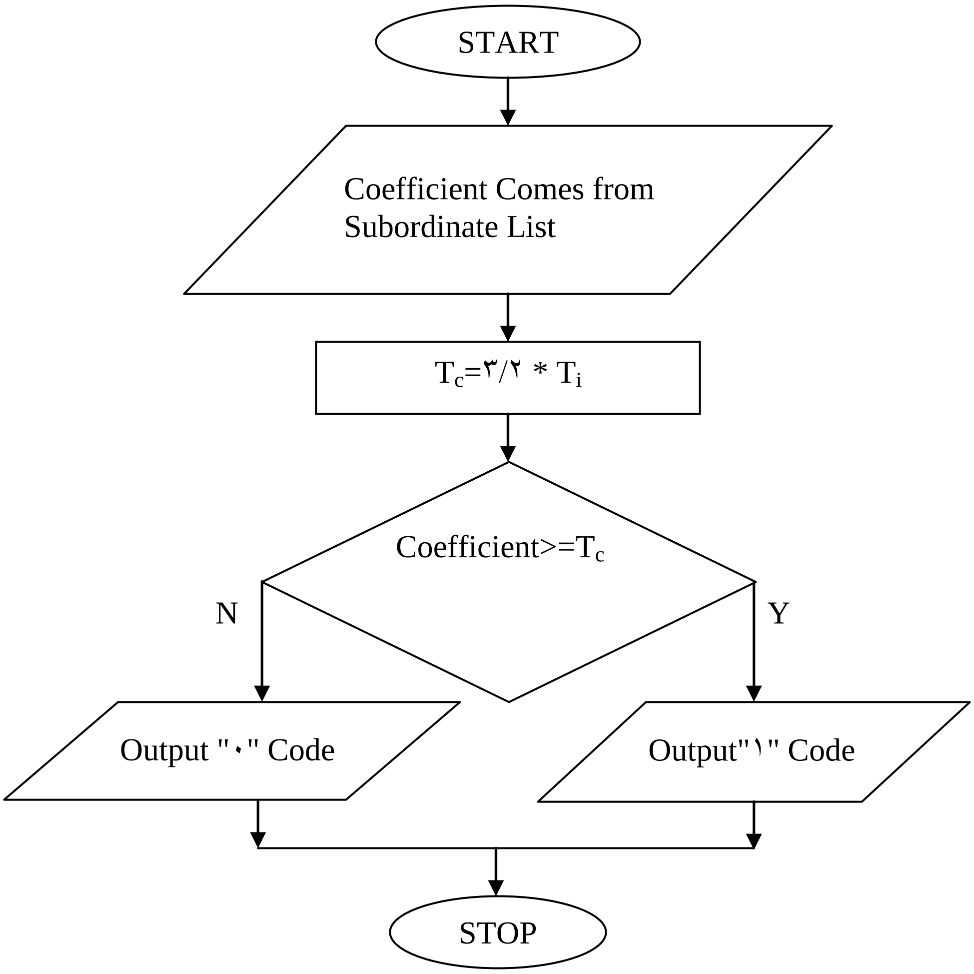


Figure(٤.٥): Flowchart of Dominant Pass

<u>Subordinate Pass</u>

The subordinate Pass(sometimes called the refinement 'refines' the value of each significant). For each coefficient in the subordinate list, the subordinate pass as showing in Figure (٤.٦) checks if there is value larger or smaller than the current threshold. Current threshold compute by

T_c=٣/٢ *T_i , where T_c refers to current threshold which compute to assign zero or one to the coefficient, T_i represents the threshold comes from dominant pass for current phase. If the coefficient is larger than the threshold a '١' is assign to the coefficient and if the coefficient is smaller than the threshold a '٠' assign to the coefficient.



Figure(٤.٦): Flowchart of ِSubordinate Pass

Filter

This procedure search for the values to be coded as significant to extract it and replaced their positions with zeros(some literatures replaced this positions with stars). This will prevent them from being coded a gain.

٤.٣ The Proposed Decompression ِAlgorithm Steps

The steps of decompression algorithm are illustrated in Figure(٤.٧).

Where:

I is a counter for the number of symbols start with zero value ,

K is a counter for blocks,

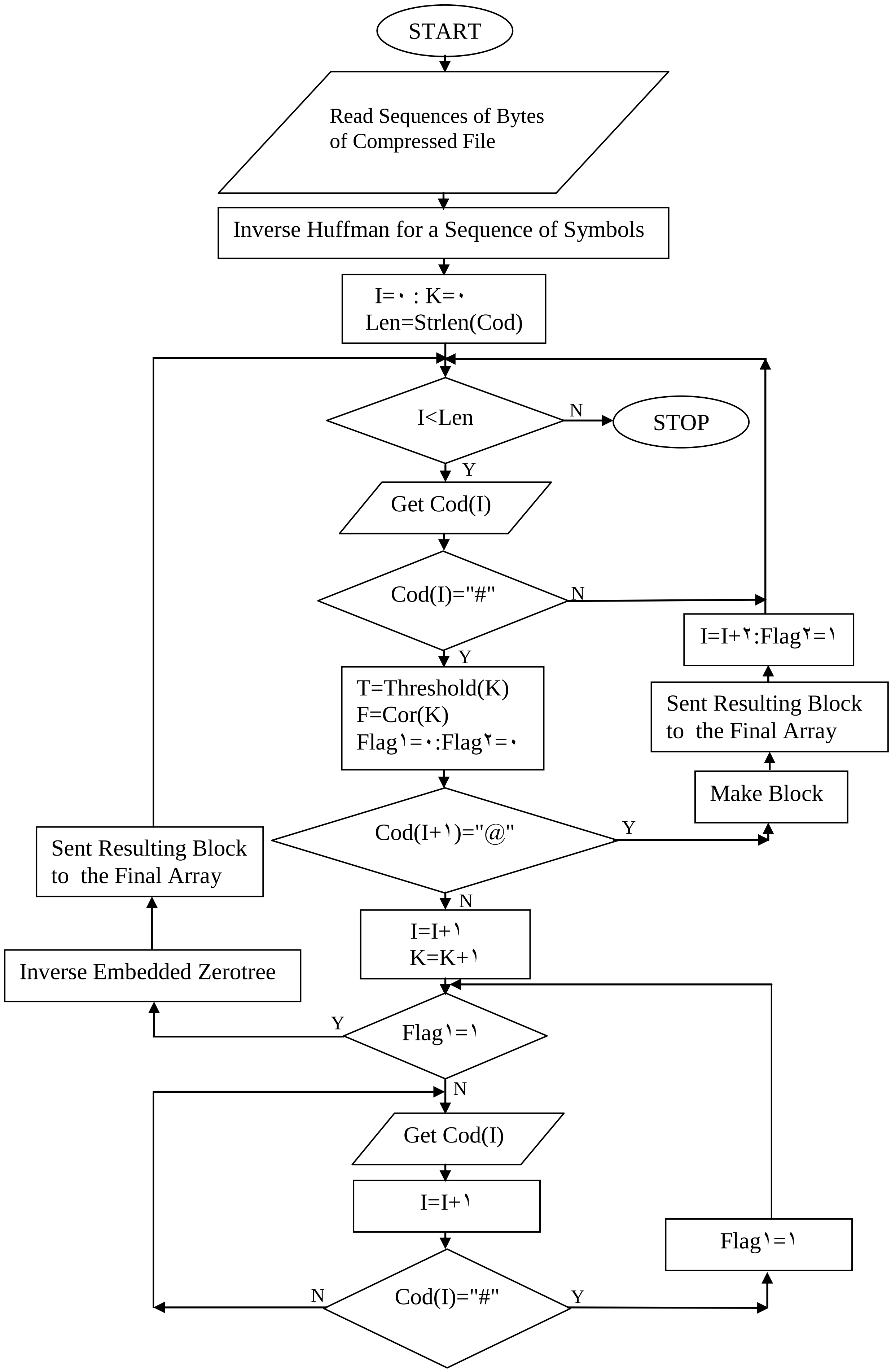
Len represent the length of string results from inverse Huffman algorithm,

Cod sequence of string results from apply inverse Huffman algorithm,

Cor array of DC's of block comes from compressed file,

Flag١ flag refers to the similarity case,

Flag٢ flag refers to new block.


START
Read Sequences of Bytes of Compressed File
Inverse Huffman for a Sequence of Symbols
I=٠ : K=٠
Len=Strlen(Cod)
I<Len
N
STOP
Y
Get Cod(I)
Cod(I)="#"
N
Y
I=I+٢:Flag٢=١
Sent Resulting Block to the Final Array
Make Block
T=Threshold(K)
F=Cor(K)
Flag١=٠:Flag٢=٠
Cod(I+١)="@"
Y
N
Sent Resulting Block to the Final Array
I=I+١
K=K+١
Inverse Embedded Zerotree
Y
Flag١=١
N
Get Cod(I)
I=I+١
Flag١=١
N
Cod(I)="#"
Y

The alg(Figure(٤.٧): Flowchart of Decompression Algorithm

Step١: Read the coming compressed file. Then the Inverse Algorithm converts arrays which are contain a numbers corresponding to the sequence of (٠,١).

Step٢: Apply Inverse Huffman Algorithm on the array result from the previous step, this resulting Cod array.

Step٣: Initial value is assigned to the counters, usually zero. Also assign length of string to Len variable.

Step٤: While counter less than length of string, loop will continued. Then read the current symbol from.

Step٥: *If* the next character is '#' (which refers to new block), go to the following steps: assign current threshold to the T variable, assign current DC to the F variable, also assign initial value to Flag١, Flag٢.

Else go to Step٤.

Step٦: If Cod(I+١) character is '@' (which refers to similarity case), go to the following steps:

- ❖ Go to function that make a new block by using DC for first element at position(٠,٠), while all other position filled with the same value which are threshold.
- ❖ Increment counter with ٢ to reach the next '#'character.
- ❖ Change value of Flag١ to ١.

Step٧: Else increment counters I,K. Specially for I counter to reach the character which followed '#' character.

Step٨: *If* Flag١ equaled to ١, in this case

- ❖ Go to the Inverse Embedded Zerotree Algorithm.
- ❖ Sent the result block to the final array.

Step٩: *Else* Read the character.

Step١٠: Increment counter of character.

Step١١: *if* the coming character is '#', in this case change the value of Flag١ to ١ . *Else* go to step ١٠.

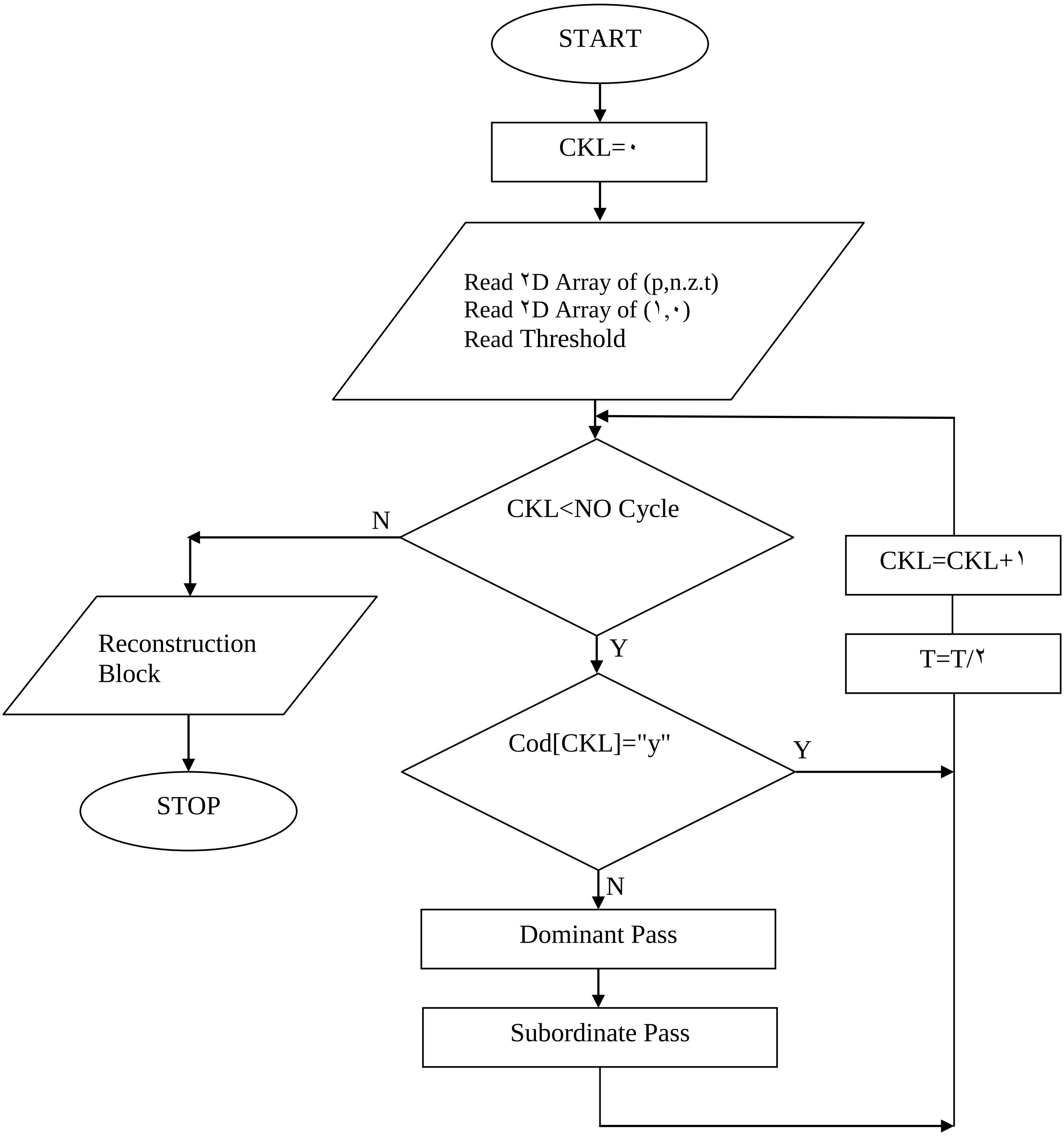
Inverse Embedded Zerotree Algorithm

The main stages of Inverse Embedded Coding is shown in Figure(٤.٨).

Where:

CKL is a counter for cycle or phases.

T is variable refers to the threshold.



Figure(٤.٨): Flowchart of Inverse Embedded Zerotree

In briefly the following steps for performing the Inverse Embedded Zerotree Algorithm:

Step١: Initial value to CKL variable, usually zero.

Step٢: Read the string array which contain p, n, a, t ; also read the another array which contain ١,٠.

Step٣: While the number of cycle still less than the final cycle(number of cycle taking from number of rows of coming array) perform the following steps. *Else* the algorithm reach to final values of block and the reconstruction block is getting, then STOP.

Step٤: *If* coming row contain just one character which are 'y', This refers to special case when there are at least one absolute value greater than zero, but in the same time this value is smaller than threshold. In this case only increment CKL counter and halving threshold.

Step٥: Dominant Pass, procedure for assign one of four symbols (p, n, z, t) to each position of default block.

Step٦: Subordinate Pass, by using default array coming from Dominant Pass procedure, and array of ١,٠ and threshold coming from Step٢ this procedure reconstructed the nearly original block following the same steps in the Subordinate Pass of compression algorithm, also the example of chapter three illustrates the decoding operations.

٤.٤ Characteristics of The Proposed Compression System Over EZW

- The proposed algorithm used eight symbols as alphabetic while **EZW** used just four symbols, in the first glance this is useless, but there a logical justification illustrated in the steps of compression/decompression algorithm.

- The proposed system treated main drop of **EZW** which is special case that shown in Figure (٤.٤), the treating was accomplished using **Test** Function.
- **EZW** compress all blocks of data of image and don't care if the coming block has the same block (similarity case), this consume additional computation, time and resulting less compression ratio. In the suggested system there are specific symbol (@) to refers to this case to avoided this disadvantage.
- **Numbers** of cycle or level for each block derived from decompressed algorithm, there are not need for store this number in compressed file as **EZW** work.

٥.١ The Experimental Results

The implementation of the proposed system on different images in size and details, this images are of BMP format with ٢٤ bits. All test have been carried out on a ١.٧ GHz Pentium ٤ with ٢٥٦ MB of main memory. The programming language was VB ver.٦.

٥.١.١ Results of The Proposed Compression System

There are a number of images used to test the suggested compression system, this images illustrated in Figure(٥.١) as well as to the reconstructed images. The results of proposed system are summarized in Table (٥.١).

Original Image١

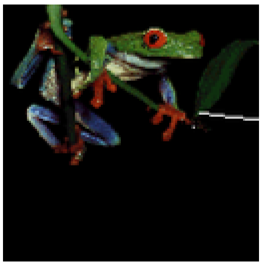
Reconstruct Image١

Original Image٢

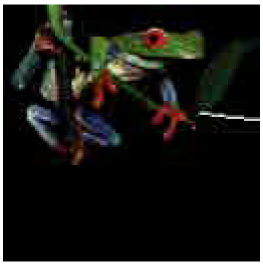
Reconstruct Image٢

Original Image

Reconstruct Image



Original Image٤



Reconstruct Image٤

Original Image٥

Reconstruct Image٥

Original Image٦

Reconstruct Image٦

Figure(٥.١): Original , Reconstructed of The Test

Table(٥.١): Results of Proposed Compression System

Source File Name	Source File Size (byte)	Compressed File Size(byte)	Compression Ratio	PSNR
Image١	٤٩١٥٢	٤٨٦٤	٩٠٪	٣٣.٦
Image٢	٩٨٣٠٤	٧٧١٠	٩٢٪	٢٧.٧٧
Image٣	٩٨٣٠٤	٧٤٦٤.٩٦	٩٢%	٢٧.١٥
Image٤	٤٩١٥٢	٣٦٩٦.٦٤	٩٣٪	٢٨.٩٦
Image٥	١٩٦٦٠٨	١١٠٥٩.٢	٩٤٪	٣٤.٥٨
Image٦	٩٨٣٠٤	٦٣١٨.٠٨	٩٤٪	٣١.٧

٥.١.٢ Comparison of The Proposed System Results with Other Systems

Its Known that JPEG which consider as most famous compression algorithm used RLE algorithm incorporates RLE, so its very nice to compare the proposed algorithm performance with RLE performance, the comparison related with encoding scheme, the results summarized in Table(٥.٢), tested images shown in Figure(٥.٢).

Table(٥.٢): Comparison between the Proposed Compression System Results and RLE Results

Compressed File Size	Source File Size(byte)	Compressed File Size(byte) by E.C.	Compressed File Size by RLE	Difference
Image٧	١٩٦٦٠٨	١٣٢٠٩.٦	٢١٦٠٦.٤	٨٣٩٦.٨
Image٨	٤٩١٥٠	٤٥٢٦.٠٨	٩٤١٠.٥٦	٤٨٨٤.٤٨
Image٩	٤٩١٥٠	٣٧٦٨.٣٢	٦٨٧١.٠٤	٣١٠٢.٧٢
Image١٠	٩٨٣٠٤	١٠٧٥٢	٢٩٥٩٣.٦	١٨٨٤١.٦

Original Image٧

Reconstruct Image٧

Original Image٨

Reconstruct Image٨

Original Image٩

Reconstruct Image٩

Original Image١٠

Reconstruct Image١٠

Figure(٥.٢): Original, Reconstructed Images of Comparison between The Proposed System Results and RLE Results

٥.٢ Conclusions

In this section there are many conclusions:

١-The suggested system highlights the coding gained by DCT and Embedded Zerotree Algorithm, this allows to address the real issues involved in image coding, which are quantization and Entropy Coding.

٢-Isolated DC's vector of blocks gives better visual quality and PSNR, with degrade the compression ratio.

٣-Many researches considered EZW algorithm "does not really compression thing, it only reorder coefficients in such a way that can be compressed very efficiently"[٢٤], while others showed "EZW algorithm which is considerably effective and important compression algorithm"[١]. The proposed system proved that Embedded Zerotree algorithm is effective compression algorithm.

٤-The proposed system treats important drop included in EZW that illustrated in Figure (٤.٤), this drop caused a stuck case in implementation of algorithm.

٥.The proposed system optimize the compression results by applying Huffman algorithm on the all result string instead of apply arithmetic algorithm for each block as in EZW

٦-Using HDCT results high compression ratio, but with greater error ratio.

٥.٣ Methods for Incrementing Compression Ratio of Proposed Algorithm

١-Using another ready tables for quantization such as $Q_{١٠}$ which results more zeros with few coefficients.

٢-Using high value for parameter R in equation (٤.١) of quantization step.

٣-Apply the proposed algorithm with only vector of DC's of blocks.

٤-Using multilevel of DCT for each block with store DC of level, this mean when we apply DCT on each block we store the DC in individual vector before apply DCT on the same block and go on.

٥-Using specific number of level for Embedded Algorithm which apply on the block, usually small to decrease the resulting coefficients.

٥.٤ Future Works

١-Used this system to compress other type of files such as movie, sound files.

٢-"One of the major disadvantages of block-based image coding techniques is block effect [١٣] ", "image quality degrades because of artifacts of block-based DCT scheme[١٤]", so postprocessing techniques are required using many filters or operators such as sobel operator.

٣-Using variable block size instead of fixed block to improve the performance and overcome on the block effect which cause the distortion.

[١] Neyre T. and Hakan S., "**Embedded Zerotree Wavelet Compression**", Eastern Mediterranean University , ٢٠٠٥.

[٢] Jonas G. and Luiz V., " **Image Processing for Computer Graphics**", Springer, ١٩٩٧.

[٣] Guy E., "**Introduction to Data Compression**", Carnegie Mellon University, ٢٠٠١.

[٤] Scott V., "**Computer Vision and Image Processing a Practical Approach Using CVIP tools**", Prentice Hall, ١٩٩٨.

[٥] David B., "**Optimisation of Still Image Compression Techniques**", University of Bath, Doctor Thesis, ١٩٩٧.

[٦] Tawfiq A., "**A Hybrid Algorithm for Image Compression**", University of Technology, Doctor Thesis, ٢٠٠٤.

[٧] Subhasis S., "**Image Compression from DCT to Wavelets: A Review**".
http://www.acm.org/crossroads/xrds٦-٣/sahaimgcoding.html

[٨] Plataniotis K. and Venetsanopoulos, "**Color Image Processing and Application**", Springer, ٢٠٠٠.

[٩] Panrony X., "**Image Compression by Wavelet Transform**", East Tennessee State University, Master Thesis, ٢٠٠١.

[١٠] Shapiro J., "**Embedded image using zerotrees of wavelet coefficient**", IEEE Trans. Signal Processing, vol. ٤١, pp٣٤٤٥-٣٤٦٢, Dec. ١٩٩٣.

[١١] Oliver E., Pascal F and Touradj E., "**Shape-Adaptive Wavelet Transform for Zerotree Coding**", In Proceedings of the Inter-national Conference on Acoustics, Speech, and Signal Processing ICASSP, Detroit, USA, May ١٩٩٥.

[١٢] Said A. and Pearlman W., "**A new Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees**", IEEE

Transactions on Circuits and Systems for Video Technology , Vol. ٦, June ١٩٩٦.

[١٣] Amir A., Moshe I. and Francois M., "**Speed Versus Quality in Low Bit-Rate Still Image Compression**", Signal Processing, ١٩٩٩.

[١٤] Debin Z. , Dapeng Z and Wen G., "**Embedded Based on Hierarchical Discrete Cosine Transform**", IEEE Transactions on Circuits and Systems for Video Technology, ١٩٩٨.

[١٥] Hongyu S. and Yi Z., "**Image Compression Techniques**", ECE٥٣٣ Digital Image Processing, ٢٠٠٣. http://www.ee.bgu.ac.il/~greg/graphics/compress.html

[١٦] Stephen G., " **The Compression Technology in Multimedia**", Wolverhampton University, UK, ١٩٩٩.

http://www.scit.wlv.ac.uk/index.html

[١٧] David S., "**Data Compression the Complete Reference**", Spring-Verlag New York, ١٩٩٨.

[١٨] Mark N., "**Arithmetic Coding + statistical Modeling=Data Compression**", Dr. Dobb's Jornal, ١٩٩١.

[١٩] Omid E., "**Document Image Compression and Analysis**". http://www.cfar.umd.edu /Publications/Thesis/Thesis-HTML.html

[٢٠] Andrew B. "**Image Compression Using the Discrete Cosine Transform**", Mathematica, PP. ٨١-٨٨, ١٩٩٤.

[٢١] Ken C. and Peter G., "**Image Compression and the Discrete Cosine Transform**", College of the Redwoods, ١٩٩١.

[٢٢] Akuri M., "**Image Compression Using Discrete Cosine Transform**", Indiana State University, Mini Thesis, ٢٠٠٤.

[٢٣] Valens C., "**EZW Coding**".
http://perso.wanadoo.fr/polyvalens/clemens/ezw

[٢٤] Suresh S., "**Hardware Acceleration of the Embedded Zerotree Wavelet Algorithm**", East Tennessee State University, Master Thesis, ٢٠٠٤.

[٢٥] Bryan E., **"A Tutorial on Modern Lossy Wavelet Image Compression: Foundations of LPEG ٢٠٠٠"**, IEE, ٢٠٠١.