THE REPUBLIC OF IRAQ
MINISTRY OF HIGHER EDUCATION
& SCIENTIFIC RESEARCH
THE UNIVERSITY OF AL-MUSTANSIRIYA
COLLEGE OF SCIENCES
COMPUTER SCIENCE DEPARTMENT

DIGITAL IMAGE COMPRESSION USING DISCRETE COSINE TRANSFORM METHOD

A THESIS
SUBMITTED TO THE COLLEGE OF SCIENCES
THE UNIVERSITY OF AL-MUSTANSIRIYA
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN COMPUTER SCIENCE

By

**HAMID SADEQ MAHDI SAOUD
AL-SULTANI**

Supervised By
Dr. JAMILA H. AL-A'MIRI

October 2006

To

my parents,

brothers,

sister,

brother's wife

&

my nieces

Teeba, Mawada

&

Wateen

ACKNOWLEDGEMENTS

I start my gratitude Allah, the owner of grace and favor who was and is still my support in preparing and achieving this thesis.

I wish to express my thanks and deep gratitude to my supervisor Dr. Jamila Harbi Al-A'miri, for invaluable advice, criticism and encouragement throughout this work.

I would like to thank the staff-member of the University of Al-Mustansiriya and the college of sciences which offered me the chance to gain the Master degree.

Also I would like to express my thanks to my dear family for their patience and support throughout the preparation of this work.

Needless to say, I am solely responsible for any errors and imperfection that still remain in this work.

Finally, thanks go to all my friends especially Muhsin Jafar.

SUPERVISOR CERTIFICATION

I certify that this thesis entitled "**Digital Video Transmission over Wireless Channels**" was prepared under my supervision at the University of Al-Mustansiriya, in partial fulfillment of the requirements for the Degree of Master of Science in Computer Sciences

Signature:
Name: Dr. Jamila Al-A'miri (supervisor)
Title: Lecturer
Date: /   /

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

Signature:
Name: Dr. Amir Sadiq Al-Malah
Title: Lecturer
Date: /   /

SUERVISOR CERTIFICATE

We, the members of examining committee certify that after reading this thesis entitled *"Digital Video Transmission over Wireless Channels"* ***and we have examined the student*** **"Hamid Sadiq Mahdi So'wood"** ***in its contents, and in what connected with it, and that in our opinion, it meets the standard of thesis for the degree of Master of Science in Computer Science.***

CONTENTS

List of Abbreviations	
Title	**Details**
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
IDCT	Inverse Discrete Cosine Transform
DWT	Discrete Wavelet Transform
PDF	Probability Density Function
PDD	Probability Density Distribution
MSE	Mean Square Error
RLE	Run Length Encoding
JPEG	Joint Photographic Experts Group
GIF	Graphs Interchange Format
CR	Compression Ratio
E_{max}	Maximum Absolute Error
MAE	Mean Absolute Error
SNR	Signal-to-Noise Ratio
PSNR	Peak Signal-to-Noise Ratio
TCP/IP	Transport Control Protocol/Internet Protocol
STP	Shielded Twisted Pair
UTP	Unshielded Twisted Pair
IR	Infrared
UHF	Ultra-High Frequency
EHF	Extremely High Frequency

List of Abbreviations

Title	Details
WLL	Wireless Local Loop
LMDS	Local Multipoint Distribution Services
MMDS	Multichannel Multipoint Distribution Services
WLANs	Wireless Local Area Networks
RF	Radio Frequency
GEO	Geosynchronous Earth-Orbiting
LEO	Low Earth-Orbiting
MEO	Middle Earth-Orbiting
OSI	Open System Interconnection
IOS	International Organization for Standardization
PDU	Protocol Data Unit
MAC	Media Access Control
CRC	Correction Redundancy Code
ATM	Asynchronous Transfer Mode
HDLC	High-level Data Link Control
LLC	Logical Link Control
DARPA	Developed Advanced Research Projects Agency
HTTP	HyperText Transfer Protocol
FTP	File Transfer Protocol
SMTP	Simple Mail Transfer Protocol
DNS	Domain Name System

List of Abbreviations

Title	**Details**
RIP	Routing Information Protocol
SNMP	Simple Network Management Protocol
UDP	User Datagram Protocol
ARP	Address Resolution Protocol
ICMP	Internet Control Message Protocol
IGMP	Internet Group Management Protocol
LAN	Local Area Network
WAN	Wide Area Network
LBJS	Like Baseline JPEG Standard

List of Figures

List of Figures

List of Figures

List of Tables

Abstract

The processing of digital images took a wide importance in the knowledge field in the last decades ago due to the rapid development in the communication techniques and the need to find and develop methods assist in enhancing and exploiting the image information. The field of digital images compression becomes an important field of digital images processing fields due to the need to exploit the available storage space as much as possible and reduce the time required to transmit the image.

In our work, we developed the ***Baseline JPEG Standard*** technique that is used in compression of images with 8-bit color depth. Basically, this scheme consists of seven operations which are the sampling, the partitioning, the transform, the quantization, the entropy coding and Huffman coding. First, the sampling process is used to reduce the size of the image and the number bits required to represent it. Next, the partitioning process is applied to the image to get (8×8) image block. Then, the discrete cosine transform is used to transform the image block data from spatial domain to frequency domain to make the data easy to process. Later, the quantization process is applied to the transformed image block to remove the redundancies in the image block. Next, the conversion process is used to convert the quantized image block from (2D) array to (1D) vector. Then, the entropy coding is applied to convert the (1D) vector to intermediate form that is easy to compress. Finally, Huffman coding is used to convert the intermediate form to a string of bits that can be stored easily in a file which can be either stored on a storage device or transmitted to another computer. To reconstruct, the compressed image, the operations mentioned above are reversed.

Finally, it must be mentioned that the program used in this work is designed by using Visual Basic version 6.0.

CHAPTER ONE

GENERAL INTRODUCTION

1.1 Introduction

Image compression is an extremely important part of modern computing. By having the ability to compress images to a fraction of their original size, valuable (and expensive) disk space can be saved. In addition, transportation of images from one computer to another becomes easier and less time consuming (which is why compression has played such an important role in the development of the internet) [1].

In order to be useful, a compression algorithm has a corresponding decompression algorithm that, given the compressed file, reproduces the original file. There have been many types of compression algorithms developed. These algorithms fall into two broad types, lossless algorithms and lossy algorithms. A lossless algorithm reproduces the original exactly. A lossy algorithm, as its name implies, loses some data. Data loss may be unacceptable in many applications [29].

The JPEG image compression algorithm provides a very effective way to compress images with minimal loss in quality (lossy image compression method). Although the actual implementation of the JPEG algorithm is more difficult than other image formats and the compression of images is expensive computationally, the high compression ratios that can be routinely attained using the JPEG algorithm easily compensates for the amount of time spent implementing the algorithm and compressing an image [1].

Uncompressed multimedia (graphics, audio and video) data requires considerable storage capacity and transmission bandwidth. Despite rapid progress in mass-storage density, processor speeds, and digital communication system performance, demand for data storage capacity and data-transmission bandwidth continues to outstrip the capabilities of available technologies. The recent growth of data intensive multimedia-based web applications have not only sustained the need for more efficient ways to encode signals and images but have made compression of such signals central to storage and communication technology [21].

Image compression has applications in transmission and storage of information. ***Image transmission applications*** are in broadcast television, remote sensing via satellite, military communications via aircraft, radar and sonar, teleconferencing, computer communications. ***Image storage applications*** are in education and business, documentation, medical image that arises in computer tomography, magnetic resonance imaging and digital radiology, motion pictures, satellite images weather maps, geological surveys and so on [13].

1.2 Review

1. In 1998 Sami I. and et. al. [15] introduced an image compression and transmission system for battlefield networks. The system is based on network-conscious image compression. They combined network-conscious image compression with an embedded focusing feature to provide a system that can be used in battlefield scenarios such as telemedicine or intelligence gathering.
2. In 1999 Sheila S. Hemami [11] designed Resynchronizing variable-length codes (RVLCs) to transmit images over imperfect channels suffering bit errors or packet losses without channel coding for the image data, or with less channel coding than would be required if the encoded image data could tolerate no bit errors.
3. In 2000 Vladimir C. and et. al. [6] researched image compression methods for storage and transmission of digital images at the Minnesota Department of Transportation (Mn/DOT) by choosing a commercial software package for

image compression (MrSID) to improve utilization of storage and transmission resources and a multi-resolution browsing capability.

4. In 2000 Michael J. Gilbert [10] described the application of image compression and networking techniques to the transmission of text / graphics and image data over bandlimitted and lossy links. He applied application-independent techniques to the acceleration of the delivery of World Wide Web (WWW) pages over modem and wireless links. He also illustrated Application-specific techniques using a web-based VLSI layout viewer.
5. In 2001 Hang Z. [30] built a model to simulate the real-time transporting of MPEG2 video streams over ATM networks. He performed extensive simulation experiments in a multiple hops ATM network using constant bit rate MPEG2 video streams produced by the software encoder.
6. In 2003 Rakshith K. [17] discussed the DCT based JPEG compression and DWT based JPEG2000 compression technique for wireless applications and also the adaptations of MPEG-4 video compression for various types of wireless networks.
7. In 2004 Xin B. and et. al. [2] proposed a segmentation-based multilayer (SML) coding scheme for lossless medical image compression that provided efficient compression for various medical imaging data and offer potential advantages in content-based medical image retrieval and semantic progressive transmission in telemedicine.
8. In 2005 Tony L. and Pengwei H. [19] presented a compound image compression algorithm for real-time applications of computer screen image transmission which is called shape primitive extraction and coding (SPEC).
9. In 2005 Georgiy P. and et. al. [23] developed an image sensor network platform for testing transmission of images over ZigBee networks that support multi-hopping.

1.3 Aim of thesis

Like many anterior works, the purpose of this thesis is to study and develop JPEG compression techniques which are able to reach higher compression ratios and high quality of the reconstructed image just like the ones which attained before, both for lossy and lossless coding.

The reconstructed file, i.e. compressed image is used to be transmitted to another PC by using Bluetooth.

1.4 Thesis Organization

In addition to the current chapter, this thesis includes another four chapters:

Chapter Two: in this chapter, the lossy and lossless compression techniques are explained. The JPEG compression technique with its types has also been explained. Then discrete cosine transform (DCT), quantization process, run length encoding (RLE), Huffman coding and compression measurements have been clarified.

Chapter Three: The topic of this chapter is data transmission. The types of transmission and transmission media types have been explained. Then, Internet transmission models with their layers have also been explained.

Chapter Four: In this chapter, a developed standard of baseline JPEG mode is introduced with the algorithms used in this work and the results obtained after using these algorithms.

Chapter Five: conclusions have been drawn and future works have been outlined.

CHAPTER TWO

COMPRESSION TECHNIQUES

2.1 Image Compression

Image files, especially raster images tend to be very large. It can be useful or necessary to compress them for ease of storage or delivery. However, while compression can save space or assist delivery, it can slow or delay the opening of the image, since it must be decompressed when displayed. Some forms of compression will also compromise the quality of the image. Today, compression has made a great impact on the storing of large volume of image data. Even hardware and software for compression and decompression are increasingly being made part of a computer platform [27].

There is an extrusive relation between the efficiency of the compression technique and its algorithm complexity. The more efficient the compression technique, the more complicated the algorithm will be and thus, requires more computational resources or more time to decompress. This tends to affect the speed. Speed is not so much of an importance to still images but weighs a lot in motion-pictures [12].

2.2 System Model

A typical image compression system is illustrated in figure (2-1). The original digital image is usually transformed into another domain, where it is highly de-correlated by the transform. This de-correlation concentrates the important image information into a more impact form. The compressor (encoder) then removes the redundancy in the transformed image and stores it into a compressed file or data stream. The decompressor (decoder) reverses this process to produce the recovered image. The recovered image may have lost some information, due to the compression, and may have an error or distortion compared to the original image [3].

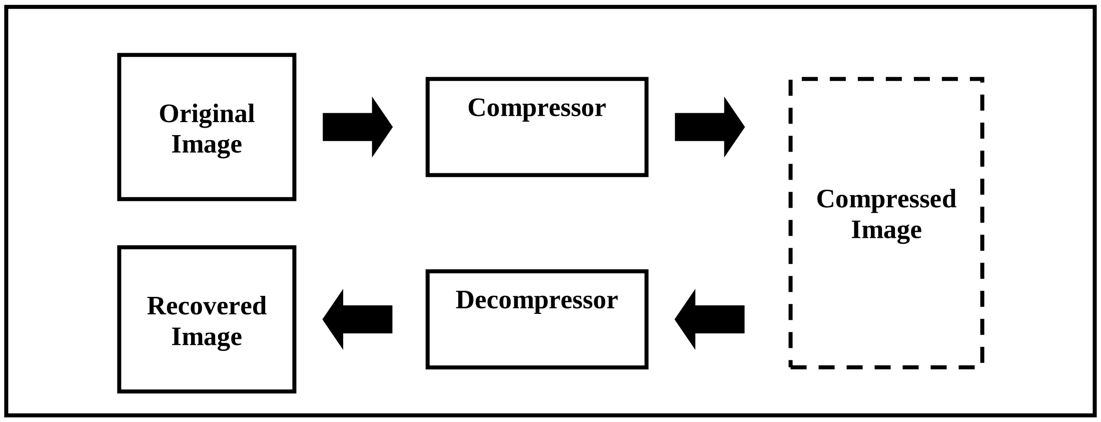


Figure (2-1): A typical image compression system [3]

2.2.1 The compressor

There are three basic parts to an image compressor, as shown in figure (2-2):

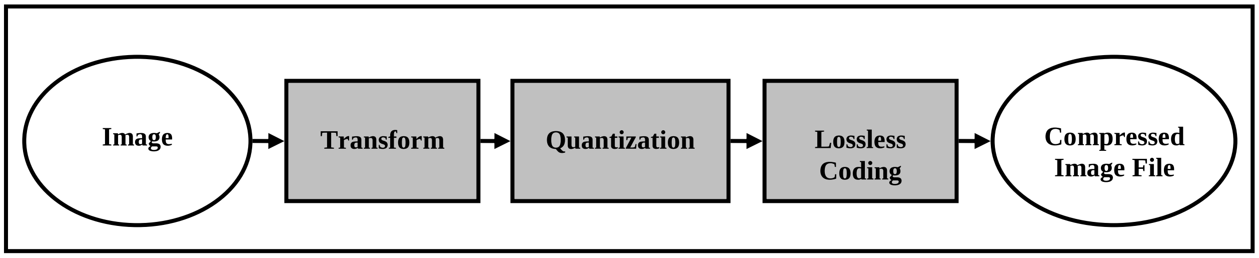


Figure (2-2): Block diagram of general image compressor [3]

I. Transforms

The first stage in the compressor is the transform. This stage is necessary to convert the image data from spatial domain into frequency domain. This transform converts the image data into a form that can be easily compressed. There are two basic types of transforms that can be used in image compression:

1. Discrete Cosine Transform (DCT) [28]

The discrete cosine transform (DCT) is a technique for converting a signal into elementary frequency components. The DCT was developed by Ahmed et al. (1974). The DCT is a close relative of the discrete Fourier transform (DFT). Its application to image compression was pioneered by Chen and Pratt in 1984.

•The One-Dimensional Discrete Cosine Transform

The discrete cosine transform of a list of ***n*** real numbers ***s(x)***, ***x***=0… ***n-1***, is the list of length n given by:

$$S(u) = \left(\frac{2}{n}\right)^{\frac{-1}{2}} C(u) \sum_{x=0}^{n-1} s(x) \cos\left(\frac{(2x+1)u\pi}{2n}\right) \qquad \text{for } \boldsymbol{u}=0\ldots \boldsymbol{n-1} \qquad (2.1)$$

where $C(u) = \begin{cases} 2^{\frac{-1}{2}} & \text{for } u = 0 \\ 1 & \text{otherwise} \end{cases}$

Each element of the transformed list ***S (u)*** is the inner product of the input list ***s(x)*** and a ***basis vector***. The constant factors are chosen so that the basis vectors are orthogonal and normalized.

The list ***s(x)*** can be recovered from its transform ***S (u)*** by applying the inverse discrete cosine transform (IDCT):

$$s(x) = \left(\frac{2}{n}\right)^{\frac{-1}{2}} \sum_{u=0}^{n-1} C(u) S(u) \cos\left(\frac{(2x+1)u\pi}{2n}\right) \qquad \text{for } \boldsymbol{x}=0\ldots \boldsymbol{n-1} \qquad (2.2)$$

where $C(u) = \begin{cases} 2^{\frac{-1}{2}} & \text{for } u = 0 \\ 1 & \text{otherwise} \end{cases}$

Equation (2.2) expresses ***s*** as a linear combination of the basis vectors. The coefficients are the elements of the transform ***S***, which may be regarded as reflecting the amount of each frequency present in the input ***s***.

•The Two-Dimensional Discrete Cosine Transform

The one-dimensional DCT is useful in processing one-dimensional signals such as speech waveforms. For analysis of two-dimensional (2-D) signals such as images, a (2-D) version of the DCT is needed. For an ***n×m*** matrix ***s***, the (2-D) of DCT is computed in a simple way:

The (1-D) of DCT is applied to each row of *s* and then to each column of the result. Thus, the transform of ***s*** is given by:

$$S(u,v) = (\frac{2}{n})^{\frac{-1}{2}} C(u)C(v)\sum_{x=0}^{n-1}\sum_{y=0}^{m-1} s(x,y)\cos(\frac{(2x+1)u\pi}{2n})\cos(\frac{(2x+1)v\pi}{2m}) \text{ for } \boldsymbol{u}=0\ldots \boldsymbol{n-1},\ \boldsymbol{v}=0\ldots \boldsymbol{m-1} \qquad (2.3)$$

where $C(u),C(v) = \begin{cases} 2^{\frac{-1}{2}} & \text{for } u,v=0 \\ 1 & \text{otherwise} \end{cases}$

Since the (2-D) of DCT can be computed by applying (1-D) transforms separately to the rows and columns, then the (2-D) of DCT is ***separable*** in the two dimensions.

As in the one-dimensional case, each element ***S*(*u*,*v*)** of the transform is the inner product of the input and a basis function(vector), but in this case, the basis functions are ***n*×*m*** matrices. Each two-dimensional basis matrix is the outer product of two of the one-dimensional basis vectors.

The inverse discrete cosine transform (IDCT) for the two-dimensional DCT is:

$$s(x,y) = (\frac{2}{n})^{\frac{-1}{2}} \sum_{u=0}^{n-1}\sum_{v=0}^{m-1} C(u)C(v)S(u,v)\cos(\frac{(2x+1)u\pi}{2n})\cos(\frac{(2x+1)v\pi}{2m}) \text{ for } \boldsymbol{x}=0\ldots \boldsymbol{n-1},\ \boldsymbol{y}=0\ldots \boldsymbol{m-1} \qquad (2.4)$$

where $C(u),C(v) = \begin{cases} 2^{\frac{-1}{2}} & \text{for } u,v=0 \\ 1 & \text{otherwise} \end{cases}$

Figure (2-3) illustrates an (8×8) array processed by applying the (2-D) DCT:

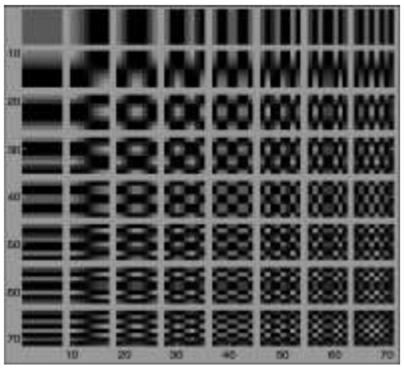


Figure (2-3): The (8×8) array of basis images for the (2-D) DCT [26]

2. Wavelet Transform

Wavelets are functions defined over a finite interval. The basic idea of the wavelet transform is to represent an arbitrary function ***f(x)*** as a linear combination of a set of such wavelets or basis functions. These basis functions are obtained from a single prototype wavelet called the mother wavelet by dilations (scaling) and translations (shifts).

The purpose of wavelet transform is to change the data from time-space domain to time-frequency domain which makes better compression results [29].

Wavelets are signals, which are local in time and generally have an irregular shape. The name wavelet comes from the fact that they integrate to zero; they wave up and down across the axis. One ideal property in wavelet is orthogonality which ensures that data is not over represented. A signal can be decomposed into many shifted and scaled representations of the original mother wavelet. Wavelets have the great advantage of being able to separate the fine details in a signal. Very small wavelets can be used to isolate very fine details in a signal, while very large wavelets can identify coarse details [16].

In image compression, sampled data that are discrete in time are dealt with. The discrete representation of time and frequency is called the discrete wavelet transform (DWT) [29].

DWT is based on sub-band coding and found to yield a fast computation of wavelet transform. It is easy to implement and reduce the computation time and resources required [16].

The discrete wavelet transform usually is implemented by using a hierarchical filter structure. It is applied to image blocks generated by the preprocessor [29]. Filters are one of the most widely used signal processing functions [16].

A simple way to perform wavelet decomposition on an image is to alternate between operations on the rows and columns. First, wavelet decomposition is performed on the pixel values in each row of the image. Then, wavelet decomposition is performed to each column of the previous result. The process is repeated to perform the complete wavelet decomposition [29].

A low pass and a high pass filter at each decomposition stage, is commonly used in image compression [16].

Figure (2-4) illustrates each step of the 2-dimensional wavelet decomposition. The shaded parts are wavelet coefficients [29].

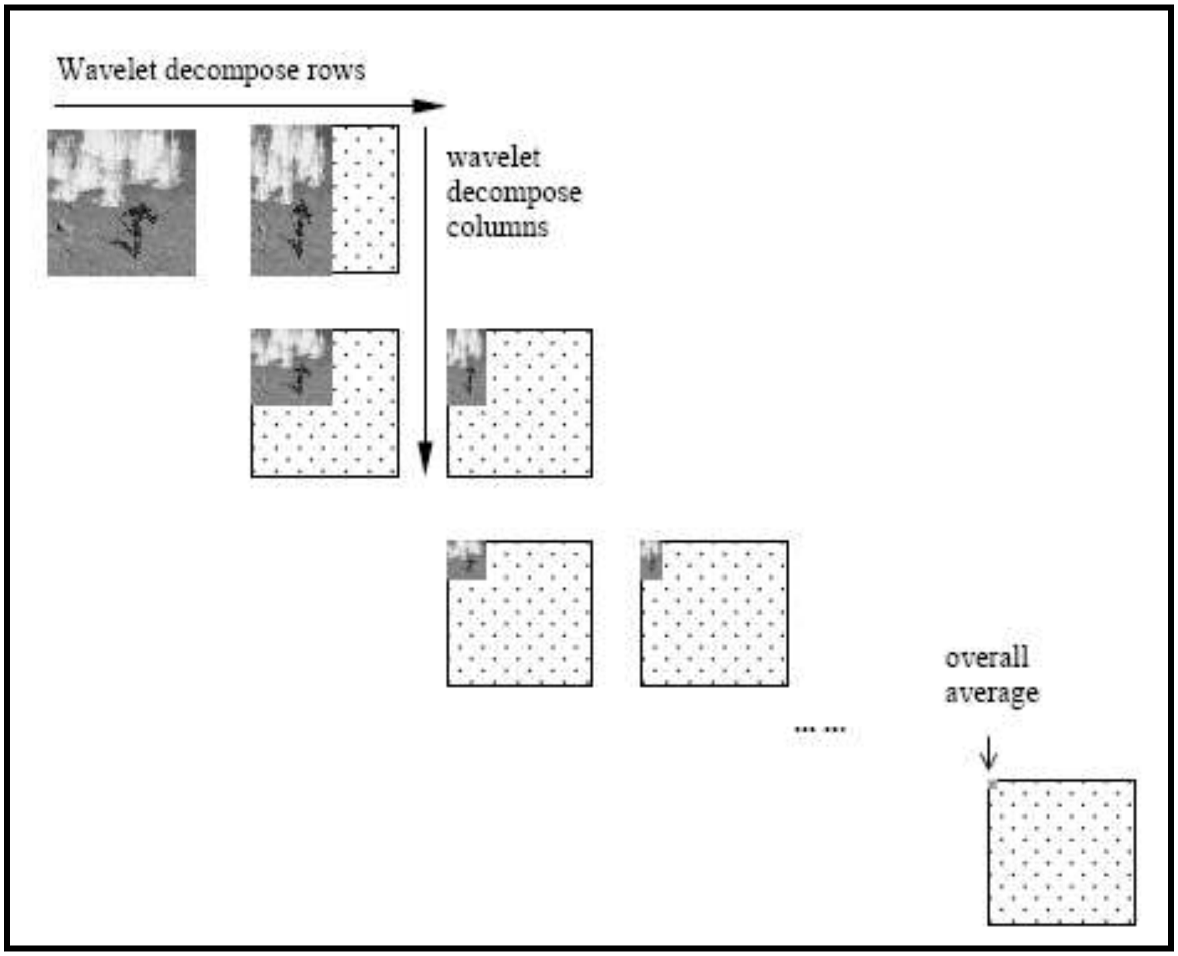


Figure (2- 4): The 2D wavelet decomposition of an image [29]

П. Quantization

Quantization refers to the process of approximating the continuous set of values in the image data with a finite (preferably small) set of values [18]. Quantization is a necessary component in lossy coding and has a direct impact on the bit rate and the distortion of reconstructed images or videos [26]. The input to a quantizer is the original data, and the output is always one among a finite number of levels. The quantizer is a function whose set of output values are discrete, and usually finite. Obviously, this is a process of approximation, and a good quantizer is one which represents the original signal with minimum loss or distortion. There are three types of quantization:

1. Linear Quantization [3]

Linear quantization is the most basic form of quantization. The transform coefficients are divided by a quantization step and the result is converted to an integer, by truncation of the decimal point as illustrated in equation (2.6):

$$c_q = Integer\left(\frac{c_i}{q_i}\right) \tag{2.6}$$

where, $\boldsymbol{q_i}$ is the quantization step,

$\boldsymbol{c_i}$ is the transform coefficient,

and $\boldsymbol{c_q}$ is the integer quantized coefficient.

The transform data is limited based on the 8 bit/pixel intensities of standard images. This allows a quantization step to be chosen, which limits the number of quantized states available, hence compressing the coefficients to a desired number of bits. However, it is not possible to control compression in this way, since a real system losslessly codes the quantized coefficients and this operation is not well defined.

It can be seen that equation (2.6) is not ideal and it can be improved to a more effective form,

$$c_q = Round\ to\ nearest\ Integer\left(\frac{c_i}{q_i}\right) \tag{2.7}$$

This means that the transform coefficients are mapped to the quantization values that cause the least error. This effectively forms quantization boundaries, inside which all values are mapped to the same quantization step.

2. Uniform Quantization [16]

The idea behind this quantization method is to release the range of the input set of data values into positive valued integers of, often, less width range. In this quantization method, the minimum MIN and the maximum MAX values of the input data must be specified, and then the input image of data is divided into a number of equidistant bins. The uniform quantization algorithm uses equally spaced bins it can, easily, found by equation (2.8):

$$\mathrm{F_Q}(x,\mathrm{y}) = \mathrm{Roundoff}\left(\frac{\mathrm{f(x,\ y)\text{-}\ MIN}}{\mathrm{MAX\text{-}\ MIN}}(G-1)\right) \tag{2.8}$$

where, Roundoff{x} approximates the enclosed x-value to nearest integer,

f(x, y) is the original input data,

F_Q(x, y) their quantized value,

and G is the desired number of quantized level.

3. Non-Uniform Quantization [3]

Non-Uniform quantization uses a more complex model to decide what each quantization value should be. The quantizer is implemented by defining 'data bins' and all transform coefficients inside a bin are quantized to the same value.

The process of choosing the appropriate bin size is known as Lloyd-Max quantization, which works by trying to minimize the error between the quantized coefficients and the original coefficients for a fixed number of quantization values. The bins are calculated by using the probability density function (PDF) of the data as this indicates where the 'bin density' should increase.

There are two methods that can be used to calculate the bin position using Lloyd-Max quantization:

1. Obtain the true PDF from the data and derive the bins directly from this.
2. Iterate the bin positions, and slowly converge to a stable set of bins.

Since most image compression applications can not form an accurate PDF function for the data (only an experimental histogram), the second method is often preferred.

There are several steps in Lloyd-Max quantization:

1. Provide a set of initial 'seed' quantization values, such as the values produced by linear quantization.
2. Find the quantization values, which produce the smallest error in the coefficient after quantization, for each coefficient in the data set.
3. Partition the coefficients by grouping them with coefficient that used the same quantization value.
4. Average all the coefficients in each partition to produce a new quantization value for that partition.
5. Repeat 2-4, until the partitions become stable.
6. Find the coefficient boundary, which marks each partition and mark these as the bin edges.

Lloyd-Max quantization produces a set of quantization bins, which minimizes the error in each quantized coefficients. Unfortunately, it does this at the expense of a reduced data entropy and as a result quantized coefficients are not substantially improved by the application of a lossless coding stage (since the quantized coefficients already have little statistical redundancy, which can be exploited by lossless coding). This can be a benefit, since lossless coding adds to computation, but it has been shown that linear quantization with a lossless coding stage is more efficient in rate distortion terms, than Lloyd-Max quantization.

4. Embedded Quantization [3]

Embedded quantization is a method used to quantize coefficients so that they can be decoded to any error. This method of quantization is common in wavelet compressors, where it is most suited.

The basis of embedded quantization is to apply many steps to a single coefficient and produce a stream of symbols to describe that coefficient. If the stream is stopped at any point, the data received should describe the coefficient as accurately as possible.

Consider a set of quantization steps

$$q \subset \{q_0, q_1, q_2, \ldots\, q_n\} \tag{2.9}$$

where $q_0 > q_1 > q_2 > \ldots > q_n$.

These quantization steps can be recursively applied to a particular transform coefficient to accurately describe it, as shown in equation (2.10).

$$c'_{i+1} = c'_i - q_i . R\,\mathrm{int}\left(\frac{c'_i}{q_i}\right) \tag{2.10}$$

where, $\boldsymbol{c_i'}$ is residual coefficient,

and $c_0' = c_0$.

It can be seen that iterating equation (2.10) produces two pieces of information, a quantized coefficient that can be used to refine the coefficients representation and a residual coefficient which can be quantized further. It is important to space $\boldsymbol{q_i}$ and $\boldsymbol{q_{i+1}}$ apart, so that they are different enough to produce quantized coefficients but similar enough to give a good resolution.

Generally in wavelet compression $2\boldsymbol{q_{i+1}} = \boldsymbol{q_i}$ and $\boldsymbol{q_n} = 1$, so that a binary masking operation can be used for quantization. This is very easy and fast to implement on most processors and this usually compensates for the more complex quantization method. Another method adopted by wavelet compressors is to only pass the sign of the quantized value once. In the above scheme this means that the absolute value of the coefficient is used in equation (2.10) and the sign is only transmitted once for each coefficient.

III. Lossless Coding

Lossless coding aims to reduce the redundancy of a set of data, by exploiting its statistics. Theoretically this coding method should compress a data source without introducing any new errors into the data. There are two methods that can be used for lossless coding [3]:

• Entropy Coding: it is possible to reduce the information required to store data to a theoretical minimum, by exploiting the 'blind' statistics of the data, without considering the order in which it is received. This is usually achieved with a Huffman coder, but arithmetic coders can also be useful.

•Pre-processing: the data can be processed before being compressed by an entropy coder to improve the overall compression.

1. Entropy Coding

This is the most effective method of lossless coding and is nearly always present in image compression. Entropy is the average minimum number of bits that a data symbol stream can be compressed into, when each symbol is considered in isolation, based on its statistics. It can be calculated directly (2.11), but it is sometimes not possible to reach the theoretical minimum due to the implementation of the entropy coder.

$$Entropy = \sum_{x} PDD(x)\log_2\left(PDD(x)\right) \qquad (2.11)$$

where, ***PDD(x)*** is the probability density distribution of symbol ***x***.

There are two different approaches to entropy coding, Huffman coding and arithmetic coding. Huffman compression is the most common and the most robust method, but it can not compress data to less than one bit/symbol. Arithmetic coding is less controllable and does not compress well at higher bits/symbol, but it can reduce its entropy below one bit/symbol. For this reason, arithmetic coders are not often used, unless the average entropy of the source is expected to drop below one bit/symbol [3].

I. Huffman Coding

Huffman coding, developed by D.A. Huffman, is a classical data compression technique. It has been used in various compression applications, including image compression. It uses the statistical property of characters in the source stream and then produces respective codes for these characters. These codes are of variable code length using an integral number of bits. The codes for characters having a higher frequency of occurrence are shorter than those codes for characters having lower frequency. This simple idea causes a reduction in the average code length, and thus the overall size of compressed data is smaller than the original. Huffman coding is based on building a ***binary tree*** that holds all characters in the source at its ***leaf nodes***, and with their corresponding characters' probabilities at the side [29].

The tree is built by going through the following steps [26]:

1. Arrange all source symbols in such a way that their occurrence probabilities are in a non-increasing order.
2. Combine the two least probable source symbols:

• Form a new source symbol with a probability equal to the sum of the probabilities of the two least probable symbols.

• Assign a binary 0 and a binary 1 to the two least probable symbols.

3. Repeat until the newly created auxiliary source alphabet contains only one source symbol.
4. Start from the source symbol in the last auxiliary source alphabet and trace back to each source symbol in the original source alphabet to find the corresponding codewords.

Example [26]:

Consider a source alphabet whose six source symbols and their occurrence probabilities are listed in table (2-1) and figure (2-5) demonstrates the Huffman coding procedure applied. In the example, among the two least probable source symbols encountered at each step binary 0 is assigned to the top symbol and binary 1 to the bottom symbol.

Table (2-1): Source Alphabet and Huffman Codes

Source Symbol	**Occurrence Probability**	**Codeword Assigned**	**Length of Codeword**
S1	**0.3**	**00**	**2**
S2	**0.1**	**101**	**3**
S3	**0.2**	**11**	**2**
S4	**0.05**	**1001**	**4**
S5	**0.1**	**1000**	**4**
S6	**0.25**	**01**	**2**

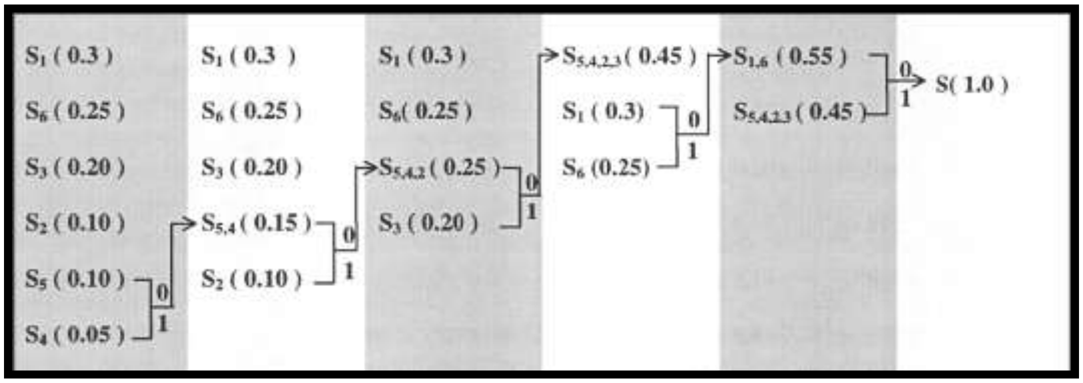


Figure (2-5): Huffman coding procedure [26]

Π. Arithmetic Coding

Arithmetic coding is also a kind of statistical coding algorithm similar to Huffman coding. However, it uses a different approach to utilize symbol probabilities, and performs better than Huffman coding. In Huffman coding, optimal codeword length is obtained when the symbol probabilities are of the form $(1/2)^x$, where ***x*** is an integer. This is because Huffman coding assigns code with an integral number of bits. This form of symbol probabilities is rare in practice. Arithmetic coding is a statistical coding method that solves this problem. The code form is not restricted to an integral number of bits. It can assign a code as a fraction of a bit. Therefore, when the symbol probabilities are more arbitrary, arithmetic coding has a better compression ratio than Huffman coding. In brief, this is can be considered as grouping input symbols and coding them into one long code. Therefore, different symbols can share a bit from the long code [29].

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 1. 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 encoding a symbol by the coder requires the ***range*** to be reduced in the following way [3]:

$$\textit{Range} = \textit{Upper Limit} - \textit{Lower Limit} \tag{2.12}$$

$$\textit{New Upper Limit} = \textit{Lower Limit} + \textit{Range} * P_{HIGH}\,(\textit{Symbol}) \tag{2.13}$$

$$\textit{New Lower Limit} = \textit{Lower Limit} + \textit{Range} * P_{LOW}\,(\textit{Symbol}) \tag{2.14}$$

where, P_{HIGH} ***(Symbol)*** is the higher probability range of the symbol,

and P_{LOW} ***(Symbol)*** is the lower probability range of the symbol.

Although arithmetic coding is more powerful than Huffman coding in compression ratio, arithmetic coding requires more computational power and memory. Huffman coding is more attractive than arithmetic coding when simplicity is the major concern [29].

2. Preprocessing

There are two basic methods of pre-processing for data source compression:

• Delta Coding: this removes linear correlation in a data source from symbol to symbol.

• Run Length Coding: this also removes linear correlation in a data source, but is more suitable, when large runs of the same symbol occur in the data stream.

If correlation does not exist in the data sources then applying pre-processing can increase the information required to store the data, reducing the efficiency of the system [3].

I. Delta Coding [3].

Delta coding is the simplest form of pre-processing and it is described by equation (2.15):

$$\delta c_i = c_i - c_{i-1} \tag{2.15}$$

where, c_i is the data symbol at i,

and δc_i is the data coded symbol.

The original data item in the source is considered to be unchanged since no previous symbol exists. It is common to split the data source into sections, which are highly correlated. In image compression this usually means taking separate 'lines' of coefficient symbols out of the image.

The effect of delta coding is to give a set of source symbols that have a greater dynamic range, but the entropy of the data source is reduced by the process. In 8 bit grayscale images the intensity can vary from (0) to (255), but after delta coding the range of the symbols can be (-255) to (255), but very few symbols have these extreme ranges so the entropy of the encoded source is lower than the original.

П. Run Length Coding

The term ***run*** is used to indicate the repetition of a symbol, while the term ***run-length*** is used to represent the number of repeated symbols, in other words, the number of consecutive symbols of the same value [26].

Probably the simplest coding scheme that takes advantage of the context is run-length coding. Although there are many variants, the basic idea is to identify strings of adjacent messages of equal value and replace them with a single occurrence along with a count. For example, the message sequence ***acccbbaaabb*** could be transformed to (a, 1), (c, 3), (b, 2), (a, 3), (b, 2). Once transformed, a probability coder (***e.g.***, Huffman coder) can be used to code both the message values and the counts. It is typically important to probability code the run-lengths since short lengths (***e.g.***, 1 and 2) are likely to be much more common than long lengths (***e.g.***, 1356).

There is a form of run-length coding that is used to compress the linear-order. It code using as a sequence of (skip, value) pairs, where skip is the number of zeros before a value, and value is the value. For example, the sequence [4, 3, 0, 0, 1, 0, 0, 0, 1, 0, 0 ,0] is represented as

[(0,4),(0,3),(2,1),(3,1),(0,0)]. This sequence is then compressed using Huffman coding or another coding scheme [5].

2.2.2 The decompressor

The decompressor reverses the order of operations that the compressor does. It begins with the inverse lossless coding step and return back to the inverse transform step through the dequantization step. This is illustrated in figure (2-6).

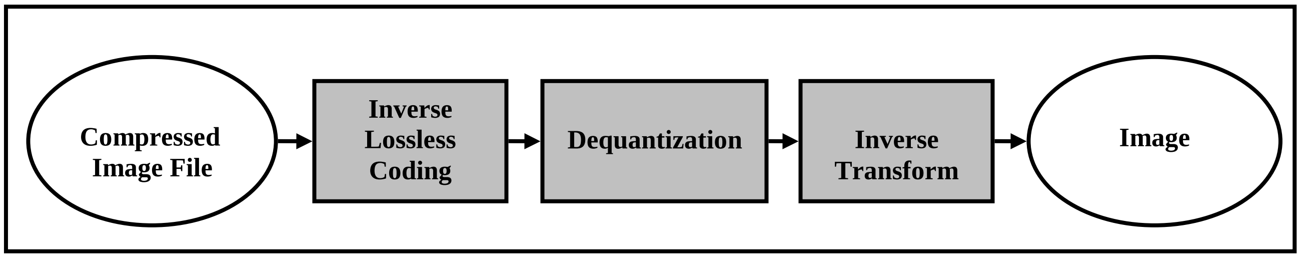


Figure (2-6): Block diagram of general image decompressor [3]

2.3 Types of Compression Techniques

Compression of digital data is based on various computational algorithms, which can be implemented either in software or in hardware. Compression techniques are classified into two categories [8]:

2.3.1 Lossless Techniques

Lossless techniques are capable of recovering the original representation perfectly [8]. The decompressed image must be identical to the original. No distortion is admitted at all, hence no distortion measure is needed. The objective is only to minimize the bit-rate. The compression ratio is usually modest (two is a common value for 256–level gray–scale images) [20].

2.3.2 Lossy Techniques

Lossy compression techniques transform and simplify the image information in a way that gives much larger reductions in file size than lossless techniques. A lossy compression permanently disposes of information is, i.e. it is irreversible [27].

The lossy techniques provide higher compression ratios, and, therefore, they are more often applied to image and video compression than lossless techniques [8].

Computers can use a distortion measure that must be defined between the original image and the decompressed one. The Mean Square Error (MSE) is often used. The compression problem is defined as a constrained minimization one. It is either the minimization of the bit-rate when the acceptable distortion is given or the minimization of the distortion when the available bit-rate is known [20].

2.4 Comparison of Compression Properties [16]

The following table shows the property comparison of lossy and lossless types of compression methods:

Table (2-2): Comparison between lossy and lossless compression methods

Lossy	**Lossless**
Degrades the quality of the image.	Preserves image quality perfectly.
Provides a significant amount of compression ratio.	Provides a low compression ratio.
Lossy operation is irreversible.	Lossless operation is reversible.
Lossy techniques contain a quantization stage.	Lossless techniques do not contain a quantization stage.
Lossy techniques such as: vector quantization, fractal image	Lossless techniques such as: RLE coding, Huffman coding,

compression, discrete cosine transform (DCT), Wavelet …etc.	arithmetic coding, Lempel-Zive algorithm…etc.
Joint Photographic Experts Group (JPEG) is an example of lossy techniques.	Graphs Interchange Format (GIF) is an example of lossless techniques.
Lossy used in applications such as: broadcast television, video conferencing and facsimile transmission…etc.	Lossless used in applications such as: archival of medical or business document…etc.

2.5 JPEG Technique

Joint Photographic Experts Group (JPEG) is becoming increasingly popular and has allowed digital image applications involving storage or transmission across different computing environments to be more widespread [24].

JPEG provides a compression method that is capable of compressing color or gray scale images of real world subject such as photograph, still video or any complex graphics that resemble nature subjects. But JPEG does not compress so well on lettering, sample cartoons or line drawings [12]. JPEG does not handle bilevel (black and white) images, at least not well. It does not handle color mapped images either; those have to be pre-expanded into an unmapped full-color representation. Images with many sudden jumps in color value will not compress well [9].

Although JPEG was only used for still images, developments have seen that it is slowly branching into compression for motion-picture. JPEG does not operate on a single algorithm, instead it has been built up by varies compression techniques which serves as its tools. JPEG allows varies configuration of these tools depending on the needs of the user.

There are two schemes of compression in JPEG. One is a lossy scheme which means compressed image when decompressed back, isn't quite the same as the one first started. The other is a lossless scheme. This scheme does not lose any of the image data when the compressed image is decompressed back. That is the image looks exactly the same as the first one. But the compression achieve by of this lossless scheme isn't quite as high as lossy, usually about 2:1 [12].

JPEG was developed specifically to discard information that the human eye cannot see. Slight changes in color are not perceived well by the human eye, while slight changes in intensity are perceived. Due to this fact it can be seen that JPEG does not compress gray scale images as well as colored, usually about 5:1, whereas a colored photographic-quality image may be compressed from 20:1 to 25:1 without experiencing any noticeable degradation in quality [12].

A useful property of JPEG is that the degree of lossness can be varied by adjusting compression parameters. This means that the image maker can trade off file size against output image quality.

Another important aspect of JPEG is that decoders can trade off decoding speed against image quality, by using fast but inaccurate approximations to the required calculations.

Using JPEG is essentially a time/space tradeoff: it can give up some time in order to store or transmit an image more cheaply. But it is worth noting that when network transmission is involved, the time savings from transferring a shorter file can be greater than the time needed to decompress the file.

JPEG makes use of a mathematical transformation known as the Discrete Cosine Transform (DCT) to shift the image’s color values into a mode that can be more efficiently compressed and coded. The DCT is not in itself lossy, but the next step in the compression process, known as quantization, simplifies and rounds the color values before they are encoded, throwing away real information. This is where the JPEG quality slider operates – it governs how much simplification occurs [27].

2.6 JPEG Modes

The JPEG standard defines four modes for image compression:

I. Sequential Lossy Discrete Cosine Transform (DCT)-based Mode

In this mode, an image is first partitioned into blocks of size 8×8 pixels. The blocks are processed from left to right and top to bottom. The (8×8) two-dimensional Forward DCT is applied to each block and the (8×8) DCT coefficients are quantized. Finally, the quantized DCT coefficients are entropy encoded and output as part of the compressed image data [8]. Figure (2-7) shows the block diagram of the sequential DCT-based JPEG encoder [26]. This mode includes the JPEG baseline format which will be explained in chapter four.

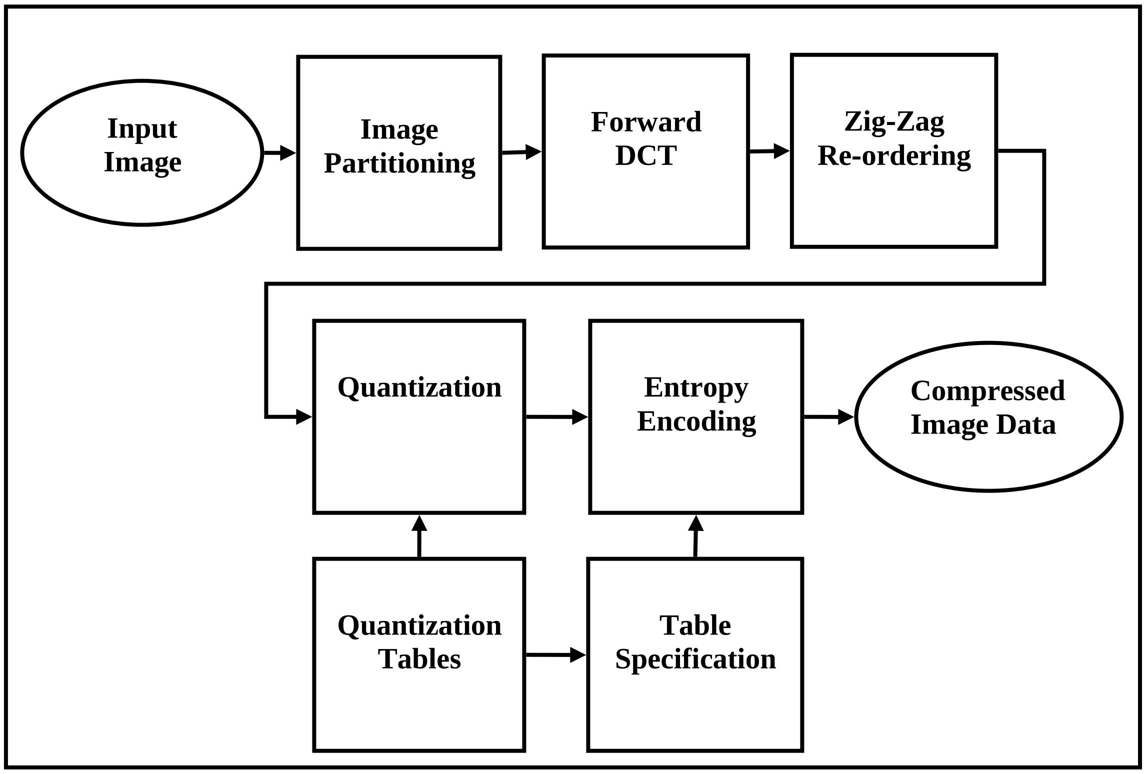


Figure (2-7): Block diagram of the sequential DCT-based JPEG encoder [26]

II.Sequential Lossless Mode

JPEG standard also supports a lossless mode of operation, by providing a simple predictive compression algorithm, rather than DCT-based technique, which is a lossy one. Figure (2-8) shows the block diagram of the lossless JPEG encoder [8].

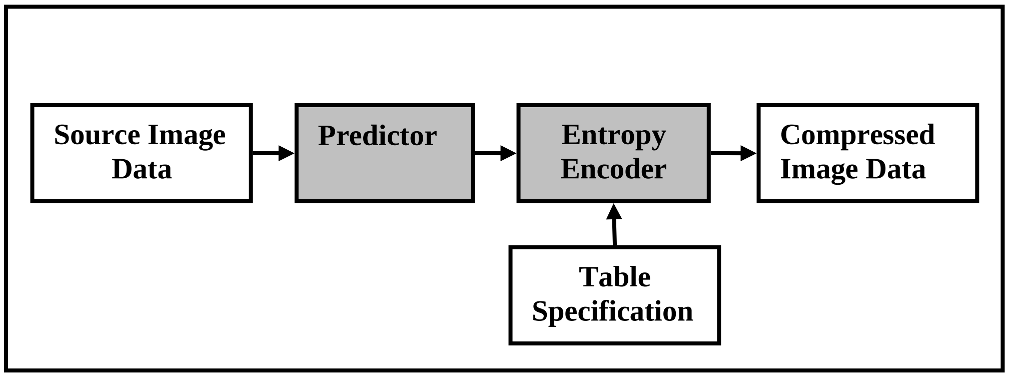


Figure (2-8): Block diagram of the lossless JPEG encoder [8]

The predictor block works in a way that a prediction of the sample **X^** is calculated on the basis of previous samples **A**, **B** and **C**, and then the difference **ΔX=X- X^** is computed, where **X** is the actual value of the sample see (figure (2-9)). Then, the difference **ΔX** is coded using the Huffman or arithmetic encoder. Table (2-3) illustrates several different predictor formulae than can be used for lossless prediction. Lossless JPEG compression typically gives approximately 2:1 compression ratio for moderately complex color images [8].

Table (2-3): Predictors for lossless JPEG compression

Selection value	Predictor formula
0	**No prediction**
1	**X=A**
2	**X=B**
3	**X=C**
4	**X= A+B-C**
5	**X= A+(B-C)/2**
6	**X= B+(A-C)/2**
7	**X= (A+B)/2**

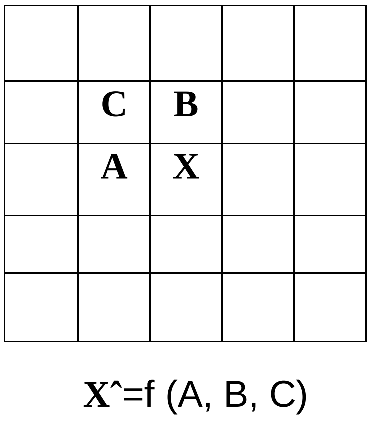


Figure (2-9): Lossless JPEG encoding: location of four samples in the predictor [8]

III.Progressive DCT-based Mode

In some applications, an image may have large numbers of pixels and the decompression process, including transmission of the compressed image over the network, may take several minutes [8]. The progressive mode is intended to support real-time transmission of images. It allows the DCT coefficients to be sent piecemeal in multiple "scans" of the image. With each scan, the decoder can produce a higher-quality rendition of the image. Thus a low-quality preview can be sent very quickly, and then refined as time allows [9]. Figure (2-10) illustrates the progressive coding [26].

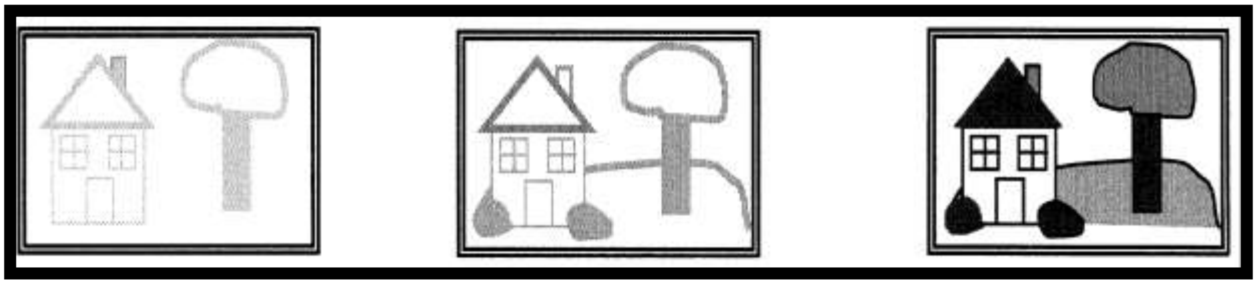


Figure (2-10): Progressive Coding [26]

IV.Hierarchical mode

This format makes it possible to hold a number of images with different resolutions within one file [22]. For example, one could provide 512x512, 1024x1024, and 2048x2048 versions of the image. The higher-resolution images are coded as differences from the next smaller image, and thus require many fewer bits than they would if stored independently. However, the total number of bits will be greater than that needed to store just the highest-resolution frame in baseline form. The individual frames in a hierarchical sequence can be coded progressively if desired. Hierarchical mode is not widely supported at present [9]. Like progressive DCT mode, it uses more than one scan to define the image [22]. Hierarchical JPEG encoder requires significantly more buffer space. However, the benefits are that the encoded image is immediately available at different resolutions [8]. Figure (2-11) illustrates the Hierarchical coding [26].

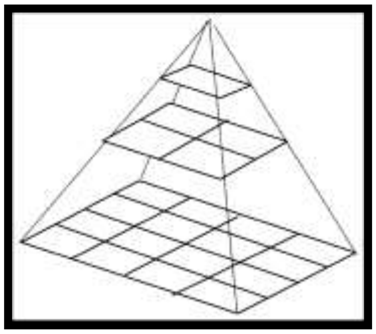


Figure (2-11): Hierarchical coding [26]

2.7 Compression Efficiency Parameters

JPEG image compression is classified as a lossy compression method. Therefore, to evaluate the compression efficiency of the JPEG image compression, the following parameters are utilized in the current analysis [13]:

2.7.1 Compression Ratio [16]

The compression ratio (CR) is the ratio of the number of bits per sample before compression to the encoded data rate:

$$\text{Compression Ratio (CR)} = \frac{\text{uncompressed\_file\_size}}{\text{compressed\_file\_size}} \tag{2.16}$$

2.7.2 Fidelity Criteria [13]

Generally, fidelity criteria can be divided into two classes,

I. Objective Fidelity Criteria:

These borrowed from digital signal processing and information theory. The tests that can be used to measure the amount of error in the reconstructed (decompressed) image are provided by using them.

Among these error criteria the following most common parameters are selected:

1. The Maximum Absolute Error (E_{max})

This parametric test indicates the maximum absolute difference existed between the original uncompressed value and the reconstructed pixel value:

$$\mathbf{E_{max}} = \mathbf{max}\{\mathbf{ABS}(\mathbf{B}(\mathbf{x,y}) - \mathbf{A}(\mathbf{x,y}))\}, \qquad \mathbf{x = 0 \ldots M-1, y = 0 \ldots N-1,} \tag{2.17}$$

where,

A(x, y) is the original image array,

B(x, y) is the decompressed image array,

M, N are the number of columns and rows, respectively,

and ABS ( ) stands for absolute value.

2. The Mean Absolute Error (***MAE***)

It is defined as:

$$MAE = \frac{1}{M \times N} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left| B(x,y) - A(x,y) \right|, \tag{2.18}$$

3. The Mean Square Error (***MSE***)

It is defined as:

$$MSE = \frac{1}{M \times N} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[B(x,y) - A(x,y) \right]^2 \qquad (2.19)$$

4. The Signal-to-Noise Ratio (SNR)

It is defined as:

$$SNR = 10 \log_{10} \left[\frac{\max(A) - \min(A)}{MSE} \right] \qquad (2.20)$$

5. The Peak Signal-to-Noise Ratio (PSNR)

In practice, the peak signal-to-noise ratio (PSNR) is used to measure the difference between two images. It is defined as:

$$PSNR = 20 \log_{10} \left[\frac{(b-1)}{MSE} \right] \qquad (2.21)$$

where,

b: is the largest possible value of the signal (typically 256),

and MSE: is the mean square error.

The PSNR is given in decibel units (dB) which measure the ratio of the peak signal and the difference between two images. The PSNR is very common in image processing; it gives better-sounding numbers than other measures.

II. Subjective Fidelity Criteria [16]***:***

This kind of criteria requires the definition of a qualitative scale to assess image quality. This scale can then be used by human test subjects to determine image fidelity. In order to provide unbiased results, evaluation with subjective measures requires careful selection of the test subject (image) and carefully designed evaluation experiments.

There are three main types of subjective measurements exist. The ***first*** is referred to as impairment test, where the viewers score the images in terms of how bad they are.

The ***second*** is referred to as quality test, where the viewers score the images in terms of how good they are.

The ***third*** is referred to as comparison test, where the images are evaluated side by side, which is the most useful and easiest measure for most people.

CHAPTER THREE

DATA TRANSMISSION

3.1 Data Transmission

Data is transmitted over networks using signals, which are transformed, or encoded, by computers into the voice, video, graphics, and/or the print that are seen on computer screens. The signals used by computers to transmit data are either digital or analog.

1. Analog signals are continuous signals that vary in strength. Sound is an example of an analog signal. Sound is actually a wave and is quite similar, or analogous, to electromagnetic waves, hence the name analog. Telephones have transmitters that encode sound waves into electromagnetic waves, which then travel over wires toward their destination. The receiving telephone decodes the electromagnetic waves back into sound waves. Then, brains of human decode the sound waves into the words they hear. Computer modems use the same principle. Analog signals can be represented digitally. For instance, a high electromagnetic voltage could be interpreted as "1" and low voltage as "0".
2. Digital signals are discrete rather than continuous. Either there is a signal or there is not a signal. Telegraphs transmit data with discrete signals. Discrete signals can be represented by on and off pulses; it also can be represented digitally. The presence of a signal could be coded as a "1" and the absence of a signal coded as a "0". Codes are used to group a set number of bits together and have a group of bits represent a letter, number, or other character.

Digital data is based on two states, "on" or "off". The binary numbering system uses only two digits, "0" and "1", so it makes sense to use the binary numbering system. One digit, "0" represents "off", the other digit represents "on". A single "0" or "1" is called a bit. One byte is equal to eight bits (also called an octet when discussing TCP/IP). In ASCII code, one octet is the equivalent of one alphabetic or numeric character. In order to appreciate how computers communicate over networks, it is necessary to be aware of how they encode information [32]. Data transmission has two types:

3.1.1 Connection-Oriented and Connectionless Transmissions [32]

The main difference between the two types is that with a connection-oriented transmission, the destination device acknowledges receipt. Whereas, with connectionless, there is no acknowledgement.

In connection-oriented transmissions, the sending (source) device establishes a connection with the receiving (destination) device. The connection is continued until all data packets have been transmitted and the source device receives notification that the data was received by the destination device and has been checked for errors.

A telephone conversation is an example of a connection-oriented transmission. When a call is made, data is transmitted across phone lines, the receiving party picks up the phone, and a conversation takes place.

In a connectionless transmission, the source device transmits data but the connection is not maintained. The source device does not wait for notification that the destination device actually received the information accurately. This method is faster than connection-oriented, however less reliable since there is no notification of whether the data is received or not.

3.1.2 Bit-Parallel and Bit-Serial Transmissions [25]

In parallel transmission, all bits of a piece of information are transmitted at the same time. Bit-parallel via an appropriate number of signal lines. The installation costs are high and only acceptable for short distances. The transmission of one byte alone requires eight lines and a reference potential. Therefore, this technique is presently almost only used for device busses. This application over short distances requires high transmission rates while doing without conversion methods that need a large number of components. Figure (3-1) illustrates parallel transmission.

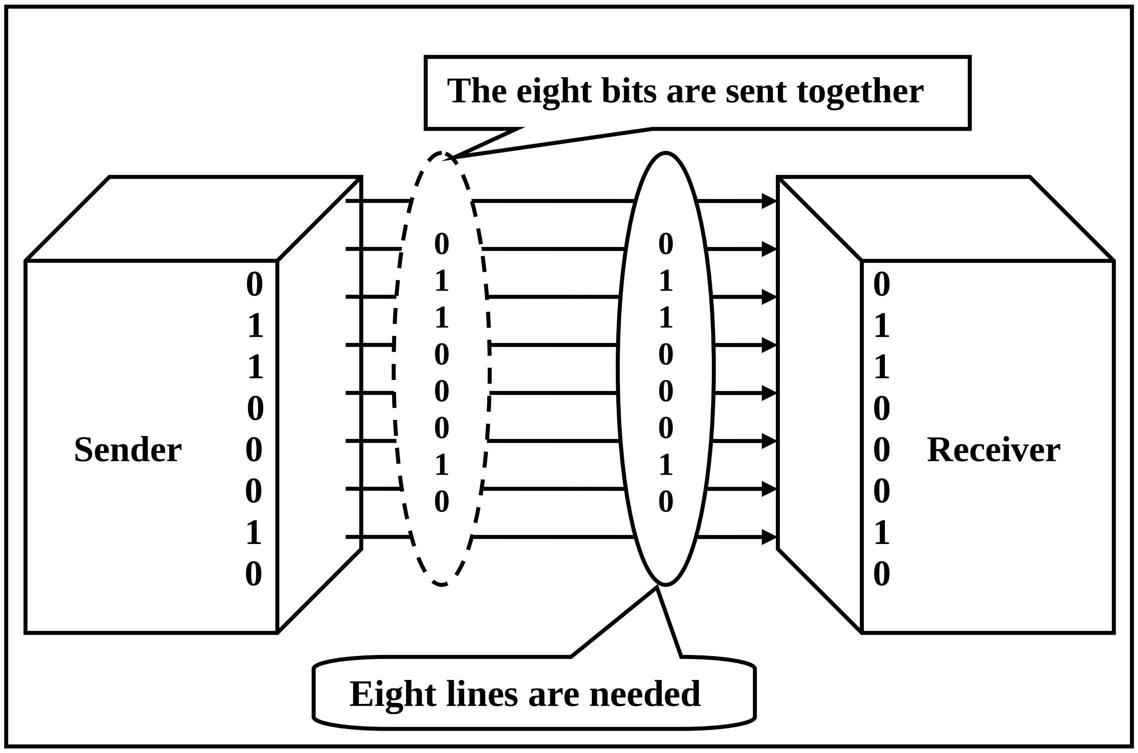


Figure (3-1): Parallel Transmission [35]

For long distances, serial transmission is a good solution. Here, only one signal line transmits the bits one after the other. As a result, the transmission of information takes more time, which is nevertheless acceptable because, on the other hand, the installation effort and the costs are considerably reduced. Since all the information is mostly generated and processed in bit-parallel mode, the transmitter must convert the data from parallel to serial, and the receiver must reconvert it from serial to parallel. This function is performed by specially operated shift registers which are already integrated in communication modules available on the market. Figure (3-2) illustrates serial transmission.

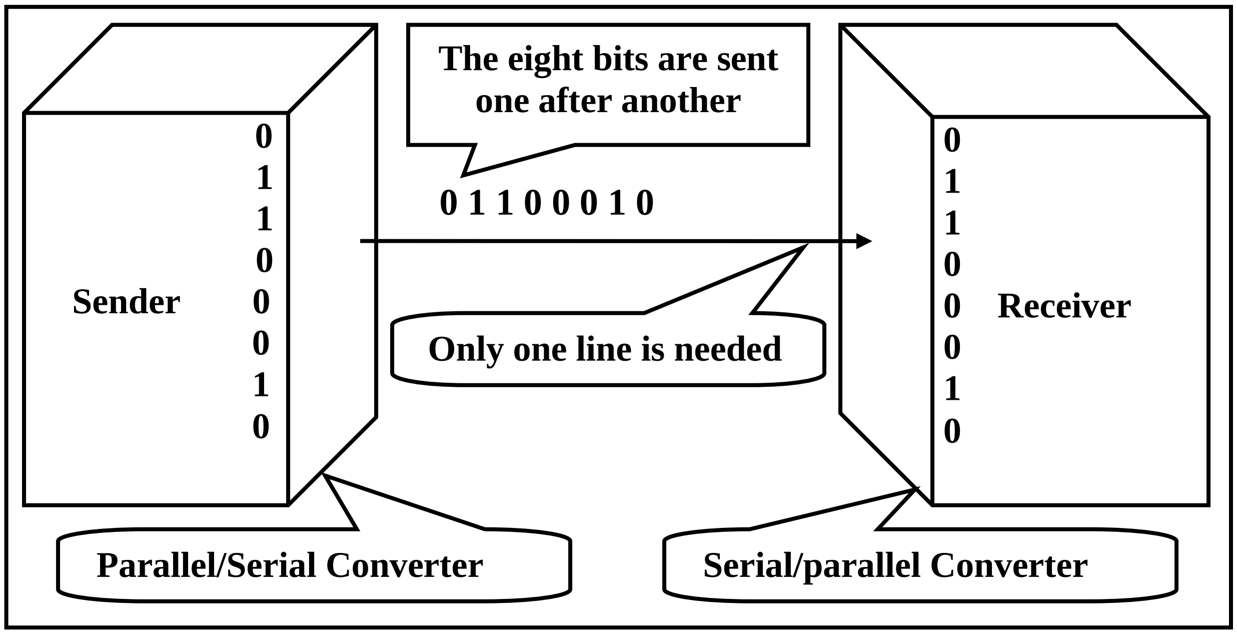

The eight bits are sent one after another
0 1 1 0 0 0 1 0
Sender
0 1 1 0 0 0 1 0
Only one line is needed
0 1 1 0 0 0 1 0
Receiver
Parallel/Serial Converter
Serial/parallel Converter

Figure (3-2): Serial Transmission [35]

Computers need to know when to expect data and where a character begins and ends. When the data are received, timing on both computer devices must be coordinated if they are to work together efficiently. This coordination is called clocking, timing, or framing. Therefore, serial transmission is classified into two types:

I. Synchronous Transmission [32]

When the data are transferred, both the transmitting and receiving nodes need to agree when the signal begins and ends, so the signals can be correctly measured and interpreted. This timing process is called bit synchronization, framing, or clocking.

It is difficult to read if there are no punctuation and spaces that can be done, because there are several different characters which are not in code what if these are coded as zeros or ones. As can be seen, synchronization of data is very important.

Synchronous transmission requires the communicating devices to maintain synchronous clocks during the entire connection. The sending device transmits on a specific schedule and the receiving device accepts the data on that same fixed schedule. The receiving device knows the timing of the sending device because the timing information is embedded within the preamble of the frame. Synchronous transmissions are common in internal computer communications and usually are sent as entire frames.

Synchronous transmission is common when large blocks of data are transferred, since it is efficient and has a low overhead (number of bytes of data/control plus data bytes). Figure (3-3) illustrates synchronous type of transmission.

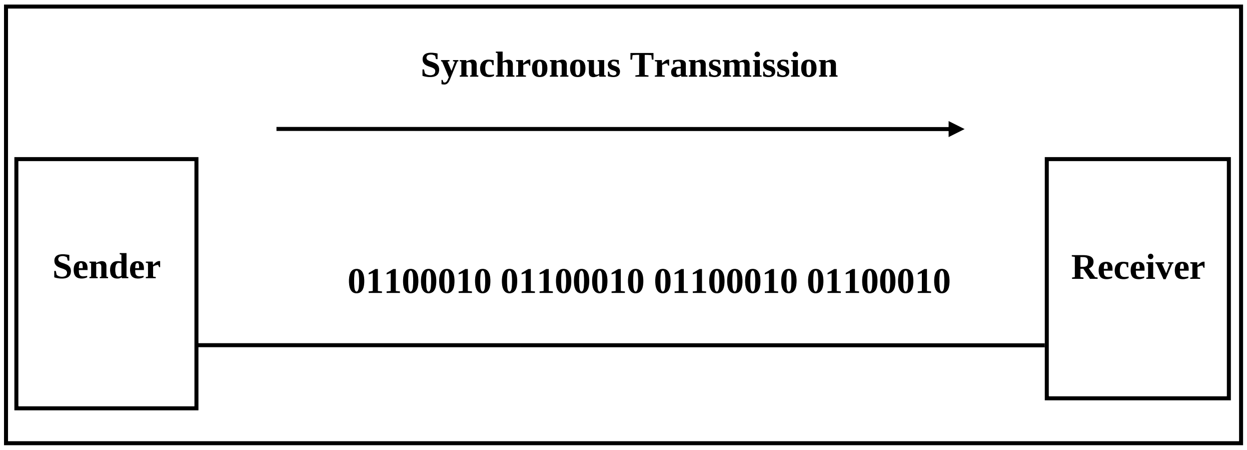


Figure (3-3): Synchronous Transmission [35]

II. *Asynchronous Transmission* [32]

Asynchronous data transmission does not involve synchronizing the clocks of the sending and receiving devices. Instead, start and stop bits are used for synchronization of data signals. The start and stop bits tell the receiving device how to interpret the data. Asynchronous sends one character at a time. Figure (3-4) illustrates Asynchronous type of transmission.

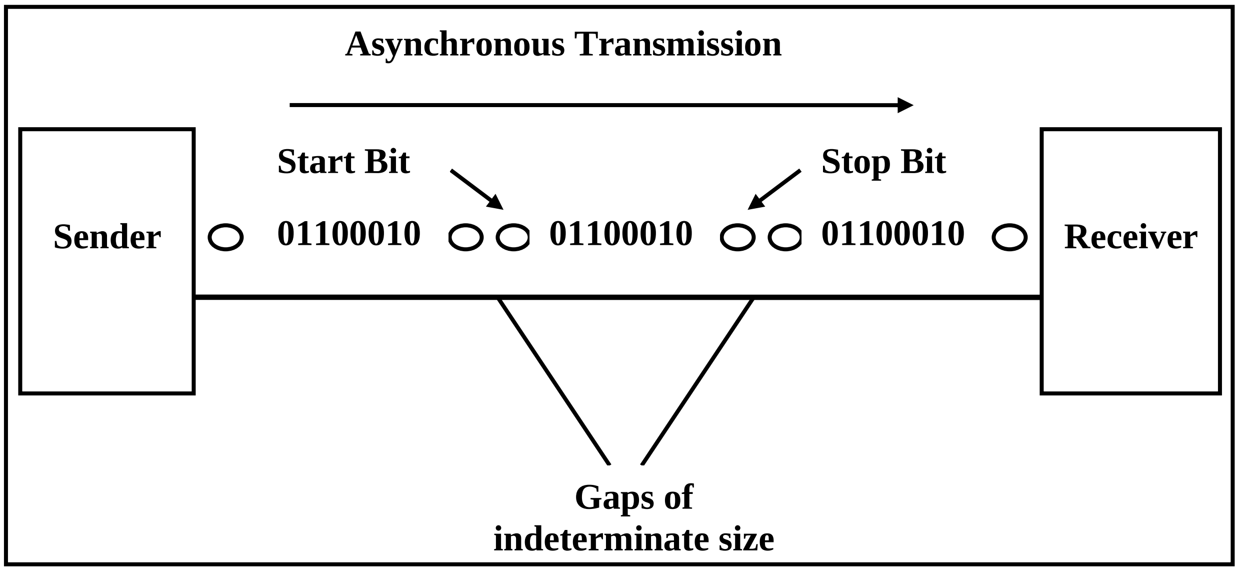


Figure (3-4): Asynchronous Transmission [35]

3.2 Advantages of Digital Data Transmission over Analog Data Transmission [31]

1. The voice data, music and images (e.g. television, fax and video) can be interspersed to make more efficient use of the circuits and equipment.

2. Much higher data rates are possible using existing telephone lines.

3. Digital transmission is much cheaper than analog transmission, since it is not necessary to accurately reproduce an analog waveform after it has passed through potentially hundreds of amplifiers on a transcontinental call. Being able to correctly distinguish a "0" from a "1" is enough.

4. Maintenance of a digital system is easier than maintenance of analog one. A transmitted bit is either received correctly or not, making it simpler to track down problems.

5. A digital signal can pass through an arbitrary number of regenerators (amplifiers in analog systems) with no loss in signal and thus travel long distances with no information loss. In contrast, analog signals always suffer some information loss when amplified, and this loss is accumulative. Hence digital transmission can be made to have low error rate.

3.3 Transmission Media Types

Telecommunications are the transfer of information across a distance. The information, in its native form, can be voice or other forms of audio, computer data, facsimile or other forms of image, video or even multimedia. There must be a transmitter, or originating device, and at least one receiver, or destination device. For supporting the transmission there must be some physical medium, with physical referring to physics and not necessarily to anything tangible.

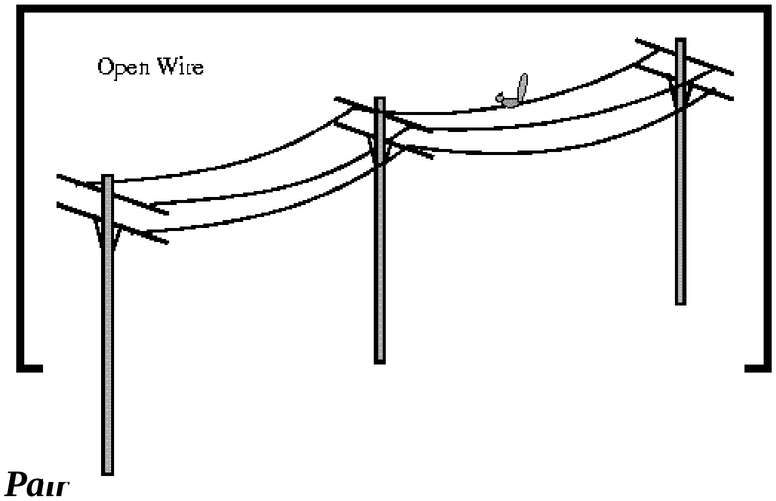
There is a number of options that can be used in selecting a transmission medium. All make use of some form of electromagnetic energy, which can be in the specific form of electricity, radio or light [14]. There are two main types of transmission media:

3.3.1 Guided Transmission Media [34]

Guided Transmission Media uses a "cabling" system that guides the data signals along a specific path. The data signals are bound by the "cabling" system. Guided Media is also known as Bound Media. Cabling is meant in a generic sense in the previous sentences and is not meant to be interpreted as copper wire cabling only. There are four basic types of Guided Media:

I. Open Wire

Open Wire is traditionally used to describe the electrical wire strung along power poles. There is a single wire strung between poles. No shielding or protection from noise interference is used. The traditional definition of Open Wire will be extended to include any data signal path without shielding or protection from noise interference. This can include multiconductor cables or single wires. This media is susceptible to a large degree of noise and interference and consequently not acceptable for data transmission except for short distances under 20 feet. See figure (3-5).



II. Twisted Pair

The wires in Twisted Pair cabling are twisted together in pairs. Each pair would consist of a wire used for the (+ve) data signal and a wire used for the (-ve) data signal. See figure (3-6).

Any noise that appears on one wire of the pair would occur on the other wire. Because the wires are opposite polarities, they are 180 degrees out of phase (180 degrees - phasor definition of opposite polarity). When the noise appears on both wires, it cancels or nulls itself out at the receiving end. Twisted Pair cables are most effectively used in systems that use a balanced line method of transmission: polar line coding (Manchester Encoding) as opposed to unipolar line coding (TTL logic).

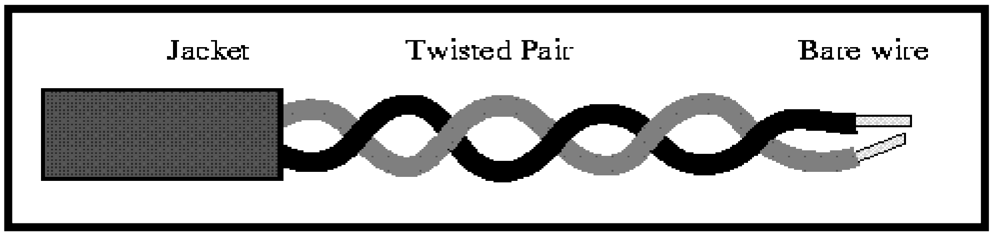


Figure (3-6): Unshielded Twisted Pair [34]

The degree of reduction in noise interference is determined specifically by the number of turns per foot. Increasing the number of turns per foot reduces the noise interference. To further improve noise rejection, a foil or wire braid shield is woven around the twisted pairs, See figure (3-7).This "shield" can be woven around individual pairs or around a multi-pair conductor (several pairs).

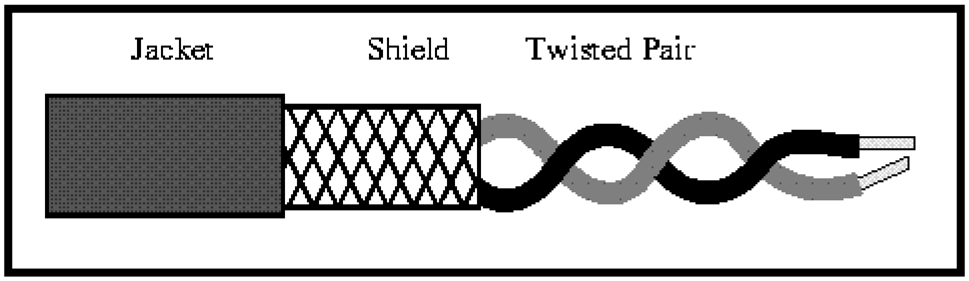


Figure (3-7): Shielded Twisted Pair [34]

Cables with a shield are called Shielded Twisted Pair (STP). Cables without a shield are called Unshielded Twisted Pair or (UTP). Twisting the wires together results in a characteristic impedance for the cable.

UTP is used on Ethernet 10BaseT and can also be used with Token Ring. It uses the RJ line of connectors (RJ45, RJ11, etc...) STP is used with the traditional Token Ring cabling or ICS - IBM Cabling System.

III. Coaxial Cable

Coaxial Cable consists of two conductors. The inner conductor is held inside an insulator with the other conductor woven around it providing a shield. An insulating protective coating called a jacket covers the outer conductor, See figure (3-8).

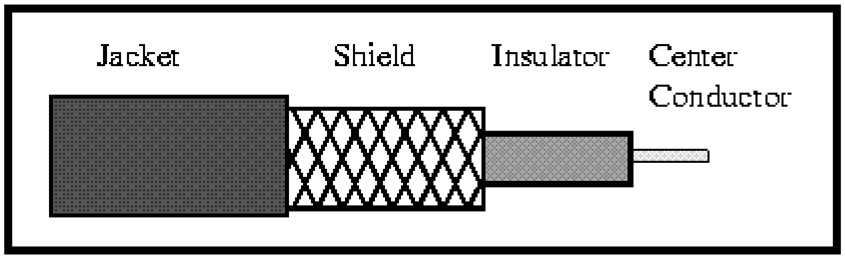


Figure (3-8): Coaxial Cable [34]

The outer shield protects the inner conductor from outside electrical signals. The distance between the outer conductor (shield) and inner conductor plus the type of material used for insulating the inner conductor determine the cable properties or impedance.

IV. Optical Fiber

Optical Fiber consists of thin glass fibers that can carry information at frequencies in the visible light spectrum and beyond. The typical optical fiber consists of a very narrow strand of glass called the Core. Around the Core is a concentric layer of glass called the Cladding. Figure (3-9) illustrates a single fibre in (a) and a sheath of three fibres in (b). Other configurations are possible.

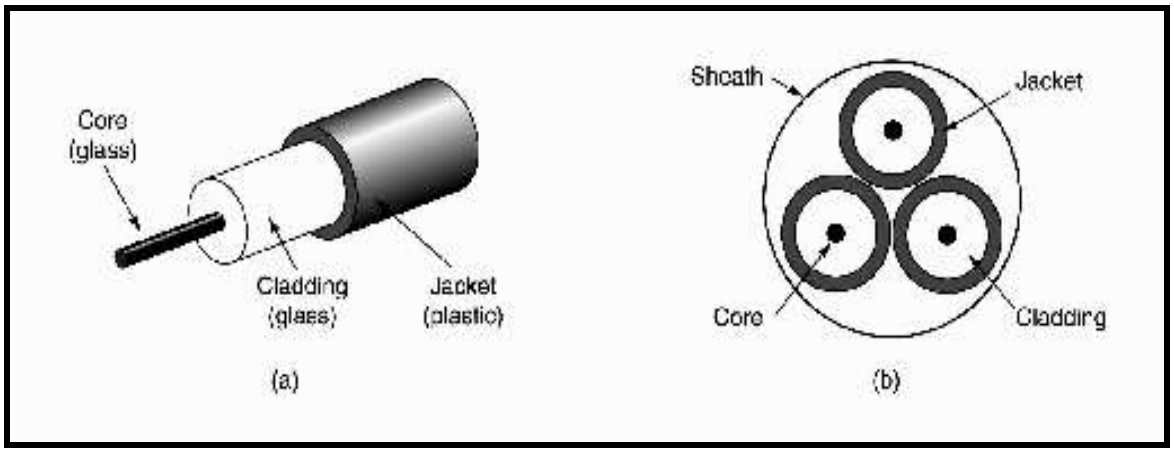


Figure (3-9): Optical Fiber [34]

3.3.2 Unguided Transmission Media [14]

Unguided media do not make use of conductors. Rather, the signal simply radiates through space between transmitter and receiver. Sometimes they are called airwave systems. They are more correctly spacewave, or free-space systems; the air, if there is any separator between transmitter and receiver, actually causes the signal to weaken and distort.

Radio systems are the most common in the category of heavy duty radiated transmission systems, with the focus on microwave and satellite. There are also a host of applications-specific variations on the theme, including paging, cellular, cordless telephony, and various packet radio systems. Free-space laser systems, which most commonly in the form of infrared (IR), are optical in nature.

I .Microwave System

Microwave radio runs in the Ultra-High Frequency (UHF) up to the Extremely High Frequency (EHF) bands, which covers the range between 300 MHz and 300 GHz.

Such systems make use of shaped transmitters and receivers, which tightly focus the radio beams for maximum effect over relatively long distances (up to and even exceeding 30 miles or so). The focusing of the beams is critical, as the signal tends to spread over a distance, and as such a high frequency signal is severely impacted by physical matter (e.g., rain, fog, smog and haze) between the transmit and receive antenna. Line-of-sight is critical, as dense physical (e.g., trees and mountains) is totally unacceptable.

Several Wireless Local Loop (WLL) systems such as Local Multipoint Distribution Services (LMDS) and Multichannel Multipoint Distribution Services (MMDS) are also being run in the microwave spectrum. These systems make use of a spread beam, rather than a tightly focused beam, but are used in short-haul applications generally limited to 3-5 miles or so. Some Wireless Local Area Networks (WLANs) make use of Radio Frequency (RF) in the microwave range, while other run in the 900 MHz range.

II.Satellite System

Satellite is essentially non-terrestrial microwave, in some cases running in the same frequency range as terrestrial systems. The most common satellite systems are Geosynchronous Earth-Orbiting (GEOs), which are always positioned in slots directly above the equator and at altitudes of approximately 22,300 miles.

In such slots and at such altitudes, the satellites always (actually, that's stretching the truth just a bit) maintain their positions relative to the Earth's surface.

Low Earth-Orbiting (LEOs) are in non-equatorial orbital paths and are at much lower altitudes, and Middle Earth-Orbiting (MEOs) are at intermediate altitudes. In such paths and at such altitudes, LEOs and MEOs do not maintain their relative positions. Rather, they whiz around the Earth much like electrons whiz around the nucleus of an atom.

Satellites offer a number of advantages, including their area of coverage, or footprint. Since they are positioned at such high altitudes, they can transmit to, and receive from, a large area. Satellites, therefore, offer great advantage in point-to-multipoint and broadcast applications. Like all microwave systems, however, their performance varies with the weather. Propagation delay is a big issue with satellites, as the signal must travel about 45,000 miles between transmitter and receiver, even at the speed of light that takes a while.

III. Infrared (IR) System

Infrared (IR), and other free-space optical systems, are used in short-haul applications, most effectively where direct line-of-sight can be achieved. Some WLANs make use of IR, although most are RF-based. IR-based WLL systems run at speeds up to 622 Mbps, although they are currently somewhat unusual.

The primary application is in LAN bridging under circumstances that do not allow wired connectivity to be achieved quickly or cost-effectively. IR systems for WLL applications are in development.

3.4 Internet Transmission Models

There are many transmission models that can be used in Internet. The most commonly used models are:

3.4.1 Open System Interconnection (OSI) Model

The OSI reference model describes how information from a software application in one computer moves through a network medium to a software application in another computer. The OSI reference model is an obsolete conceptual model composed of seven layers, each specifying particular network functions. The model was developed by the International Organization for Standardization (IOS) in 1984, and it is now used at various stupid and no so stupid certifications. The OSI model does not corresponds to TCP/IP model and has too many layers that are more conceptual exercise (each layer is reasonably self-contained) then of practical importance [4]. The layers of the Open System Interconnection (OSI) reference model are [7]:

•**Layer7.** ***Application Layer***: The highest layer. It generates or interprets data, may also provide encryption or decryption. Applications using the network learn how to send a request, specify a filename over the net, and respond to a request. Protocol Data Unit (PDU) is called Data at this layer. This layer interfaces directly to and performs common application services for the application processes. It also issues requests to the Presentation Layer. The common application services provide semantic conversion between associated application processes. Examples of common application services of general interest include the virtual file, virtual terminal, and job transfer and manipulation protocols.

•**Layer6.** ***Presentation Layer***: Determines how computers represent data [ASCII, GIF...]. PDU is called Data at this layer. This layer responds to service requests from the Application Layer and issues service requests to the Session Layer. The Presentation Layer relieves the Application Layer of concern regarding syntactical differences in data representation within the end-user systems. Note: An example of a presentation service would be the conversion of an EBCDIC-coded text file to an ASCII-coded file.

•Layer5. *Session Layer*: Establishing a communication session, security, and authentication. NetBIOS is a layer 5 protocol. PDU is called Data at this layer. This layer responds to service requests from the Presentation Layer and issues service requests to the Transport Layer. The Session Layer provides the mechanism for managing the dialogue between end-user application processes. It provides for either duplex or half-duplex operation and establishes check-pointing, adjournment, termination, and restart procedures.

•Layer4. *Transport Layer*: Provides transfer correctness, data recovery, and flow control. TCP is a layer 4 protocol. PDU is called a ***segment*** at this layer. This layer responds to service requests from the Session Layer and issues service requests to the Network Layer. The purpose of the Transport Layer is to provide transparent transfer of data between end users, thus relieving the upper layers from any concern with providing reliable and cost-effective data transfer.

•Layer3. *Network Layer*: Provides address assignment, and packet's forwarding methods. PDU is called a ***packet*** at this layer. This layer responds to service requests from the Transport Layer and issues service requests to the Data Link Layer. The Network Layer provides the functional and procedural means of transferring variable length data sequences from a source to a destination via one or more networks while maintaining the quality of service requested by the Transport Layer. The Network Layer performs network routing, flow control, segmentation/desegmentation, and error control functions.

•Layer2. *Data Link Layer*: Frame format, transmitting frames over the net (additional bit/byte stuffing, start / stop flags, checksum, and CRC). CAN bus, ATM, StarLAN, LocalTalk and HDLC are layer 2 protocols. Different network and protocol characteristics are defined by different Data Link Layer specifications.

The Data Link Layer is subdivided into the Media Access Control (MAC) which controls access and encodes data into a valid signaling format (for the physical layer), and the Logical Link Control (LLC), which provides the link to the network (for the network layer). PDU is called a ***frame*** at this layer. This layer responds to service requests from the Network Layer and issues service requests to the Physical Layer. The Data Link Layer provides the functional and procedural means to transfer data between network entities and to detect and possibly correct errors that may occur in the Physical Layer. Examples of data link protocols are HDLC and ADCCP for point-to-point or packet-switched networks and LLC for local area networks.

•Layer1. *Physical Layer*: Defines the physical (hardware) implementation and the electrical (signal level) implementation of the bus; network cabling, connector type, pin-out, physical data rates, maximum transmission distances, and data transmission encoding. At this layer information is placed on the physical network medium. **RS-232** and **RS422** are examples of a Physical Layer specification. PDU is called a ***bit*** at this layer. The Physical Layer performs services requested by the Data Link Layer. The major functions and services performed by the Physical Layer are: (a) establishment and termination of a connection to a communications medium; (b) participation in the process whereby the communication resources are effectively shared among multiple users, e.g., contention resolution and flow control; and (c) conversion between the representation of digital data in user equipment and the corresponding signals transmitted over a communications channel. Figure (3-10) illustrated all layers.

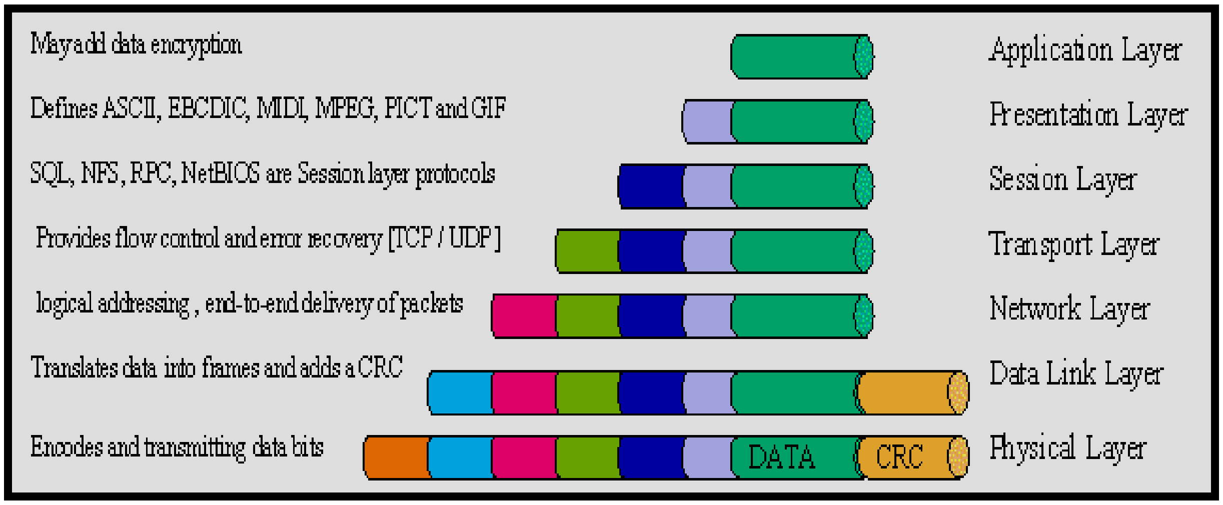

May add data encryption
Defines ASCII, EBCDIC, MIDI, MPEG, PICT and GIF
SQL, NFS, RPC, NetBIOS are Session layer protocols
Provides flow control and error recovery [TCP / UDP]
logical addressing , end-to-end delivery of packets
Translates data into frames and adds a CRC
Encodes and transmitting data bits
DATA
CRC
Application Layer
Presentation Layer
Session Layer
Transport Layer
Network Layer
Data Link Layer
Physical Layer

Figure (3-10): Open System Interconnection Model [7]

3.4.2 TCP/IP Model [33]

TCP/IP protocols map to a four-layers conceptual model known as the Developed Advanced Research Projects Agency (DARPA) model, named after the U.S. government agency that initially developed TCP/IP. The four layers of the DARPA model are: Application, Transport, Internet, and Network Interface. Each layer in the DARPA model corresponds to one or more layers of the seven-layer Open Systems Interconnection (OSI) model. Figure (3-11) shows the TCP/IP protocol architecture.

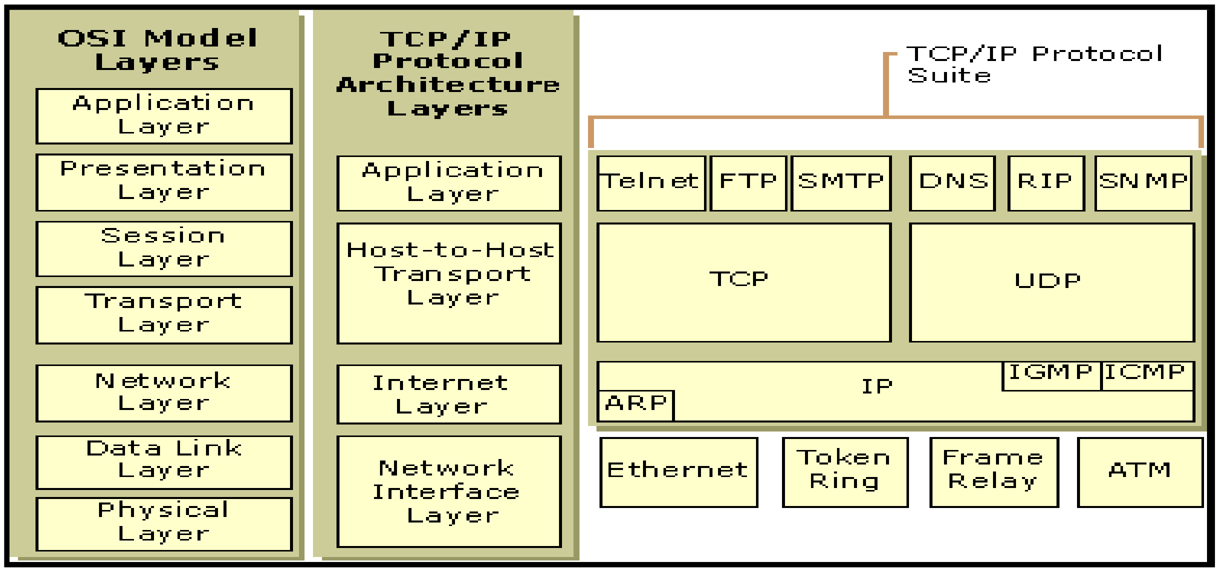


Figure (3-11): TCP/IP protocol architecture [33]

The layers of the TCP/IP reference model are:

•**Layer4.** ***Application Layer:*** The Application Layer provides applications the ability to access the services of the other layers and defines the protocols that applications use to exchange data. There are many Application Layer protocols and new protocols are always being developed.

The most widely known Application Layer protocols are those used for the exchange of user information:

1. The HyperText Transfer Protocol (HTTP) is used to transfer files that make up the Web pages of the World Wide Web.
2. The File Transfer Protocol (FTP) is used for interactive file transfer.
3. The Simple Mail Transfer Protocol (SMTP) is used for the transfer of mail messages and attachments.
4. Telnet, a terminal emulation protocol, is used for remote login to network hosts.

Additionally, the following Application Layer protocols help to facilitate the use and management of TCP/IP networks.

5. The Domain Name System (DNS) is used to resolve a host name to an Internet Protocol (IP) address.
6. The Routing Information Protocol (RIP) is a routing protocol that routers use to exchange routing information on an IP internetwork.
7. The Simple Network Management Protocol (SNMP) is used between network management console and network devices (routers, bridges, and intelligent hubs) to collect and exchange network management information.

•**Layer3.** ***Transport Layer:*** The Transport Layer (also known as the Host-to-Host Transport Layer) is responsible for providing the Application Layer with session and datagram communication services. The core protocols of the Transport Layer are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP):

1. TCP provides a one-to-one, connection-oriented, reliable communications service. TCP is responsible for the establishment of a TCP connection, the sequencing and acknowledgment of packets sent, and the recovery of packets lost during transmission.
2. UDP provides a one-to-one or one-to-many, connectionless, unreliable communications service.UDP is used when the amount of data to be transferred is small (such as the data that would fit into a single packet), when the overhead of establishing a TCP connection is not desired, or when the applications or upper layer protocols provide reliable delivery.

The Transport Layer encompasses the responsibilities of the OSI Transport Layer and some of the responsibilities of the OSI Session Layer.

•Layer2. ***Internet Layer:*** The Internet Layer is responsible for addressing, packaging, and routing functions. This layer is analogous to the Network Layer of the OSI model. The core protocols of the Internet Layer are:

1. The Internet Protocol (IP) is a routable protocol responsible for IP addressing and the fragmentation and reassembly of packets.
2. The Address Resolution Protocol (ARP) is responsible for the resolution of the Internet Layer address to the Network Interface Layer address, such as a hardware address.
3. The Internet Control Message Protocol (ICMP) is responsible for providing diagnostic functions and reporting errors or conditions regarding the delivery of IP packets.
4. The Internet Group Management Protocol (IGMP) is responsible for the management of IP multicast groups.

•Layer1. ***Network Interface Layer*****:** The Network Interface Layer (also called the Network Access Layer) is responsible for placing TCP/IP packets on the network medium and receiving TCP/IP packets off the network medium. TCP/IP was designed to be independent of the network access method, frame format, and medium. In this way, TCP/IP can be used to connect differing network types. This includes LAN technologies such as Ethernet or Token Ring and WAN technologies such as X.25 or Frame Relay.

Independence from any specific network technology gives TCP/IP the ability to be adapted to new technologies such as Asynchronous Transfer Mode (ATM).

The Network Interface Layer encompasses the Data Link and Physical Layers of the OSI Model. Note that the Internet Layer does not take advantage of sequencing and acknowledgment services that may be present in the Data Link Layer. An unreliable Network Interface Layer is assumed, and reliable communications through session establishment and the sequencing and acknowledgment of packets is the responsibility of the Transport Layer.

CHAPTER FOUR

THE PROPOSED SYSTEM

4.1 The Main Structure of the Like Baseline JPEG Standard (LBJS) System

Generally speaking, Like Baseline JPEG Standard (LBJS) system is designed for compression 8-bit-gray scale digital images. This system uses discrete cosine transform (DCT), two types of quantization: linear and uniform, three types of conversion: Horizontal, vertical and zigzag, and Huffman coding method.

First, the user must enter the correct password to pass the system as shown in figure (4-1) below. After the user has passed the system, he could use the compression option to compress any (256×256) bitmapped image with 8-bit-depth. Moreover, he can select the decompression option to reconstruct the image as shown in figure (4-2).

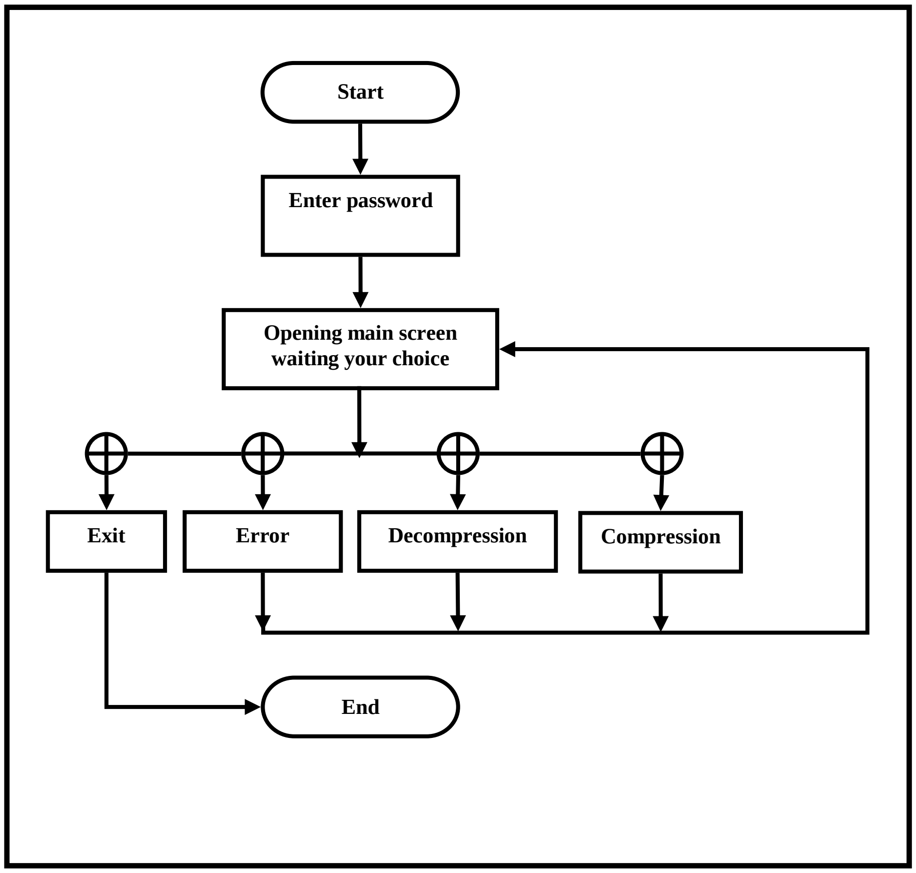


Figure (4-1): The main structure of LBJS system

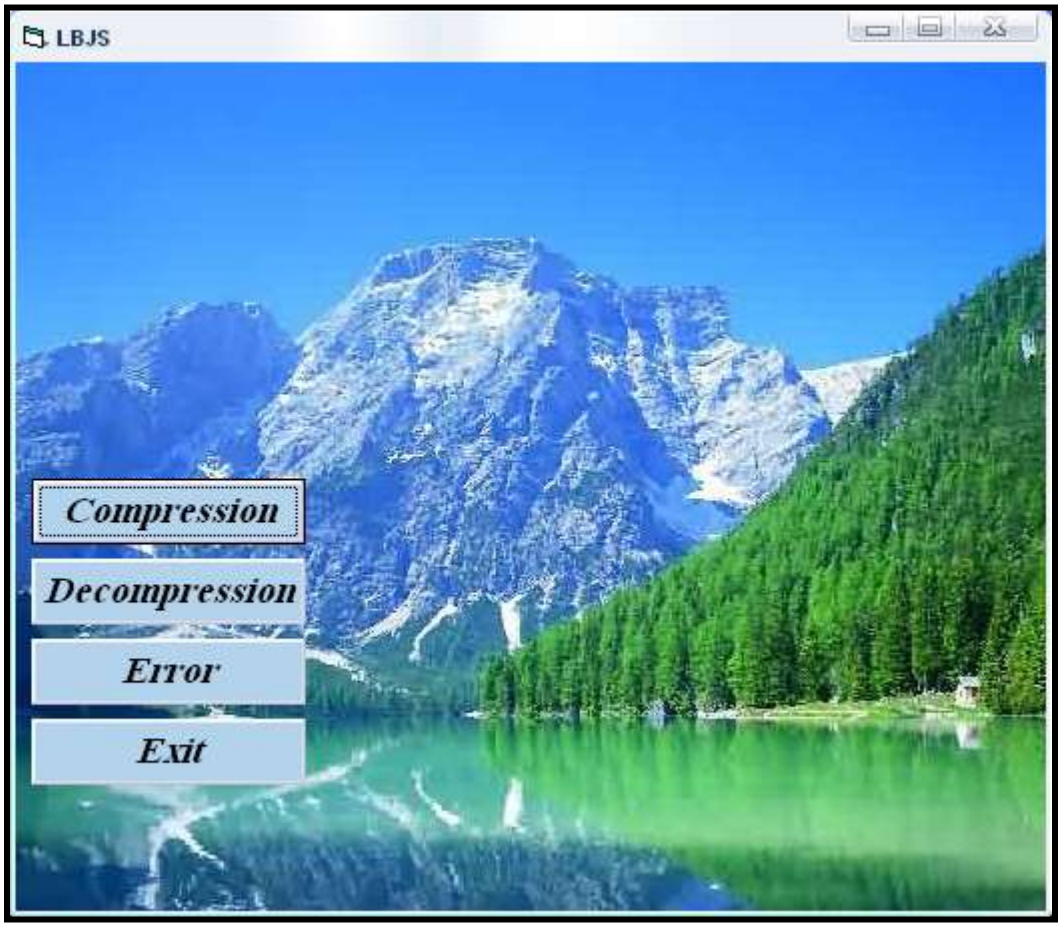


Figure (4-2): The main screen of LBJS system

1. The main steps that are used to compress a (2D) image are described as follows:

Step one: Down-sampling the image from (256×256) to (128×128).

Step two: Partitioning the image into non-overlapped (8×8) blocks from top to bottom and from left to right.

Step three: Performing a (2D-DCT) on the down sampled block.

Step four: Quantizing the transformed block.

Step five: Converting the quantized block form (2D) to (1D).

Step six: Converting the result into intermediate form by using RLE.

Step seven: Encoding the intermediate form by using Huffman Coding.

2. The main steps that are used to decompress the image are described as follows:

Step one: Reading the header and data from the encoded file.

Step two: Encoding the data according to Huffman tables to retrieve the intermediate form.

Step three: Converting the result to (1D) form.

Step four: Converting the (1D) form to (2D) form.

Step five: Dequantizing the (2D) form.

Step six: Performing the inverse (2D-DCT).

Step seven: Re-assembling the returned (8×8) blocks to obtain an (128×128) image.

Step eight: Up-sampling the retrieved (128×128) image to obtain a (256×256) image.

3. The main steps that are used to compare the original image and the reconstructed image are described as follows:

Step one: Reconstructing the compressed image.

Step two: Opening the original image.

Step three: Applying the objective measurements to calculate the C.R. and PSNR.

4. Exit: Terminating all files and exiting from the system.

4.2 The Sampling Process

The sampling process converts the image of size (256×256) to image of size (128×128) as shown in figure (4-3). This process reduces the number of bits needed to represent the image. In this section five types of sampling that are used in our work are described. These types are:

(a) (b)

Figure (4-3): Image sampling

(a) Original image (256×256) (b) Down-sampled image (128×128)

4.2.1 Classical Sampling

In this section, the image is down-sampled by factor of two in both directions according to algorithm shown in figure (4-4a). The up-sampling process is done according to algorithm shown in figure (4-4b).

4.2.2 Even-Even Sampling

In this section, the image is down-sampled by removing even rows and even columns according to algorithm shown in figure (4-5a). The up-sampling process is done according to algorithm shown in figure (4-5b).

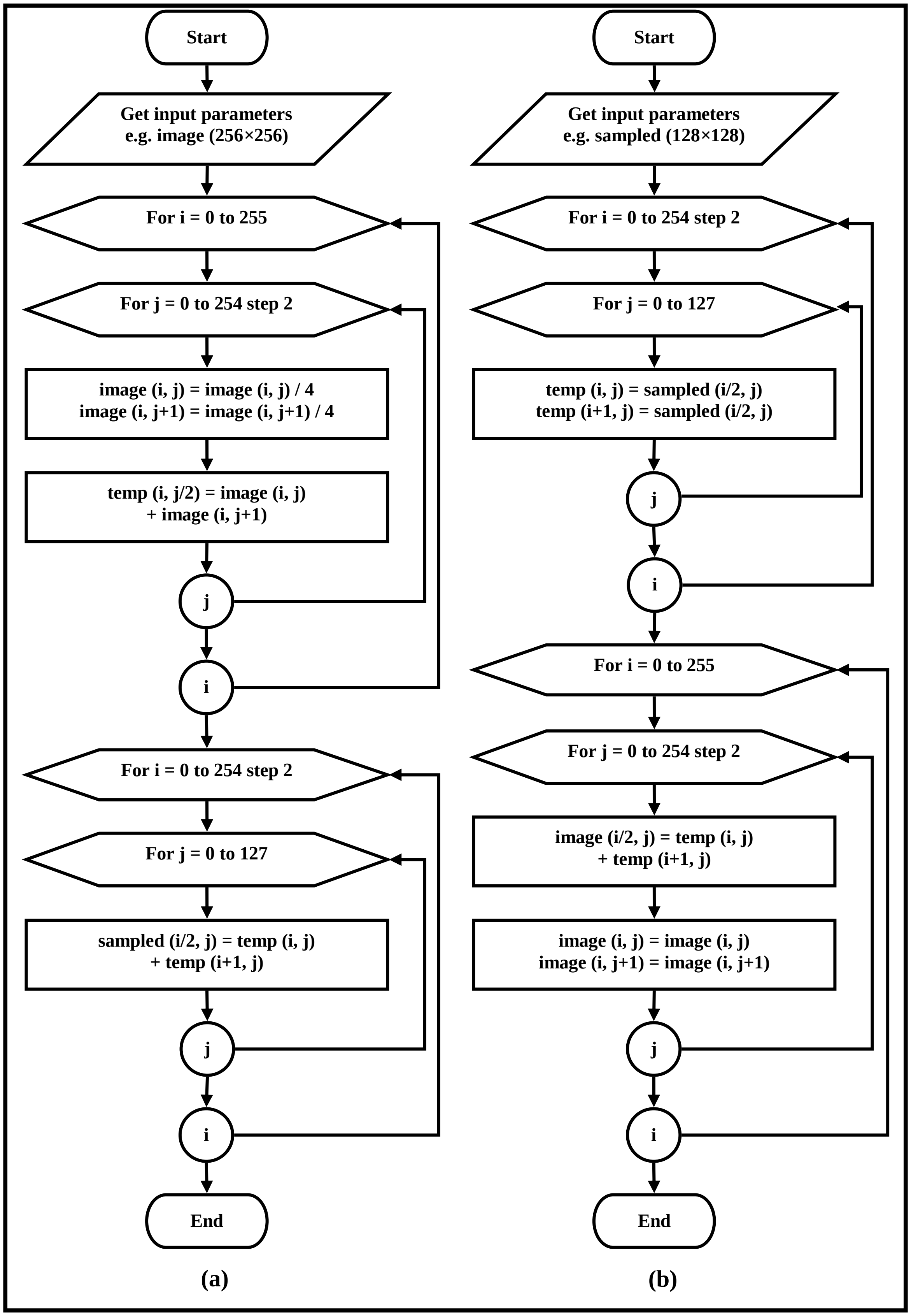

Start
Get input parameters e.g. image (256×256)
For i = 0 to 255
For j = 0 to 254 step 2
image (i, j) = image (i, j) / 4
image (i, j+1) = image (i, j+1) / 4
temp (i, j/2) = image (i, j) + image (i, j+1)
j
i
For i = 0 to 254 step 2
For j = 0 to 127
sampled (i/2, j) = temp (i, j) + temp (i+1, j)
j
i
End
(a)
Start
Get input parameters e.g. sampled (128×128)
For i = 0 to 254 step 2
For j = 0 to 127
temp (i, j) = sampled (i/2, j)
temp (i+1, j) = sampled (i/2, j)
j
i
For i = 0 to 255
For j = 0 to 254 step 2
image (i/2, j) = temp (i, j) + temp (i+1, j)
image (i, j) = image (i, j)
image (i, j+1) = image (i, j+1)
j
i
End
(b)

Figure (4-4): Classical sampling
(a) Down-sampling (b) Up-sampling

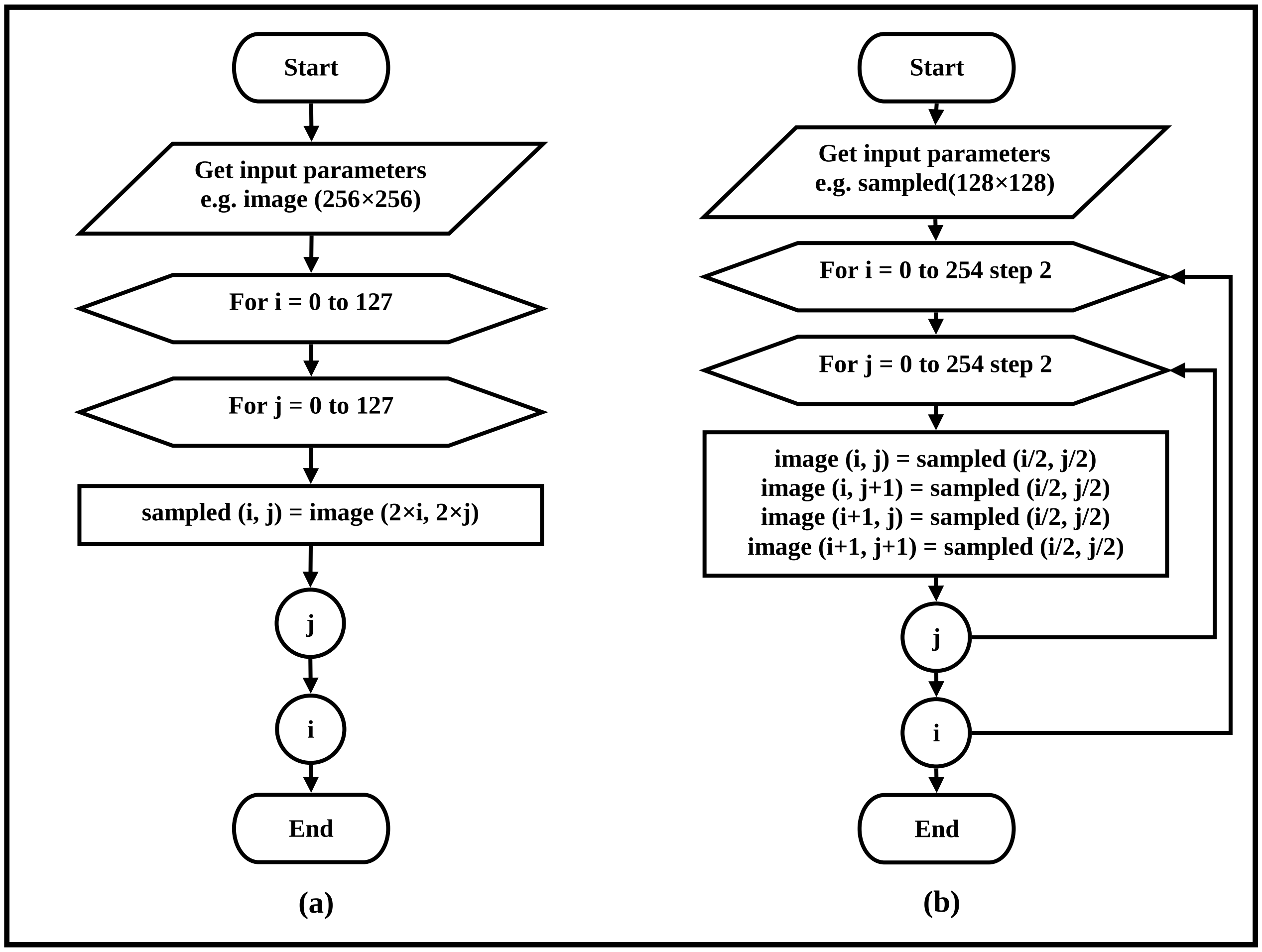


Figure (4-5): Even-Even sampling
(a) Down-sampling (b) Up-sampling

4.2.3 Even-Odd Sampling

In this section, the image is down-sampled by removing even rows and odd columns according to algorithm shown in figure (4-6a). The up-sampling process is done according to algorithm shown in figure (4-6b).

4.2.4 Odd-Even Sampling

In this section, the image is down-sampled by removing odd rows and even columns according to algorithm shown in figure (4-7a). The up-sampling process is done according to algorithm shown in figure (4-7b).

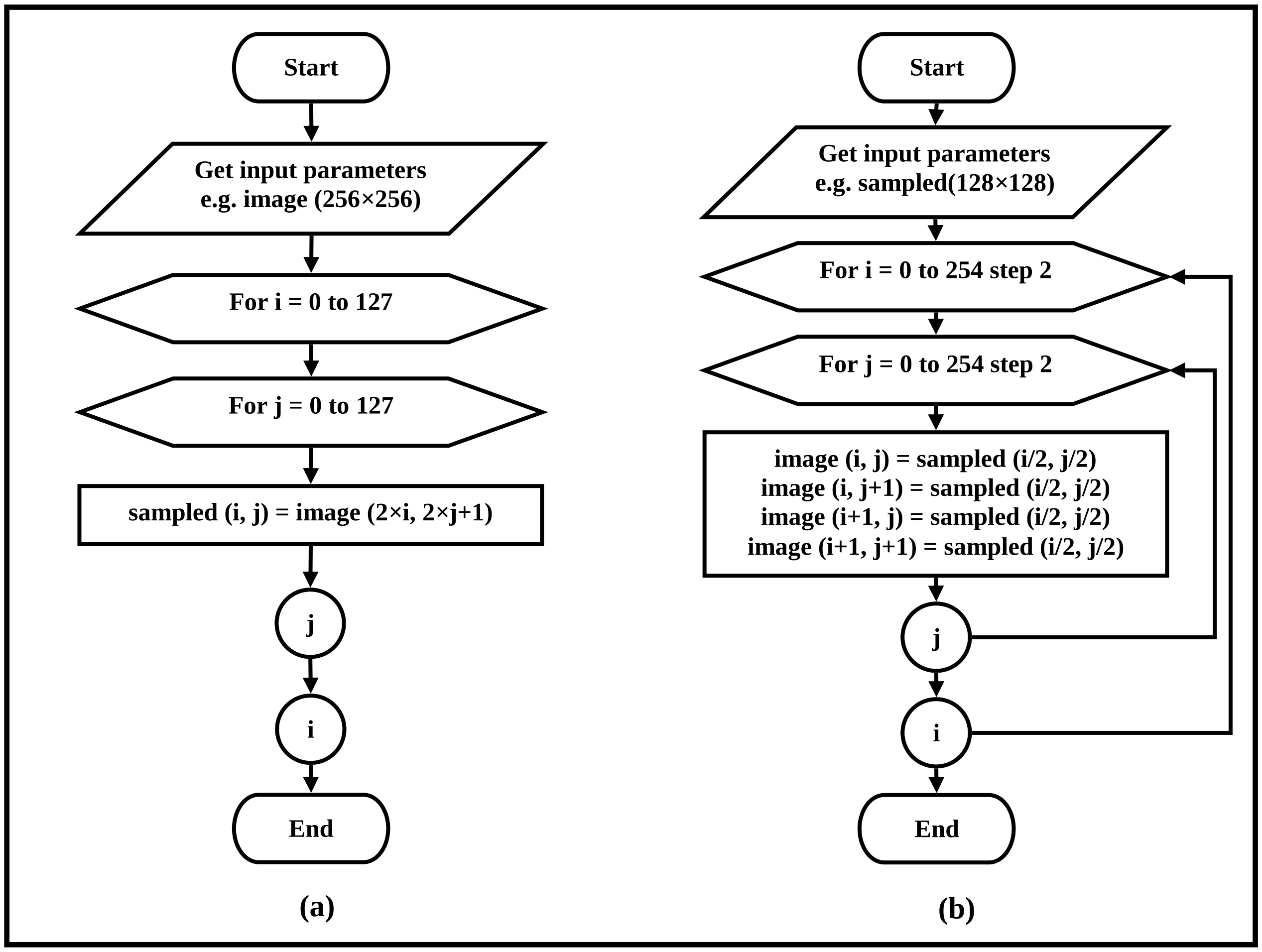


Figure (4-6): Even-Odd sampling
(a) Down-sampling (b) Up-sampling

4.2.5 Odd-Odd Sampling

In this section, the image is down-sampled by removing odd rows and odd columns according to algorithm shown in figure (4-8a). The up-sampling process is done according to algorithm shown in figure (4-8b).

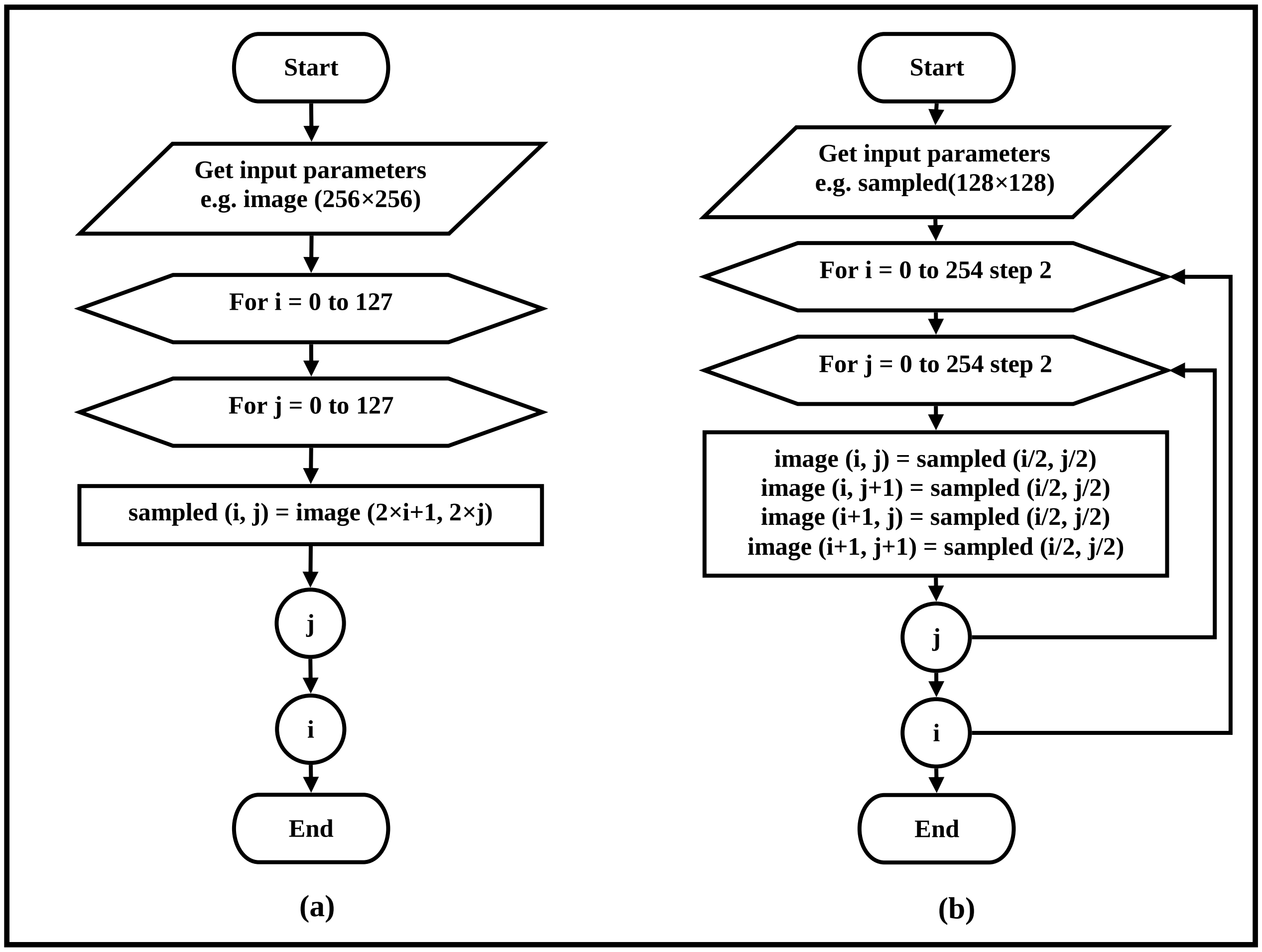


Figure (4-7): Odd-Even sampling
(a) Down-sampling (b) Up-sampling

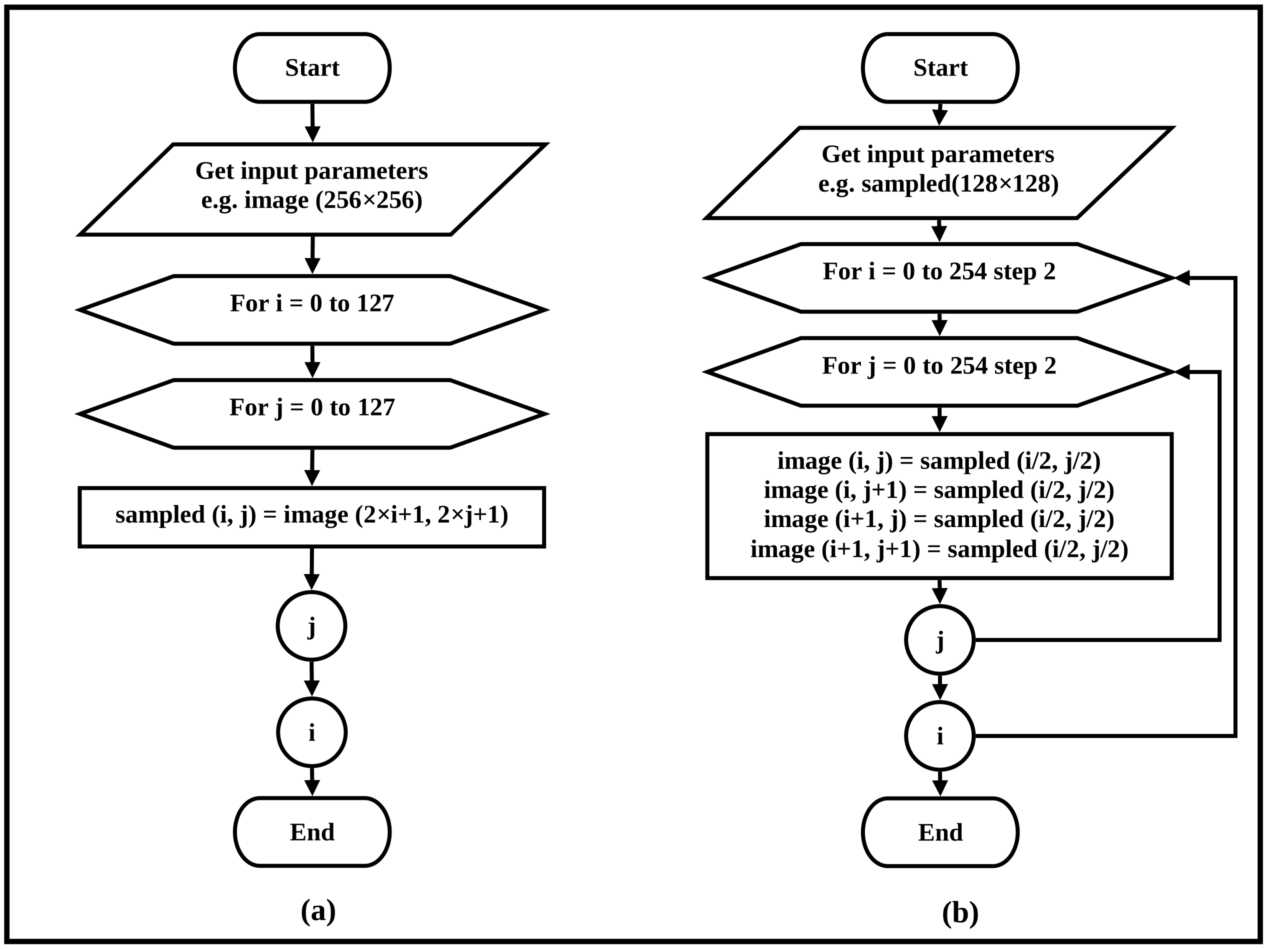


Figure (4-8): Odd-Odd sampling
(a) Down-sampling (b) Up-sampling

4.3 The Partitioning Process

This process is shown in figure (4-9). In this section, the image is partitioned from top to bottom and from left to right to get a small (8×8) blocks as illustrated in figure (4-10a). The process of re-assembling the blocks to get the whole image is illustrated in figure (4-10b).

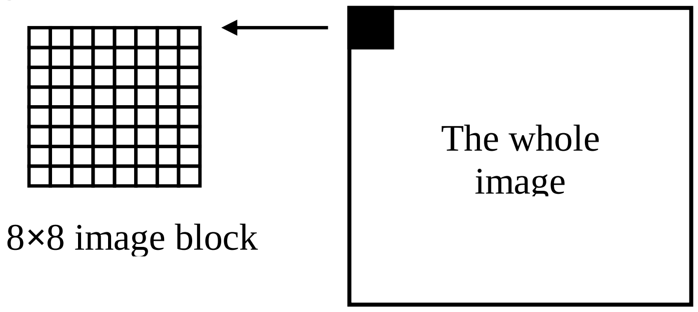


Figure (4-9): Image Partitioning

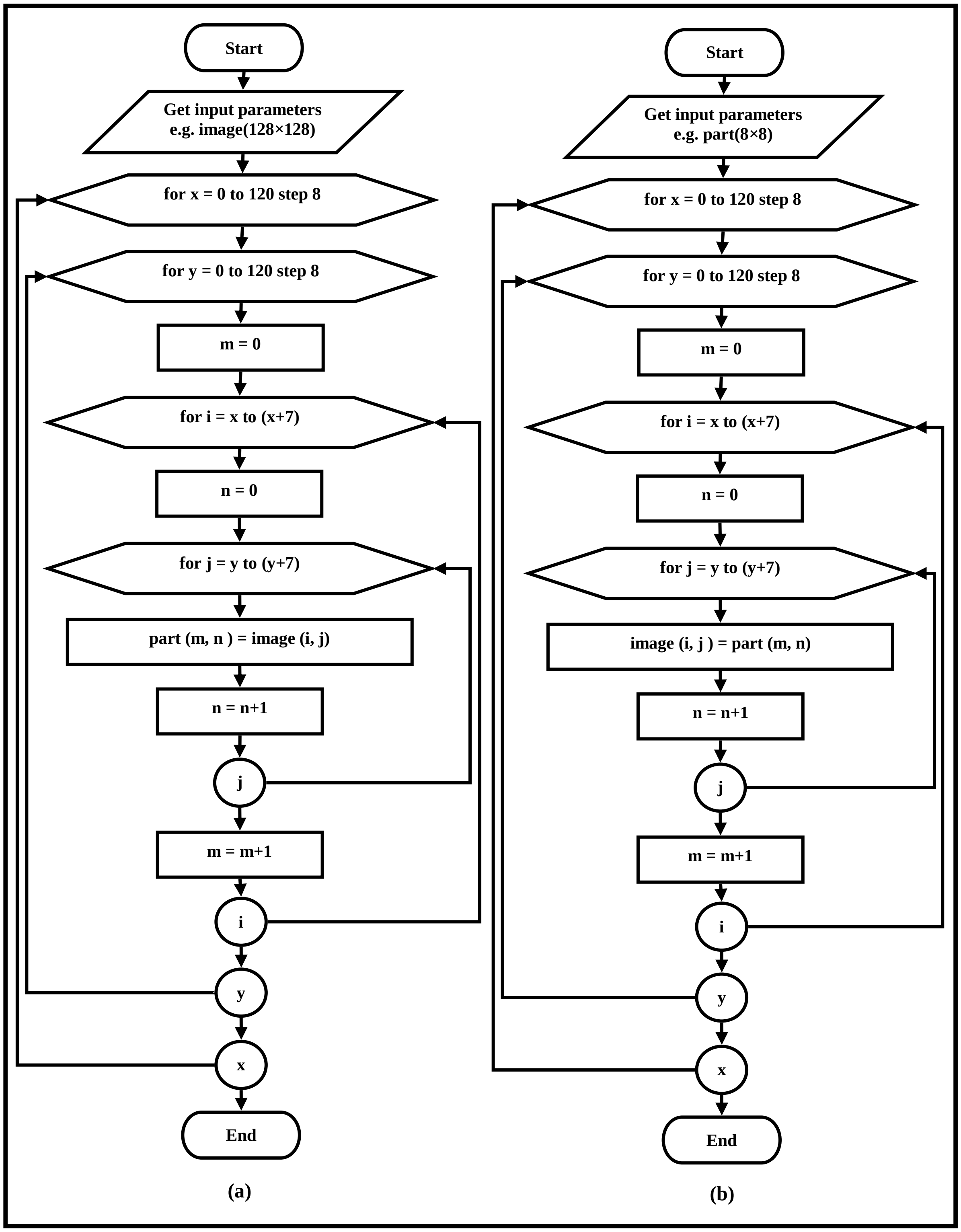


Figure (4-10): Partitioning process
(a) Partitioning (b) Re-assembling

4.4 The 2D Discrete Cosine Transform (2D-DCT)

In this section, the image block is transformed from spatial domain to frequency domain to make block data easy to compress. The forward (2D-DCT) uses equation (2-3) as illustrated in figure (4-11), while the inverse (2D-DCT) uses equation (2-4) as illustrated in figure (4-12).

4.5 The Quantization Process

This section describes two types of quantization that are used in our work. These types are:

4.5.1 Linear Quantization

In linear quantization, the computation of step size depends on a quantization table. The step size can be computed by using equation (2-7).The algorithm illustrating linear quantization is shown in figure (4-13a). In this process, the floating coefficients are rounded to nearest integer. As the quantization process is done, the quantized value is fed into the next stage of compression. The quantization table will be stored in the file to facilitate the dequantization process.

The dequantization process is done by using the algorithm illustrated in figure (4-13b) to get the nearest values of the original values.

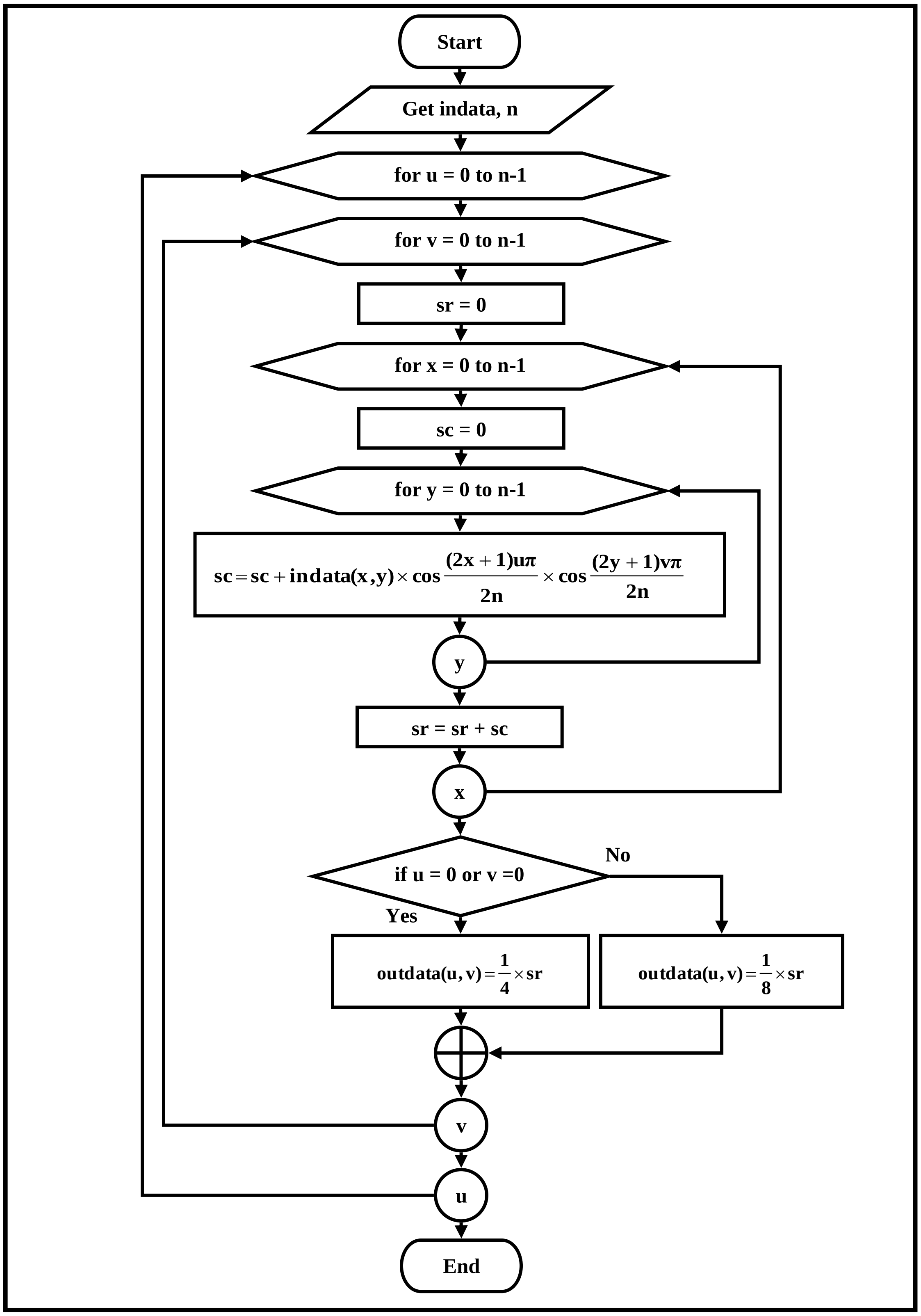

Start
Get indata, n
for u = 0 to n-1
for v = 0 to n-1
sr = 0
for x = 0 to n-1
sc = 0
for y = 0 to n-1
$sc = sc + indata(x,y) \times \cos\frac{(2x+1)u\pi}{2n} \times \cos\frac{(2y+1)v\pi}{2n}$
y
sr = sr + sc
x
if u = 0 or v =0
No
Yes
$outdata(u,v) = \frac{1}{4} \times sr$
$outdata(u,v) = \frac{1}{8} \times sr$
v
u
End

Figure (4-11): Forward 2D-DCT

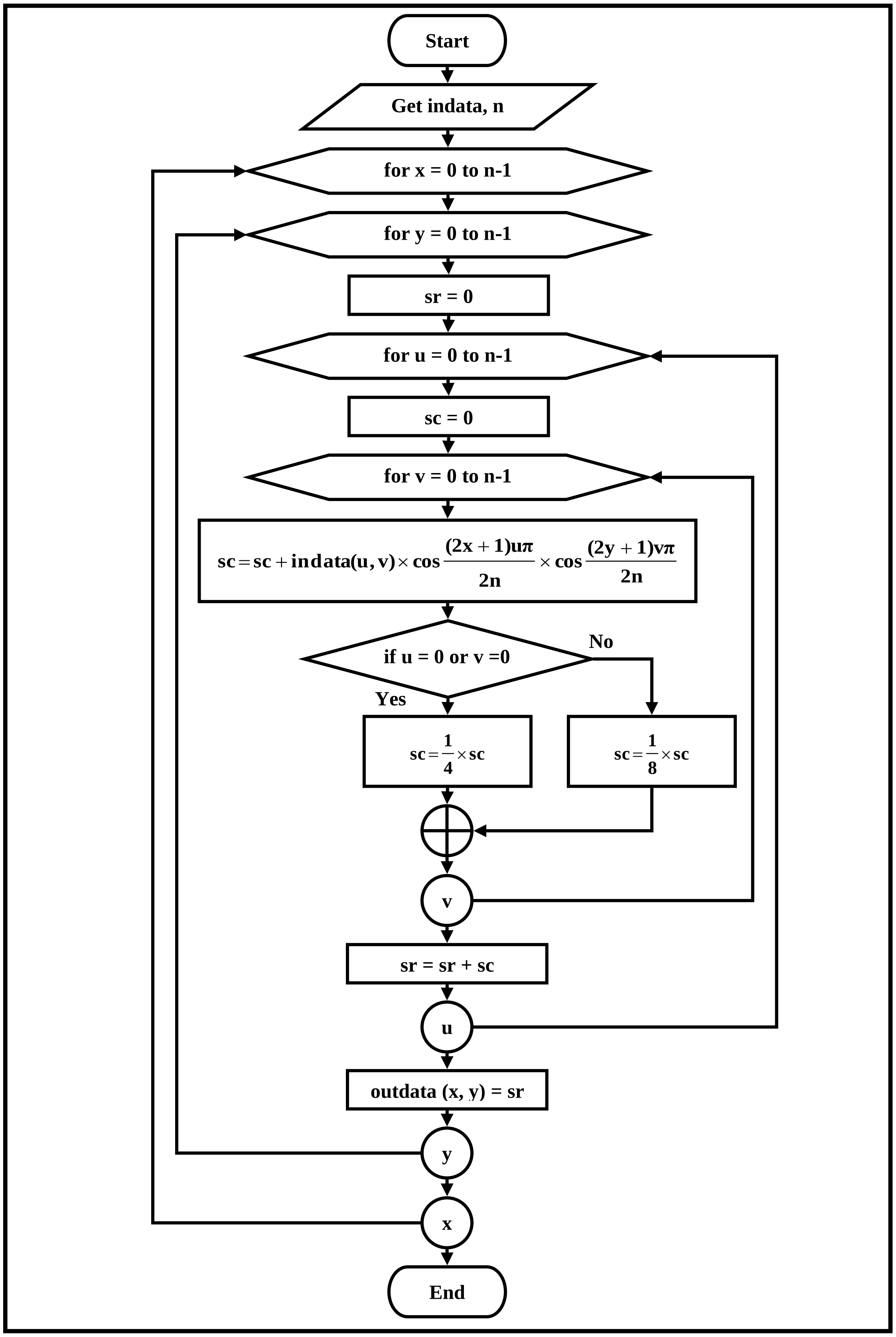


Figure (4 -12): Inverse 2D-DCT

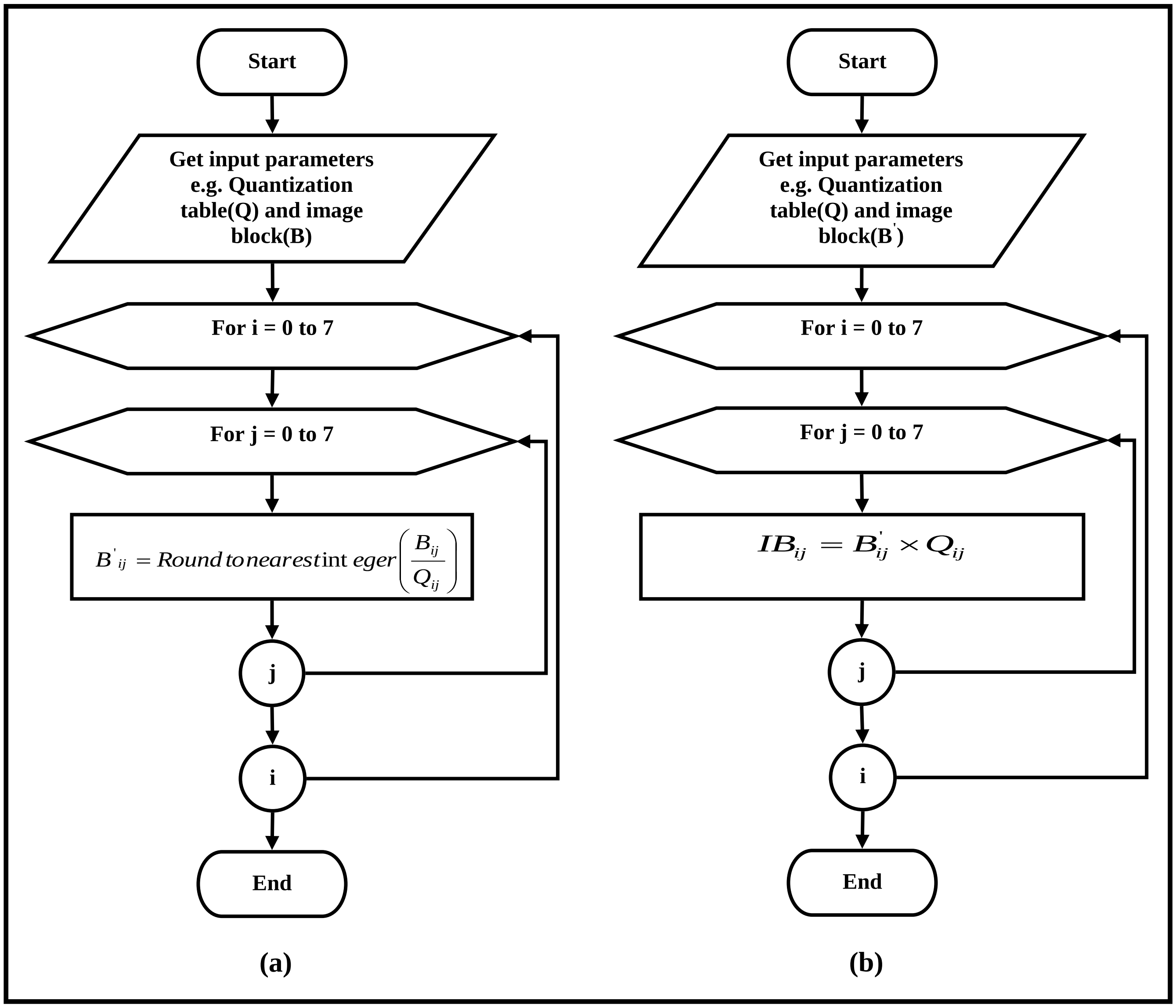

Start
Get input parameters e.g. Quantization table(Q) and image block(B)
For i = 0 to 7
For j = 0 to 7
$B'_{ij} = Round\ to\ nearest\ integer\left(\frac{B_{ij}}{Q_{ij}}\right)$
j
i
End
(a)
Start
Get input parameters e.g. Quantization table(Q) and image block(B')
For i = 0 to 7
For j = 0 to 7
$IB_{ij} = B'_{ij} \times Q_{ij}$
j
i
End
(b)

Figure (4-13): Linear Quantization:

(a) Quantization (b) Dequantization

4.5.2 Uniform Quantization

In uniform quantization, the computation of step size depends on three parameters:

- Maximum value, Max, in the matrix
- Minimum value, Min, in the matrix
- Gray level, G

As these parameters are obtained, the step size can be computed by using equation (2-8). The algorithm illustrating uniform quantization is shown in figure

(4-14a). As in linear quantization, the floating coefficients are rounded to nearest integer. As the quantization process is done, the quantized value is fed into the next stage of compression. The three parameters mentioned above are stored in the file so that the quantization table can be reconstructed in the dequantization process.

The dequantization process is done by using the algorithm illustrated in figure (4-14b) to get the nearest values of the original values.

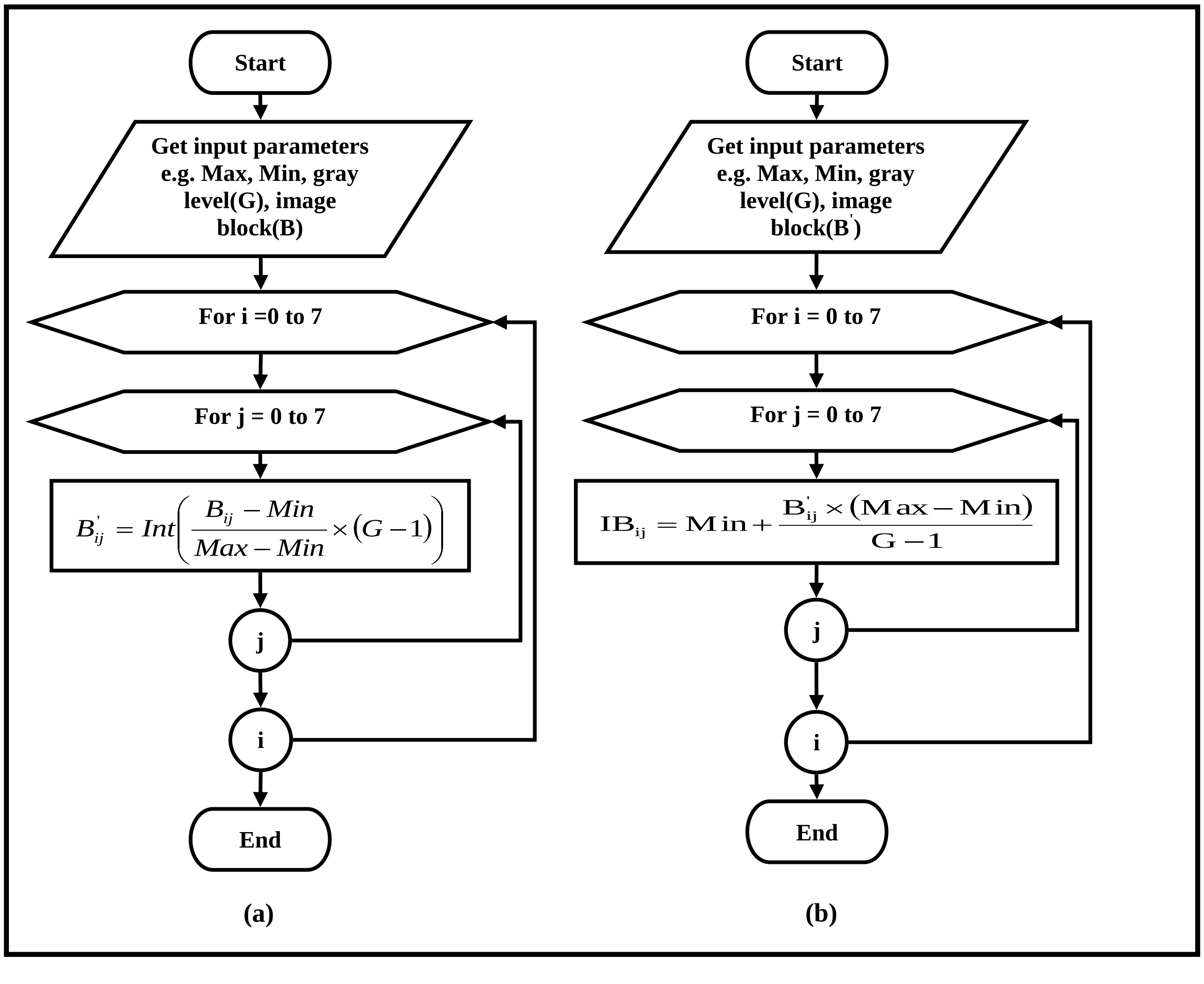

Start
Get input parameters e.g. Max, Min, gray level(G), image block(B)
For i =0 to 7
For j = 0 to 7
$B'_{ij} = Int\left(\frac{B_{ij} - Min}{Max - Min} \times (G-1)\right)$
j
i
End
(a)
Start
Get input parameters e.g. Max, Min, gray level(G), image block(B')
For i = 0 to 7
For j = 0 to 7
$IB_{ij} = Min + \frac{B'_{ij} \times (Max - Min)}{G-1}$
j
i
End
(b)

Figure (4-14): Uniform Quantization:

(a) Quantization (b) Dequantization

4.6 The Conversion Process

In this section, there is a description of how to convert the array from (2D) to (1D) so as to block the same values in one location as much as possible for the new array. There is also a description of three types of conversion that are used in our work. These types are:

4.6.1 Horizontal Conversion

Horizontal conversion is shown in figure (4-15). The algorithm converts (2D) array to (1D) vector by taking the first row followed by the second one such as the first element in the second row becomes after the last element in the first one and so on till the end of the array as illustrated in figure (4-16a). The reverse process, i.e. converting from (1D) vector to (2D) array, is illustrated in figure (4-16b).

1	**.**	**.**	**.**	**.**	**.**	**.**	**8**
9	**.**	**.**	**.**	**.**	**.**	**.**	**16**
17	**.**	**.**	**.**	**.**	**.**	**.**	**24**
25	**.**	**.**	**.**	**.**	**.**	**.**	**32**
33	**.**	**.**	**.**	**.**	**.**	**.**	**40**
41	**.**	**.**	**.**	**.**	**.**	**.**	**48**
49	**.**	**.**	**.**	**.**	**.**	**.**	**56**
57	**.**	**.**	**.**	**.**	**.**	**.**	**64**

2D array (Quantized array)

1 . . . 8	**9 . . . 16**		**57 . . . 64**

1D array (Converted array)

Figure (4-15): Horizontal Converted array

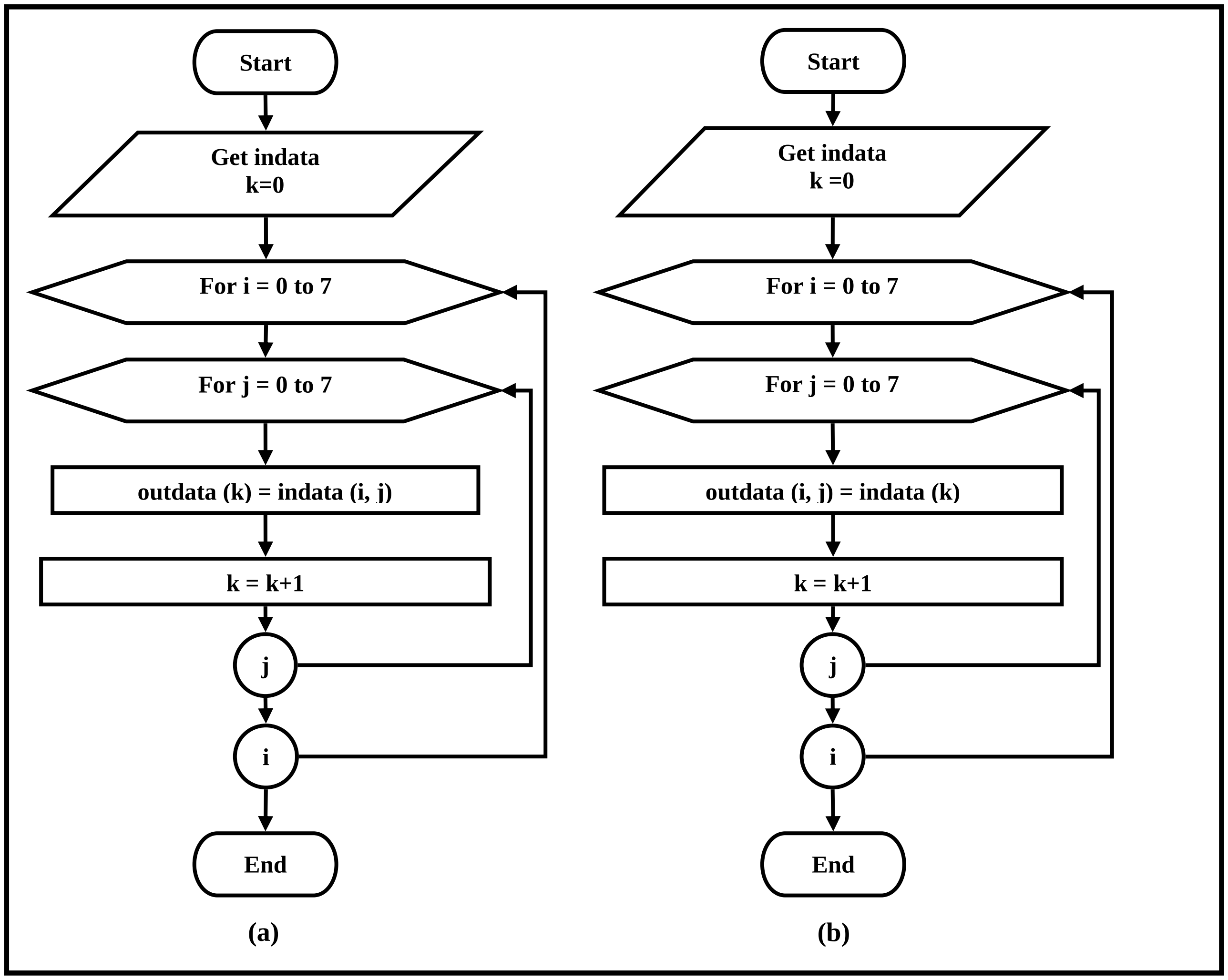


Figure (4-16): Horizontal Conversion:

(a) Conversion from 2D to 1D (b) Conversion from 1D to 2D

4.6.2 Vertical Conversion

Vertical conversion is shown in figure (4-17). The algorithm converts (2D) array to (1D) vector by taking the first column followed by the second one such as the first element in the second column becomes after the last element in the first one and so on till the end of the array as illustrated in figure (4-18a). The reverse process, i.e. converting from (1D) vector to (2D) array, is illustrated in figure (4-18b).

1	**9**	**17**	**25**	**33**	**41**	**49**	**57**
.	**.**	**.**	**.**	**.**	**.**	**.**	**.**
.	**.**	**.**	**.**	**.**	**.**	**.**	**.**
.	**.**	**.**	**.**	**.**	**.**	**.**	**.**
.	**.**	**.**	**.**	**.**	**.**	**.**	**.**
.	**.**	**.**	**.**	**.**	**.**	**.**	**.**
.	**.**	**.**	**.**	**.**	**.**	**.**	**.**
8	**16**	**24**	**32**	**40**	**48**	**56**	**64**

2D array (Quantized array)

1 . . . 8	**9 . . . 16**		**57 . . . 64**

1D array (Converted array)

Figure (4-17): Vertical Converted array

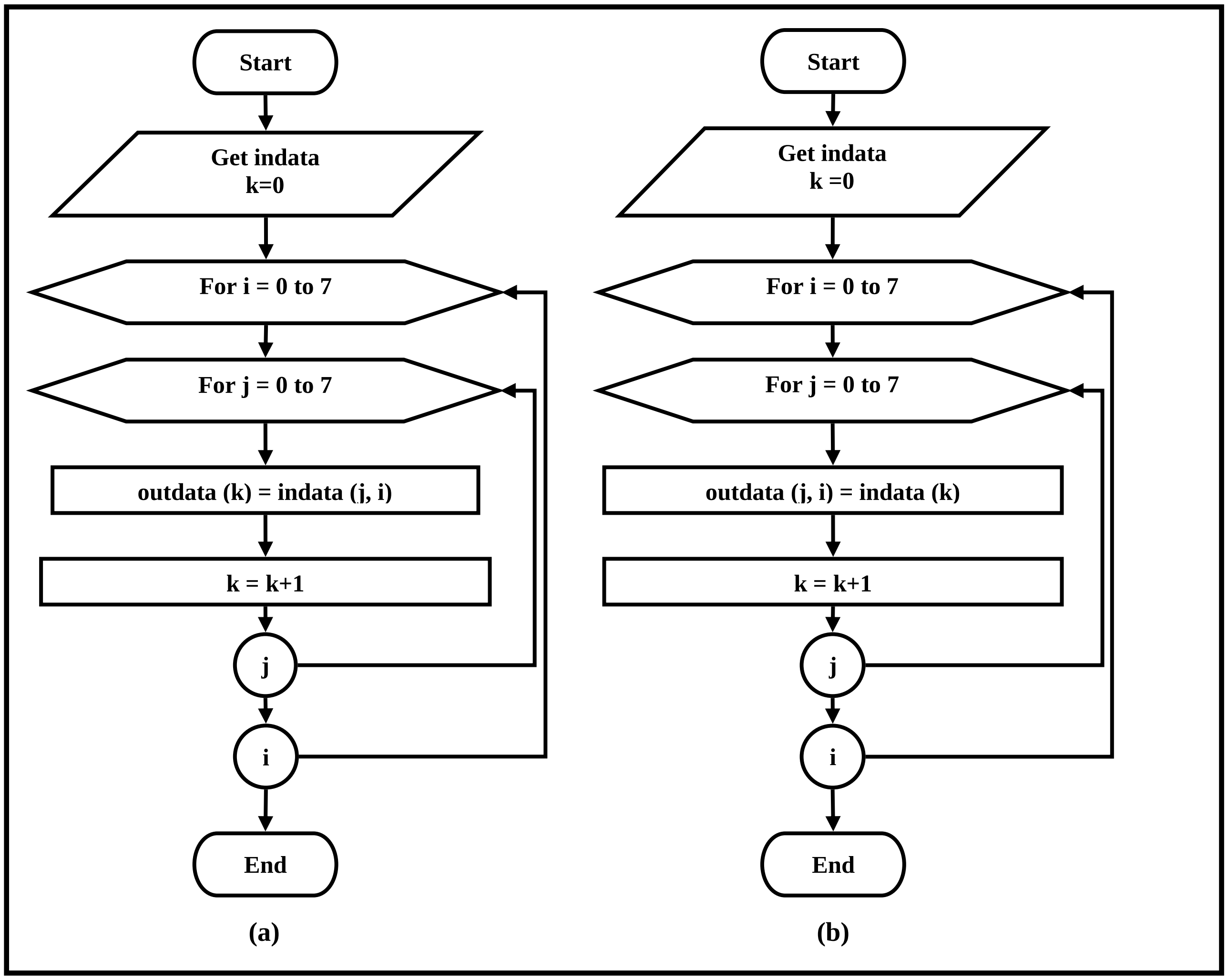


Figure (4-18): Vertical Conversion:

(a) Conversion from 2D to 1D (b) Conversion from 1D to 2D

4.6.3 Zig-Zag Conversion

Zig-Zag conversion is shown in figure (4-19). The algorithm converts (2D) array to (1D) vector by taking the elements of the array in the zig-zag order beginning from the first element of the array and ending at the last element of the array as illustrated in figure (4-20a). The conversion from (1D) to (2D) is illustrated in figure (4-20b).

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
9	**10**	**11**	**12**	**13**	**14**	**15**	**16**
17	**18**	**19**	**20**	**21**	**22**	**23**	**24**
25	**26**	**27**	**28**	**29**	**30**	**31**	**32**
33	**34**	**35**	**36**	**37**	**38**	**39**	**40**
41	**42**	**43**	**44**	**45**	**46**	**47**	**48**
49	**50**	**51**	**52**	**53**	**54**	**54**	**55**
57	**58**	**59**	**60**	**61**	**62**	**63**	**64**

2D array (Quantized array)

1	**2**	**6**	**7**	**15**	**16**	**28**	**29**
3	**5**	**8**	**14**	**17**	**27**	**30**	**43**
4	**9**	**13**	**18**	**26**	**31**	**42**	**44**
10	**12**	**19**	**25**	**32**	**41**	**45**	**54**
11	**20**	**24**	**33**	**40**	**46**	**53**	**55**
21	**23**	**34**	**39**	**47**	**52**	**56**	**61**
22	**35**	**38**	**48**	**51**	**57**	**60**	**62**
36	**37**	**49**	**50**	**58**	**59**	**63**	**64**

Zig-Zag order for the quantized array

Figure (4-19): Zig-Zag Converted array

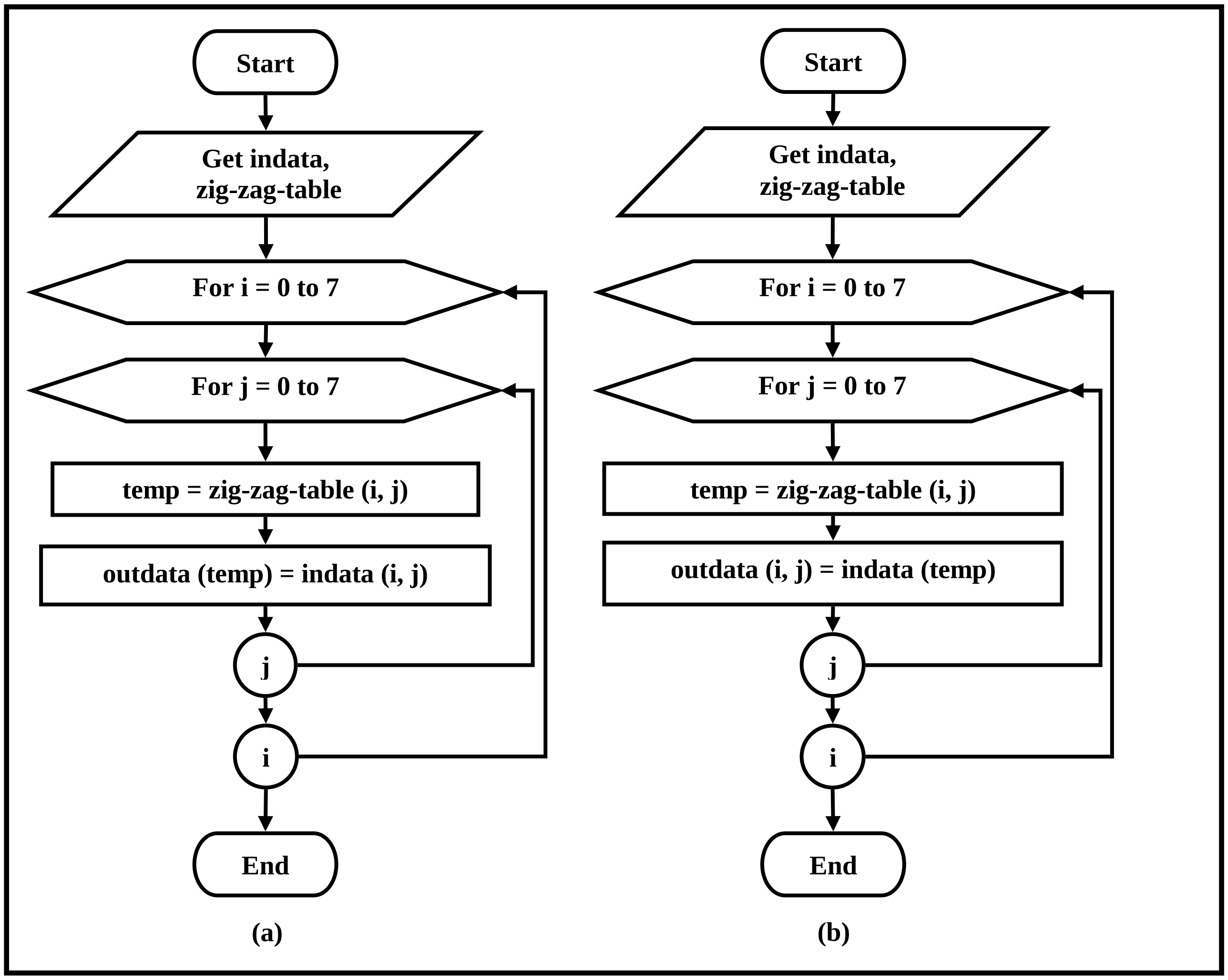

Start
Get indata, zig-zag-table
For i = 0 to 7
For j = 0 to 7
temp = zig-zag-table (i, j)
outdata (temp) = indata (i, j)
j
i
End
(a)
Start
Get indata, zig-zag-table
For i = 0 to 7
For j = 0 to 7
temp = zig-zag-table (i, j)
outdata (i, j) = indata (temp)
j
i
End
(b)

Figure (4-20): Zig-Zag Conversion:

(a) Conversion from 2D to 1D (b) Conversion from 1D to 2D

4.7 Entropy Encoding

In this section, the (1D)-vector is converted to intermediate representation as shown in figure (4-21). The first element in the vector which is called (DC) can be calculated by taking the difference between it and (DC) element of the previous block except that (DC) element of the first block stays as it is. Then it is represented by using two symbols. Symbol1 represents the ***size*** of bits needed to represent the ***value*** of (DC) element. Symbol2 represents the actual bits of the ***value*** as shown in figure (4-22) and table (4-1). The remaining elements which are called (AC) can be represented by using (RLE). They are also represented by using two symbols. Symbol1 consists of two parts. Part1 represents the number of consecutive zeros (***Run***) proceeding a non-zero element while part2 represents the ***size*** of bits needed to represent the ***value*** of (AC) element. Symbol2 represents the actual bits of the ***value*** as shown in figure (4-23) and table (4-2).

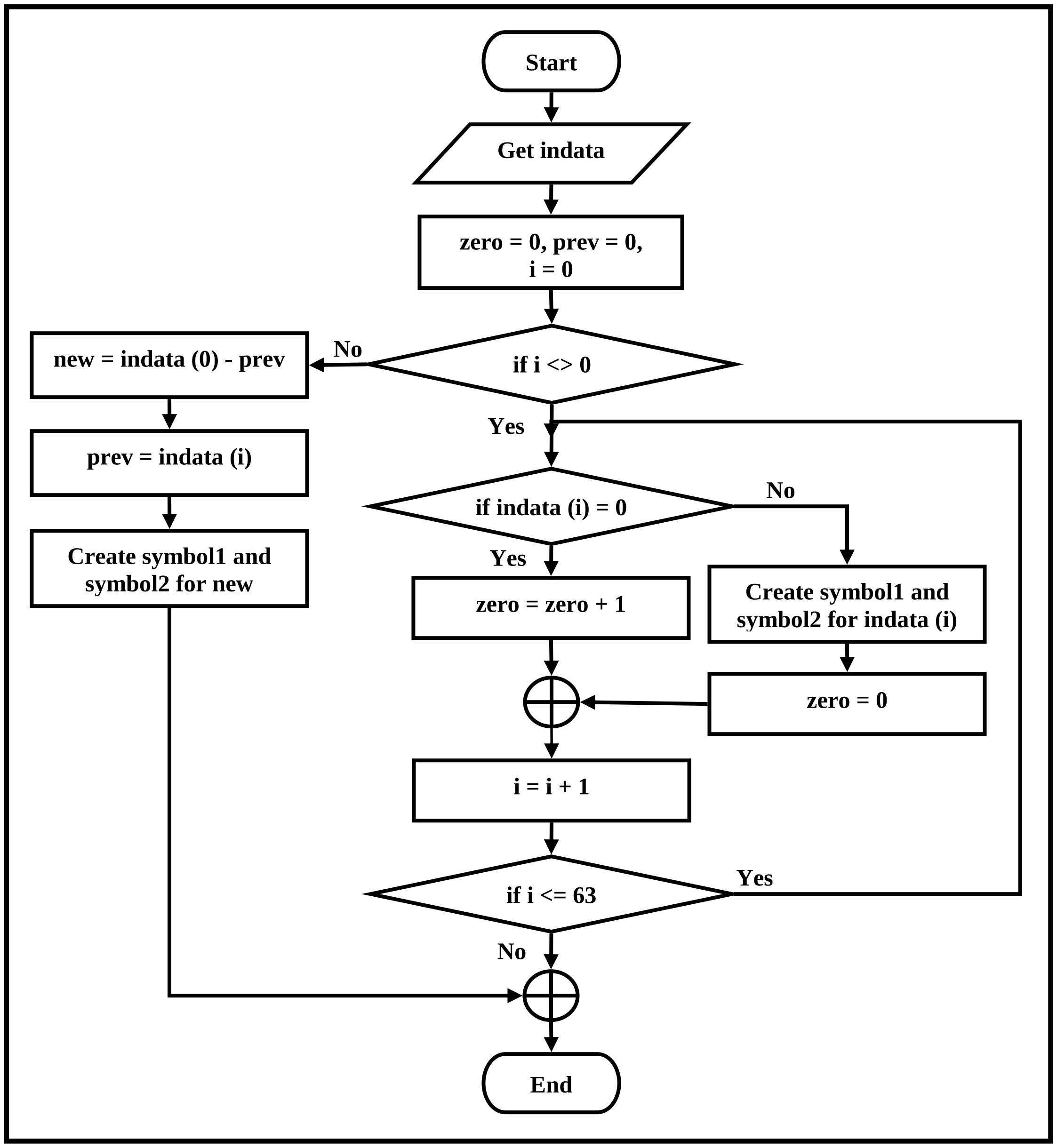


Figure (4-21): Entropy Encoding

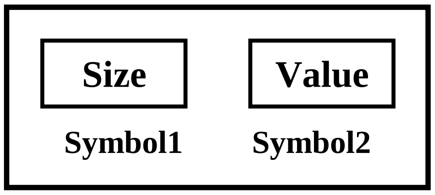


Figure (4-22): Symbol1 and Symbol2 for DC element

Table (4-1): Bits needed to represent symbol1 and symbol2 for DC element

Size	DC Values	**Actual bits**
0	**-**	**0**
1	**0,1**	**1**
2	**-3,-2,2,3**	**2**
3	**-7,…,-4,4,…7**	**3**
4	**-15,…,-8,8,…,15**	**4**
5	**-31,…,-16,16,…,31**	**5**
6	**-63,…,-32,32,…,63**	**6**
7	**-127,…,-64,64,…,127**	**7**
8	**-255,…,-128,128,…,255**	**8**
9	**-511,…,-256,256,…,511**	**9**
10	**-1023,…,-512,512,…,1023**	**10**
11	**-2047,…,-1024,1024,…,2047**	**11**

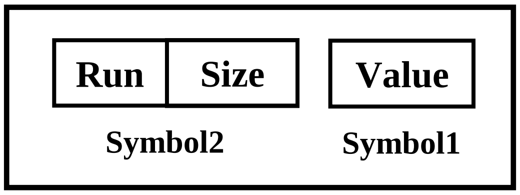


Figure (4-23): Symbol1 and Symbol2 for AC element

Table (4-2): Bits needed to represent symbol1 and symbol2 for AC element

Size	AC Values	**Actual bits**
0	**-**	**0**
1	**0,1**	**1**
2	**-3,-2,2,3**	**2**
3	**-7,…,-4,4,…7**	**3**
4	**-15,…,-8,8,…,15**	**4**
5	**-31,…,-16,16,…,31**	**5**
6	**-63,…,-32,32,…,63**	**6**
7	**-127,…,-64,64,…,127**	**7**
8	**-255,…,-128,128,…,255**	**8**
9	**-511,…,-256,256,…,511**	**9**
10	**-1023,…,-512,512,…,1023**	**10**

4.8 Huffman Coding

This is the final section in which the intermediate representation of the (1D)-vector, i.e. symbol1 and symbol2 for (DC) and (AC) elements, is encoded using Huffman tables (see appendix A). The encoding process converts symbol1 and symbol2 for (DC) and (AC) elements into a string of bits as shown in figure (4-24), while the decoding process is used to retrieve symbol1 and symbol2 for (DC) and (AC) elements from a string of bits as shown in figure (4-25).

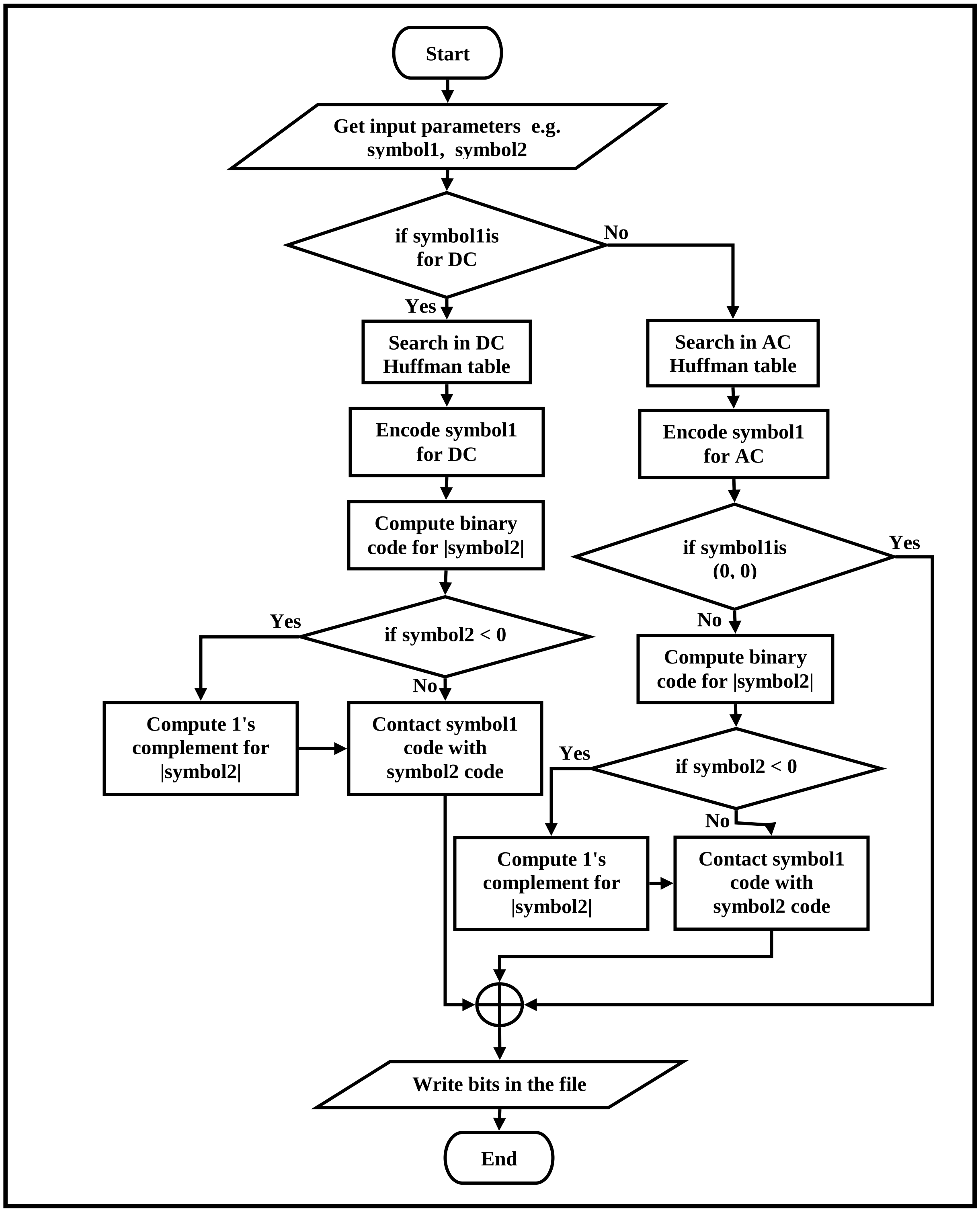

Start
Get input parameters e.g. symbol1, symbol2
if symbol1is for DC
No
Yes
Search in DC Huffman table
Encode symbol1 for DC
Compute binary code for |symbol2|
if symbol2 < 0
Yes
No
Compute 1's complement for |symbol2|
Contact symbol1 code with symbol2 code
Search in AC Huffman table
Encode symbol1 for AC
if symbol1is (0, 0)
Yes
No
Compute binary code for |symbol2|
if symbol2 < 0
Yes
No
Compute 1's complement for |symbol2|
Contact symbol1 code with symbol2 code
Write bits in the file
End

Figure (4-24): Huffman Encoding

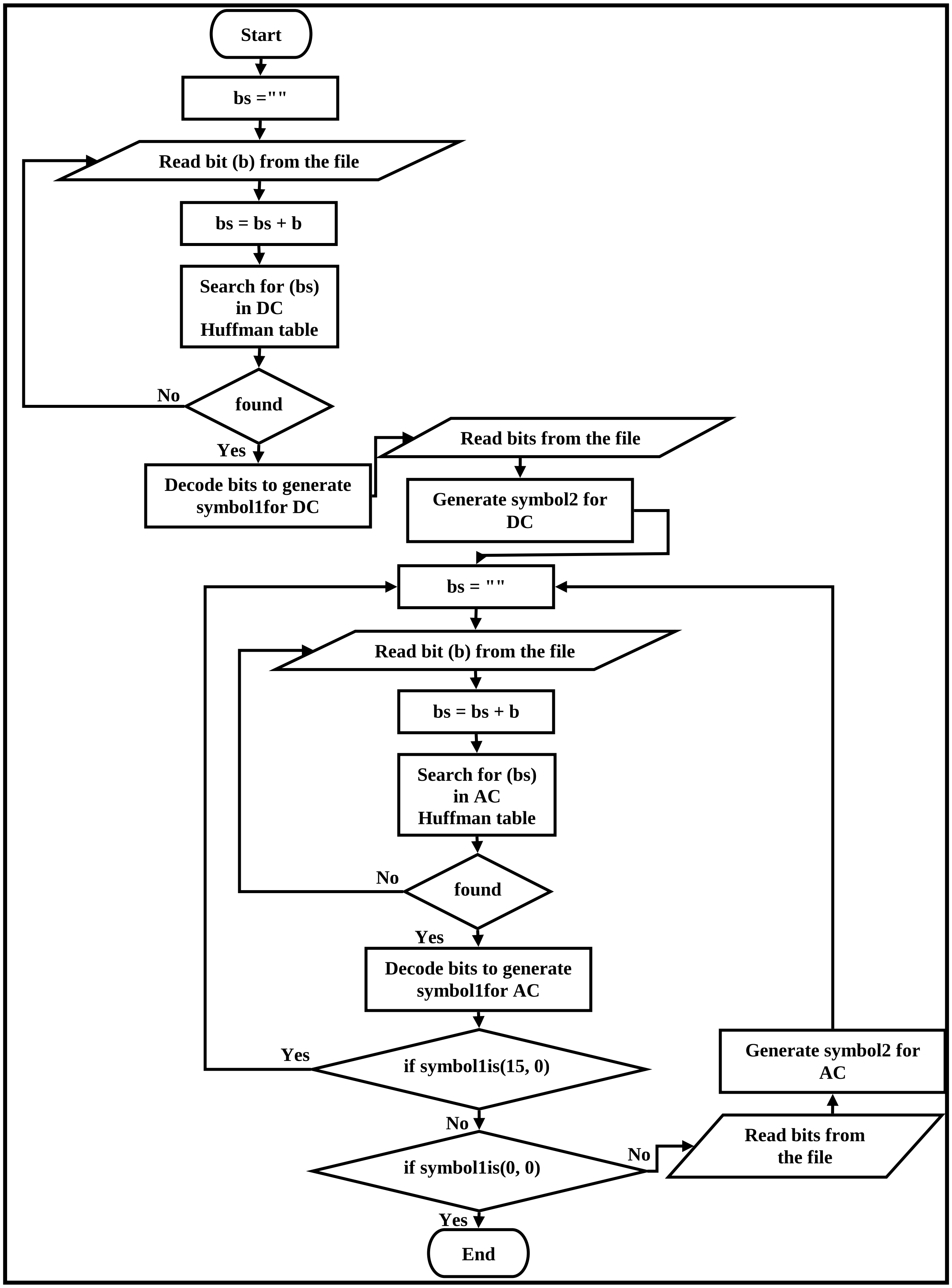


Figure (4-25): Huffman Decoding

4.9 The Format of LBJS Image File

The file format consists of two main parts:

1 The Header

The header of the file contains the information that is used to reconstruct the compressed image as shown in table (4-3).

Table (4-3): The header of LBJS image file

File header	Type	Size(Bytes)	Contents
Sampling Type	Byte	1	1: Classical. 2: Even-Even. 3: Even-Odd. 4: Odd-Even. 5: Odd-Odd.
Quantization Type	Byte	1	1: Linear. 2: Uniform.
Maximum	Integer	2	Maximum value in the data.
Minimum	Integer	2	Minimum value in the data.
Conversion Type	Byte	1	1: Horizontal. 2: Vertical. 3: Zigzag.
Encoding Type	String*3	3	Huf: Huffman Coding

2 The Data

The coded image data is saved after the header of the file.

4.10 Simulation of LBJS System

In our system, we use the configurations shown in table (4-4).

Table (4-4): System Configuration

Item	Sending Computer	Receiving Computer
Processor Speed	1.8 GHZ.	1.8 GHZ.
Memory	256 MB.	256 MB.
System	Windows XP Professional.	Windows XP Professional.
Language	Visual Basic 6.	Visual Basic 6.
Parameters	Sending Computer	Receiving Computer
Image Type	Gray Scale Image.	LBJS Image.
Image Size	256 × 256.	LBJS File Size.
Sampling Type	Classical, Even-Even, Even-Odd, Odd-Even or Odd-Odd.	Used In Compression.
Quantization Type	Linear or Uniform.	Used In Compression
Conversion Type	Horizontal, Vertical or Zigzag.	Used In Compression.
Compression Algorithm	Huffman Algorithm.	Huffman Algorithm.

4.11 The Results

In this section, we will discuss the results obtained by using the five types of sampling (Classical, Even-Even, Even-Odd, Odd-Even and Odd-Odd), the two types of quantization (Linear and Uniform) and the three types of conversion (Horizontal, Vertical and Zigzag) to calculate the C.R. PSNR and Transmission time. This discussion will help us to decide which sampling type, quantization type and conversion type are the best according to compression quality, and transmission time needed to transmit the compressed image file.

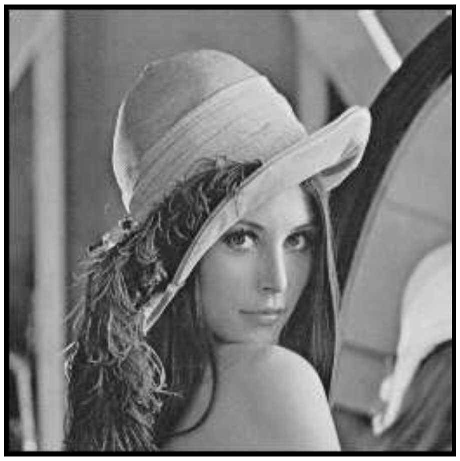
4.11.1 Sampling Process

I. Classical Sampling

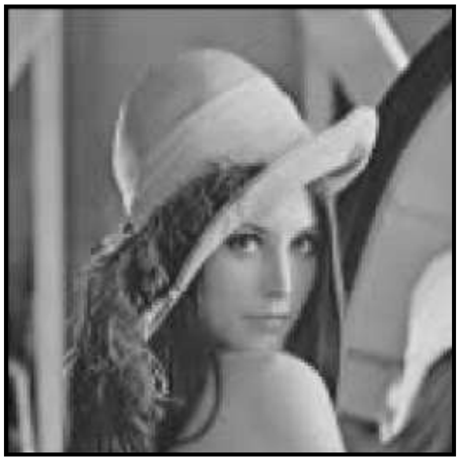
In this section, the effects of using Classical sampling with each type of quantization and each type of conversion will be discussed. These effects are shown in table (4-5) and figure (4-26).

Table (4-5): Classical Sampling on Lena Image

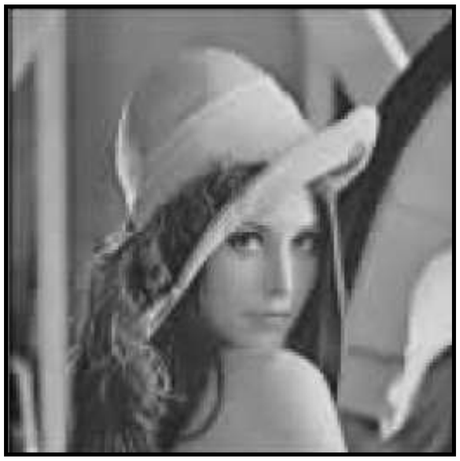
Picture	Quantization	Horizontal Conversion				
Lena 256×256 65 KB	Linear	Quality (Q)	New File Size (KB)	Transmission Time (Sec.)	C.R.	PSNR
		1	8.59	4.98	7.57	24.71
		2	5.80	3.36	11.21	24.37
		3	4.40	2.55	14.77	24.03
		4	3.65	2.12	17.81	23.78
		5	3.09	1.79	21.04	23.50
	Uniform	None	2.39	1.39	27.20	21.88
	Quantization	Vertical Conversion				
	Linear	Quality (Q)	New File Size (KB)	Transmission Time (Sec.)	C.R.	PSNR
		1	7.79	4.52	8.34	24.71
		2	5.25	3.05	12.38	24.37
		3	3.97	2.30	16.37	24.03
		4	3.27	1.90	19.88	23.78
		5	2.76	1.60	23.55	23.50
	Uniform	None	2.10	1.22	30.95	21.88
	Quantization	Zigzag Conversion				
	Linear	Quality (Q)	New File Size (KB)	Transmission Time (Sec.)	C.R.	PSNR
		1	6.89	4	9.43	24.71
		2	4.41	2.56	14.74	24.37
		3	3.22	1.87	20.19	24.03
		4	2.61	1.51	24.90	23.78
		5	2.15	1.25	30.23	23.50
	Uniform	None	1.93	1.12	33.68	21.88



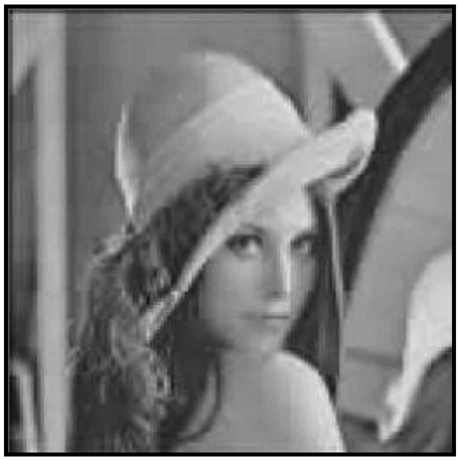
(a)
"Original Image"
256×256



(b)
C.R.: 11.21
T.T.: 3.36
PSNR: 24.37



(c)
C.R.: 16.37
T.T.: 2.30
PSNR: 24.03



(d)
C.R.: 24.90
T.T.: 1.51
PSNR: 23.78

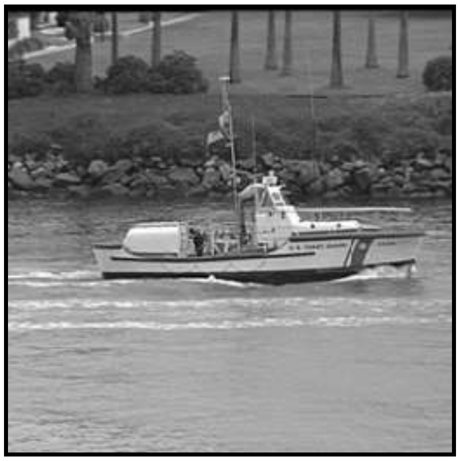
Figure (4-26): Applying Classical Sampling on Lena Image
a) Original Image b) Linear-Horizontal (Q=2)
c) Linear-Vertical (Q=3) d) Linear-Zigzag (Q=4)

II. Even-Even Sampling

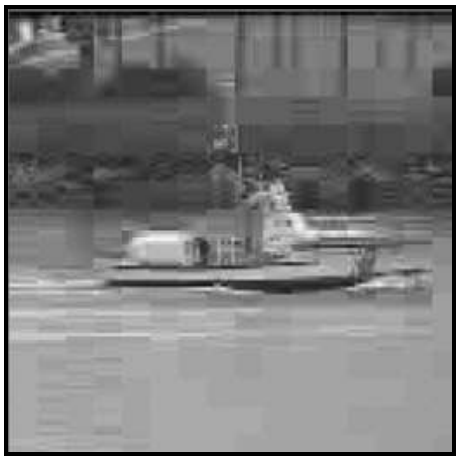
In this section, the effects of using Even-Even sampling with each type of quantization and each type of conversion will be discussed. These effects are shown in table (4-6) and figure (4-27).

Table (4-6): Even-Even Sampling on Boat Image

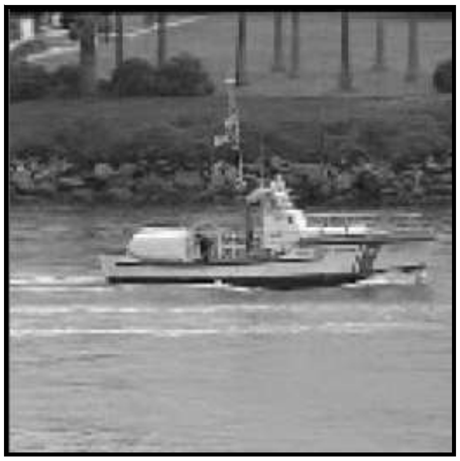
Picture	Quantization	Horizontal Conversion				
Boat 256×256 65 KB	Linear	Quality (Q)	New File Size (KB)	Transmission Time (Sec.)	C.R.	PSNR
		1	8.30	4.81	7.83	24.02
		2	4.79	2.78	13.57	24.05
		3	3.40	1.97	19.12	24
		4	2.57	1.49	25.29	23.91
		5	2.14	1.24	30.37	23.81
	Uniform	None	1.32	0.77	49.24	23.27
	Quantization	Vertical Conversion				
	Linear	Quality (Q)	New File Size (KB)	Transmission Time (Sec.)	C.R.	PSNR
		1	9.96	5.78	6.53	24.02
		2	6.24	3.62	10.42	24.05
		3	4.66	2.70	13.95	24
		4	3.58	2.08	18.16	23.91
		5	3.02	1.75	21.52	23.81
	Uniform	None	1.94	1.13	33.51	23.27
	Quantization	Zigzag Conversion				
	Linear	Quality (Q)	New File Size (KB)	Transmission Time (Sec.)	C.R.	PSNR
		1	8.50	4.93	7.65	24.02
		2	4.99	2.89	13.03	24.05
		3	3.55	2.06	18.31	24
		4	2.64	1.53	24.62	23.91
		5	2.17	1.26	29.95	23.81
	Uniform	None	1.52	0.88	42.76	23.27



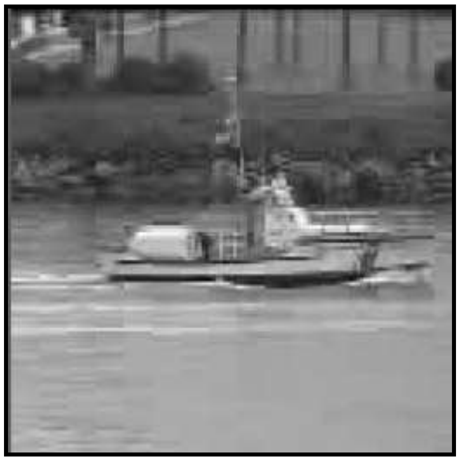
(a)
"Original Image"
256×256



(b)
C.R.: 49.24
T.T.: 0.77
PSNR: 23.27



(c)
C.R.: 6.53
T.T.: 5.78
PSNR: 24.02



(d)
C.R.: 29.95
T.T.: 1.26
PSNR: 23.81

Figure (4-27): Applying Even-Even Sampling on Boat Image

a) Original Image b) Uniform-Horizontal

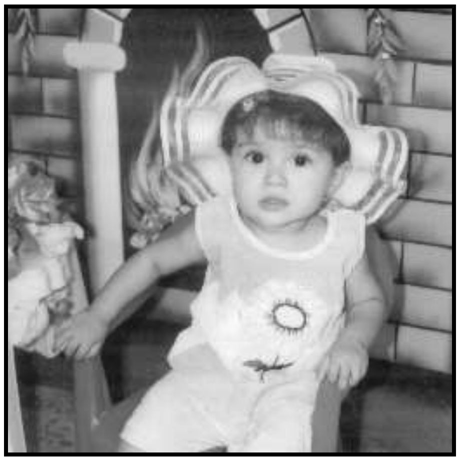
c) Linear-Vertical (Q=1) d) Linear-Zigzag (Q=5)

III. Even-Odd Sampling

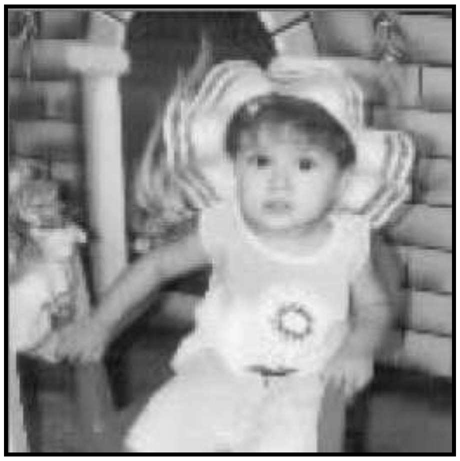
In this section, the effects of using Even-Odd sampling with each type of quantization and each type of conversion will be discussed. These effects are shown in table (4-7) and figure (4-28).

Table (4-7): Even-Odd Sampling on Teeba Image

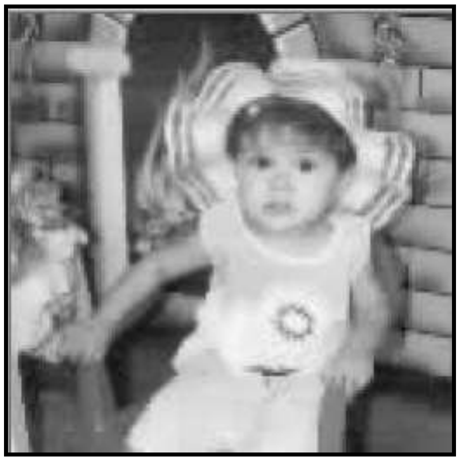
Picture	Quantization	Horizontal Conversion				
Teeba 256×256 65 KB	Linear	Quality (Q)	New File Size (KB)	Transmission Time (Sec.)	C.R.	PSNR
		1	10	5.80	6.5	26.14
		2	6.53	3.79	9.95	26.05
		3	4.89	2.84	13.29	25.81
		4	3.94	2.29	16.50	25.57
		5	3.35	1.94	19.40	25.33
	Uniform	None	3.12	1.81	20.83	20.77
	Quantization	Vertical Conversion				
	Linear	Quality (Q)	New File Size (KB)	Transmission Time (Sec.)	C.R.	PSNR
		1	9.48	5.50	6.86	26.14
		2	6.27	3.64	10.37	26.05
		3	4.65	2.70	13.98	25.81
		4	3.75	2.18	17.33	25.57
		5	3.20	1.86	20.31	25.33
	Uniform	None	2.96	1.72	21.96	20.77
	Quantization	Zigzag Conversion				
	Linear	Quality (Q)	New File Size (KB)	Transmission Time (Sec.)	C.R.	PSNR
		1	8.09	4.69	8.03	26.14
		2	4.95	2.87	13.13	26.05
		3	3.60	2.09	18.06	25.81
		4	2.83	1.64	22.97	25.57
		5	2.37	1.38	27.43	25.33
	Uniform	None	2.66	1.54	24.44	20.77



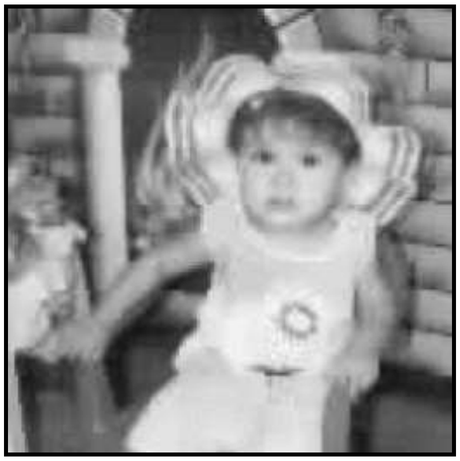
(a)
"Original Image"
256×256



(b)
C.R.: 9.95
T.T.: 3.79
PSNR: 26.05



(c)
C.R.: 13.98
T.T.: 2.70
PSNR: 25.81



(d)
C.R.: 22.97
T.T.: 1.64
PSNR: 25.57

Figure (4-28): Applying Even-Odd Sampling on Teeba Image

a) Original Image b) Linear-Horizontal (Q=2)

c) Linear-Vertical (Q=3) d) Linear-Zigzag (Q=4)

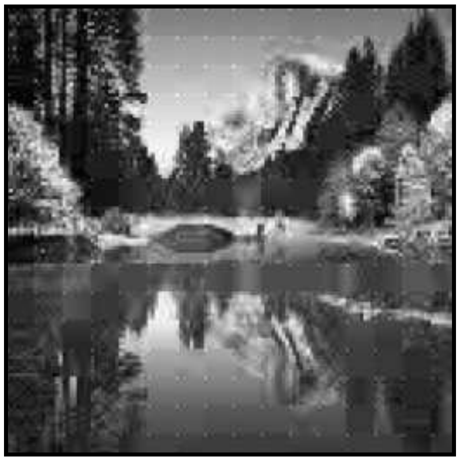
IV. Odd-Even Sampling

In this section, the effects of using Odd-Even sampling with each type of quantization and each type of conversion will be discussed. These effects are shown in table (4-8) and figure (4-29).

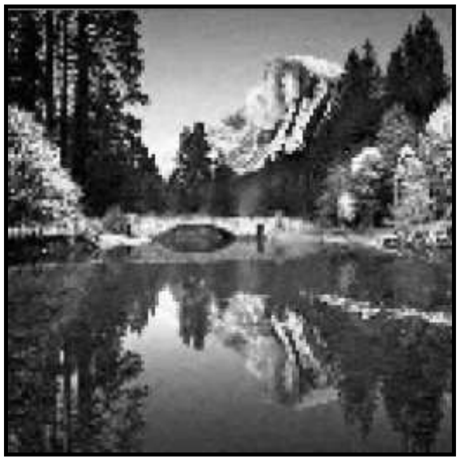
Table (4-8): Odd-Even Sampling on Nature Image

Picture	Quantization	Horizontal Conversion				
Nature 256×256 65 KB	Linear	Quality (Q)	New File Size (KB)	Transmission Time (Sec.)	C.R.	PSNR
		1	14.8	8.58	4.39	18.29
		2	10.5	6.09	6.19	18.47
		3	7.84	4.55	8.29	18.59
		4	6.24	3.62	10.42	18.64
		5	5.10	2.96	12.75	18.65
	Uniform	None	4.26	2.47	15.26	18.41
	Quantization	Vertical Conversion				
	Linear	Quality (Q)	New File Size (KB)	Transmission Time (Sec.)	C.R.	PSNR
		1	14.9	8.64	4.36	18.29
		2	10.5	6.09	6.19	18.47
		3	7.81	4.53	8.32	18.59
		4	6.15	3.57	10.57	18.64
		5	5.05	2.93	12.87	18.65
	Uniform	None	4.19	2.43	15.51	18.41
	Quantization	Zigzag Conversion				
	Linear	Quality (Q)	New File Size (KB)	Transmission Time (Sec.)	C.R.	PSNR
		1	13.7	7.95	4.75	18.29
		2	9.31	5.40	6.98	18.47
		3	6.76	3.92	9.62	18.59
		4	5.26	3.05	12.36	18.64
		5	4.20	2.44	15.48	18.65
	Uniform	None	3.95	2.29	16.46	18.41

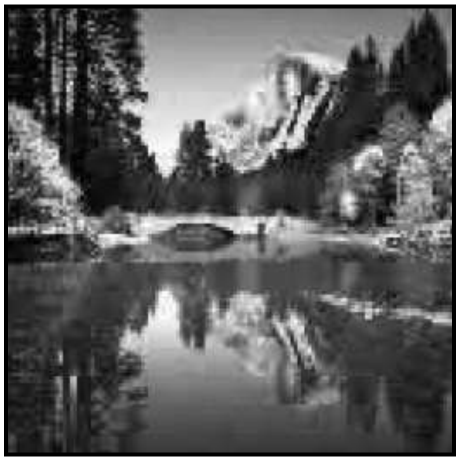
(a)
"Original Image"
256×256



(b)
C.R.: 15.26
T.T.: 2.47
PSNR: 18.41



(c)
C.R.: 4.36
T.T.: 8.64
PSNR: 18.29



(d)
C.R.: 15.48
T.T.: 2.44
PSNR: 18.65

Figure (4-29): Applying Odd-Even Sampling on Nature Image

a) Original Image b) Uniform-Horizontal

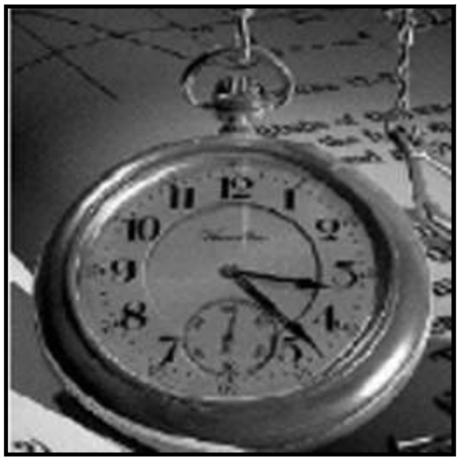
c) Linear-Vertical (Q=1) d) Linear-Zigzag (Q=5)

V. Odd-Odd Sampling

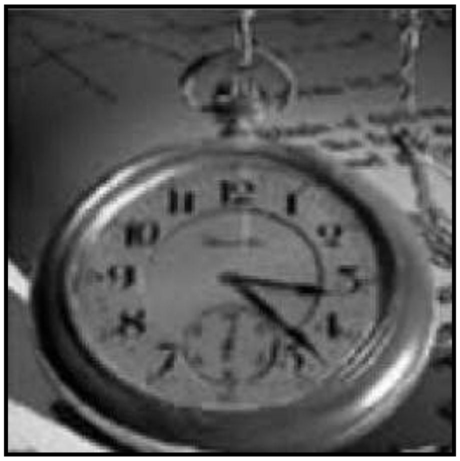
In this section, the effects of using Odd-Odd sampling with each type of quantization and each type of conversion will be discussed. These effects are shown in table (4-9) and figure (4-30).

Table (4-9): Odd-Odd Sampling on Watch Image

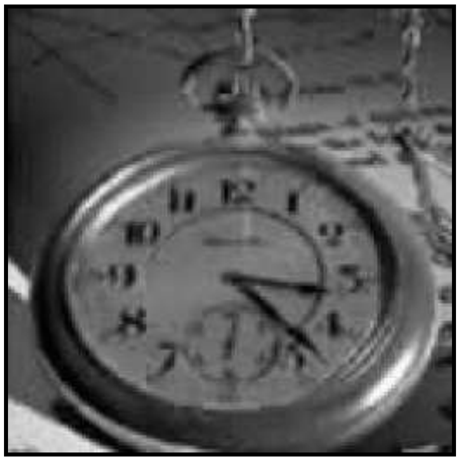
Picture	Quantization	Horizontal Conversion				
Watch 256×256 65 KB	Linear	Quality (Q)	New File Size (KB)	Transmission Time (Sec.)	C.R.	PSNR
		1	9.56	5.54	6.80	24.91
		2	6.35	3.68	10.24	24.65
		3	4.83	2.80	13.46	24.38
		4	3.88	2.25	16.75	24.10
		5	3.26	1.89	19.94	23.83
	Uniform	None	1.92	1.11	33.85	22.80
	Quantization	Vertical Conversion				
	Linear	Quality (Q)	New File Size (KB)	Transmission Time (Sec.)	C.R.	PSNR
		1	9.57	5.55	6.79	24.91
		2	6.30	3.65	10.32	24.65
		3	4.81	2.79	13.51	24.38
		4	3.89	2.26	16.71	24.10
		5	3.28	1.90	19.82	23.83
	Uniform	None	1.87	1.09	34.76	22.80
	Quantization	Zigzag Conversion				
	Linear	Quality (Q)	New File Size (KB)	Transmission Time (Sec.)	C.R.	PSNR
		1	7.68	4.45	8.46	24.91
		2	4.95	2.87	13.13	24.65
		3	3.70	2.15	17.57	24.38
		4	2.93	1.70	22.18	24.10
		5	2.42	1.40	26.86	23.83
	Uniform	None	1.55	0.90	41.94	22.80



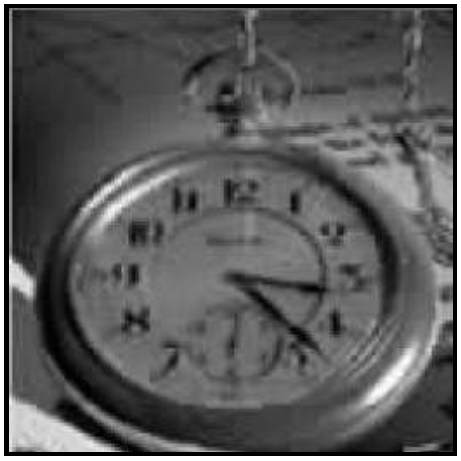
(a)
"Original Image"
256×256



(b)
C.R.: 10.24
T.T.: 3.68
PSNR: 24.65



(c)
C.R.: 13.51
T.T.: 2.79
PSNR: 24.38



(d)
C.R.: 22.18
T.T.: 1.70
PSNR: 24.10

Figure (4-30): Applying Odd-Odd Sampling on Watch Image
a) Original Image b) Linear-Horizontal (Q=2)
c) Linear-Vertical (Q=3) d) Linear-Zigzag (Q=4)

4.11.2 Quantization Process

In this section, the resulting PSNR from linear quantization and uniform quantization with each type of sampling are shown in tables from (4-10) to (4-14).

Table (4-10): Applying Quantization on Lena Image

Quantization	Quality (Q)	Classical Sampling	Even-Even Sampling	Even-Odd Sampling	Odd-Even Sampling	Odd-Odd Sampling
Linear	1	24.71	22.08	22.32	22.03	22.03
	2	24.37	22.18	22.38	22.12	22.12
	3	24.03	22.22	22.37	22.14	22.14
	4	23.78	22.18	22.30	22.09	22.09
	5	23.50	22.12	22.22	22.01	22.01
Uniform	None	21.88	21.76	21.90	21.64	21.64

Table (4-11): Applying Quantization on Boat Image

Quantization	Quality (Q)	Classical Sampling	Even-Even Sampling	Even-Odd Sampling	Odd-Even Sampling	Odd-Odd Sampling
Linear	1	26.4	24.02	24.12	23.99	23.99
	2	25.93	24.05	24.14	24.04	24.04
	3	25.52	24	24.09	23.97	23.97
	4	25.14	23.91	24	23.88	23.88
	5	24.85	23.81	23.88	23.80	23.80
Uniform	None	23.88	23.27	23.28	23.13	23.13

Table (4-12): Applying Quantization on Teeba Image

Quantization	Quality (Q)	Classical Sampling	Even-Even Sampling	Even-Odd Sampling	Odd-Even Sampling	Odd-Odd Sampling
Linear	1	28.27	26.05	26.14	26.06	26.06
	2	27.58	26	26.05	26	26
	3	26.98	25.78	25.81	25.78	25.78
	4	26.48	25.58	25.57	25.53	25.53
	5	26.07	25.29	25.33	25.30	25.30
Uniform	None	20.57	20.58	20.77	20.61	20.61

Table (4-13): Applying Quantization on Nature Image

Quantization	Quality (Q)	Classical Sampling	Even-Even Sampling	Even-Odd Sampling	Odd-Even Sampling	Odd-Odd Sampling
Linear	1	20.85	18.29	18.26	18.29	18.29
	2	20.57	18.48	18.46	18.47	18.47
	3	20.29	18.58	18.58	18.59	18.59
	4	20.02	18.64	18.64	18.64	18.64
	5	19.82	18.67	18.64	18.65	18.65
Uniform	None	19.32	18.41	18.38	18.41	18.41

Table (4-14): Applying Quantization on Watch Image

Quantization	Quality (Q)	Classical Sampling	Even-Even Sampling	Even-Odd Sampling	Odd-Even Sampling	Odd-Odd Sampling
Linear	1	27.30	25.05	25	24.91	24.91
	2	26.60	25.02	24.87	24.65	24.65
	3	25.99	24.84	24.68	24.38	24.38
	4	25.44	24.63	24.40	24.10	24.10
	5	25	24.35	24.18	23.83	23.83
Uniform	None	23.56	23.30	23.19	22.80	22.80

CHAPTER FIVE

CONCLUSIONS & FUTURE WORKS

5.1Conclusions

JPEG standard is so important and has many uses. These uses make JPEG standard as a suitable way for still image compression. So, from our work we conclude the following points:

1. The sampling process approximately reduces the image size (in kilobytes) by the ratio 6:1.
2. Odd-even sampling and odd-odd sampling processes often achieves less PSNR than the others.
3. Quantization process removes the redundancies in the image block and the quality of compressed image depends on quantization type.
4. Uniform quantization reduces the redundancies in the image block much more than linear quantization. Therefore, uniform quantization achieves best compression for the image.
5. Zig-zag conversion is the best way to block the similar values and it achieves a high compression ratio compared with horizontal and vertical conversion.

5.2 Future Works

To develop this work:

1. Adding other color properties to the image such as RGB images.
2. Adding other image sizes such as (512×512) and (1024×1024)
3. Adding other block size such as (16×16) or (32×32).
4. Adding other types of quantization such as non-uniform quantization to achieve a high fidelity.
5. Adding other types of lossless compression algorithms such as arithmetic coding algorithm.

APPENDIX A

Appendix (A)

A1. DC Huffman Table for Luminance Component

Category	Code Length	Code Word
0	2	00
1	3	010
2	3	011
3	3	100
4	3	101
5	3	110
6	4	1110
7	5	11110
8	6	111110
9	7	1111110
10	8	11111110
11	9	111111110

A2. AC Huffman Table for Luminance Component

Run/Size	Code Length	Code Word
0/0(EOB)	4	1010
0/1	2	00
0/2	2	01
0/3	3	100
0/4	4	1011
0/5	5	11010
0/6	7	1111000
0/7	8	11111000
0/8	10	1111110110
0/9	16	1111111110000010
0/A	16	1111111110000011
1/1	4	1100
1/2	5	11011

Run/Size	Code Length	Code Word
1/3	7	1111001
1/4	9	111110110
1/5	11	11111110110
1/6	16	1111111110000100
1/7	16	1111111110000101
1/8	16	1111111110000110
1/9	16	1111111110000111
1/A	16	1111111110001000
2/1	5	11100
2/2	8	11111001
2/3	10	1111110111
2/4	12	111111110100
2/5	16	1111111110001001
2/6	16	1111111110001010
2/7	16	1111111110001011
2/8	16	1111111110001100
2/9	16	1111111110001101
2/A	16	1111111110001110
3/1	6	111010
3/2	9	111110111
3/3	12	111111110101
3/4	16	1111111110001111
3/5	16	1111111110010000
3/6	16	1111111110010001
3/7	16	1111111110010010
3/8	16	1111111110010011
3/9	16	1111111110010100
3/A	16	1111111110010101
4/1	6	111011
4/2	10	1111111000
4/3	16	1111111110010110
4/4	16	1111111110010111
4/5	16	1111111110011000
4/6	16	1111111110011001
4/7	16	1111111110011010
4/8	16	1111111110011011
4/9	16	1111111110011100
4/A	16	1111111110011101

Run/Size	Code Length	Code Word
5/1	7	1111010
5/2	11	11111110111
5/3	16	1111111110011110
5/4	16	1111111110011111
5/5	16	1111111110100000
5/6	16	1111111110100001
5/7	16	1111111110100010
5/8	16	1111111110100011
5/9	16	1111111110100100
5/A	16	1111111110100101
6/1	7	1111011
6/2	12	111111110110
6/3	16	1111111110100110
6/4	16	1111111110100111
6/5	16	1111111110101000
6/6	16	1111111110101001
6/7	16	1111111110101010
6/8	16	1111111110101011
6/9	16	1111111110101100
6/A	16	1111111110101101
7/1	8	11111010
7/2	12	111111110111
7/3	16	1111111110101110
7/4	16	1111111110101111
7/5	16	1111111110110000
7/6	16	1111111110110001
7/7	16	1111111110110010
7/8	16	1111111110110011
7/9	16	1111111110110100
7/A	16	1111111110110101
8/1	9	111111000
8/2	15	111111111000000
8/3	16	1111111110110110
8/4	16	1111111110110111
8/5	16	1111111110111000
8/6	16	1111111110111001
8/7	16	1111111110111010
8/8	16	1111111110111011

Run/Size	Code Length	Code Word
8/9	16	1111111110111100
8/A	16	1111111110111101
9/1	9	111111001
9/2	16	1111111110111110
9/3	16	1111111110111111
9/4	16	1111111111000000
9/5	16	1111111111000001
9/6	16	1111111111000010
9/7	16	1111111111000011
9/8	16	1111111111000100
9/9	16	1111111111000101
9/A	16	1111111111000110
A/1	9	111111010
A/2	16	1111111111000111
A/3	16	1111111111001000
A/4	16	1111111111001001
A/5	16	1111111111001010
A/6	16	1111111111001011
A/7	16	1111111111001100
A/8	16	1111111111001101
A/9	16	1111111111001110
A/A	16	1111111111001111
B/1	10	1111111001
B/2	16	1111111111010000
B/3	16	1111111111010001
B/4	16	1111111111010010
B/5	16	1111111111010011
B/6	16	1111111111010100
B/7	16	1111111111010101
B/8	16	1111111111010110
B/9	16	1111111111010111
B/A	16	1111111111011000
C/1	10	1111111010
C/2	16	1111111111011001
C/3	16	1111111111011010
C/4	16	1111111111011011
C/5	16	1111111111011100
C/6	16	1111111111011101

Run/Size	Code Length	Code Word
C/7	16	1111111111011110
C/8	16	1111111111011111
C/9	16	1111111111100000
C/A	16	1111111111100001
D/1	11	11111111000
D/2	16	1111111111100010
D/3	16	1111111111100011
D/4	16	1111111111100100
D/5	16	1111111111100101
D/6	16	1111111111100110
D/7	16	1111111111100111
D/8	16	1111111111101000
D/9	16	1111111111101001
D/A	16	1111111111101010
E/1	16	1111111111101011
E/2	16	1111111111101100
E/3	16	1111111111101101
E/4	16	1111111111101110
E/5	16	1111111111101111
E/6	16	1111111111110000
E/7	16	1111111111110001
E/8	16	1111111111110010
E/9	16	1111111111110011
E/A	16	1111111111110100
F/0(ZRL)	11	11111111001
F/1	16	1111111111110101
F/2	16	1111111111110110
F/3	16	1111111111110111
F/4	16	1111111111111000
F/5	16	1111111111111001
F/6	16	1111111111111010
F/7	16	1111111111111011
F/8	16	1111111111111100
F/9	16	1111111111111101
F/A	16	1111111111111110

REFERENCES

REFERENCES

1. Arronte, C. S., "Implementation of an eye movement controlled DCT coder", Master thesis, Linköping University, Department of Electrical Engineering, 2000.

2. Bai, X.; Jin, J. S. and Feng, D., "Segmentation-based multilayer diagnosis lossless medical image compression", University of Sydney, School of Information Technologies, 2004.

3. Bethel, D., "Optimization of still image compression techniques", PhD thesis, University of Bath, 1997.

4. Bezroukov N., "OSI protocol layers", Http://www.softpanorama. org/Net/osi_protocol_layers.shtml, 2006.

5. Blelloch, G. E., "Introduction to data compression", Carnegie Mellon University, Computer Science Department, 2001.

6. Cherkassky, V., He, X. and Shao, J., "Image compression for storage transmission of digital images", University of Minnesota, Department of Electrical Engineering & Computer Science, 2000.

7. Davis, L., "OSIstack",Http://www.interfacebus.com/Design_OSI _Stack.html, 2006.

8.Furht, B., "Image presentation and compression", Florida Atlantic University, Department of Computer Science and Engineering, 1995.

9. Gailly, J., "Frequently Asked Questions (part 2/3)", Http://www.faqs. org/faqs/compression-faq/part2/preamble.html/, 2004.

10. Gilbert, J. M., "Text / graphics and image transmission over bandlimited lossy links", PhD thesis, University of California, 2000.

11. Hemami, S. S., "Robust image transmission using resynchronizing variable-length codes and error concealment", Cornell University, School of Electrical Engineering, 1999.

12. Heng, C. L., "Image compression: JPEG", 1997.

13.Herbi, J., "Fractal image compression", PhD thesis, University of Baghdad, College of Science, 2001.

14. Horak, R., "Transmission media: an overview", Http:// www. communicationsconvergence.com/shared/article/showArticle.jhtml, 2002.

15. Iren, S., Amer, P. D. and Conrad, P. T., "Network-conscious compressed image transmission over Battlefield networks", University of Delaware, Department of Computer and Information Sciences, 1998.

16. Jasim, A., "Image compression using wavelet transform", Master thesis, University of Al-Mustansiriya, College of Science, 2005.

17. Krishnappa, R., "Image compression techniques and video streaming for wireless multimedia communication", Illinois Institute of technology, Electrical and Computer Engineering Department, 2003.

18. Kumar, S., "An introduction to image compression", 2001.

19. Lin, T. and Hao, P., "Compound image compression for real-time computer screen image transmission", IEEE Transactions on Image Processing, Vol. 14, No. 8, 2005.

20. Maison,B.,"Adaptiveoperatorsand higher orderstatistics for digital image compression",PhD thesis,Universityof Catholica, 1998.

21. Manoj, A., "Image compression using discrete cosine transform", Mini-thesis, Indiana State University, Department of Electronics and Computer Technology, 2004.

22. Mi, C., "JPEG standard and sequential baseline jpeg", 2003.

23. Pekhteryev, G., Sahinoglu, Z. and Orlik, P., "Image transmission over IEEE 802.15.4 and ZigBee networks", Mitsubishi Electric Research Laboratories, 2005.

24.Ramos,M.G.and Hemami S. S., "Edge-adaptive jpeg image compression", Cornell University, School of Electrical Engineering, 1996.
25. SAMSON AG, "Digital signals", Http://www.samson.de/pdf_en /l150en.pdf, 1999.
26. Shi, Y. Q. and Sun H., "Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms and Standards", 1st Edition, CRC press, USA, 2000.
27. Technical Advisory Service for Images (TASI),"File Formats and Compression", Http://www.tasi.ac.uk/, 2005.
28. Watson, A. B., "Image compression using the discrete cosine transform", NASA Ames Research Center, 1994.
29. Xiao, P. "Image compression by wavelet transform", Master thesis, East Tennessee State University, Department of Computer and Information Sciences, 2001.
30. Zhao, H., "MPEG2 bit rate control for native ATM", Master thesis, San Diego State University, 2001.
31.Http://www.bpbonline.com/Chapters/datacommunication/Bdatach2.pdf , "Data transmission".
32.Http://k-12.pisd.edu/currinst/network/if2_2st.pdf,"Lesson2-2:data transmission".
33. Http://www-ti.informatik.uni-tuebingen.de/os390/communic/inet/ tcparch.pdf, "The TCP/IP protocol suite".
34. Http://www.techbooksforfree.com/intro_to_data_com/page37.html**,** "Introduction to data communications".
35.it.rit.edu/~wmm/.../Lectures/Chapter 6 Transmission McVea v4.pdf, "Data communications & computer networks".

المستخلص

إن معالجة الصور الرقمية أخذت اهتماماً واسعاً في حقل المعرفة في العقود الأخيرة نظراً للتطور السريع في تقنيات الاتصال و الحاجة لإيجاد و تطوير طرق تساعد في تحسين واستغلال معلومات الصورة. إن حقل ضغط الصور الرقمية أصبح حقلاً مهماً من حقول معالجة الصور الرقمية نظراً للحاجة إلى استغلال مساحة الخزن المتوفرة بأكبر قدر ممكن وتقليل الوقت المطلوب لنقل الصورة.

الهدف الأساسي من هذه الرسالة هو تطوير مقياس (***Baseline JPEG***) المستخدم في ضغط الصور ذات عمق لوني مقداره (٨ بتات). بصورة أساسية تتألف هذه الطريقة من سبع عمليات هي الإعتيان(Sampling)، التجزئة(Partitioning)، التحويل(Transform)، التكميم(Quantization)،التغيير(Conversion)، ترميز(Entropy Coding)، ترميز هوفمان(Huffman Coding). أولاً تستخدم عملية أخذ العينة لتقليل حجم الصورة وعدد الـ (bits) المطلوبة لتمثيلها. ثم تطبق عملية التجزئة للحصول على جزء من الصورة بحجم (٨×٨). بعدها يستخدم تحويل الجيب تمام المتقطع لتحويل بيانات جزء الصورة من الحيز المكاني إلى الحيز الترددي لجعل البيانات أسهل عند المعالجة. ثم تطبق عملية التكميم على جزء الصورة المحول لحذف التكرارات الموجودة فيه. بعدها تستخدم عملية التغيير لتغيير جزء الصورة المكمم من مصفوفة ثنائية البعد إلى مصفوفة أحادية البعد(متجه). ثم يطبق ترميز(Entropy) لتحويل المتجه إلى شكل وسطي بحيث يكون من السهل ضغطه. أخيراً يستخدم ترميز هوفمان لتحويل الشكل الوسطي إلى سلسلة من البتات التي يمكن خزنها بسهولة في ملف يمكن خزنه على واسطة خزن أو نقله إلى حاسوب آخر. لغرض استرجاع الصورة المضغوطة يتم عكس العمليات المذكورة أعلاه.

أخيراً، من المهم أن يذكر أن البرنامج المستعمل في هذا العمل قد تم تصميمه باستخدام لغة فيجوال بيسك الإصدار السادس.

بسم اللّٰه الرحمن الرحيم

سبح اسم ربك الأعلى

الذي خلق فسوى

والذي قدر فهدى

والذي اخرج المرعى

فجعله غثاء أحوى

صدق اللّٰه العظيم

سورة الأعلى

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
الجامعة المستنصرية
قسم علوم الحاسبات

ضغط الصورة الرقمية باستخدام طريقة تحويل الجيب تمام المتقطع

رسالة مقدمة إلى
قسم علوم الحاسبات
كلية العلوم / الجامعة المستنصرية
وهي جزء من متطلبات نيل شهادة ماجستير في الحاسبات

من قبل

حامد صادق مهدي سعود السلطاني

المشرف

د. جميلة حربي العامري

تشرين الأول ٢٠٠٦