IMAGE COMPRESSION USING DISCRETE COSINE TRANSFORM

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

OF

CANKAYA UNIVERSITY

119165

BY

AYŞE NURDAN BUZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

OF

MASTER OF SCIENCE

119165

SEPTEMBER 2002

Approval of the Graduate School of Natural and Applied Sciences

[illegible]

Prof. Dr. Üstün Dikeç
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

[illegible]

Prof. Dr. Turhan Alper
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

[illegible]

Prof. Dr. Turhan Alper
Supervisor

Examining Committee Members

Prof. Dr. Turhan Alper	[illegible]
Prof. Dr. Ziya Güvenç	[illegible]
Yrd. Doç. Halil Eyyüboğlu	[illegible]

ABSTRACT

In today's world of computing, it is hardly possible to do without graphics, images and sound. Uncompressed data needs very large amount of physical storage space and transmission time. The availability of storage media and transmission channels are limited. Transmission of images requires high bandwidth or expensive cables. JPEG is the current standard for compression and decompression of still, monochrome and color images. The purpose of this study is to develop a compression algorithm to reduce time in image transmission. The C/C++ language is used for implementation. As in JPEG, Discrete Cosine Transform (DCT) is used as coding transformation. Static Huffman Tree is constituted for our requirements. As a case study, Windows Bitmap (BMP) files are used. The encoded data is formed as binary file and after transmission it is stored as it was before.

ÖZ

Günümüz bilgisayar dünyasında, grafikler, görüntüler ve sesler olmaması mümkün değildir. Sıkıştırılmamış veri büyük miktarda fiziksel depolama alanı ve uzun süreli iletim zamanına ihtiyaç duyar.Bununla beraber depolama ortamları ve ileti-şim kanalları sınırlıdır. Görüntü iletimi yüksek bant genişliği ve pahalı iletişim ortamları (fiberoptik vs.) gerektirir. Durgun, tek renkli yada renkli görüntünün sıkış-tırılmasında JPEG standardı kullanılmaktadır. Bu çalışmanın amacı veri iletiminde zamandan kazanmak için bir sıkıştırma algoritması geliştirmektir.Bu çalışmada C/C++ programlama dili kullanılmıştır. JPEG standardında da olduğu gibi kodla-ma sırasında Ayrık Kosinüs Dönüşümü kullanılmıştır. Bizim ihtiyaçlarımız için Static Huffman Ağacı uygun görülmüştür. Algoritmanın uygulanması sırasında Windows Bitmap (BMP) dosyaları kullanılmıştır. Kodlanmış veri binary (ikili) dosya formatında iletilmiş ve iletim sonunda açılarak eski haline dönüştürülmüştür.

Dedicated to my parents,

Enise and Hasan Buz.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to Prof. Dr. Turhan Alper for his encouragement, guidance and support throughout the study.

My special thanks go to Inst. Reza Hassanpour for his crucial help.

I would also like to express my appreciation to Res. Asst. Sibel Çevik and Res. Asst. Murat Saran, along with all of my friends in the department, for their support, understanding, and friendship through the preparation of this thesis.

Finally, my sincere thanks go to my family for their love, patience and continuous support throughout my whole life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

A few years ago home computers were used primarily for text-based applications; now, with the advent of digital still and movie cameras, we routinely expect to be able to hold large collections of personal multi-media documents. This huge amount of data is difficult to handle so we should use compression to gain physical storage. A second reason for the increase in the use of compression is the strict boundedness of some communication channels. Storing images on computers and transmitting them over the Internet, over local-area networks, and over the phone lines without compression consumes huge amount of physical storage and transmission time. Therefore, compression technology would obtain benefits in decreased storage and transmission requirements.

Not only images but also document storage and transmission systems like fax machines use some kind of image compression technology to compress the data. Thus, no special communication link is required, as an ordinary twisted pair telephone connection suffices. In this sense, pages can be stored in less space and transmitted in less time. This study comprises the compression of images. Nevertheless, our algorithm can be applied with a few modifications to the other types of documents.

The aim of this study is to compress image data and transmit it over the Internet and decompress data in the receiver side. Thus the original image is taken and after encoding, it is transmitted as a binary file to the receiver side. At the receiver side, using the decoding algorithm the output file is formed as it was before.

There are various compression techniques that reduce the amount of data. For increasing the amount of compression ratio, information loss is tolerated in the receiver side in our model. Thus a combination of lossy and lossless compression techniques is used. These techniques that are used in this study are described in the next chapter. Details of our method are given in Chapter 3. Results and discussion are presented in Chapter 4 and conclusions based on this work are in Chapter 5. First appendix is about Windows Bitmap File Format that is used to test the algorithm. The color scheme and representations are described in Appendix B. A brief description of JPEG compression standard is stated in Appendix C.

CHAPTER 2

2.4.1 Image Concepts and Types

A digital image is represented by a two-dimensional array of samples. Each sampled point is called a *pixel* that is given as:

$$x[n_1,n_2]\,, \qquad 0 \le n_1 \le N_1,\ 0 \le n_2 \le N_2$$

where N_1 and N_2 are array sizes, n_1 is the row index, n_2 is the column index of the pixel. The number of intensity levels can be represented is expressed as the number of bits per pixel (bpp).

There are four types of images.These are

1. Binary images (represented by 1 bpp).
2. Computer graphics (generally represented by 4 bpp).
3. Grayscale images (generally represented by 8 bpp).
4. Color images (represented by 16 or 24 bpp).

2.4.2 Image Compression

Compression is a method that reduces the amount of space needed to store the images or amount of time necessary to transmit it. In other words, the mapping from source symbols into fewer target symbols is referred to as *compression*. The transformation from target symbols back into the source symbols is called *decompression*. It may represent the original information or a close form of the original.

2.4.3 Fundamentals of Image Compression

Redundancy: Some sources may deliver data that are represented with more bits per pixel than is strictly necessary. This is called *redundancy*. The aim of this study is to reduce the redundancies and yield a new representation with fewer bits per pixel.

Fidelity: It is preferred that the decompressed image is the exact replica of the original image. The fidelity is the measure of similarities.

2.4.4 Image Compression Techniques

A compression system consists of two distinct fundamental parts: an encoder and a decoder as sketched in Figure2.1. The original image (source image) is fed into the encoder, which creates a set of symbols from the input data. The encoded data can be either stored in any storage device for future use or transmitted over the channel. After transmission over the channel, the encoded data is fed into the decoder to generate a reconstructed output image.

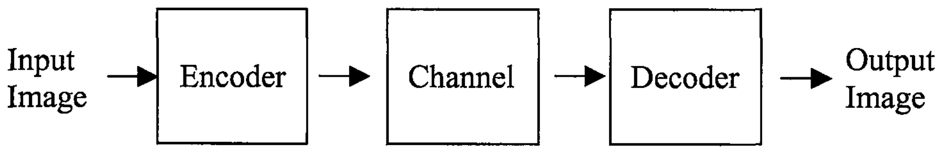


Figure 2.1 A general compression system model

Output image may not be an exact replica of original image. Thus the system may be error free (information preserving) or lossy.

Let's explain each part in Figure 2.1 in detail, starting from the encoder

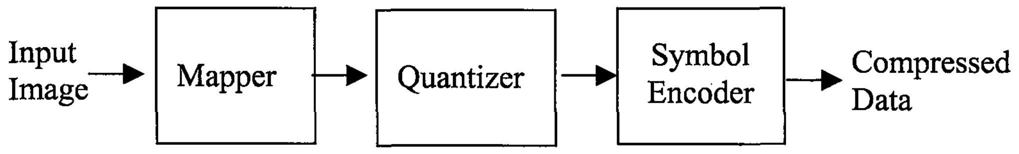


Figure 2.2 Encoder

Encoder is responsible for reducing coding redundancies in the input image. It has three main parts as shown in Figure 2.2.

1. The *mapper* transforms the input data into a format that is designed to reduce the interpixel redundancies in the input image. This operation is reversible and may or may not reduce directly the amount of data required to represent the image. The mapper transforms the image into an array of coefficients [5]. In this study, Discrete Cosine Transform (DCT) Coding technique is used as mapper. It generates array of coefficients, the array size is equal to input data. The generated array of coefficients does not represent any image. Therefore, it is non-visual.

2. The Q*uantizer* quantizes the array of coefficients that is produced by the mapper. It reduces the amount of data required to represent the image. Therefore, it is responsible for introducing distortion. Increasing the values of quantization level may increase the amount of compression. Hence it may distort the replica of the image.

3. *Symbol Encoder* creates a fixed or variable-length code to represent the quantizer output. In this study, Huffman Coding is used as Symbol Encoder. The Huffman Coding assigns the shortest code words to the most frequently occurring output values, thus it reduces coding redundancy.

The decoder performs the inverse operations of source encoder, as seen in Figure 2.3. It has to use the same model that encoder uses. If compressed data may not affected by transmission or storing, the difference between input image and output image will only arise from the coding model.

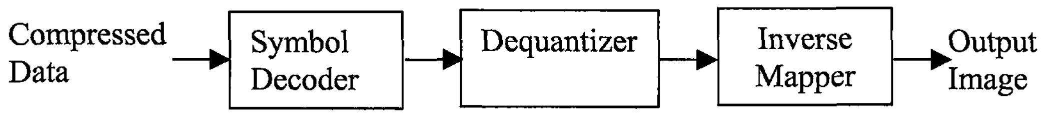


Figure 2.3 Decoder

Digital image compression can be classified into two categories: *lossless* and *lossy compression*. Lossless compression is a method where in the identical source image data can be reconstructed from the compressed data of the original image. Lossy compression is a method where in the decompressed image is not the exact replica of to the original image but close to it. If the information loss can be tolerated at the receiver site, lossy compression may be employed. It is well known that the eye can tolerate certain image imperfections. The lossy techniques provide higher compression ratios. Therefore, lossy techniques are more often applied to image and video compression than lossless techniques. Let's explain lossless and lossy compression in details.

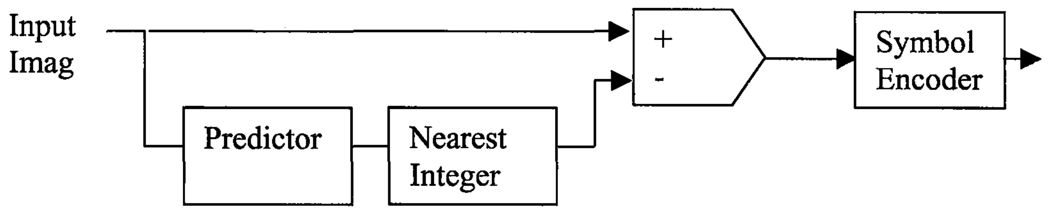
2.4.1 Lossless Compression

In lossless compression there are various models. Some of these are:

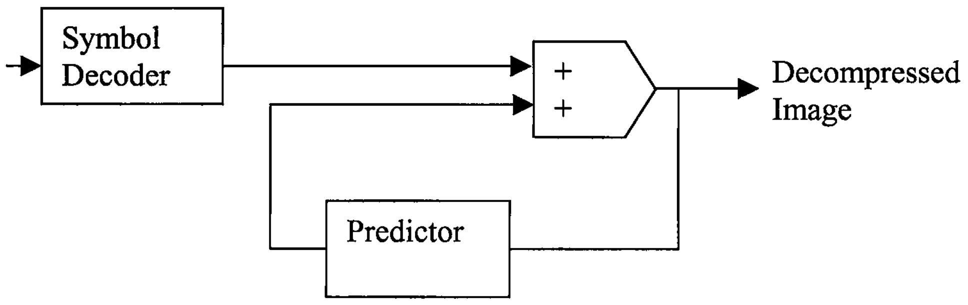
Lossless Predictive Coding (Differential Pulse Code Modulation Model)

It is time domain coding method that is based on eliminating pixels with a prediction rule; a pixel value can be predicted from the previous pixel values. If the new information of a pixel is defined as the difference between the actual and predicted value of that pixel, the prediction method is called **Differential Pulse Code Modulation (DPCM)**.

If it is assumed that the data is coded from left to right, the actual value may be predicted from the value of the left sample as shown in Figure 2.4.



a. DPCM Encoder



b. DPCM Decoder

Figure 2.4 A lossless predictive coding model.

Predictive coding plays an important role in image compression standards such as JPEG, JBIG and MPEG due to its compression efficiency and its simplicity [6].

Run-Length Coding

This is another type of time domain coding method. If there are sequences of repeated pixel this coding is proved to be useful. For example if an image has extensive background, it will have an extensive correlation among neighboring pixels. There will be sequences of repeated pixels. Run length coding uses a count number to replace a sequence of repeated pixels.

This technique is demonstrated in Figure 2.5:

Digital Image	**Run-length coding results**
$$$$$$$$$$$$$$$$$$$$	20$
$$$$$$$$$**$$$$$$$$$	9$2*9$
$$$$$$$**^^**$$$$$$$	7$2*2^2*7$
$$$$$**^^^^^^**$$$$$	5$2*6^2*5$
$$$$$**^^^^^^**$$$$$	5$2*6^2*5$
$$$$$**^^^^^^**$$$$$	5$2*6^2*5$
$$$$$**^^^^^^**$$$$$	5$2*6^2*5$
$$$$$$$**^^**$$$$$$$	7$2*2^2*7$
$$$$$$$$$**$$$$$$$$$	9$2*9$
$$$$$$$$$$$$$$$$$$$$	20$

Figure 2.5 Example of run-length coding for a digital image represented by the source symbols {$, *, ^}.

Huffman Coding

Instead of using fixed number of bits per pixel, statistically most common symbols are encoded using fewer bits than less frequent symbols. D. A. Huffman [7] developed a coding technique that produces the shortest possible average code length given the source symbol set and associated probabilities.

A Huffman code may be obtained using binary tree with branches assigned the values 0 or 1. The top of the tree is called *root node*, and other points are called *branch node* and *leaf nodes*.

The Huffman algorithm

The Huffman coding may be formed using following steps [6-22]:

Step 1: List the probabilities of the symbols according to frequency in ascending order. Each character is now a leaf node of a tree.

Step 2: Take two nodes with the two smallest frequency weightings and generate a new node (branch nodes), which is the sum of these two frequency weightings. Remember the weights of the two nodes chosen must be smaller than the combination of any other possible choices.

Step 3: Mark the branch of left leaf node as 0 and the branch of its right leaf node as 1.

Step 4: Update the node set by replacing the two nodes with the two smallest probabilities for the newly produced node. If the node set contains only one node, quit. Otherwise go to Step 2.

Let's take any text as an example to prepare a Huffman Code. In this hypothetical example if the frequency of occurrence of each character is assumed to be:

'A' and 'B' =0.25, 'C' and 'D' = 0.14, 'E','F', 'G', 'H' = 0.055

By using steps described above, let's prepare Huffman Tree:

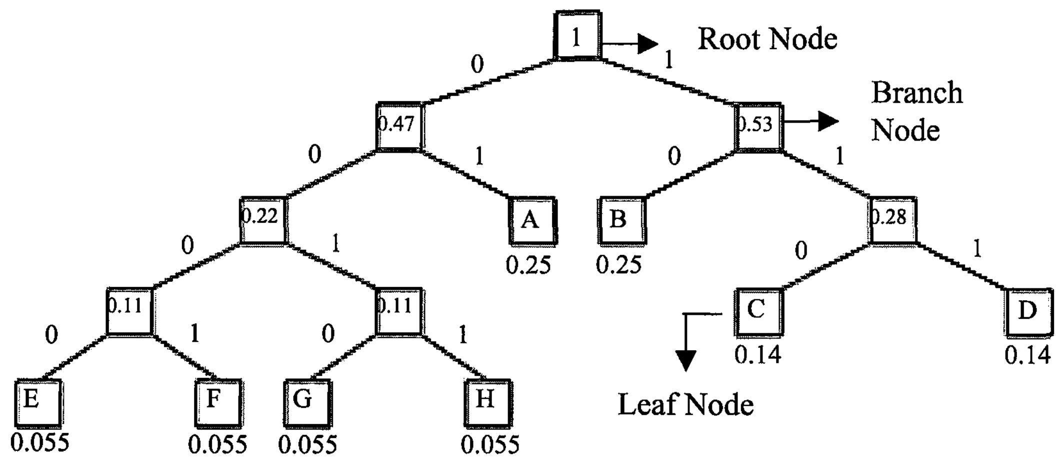


Figure 2.6 Huffman code assignment procedure

From the tree shown in Figure 2.6, the most commonly used letters in the text A and B require only 2 bits, less commonly used values require 3 or 4 bits. A may be represented by 01. The binary representation of the other letters are :

A = 0 1

B = 1 0

C = 1 1 0

D = 1 1 1

E = 0 0 0 0

F = 0 0 0 1

G = 0 0 1 0

H = 0 0 1 1

To illustrate, if the string ‘AAACCF' is taken, the Huffman code of the string will be 0101011101100001 (20 bits).

If each character requires 8-bits, the string above would require at least 6 * 8 = 48 bits. About 50% compression is achieved.

Once the code has been created, coding and decoding is accomplished in a simple look-up table.

2.4.1 Lossy Compression

Predictive Coding

In this technique, a quantizer is added to the model introduced in Lossy Predictive Model. Prediction error appears that establishes the amount of compression and distortion. In this study, this type of coding is not mentioned.

Other Models

There are some other models such as block truncating, vector quantization, sub-band, fractial coding etc. After January 1988 ISO (International Standard Organization) supported DCT model and applications of other models had been reduced.

2.4.2 Transform Coding

In this part, compression techniques that are based on the transform of an image will be introduced. Transforms, generally integral transforms, are used primarily for the reduction of complexity in mathematical problems. Differentials and integrals may be replaced into algebraic equations whose solutions are more easily obtained, by applying appropriate transforms.

A reversible, linear transforms (such as Fourier, Cosine, Wavelet transform) are used to map the image into a set of transform coefficients. Transform coding denotes a procedure, in which the image is subjected, prior to coding and transmission, to an invertible transform, with the aim of converting the statistically dependent image elements to independent coefficients.

Two-dimensional linear transformations are defined as

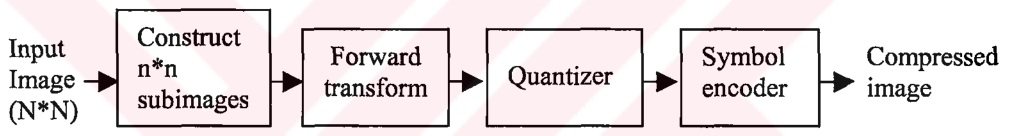
$$P(m_1,m_2) = \sum_{n_1=1}^{N_1} \sum_{n_2=1}^{N_2} F(n_1,n_2)T(n_1,n_2;m_1,m_2)$$

where P is the transformed point, F is the pixel of the original image, T is the transform coefficient.

An efficient transformation produces fewer correlated transform coefficients than the original. The transformation process itself does not reduce the data; it prepares data such that the quantization process can efficiently carry out compression.

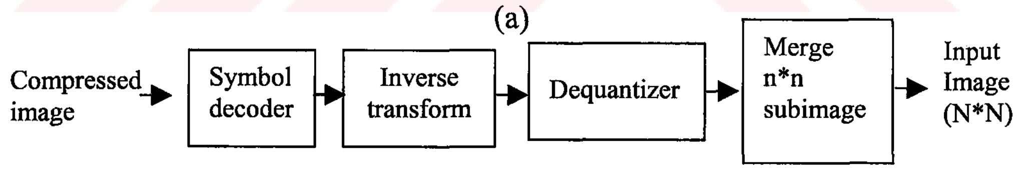
In transformation, images are divided in subimages so that redundancy adjacent subimages are reduced to some acceptable levels. n, subimage dimension, is power of 2. The most popular subimage sizes are 8 by 8 and 16 by 16.

Subimages are transformed and encoded as explained in Section 2.4. The encoding and decoding operations are shown in schematic form below. The Forward Transform takes place of the mapper in Figure 2.2, and also inverse transform takes the place of inverse mapper.



(a) Encoder of Linear Transform

(a)



(b) Decoder of Linear Transform

Figure 2.7: A transform coding system: (a) encoder, (b) decoder

Transform selection depends on the amount of reconstruction error that can be tolerated. There are various transformation techniques:

2.5.1 Fourier Transform

The discrete two-dimensional Fourier transform of an image is defined as

$$F(u,v)=\frac{1}{N}\sum_{j=0}^{N-1}\sum_{k=0}^{N-1}f(j,k)\exp\left\{\frac{-2\pi i}{N}(uj+vk)\right\}$$

where $f(j,k)$ is the input array, $i=\sqrt{-1}$, u, $v, j, k = 0, 1, N$-1,

the indices (u, v) are called *spatial frequencies of the transformations* and the discrete inverse transform is given as:

$$f(j,k)=\frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}F(u,v)\exp\left\{\frac{2\pi i}{N}(uj+vk)\right\}$$

The explanation of the parameters is given above.

2.5.2 Discrete Cosine Transform (DCT)

The Discrete Cosine Transform has been selected throughout this thesis. Although, reasons of this choice will be mentioned later, briefly it can be said that DCT is fast and efficient in real arithmetic.

Derivation of DCT from Fourier Transform

The real part of the Fourier series of any waveform is cosine terms of the series [8].

If *f (j, k)* is real and symmetric about the point j = -1/2, k = -1/2

$$F\ (u,v)=\frac{2}{N}\sum_{j=0}^{N-1}\sum_{k=0}^{N-1}f(j,k)\cos\left[\frac{\pi}{N}u\left(j+\frac{1}{2}\right)\right]\cos\left[\frac{\pi}{N}v\left(k+\frac{1}{2}\right)\right]$$

The 2-D forward cosine transform (FDCT) is defined to be normalized version of the equation above.

$$F(u,v)=C(u)C(v)\sum_{j=0}^{N-1}\sum_{k=0}^{N-1}f(j,k)\cos\left(\frac{(2j+1)u\pi}{2N}\right)\cos\left(\frac{(2k+1)v\pi}{2N}\right)$$

and 2-D inverse cosine transform (IDCT) is defined as

$$f(j,k)=\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}C(u)C(v)F(u,v)\cos\left(\frac{(2j+1)u\pi}{2N}\right)\cos\left(\frac{(2k+1)v\pi}{2N}\right)$$

where $C(u)=\begin{cases}\dfrac{1}{\sqrt{N}} & ,u=0\\ \sqrt{\dfrac{2}{N}} & ,u\neq 0\end{cases}$

1-D DCT is defined as

$$F(u)=C(u)\sum_{j=0}^{N-1}f(j)\cos\left(\frac{(2j+1)u\pi}{2N}\right)$$

$$f(j)=\sum_{u=0}^{n-1}C(u)F(u)\cos\left(\frac{(2x+1)u\pi}{2N}\right)$$

The implementation of the direct 2-D DCT requires much more effort than that of the separable 2-D DCT.

Row-Column Method:

Both 2-D Forward Discrete Cosine Transform (FDCT) and Inverse Discrete Cosine Transform (IDCT) are separable transformations, which means that they can be obtained by first performing 1-D FDCT/IDCT on the rows, then performing 1-D FDCT/IDCT on the columns [19]. This method is called row-column method or indirect method.

The general block diagram is shown in Figure 2.8

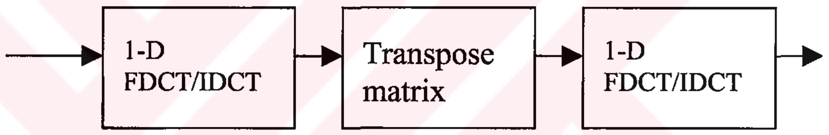


Figure 2.8 2-D FDCT/IDCT using row-column method

Various fast algorithms have been reported for DCT in the literature [21]. The aim of these algorithms is to reduce the number of additions and multiplications. These algorithms are usually taking advantage of the symmetry in the cosine basis functions, and the computation complexity is fixed for all input data. Since multiplication requires more hardware and computation time than adding, fewer multiplication imply more power.

A Fast Cosine Transform Algorithm (Chen's Algorithm)

Chen's fast algorithm is the most widely used DCT/IDCT algorithm. It is based on the symmetry in the DCT / IDCT matrix [10-18].

If these 1D-DCT is represented as matrix, it will be a matrix as X = A **.** x where is the transformed data, A is the matrix of transformation coefficients and x is the matrix of source data values. It can be written as following:

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ b & d & e & g & -g & -e & -d & -b \\ c & f & -f & -c & -c & -f & f & c \\ d & -g & -b & -e & e & b & g & -d \\ a & -a & -a & a & a & -a & -a & a \\ e & -b & g & d & -d & -g & b & -e \\ f & -c & c & -f & -f & c & -c & f \\ g & -e & d & -b & b & -d & e & -g \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \\ x(5) \\ x(6) \\ x(7) \end{bmatrix} \qquad \textbf{(Eq.2.1)}$$

where the coefficients a, b, c, d, e, f, g are given as:

[a b c d e f g] = 1/2[cos (π/4) cos (π/16) cos (π/8) cos (3π/16) cos (5π/16) cos (3π/8) cos (7π/16)]

Since the even rows of the matrix are even-symmetric and odd rows are odd-symmetric, (eq.2.1) can be written as:

$$\begin{bmatrix} X(0) \\ X(4) \\ X(2) \\ X(6) \\ X(1) \\ X(5) \\ X(3) \\ X(7) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ a & -a & a & -a & a & -a & a & -a \\ c & -f & -c & f & c & -f & -c & f \\ f & c & -f & -c & f & c & -f & -c \\ b & e & -g & d & -b & -e & g & -d \\ e & g & -d & b & -e & -g & d & -b \\ d & -b & e & g & -d & b & -e & -g \\ g & d & b & e & -g & -d & -b & -e \end{bmatrix} \begin{bmatrix} x(0) \\ x(2) \\ x(4) \\ x(6) \\ x(7) \\ x(5) \\ x(3) \\ x(1) \end{bmatrix}$$

By dividing this matrix into four matrix as shown below,

$$\begin{bmatrix} X(0) \\ X(4) \\ X(2) \\ X(6) \\ X(1) \\ X(5) \\ X(3) \\ X(7) \end{bmatrix} = \left[\begin{array}{cccc|cccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ a & -a & a & -a & a & -a & a & -a \\ c & -f & -c & f & c & -f & -c & f \\ f & c & -f & -c & f & c & -f & -c \\ \hline b & e & -g & d & -b & -e & g & -d \\ e & g & -d & b & -e & -g & d & -b \\ d & -b & e & g & -d & b & -e & -g \\ g & d & b & e & -g & -d & -b & -e \end{array}\right] \begin{bmatrix} x(0) \\ x(2) \\ x(4) \\ x(6) \\ x(7) \\ x(5) \\ x(3) \\ x(1) \end{bmatrix} \qquad \textbf{(Eq.2.2)}$$

It can be easily represented as $\begin{bmatrix} A & A \\ B & -B \end{bmatrix}$ matrix thus (eq.2.2) can be rewritten as following:

$$\begin{bmatrix} X(0) \\ X(2) \\ X(4) \\ X(6) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ c & f & -f & -c \\ a & -a & -a & a \\ f & -c & c & -f \end{bmatrix} \begin{bmatrix} x(0)+x(7) \\ x(1)+x(6) \\ x(2)+x(5) \\ x(3)+x(4) \end{bmatrix}$$

$$\begin{bmatrix} X(1) \\ X(3) \\ X(5) \\ X(7) \end{bmatrix} = \begin{bmatrix} b & d & e & g1 \\ d & -g & -b & -e \\ e & -b & g & d \\ g & -e & d & -b \end{bmatrix} \begin{bmatrix} x(0)-x(7) \\ x(1)-x(6) \\ x(2)-x(5) \\ x(3)-x(4) \end{bmatrix}$$

Similarly the 1-D IDCT can be rewritten as follows:

$$\begin{bmatrix} Y(0) \\ Y(1) \\ Y(2) \\ Y(3) \end{bmatrix} = \begin{bmatrix} a & c & a & f \\ a & f & -a & c \\ a & -f & -a & c \\ a & -c & a & -f \end{bmatrix} \begin{bmatrix} X(0) \\ X(2) \\ X(4) \\ X(6) \end{bmatrix} + \begin{bmatrix} b & d & e & g \\ d & -g & -b & -e \\ e & -b & g & d \\ g & -e & d & -b \end{bmatrix} \begin{bmatrix} X(1) \\ X(3) \\ X(5) \\ X(7) \end{bmatrix}$$

$$\begin{bmatrix} Y(7) \\ Y(6) \\ Y(5) \\ Y(4) \end{bmatrix} = \begin{bmatrix} a & c & a & f \\ a & f & -a & c \\ a & -f & -a & c \\ a & -c & a & -f \end{bmatrix} \begin{bmatrix} X(0) \\ X(2) \\ X(4) \\ X(6) \end{bmatrix} - \begin{bmatrix} b & d & e & g \\ d & -g & -b & -e \\ e & -b & g & d \\ g & -e & d & -b \end{bmatrix} \begin{bmatrix} X(1) \\ X(3) \\ X(5) \\ X(7) \end{bmatrix}$$

CHAPTER 3

Methods and Discussion

In this chapter, the algorithm that is proposed will be discussed. Since our aim is to compress image data and transmit it over a network channel, compression techniques are examined. There are some other compression techniques that we did not mention in this study. We preferred lossy compression techniques to increase the compression ratio. Of course it will cause some distortion, which means that output image will not be the exact replica of original image. This information loss is tolerated because the aim of this algorithm is to transmit data over Internet thus to increase compression ratio, we tolerated the imperfections that eye can tolerate some imperfections. As stated earlier, Discrete Cosine Transform is used as a lossy compression model. Some of the reasons can be listed as following:

- Since DCT is input independent, DCT provides a good compromise between information packing ability and computational complexity. In fact, the properties of DCT have proved to be of such practical that it has become an international standard as the Joint Photographic Experts Group (JPEG) image compression method.

- The DCT has become the foundation of wide applications in image/video processing. It has become the heart of many other international standards such as H.26x, and the MPEG family [2]-[9].

- When compared to DFT (Discrete Fourier Transform), DCT reduces some of the problems, which arise in the application of DFT to a data series. Naturally, DFT is applied to sampled data, and so the transform domain has a "repeat" spectrum. Sampling rate should be such that aliasing does not occur. Transform coefficients are also sampled. Therefore the DFT representation is not that of isolated segment of the input, but is of that sample periodically repeated. Such a waveform contains severe discontinuities due to the level difference between start and end of the repeated segment as shown in Figure 3.1. However, in the case of DCT the segment is made even symmetric before transforming. The start and end of the new, even symmetric segment is at the same level [5]. This fact is, also, the reason why even sized DCT is preferred.

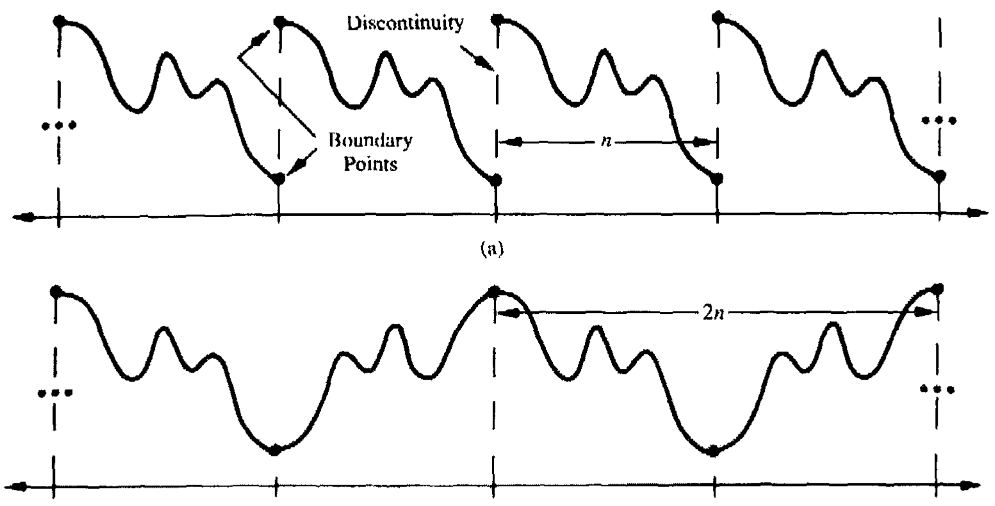


Figure 3.1 The periodicity implicit in the (a) DFT and (b) DCT

Applying DCT we have an array of coefficients that has many 0 values. After transform coding, we applied run-length coding. This method is lossless and it is appropriate for the images that have sequences of repeated pixel. Thus run-length is suitable in this stage of our algorithm.

The third stage of our algorithm is Huffman Coding; it is another type of lossless compression. As mentioned in Chapter 2, an appropriate Huffman tree must be formed. We examined some different pictures and concluded that after the steps mentioned above we have many 0s and small numbers, and few big numbers. Thus we prepared a table, which has fewer bits for 0 and other numbers, more bits for big numbers.

The steps of our algorithm can be given as:

Encoder

- Examining the Properties of Source File
- Transform Coding
- Quantization
- Zigzag Scan, Run Length Coding
- Huffman Coding

Decoder

- Reversing all the steps above

Examining the Properties of Source File

In this study, Windows Bitmap (BMP) files are used as an input file and output file to test the compression algorithm. For more details about the structure of BMP files, refer to Appendix A. In BMP, the first 58 byte of the source file is a header (it includes BMP file header and information header). This header preserves the information about the images such as width, height, resolutions, number of colors, etc.

BMP uses either color indexes or RGB if it is not monochrome. If BMP is color indexed, it means that it uses a color table. Each entry in his color table includes red, green and blue intensities for a color. Thus, a 4 bytes palette entry is needed for each color (for more information about color indexes refer to Appendix B). Therefore, table size will be 256*4 = 1024 byte after information header.

If the BMP is color indexed (in another word, 8 bpp BMP), the header and color table, which are 1024 + 58 =1082 bytes, are written to the output without any process. If it is in RGB format (24 bpp), only the first 58 byte of the input image are written to the output file. If you apply compression to the header file, it will cause an important damage.

Transform Coding

As an example to show the loss of data in DCT, the numbers between 1, 64 is taken as the elements of an 8 by 8 matrix, shown in Table (3.1.a). DCT of these numbers are calculated as Table (3.1.b), IDCT of calculated values are shown in Table (3.1.c).

1.000	2.000	3.000	4.000	5.000	6.000	7.000	8.000
9.000	10.000	11.000	12.000	13.000	14.000	15.000	16.000
17.000	18.000	19.000	20.000	21.000	22.000	23.000	24.000
25.000	26.000	27.000	28.000	29.000	30.000	31.000	32.000
33.000	34.000	35.000	36.0000	37.000	38.000	39.000	40.000
41.000	42.000	43.000	44.000	45.000	46.000	47.000	48.000
49.000	50.000	51.000	52.000	53.000	54.000	55.000	56.000
57.000	58.000	59.000	60.000	61.000	62.000	63.000	64.000

(a) Original image data

260.000	-145.775	0.000	-15.237	0.000	-4.544	0.000	-1.141
-18.221	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
-1.904	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
-0.568	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
-0.142	0.000	0.000	0.000	0.000	0.000	0.000	0.000

(b) After 2D Cosine Transform

1.000	1.999	3.000	3.999	5.000	5.999	7.000	8.000
8.996	9.996	10.996	11.996	12.996	13.996	14.997	15.996
17.002	18.002	19.003	20.002	21.003	22.002	23.003	24.002
24.997	25.996	26.997	27.996	28.997	29.996	30.997	31.997
33.002	34.002	35.003	36.002	37.003	38.002	39.003	40.002
40.997	41.996	42.997	43.996	44.997	45.996	46.997	47.997
49.003	50.002	51.003	52.003	53.003	54.003	55.003	56.003
56.999	57.999	59.000	59.999	61.000	61.999	63.000	63.999

(c) After 2-D Inverse Discrete Cosine Transform

Table 3.1. Example of DCT / IDCT

Table 3.1 shows that after cosine transform and inverse transform, there is not much difference between input and output data. It can be said that if data are rounded, there is no difference between input and output data.

Notice that 64 coefficients transformed by DCT, shown in Table (3.1.b) have many 0 values.

Quantization

After 64 coefficients came into existence, the quantization table (Table 3.2.a) is represented as:

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

(a) Quantization Table as in JPEG

16.25	-12.14794	0	-1.08841	0	-0.189362	0	-0.015848
-1.656537	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
-0.11904	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
-0.014202	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
-0.00234	0	0	0	0	0	0	0

(b) transformed_data / quantization_value

16	-12	0	-1	0	0	0	0
-2	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(c) Rounded (transformed_data / quantization_value)

Table 3.2 Quantized DCT Coefficients

It is calculated with the equation that was given before

$$F_q(u,v) = round\left[\frac{F(u,v)}{Q(u,v)}\right]$$

where F(u,v) is the DCT coefficient, Q(u,v) is quantization coefficient. It is responsible for the loss of compression.

Zigzag Scan, Run Length Coding

The Zigzag scan table, which JPEG uses, is used to sort the data.

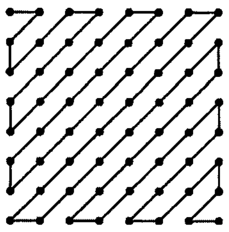


Table 3.3 ZigZag Scan Table as given in JPEG standard

After zigzag scanning, data will be 16 -12 -2 0 0 0 0 0 0 -1 0 ... 0

All data after –1 are 0s. Run length of zeros can be calculated by subtracting the number of data, other than 0, from 64. So the zigzag scan data will be [16 -12 -2 0 0 0 0 0 0 –1 EOB]

Huffman Coding

The default JPEG Huffman Codes for luminance can be formed from pages 396-398 on [5].

Coefficient Categories and Coefficient Coding Tables used to decrease number of bits in this study are as following:

Range	Category
0	0
-1 1	1
-3 , -2 2 , 3	2
-7 , . . . , -4 4 , . . . , 7	3
-15 , . . . , -8 8 , . . . , 15	4
-31 , . . . , -16 16 , . . . , 31	5
-63 , . . . , -32 32 , . . . , 63	6
-127 , . . . , -64 64 , . . . , 127	7
-255 , . . . , -128 128 , . . . , 255	8

Table 3. 4 Coefficient Coding Categories

Range in Table 3.4 is data range. Since our data type is signed character, our data are between 0-128. We assign a Category to each data. For example if we have the data 14, Category will be 4.

Category	Base Code	Length	Category	Base Code	Length
0	010	3	4	101	7
1	011	4	5	110	8
2	100	5	6	1110	10
3	00	5	7	11110	12

Table 3.5 Appointing Base Codes and Length for Categories

If the Category is 4, from the Table 3.5, our base code will be 101 and length will be 7. 14 can be expressed as 00001110 in binary. Thus we may express 14 as 1011110 (7 bits).

The most commonly used value requires only 3 or 4 bits, less commonly used values requires 8 or 10 bits.

As an example of sequence,

[16, -12, -2, 0, 0, 0, 0, 0, 0, –1, EOB] (11 byte) will be

[1011000, 1010011, 10001, 010, 010, 010, 010, 010, 010, 0110, 111110] (47 bit)

code word of EOB is defined as 111110.

Decoder

Reversing all the steps above

Results after dequantization is shown below, the distortion can be seen by comparing Table 3.6 with Table 3.1.b.

256	-144	0	-14	0	0	0	0
-22	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Table 3.6 Dequantization

Results after dequantization is shown below, the distortion can be seen by comparing Table 3.7 with Table 3.1.a

1.161	1.741	2.815	4.216	5.734	7.136	8.209	8.790
7.500	8.080	9.154	10.555	12.073	13.475	14.548	15.129
16.472	1.705	18.126	19.528	21.046	22.447	23.521	24.101
24.592	2.517	26.246	27.647	29.166	30.567	31.640	32.227
31.778	3.235	33.432	34.833	36.352	37.753	38.827	39.407
39.898	4.047	41.552	42.953	44.471	45.873	46.946	47.527
48.870	4.945	50.524	51.926	53.444	54.845	55.919	56.499
55.209	5.579	56.863	58.265	59.783	61.184	62.258	62.838

Table 3.7 Inverse DCT after Dequantization

CHAPTER 4

Results And Conclusion

For testing the algorithm, an aquarium image, as shown in Figure 4.1, is used. It is a 480 by 224, 24-bit BMP image. Original size of this image is 322560 bytes. Compressed size is 89634 bytes. Decompressed image is shown in Figure 4.2.

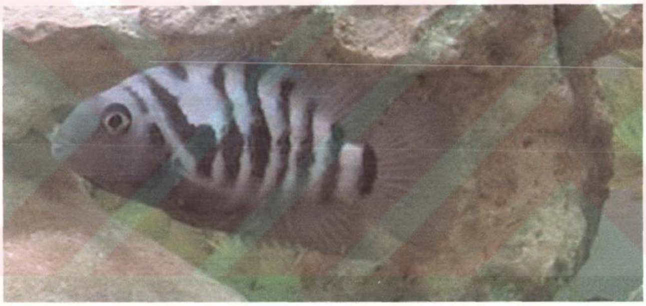


Figure 4.1 Aquarium image (24-bit BMP)

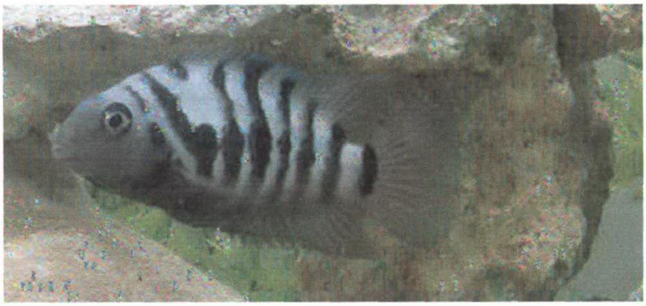


Figure 4.2 Decompressed Aquarium image

Compression ratio is changing from image to image. It is dependent on image. To illustrate it, another image that is not much different from first one is examined. If the image as shown in Figure 4.3 is given;

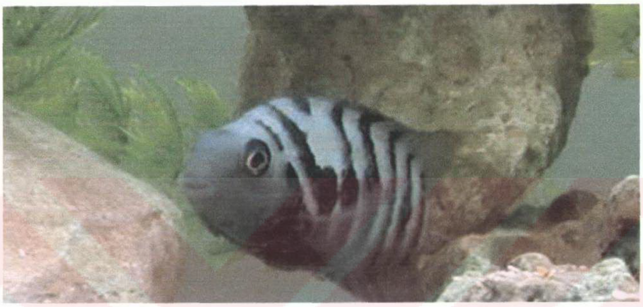


Figure 4.3 Aquarium image 2

Original size of Figure 4.3 is equal to the image in Figure 4.1 (322560 bytes). Compressed size is 91450 bytes.

Thus we can calculate the compression ratio as following but we can't say that it is a fixed ratio. It differs from image to image.

Compression Performance:

Compression Ratio (C*r*) for Figure 4.1 may be calculated as

$$\frac{(322560 - 89634)}{322560} \cong 0.72 = 72\%$$

If we use 8 bpp BMP, it means we use color indexed BMP. The compression results are shown in Figure 4.4.

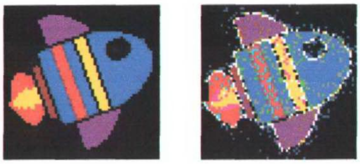


Figure 4.4 Example for loss of DCT/IDCT for 8bpp BMP

Since 8 bpp BMP uses color table, index number 11 is much different from index number 12. So the distortion of the image is increasing. Hence using RGB format (24 bpp) image is much better.

As a conclusion, our algorithm is running under BMPs but true color BMPs give better results.

Future Work

In the past few years, much effort has been spent to provide visual communication over the existing telephony network. The telephone lines were designed for transmitting vocal data only, so the bandwidth is bounded. This means that compression is an important part of communication. The speed and the efficiency of the compression algorithms are considerable. Till now, an algorithm for compression of still color images is worked out. The next step of this study may be extended as video coding. The main idea of video coding as in H.261 (video coding standard published by the ITU (International Telecom Union) in 1990) is based on inter-frames and intra-frames. The term ***intra frame coding*** refers to the fact that the various compression techniques are performed relative to information that is contained only within the current frame, and not relative to any other frame in the video sequence. In this stage of coding our algorithm may be applied. And then ***inter frame coding*** which refers to pseudo-differences from previous frame (predicted) can be implemented. In video coding, the speed of the algorithm is more important than still image coding. Thus a new implementation may be done by using DSPs (Digital Signal Processors). Multiplication is carried out faster whith yhese boards than the computers. By adding a DSP in this project, a faster algorithm may be worked out. Another implementation for making algorithm faster may be parallel computing. Either computers or DSPs can be used for parallel computing. It may cause a faster and efficient algorithm.

APPENDIX A

BITMAP FILE FORMAT

BITMAP file format stores raster image data that is independent of the color specification scheme used on any single hardware device. BMP is a native bitmap format of MS Windows and it is used to store (virtually) any type of bitmap data. Most applications running under MS Windows (MS DOS) and under other operating systems support read and write to BMP files.

Major Type of Data: 2D raster

Color Representation: Monochrome, Color lookup table, RGB

Data Organization: Sequential; 2D array of pixel values

Data Encoding: Binary

Data Compression: None, run-length

Resolution: Pixels per meter

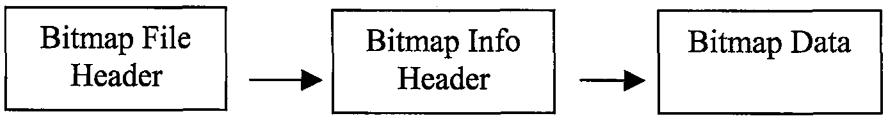


Figure A.1 Data Organization

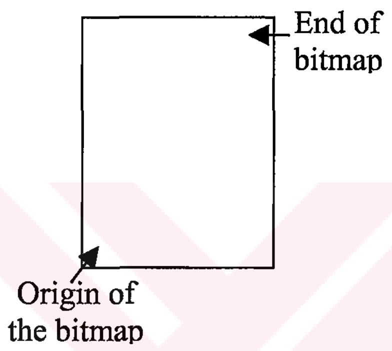
Byte #	Data	Details
1-2	Must be ASCII text 'BM'	File type should be BM
3-6	Physical file size	In double words (32 bit integers)
7-10	Reserved for future use	Must be zero
11-14	Offset of bitmap data	Start of image data offset in bytes

Table A.1 Bitmap File Header

Byte #	Data	Details
1-4	Info Header Size	Currently 40 bytes
5-8	Width of Bitmap	In pixels
9-12	Height of Bitmap	In pixels
13-14	Number of color planes	Must be set to 1
15-16	Number of bits per pixel	Valid choices are 1,4,8,24
17-20	Type of Compression	0:No compression, 1:run length (8 bpp) 2:run length (4 bpp)
21-24	Size of image	Bytes
25-28	Horizontal resolution	Pixels/meter
29-32	Vertical resolution	Pixels/meter
32-36	Number of color indexes	Zero indicates all colors are important
37-40	Number of important colors for displaying bitmap	Zero indicates all colors are important
41	Blue color value	Beginning color palette
42	Green color value	
43	Red color value	
44	Reversed for future use	Must be zero
...	...	Remaining color palette entries
...	...	...
...	...	...

Table A.2 Bitmap Information Header

4 bytes per palette entry, the number of entries is based on the per pixel value above. Colors should be listed in order of importance. (It is usually best to sort to colors in the color table and place the colors occurring with the greatest frequency in the image first. This value can then be stored in Number of important colors field in the bitmap info header.)



Bitmap Data: The pixels are stored left to right within each row. The rows are stored bottom to top. So image data is always displayed starting at the lower-left corner of the screen.

A.1 Bitmap File Types

There are four BMP formats:

1-bit BMP data $\Longrightarrow$ each bit represents a pixel,
it is monochrome,
the most significant bit in byte is the first pixel value.

4-bit BMP data $\Longrightarrow$ each 4 bits in the bitmap array represents a pixel,
has a maximum of 16 colors

8-bit BMP data $\Longrightarrow$ each byte represents one pixel,
has a maximum of 256 colors

24-bit BMP data $\Longrightarrow$ three bytes represent one pixel, has a maximum of 16777216 colors

stored in blue, green, red order. (each byte represents the relative intensities of RGB.)

Because 1and 8-bit images use palettes, the pixel values read from the BMP data are index values into the palette that hold the actual pixel color. 24-bit images do not use a palette; their pixel color data is stored directly in the image data.

A.2 Bitmap Compression

Bitmap data could be compressed using one of the three types of compression. Type of compression used by the bitmap is specified in Bitmap Info Header structure in the type of Compression member. It can be set to one of the following values:

0. BI - RGB

   This means that the bitmap is actually not compressed. 1 bpp and 24 bpp bitmaps are always using this type of compression.

1. BI - RLE 8

   The bitmap data is compressed using run-length encoded format for a 256 -color bitmap.

2. BI - RLE 4

   The bitmap data is compressed using run-length encoded format for a 16 - color bitmap.

APPENDIX B

COLOR

B.1 Color Scheme

In this scheme a color value represents an index into a table, not an actual intensity. This table is named lookup table, color map, palette etc.

- Grayscale images use a single lookup table since they have only one intensity value per pixel.

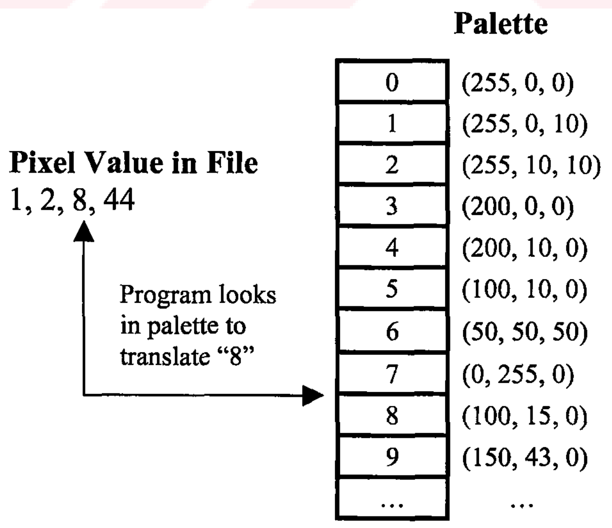


Figure B.1 An Example of Color Palette

A reason for using color palette is to lower the memory requirements for a raster image. For instance, the memory requirements for a raster image is

$$\frac{\text{width*height*(bits/intensity)*(intensity/pixel)}}{\text{8 bits per byte}}$$

Storage Need For an Image:

Bits/intensity	**Intensities / pixel**	**Number of bytes needed for a 512*512 image**	
1	1	32,768 = 32 KB	Black and White
8	1	262,144 = 256KB	Grayscale
8	3	786,432 = 768KB	Full color

An ideal color scheme would allow for a large range of color while reducing the number of bits needed to represent each pixel's color.

$M=2^m$ m: number of bits per color

M: possible colors

For BMP lookup table size is:

8 bits represents each pixel so 2^8=256 colors

m=24, $M=2^{24}$ =16,777,216 256 colors out of possible 16,000,000

4 bytes per color*256 colors= 1024 bytes in length.

For more information about BMP, refer to Appendix A.

True Color

Human eyes can discriminate between 2^{24} (16,777,216 colors), although many fewer colors can be perceived simultaneously. The actual number certainly varies from person to person and under different conditions of illumination, health, genetics, and attention. A device capable of matching or exceeding the color-resolving power of the human eye under most conditions is said to display *true-color.*

BMP format supports both color lookup tables and true color.

B.2 Color Representation

There are many types of color images. The trichromatic theory tells us that, ideally, three arrays of samples (three components) should be sufficient to present a color image. [1]

In general pixels in a color image have information from the samples of each component, and the color image is comprised of the two-dimensional arrays of the component samples. [2]

RGB is one of the types of a color representation that requires three independent values. Other representations are available that use color components that are closely related to the criteria used to describe color perception: brightness (intensity-whether it is white, gray or black), hue (color- red, green, blue, etc.), and saturation (vivid- strong, pastel).

Varying intensities of light that contain only illumination and no hue are said to be *grayscale* colors. Grayscale colors can be described by a single value that represents the intensity of light. The physics term *luminance* is often used to describe light's intensity, as is the psychological term *brightness*. Luminance and brightness are not equivalent terms, but from a data representation standpoint they are typically stored as single numerical value [3].

One component is for luminance and others are related to hue and saturation are called *luminance-chrominance* representations. The human is less sensitive to rapid changes in the hue and saturation properties of the image than to intensity changes.

B.3 Linear Color transformations

YUV

An image represented in RGB color, can be converted into YUV using the following transformations:

The luminance (Y) can be calculated as:

$Y = 0.3\,R\ + 0.6\,G + 0.1\,B$

where R is Red, G is green, and B is blue value.

The chrominance is defined as the difference between a color and a reference white at the same luminance:

$V\ = R\text{ -}Y$

$U = B - Y$

where V: color ranging from red ($V > 0$) to blue-green ($V < 0$), *U:* color ranging from blue ($U < 0$) to yellow ($U < 0$)

If $R = G = B$, the color differences are zero so it has no chrominance, it produces gray.

YIQ

It uses the same Y coordinate as the YUV

$I = 0.74\ V\text{ - }0.27\ U$

$Q = 0.48\ V + 0.41\ U$

YcbCr

It uses the same Y coordinate as the YUV whereas U and V are scaled and zero-shifted.

$Cb = (U / 2) + 0.5$

$Cr = (V / 1.6) + 0.5$

It may be represented as:

$$\begin{pmatrix} Y \\ Cb \\ Cr \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.5 \\ 0.5 & -0.419 & -0.0813 \end{pmatrix} . \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

APPENDIX C

JPEG Compression Standard

JPEG (Joint Photographic Experts Group) is the name of ISO (International Standardization Organization) working group. The JPEG standard is the first international digital image compression standard for still images including both gray scale and color images.

JPEG standard defines four operation modes: sequential DCT based mode, sequential lossless mode, progressive DCT based mode, and hierarchical mode [6]. In this study, sequential DCT based mode will be examined.

C.1 JPEG Encoder

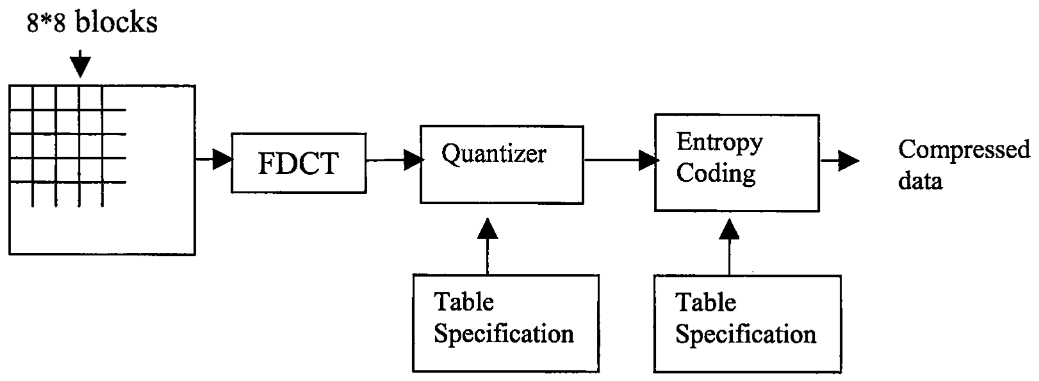


Figure C.1 DCT based JPEG encoder

The original image data, in the range [0, 2^p-1] , are shifted to sign in the range [-2^{p-1}-1]. For a grayscale image, where p = 8, the original image data in the range [0, 255] , are shifted in the range [-128, +127].

The input image samples are grouped into 8*8 blocks, and each block is transformed by forward DCT (FDCT) . After transformation 64 values are obtained that are called as DCT coefficients. The top-left value is referred to as the DC coefficient, other 63 values as the AC coefficients.

In the next step, all 64 DCT coefficients are quantized using a 64-element quantization table, shown in Table C.1.

In the JPEG system the quantized DCT coefficients are always integers, as are the quantization values. The integer representation is chosen such that 8-bit precision image samples to 11 bit-precision quantized DCT coefficients for quantization values of 1. Consequently, for 8 bit precision input samples, a quantization value of 16 produces a quantized DC coefficient with 7-bit precision (128 levels)[2].

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

Table C.1 Typical Luminance Quantization Table

The amplitude of the coefficients are reduced by this step, it contributes little to the quality of the image, and it increases the number of zero-value coefficients. It discards information, which is not visually significiant.

The quantization is performed as:

$$F_q(u,v) = round\left[\frac{F(u,v)}{Q(u,v)}\right]$$

where F(u,v) is the input image data, Q(u,v) is quantization coefficient.

After quantization, coefficients are ordered into the “zig-zag” sequence, as shown below.

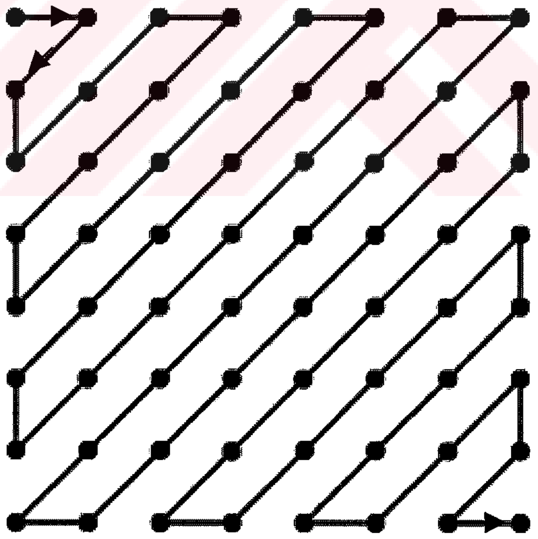


Figure C.2 Zigzag Scan

The zigzag scan places low-frequency quefficients ,which are more likely to be nonzero, before high-frequency coefficients. The next step will be entropy coding will be entropy coding and it facilitate entropy coding. This is confirmed by the experiment presented by [2].

The DC coefficients are coded using the predictive coding techniques as illustrated in Figure C.3

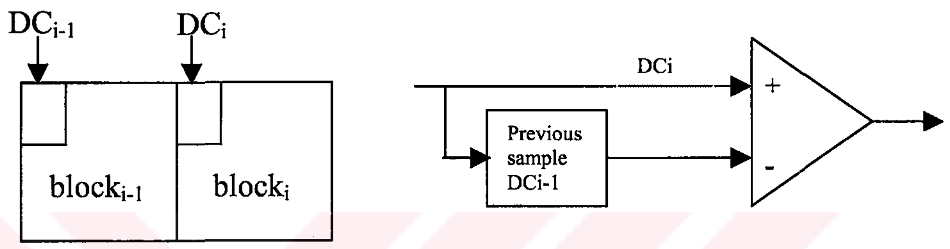


Figure C.3 Predictive Coding for DC coefficients

There is usually a strong correlation between the DC coefficients of adjacent 8*8 blocks thus the predictive coding is used for DC coefficients. Either Huffman Coding or Arithmetic Coding can be used for predictive coding. In this thesis, Huffman Coding is used.

C.2 JPEG Decoder

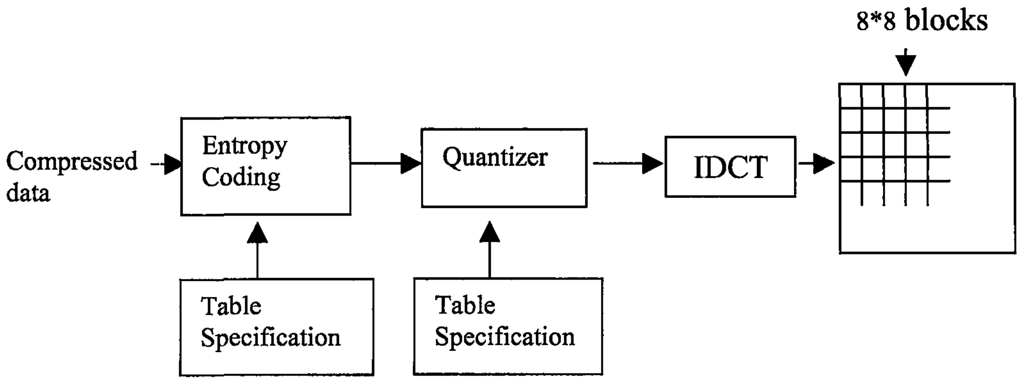


Figure C.4 JPEG Decoder

All steps from encoding process are inversed and implemented in reverse order,as illustrated in Figure C.4.

The dequantization is implemented as

$F(u,v) = F_q(u,v) * Q(u,v)$.

References

1. R. W. G. Hunt., *Measuring Colour* , New York: Halsted Press 1987
2. W. B. Pennebaker, J. L. Mitchell, *JPEG Still Image Data Compression Standard,* New York: Van Nostrand Reinhold, 1993
3. C. Wayne Brown, Barry J.Shepherd, *Graphics File Formats*, Prentice Hall 1995
4. James D.Murray, William Vanryper, *Encyclopeda of Graphics File Formats*, O'Reilly& William Vanryper, Inc. 1994
5. Rafael C. Gonzalez, Richard E.Woods, *Digital Image Processing*, Addison-Wesley, September, 1993
6. Weidong Kou, *Digital Image Compression Algorithms and Standars,* Kluwer Academic Publishers, 1995
7. D. A. Huffman, *A Method for the construction of Minimum –Redundancy Code* , Proceedings of the IRE, Vol. 40, No.9, pp.1098-1101, 1952
8. William K. Pratt , *Digital Image Processing*, John Wiley&Sons , 2001
9. V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures,* Boston, M. A: Kluwer, 1997.
10. K. R. Rao , P. Yip, *Discrete Cosine Transform Algorithms, Advantages, Applications,* Academic Press, 1990
11. David S. Taubman, Michael W. Marcellin, *JPEG2000 Image Compression Fundamentals, Standards and Practice,* Kluwer Academic Publishers, 2002

12. Fred Halsall, *Data Communications, Computer Networks and Open Systems* ,Addison-Wesley, 1996

13. Okan Ersoy, *Fourier-Related Transforms, Fast Algorithms and Applications,* Prentice-Hall PTR, 1997

14. I.Pitas, *Digital Image Processing Algorithms and Applications,* John Wiley&Sons, 2000

15. Stephen J.Soları, *Digital Video and Audio compression,* McGraw-Hill, 1997

16. John Makhoul, IEEE Transactions on Acoustics, Speech, And Signal Processing, *A Fast Cosine Transform in One and Two Dimensions,*Vol. ASSP-28, No. 1, February 1980

17. Henrique S. Malvar, *Fast Computation of the Discrete Cosine Transform and The Hartley Transform,* IEEE Transactions on Acoustics, Speech, And Signal Processing, Vol. ASSP-35, No. 10, October 1987

18. Hsieh S.Hou, *A Fast Recursive Algorithm For Computing the Discrete Cosine Transform,* IEEE Transactions on Acoustics, Speech, And Signal Processing, Vol. ASSP-35, N0. 10, October 1987

19. S:C.Chan and K.L.Ho*, A new Two-Dimensional Fast Cosine Transform Algorithm,* IEEE Transactions on Signal Processing, Vol.39 , No.2, February 1991

20. Ephraim Feig, *Fast Algorithms for the Discrete Cosine Transform,* IEEE Transactions on Signal Processing, Vol. 40, N0.9, September 1992

21. Behrouz Forouzan, Introduction to Data Communications and Networking, McGraw-Hill, 1998