JPEG Image Compression using Huffman Coding and Discretre Cosine Transfer

Basavaraj Patil
Asst. Professor, Dept. of ISE
S.D.M. Institute of Technology
Ujire, Karnataka, India

Avinash S
Asst. Professor, Dept. of ECE
S.D.M. Institute of Technology
Ujire, Karnataka, India

Amit K S
Asst. Professor, Dept. of CSE
S.D.M. Institute of Technology
Ujire, Karnataka, India

***Abstract:* The need for an efficient technique for compression of Images ever increasing because the raw images need large amount of disk space seems to be a big disadvantage during transmission & storage. Even though there are so many compression technique already present a better technique which is faster, memory efficient and simple surely suits the requirements of the user. In this paper we proposed the Lossless method of image compression and decompression using a simple coding technique called Huffman coding. It is simple in implementation and uses less memory. The algorithm has been developed and implemented for the given image using Huffman coding techniques in a MATLAB platform.**

Keywords* — *Compression, JPEG, DCT, Entropy

I. INTRODUCTION

Basically an image is a rectangular array of pixels. Pixel is the smallest unit of the image. The size of images is the number of pixels i.e., width and height. In the real world an image is defined as a function of two real variables. Example image a(x, y) with a as the amplitude like brightness of the image at the real coordinate position (x, y). Image compression is the method of Data compression on digital images. The objective of image compression is to reduce redundancy of the image data in order which be able to store or transmit in an efficient manner.

Image compression can categorized in to two types

a) Lossy compression
b) Lossless Compression

Lossless compression method is preferable for artificial images such as technical drawings, icons. Because of lossy compression methods the use low bit rates are introduced in compression artifacts. Lossless compression methods can also be chosen for high value content such as medical images, image scans made for research purposes. As the name says, in lossless compression methods no information is lost regarding the image. In other words, we say that the reconstructed image from the compressed image is similar to the actual image. The purpose for image compression is to reduce the quantity of data required for representing sampled digital images and to decrease the cost of transmission & storage. Image compression acts as major task in many real time applications like image database, satellite imagery for weather, remote sensing image communications, and earth-resource applications. The compressed images are gray scale values between 0 to 255.

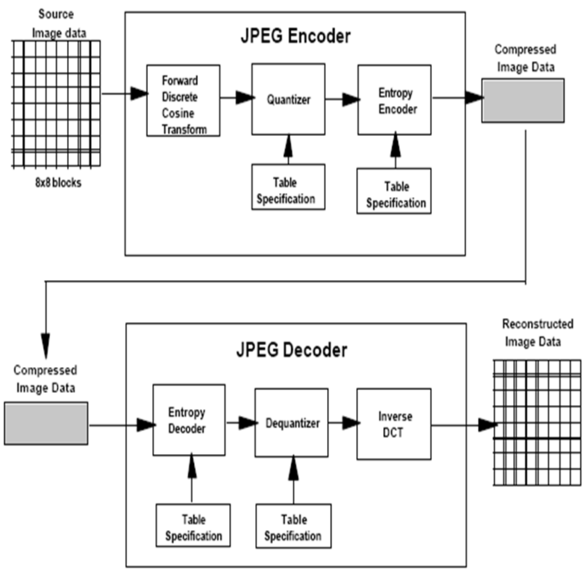


Fig 1: Block Diagram for JPEG Encoder and Decoder

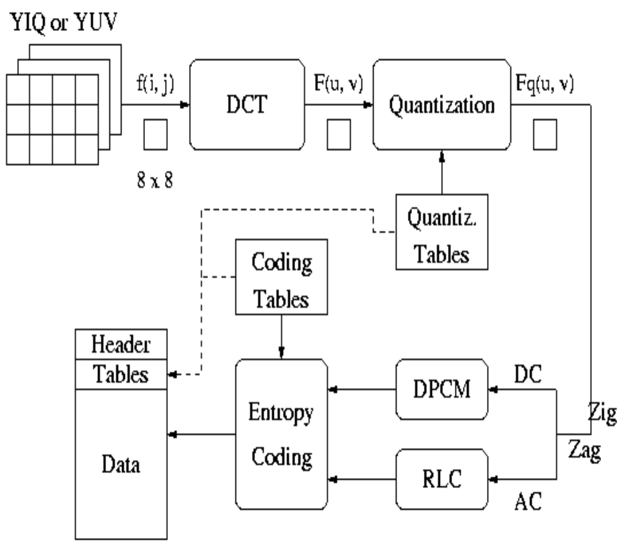


Fig 2: Process View

II. DISCRETE COSINE TRANSFORM

Each 8×8 block of component is converted to a frequency-domain represented using regularized discrete cosine transform (DCT). For example, one such 8×8 8-bit sub image can be represented as:

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}.$$

Before calculating the DCT of the 8×8 block, its values are shifted from a positive value to one centered around zero. In this step, it reduces the dynamic values in the DCT processing stage that follows and the results as follows:

$$\overset{x}{\longrightarrow}$$

$$g = \begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34 \end{bmatrix} \Big\downarrow y.$$

The next step is to receive the two-dimensional inverse DCT i.e., 2D type-III DCT, given by:

$$f_{x,y} = \sum_{u=0}^{7}\sum_{v=0}^{7} \alpha(u)\alpha(v)F_{u,v}\cos\left[\frac{\pi}{8}\left(x+\frac{1}{2}\right)u\right]\cos\left[\frac{\pi}{8}\left(y+\frac{1}{2}\right)v\right]$$

Where

- x is the pixel row, for the integers 0≤ x ≤8.
- y is the pixel column, for the integers 0≤ y ≤8.
- α(u) is defined as above, for the integers 0≤ u ≤8.
- $F_{u,v}$ is the reconstructed approximate coefficient at coordinates(u,v).
- $f_{x,v}$ is the reconstructed pixel value at coordinates (x,y)

$$\overset{u}{\longrightarrow}$$

$$G = \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.13 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.88 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix} \Big\downarrow v.$$

DCT converts spatial domain to frequency domain. DCT converts the image in appropriate format so that compression can be done in the form of Quantization but DCT does not do any compression.

III. QUANTIZATION

Quantization is achieved by compressing a range of values to a single quantum value. Quantization is the step where most of the compression takes place. Quantization is achieved by dividing transformed image matrix by the quantization matrix used. Values of the resultant matrix are then rounded off. In the resultant matrix coefficients situated near the upper left corner have lower frequencies.
A typical quantization matrix, as specified in the original JPEG Standard, is as follows:

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

Using the above quantization matrix along with the DCT coefficient matrix from above results:

$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

IV. DC & AC CODING WITH ZIGZAG SEQUENCE

It method accomplishes the compression lossless by encoding the quantized DCT coefficient. After quantization process the DC coefficient is considered as separate from 63 AC coefficients. The DC coefficient is calculated as the average value of the 64 image samples. The quantized DC coefficients are encoded as the difference between the new DC coefficient and previous coefficient of 8x8 blocks in encoding order as there is strong correlation between DC coefficients of adjacent blocks. DC coefficient consists of significant portion of the energy of image, so that the difference between them provides sufficient compression. After this step, all set of quantized coefficients are structured into the “Zigzag” sequence. The Zigzag scanning of the 2D quantized coefficients arranges the frequencies into a 1D stream of frequencies from lower to higher. This arrangement helps to make possible the entropy coding by placing high frequency coefficients after low frequency coefficients.

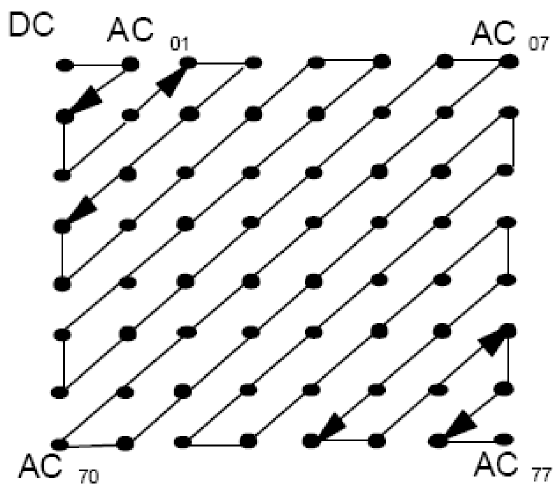


Fig 3: Zig-Zag Sequence

The zigzag sequence for the fig 3 quantized coefficients is shown in fig 4.

−26							
−3	0						
−3	−2	−6					
2	−4	1	−3				
1	1	5	1	2			
−1	1	−1	2	0	0		
0	0	0	−1	−1	0	0	
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	
0	0	0	0	0	0		
0	0	0	0	0			
0	0	0	0				
0	0	0					
0	0						
0							

Fig 4: Zig-Zag Sequeunce for quantized coefficients

V. ENTROPY CODING

Entropy coding attains the addition lossless compression by encoding the quantized DCT coefficient more densely based on their statistical distinctiveness. In JPEG two Entropy coding methods are available

1) Huffman Coding
2) Arithmetic Coding

Entropy coding is processed in 2-phases.

a) During the first phase, the zigzag sequence of quantized coefficient is converted into set of transitional symbols.
b) In the second phase the symbols are converted in to a data stream in which the symbols without having externally identified boundaries.

Huffman coding uses Huffman table defined by application for compress an image and then the same table is used for decompression. These Huffman tables are predefined or computed specifically for a given image during initialization, prior to compression.

Arithmetic coding doesn't requires tables like Huffman coding because it is able to adapt to the image statistics as it encodes the image. Arithmetic coding is a little complex than Huffman coding for certain implementation, for example, the highest-speed hardware implementation. Trans coding between two coding method is possible by simply entropy decoding with one method and entropy recoding with the other.
In this work we will use a lossless compression and decompression through a technique called Huffman coding (i.e. Huffman encoding and decoding).

The Huffman coding technique collects unique symbols from the source image and calculates its probability value for each symbol and sorts the symbols based on its probability value. Further, from the lowest probability value symbol to the highest probability value symbol, two symbols combined at a time to form a binary tree. Moreover, allocates zero to the left node and one to the right node starting from the root of the tree. To obtain Huffman code for a particular symbol, all zero and one collected from the root to that particular node in the same order.
The basic concept in Huffman coding is to assign some shortest code words to the input blocks with more probabilities and long code words to those with low probabilities. A Huffman code is designed by combing the two least probable characters together and looping this same process until there remains only one character remaining. A code tree is thus generated and the Huffman code is obtained from the labeling of the code tree.

Huffman code procedure is based on the two observations More frequently occurred symbols will have shorter code words than symbol that occur less frequently. The two symbols that occur least frequently will have the same length.

The Huffman code is designed by merging the lowest probable symbols and this process is repeated until only two probabilities of two compound symbols are left and thus a code tree is generated and Huffman codes are obtained from labeling of the code tree. In Huffman's procedure is to code each reduced source, starting with the smallest source and working back to its original source. The minimal length binary code for a two-symbol source, of course, is the symbols 0 and 1.

After the code has been created, coding and/or decoding is accomplished in a simple look-up table manner. The code itself is an instantaneous uniquely decodable block code. It is called a block code, because each source symbol is mapped into a fixed sequence of code symbols. It is instantaneous, because each code word in a string of code symbols can be decoded without referencing succeeding symbols. It is uniquely decodable, because any string of code symbols can be decoded in only one way. Thus, any string of Huffman encoded symbols can be decoded by examining the individual symbols of the string in a left to right manner.

At first we have as much as the compressor does a probability distribution. The compressor made a code table. The decompressor does not use this method though. It instead keeps the whole Huffman binary tree, and of course a pointer to the root to do the recursion process. In our implementation we'll make the tree as usual and then you'll store a pointer to last node in the list, which is the root. Then the process can start. We'll navigate the tree by using the pointers to the children that each node has. This process is done by a recursive function which accepts as a parameter a pointer to the current node, and returns the symbol.
Taking the DCT coefficient matrix (after adding the difference of the DC coefficient back in)

$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

And taking the entry-for-entry product with the quantization matrix from above results in

$$\begin{bmatrix} -416 & -33 & -60 & 32 & 48 & -40 & 0 & 0 \\ 0 & -24 & -56 & 19 & 26 & 0 & 0 & 0 \\ -42 & 13 & 80 & -24 & -40 & 0 & 0 & 0 \\ -42 & 17 & 44 & -29 & 0 & 0 & 0 & 0 \\ 18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

This closely resembles the original DCT coefficient matrix for the top-left portion. Rounding the output to integer values (since the original had integer values) results in an image with values (still shifted down by 128)

$$\begin{bmatrix} -66 & -63 & -71 & -68 & -56 & -65 & -68 & -46 \\ -71 & -73 & -72 & -46 & -20 & -41 & -66 & -57 \\ -70 & -78 & -68 & -17 & 20 & -14 & -61 & -63 \\ -63 & -73 & -62 & -8 & 27 & -14 & -60 & -58 \\ -58 & -65 & -61 & -27 & -6 & -40 & -68 & -50 \\ -57 & -57 & -64 & -58 & -48 & -66 & -72 & -47 \\ -53 & -46 & -61 & -74 & -65 & -63 & -61 & -45 \\ -47 & -34 & -53 & -74 & -60 & -47 & -47 & -41 \end{bmatrix}$$

And adding 128 to each entry

$$\begin{bmatrix} 62 & 65 & 57 & 60 & 72 & 63 & 60 & 82 \\ 57 & 55 & 56 & 82 & 108 & 87 & 62 & 71 \\ 58 & 50 & 60 & 111 & 148 & 114 & 67 & 64 \\ 65 & 55 & 66 & 120 & 155 & 114 & 68 & 70 \\ 70 & 63 & 67 & 101 & 122 & 88 & 60 & 78 \\ 71 & 71 & 64 & 70 & 80 & 62 & 56 & 81 \\ 75 & 82 & 67 & 54 & 63 & 65 & 66 & 83 \\ 81 & 95 & 75 & 54 & 68 & 81 & 81 & 87 \end{bmatrix}.$$

The formula of Mean Square Error, MSE is as followed:

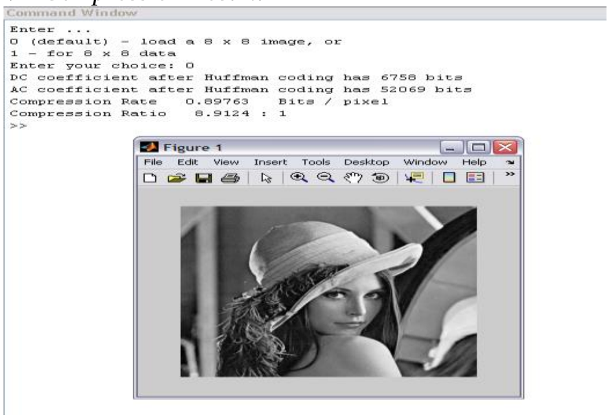
$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j-0}^{n-1} \left\| I - im \right\|^2$$

VI. HUFFMAN CODING AND DECODING ALGORITHM

Step1- Read the image on to the workspace.
Step2- Convert the given colour image into grey level image.
Step3- Call a function which will find the symbols (i.e. pixel value which is non-repeated).
Step4- Call a function which will calculate the probability of each symbol.
Step5- Probability of symbols are arranged in decreasing order and lower probabilities are merged and this step is continued until only two probabilities are left and codes are assigned according to rule that the highest probable symbol will have a shorter length code.
Step6- Further Huffman encoding is performed i.e. mapping of the code words to the corresponding symbols will result in a compressed data.
Step7- The original image is reconstructed i.e. decompression is done by using Huffman decoding.
Step8- Generate a tree equivalent to the encoding tree.
Step9- Read input character wise and left to the table II until last element is reached.
Step10-Output the character encodes in the leaf and returns to the root, and continues the step9 until all the codes of corresponding symbols are known.

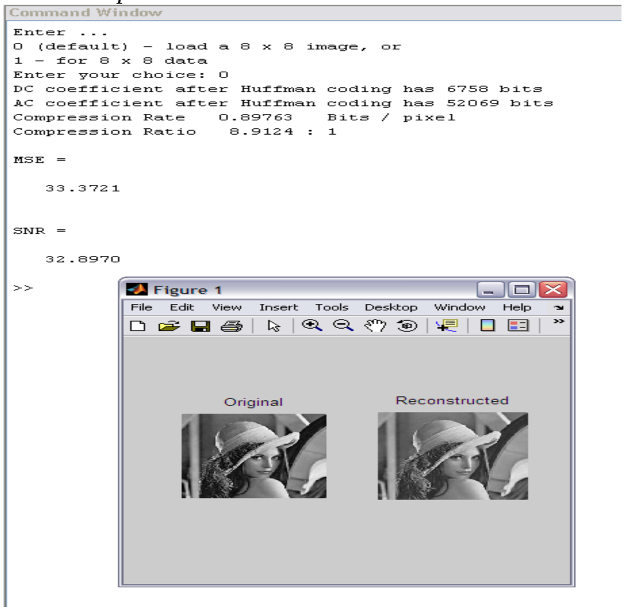
VII. RESULTS

a. Compression Result:



```
Enter ...
0 (default) - load a 8 x 8 image, or
1 - for 8 x 8 data
Enter your choice: 0
DC coefficient after Huffman coding has 6758 bits
AC coefficient after Huffman coding has 52069 bits
Compression Rate    0.89763    Bits / pixel
Compression Ratio    8.9124 : 1
>>
```

b. Decompression Result:



```
Command Window
Enter ...
0 (default) - load a 8 x 8 image, or
1 - for 8 x 8 data
Enter your choice: 0
DC coefficient after Huffman coding has 6758 bits
AC coefficient after Huffman coding has 52069 bits
Compression Rate    0.89763    Bits / pixel
Compression Ratio    8.9124 : 1

MSE =

   33.3721


SNR =

   32.8970

>>
```

VIII. CONCLUSION & FUTURE WORK

The experiment shows that the higher data redundancy helps to achieve more compression. The above presented a new compression and decompression technique based on Huffman coding and decoding for scan testing to reduce test data volume, test application time. We conclude that Huffman coding is efficient technique for image compression and decompression to some extent. The compression ratio achieved by using Discrete Cosine Transform in Huffman coding can be further increased by using Discrete Fourier Transform.

REFERENCES

[1] A.M Raid, W.M.Khedr, M. A. El-dosuky,Wesam Ahmed,"Jpeg Image Compression Using Discrete Cosine Transform -A Survey",(IJCSES) Vol.5, No.2, April 2014.
[2] Jagadish h. Pujar,lohit m. Kadlaskar," A New Lossless method of image compression and decompression using huffman coding techniques",Journal of Theoretical and Applied Information Technology.
[3] Mridul Kumar Mathur,Seema Loonker,Dr. Dheeraj Saxena," Lossless Huffman Coding Technique For Image Compression Andreconstruction Using Binary Tree",IJCTA,Jan-Feb 2012,Vol3(1),76-79.
[4] Aarti,"Performance Analysis of Huffman Coding Algorithm",IJARCSSE, Volume3,Issue 5,May 2013.
[5] Deepak Kumar Jain,Devansh Gaur,Kavya Gaur,Neha Jain, "Image Compression using Discrete Cosine Transform and Adaptive Huffman Coding"IJETTCS,Volume 3, Issue 1, January-February 2014.
[6] Harjeetpal singh,Sakhi Sharma,"Hybrid Image Compression Using DWT, DCT & Huffman Encoding Techniques",International Journal of Emerging Technology and Advanced Engineering, Volume 2, Issue 10, October 2012.
[7] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," IEEE Trans. on Computers, vol. C-23, pp. 90-93,1974.
[8] A.B.Watson,"Image Compression using the DCT", Mathematical Journal, 1995,pp.81-88.
[9] D.A.Huffman, A Method for the construction of Minimum-redundancy Codes, Proc. IRE, vol.40, no.10, pp.1098-1101,1952.
[10] Léger, A., Omachi, T., and Wallace, G. The JPEG still picture compression algorithm. In Optical Engineering, vol. 30, no. 7 (July 1991), pp. 947954.
[11] Robert M. Gray,IEEE, and David L. Neuhoff, IEEE"Quantization", IEEE Trans. on Information Theory, Vol. 44, NO. 6,pp. 2325-2383, OCTOBER 1998. (Invited paper).
[12] Jain, A. K. (1989), Fundamentals of Digital Image Processing, Prentice Hall International, Inc.
[13] P.Ramesh Babu, Digital Image Processing. Scitech Publications., 2003
[14] Rudra Pratap, Getting Started With MATLAB 7. Oxford University Press, 2006
[15] Bracewell, R.N., Two Dimensional Imaging. Prentice Hall, 1995.