A Loss-less Image Compression Technique Using Divide and Conquer Strategy

Khalid Imtiaz
Computer Science & Engineering Department, University of Engineering and Technology, Lahore, Pakistan
engr.khalidimtiaz@gmail.com

Abstract– **Data or image compression is the method of reducing the size of data. In a compressed data set a fewer bits are used to represent the same data element as in the uncompressed data set. Huffman Coding uses the probability of occurrence of a data value in the data set and assigns a specific code to this probability, replacing the original value with this special code. This paper makes use of this technique by dividing the data set into two parts and then applying Huffman coding on each divided part separately. It is found that the compression rate is increased if the data part is divided before applying Huffman coding.**

Keywords– **Huffman Coding, Image Compression, Matrix and Image**

I. INTRODUCTION

Media such as videos, images, sound recordings etc are nowadays expanding exponentially in size, quality and length [1]. When this huge amount of data is needed to be transmitted over a network the problem arises because a huge bandwidth is required to send this data and due to slow progress in networking innovation, it becomes very hard to transmit the data over the network of present day. A solution is presented to compress the data before transmitting it over the network. The data in compressed form might not even be in a readable form but since the size is reduced it is acceptable. The data, compressed into a format which might not be readable, is sent over the network in this reduced size and then on the destination it is decompressed and is converted back into a readable form. This paper uses an image as its data source which will be compressed using Huffman encoding technique.

An image is a sampled array of continuous data stored in a PC in digital form. A digital image is two or three dimensional array of variating values. A monotone image can be a 2-D array of values with each of the value containing 8 bits. It means that each pixel in the image can have a value ranging between 0 and 255. If there is a 1024 x 1024 x 8 bit image, it takes 8192Kb or 8Mb. Since storage devices are relatively cheaper now so storing a single or few 8Mb pictures is affordable. But the problem arises when this 8Mb picture is required to be transferred over the internet. It takes a lot of bandwidth for an image. Even for the storage purposes, if it is required to store thousands of images of an even larger resolution, it gets very costly to store in the images if they remain in this pattern. Here arises a need for image compression. Several algorithms are being used for image compression but each one has its own give and take.

Huffman coding is used to form a minimum redundancy code which eliminates the redundant values from any data set and replace them with the probability of occurrence of a particular value in the data set. Huffman coding is being used for file, image and sound compression i.e. JPEG and PNG image format, GZIP and BZIP file compressing algorithms and MP3 and MPEG etc [2].

The main steps in the process of image compressing are shown in Fig. 1. An image is provided as input which is compressed using any algorithm that reduces the picture size. This image in the compressed form may not be in a readable state unless it is decompressed (or decoded) which is a similar but reverse process of the compressing algorithm.

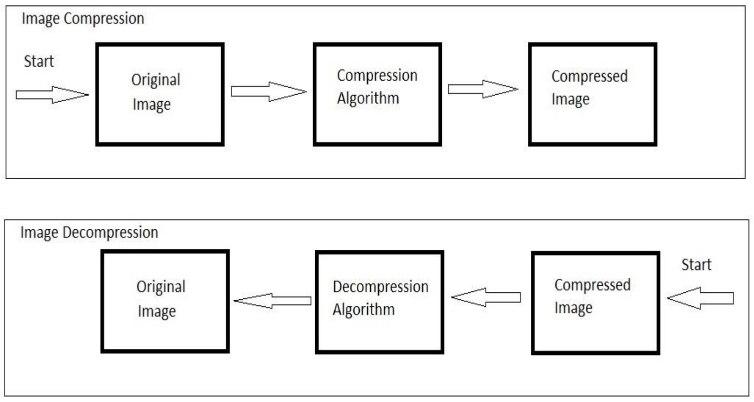


Fig. 1. Compression & Decompression

In this paper, we take a look at the basic structure and representation of an image, how an image is compressed using Huffman Coding, what amount of compression can be achieved using the classical Huffman coding technique, what amount of compression is achieved by dividing the data set before applying Huffman coding and the comparison of compression rates of two techniques. The next section focuses on the work that has already been implemented. The proposed solution sections focuses on the new technique prescribed for image compression. Then in the next section, the results are compared with the classical image compression techniques and then the paper is concluded.

II. RELATED WORK

Huffman coding is an encoding technique which makes use of variable length codes to represent the source data. The source data values are mapped into variable length binary values which are derived using a probability table that is the core part of Huffman coding [3].

This technique will be applied on a picture to elaborate further. A picture is an array of pixels. Each pixel in an image is actually a value of a certain color. In a grayscale image (any image that is converted into a range of gray and white color only) each pixel value represents a shade of gray. For an 8-bit image the shades of gray will range between 0 and 255 where 0 may represent the darkest shade of gray and 255 may represent the lightest shade of gray (or white) or vice versa. Since a digital image is a mere representation of analog data converted into the form of 0’s and 1’s for the purpose of storing/representing it into a PC, a lot of detail is lost. An analog set of data passes through the steps of sampling and quantization when it is desired to be converted into a digital image. As analog data is continuous, it cannot be stored in a PC in its analog form because a PC only deals with digital (or discrete) data. So, the analog data is sampled at short intervals and this sample is then given different values in a given range (as for the case of an 8-bit image, the sampled data will be given the values between the range 0 and 255). The larger the sampling ratio, the more detailed representation of the analog data it achieves. So, a grayscale image of size 256 x 256 pixels and in it, each pixel represented by 8-bits will have a minimum pixel value of 0 and the largest pixel value of 255. This structure of image is very important in image compression.

Delphi image control tool is also used to convert an image into array. Image control can be used to display a graphical image - Bitmap (BMP), Icon (ICO), GIF, Metafile (WMF), JPEG, etc. An Algorithm that is generated in Delphi using Huffman coding converts the image into binary codes and then removes the redundant codes from the gray scale image, compresses the BMP image (especially gray scale image) and then reconstructs the image using binary tree [4].

Reducing the image file size by decreasing the number of bits per pixel required to represent it also reduces the transmission time and bandwidth required for the compressed image to transmit it over network. After all the image is reconstructed by decoding the Huffman codes. All the compression is done by removing the redundant bits from every pixel of image and reconstructing the image in original form using Huffman coding [2].

A picture contains redundant information due to neighboring pixels which are linked together and contain similar information. Image compression means removing redundant information from image pixels by keeping image resolution as fine as possible. Image compression is done by both Huffman coding and also through arithmetic coding. Huffman image compression shows better results even when the size of image is large, it compresses the image in a way that result is more close to real image [5].

To compress an image and get more efficient results we use Lempel ZivWelch algorithm after using Huffman coding. In first stage we use Huffman coding algorithm to compress the image that gives a Huffman tree and Huffman codes. After that all the Huffman codes are concatenated and another compression technique that is Lempel ZivWelch coding is applied to compress it even further. In the last stage Retinex algorithm is applied on compressed image to enhance the image quality and colors of image. So this compressed image is very similar to the uncompressed image that takes more space for specifying the features of each pixel [6].

Bitmap and tiff images use 32 bits to encode the color at each pixel hence they require a lot of information to store the image. On the other hand, JPEG images have three levels of compression, chunks of discrete information are patterns of 2D arcs. Secondly the information that human eye cannot resolve is taken out from image and after that Huffman coding is applied on image for further compression [7].

Image or videos require very large storage space when these are in raw form. A hybrid technique that uses discrete cosine transform for compaction along with discrete wavelet transform for multi-resolution images uses Huffman coding algorithm for the purpose of image compression. In this approach similarly, bit patterns are encoded using Huffman coding scheme and compression is achieved [8].

Two types of compressions categories are being used today for the purpose of image compression. The first is the lossy compression method in which some information is lost in the image when it passes through the process of image compression. Actually, the loss of information is the key factor in achieving compression in this type of image compression techniques. However, the image is still informative although some loss has been experienced by it. The other technique is the lossless technique in which no information is lost during the compression process and the image presents all the information as it was in the uncompressed image [9].

Different data redundancy schemes are present in the images. Inter-pixel redundancy is related with different pixels of the image having the same data value. This redundant information can be removed using some technique to achieve image compression. Data-redundancy scheme is used when image has coding redundancy. And psycho-visual redundancy scheme deals with the identification of such data values that are invisible to human eye [10]. These redundant values can be removed from an image for the purpose of achieving image compression.

Along with the compression of data, security of information while compression is also an important and integral part of image processing. Algorithms are devised for achieving both data compression and data security [11].

III. PROPOSED SOLUTION

An image is a matrix of pixels; we can represent it as follows:

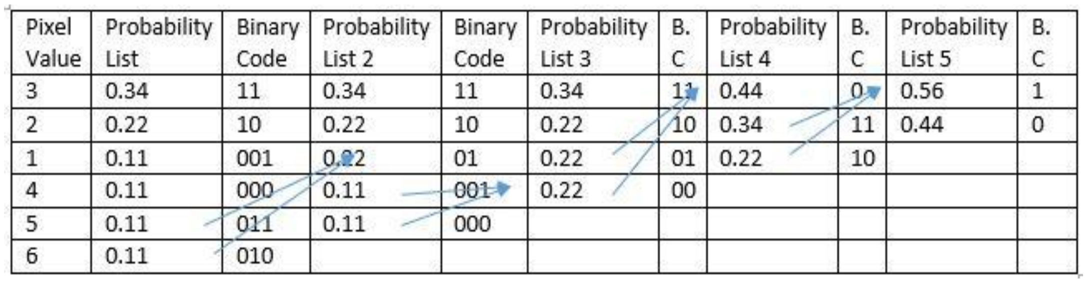
2	6	3
5	4	3
1	2	3

Binary representation of this matrix (image) will be as follows:

00000010	00000110	00000011
00000101	00000100	00000011
00000001	00000010	00000011

Since each pixel is represented by 8 bits, it will take the size of 72bits or 9 bytes. This size is too large for a 3x3 resolution, 8-bit, grayscale image. If the same technique is used for a colored image it will take 27 bytes to store a 3x3 resolution image since a colored image uses three 3x3 matrices and each 3x3 matrix represents a Red, Green or Blue frame (RGB) which are combined as one on top of the other to make a colored image. This storage technique is very costly so this image needs to be compressed. Huffman coding will be used for the compression of this image.

Huffman coding assigns a probability value to each of the redundant or non-redundant values in a data set and lists down the values on base of their probabilities [12]. A very redundant value will have a higher probability and a non-redundant value will have a very low probability. Probabilities are listed against each redundant value where the highest probability value will come 1st and the probability will decrease down the list. After listing down the probability list, starting from the lowermost probability values, two values are added and a new list is created where the least two probability values in the previous list are replaced with the sum of those two least values and the list is re-ordered. The same step is repeated for all the lower values until we arrive on the list having only two values. All the values in all the list are then assigned a unique binary value and the image then uses these probabilities in binary values to represent the data values. This technique allows for storing a larger data value in a shorter binary code based on its probability of occurrence. Assignments of binary values to each probability value in the lists are done using a binary tree structure. A zero is allocated to each left node and a 1 to each right node starting from the root node to all the branches. This way a unique code is generated for each node reading from the root node to the desired node. For the above described data matrix, encoding technique is shown in Fig. 2.



Pixel Value	Probability List	Binary Code	Probability List 2	Binary Code	Probability List 3	B. C	Probability List 4	B. C	Probability List 5	B. C
3	0.34	11	0.34	11	0.34	11	0.44	0	0.56	1
2	0.22	10	0.22	10	0.22	10	0.34	11	0.44	0
1	0.11	001	0.22	01	0.22	01	0.22	10		
4	0.11	000	0.11	001	0.22	00				
5	0.11	011	0.11	000						
6	0.11	010								

Fig. 2. Huffman Coding

Now if we represent the same matrix using the new codes based upon the probability, it will look like the following:

10	010	11
011	000	11
001	10	11

We can calculate the size this new matrix will take as follows:

Size= Probability * (number of bits) = (0.34) (2) +(0.22) (2) +(0.11) (3) + (0.11) (3) + (0.11) (3) + (0.11) (3)

=2.44 bits/symbol

So for 9 elements as in the above array we will need 2.44 * 9 =21.96 bits, or 2.745 bytes

In this paper a different approach to image compression is presented. A grayscale image is an array of different numbers and each number represents a specific gray level. Huffman coding is applied on these numbers to obtain image compression. This paper introduces a new approach in a divide and conquers way. An image is divided into two parts and Huffman coding is applied on each part separately. Since the size of the image is now half of the original image, the number of Huffman coding probabilities will now decrease. Due to this decrease in the probability values, a fewer binary codes are required to represent each probability value as compared to the original image. This paper focuses on only the compression of the image which is required to be transmitted. The decompression and conversion of the image is not in the scope of this paper. This technique is applied on the above array as follows. First the array is divided into two parts, since it is a 3 x 3 array, the sizes cannot be same, so it is divided as:

A = 2 6 3

$$B = \begin{matrix} 5 & 4 & 3 \\ 1 & 2 & 3 \end{matrix}$$

The probability lists for matrices A and B are shown in Fig. 3 and Fig. 4, respectively.

Pixel Value	Probability List	Binary Code	Probability List 2	Binary Code
2	0.34	0	0.66	1
6	0.33	11	0.34	0
3	0.33	10		

Fig. 3. Probability List for Matrix A

Pixel Value	Probability List	Binary Code	Probability List 2	Binary Code	Probability List 3	Binary Code	Probability List 4	Binary Code
3	0.32	11	0.34	0	0.34	0	0.66	1
5	0.17	101	0.32	11	0.34	10	0.34	0
4	0.17	100	0.17	101	0.32	11		
2	0.17	01	0.17	100				
1	0.17	00						

Fig. 4. Probability List for Matrix B

According to these probability lists, the above two matrices after applying Huffman coding become

$$A = \begin{matrix} 0 & 11 & 10 \end{matrix}$$

$$B = \begin{matrix} 101 & 100 & 11 \\ 00 & 01 & 11 \end{matrix}$$

So for 9 elements as in the above array we will need 19 bits, or 2.375 bytes.

For storing the same matrix, this split approach requires 19 bits, whereas Huffman coding requires 22 bits. It is clear that this approach provides better results than the classical Huffman coding technique. The different stages of this compression method are shown in Fig. 5.

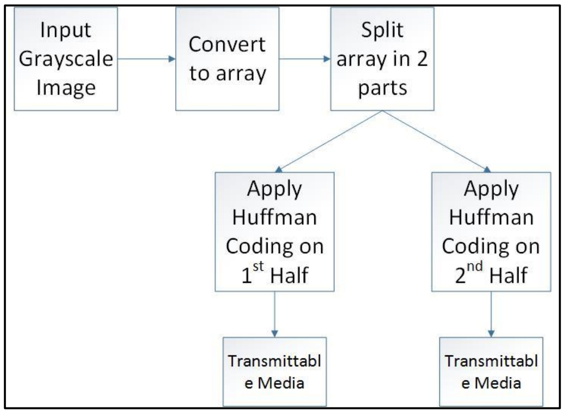


Fig. 5. Flow of Split Algorithm

This algorithm is implemented in Matlab. First a grayscale image is imported and it is converted into a matrix. This image is then split into two equal parts. Huffman dictionaries for these two images are created separately and then Huffman Coding is applied on each part. The resultant images are converted into binary and their binary lengths are measured.

The algorithm for this method is as follows:

- Read an image and convert it into matrix
- Divide the matrix into two parts
- Apply Huffman Coding on 1st part
- Apply Huffman Coding on 2nd part
- Convert the two (halved) matrices into binary
- Compare the length as follows
  - Binary Length of 1st compressed part + Binary length of 2nd compressed part < Binary length of the whole image compressed

Let the whole compressed image be represented by the letter z, the first divided and compressed part of the image be represented by x and the second compressed part be represented by y, then the result can be represented through the following formula:

$$Z > X + Y$$

This formula shows that the divided and compressed images have a smaller binary length as compared to the original image compressed using Huffman coding.

IV. RESULTS AND COMPARISONS

The input image is cameraman.tif as shown in Fig. 6. It is split into two parts as "a1" as shown in Fig. 7 and "a2" as shown in Fig. 8.

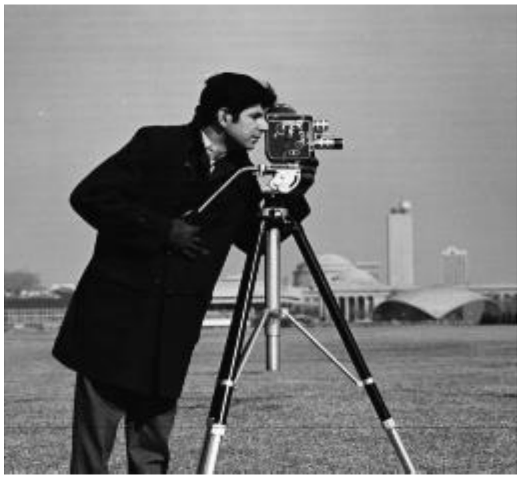


Fig. 6. Cameraman.tif

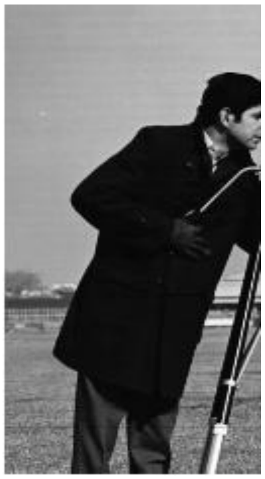


Fig. 7. Cameraman1.tif

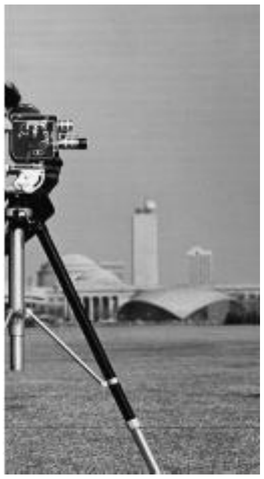


Fig. 8. Cameraman2.tif

Image	Original Binary Length (Bits)	Length after applying Huffman coding (Bits)
Cameraman.tif	524288	461690
Cameraman1.tif	262144	217810
Cameraman2.tif	262144	230686

Fig. 9. Results

S. No.	Image Name	Full Image (Bits)	Part1 Length (Bits)	Part2 Length (Bits)	Combined length (Bits)	Difference (Bits)
1	testpat	166114	83087	82801	165888	226
2	cameraman	461690	217810	230686	448496	13194
3	coins	416323	214676	193962	408638	7685
4	circuit	472400	224976	235436	460412	11988
5	cell	298320	158831	138213	297044	1276
6	eight	343408	149962	175270	325232	18176
7	glass	485050	231372	240548	471920	13130
8	mandi	464861	230684	227193	457877	6984
9	moon	357895	182490	164017	346507	11388
10	rice	461821	231258	229743	461001	820
11	pout	407449	202734	197579	400313	7136
12	tire	456412	230584	224682	455266	1146
13	westconcordorthophoto.png	507551	252622	253588	506210	1341
14	concordaerial	462606	229478	226769	456247	6359
15	fabric	479097	239021	238760	477781	1316
16	football.jpg	441259	221400	218016	439416	1843
17	gantrycrane.png	437636	218349	207607	425956	11680
18	greens.jpg	486336	246370	238349	484719	1617
19	hestain.png	477829	239940	235502	475442	2387
20	peppers.png	459912	229212	228505	457717	2195

Fig. 10. Image Size in Binary for Full and Divided Images

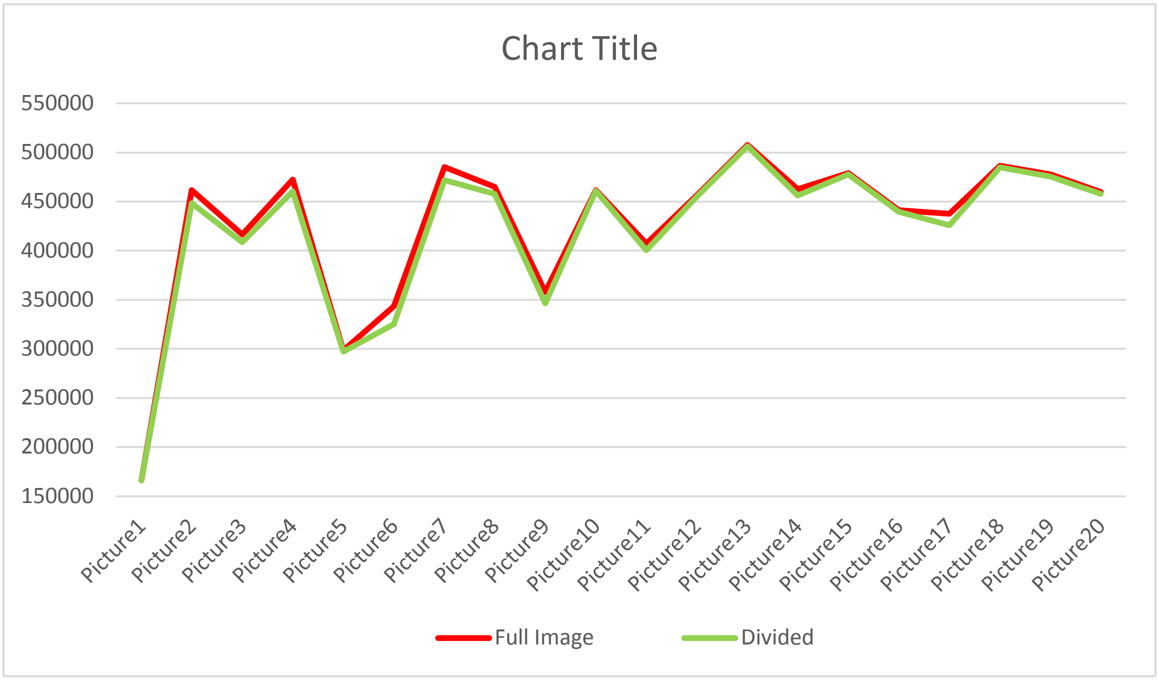


Fig. 11. Image size for Full and Binary Images

Huffman Coding is then applied to Cameraman.tif and then on a1.tif and a2.tif separately. The resultant compressed images are then converted into binary and their binary lengths are compared. Results are summarized in Fig. 9.

It is evident from the above table that if Huffman coding is applied on the original image 11.94\% of compression is achieved. If the split method is used 14.46\% compression is achieved.

Using the same method, 20 different images are used and the results are summarized in Fig. 10. The images have been taken from the built-in Matlab library.

The above table is represented in the graphical form as shown in Fig. 11.

The graph shows that for a number of sample pictures, the proposed scheme of image compression has a significant drop in the image size as compared to the image compressed using classical Huffman Coding technique.

V. CONCLUSIONS AND FUTURE WORK

This paper concludes that if an image is split into two parts before applying Huffman coding algorithm, the compression ratio is increased. The two compressed parts take lesser space as compared to the whole image compressed. This technique can be used for storing or for transmission purposes. It is found that if an image is divided into two parts and then compressed, it gives out a better compression ratio as compared to the Huffman coding being applied upon the whole image. In future, the split method will be applied on the images by splitting them into more than one part to see if the compression rate further increases.

REFERENCES

[1]. Shahbahrami, A., Bahrampour, R., Rostami, M. S., Mobarhan, M. A, "Evaluation of Huffman and Arithmetic Algorithms for Multimedia Compression Standards", Journal of Theoretical and Applied Information Technology, April 24, 2016.

[2]. Pujar, J. H., Kadlaskar, L. M, "A New Lossless Method of Image Compression and Decompression Using Huffman Coding Techniques", Journal of Theoretical and Applied Information Technology, April 23, 2016.

[3]. Mathur, M. K., S. L., & Saxena, D. D. (2012). Lossless Huffman Coding Technique for Image Compression and Reconstruction Using Binary Trees", Int. J. Comp. Tech. Appl., 3(1), 76-79, May 12, 2016.

[4]. A. S., "Lossless Image Compression and Decompression Using Huffman Coding", International Research Journal of Engineering and Technology (IRJET), 02(1), 240-247, April 2015.

[5]. D. K., and K. K., "Huffman Based LZW Lossless Image Compression Using Retinex Algorithm", International Journal of Advanced Research in Computer and Communication Engineering, 02 (8), August 2013.

[6]. Sayood, K., "Introduction to Data Compression", Morgan Kaufmann, 2006.

[7]. Bharath, K. N., & G. P., "Hybrid Compression Using DWT-DCT and Huffman Encoding Techniques for Biomedical Image and Video Applications", International Journal of Computer Science and Mobile Computing, 2(5), 255-261, May 2013..

[8]. R. A., & Kamal, I. W., "A New Lossless Image Compression Technique Based on Bose, Chandhuri and Hocquengham (BCH) Codes", International Journal of Software Engineering and Its Applications, 5(3), 15-22, July 2011.

[9]. B. D., & V. A., "Image Compression Using DCT and DWT", International Journal of Innovative Research in Computer and Communication Engineering, 3(6), June 2015.

[10]. Ashok, D. M., & Yaragunti, D. S., "Image Masking and Compression using Bit shifting", Global Journal of Computers & Technology, 2(1), 2394-501x, 66-74, June, 2015.

[11]. Sharma, M., "Compression Using Huffman Coding. IJCSNS International Journal of Computer Science and Network Security, 10(5), 133-141, 2010.