A NEW LOSSLESS METHOD OF IMAGE COMPRESSION AND DECOMPRESSION USING HUFFMAN CODING TECHNIQUES

[1]JAGADISH H. PUJAR, [2]LOHIT M. KADLASKAR

[1] Faculty, Department of EEE, B V B College of Engg. & Tech., Hubli, India-580 031

[2] Student, Department of EEE, B V B College of Engg. & Tech., Hubli, India-580 031

E-mail: jhpujar@bvb.edu , lmkadlaskar@gmail.com

ABSTRACT

The need for an efficient technique for compression of Images ever increasing because the raw images need large amounts 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. This technique is simple in implementation and utilizes less memory. A software algorithm has been developed and implemented to compress and decompress the given image using Huffman coding techniques in a MATLAB platform.

Keywords: *Huffman codes, Huffman encoding, Huffman decoding, symbol, source reduction.*

1. INTRODUCTION

A digital image obtained by sampling and quantizing a continuous tone picture requires an enormous storage. For instance, a 24 bit colour image with 512x512 pixels will occupy 768 Kbyte storage on a disk, and a picture twice of this size will not fit in a single floppy disk. To transmit such an image over a 28.8 Kbps modem would take almost 4 minutes. The purpose for image compression is to reduce the amount of data required for representing sampled digital images and therefore reduce the cost for storage and transmission. Image compression plays a key role in many important applications, including image database, image communications, remote sensing (the use of satellite imagery for weather and other earth-resource application). The image(s) to be compressed are gray scale with pixel values between 0 to 255. There are different techniques for compressing images. They are broadly classified into two classes called lossless and lossy compression techniques. As the name suggests in lossless compression techniques, no information regarding the image is lost. In other words, the reconstructed image from the compressed image is identical to the original image in every sense. Whereas in lossy compression, some image information is lost, i.e. the reconstructed image from the compressed image is similar to the original image but not identical to it. In this work we will use a lossless compression and decompression through a technique called Huffman coding (i.e. Huffman encoding and decoding).

It's well known that the Huffman's algorithm is generating minimum redundancy codes compared to other algorithms. The Huffman coding has effectively used in text, image, video compression, and conferencing system such as, JPEG, MPEG-2, MPEG-4, and H.263 etc.. 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 main objective of this paper is to compress images by reducing number of bits per pixel required to represent it and to decrease the transmission time for transmission of images .and then reconstructing back by decoding the Huffman codes.

The entire paper is organized in the following sequence. In section -1 need for the compression is stated, section-2 various types of data redundancies are explained, section-3Methods of compressions are explained, In section-4 the implementation of lossless method of compression and decompression (i.e. Huffman Coding & Decoding) is done,section-5. The algorithm is developed and in section-6 the results were presented with explanation. Lastly, paper concludes with References.

2. NEED FOR COMPRESSION

The following example illustrates the need for compression of digital images [3].

a. To store a colour image of a moderate size, e.g. 512×512 pixels, one needs 0.75 MB of disk space.
b. A 35mm digital slide with a resolution of 12µm requires 18 MB.
c. One second of digital PAL (Phase Alternation Line) video requires 27 MB.

To store these images, and make them available over network (e.g. the internet), compression techniques are needed. Image compression addresses the problem of reducing the amount of data required to represent a digital image. The underlying basis of the reduction process is the removal of redundant data. According to mathematical point of view, this amounts to transforming a two-dimensional pixel array into a statistically uncorrelated data set. The transformation is applied prior to storage or transmission of the image. At receiver, the compressed image is decompressed to reconstruct the original image or an approximation to it.

The example below clearly shows the importance of compression. An image, 1024 pixel×1024 pixel×24 bit, without compression, would require 3 MB of storage and 7 minutes for transmission, utilizing a high speed, 64 Kbits/s, ISDN line. If the image is compressed at a 10:1 compression ratio, the storage requirement is reduced to 300 KB and the transmission time drop to less than 6 seconds.

2.1 Principle behind Compression

A common characteristic of most images is that the neighboring pixels are correlated and therefore contain redundant information [3]. The foremost task then is to find less correlated representation of the image. Two fundamental components of compression are redundancy and irrelevancy reduction.

a. Redundancies reduction aims at removing duplication from the signal source (image/video).
b. Irrelevancy reduction omits parts of the signal that will not be noticed by the signal receiver, namely the Human Visual System.

In an image, which consists of a sequence of images, there are three types of redundancies in order to compress file size. They are:

a. Coding redundancy: Fewer bits to represent frequently occurring symbols.
b. Interpixel redundancy: Neighbouring pixels have almost same value.
c. Psycho visual redundancy: Human visual system cannot simultaneously distinguish all colours.

3. VARIOUS TYPES OF REDUNDANCY

In digital image compression, three basic data redundancies can be identified and exploited:

a. Coding redundancy
b. Inter pixel redundancy
c. Psycho visual redundancy

Data compression is achieved when one or more of these redundancies are reduced or eliminated.

3.1 Coding Redundancy

A gray level image having n pixels is considered. The number of gray levels in the image is L (i.e. the gray levels range from 0 to *L*-1) and the number of pixels with gray level r_k is n_k. Then the probability of occurring gray level *is* r_k is $P_r(r_k)$. If the number of bits used to represent the gray level r_k is $l(r_k)$, then the average number of bits required to represent each pixel is.

$$L_{avg} = \sum_{K=0}^{L-1} l(r_k) P_r(r) \qquad (1.2)$$

Where,

$$P_r(r_k) = n_k / n \qquad (1.4)$$

$$k = 0, 1, 2, \ldots\ldots\ldots\ldots, L-1$$

Hence the number of bits required to represent the whole image is n x Lavg. Maximal compression ratio is achieved when Lavg is minimized (i.e. when $l(r_k)$, the length of gray level representation function, leading to minimal Lavg , is found). Coding the gray levels in such a way that the Lavg is not minimized results in an image containing coding redundancy.

Generally coding redundancy is presented when the codes (whose lengths are represented here by $l(r_k)$ function) assigned to a gray levels don't take full advantage of gray level's probability ($P_r(r_k)$ function). Therefore it almost always presents when an image's gray levels are represented with a straight or natural binary code [1]. A natural binary coding of their gray levels assigns the same number of bits to both the most and least probable values, thus failing to minimize equation 1.2 and resulting in coding redundancy**.**

Example of Coding Redundancy: An 8-level image has the gray level distribution shown in table I. If a natural 3-bit binary code (see code 1 and $l\ rk$ in table I) is used to represent 8 possible gray levels, Lavg is 3- bits, because $l\ rk$ = 3 bits for all *rk*. If code 2 in table I is used, however the average number of bits required to code the image is reduced to: Lavg = 2(0.19) + 2(0.25) + 2(0.21) + 3(0.16) + 4(0.08) + 5(0.06) + 6(0.03) + 6(0.02) = 2.7 bits. From equation of compression ratio(n2/n1) the resulting compression ratio C_R is 3/2.7 or 1.11.Thus approximately 10% of the data resulting from the use of code 1 is redundant. The exact level of redundancy can be determined from equation (1.0).

Table I: Example of Variable Length Coding

r_k	$P_r(r_k)$	code1	$l_1(r_k)$	code2	$l_2(r_k)$
$r_0 = 0$	0.19	000	3	11	2
$r_1 = 1/7$	0.25	001	3	01	2
$r_2 = 2/7$	0.21	010	3	10	2
$r_3 = 3/7$	0.16	011	3	001	3
$r_4 = 4/7$	0.08	100	3	0001	4
$r_5 = 5/7$	0.06	101	3	00001	5
$r_6 = 6/7$	0.03	110	3	000001	6
$r_7 = 1$	0.02	111	3	000000	6

$R_{D=}$ 1-1/1.1 = 0.099=9.9%

It is clear that 9.9% data in first data set is redundant which is to be removed to achieve compression.

3.1.1 Reduction of Coding Redundancy

To reduce this redundancy from an image we go for the Huffman technique were we are, assigning fewer bits to the more probable gray levels than to the less probable ones achieves data compression. This process commonly is referred to as variable length coding. There are several optimal and near optimal techniques for constructs such a code i.e. Huffman coding, Arithmetic coding etc.

3.2 Inter pixel Redundancy

Another important form of data redundancy is inter pixel redundancy, which is directly related to the inter pixel correlations within an image. Because the value of any given pixel can be reasonable predicted from the value of its neighbors, the information carried by individual pixels is relatively small. Much of the visual contribution of a single pixel to an image is redundant; it could have been guessed on the basis of its neighbor's values. A variety of names, including spatial redundancy, geometric redundancy, and interframe *Redundancies have* been coined to refer to these interpixel dependencies. In order to reduce the interpixel redundancies in an image, the 2-D pixel array normally used for human viewing and interpretation must be transformed into a more efficient but usually non-visual format. For example, the differences between adjacent pixels can be used to represent an image. Transformations of this type are referred to as *mappings*. They are called *reversible* if the original image elements can be reconstructed from the transformed data set [1, 2].

3.2.1 Reduction of Interpixel Redundancy

To reduce the interpixel redundancy we use various techniques such as:

1. Run length coding.
2. Delta compression.
3. Constant area coding.
4. Predictive coding.

3.3 Psycho visual Redundancy

Human perception of the information in an image normally does not involve quantitative analysis of every pixel or luminance value in the image. In general, an observer searches for distinguishing features such as edges or textural regions and mentally combines them into recognizable groupings. The brain then correlates these groupings with prior knowledge in order to complete the image interpretation process. Thus eye does not respond with equal sensitivity to all visual information. Certain information simply has less relative importance than other information in normal visual processing. This information is said to be psycho visually redundant. It can be eliminated without significantly impairing the quality of image perception. Psycho visual redundancy is fundamentally different from the coding Redundancy and interpixel redundancy .Unlike coding redundancy and interpixel redundancy, psychovisual redundancy is associated with real or quantifiable visual information. Its elimination is possible only because the information itself is not essential for normal visual processing. Since the elimination of psychovisual redundant data results in a loss of quantitative information. Thus it is an *irreversible process.*

3.3.1 Reduction of Psycho visual Redundancy

To reduce psycho visual redundancy we use Quantizer. Since the elimination of psychovisually redundant data results in a loss of quantitative information. It is commonly referred to as quantization. As it is an irreversible operation (visual information is lost) quantization results in lossy data compression.

Table II: Huffman Source Reductions

Original source		Source reduction			
S	P	1	2	3	4
a_2	0.4	0.4	0.4	0.4	0.6
a_6	0.3	0.3	0.3	0.3	0.4
a_1	0.1	0.1	0.2	0.3	
a_4	0.1	0.1	0.1		
a_3	0.06	0.1			
a_5	0.04				

S-source, P-probability

4. TYPES OF COMPRESSION

Compression can be divided into two categories, as Lossless and Lossy compression. In lossless compression, the reconstructed image after compression is numerically identical to the original image [3]. In lossy compression scheme, the reconstructed image contains degradation relative to the original. Lossy technique causes image quality degradation in each compression or decompression step. In general, lossy techniques provide for greater compression ratios than lossless techniques. The following are the some of the lossless and lossy data compression techniques:

4.1 Lossless coding techniques

a. Run length encoding
b. Huffman encoding
c. Arithmetic encoding
d. Entropy coding
e. Area coding

4.2 Lossy coding techniques

a. Predictive coding
b. Transform coding (FT/DCT/Wavelets)

5. IMPLEMENTATION OF LOSS LESS COMPRESSION AND DECOMPRESSION TECHNIQUES

5.1 Huffman coding

Huffman code procedure is based on the two observations [3].

a. More frequently occurred symbols will have shorter code words than symbol that occur less frequently.

b. 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 labelling of the code tree. This is illustrated with an example shown in table II:

Table III: Huffman Code Assignment Procedure

Original source reduction			Source reduction							
S	P	code	1		2		3		4	
a_2	0.4	1	0.4	1	0.4	1	0.4	1	0.6	0
a_6	0.3	00	0.3	00	0.3	00	0.3	00	0.4	1
a_1	0.1	011	0.1	011	0.2	010	0.3	01		
a_4	0.1	0100	0.1	0100	0.1	011				
a_3	0.06	01010	0.1	0101						
a_5	0.04	01011								

S-source, P-probability

At the far left of the table I the symbols are listed and corresponding symbol probabilities are arranged in decreasing order and now the least t probabilities are merged as here 0.06 and 0.04 are merged, this gives a compound symbol with probability 0.1, and the compound symbol probability is placed in source reduction column1 such that again the probabilities should be in decreasing order. so this process is continued until only two probabilities are left at the far right shown in the above table as 0.6 and 0.4. The second step in Huffman’s procedure is to code each reduced source, starting with the smallest source and working back to its original source [3]. The minimal length binary code for a two-symbol source, of course, is the symbols 0 and 1. As shown in table III these symbols are assigned to the two symbols on the right (the assignment is arbitrary; reversing the order of the 0 and would work just and well). As the reduced source symbol with probabilities 0.6 was generated by combining two symbols in the reduced source to its left, the 0 used to code it is now assigned to both of these symbols, and a 0 and 1 are arbitrary appended to each to distinguish them from each other. This operation is then repeated for each reduced source until the original course is reached. The final code appears at the far-left in table 1.8. The average length of the code is given by the average of the product of probability of the symbol and number of bits used to encode it. This is calculated below: Lavg = (0.4)(1) + (0.3)(2) + (0.1)(3) + (0.1)(4) + (0.06)(5) + (0.04)(5) = 2.2 bits/ symbol and the entropy of the source is 2.14 bits/symbol, the resulting Huffman code efficiency is 2.14/2.2 = 0.973.

Entropy, H=- $\sum_{k=1}^{L} P(a_j) \log P(a_j)$ (1)

Huffman’s procedure creates the optimal code for a set of symbols and probabilities subject to the constraint that the symbols be coded one at a time.

5.2 Huffman decoding

After the code has been created, coding and/or decoding is accomplished in a simple look-up table manner[9]. 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. For the binary code of table 3.2, a left-to-right scan of the encoded string 010100111100 reveals that the first valid code word is 01010, which is the code for symbol a3. The next valid code is 011, which corresponds to symbol a1. Valid code for the symbol a2 is 1,valid code for the symbols a6 is 00,valid code for the symbol a6 is Continuing in this manner reveals the completely decoded message a_5 a_2 a_6 a_4 a_3 a_1 , so in this manner the original image or data can be decompressed using Huffman decoding as explained above .

At first we have as much as the compressor does a probability distribution. The compressor made a code table. The decompressor doesn't 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.

5.3 Development of Huffman Coding and Decoding Algorithm

Step1- Read the image on to the workspace of the mat lab.

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 in the table II.

Step10-Output the character encode in the leaf and return to the root, and continue the step9 until all the codes of corresponding symbols are known.

6. RESULTS

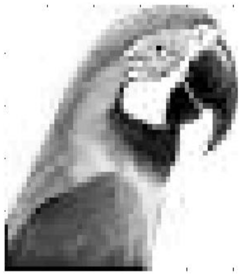
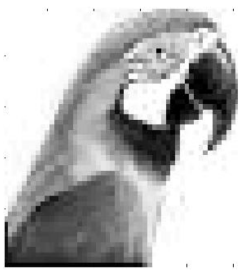

Fig.1 Input image Fig.2 Decompressed image

The input image shown in Fig.1 to which the above Huffman coding algorithm is applied for the generation of codes and then decompression algorithm(i.e. Huffman decoding) is applied to get the original image back from the generated codes, which is shown in the Fig.2.The number of saved bits is the difference between the no of bits required to represent the input image i.e. shown in the fig1 by considering each symbol can take a maximum code length of 8 bits and the no of bits taken by the Huffman code to represent the compressed image i.e. Saved bits = (8*(r*c)-(l1*l2))=3212, r and c represents size of the input matrix, l1 and l2 represents the size of Huffman code. The compression ratio is the ratio of number of bits required to represent the image using Huffman code to the no of bits used to represent the input image. i.e. Compression ratio = (l1*l2)/ (8*r*c) = 0.8456, The output image is the decompressed image i.e. from the Fig.2 it is clear that the decompressed image is approximately equal to the input image.

7. CONCLUSION

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. Experimental results show that up to a 0.8456 compression ratio for the above image is obtained .hence we conclude that Huffman coding is efficient technique for image compression and decompression to some extent. As the future work on compression of images for storing and transmitting images can be done by other lossless methods of image compression because as we have concluded above the result the decompressed image is almost same as that of the input image so that indicates that there is no loss of information during transmission. So other methods of image compression can be carried out as namely JPEG method, Entropy coding, etc.

REFERENCES

[1] Ternary Tree & FGK Huffman Coding Technique Dr. Pushpa R.Suri † and Madhu Goel Department of Computer Science & Applications, Kurukshetra University, Kurukshetra, India

[2] Massachusetts Institute of Technology Department of Electrical Engineering and Computer Science.

[3] *Compression Using Fractional Fourier Transform* A Thesis Submitted in the partial fulfillment of requirement for the award of the degree of *Master of Engineering In Electronics and Communication. By Parvinder Kaur.*

[4] RL-Huffman Encoding for Test Compression and Power Reduction in Scan Applications-MEHRDAD NOURANI and MOHAMMAD H. TEHRANIPOUR, The University of Texas at Dallas.

[5] A Novel VLSI Architecture of Hybrid Image Compression Model based on Reversible Blockade Transform C. Hemasundara Rao, *Student Member, IEEE,* M. Madhavi Latha, *Member, IEEE*

[6] D.A.Huffman, A Method for the construction of Minimum-redundancy Codes, Proc. IRE, vol.40, no.10, pp.1098-1101,1952.

[7] A.B.Watson,“Image Compression using the DCT” , Mathematica Journal, 1995,pp.81-88.

[8] DAVID A. HUFFMAN, Sept. 1991, profile Background story: Scientific American, pp. 54-58.

[9] Efficient Huffman decoding by MANOJ AGGRAWAL and AJAI NARAYAN.

[10] C. SARAVANAN Assistant Professor, Computer Centre, National Institute of Technology,Durgapur,WestBengal, India, Pin – 713209.R. PONALAGUSAMY Professor, Department of Mathematics, National Institute of Technology, Tiruchirappalli, Tamilnadu, India, Pin – 620015.

AUTHOR PROFILES:

Jagadish H. Pujar received the M. Tech degree in Power and Energy Systems from NITK Surthkal, Mangalore University in the year 1999. Currently, he is working as Asst. Professor in the Department of Electrical & Electronics Engineering, B. V. B. College of Engineering & Technology, Hubli, Karnataka, India and simultaneously pursuing his Ph.D in EEE from the prestigious Jawaharlal Nehru Technological University, Anatapur, India. He has published a number of research papers in various National and International Journals and Conferences. His areas of interests are Soft Computing Techniques based Digital Control Systems, Power Electronics, AI based Digital Image Processing, MATLAB, etc.

Lohit M. Kadlaskar is pursuing his B.E degree in the Department of Electrical & Electronics Engineering, B.V.B. College of of Engineering & Technology, Hubli, Karnataka, India. His areas of interests are Digital Image Processing, Fuzzy Logic and Artificial Neural Networks, MATLAB, etc.