Journal of Theoretical and Applied Information Technology
15th December 2012. Vol. 46 No.1
© 2005 - 2012 JATIT & LLS. All rights reserved·

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

AN OPTIMIZED LOSSLESS IMAGE COMPRESSION TECHNIQUE IN IMAGE PROCESSING

[1]MAHENDRA PRATAP PANIGRAHY, [2]NEERAJ KUMAR
Associate Professor, Department of ECE, Institute of Technology Roorkee, Roorkee
Associate Professor, Department of CSE, Institute of Technology Roorkee, Roorkee

E. mail: [1]contact_mahen@rediffmail.com, [2]kmneeraj@rediffmail.com

ABSTRACT
Data compression is a common requirement for most of the computerized applications. There are number of data compression algorithms, which are dedicated to compress different data formats. Even for a single data type there are number of different compression algorithms, which use different approaches. This paper examines lossless data compression algorithms and compares their performance. The article is concluded by stating which algorithm performs well for Image data. This paper addresses the area of image compression as it is applicable to various fields of image processing. It also includes various benefits of using image compression techniques. The decryption techniques are used for the best possible way of reconstruction of Image.

Keywords: *Data Compression, Encryption, Decryption, Lossless Compression, Lossy Compression*.

1. INTRODUCTION:

Data compression is widely used in data management to save storage space and network bandwidth. Compression is the art of representing the information in a compact form rather than its original or uncompressed form. This is very useful when processing, storing or transferring a huge file, which needs lots of resources. If the algorithms used to encrypt works properly, there should be a significant difference between the original file and the compressed file. When data compression is used in a data transmission application, speed is the primary goal. In a data storage application, the degree of compression is the primary concern. Compression can be classified as either lossy or lossless. Lossless compression techniques reconstruct the original data from the compressed file without any loss of data. These kinds of compression algorithms are called reversible compressions. Lossless compression techniques are used to compress medical images, text and images preserved for legal reasons, computer executable file and so on. But there are lots of algorithrims based on lossy compression techniques which reconstruct the original message with loss of some information. Such techniques could be used for multimedia images, video and audio to achieve more compact data compression. Various lossless data compression algorithms have been proposed and used. Some of the main techniques in use are the Huffman Coding, Run Length Encoding, Arithmetic Encoding and Dictionary Based Encoding. The novel idea is to leave data in compressed state as long as possible, and to only uncompress data when absolutely necessary. An image is essentially a 2-D signal processed by the human visual system. The signals representing images are usually in analog form. However, for processing, storage and transmission by computer applications, they are converted from analog to digital form. A digital image is basically a 2-Dimensional array of pixels. Images from the significant part of data, particularly in remote sensing, biomedical and video conferencing applications. The use of and dependence on information and computers continue to grow, so too does our need for efficient ways of storing and transmitting large amounts of data.

1.1. Image Basics:

An 8-bit digital image can be viewed as a matrix whose entries (known as pixels) range from 0 (black) to 255 (white).

Thus if we store an intensity value, say 121, to disk, we don’t store 121, we store its ASCII character y, y also has a binary representation: 011110012. Thus if an image has dimensions N × M, we need 8MN bits to store it on disk (modulo some header information). We will refer to the stored image as the bit stream and note that the bits per pixel (bpp) is 8. Rather than use the same number of bits to represent each character, why not use a short bit stream for characters that appear often in an image and a longer bit stream for characters that appear infrequently in the image?

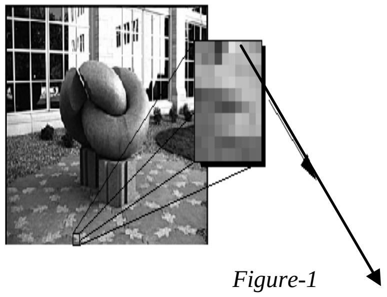

Figure-1

$$\begin{pmatrix}
129 & 128 & 121 & 51 & 127 & 224 & 201 & 179 & 159 & 140 \\
148 & 116 & 130 & 75 & 184 & 191 & 182 & 185 & 186 & 180 \\
175 & 169 & 166 & 195 & 195 & 192 & 168 & 173 & 166 & 158 \\
157 & 171 & 169 & 182 & 199 & 205 & 191 & 191 & 180 & 172 \\
73 & 89 & 96 & 100 & 122 & 143 & 166 & 190 & 188 & 180 \\
93 & 107 & 103 & 81 & 70 & 77 & 106 & 139 & 165 & 181 \\
106 & 105 & 112 & 132 & 144 & 147 & 189 & 183 & 158 & 184 \\
102 & 100 & 106 & 124 & 140 & 157 & 179 & 175 & 168 & 175 \\
91 & 105 & 112 & 93 & 86 & 85 & 100 & 104 & 110 & 106 \\
97 & 97 & 112 & 102 & 113 & 111 & 105 & 94 & 103 & 104
\end{pmatrix}$$

1.2. Image Compression:

Image compression addresses the problem of reducing the amount of data required to represent a digital image. It is a process intended to yield a compact representation of an image, thereby reducing the image storage/transmission requirements. Compression is achieved by the removal of one or more of the three basic data redundancies:

1. Coding Redundancy
2. Interpixel Redundancy
3. Psychovisual Redundancy

Coding redundancy is present when less than optimal code words are used. Interpixel redundancy results from correlations between the pixels of an image. Psychovisual redundancy is due to data that is ignored by the human visual system (i.e. visually non essential information). Image compression techniques reduce the number of bits required to represent an image by taking advantage of these redundancies. An inverse process called decompression (decoding) is applied to the compressed data to get the reconstructed image. The objective of compression is to reduce the number of bits as much as possible, while keeping the resolution and the visual quality of the reconstructed image as close to the original image as possible. Image compression systems are composed of two distinct structural blocks: an encoder and a decoder.

1.3. Types Of Compression:

The image compression techniques are broadly classified into two categories depending whether or not an exact replica of the original image could be reconstructed using the compressed image. These are:

i. Lossless Technique
ii. Lossy Technique

1.3.1. Lossless compression technique:

In lossless compression techniques, the original image can be perfectly recovered from the compressed (encoded) image. These are also called noiseless since they do not add noise to the signal (image).It is also known as entropy coding since it use statistics/ decomposition techniques to eliminate/minimize redundancy. Lossless compression is used only for a few applications with stringent requirements such as medical imaging. Following techniques are included in lossless compression:

a. Run Length Encoding
b. Huffman Encoding
c. LZW Coding
d. Area Coding

1. 3.2. Lossy compression technique:

Lossy methods deliver higher compression ratios, but sacrifice the ability reproduces the original, uncompressed pixel for pixel. JPEG is the best known lossy compression standard and widely used to compress still images stored on compact disc. It is considerably more complicated than RLE, but it produces correspondingly higher compression ratios – even for images containing little or no redundancy. Except where every piece of information of a scan is critical – for example, scientific data – a scan must only provide enough information to meet the needs of the reproduction process and the viewer.

a. Transformation Coding
b. Vector Quantization
c. Fractal Coding
d. Block Truncation Coding
e. Sub-band Coding

2. IMPLEMENTATION OF LOSSLESS COMPRESSION AND DECOMPRESSION TECHNIQUES:

2.1. Huffman's Encoding & Decoding:

This algorithm, developed by D.A. Huffman, is based on the fact that in an input stream certain tokens occur more often than others. Based on this knowledge, the algorithm builds up a weighted binary tree according to their rate of occurrence. Each element of this tree is assigned a new code word, whereat the length of the code word is determined by its position in the tree. Therefore, the token which is most frequent and becomes the root of the tree is assigned the shortest code. Each less common element is assigned a longer code word. The least frequent element is assigned a code word which may have become twice as long as the input token.

The compression ratio achieved by Huffman encoding uncorrelated data becomes something like 1:2. On slightly correlated data, as on images, the compression rate may become much higher, the absolute maximum being defined by the size of a single input token and the size of the shortest possible output token (max. compression = token size[bits]/2[bits]). While 3 standards palletized images with a limit of 256 colors may be compressed by 1:4 if they use only one color, more typical images give results in the range of 1:1.2 to 1:2.5. Entropy coding (Lempel/Ziv). The typical implementation of an entropy coder follows J. Ziv/A. Lempel's approach. Nowadays, there is a wide range of so called modified Lempel/Ziv codings. These algorithms all have a common way of working. The coder and the decoder both build up an equivalent dictionary of meta-symbols, each of which represents a whole sequence of input tokens. If a sequence is repeated after a symbol was found for it, then only the symbol becomes part of the coded data and the sequence of tokens referenced by the symbol becomes part of the decoded data later. As the dictionary is build up based on the data, it is not necessary to put it into the coded data, as it is with the tables in a Huffman coder. This method becomes very efficient even on virtually random data. The average compression on text and program data is about 1:2; the ratio on image data comes up to 1:8 on the average GIF image. Here again, a high level of input noise degrades the efficiency significantly. Entropy coders are a little tricky to implement, as there are usually a few tables, all growing while the algorithm runs.

2.2. Dct (Discrete Cosine Transform):

The DCT is a widely used transformation in transformation for data compression. It is an orthogonal transform, which has a fixed set of (image independent) basis functions, an efficient algorithm for computation, and good energy compaction and correlation reduction properties. Ahmed et al found that the Karhunen Loeve Transform (KLT) basis function of a first order Markov image closely resemble those of the DCT. They become identical as the correlation between the adjacent pixel approaches to one.

The DCT belongs to the family of discrete trigonometric transform. The 1D DCT of a 1× N vector x (n) is defined as

$$Y[k] = C[k]\sum_{n=0}^{N-1} x[n]\cos\left[\frac{(2n+1)k\pi}{2N}\right] \quad \textbf{(1)}$$

where k = 0, 1, 2... N −1 and

$$C[k] = \begin{bmatrix} \sqrt{\frac{1}{N}} & \text{for } k = 0 \\ \sqrt{\frac{1}{N}} & \text{for } k = 1,2,\dots,N-1 \end{bmatrix} \quad \textbf{(2)}$$

The original signal vector x (n) can be reconstructed back from the DCT coefficients Y[k] using the Inverse DCT (IDCT) operation

$$Y[j,k] = C[j]C[k]\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} x[m,n]\cos\left(\frac{(2m+1)j\pi}{2N}\right)\cos\left(\frac{(2n+1)k\pi}{2N}\right) \quad \textbf{(3)}$$

where n = 0, 1, 2,..., N −1

The DCT can be extended to the transformation of 2D signals or images. This can be achieved in two steps: by computing the 1D DCT of each of the individual rows of the two dimensional image and then computing the 1D DCT of each column of the image. If represents a 2D image of size x(n1 , n2) N × N, then the 2D DCT of an image is given by:

$$C[j] \ \ and \ \ C[k] = \begin{bmatrix} \sqrt{\frac{1}{N}} & \text{for j,k} = 0 \\ \sqrt{\frac{1}{N}} & \text{for j,k} = 1,2,\dots,\text{N-1} \end{bmatrix}$$

Similarly the 2D IDCT can be defined as

$$x[m,n] = \sum_{j=0}^{N-1}\sum_{k=0}^{N-1} C[j]C[k]Y[j,k]\cos\left(\frac{(2m+1)j\pi}{2N}\right)\cos\left(\frac{(2n+1)k\pi}{2N}\right) \quad \textbf{(4)}$$

The DCT is a real valued transform and is closely related to the DFT. In particular, a N ×N DCT of x (n1, n2) can be expressed in terms of DFT of its even-symmetric extension, which leads to a fast computational algorithm. Because of the even-symmetric extension process, no artificial discontinuities are introduced at the block boundaries. Additionally the computation of the DCT requires only real arithmetic. Because of the above properties the DCT is popular and widely used for data compression operation.

The DCT presented in equations 3 and 4 are orthonormal and perfectly reconstructing provided the coefficients are represented to an infinite precision. This means that when the coefficients are compressed it is possible to obtain a full range of compressions and image qualities. The coefficients of the DCT are always quantized for high compression, but DCT is very resistant to quantisation errors due to the statistics of the coefficients it produces. The coefficients of a DCT are usually linearly quantised by dividing by a predetermined quantisation step.

2.3. Measuring Compression Performances:

Basing on the nature of the application there are various criteria to measure the performance of a compression algorithm. When measuring the performance the main concern would be the space efficiency. The time efficiency is another factor. Since the compression behavior depends on the redundancy of symbols in the source file, it is difficult to measure performance of a compression algorithm in general. The performance depends on the type and the structure of the input source. Additionally the compression behavior depends on the category of the compression algorithm: lossy or lossless. If a lossy compression algorithm is used to compress a particular source file, the space efficiency and time efficiency would be higher than that of the lossless compression algorithm. Thus measuring a general performance is difficult and there should be different measurements to evaluate the performances of those compression families. Following are some measurements used to evaluate the performances of lossless algorithms.

2.3.1. Compression ratio:

Is the ratio between the size of the compressed file and the size of the source file.

$$\text{Compression Ratio} = \frac{\text{Size after Compression}}{\text{Size before Compression}}$$

2.3.2. Space saving percentage:

Calculates the shrinkage of the source file as a percentage.

Space Saving Percentage=

$$\frac{\text{Size before Compression} - \text{Size after Compression}}{\text{Size before Compression}} \%$$

3. PROPOSED HUFFMAN CODING ALGORITHM:

Step1 Read a JPEG image using image box control in MATLAB Language. This control will read an image and convert them in a text file.

Step2 Call a function that will Sort or prioritize characters based on frequency count of each characters in file.

Step3 Call a function that will create an initial heap, and then reheap that tree according to occurrence of each node in the tree,

Step4 Build Huffman code tree based on prioritized list. Chop-off those two elements in the sorted list as they are now part of one node and add the probabilities. The result is the probability for the new node.

Step 5 Perform insertion sort on the list with the new node.

Step 6 Repeat Steps 3, 4, 5 until you only have 1 node left.

Step 7 Perform a traversal of tree to generate code table.

Step 8 Once a Huffman tree is built, Canonical Huffman codes, which require less information to rebuild, may be generated by the following steps:

Step 1 Remember the lengths of the codes resulting from a Huffman tree generated per above.

Step 2 Sort the symbols to be encoded by the lengths of their codes (use symbol value to break ties).

Step 3 Initialize the current code to all zeros and assign code values to symbols from longest to shortest code as follows:

a. If the current code length is greater than the length of the code for the current symbol, right shift off the extra bits.
b. Assign the code to the current symbol.
c. Increment the code value.
d. Get the symbol with the next longest code.
e. Repeat from A until all symbols are assigned codes.

Step 9 Encoding Data- Once a Huffman code has been generated, data may be encoded simply by replacing each symbol with its code.

Step 10 The original image is reconstructed i.e. decompression is done by using Huffman Decoding.

Step 11 Generate a tree equivalent to the encoding tree. If you know the Huffman code for some encoded data, decoding may be accomplished by reading the encoded data one bit at a time. Once the bits read match a code for symbol, write out the symbol and start collecting bits again.

Step 12 Read input character wise and left to the tree until last element is reached in the tree.

Step 13 Output the character encodes in the leaf and returns to the root, and continues the step 12 until all the codes of corresponding symbols is known.

4. RESULT & INTERPRETATION:

The results for the various sets of pictures for Huffman (Encoding & Decoding) are tabulated in the 4.1 Table 1. The compressed and decompressed images are shown in the Figures 2.a, 2.b, and 2.c.

4.1 Table 1: Huffman Encoding Results:

File	Original Size	Compressed Size	Compression Ratio	Space Saving Percentage
1.	102KB	71.4 KB	0.7	30.6 KB
2.	25.6KB	15.3 KB	0.6	10.3 KB
3.	8.20KB	5.495 KB	0.67	2.705 KB
4.	7.25 KB	4.9575 KB	0.68	2.2925 KB
5.	13.6 KB	9.384 KB	0.69	4.216 KB
6.	10.7 KB	7.276 KB	0.68	3.424 KB
7.	36.1 KB	23.826 KB	0.66	12.274 KB
8.	25.3 KB	17.457 KB	0.69	7.843 KB
9.	34.3 KB	22.981 KB	0.67	11.319 KB
10.	56.2 KB	37.654 KB	0.67	18.546 KB

4.2: Huffman Original, Compressed & Decompressed Image Results:

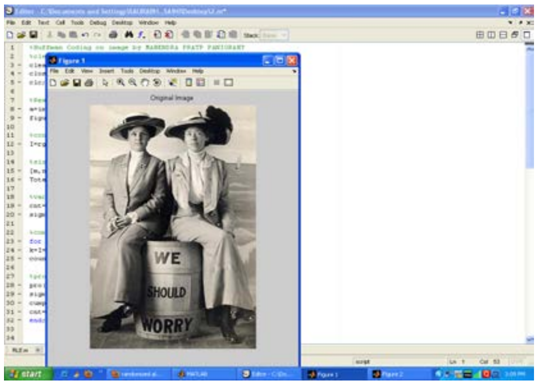

Figure- 2.A (Original Image)

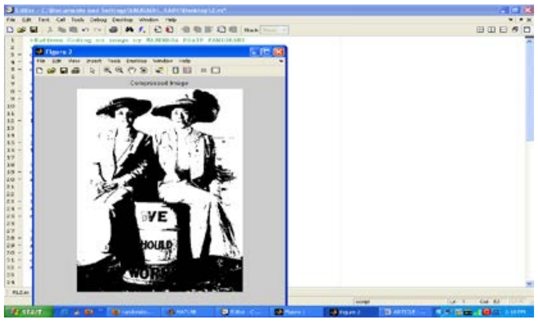

Figure- 2.B (Compressed Image)

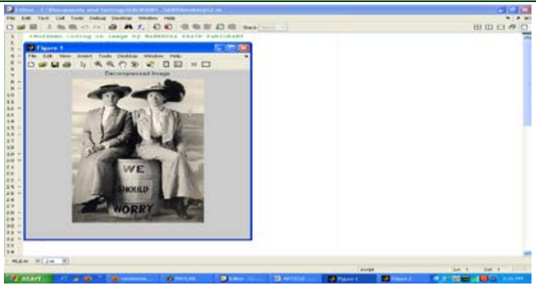

Figure- 2.C (Decompressed Image)

5. CONCLUSION

In this research work, a new methodology for performing Image Compression in Image Processing domain is presented. We have introduced a technique for extracting the compressed Image by using Huffman Encoding & Decoding Technique along with DCT concepts. The results of compression of ten images as listed in Table-1 has the average compression ratio of 0.67 i.e. 67 % and the decompression image has got a approximate Image as the original image. But the Decompressed Image can be more than what we achieved. Future research will include finding a method for extracting the best way of finding best decompressed image from compressed image, so this will be therefore the best compression & decompression of images completely in face recognition. Also a better algorithm which will further enhance the recognition rate of images with varying expression is another interested area of further research.

6. 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- , 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.
[11]. Stephen A. Martucci and Iraj Sodagar, 1996. Zerotree Entropy Coding of Wavelet Coefficients For Very Low Bit Rate Vide, IEEE, PP 533-536.
[12]. Woods, R. C. 2008. Digital Image processing. New Delhi: Pearson Pentice Hall, Third Edition, Low price edition, Pages 1-904.
[13]. Sonja Grgic, M. M. 2001. Comparison of JPEG Image Coders.; Proceedings of the 3rd International symposium on Video Processing and Multimedia Communications, June, (pp. 79-85). Zadar, Croatia.
[14]. Austin, D. 2011. Image Compression: Seeing wht's not there, Feature Column, American Mathematical Society. May.
[15]. Wang, E.-h. Y. 2009. Joint Optimisation Run-Length Coding, Huffman Coding, and Quantization Table with Complete Baseline JPEG Decoder Compatibility. IEEE Transactions on Image Processing, Volume No. 18, No.1, January.