Volume 3, Issue 5, May 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Performance Analysis of Huffman Coding Algorithm

Aarti
Departmentof CSE,ACET
India.

***Abstract*— Huffman coding is such a widespread method for creating prefix-free codes. Huffman code is used as a synonym for prefix-free code even when such a code is not produced by Huffman's algorithm. Huffman coding is an efficient source coding algorithm for source symbols that are not equally probable. It yields the smallest possible number of code symbols per source symbol [7]. In this paper, the Lossless method of compression and decompression is proposed using a simple coding technique called Huffman coding which is used on text string as well as on different image file formats for performance analysis. The results reveals that the original image used for coding is almost close to the decoded output image.**

***Keywords*— Huffman coding, bpp, CR, JPEG, PSNR.**

I. INTRODUCTION

Huffman coding is one of the most popular technique for removing coding redundancy. It has been used in various compression applications, including image compression. It is a simple, yet elegant, compression technique that can supplement other compression algorithms. It is also used in CCITT Group 3 compression. Group 3 is a compression algorithm developed by the International Telegraph and Telephone Consultative Committee in 1985 for encoding and compressing 1-bit (monochrome) image data. Group 3 and 4 compression are most commonly used in the TIFF file format. It utilizes the statistical property of alphabets in the source stream, and then produces respective codes for these alphabets. These codes are of variable code length using integral number of bits. The codes for alphabets having higher probability of occurrence are shorter than those codes for alphabets having lower probability. So , It is based on the frequency of occurrence of a data item (pixels or small blocks of pixels in images). It uses a lower number of bits to encode more frequent data. The Codes are stored in a Code Book. Code book is constructed for each image or a set of images. Huffman coding is the optimal lossless scheme for compressing a bit stream. It works by first calculating probabilities. Define permutations $\{0, 1\}^n$ by assigning symbols, say A;B;C;D. The bit stream might look like AADAC for example. Now the symbols are assigned new codes, the higher the probability the smaller the number of bits in the code [3]. The codes are the output of the Huffman coder in the form of a bit stream. Now we need to know where one code stops and a new one begins. This is solved by enforcing the unique prefix condition: no code is the prefix of any other code. The first few codes are 01; 11; 001; 101; 0000; 1000; 1001. In Huffman coding, you assign shorter codes to symbols that occur more frequently and longer codes to those that occur less frequently [1].

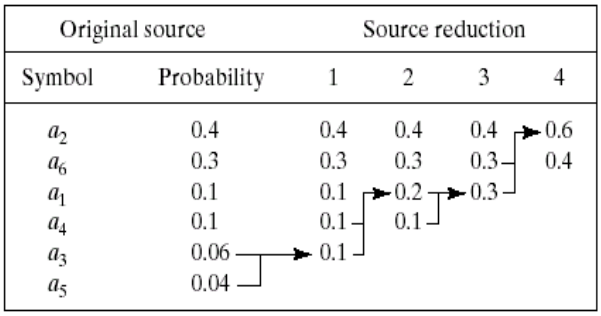

Fig. 1 Huffman Source Reductions

1.1 Huffman Code Construction

Huffman Coding Algorithm is a bottom-up approach.

ALGORITHM

The steps of Huffman coding algorithm are given below[9]:

1. Create a series of source reduction: combine the two lowest probability symbol into a single symbol; repeated until a reduced source with two symbols is reached.
2. Code each reduced symbol: start with the smallest source and working back to the original source;
   (a) Creates the optimal code for a set of symbols; the symbol is coded one at a time;
   (b) The code itself (or block code) –each code is mapped

into a fixed sequence of code symbols

(c) Create the optimal codes for a set of symbols and probabilities

(d) Coding and decoding is accomplished in a simple lookup table

(e) Block code (because source symbol is mapped into a sequence of codes)

(f) Instantaneous, unique decodable

Huffman codes are optimal prefix codes generated from a set of probabilities by a particular algorithm i.e. the Huffman Coding Algorithm.

1.2 Building a Huffman Tree

The compression process is based on building a binary tree that holds all alphabets in the source at its leaf nodes, and with their corresponding alphabets' probabilities at the side. The code word for each symbol is obtained traversing the binary tree from its root to the leaf corresponding to the symbol. Symbols with the highest frequencies end up at the top of the tree, and result in the shortest codes[4].

The tree is built by going through the following steps:

1. Each of the alphabets is laid out as leaf node which is going to be connected. The alphabets are ranked according to their weights, which represents the probabilities of their occurrences in the source.

2. Two nodes with the lowest weights are combined to form a new node, which is a parent node of these two nodes. This parent node is then considered as a representative of the two nodes with a weight equal to the sum of the weights of two nodes. Moreover, one of the children is assigned a "0" and the other is assigned a "1".

3. Nodes are then successively combined as above until a binary tree containing all of nodes is created.

4. The code representing a given alphabet can be determined by going from the root of the tree to the leaf node representing the alphabet. The accumulation of symbol "0" and "1" is the code of that alphabet.

By using this procedure, the alphabets are naturally assigned codes that reflect probability distribution. Highly probable alphabets will be given short codes, and improbable alphabets will have long codes. Therefore, the average code length will be reduced. If the statistic of alphabets is very biased to some particular alphabets, the reduction will be very significant [3].

For example, imagine we have a text file that uses only five characters (A, B, C, D, E). Before we can assign bit patterns to each character, we assign each character a weight based on its frequency of use.

Table 1 Frequency of Characters

Character	A	B	C	D	E
Frequency	17	12	12	27	32

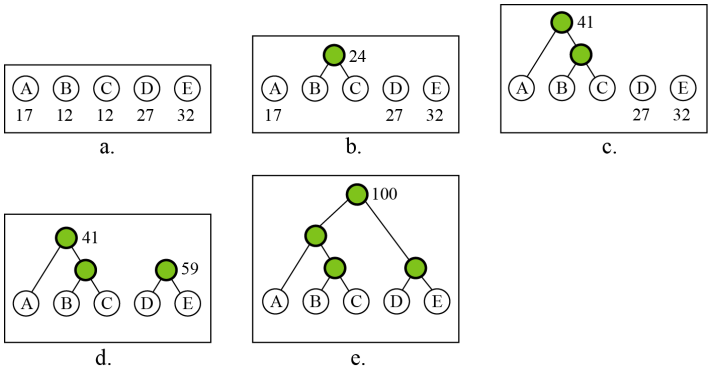

Fig. 1.1 Stages of Building a Huffman Tree

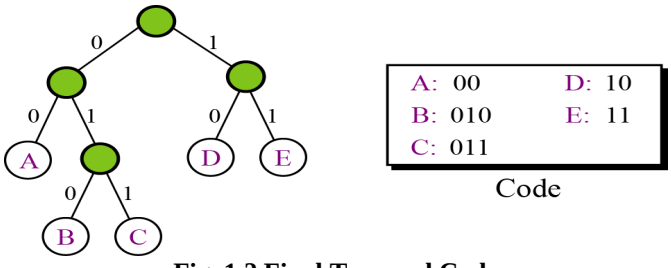

Fig. 1.2 Final Tree and Code

The algorithm is optimal in the sense that the average number of bits required to represent the source symbols is a minimum provided the prefix condition is met [8].

1.3 Properties of Huffman Coding

1. Unique Prefix Property : No Huffman code is a prefix of any other Huffman code -precludes any ambiguity in decoding. Eg. If the letter **e** is represented as 01, we could not encode another letter as 010 . So tried to represent **b** as 010. As the decoder scanned the input bit stream 0 10 .... as soon as it saw 01, it would output an **e** and start the next code with 0 In order to avoid this problem, all variable length encoding schemes must have the prefix property.
2. Optimality: minimum redundancy code -proved optimal for a given data model (i.e., a given, accurate, probability distribution):

a) The two least frequent symbols will have the same length for their Huffman codes, differing only at the last bit.
b) Symbols that occur more frequently will have shorter Huffman codes than symbols that occur less frequently.
c) The average code length for an information source S is strictly less than η+ 1[8].

II. RELATED WORK

Compressing an image is significantly different than compressing raw binary data. Compression programs can be used to compress images, but the result is less than optimal. This is because images have certain statistical properties which can be exploited by encoders specifically designed for them. Also, some of the finer details in the image can be sacrificed for the sake of saving a little more bandwidth or storage space. Image compression algorithms are designed to minimize image file size in order to speed up image data transmission. Any compression algorithm can be viewed as a function that maps sequences of units (normally octets) into other sequences of the same units. Compression is successful if the resulting sequence is shorter than the original sequence plus the map needed to decompress it. Decompression restores the original image without loss of fidelity. Continuous-tone greyscale images generally use 8 bits per pixel (bpp) and color images use 24 bits per pixel (8 each for red, green and blue). Medical and scientific images typically use more bits per pixel, sometimes up to 16 bpp for gray scale. Taken together, the values of all the pixels in an image constitute the raw data representation of the image. The amount of storage required by this raw data can be calculated as the product of the number of pixels and the bits used per pixel. As an example, consider a continuous-tone color image of dimensions 1024x768 using 24 bpp. The storage requirements for the raw data of such an image would be: 1024 x 768 x 24 = 18874368 bits = 2359296 Bytes = 2:25 MB. Texts are always compressed with lossless compression techniques, because a loss in a text will change its originality. Repeated data is important in compression. If a text has many repeated data, it can be compressed to a high ratio. It can be thought that if a natural language has many words that have similar character combinations in them, this language can be more compressible. The primary objective of an image compression is to minimize the average number of bits and to achieve a reasonable compression ratio as well as better quality of reproduction of image with a low power consumption.

For a compressed image, the bit rate achieved by the compressor, measured in bits/pixel, is defined as the number of bits used in the compressed representation of the image divided by the number of pixels in the image.

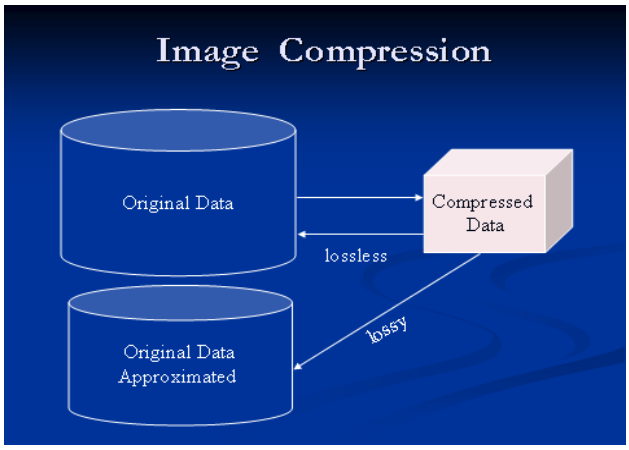

Fig. 2

The principle of image compression algorithms are (i) reducing the redundancy in the image data and (or) (ii) producing a reconstructed image from the original image with the introduction of error that is insignificant to the intended applications. The aim here is to obtain an acceptable representation of digital image while preserving the essential information contained in that particular data set [2]. There are two types of Image Compression Algorithms: Lossless Compression Algorithms and Lossy compression algorithms.

1) Lossless image compression is the process of compressing and subsequently decompressing images without the loss of data. lossless properties generate an exact duplicate of the original image upon decompression. In lossless compression schemes, the reconstructed image, after compression, is numerically identical to the original image. The biggest drawback for lossless image compression is that images can only be reduced to about one-third of their original image size [5]. The goal of lossless image compression is to represent an image signal with the smallest possible number of bits without loss of *any* information, thereby speeding up transmission and minimizing storage requirements [6].

Example: Entropy Encoding Schemes (Shannon-Fano, Huffman coding), arithmetic coding, LZW algorithm used in GIF image file format.

Lossy image compression algorithms, relinquish some accuracy in exchange for increased compression. An image reconstructed following lossy compression contains degradation relative to the original. Often this is because the compression scheme completely discards redundant information. However, lossy schemes are capable of achieving much higher compression. The goal of lossy compression is to achieve the best possible fidelity given an available communication or storage bit rate capacity or to minimize the number of bits representing the image signal subject to some allowable loss of information.They are capable of reducing images to one-tenth of their actual size with[10] little or no humanly perceptual loss in image detail. They do not guarantee that the original and recovered data are identical, but it often provides better performance (in terms of compression ratio) than lossless compression techniques. Lossy compression can be applied to voice, image, and video media applications.

Image file formats

Images are an important part of today's digital world [6]. Some of these compression methods are designed for specific kinds of images, so they will not be so good for other kinds of images. Some algorithms even let you change parameters they use to adjust the compression better to the image. File formats are essential when it comes to compatibility and storing images .Different image formats use different types of data compression.An image file format is a standard way to organize and store image data. It defines how the data is arranged and the type of compression (if any) to be used.

There are several dozens of different image file formats. Some of them use compression techniques while others do not. Some formats support only lossy while others also allow lossless compression. Some compression techniques are more useful for images of natural scenes rather than computer generated or artificial images[11].

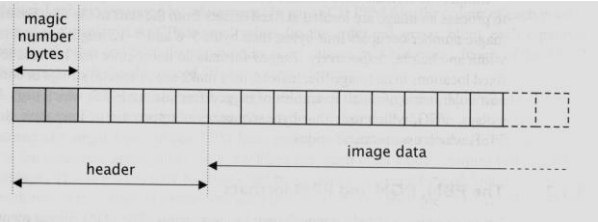

Fig. 3

CCITT (Huffman) Encoding

Many facsimile and document imaging file formats support a form of lossless data compression often described as CCITT encoding. The CCITT (International Telegraph and Telephone Consultative Committee) is a standards organization that has developed a series of communications protocols for the facsimile transmission of black-and-white images over telephone lines and data networks. These protocols are known officially as the CCITT T.4 and T.6 standards but are more commonly referred to as CCITT Group 3 and Group 4 compression, respectively. Sometimes CCITT encoding is referred to, not entirely accurately, as Huffman encoding. Huffman encoding is a simple compression algorithm introduced by David Huffman in 1952. CCITT 1-dimensional encoding, is a specific type of Huffman encoding. The other types of CCITT encodings are not[12].

Table 2 Image Compression standards and formats

ISO/IEC/ITU-T	JPEG · JPEG 2000 · lossless JPEG · JBIG · JBIG2 · PNG · WBMP
Others	BMP · GIF · ICER · ILBM · PCX · PGF · TGA · TIFF · JPEG XR / HD Photo

Performance Measures

Various metrics are used to evaluate the performance of Image compression algorithms.

The effectiveness of lossless compression schemes can be described using a relative measure, "compression ratio" or by describing an absolute measure, the "bit rate" of an image. The true effectiveness of the compression scheme may be better indicated by comparing the encoded bit rate with a measure of how much information is "really" encoded in the image. One such measure is the "entropy" of the image, a term from information theory [14] which is essentially the average amount of information per pixel value in the image. The "zero order" entropy does not take into account any information contained in surrounding pixels[13]. The performance of an image compression technique must be evaluated by considering different aspects.

- Compression efficiency (Compression Ratio/Factor, bit per pixel bpp or bit rate)
- Computational cost
- Fidelity i.e, Objective and Subjective
- Image quality (Distortion Measure)

III Problem Statement

The problem we intend to solve is based on commonly used lossless image compression algorithm which are presented and then compared in terms of their performance using Metrics. One of the problem faced while evaluating the performance of image compression algorithms is that they rely on different image file formats and use different performance metrics.

IV Research Methodology

In order to test the performance of lossless compression algorithm i.e. Huffman coding Algorithm is used to test the text string and images of different file formats. Performance is evaluated by computing the performance metrics like Compression efficiency i.e. Compression Ratio, bit per pixel *bpp* or bit rate). The mat lab code is presented for encoding and decoding of images and text string using Huffman coding, which is applied on colored and gray scale images.

The Images of various file formats have been taken as input and further pre-processed as required for analysis. The fundamental difficulty in testing Huffman coding algorithm is, how to decide which test image to use for the evaluations. Normally, a series of pictures, which are average in terms of how difficult they are for system being evaluated, has been selected. Huffman Coding Approach is implemented and executed to compress and decompress the given image and text string using MATLAB platform.

For this approach, file sizes, compression and decompression times, entropy and compression ratio & code efficiency are calculated.

V CONCLUSIONS

The result shows that the higher Code redundancy helps to achieve more compression. The above presented Huffman coding and decoding is used for scan testing to reduce test data volume, test data compression and decompression time. Hence we conclude that Huffman coding is efficient technique for image compression and decompression to some extent. The results also reveal that the original image used for coding is almost close to the decoded output image. This study presented an analysis of Huffman compression using metrics. Image compression using the Huffman Coding depends on the no. of pixels in an image , size of an image and compression ratio. Huffman is a good coding technique for compressing the data and general types of images.

VI FUTURE SCOPE

As the future work on compression of images for storing and transmitting can be done by other lossless methods of image compression because as it is concluded above, that the result of the decompressed image is almost same as that of the input image so it indicates that there is no loss of information during transmission. So other methods of image compression, any of the type i.e lossless or lossy can be carried out as namely JPEG method, LZW coding, etc. Use of different metrics can also take place to evaluate the performance of compression algorithms.

REFERENCES

[1] http://www.prepressure.com/library/compressionalgorithms/huffman
[2] David Salomon, Data Compression, The Complete Reference, 2nd Edition Springer-Verlag 1998.
[3] http://www.cprogramming.com/tutorial/computersciencetheory/huffman.h tml
[4] http://www.webopedia.com/TERM/H/Huffman_compression.html
[5] http://gradworks.umi.com/14/39/1439199.html
[6] www.hpl.hp.com/research/papers/seroussiIEEE.pdf
[7] http://ee.lamar.edu/gleb/dip/index.html
[8] http://www.prepressure.com/library/compression_algorithms/huffman
[9] http://www.webopedia.com/TERM/H/Huffman_compression.html
[10] Introduction to Data Compression, K. Sayood, Morgan Kauffman, Second Edition, 2000. (Primary)
[11] http://www.prepressure.com/library/file-formats
[12] www. FileFormt.info
[13] D. Clunie, Lossless Compression of Grayscale Medical Images, "Effectiveness of Traditional and State of the Art Approaches," in Proc. SPIE (Medical Imaging), vol.3980, Feb. 2000.
[14] Shannon CE, Weaver W, The mathematical theory of communication. University of Illinois Press, Urbana IL, 1949.

