ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 8, August 2015

Study on Image Compression Algorithms using Ideal Images

Dr.G.Boopathi[1], Dr. Narayanan Sreekumar[2]

Head, Department of Computer Applications, SNR Sons College, Coimbatore, Tamil Nadu, India[1]

Fellow, Faculty of Computing, Botho University, Gaborone, Botswana[2]

Abstract: Compression basically employs redundancy in the data. Compression is the process of reducing the size of a file by encoding its data information more efficiently. By doing this, the result is a reduction in the number of bits and bytes used to store the information. In effect, a smaller file size is generated in order to achieve a faster transmission of electronic files and a smaller space required for its downloading. Compression is done by using compression algorithms that rearrange and reorganize data information so that it can be stored more economically. This is done by using a compression/decompression program that alters the structure of the data temporarily for transporting, reformatting, archiving, saving, etc. Lossless Compression is a type of compression that can reduce files without a loss of information in the process. The original file can be recreated exactly when uncompressed. Huffman compression is a lossless compression algorithm that is ideal for compressing text or program files. This probably explains why it is used a lot in compression programs like ZIP or ARJ. In this article we represent principles of Huffman code which is used for compression. Huffman coding is based on the frequency of occurrence of a data item. The principle is to use a lower number of bits to encode the data that occurs more frequently. Codes are stored in a Code Book which may be constructed for each image or a set of images. In all cases the code book plus encoded data must be transmitted to enable decoding.

Keywords: JPEG, image compression, JPEG compression, Huffman code, reconstruction.

I. INTRODUCTION

Compressing an image is significantly different than compressing raw binary data. Of course, general purpose compression programs can be used to compress images, but the result is less than optimal. Image compression is minimizing the size in bytes of a graphics file without degrading the quality of the image to an unacceptable level. The reduction in file size allows more images to be stored in a given amount of disk or memory space [1,4]. It also reduces the time required for images to be sent over the Internet or downloaded from Web pages. Lossless compression involves with compressing data which, when decompressed, will be an exact replica of the original data.

This is the case when binary data such as executables, documents etc. are compressed[1].. They need to be exactly reproduced when decompressed. On the other hand, images need not be reproduced 'exactly'. An approximation of the original image is enough for most purposes, as long as the error between the original and the compressed image is tolerable [2]. Huffman coding A data compression technique which varies the length of the encoded symbol in proportion to its information content, that is the more often a symbol or token is used, the shorter the binary string used to represent it in the compressed stream[2].

Huffman codes can be properly decoded because they obey the prefix property, which means that no code can be a prefix of another code, and so the complete set of codes can be represented as a binary tree, known as a Huffman tree.

II. HISTORY OF HUFFMAN CODE

David Huffman in 1954 designed a lossless coding procedure for a frame of values that has the smallest possible average code length. It assigns a single bit to the most probable value and successively more bits to less probable values. Shannon-Fano Coding already existed, but did not produce optimal outcode length Robert Fano taught a class at MIT in Information Theory, giving the students a choice between a term paper and a final[5]. Huffman hit upon the idea of using a frequency-sorted binary tree and quickly proved this method the most efficient [3].Huffman built the tree from the bottom up instead of from the top down. One of the most basic lossless syntactic techniques is Huffman coding.

III. ENCODING AND DECODING MECHANISM

Encoding

Encoding is the process of transforming information from one format into another. The opposite operation is called decoding[6].

Given a code and a message it is easy to encode the message. Just replace the characters by the codewords.

Example: * = {a, b, c, d}

If the code is

C1 {a = 00, b = 01, c = 10, d = 11}

Then code **dad** is encoded into 110011

If the code is

C2 = { a = 0, b = 110 , c = 10, d = 111}

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 8, August 2015

Then **dad** is encoded into 1110111

Decoding
A1 = {a = 00, b = 01, c = 10, d = 11}
A2 = {a = 0, b = 110, c = 10, d = 111}
A3 = {a = 1, b = 110, c = 10, d = 111}

Given an encoded message, decoding is the process of turning it back into the original message. A message is uniquely decodable if it can only be decoded in one way[6].

For example

Relative to
A1 110011is uniquely decodable to **dad**.

Relative to
A2 1110111is uniquely decodable to **dad**.

IV. FREQUENCY VS HUFFMAN CODE

In ASCII codes, every character takes 8 bits.

To compress data, the most common approach is to reduce the number of bits that represent the most used characters[8].

Frequency Table		Huffman Code	
Character	**Count**	**Character**	**Code**
A	2	A	010
E	2	E	1111
I	3	I	110
S	6	S	10
T	1	T	0110
U	1	U	01111
Y	2	Y	1110
Space	4	Space	00

Table:1 Frequency Table vs Huffman Code

V. STEPS FOR HUFFMAN CODING ALGORITHM[13]

Step 1: Pick two letters x and y from alphabet A with the smallest frequencies and create a subtree that has these two characters as leaves. (greedy idea) Label the root of this subtree as z.

Step 2: Set frequency f(z) = f(x) + f(y)
Remove x , y and add z creating new alphabet
A' = A U {z} – {x,y}.
Note that | A' | = | A | - 1
Repeat this procedure, called merge, with new alphabet A' until and alphabet with only on symbol is left.
The resulting tree is the Huffman code.

Let A = { a/15, b/10, c/5, d/15, e/35} be the alphabet and its frequency distribution. In the first step Huffman coding merges c and d.

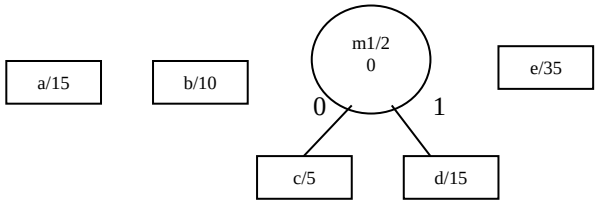

Fig:1 Huffman coding merges c and d

Alphabet is now A1 = { a/15, b/10, m1/20, e/35}

Alphabet is now A1 = { a/15, b/10, m1/20, e/35} be the alphabet and its frequency distribution.

Algorithm merges a and b.(could also have merged m1 and b)

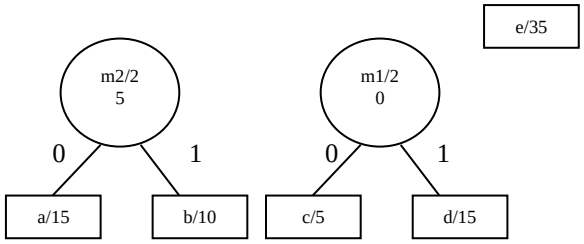

Fig:2 Algorithm merges a and b

New alphabet is A2 = { m2/25, m1/20, e/35}
Alphabet is A2 = { m2/25, m1/20, e/35}

Algorithm merges m1 and m2

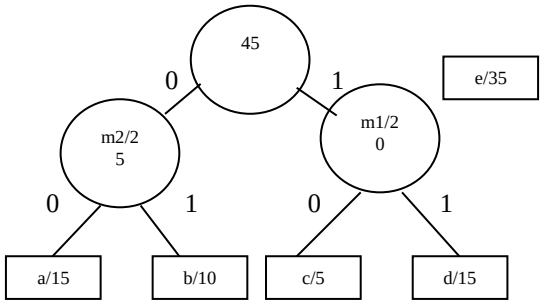

Fig:3 Algorithm merges m1 and m2

Current alphabet is A3 = {m3/45, e/35}
Algorithm merges e and m3 and finishes.

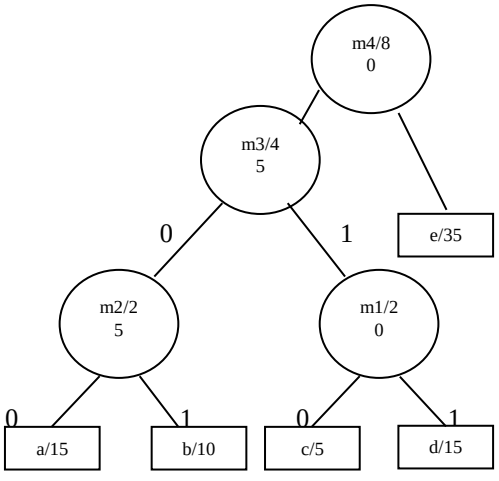

Fig:4 Algorithm merges e and m3

Huffman code is obtained from the Huffman tree.

Huffman code is
a = 001, b = 001, c = 010, d = 011, e = 1.

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 8, August 2015

This is the optimum (minimum-cost) prefix code for this distribution.

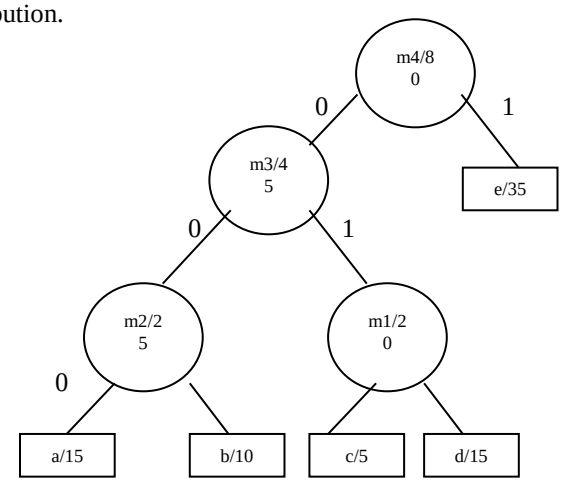

Fig:5 Huffman tree

Binary Image
Entropy (bits/pixel): 0.362
Huffman Encoded (bits/pixel): 1.000

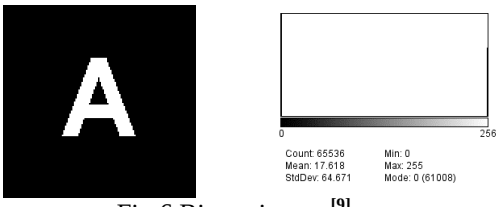

Fig:6 Binary image [9]

Image with 7 colors (due to JPEG compression)
Entropy (bits/pixel): 1.140
Huffman Encoded (bits/pixel): 1.530

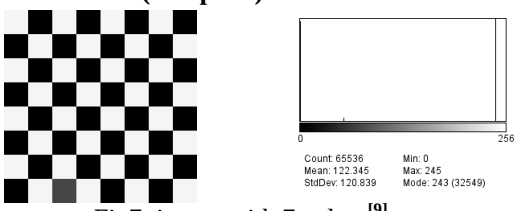

Fig7: image with 7 colors[9]

Image with 106 colors
Entropy (bits/pixel): 6.006
Huffman Encoded (bits/pixel): 6.032

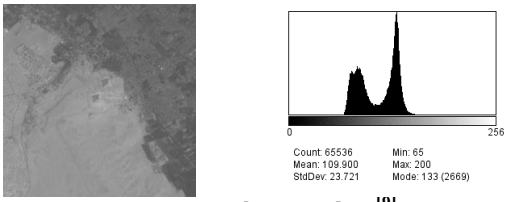

Fig:8 image with 106 colors[9]

Image with 249 colors
Entropy (bits/pixel): 7.384
Huffman Encoded (bits/pixel): 7.409

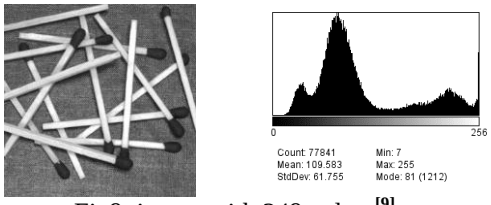

Fig9: image with 249 colors[9]

Image with 235 colors
Entropy (bits/pixel): 7.595
Huffman Encoded (bits/pixel): 7.624

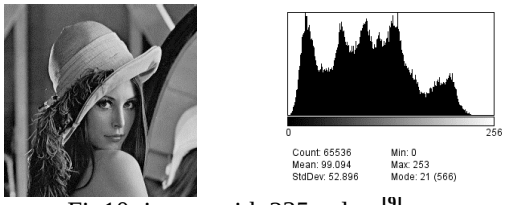

Fig10: image with 235 colors[9]

Image with 255 colors
Entropy (bits/pixel): 7.875
Huffman Encoded (bits/pixel): 7.916

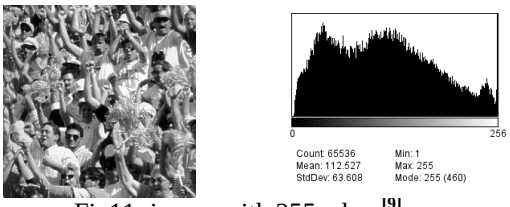

Fig11: image with 255colors[9]

Image with 256 colors
Entropy (bits/pixel): 8.000
Huffman Encoded (bits/pixel): 8.000

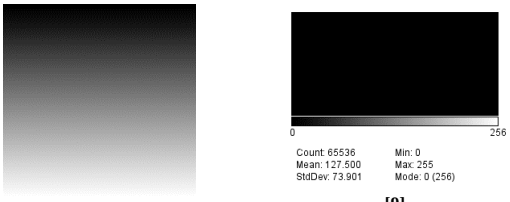

Fig12: image with 256 colors[9]

Huffman coding algorithm

Given an alphabet A with frequency distribution { f(a): a Є A }. The binary Huffman tree is constructed using a priority queue, Q, of nodes, with labels (frequencies) as keys[11,12].

```
Huffman (A)
{       n = |A|;
        Q = A;
        For I = 1 to n – 1
        {       z = new node;
                Left[z] = Extract-Min(Q);
        Right[z] = Extract-Min(Q);
        F[z] = f[left[z]]+ f[right[z]];
        Insert(Q,z);
        }
        Return Extract-Min(Q       ) root of the tree
}
```

Running time is O(n log n), as each priority queue operation takes time O(log n).

DOI 10.17148/IJARCCE.2015.48109

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 8, August 2015

VI. WORKING PRINCIPLE OF HUFFMAN COMPRESSION[10]

Huffman compression belongs into a family of algorithms with a variable codeword length. That means that individual symbols (characters in a text file for instance) are replaced by bit sequences that have a distinct length. So symbols that occur a lot in a file are given a short sequence while other that are used seldom get a longer bit sequence [10]. A practical example will show you the principle: Suppose you want to compress the following piece of data: ACDABASince these are 6 characters, this text is 6 bytes or 48 bits long. With Huffman encoding, the file is searched for the most frequently appearing symbols (in this case the character 'A' occurs 3 times) and then a tree is build that replaces the symbols by shorter bit sequences. In this particular case, the algorithm would use the following substitution table: A=0, B=10, C=110, D=111. If these code words are used to compress the file, the compressed data look like this: 01101110100
This mean that 11 bits are used instead of 48, a compression ratio of 4 to 1 for this particular file[10].

Huffman encoding can be further optimized in two different ways:

- Adaptive Huffman code dynamically changes the code words according to the change of probabilities of the symbols.
- Extended Huffman compression can encode groups of symbols rather than single symbols.

VII. ADVANTAGES AND DISADVANTAGES

This compression algorithm is mainly efficient in compressing text or program files. Images like they are often used in prepress are better handled by other compression algorithms[5].

VIII. USE OF HUFFMAN COMPRESSION

Huffman compression is mainly used in compression programs like pkZIP, lha, gz, zoo and arj. It is also used within JPEG and MPEG compression [7].

- Huffman is still used as back-end compression for popular compressed media files, such as JPEG and MP3
- Combined with run-length encoding, Huffman is used to send bitmap images across fax machines[12]
- Often combined with other coding schemes because of enhanced compression ratio and entropy

IX. CONCLUSION

The discussion of JPEG has centered on gray-scale images. Color images may assign a red, green and blue triple (rgb) to each pixel, although other choices are possible. Color specified in terms of brightness, hue and saturation, known as luminance-chrominance representations, may be desirable from a compression viewpoint, since the human visual system is more sensitive to errors in the luminance component than in chrominance. This article was adapted for data compression methods with zero information loss have been used on image data for some time. PNG is more sophisticated and capable, of using a predictor to prepare the data for a gzip-style compressor. JPEG falls under the general category of transform methods. The process of using a basis to resolve an image into a collection of weights is called a transform. We'll consider gray-scale images which can be represented as m x n arrays of integers. The range of values isn't important in understanding the mathematical ideas, although it is common to restrict values to the interval [0,255], giving a total of 256 levels of gray. Above figure shows an image containing 256x256 pixels with 145 shades of gray represented. Mathematically, any basis for the space of m x n gray-scale images must contain exactly mn images, the number of pixels in an m x n image. Consequently, the transform of an m x n image will have mn weights. However, applications using high-resolution images with thousands of colors may require more compression than can be achieved with these lossless methods.

REFERENCES

[1] An Introduction to Image Compression at http://debugmode.com/imagecmp
[2] Napoleon, D., and M. Praneesh. "Image Enhancement Of Under Water Images Using Structured Preserving Noise Reduction Algorithm."
[3] Philip M. Parker Huffman Coding, March 18-25, 2005.
[4] Abramson, A., Information Theory and Coding, McGraw-Hill, New York, 1963.
[5] Greg A. Harris and Darrel Hankerson Transform Methods and Image Compression, January 1st, 1999.
[6] Napoleon, D., Sathya, S., Praneesh, M., & Subramanian, M. S. (2012). REMOTE SENSING IMAGE COMPRESSION USING 3D-SPIHT ALGORITHM AND 3D-OWT. International Journal on Computer Science and Engineering,4(5), 899.
[7] MacKay, D.J.C., Information Theory, Inference, and Learning Algorithms, Cambridge University Press, 2003.
[8] Wikipedia, http://en.wikipedia.org/wiki/Encoding
[9] Hu, Y.C. and Chang, C.C., "A new losseless compression scheme based on Huffman coding scheme for image compression"
[10] Prof. Kyu Ho Park, Lecture 15 Trees, April 10,2008.
[11] Juliet Bernstein Data Structures and Huffman Coding DIP II Jan. 29, 2008
[12] Huffman compression at http://www.prepressure.com/library
[13] Delp. E.J. and Mitchell, O.R., Image compression using block truncation coding, IEEE transaction Comm. Com-27:1335-1342,1979.
[14] Dutta Majumder, D. Chanda, B. and Mali, P.C. Mathematical tools for image restoration and data compression, J.Inst. Electronics and Tellcom, Engrs., 35: 120-135, 1989.
[15] Chanda. B, Dutta Majumder. D, Digital Image Procession and Analysis, Prentice-Hall of India, ISBN-81-203-1618-5, 20074.
[16] Praneesh, M., and Jaya R. Kumar. "Article: Novel Approach for Color based Comic Image Segmentation for Extraction of Text using Modify Fuzzy Possiblistic C-Means Clustering Algorithm." IJCA Special Issue on Information Processing and Remote Computing IPRC (1) (2012): 16-18.
[17] Napoleon, D., M. Praneesh, S. Sathya, and M. SivaSubramani. "An efficient modified fuzzy possibilistic c-means algorithm for segmenting color based hyperspectral images." In Advances in Engineering, Science and Management (ICAESM), 2012 International Conference on, pp. 798-803. IEEE, 2012.
[18] Napoleon, D., and M. Praneesh. "A Comparative Study on Optimization Techniques for Classifying Remote Sensing Images."