International Journal of Computer Applications (0975 – 8887)
Volume 20– No.6, April 2011

A Novel Approach for Reduction of Huffman Cost Table in Image Compression

S.Mohankrishna
Assistant Professor
Department of IT, GIT
GITAM University

Singuru SriHari
Department of IT, GIT
GITAM University

T.V. Trinadh
Department of IT, GIT
GITAM University

G. Raja Kumar
Department of IT, GIT
GITAM University

ABSTRACT

Huffman codes are being extensively used as a very efficient technique for image compression. To obtain a high compressing ratio, the cost table need to be reduced. A new approach has been defined which reduces the cost table of the traditional Huffman Algorithm. This paper presents a minor modification to the Huffman coding of the binary Huffman compression algorithm. A study and implementation of the traditional Huffman algorithm is studied. In this paper a new methodology has been proposed for the reduction of the cost table for the image compression using Huffman coding Technique. Compared with the traditional Huffman coding the proposed method yields the best results in the generation of cost tables. The advantages of new binary Huffman table are that the space requirement and time required to transmit the image is reduced significantly.

Keywords: Huffman codes, Cost tables, image compressions, Redundancy

1. INTRODUCTION

In computer science and information theory, Image compression [1,2,3,9] is the process of encoding Image using less than an un-encoded representation would use, through use of specific encoding schemes. Huffman coding as a high efficient method is playing a larger role in image compression [1,2,3,9,10]. Therefore, how to save the algorithm’s storage space and how to increase its compression ratio have always been concerned. The compressed image obtained by Huffman algorithm [4,6,7,9] consists of two parts, one is the compressed source file and the other is the file head (Huffman Table) used to decode or the mapping table between the symbols in the source file and the related codes in the compressed image. The nature of Huffman coding [2,4,8] algorithm decides the constancy of the source image’s compression ratio, so the algorithm’s compression ratio is directly related to the cost of Huffman table.

With the rapid development of information technology, Huffman coding as a high efficient method is playing a larger role in text/ picture/ video compression, telecommunication and so on. Therefore, how to save the algorithm’s storage space and how to increase its compression ratio have always been concerned [1, 4-6]. The compressed file obtained by Huffman algorithm consists of two parts: one is the compressed source file and the other is the file head (Huffman Table) used to decode, or the mapping table between the symbols in the source file and the related codes in the compressed file

This paper studies and implements the high compression ratio Huffman algorithm based on Huffman cost table. In the study, we manage to obtain more reduced Huffman table using traditional Huffman tree.

2. VARIOUS TECHNIQUES

Some of the Image Compression techniques are,
1. Binary Huffman Coding
2. Shannon Coding
3. Arithmetic Coding
4. Huffman Shift Coding

3. BINARY HUFFMAN ALGORITHM

The binary Huffman coding [1, 2, 4, 8,13] is the process of dividing the image into matrix of colour codes representing various colours and then it will calculate the probabilities of occurrences of each pixel, put them into descending order and then will append the binary digits to each probability/pixel in a systematic way such that no two or more pixels should have same binary code. After obtaining these values we can make two types of files. One representing compressed file and other is Huffman cost table[1,2,3,4] used to decode or the mapping table between the symbols in the source file and the related codes in the compressed image.

Algorithm for constructing the Binary Huffman tree is,
Step1. Start
Step2. Convert image into matrix of colour codes.
Step3. Calculate probabilities for each colour code.
Step4. Select the two parentless nodes with the lowest probabilities.
Step5. Create a new node which is the parent of the two lowest probability nodes.
Step6. Assign the new node a probability equal to the sum of its children's probabilities.
Step7. Repeat Step 1 until there is only one parentless node left.
Step8. Stop

The flow chart for the above algorithm is,

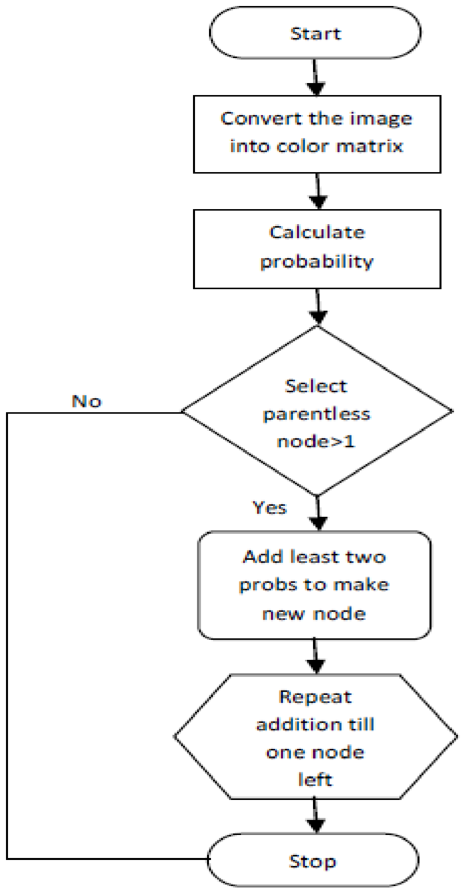
Start
Convert the image into color matrix
Calculate probability
Select parentless node>1
No
Yes
Add least two probs to make new node
Repeat addition till one node left
Stop

Figure 1: Binary Huffman Tree

Once a Huffman tree[1,2,5,6,9] is built, Canonical Huffman codes, which require less information to rebuild, it will be generated by the following steps:

Step1. Start

Step2. Assign binary digits 0 and 1 to the last iteration probabilities.

Step3. For the previous iteration assign the same codes of parent node to its children nodes.

Step4. Then append binary digits 0 and 1 again to child nodes respectively.

Step5. If it is not a parent node then assign the same code as in the next iteration.

Step6. Repeat the process until all symbols or pixels are assigned codes.

Step7. Stop

Encoding Data

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

Decoding Data

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.

4. PROPOSED APPROACH FOR REDUCING THE COST TABLE

In our proposed system the binary Huffman coding is divided into two parts. First part is separate the highest two probabilities from the image and assign 0 and 1 to those probabilities and for second part do binary Huffman coding for remaining probabilities as it is. By doing this we can send the pixel with highest occurrences by single bit itself, otherwise if we do Normal binary Huffman coding[1,2,13] we may occur more than one bit code for the highest probability pixel, so its waste of memory to send more bits for more times for higher probability pixel.

Algorithm for constructing the Huffman tree from the image is,

Step1. Start

Step2. Convert image into matrix of colour codes.

Step3. Calculate probabilities for each colour code and place them in descending order.

Step4. Select the top two probabilities and assign them 0 and 1.

Step5. Select the two parentless nodes with the lowest probabilities.

Step6. Create a new node which is the parent of the two lowest probability nodes.

Step7. Assign the new node a probability equal to the sum of its children's probabilities.

Step8. Repeat Step 1 until there is only one parentless node left.

Step9. Stop

The flow chart for the above algorithm is,

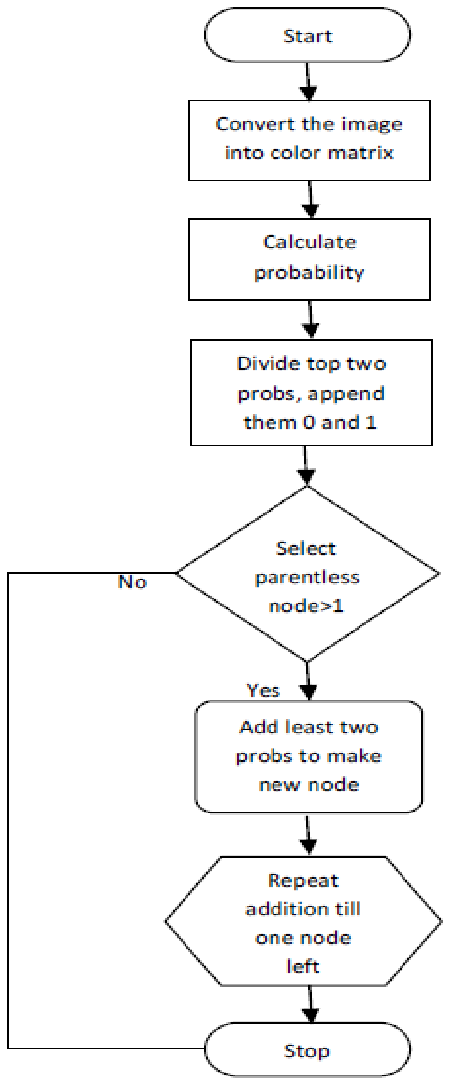
Start
Convert the image into color matrix
Calculate probability
Divide top two probs, append them 0 and 1
Select parentless node>1
No
Yes
Add least two probs to make new node
Repeat addition till one node left
Stop

Figure 2: Huffman Tree from Image

Once a Huffman tree is built, Canonical Huffman codes[1,3,5,9], which require less information to rebuild, it will be generated by the following steps:

Step1. Start
Step2. Assign binary digits 0 and 1 to the higher two probability pixels.
Step3. For the remaining part of an image assign binary digits 0 and 1 to the last iteration probabilities.
Step4. For the previous iteration assign the same codes of parent node to its children nodes.
Step5. Then append binary digits 0 and 1 again to child nodes respectively.
Step6. If it is not a parent node then assign the same code as in the next iteration.
Step7. Repeat the process until all symbols or pixels are assigned codes.
Step8. Stop

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

Decoding Data
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.

5. ASSUMPTION
The third probability of an image must be lesser than the sum of probabilities of the image from fifth probability.

6. EXPERIMENTAL RESULTS
This software tutorial gives a comparative study on both of the two algorithms Huffman and proposed. According to the experimental results obtained, it is observed that the efficiency of the proposed cost table is high and the space and time complexity is low The results can be seen from the figure 3

6.1 According to traditional Binary Huffman algorithm

Pix	**Prob**		**Ite-1**		**Ite-2**		**Ite-3**		**Ite-4**		**Ite-5**		**Ite-6**		**Ite-7**	
9	.30	00	.30	00	.30	00	.30	00	.30	00	.30	00	.40	1	.60	0
7	.17	010	.17	010	.17	010	.18	11	.22	10	.30	01	.30	00	.40	1
4	.13	011	.13	011	.13	011	.17	010	.18	11	.22	10	.30	01		
8	.12	100	.12	100	.12	100	.13	011	.17	010	.18	11				
6	.10	101	.10	101	.10	101	.12	100	.13	011						
3	.09	110	.09	110	.09	110	.10	101								
1	.05	1110	.05	1110	.09	111										
5	.03	11100	.04	1110												
2	.01	11101														

Table1: Huffman Tree Formation

Pix	9	7	4	8	6	3	1	5	2
Prob	.30	.17	.13	.12	.10	.09	.05	.03	.01
Avg	.60	.51	.39	.36	.30	.27	.20	.15	.05

Table 2: Huffman cost Table obtained form Table1

Efficiency

Avg. Number of bits per pixel to send: **2.83**

Compression ratio is: **[(4-2.83)/4]*100**

= 29.25%

The number of iterations is: **7**

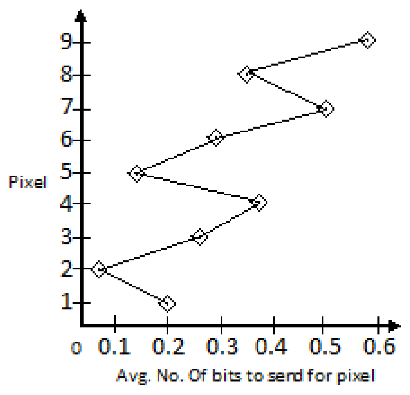

Figure 3: Huffman Tree from Table 2

6.2 According to Proposed algorithm

Pix	Prob		Ite-1		Ite-2		Ite-3		Ite-4		Ite-5	
9	.30	0										
7	.17	1										
4	.13	01	.13	01	.13	01	.18	00	.22	1	.31	0
8	.12	10	.12	10	.12	10	.13	01	.18	00	.22	1
6	.10	11	.10	11	.10	11	.12	10	.13	01		
3	.09	000	.09	000	.09	000	.10	11				
1	.05	0010	.05	0010	.09	001						
5	.03	00110	.04	0011								
2	.01	00111										

Table 3: Proposed algorithm Tree Formation

Pix	9	7	4	8	6	3	1	5	2
Prob	.30	.17	.13	.12	.10	.09	.05	.03	.01
Avg	.30	.17	.26	.24	.20	.27	.20	.15	.05

Table 4: Proposed cost Table obtained form Table 3

Efficiency

Avg. Number of bits per pixel to send: **1.84**

Compression ratio is: **[(4-1.84)/4]*100**

= 54%

The number of iterations is: **5**

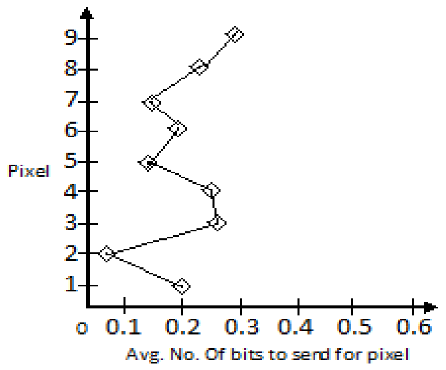

Figure 4: Proposed algorithm Tree from Table 4

7. CONCLUSION AND FUTURE SCOPE

With the advancements in compression technology, it is now very easy and efficient to compress the images. Various image compression techniques are available. The most common image compression standard is JPEG (Joint Picture Experts Group). Many advances are being made for improving the image quality. Applied fields that are making use of this technique in the coming future include astronomy, acoustics, nuclear engineering, sub-band coding, signal and image processing, neurophysiology, magnetic resonance imaging, speech discrimination, optics, fractals, turbulence, earthquake-prediction, radar, human vision, and pure mathematics applications such as solving partial differential equations.

8. REFERENCES

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

[2] Wang Ling, Cheng Li, “A new construction method for arbitrary Huffman trees with K elements [J]”, Aeronautical Computer Technique, 1998, 28(4): 12-15

[3] Optimization of Audio Codecs on NEON, www.iqmagazineonline.com/archive26/pdf/Pg36-39.pdf

[4] Chirantan Kumar, Pranav Mishra, Prachi Choudhary ,Kaustubh Joshi, Video Codec Optimizations on Cortex A8, “Proceedings of the 5th International Conference on Wireless communications, networking and mobile computing” Beijing, China, 4697-4700, 2009.

[5] MPEG, Coding of moving pictures and associated audio for digital storage media at up to 1.5 Mbit/s, part 3: Audio. “International Standard IS 11172-3, ISO/IEC JTC1/SC29 WG11”, 1992.

[6] Vladimir Z Mesarovic, Raghunath Rao, Miroslav V Dokic, Sachin Deo, Selecting an Optimal Huffman Decoder for AAC, “Audio Engineering Society Convention 111” New York, USA, Nov. 2001.

[7] Gutemberg G.S, Lima M.A.M, Filho W.O.G, Perkusich A, Morais M.R.A, Lima A.M.N, A Fast and Memory Efficient Huffman Decoding Method for the MPEG-4 AAC Standard, “ICCE 2008” Las Vegas, USA, Jan. 2008.

[8] System reference manual of BeagleBoard, “http://beagleboard.org/static/BBSRM latest.pdf”.

[9] ARM Corp. RealView Compilation Tools Compiler Reference Guide Version 4.0.

[10] Bei Chen, Hongcai Zhang, Wenlun Cao, Jianhu Feng. Huffman Coding Method Based on Number Character. Machine Learning and Cybernetics, 2007 International Conference on. Aug. 2007. 2296 – 2299.

[11] Kavousianos. X, Kalligeros. E, Nikolos. D. Multilevel-Huffman Test-Data Compression for IP Cores With Multiple Scan Chains. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on. July 2008. 926 – 931.

[12] ARM Corp. ARM Architecture Reference Manual, Advanced SIMD Extension and VFPv3 supplement

[13] ARM Corp. Technical Reference Manual Revision: r3p2