

Lossless Grey-scale Image Compression using Source Symbols Reduction and Huffman Coding

C. SARAVANAN

cs@cc.nitdgp.ac.in

Assistant Professor, Computer Centre, National Institute of Technology,
Durgapur, West Bengal, India, Pin – 713209.

R. PONALAGUSAMY

rpalagu@nitt.edu

Professor, Department of Mathematics, National Institute of Technology,
Tiruchirappalli, Tamilnadu, India, Pin – 620015.

Abstract

Usage of Image has been increasing and used in many applications. Image compression plays vital role in saving storage space and saving time while sending images over network. A new compression technique proposed to achieve more compression ratio by reducing number of source symbols. The source symbols are reduced by applying source symbols reduction and further the Huffman coding is applied to achieve compression. The source symbols reduction technique reduces the number of source symbols by combining together to form a new symbol. Therefore, the number of Huffman code to be generated also reduced. The Huffman code symbols reduction achieves better compression ratio. The experiment has been conducted using the proposed technique and the Huffman coding on standard images. The experiment result has analyzed and the result shows that the newly proposed compression technique achieves 10% more compression ratio than the regular Huffman coding.

Keywords: Lossless Image Compression, Source Symbols Reduction, Huffman Coding.

1. INTRODUCTION

The image compression highly used in all applications like medical imaging, satellite imaging, etc. The image compression helps to reduce the size of the image, so that the compressed image could be sent over the computer network from one place to another in short amount of time. Also, the compressed image helps to store more number of images on the storage device [1-4].

It's well known that the Huffman's algorithm is generating minimum redundancy codes compared to other algorithms [6-11]. 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. [12]. 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 [13 and 14].

2. PROPOSED COMPRESSION TECHNIQUE

The number of source symbols is a key factor in achieving compression ratio. A new compression technique proposed to reduce the number of source symbols. The source symbols combined together in the same order from left to right to form a less number of new source symbols. The source symbols reduction explained with an example as shown below.

The following eight symbols are assumed as part of an image, 1, 2, 3, 4, 5, 6, 7, 8. By applying source symbols reduction from left to right in the same sequence, four symbols are combined together to form a new element, thus two symbols 1234 and 5678 are obtained. This technique helps to reduce 8 numbers of source symbols to 2 numbers i.e. 2^n symbols are reduced to $2^{(n-2)}$ symbols. For the first case, there are eight symbols and the respective Symbols and Huffman Codes are 1-0, 2-10, 3-110, 4-1110, 5-11110, 6-111110, 7-1111110, 8-1111111. The proposed technique reduced the eight symbols to two and the reduced Symbols and Huffman codes are 1234-0, 5678-1.

The minimum number of bits and maximum number of bits required to represent the new symbols for an eight bit grayscale image calculated. The following possible combinations worked out and handled perfectly to ensure the lossless compression. The following are few different possible situations to be handled by source symbols reduction.

If all symbols in the four consecutive symbols are 0, i.e. 0 0 0 0, then the resulting new symbol will be 0.

If the four consecutive symbols are 0 0 0 1, then the resulting new symbol will be 1.

If the four consecutive symbols are 0 0 1 0, then the resulting new symbol will be 1000.

If the four symbols are 0 1 0 0, then the resulting new symbol will be 1000000.

If the four symbols are 1 0 0 0, then the resulting new symbol will be 1000000000.

If the four symbols are 255 255 255 255, then the resulting new symbol will be 255255255255.

The average number L_{avg} of bits required to represent a symbol is defined as,

$$L_{avg} = \sum_{k=1}^L l(r_k) p_r(r_k) \quad (1)$$

where, r_k is the discrete random variable for $k=1,2,\dots,L$ with associated probabilities $p_r(r_k)$. The number of bits used to represent each value of r_k is $l(r_k)$. The number of bits required to represent an image is calculated by number of symbols multiplied by L_{avg} [5].

In the Huffman coding, probability of each symbols is 0.125 and $L_{avg} = 4.175$.

In the proposed technique, probability of each symbol is 0.5 and $L_{avg}=1.0$.

The L_{avg} confirms that the proposed technique achieves better compression than the Huffman Coding.

From the above different possible set of data, the following maximum and minimum number of digits of a new symbol formed by source symbols reduction calculated for an eight bits grey-scale image. The eight bits grey-scale image symbols have values ranging from 0 to 255. The minimum number of digits required to represent the new symbol could be 1 digit and the maximum number of digits required to represent the new symbols could be 12 digits. Therefore, if the number of columns of the image is multiples of four, then this technique could be applied as it is. Otherwise, the respective remaining columns (1 or 2 or 3 columns) will be kept as it is during the source symbols reduction and expansion.

Four rows and four columns of eight bits grey-scale image having sixteen symbols considered to calculate required storage size. To represent these 16 symbols requires $16 \times 1 \text{ byte} = 16$ bytes storage space. The proposed source symbol reduction technique reduces the 16 symbols into 4 symbols. The four symbols require $4 \times 4 \text{ bytes} = 16$ bytes. Therefore, the source symbols data and the symbols obtained by the source symbols reduction requires equal amount of storage space. However, in the coding stage these two techniques make difference. In the first case, sixteen symbols generate sixteen Huffman codes, whereas the

proposed technique generates four Huffman codes and reduces L_{avg} . Therefore, the experiment confirms that the source symbols reduction technique helps to achieve more compression.

The different stages of newly proposed compression technique are shown in figure 1. The source image applied by source symbols reduction technique then the output undergoes the Huffman encoding which generates compressed image. In order to get the original image, the Huffman decoding applied and an expansion of source symbols takes place to reproduce the image.

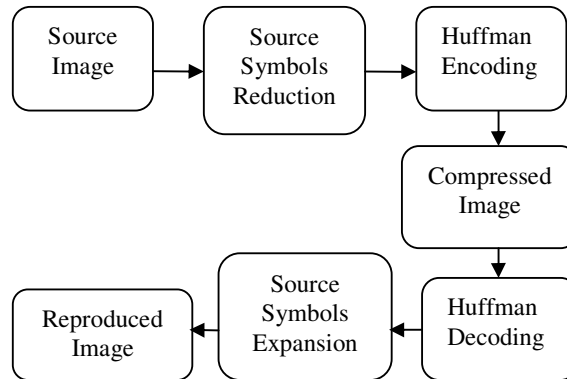


FIGURE 1: Proposed Compression Technique

Five different test images with different redundancy developed for experiment from 0% to 80% in step size of 20% i.e 0%, 20%, 40%, 60%, and 80% redundancy. The Huffman coding could not be applied on data with 100% redundancy or single source symbol, as a result 100% redundancy is not considered for the experiment. The test images with 16 rows and 16 columns will have totally 256 symbols. The images are 8 bit grey-scale and the symbol values range from 0 to 255. To represent each symbol eight bit is required. Therefore, size of an image becomes $256 \times 8 = 2048$ bit. The five different level redundancy images are applied the Huffman coding and the proposed technique. The compressed size and time required to compress and decompress (C&D) are noted.

3. EXPERIMENT RESULTS

Following table 1 shows the different images developed for the experiment and corresponding compression results using the regular Huffman Coding and the proposed technique. The images are increasing in redundancy 0% to 80% from top to bottom in the table.






IMAGE	Huffman Coding	SSR+HC Technique
	Compressed size (bits)	Compressed size (bits)
	2048	384
	1760	344
	1377	273
	944	188
	549	118

TABLE 1: Huffman Coding Compression Result

The experiment shows that the higher data redundancy helps to achieve more compression. The experiment shows that the proposed compression technique achieves more compression than the Huffman Coding. The first image has 0% redundancy and its compressed image size is 2048 bit using the Huffman coding whereas the proposed compression technique has resulted compressed image of size 384 bit. No compression takes place for the first image using Huffman coding, where as the proposed technique achieved about 81% compression.

For all images the compressed size obtained from the proposed technique better than the Huffman coding. The proposed compression technique achieves better compression. The results obtained from the present analysis are shown in figure 2.

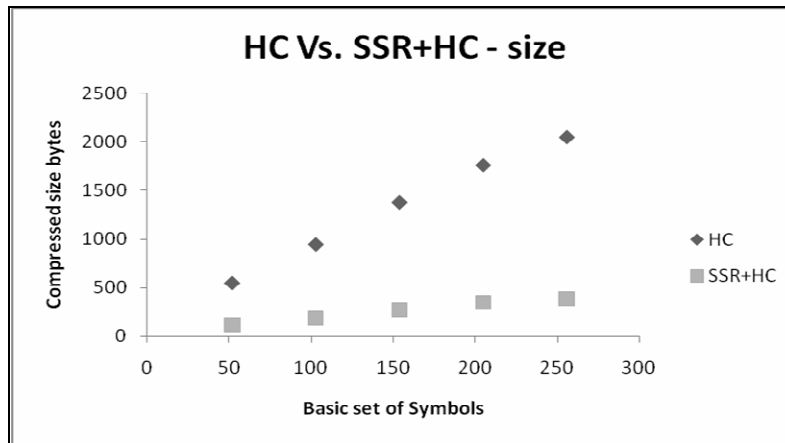


FIGURE 2: Compressed Size comparisons

Table 2 shows the comparison between these two techniques. Compression Ratio (CR) is defined as

$$CR = \frac{\text{Original size}}{\text{Compressed size}} \quad (2)$$

Redundancy	Huffman Coding	SSR+HC Technique
	Compression Ratio	Compression Ratio
0%	1.0000	5.3333
20%	1.1636	5.9535
40%	1.4873	7.5018
60%	2.1695	10.8936
80%	3.7304	17.3559

TABLE 2: Compression Ratio versus Time

From the result of the experiment it is found that the two compression techniques are lossless compression technique, therefore the compression error not considered. The following figure 3 compares the compression ratio of the experiment. From the figure it is observed that the proposed technique has performed better than the Huffman Coding. The proposed technique shows better compression ratio for the images having higher redundancy when compared with the images of lower redundancy.

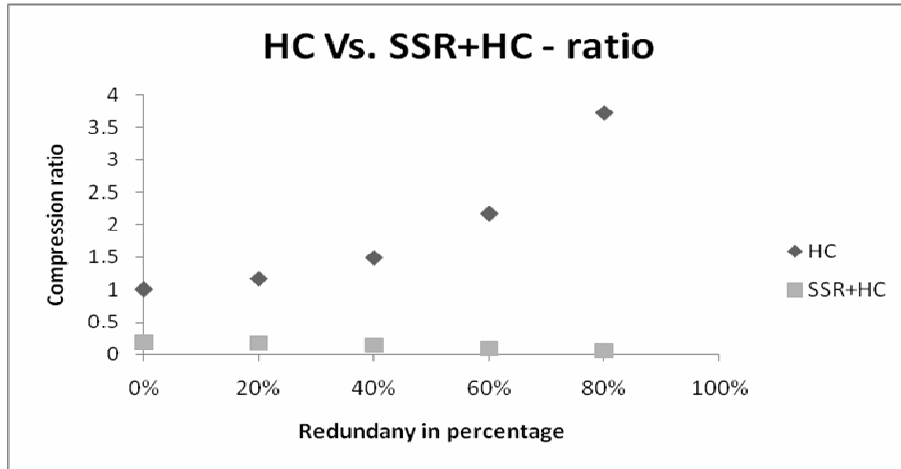


FIGURE 3: Compression ratio comparisons

In the real time, images are usually having higher data redundancy. Hence, the proposed technique will be suitable for the user who desires higher compression. Moreover, standard gray scale images considered for testing. The standard images require 65,536 bytes storage space of 256 rows and 256 columns. The image is eight bit gray scale image.

The standard images applied using the two the compression techniques and standard JPEG compression technique. The compression size of the experiment is noted. The following figure 4 is one of the source image used for the experiment and figure 5 is the reproduced image using the proposed technique.

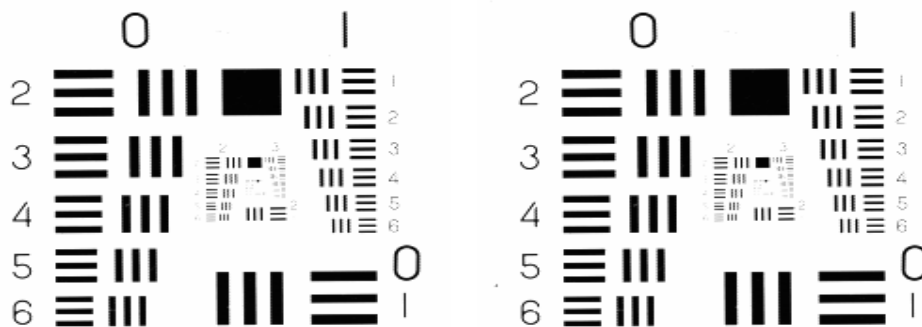


FIGURE 4: Source image chart.tif

FIGURE 5: Reproduced image chart.tif

Table 3 shows the compression result using Huffman coding, and the proposed technique for one of the standard image *chart.tif*. The proposed technique has achieved better compressed size than the Huffman coding. The source symbols reduction and expansion takes more time if the number of symbols are higher. Hence, the newly proposed technique is suitable to achieve more compression.

Source Image	Huffman Coding	SSR+HC Technique
Size (bits)	Compressed size (bits)	Compressed size (bits)
5,128,000	1,015,104	54,207

TABLE 3: Compression test results for chart.tif

4. CONCLUSIONS

The present experiment reveals that the proposed technique achieves better compression ratio than the Huffman Coding. The experiment also reveals that the compression ratio in Huffman Coding is almost close with the experimental images. Whereas, the proposed compression technique Source Symbols Reduction and Huffman Coding enhance the performance of the Huffman Coding. This enables us to achieve better compression ratio compared to the Huffman coding. Further, the source symbols reduction could be applied on any source data which uses Huffman coding to achieve better compression ratio. Therefore, the experiment confirms that the proposed technique produces higher lossless compression than the Huffman Coding. Thus, the proposed technique will be suitable for compression of text, image, and video files.

5. REFERENCES

1. Gonzalez, R.C. and Woods, R.E., Digital Image Processing 2nd ed., Pearson Education, India, 2005.
2. Salomon, Data Compression, 2nd Edition. Springer, 2001.
3. Othman O. Khalifa, Sering Habib Harding and Aisha-Hassan A. Hashim, Compression using Wavelet Transform, Signal Processing: An International Journal, Volume (2), Issue (5), 2008, pp. 17-26.
4. Singara Singh , R. K. Sharma, M.K. Sharma, Use of Wavelet Transform Extension for Graphics Image Compression using JPEG2000 Framework, International Journal of Image Processing, Volume 3, Issue 1, Pages 55-60, 2009.
5. Abramson, N., Information Theory and Coding, McGraw-Hill, New York, 1963.
6. Huffman, D.A., A method for the construction of minimum-redundancy codes. Proc. Inst. Radio Eng. 40(9), pp.1098-1101, 1952.
7. Steven Pigeon, Yoshua Bengio — A Memory-Efficient Huffman Adaptive Coding Algorithm for Very Large Sets of Symbols — Université de Montréal, Rapport technique #1081.
8. Steven Pigeon, Yoshua Bengio — A Memory-Efficient Huffman Adaptive Coding Algorithm for Very Large Sets of Symbols Revisited — Université de Montréal, Rapport technique #1095.
9. R.G. Gallager — Variation on a theme by Huffman — IEEE. Trans. on Information Theory, IT-24(6), 1978, pp. 668-674.
10. D.E. Knuth — Dynamic Huffman Coding — Journal of Algorithms, 6, 1983 pp. 163-180.
11. J.S. Vitter — Design and analysis of Dynamic Huffman Codes — Journal of the ACM, 34#4, 1987, pp. 823-843.
12. Chiu-Yi Chen; Yu-Ting Pai; Shanq-Jang Ruan, Low Power Huffman Coding for High Performance Data Transmission, International Conference on Hybrid Information Technology, 2006, 1(9-11), 2006 pp.71 – 77.
13. Lakhani, G, Modified JPEG Huffman coding, IEEE Transactions Image Processing, 12(2), 2003 pp. 159 – 169.
14. R. Ponalagusamy and C. Saravanan, Analysis of Medical Image Compression using Statistical Coding Methods, Advances in Computer Science and Engineering: Reports and Monographs, Imperial College Press, UK, Vol.2., pp 372-376, 2007.