

Efficient Lossless Colour Image Compression Using Run Length Encoding and Special Character Replacement

Debashis Chakraborty¹, Soumik Banerjee²

Department of Computer Science and Engineering,
St. Thomas' College of Engineering and Technology
Kolkata – 700023, India.

¹sunnydeba@gmail.com

²soumik.stcet@gmail.com

Abstract— Image compression, in the present context of heavy network traffic, is going through major research and development. The lossless compression techniques, presently in practise, follow three basic paradigms – character repetition removal, frequency measurement-encoding and dictionary maintenance. In the proposed method, the character repetition removal and dictionary maintenance concepts were incorporated together. The output encoded file had yielded comparably better results than standard lossless image compression techniques. The proposed method had been tested on a series of continuous and discrete tone standard test images.

Keywords: *lossless image compression, colour images, Run Length Encoding, dynamic dictionary, ASCII character set.*

I. INTRODUCTION

Image compression is the process of encoding image data into lesser number of symbols such that after decoding, the original image information can be retrieved. The compression procedure facilitates optimized space utilization for storage purposes and also enhances network utilization by using lesser bandwidth [1,8,15]. With the ever-increasing growth of multimedia applications over a network, on-line compression has become a necessity. Lossless image compression utilizes statistical redundancy in the uncompressed image. Repetitive groups of same pixel are encoded in Run Length Encoding algorithm [1,8,15]. In Huffman Encoding [1,8,15] the frequency of occurrence of a particular pixel value is used to encode the image pixel information using variable size bit-words. This process is also known as entropy encoding. The dictionary maintenance procedure postulates that image pixel information should be stored in a dictionary, and each entry will be represented using a key. These keys are later utilized for image encoding. Lempel-Ziv-Welch or LZW algorithm [1,5,6,8,15] is the most famous algorithm of this genre. But the dictionary method had an inherent disadvantage because it had to store the dictionary which was an additional overhead.

Many advanced research had been conducted in this context for achieving better compression efficiency without increasing resource requirement. The SCRCA technique [12,13] proposed to compress gray-scale images by maintaining a dictionary of repetitive character set in the image matrix. The FELICS [3,5] technique proposes to encode each pixel value by a variable size bit-code depending on the neighbourhood of the pixels. The CALIC [7,9] algorithm uses three steps to produce a context among neighbouring pixels and then use quantization to reduce context difference in the neighbourhood. The implementation of hardware level compression [2,4] of raw image information in the run time is also being heavily researched for medical image processing.

In the proposed algorithm, image information had been firstly scanned for statistical redundancy following Run length encoding schemes. Then, the frequency-pixel pairs, thus generated, were encoded using Extended ASCII character-set (8 bit) [14] using an efficient approach explained in later part of this paper. The corresponding symbol-information pairs had been stored in dictionary format that was created and deployed individually with each image. No permanent storage of the dictionary was required.

The remaining paper had been organized as follows:- section II introduces the compression algorithm and discusses the character replacement scheme in details, section III discusses the decompression algorithm, section IV tabulates the result analysis and in section V the paper is concluded.

II. COMPRESSION ALGORITHM

The compression algorithm had two interrelated procedures to facilitate the entire compression process. They were Run Length Encoding and Character Replacement scheme. The colour image information is

effectively stored in 3 or 4 channels depending on the colour model of the image. These channels are stored as matrices. For processing, each of these matrices were separated and processed independently as follows:

A. Run Length Encoding (RLE)

Individual channel matrices were retrieved and used for processing. Firstly each matrix was scanned row-wise for identifying repetitive pixels. Each group of such repetitions were then replaced by the pixel value and the frequency of occurrence. This was done exhaustively throughout the image matrix. For single occurrence of a particular value frequency was not used because that would cause an overhead affecting the compression efficiency.

Sample processing example:

Input stream: 22 22 22 57 57 57 57 33 33 33 33 33 22

Output stream: 322 457 533 22

The output stream produced a series of frequency-pixel value pairs as previously discussed. This new representation of original matrix information for individual matrices was forwarded to the next section for processing.

B. Character Replacement Scheme

In character replacement, standard ASCII character set had been utilized. Among these, five symbols were considered unfit for the purpose. The rest were divided into two distinct sets A and B and utilized as follows:

1) The first 200 characters were used for single character replacement. They were used to form set A. Unique frequency-pixel information were represented using a single character belonging to this set. The whole set of characters and corresponding maps were stored as a dictionary.

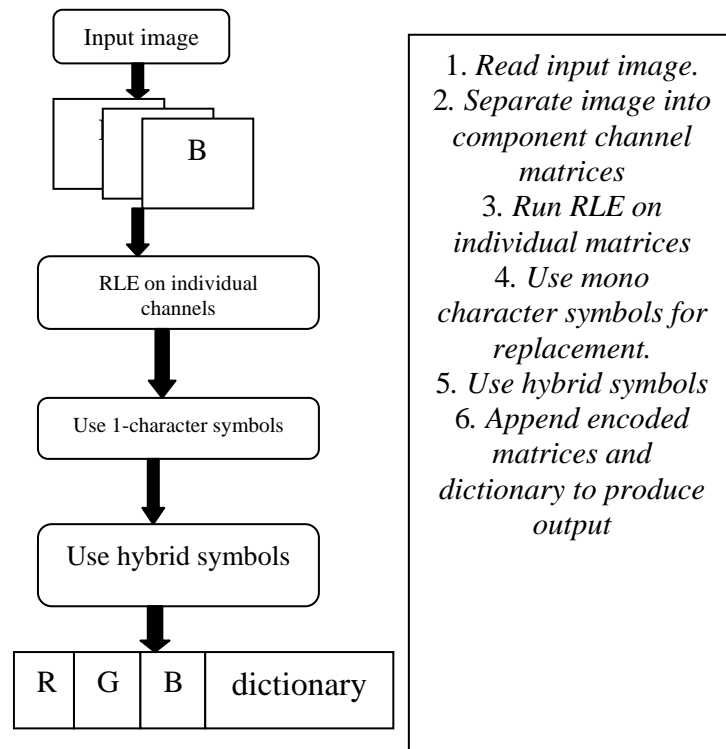
2) The rest remaining characters were stored in set B. After A was fully utilized the hybrid symbols were used. These symbols were 2-character hybrid symbols. The leading symbol was formed of elements belonging to set B and the trailing symbol was taken from set A. Each symbol in set B was coupled with all symbols belonging to set A. The remaining pixel-frequency pairs were represented using the hybrid symbols and the corresponding mapping information was stored in the same format as discussed above.

3) Repeat step 1 and step 2 for all matrices representing each channel of colour model followed by the image.

The processed matrices thus produced, were appended consecutively, separated by one of the five previously rejected symbols (C). This symbol, C, signified the beginning of a different matrix information and end of another.

The procedure gave an effective symbol space of 200 (for set A) + (50X200) (for set B) = 10,200. This space was experimentally found sufficient to represent all images in the experimental images set.

The algorithm is summarized in the following flowchart:



III. DECOMPRESSION SCHEME

The decompression process involves two interrelated phases: the Dictionary Reconstruction process and compressed file Decoding procedure to reproduce the original image.

A. Dictionary Reconstruction

From the two dictionaries exported with the compressed file, two new temporary data structures were created, preferably a link-list for search-time and space optimization, containing the mapping information of the symbols. One structure (A') contained the mapping information of single character symbols and the other (B') corresponded to the hybrid symbols. These two data-structures were referenced later for decoding of the compressed image.

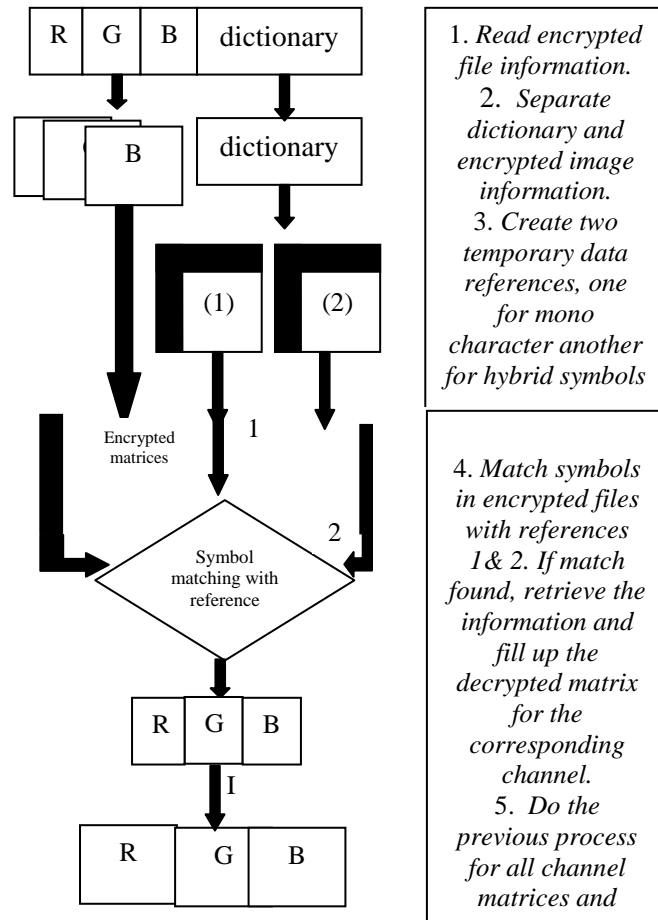
B. Decoding

The decoding scheme involves an exhaustive seek based processing of the encoded information. Firstly, each symbol is read and three checks were performed on it: if it was C, if it belonged to A' or if it belonged to B'. Secondly, according to the check result one of the following was done:-

- 1) If it was C, then one matrix was closed and finalized and the next matrix was opened for data entry. The order of matrix encoding and decoding is according to the corresponding colour model channel naming order convention.
- 2) If it belonged to A', then from the table the corresponding frequency-pixel value was extracted and number of cells equal to the frequency in the matrix under processing was assigned the pixel value.
- 3) If it belonged to B', then from the corresponding table the frequency-pixel information was extracted and number of cells equal to the frequency in the matrix under processing was assigned the pixel value.

The aforesaid process was continued till the EOF of the file was reached. The matrices thus produced were superimposed over each other according to the format of the uncompressed image file format. In our study, we used files with .PPM extension. In this file, information of a particular position in all the 3 channels (R-G-B) was written one after another. Thus the original image was recreated using proper processing.

The process is summarized in the following flowchart:



IV. SIMULATION RESULTS

The discussed algorithm was implemented in C++. A set of standard test images (samples shown in fig 1) was chosen for simulation process.

The mathematical definitions used to study the results of our simulations are stated below:-

1)

Compression percentage = [Compressed File Size/Actual File Size]*100 , and

$$RMSE = \sqrt{\frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N [f(i, j) - \hat{f}(i, j)]^2}$$

The compression percentage achieved for different test images belonged in the range of 60 to 75%. The error deviation from the original image, in all cases was found to be zero. This observation proved the lossless character of the proposed algorithm.

The following table, table 1, shows the comparative study of performance of our algorithm with other standard algorithms. The table clearly justifies a better performance of the proposed algorithm in comparison.

A simulation run on one of the channel matrices under the proposed algorithm had been shown below. In the example, , a part of the RED component matrix had been shown:

A. Compression Encoding

Initial Input stream:

...22 22 22 57 57 57 57 33 33 33 33 33 22 22 22...

1) RLE Output stream:

...322 457 533 322...

2) Identifying Unique Frequency-Pixel Pairs:

...322 457 533...

3) Assignment of ASCII Character Set:

a: 322

b: 457

c: 533

.

.

(This is an illustration, for convenience the symbols “a”, “b” and “c” were used. In practice the assignment should start with ASCII character 0 and henceforth).

4) The Processed Output Stream:

...abca... + (dictionary of step 3)

B. Decompression Process**1) Reconstructing Dictionary**

From the exported dictionary, a data structure, as shown in table II, was created containing the symbol-information mapping information.

TABLE I
REGENERATED DATA STRUCTURE

SYMBOL	INFORMATION
a	322
b	457
c	533

In the simulation this data structure was chosen as a link-list for the inherent space optimization property. The space requirement for this process is a fundamental concern because the data of mapping information for almost 3500 hybrid symbols needs to be stored, as found in an average after experimentation on the entire test case image set.

2) Decoding

Input Stream:

...abca...

Output Stream after information replacement:

...322 457 533 322

(Information replacement was done by remapping symbol to the corresponding information as shown in table II).

3) Final Output after frequency processing:

...22 22 22 57 57 57 57 33 33 33 33 22 22...

(This step involves the replacement of consecutive channel-matrix cells using the pixel values for the number of times as specified by the frequency field).

Thus, the original input stream was recovered from the encoded compressed file. The simulation run did not encase all the facets of the algorithm like the use of separator symbol (C) or the use of hybrid symbol, but gave an outlook to the proposed method's working procedure.

V. CONCLUSION

In the paper a hybrid model of image compression had been discussed that had shown increase in the efficiency of compression without involving comprehensible increase in resource needs. The proposed technique incorporates the concepts of statistical redundancy removal and dictionary method while doing away with the disadvantages of the procedures like compression efficiency for RLE and dictionary storage overhead for dictionary method. The procedure discussed had provided excellent compression percentages that are comparatively better than other standard techniques in present practice. The procedure had the potential for further development for hardware implementation and run-time image compression applications as in medical image processing machines.

ACKNOWLEDGMENT

First, we would like to thank Professor Subarna Bhattacharjee[4], for her valuable advice, support and constant encouragement. Her constant criticisms and reviews gave us the conceptual clarity. We owe a significant lot to all faculty members of the Department of Computer Science and Engineering and the Department of Information Technology. It was in one such class of Design and Analysis of Algorithms that we first envisioned the D.F.E.C.A. algorithm. We would also like to thank our friends for patiently enduring our

explanations. Their reviews and comments were extremely helpful. And of course, we owe our ability to complete this project to our families whose love and encouragement has been our cornerstone.

REFERENCES

- [1] Sami Khuri and Hsiu-Chin Hsu "Interactive Packages for Learning Image Compression Algorithms" lists, requires prior specific permission and/or a fee. ITiCSE 2000, Helsinki, Finland
- [2] S. Bhattacharjee, S. Das, D. Roy Choudhury and P. Pal Chouduri, "A Pipelined Architecture Algorithm for Image Compression", Proc. Data Compression Conference, Saltlake City, USA, March 1997.
- [3] Jorg Ritter and Paul Molitor, "A pipelined architecture for partitioned DWT based lossy image compression using FPGA's," International Symposium on FPGA, pages 201-206, 2001.
- [4] Amiya Halder, Dipak Kumar Kole and Subarna Bhattacharjee, "On-line Colour Image Compression based on Pipelined Architecture" ICCEE- 2009, Dubai, UAE, Dec 28 – 30.
- [5] Pratt, William K. "Digital Image Processing".
- [6] G. K. Wallace, "The JPEG still picture compression standard," Commun. ACM, vol. 34, pp. 31-44, April 1991.
- [7] W. B. Pennebaker and J. L. Mitchell, "JPEG: Still Image Data Compression Standard", Van Nostrand Reinhold, New York, 1993.
- [8] Rafael C. Gonzalez, Richard E. Woods, Digital Image Processing, Pearson Education, 2002
- [9] Tinku Acharya, Ping-Sing Tsai. JPEG2000 Standard for Image Compression.
- [10] E. J. Delp and O. R. Mitchell, "Image compression using block truncation coding," IEEE Transactions on Communications, 27:1335 – 1342, 1979.
- [11] C. Amerijckx, J. D. Legaty, M. Verleysenz, "Image Compression Using Self-Organizing Maps," Systems Analysis Modeling Simulation Vol. 43, No. 11, November 2003.
- [12] Debashish Chakraborty, Amiya Halder "An Efficient Lossless Image Compression Using Special Character Replacement", IEEE ICCET-2010, 13-14 November pp E-62 – E-67, Jodhpur, Rajasthan, India
- [13] Debashish Chakraborty, Sutirtha Ghosh and Joydeep Mukherjee "Efficient Text Compression using Special Character Replacement and Space Removal", International Journal of Computer Engineering & Technology (IJCET), ISSN 0976-6367(Print), ISSN 0976-6375(Online), Volume 1, Number 2, Sept-Oct (2010), pp. 323-331.
- [14] Soumik Banerjee, Debashish Chakraborty, "Fast Near-lossless Image Compression with Tree Coding having Predictable Output Compression Size", International Conference on Advances in Information Technology and Mobile Communication – AIM 2011, April 21-22, 2011, Nagpur, pp 38-42

FIGURE 1

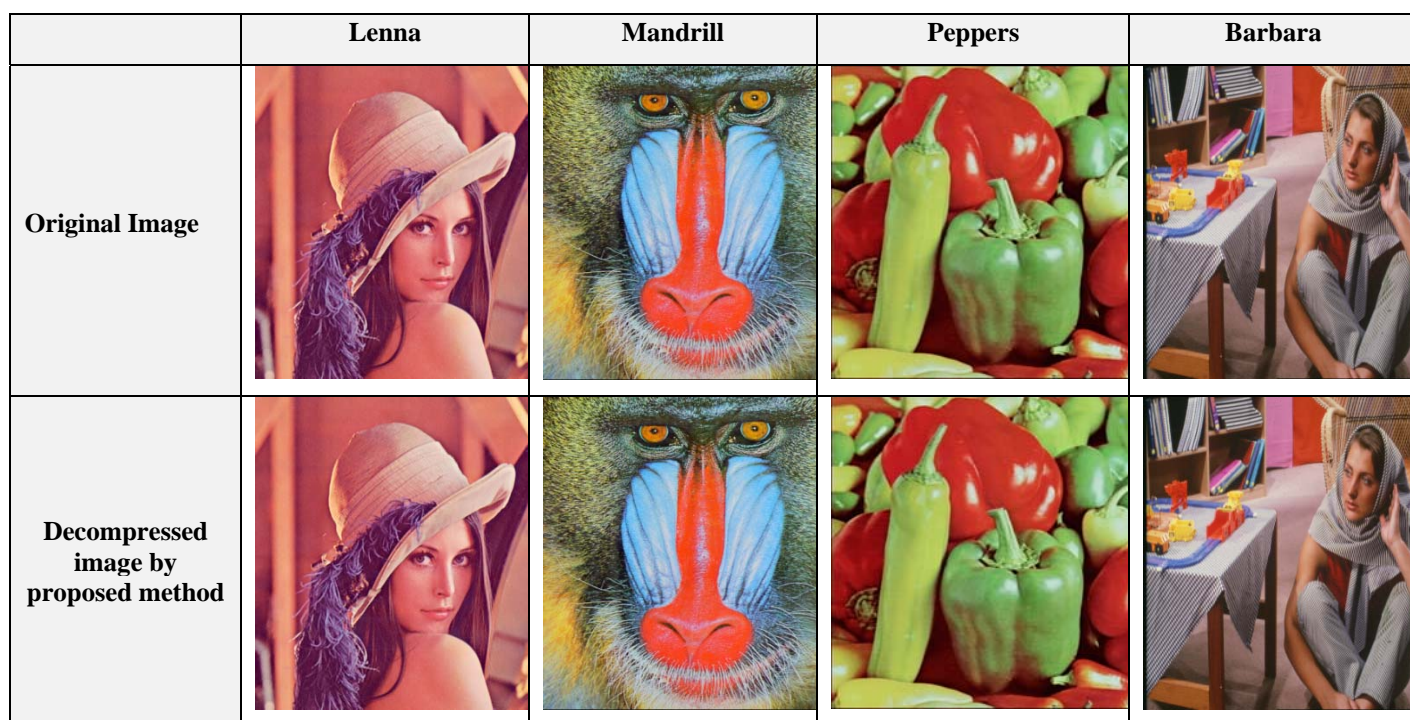


TABLE II. COMPRESSION RATIO OF IMAGES

File Name	File Size	Compression by Huffman algorithm (in KB)	Compression by LZW algorithm (in KB)	Compressed and saved in TIFF (in KB)	Compression by Proposed Method (in KB)
Lenna.ppm	1392KB	623.6KB (55.2%)	519.2KB (62.4%)	383KB (72.48%)	256.1KB (81.6%)
Peppers.ppm	2684KB	1162.2KB (56.7%)	958.2KB (64.3%)	775.3KB (71.1%)	518KB (80.7%)
Mandrill.ppm	1389KB	627.8KB (54.8%)	501.4KB (63.9%)	380KB (72.6%)	245.8KB (82.3%)
Airplane.ppm	3037KB	1333.2KB (56.1%)	1072.1KB (64.7%)	810KB (73.3%)	498.1KB (83.6%)
Jet.ppm	1675KB	740.3KB (55.8%)	616.4KB (63.2%)	436KB (73.9%)	291.4KB (82.6%)
Barbara.ppm	1317KB	597.9KB (54.6%)	467.5KB (64.5%)	365KB (72.2%)	239.6KB (81.8%)
Parrot.ppm	168KB	73.6KB (56.2%)	60.8KB (63.8%)	47.2KB (71.9%)	29.3KB (82.6%)

AUTHORS PROFILE

Mr. Debashis Chakraborty received the B.Sc. degree with Honours in Computer Science and M.Sc. degree in Computer and Information Science from the University of Calcutta, West Bengal, India, in 2001 and 2003, respectively. He obtained the M.Tech. degree in Computer Science and Engineering, from Vinayaka Mission University, Salem, Tamilnadu, India, in 2007. At present he is pursuing his Ph.D. in the field of Data and Image Compression from University of Calcutta, West Bengal, India.

Mr. Chakraborty is a Lecturer in the department of Computer Science and Engineering, st. Thomas' College of Engineering and Technology, Kolkata, West Bengal, India. He has authored or co-authored over 6 conference papers in area of Data and Image Compression.

Soumik Banerjee, Debashish Chakraborty, "Fast Near-lossless Image Compression with Tree Coding having Predictable Output Compression Size", International Conference on Advances in Information Technology and Mobile Communication – AIM 2011, April 21-22, 2011, Nagpur, Maharashtra, India, July 9-11, 2010.

Debashish Chakraborty, Amiya Halder "An Efficient Lossless Image Compression Using Special Character Replacement", IEEE ICCET-2010, 13-14 November pp E-62 – E-67, Jodhpur, Rajasthan, India.

Debashis Chakraborty, Sandipan Bera, Anil Kumar Gupta and Soujit Mondal., "Efficient Data Compression using Character Replacement through Generated Code", IEEE NCETACS 2011, Shillong, India, March 4-5, 2011, pp 31-34.