An Image Compression Algorithm for Discontinuous Series of Similar Pixels

Kshitiz Agarwal
Jaypee Inst. of Info. Tech.
804, Sai Kirpa Appartment, sector-11, Vasundhara,
Ghaziabad, India

Karm Veer Arya
AVB-Indian Inst. of Info. Tech.& Management
Morena Link road, Gwalior, India

ABSTRACT
This paper represents lossy compression technique which can be effectively used for correlated pixels in both continuous and discontinuous series. Generally, compression is achieved by reducing redundancy through removal of similar pixels. Here, dependency of effective compression over their continuity has been removed. The efficiency of the proposed method has been described mathematically, proving it suitable for higher bit images.

Keywords
Compression Ratio, Run length encoding.

1. INTRODUCTION

In computers now days, use of graphics has become common and is used almost everywhere. Thus, images are of important aspects. But due to their increased sizes, importance of compression techniques also stands with them. They need to be compressed so as they will occupy less space and can be sent to other locations through mail, etc easily. An image is nothing but a group of pixels with each pixel carrying a value which shows color, brightness and other information of that position. Depending upon the pixel values, images can be of various types like 1 bit image or binary image, 8 bit image or grayscale image, 24 bit image or RGB image, etc. Nature of image depends upon pixel values in a series. If pixel values vary gradually in a row of pixels, then it is a continuous tone image and if there is a sharp or zero difference of values between two consecutive pixels at many positions, then it is discrete tone image like in case of a chart or a cartoon, etc [7]. This approach involves pairing of image pixels which is done to shift the pixels into new positions effectively. This would help in putting up the similar pixels in series. Various techniques can be developed to place pixels in various positions so as to get maximum number of correlated pixels series. Thereby, efficient compression can be achieved for their discontinuous row.

Rest of the paper is organized as follows. Section II shows various works related to the prescribed problem. Section III explains the complete methodology which is then followed by conclusion in the last section.

2. RELATED WORK

Run length encoding (RLE) is a common and simple compression technique. Here, a series of correlated pixels is replaced with a similar pixel followed by their number, done by encoder [9]. The reverse of this process is done by decoder. RLE technique is useful in simple graphic images like icons, line drawings, animations.

RLE has many applications. Like its use can be visualized in object recognition and edge detection [1]. Various approaches have been developed so far. Parallel algorithms have been implemented over RLE [2] where the input stream is divided into a number of equal parts, each sent to encoder for RLE compression. Another approach given by J. Trein is to send an image into blocks of pixels in parallel [3] whose output is a sequence of runs describing start and end position of similar pixels. A method has been developed to optimize code of this technique [4] where compression factor is inversely proportional to the average length of each run. Efficient RLE schemes have been derived by Hatsukazu which involves mapping fixed-length blocks of source output into variable length codewords [5]. Chengjjie Tu represented adaptive RLE technique where the coded output is encoded further using context based adaptive binary arithmetic coding [6] to maximize compression efficiency.

The main problem that RLE carries is its dependency over an image to be continuous in nature. It mainly relies over the situation for pixels to be similar in a run for better compression. If the length is small, efficiency would definitely decrease. This algorithm has been developed to deal with this problem. It does not rely over the described situation for better compression. It tends to reduce the gap between similar pixels by shifting those pixels using various techniques, making it capable of providing efficient compression for discontinuous series as well.

3. THE PROPOSED METHOD

In this paper, pixels are grouped into pairs, each pair carrying 2 consecutive pixels. Original pairs have been recognized as base pairs which form other pairs by swapping their pixels. Let S be

the set of any 2 base pairs or a pair along with a pixel (depending upon the number of pixels N) such that intersection of any 2 sets

results no pair. Selection of pairs in S involves fixed number of other pairs between them (varying from 0 to N/2 -2) which may leave few pairs incapable of forming S. In S, first pixel of second pair or the pixel itself can be swapped either with first pixel of the first pair (E) or with its second pixel (I). Thus, formation of S along with I or E denote a particular swapping technique, marked by a unique character or small string. All these techniques are noted in a notepad.

Now, all the base pairs are read in row-major fashion, starting from top-left. Count the number of pairs with similar pixels. Apply all the noted techniques in base pairs and do the same thing. The situation with maximum number of pairs carrying similar pixels is observed and applied. Now, from every pair while raster scanning, one of the 2 pixels is stored in an array, say R_1, and 1 bit in another array R_2 if they are similar. If they are not, both of them are stored in the same sequence in R_1 and 0 bit is stored in R_2. The last unpaired pixel (if N is odd) is also stored in R_1 along with 0 bit in R_2. When scanning completes, the string, if any, corresponding to the applied technique is stored in R_2 which is pushed in a stack and R_1 is shifted to new array R_3.

The procedure is repeated for all pixels in R_3 by taking new R_2 and the cycle is then continued until and unless very less number of pairs with similar pixels are observed. So, output would be R_3 along with a stack of R_2.

For decompression, pixels of R_3 and R_2 (pulled from the stack) are scanned simultaneously. Every pixel in R_3 is doubled if its corresponding bit in R_2 is 1. Otherwise the pixel next to its following pixel and the next bit are observed. This would expand R_3. When R_3 completes, the swapping technique signified by the string stored in R_2 is applied in it. This cycle is continued for the whole stack. Thus, the original image (R_3) is retrieved efficiently.

Theorem 1. The compression ratio (CR) decreases with increase in number of pairs with similar pixels.

Proof: Consider a B bit image N number of pixels. Then for every cycle,

S_1 = Size of input file = BN.

Now, number of base pairs n = (N/2 + N%2).

Let n_1 and M be the number of pairs with similar pixels and length of the string respectively. For large value of N, N/2 >> N%2. Hence, n = N/2. Then size of output stream would be

Size of R_2 = n + 8M.

Size of R_3 = n_1*B + (n- n_1)2B.

S_2 = Output stream size = n (1+2B) - n_1*B + 8M.

Compression ratio = (n (1+2B) + 8M – n1*B)/ (BN), where N > M. N, n, M and B are constants. So, CR decreases as n_1 increases.

Theorem 2. To apply compression over a given series of pixels, n_1 > (N/2 + 8M)/ B.

Proof: From theorem 1,

(1+2B) N/2 + 8M – n_1*B < BN

Hence, n_1 > (N/2 + 8M)/ B, where N/2 > 8M.

As, the maximum value of n_1 can be N/2 only, N/2 >= n_1 > (N/2 + 8M)/B

This theorem is not valid for binary images. The efficiency of this algorithm can be realized but is not suitable for binary images.

Theorem 3. Higher the bit depth of an image better would be the compression obtained.

Proof: From theorem 1 and 2, for every cycle, CR would lie between

1 > CR >= (1/2B + 0.5 + 8M/BN)

For binary images, B = 1. So, minimum value of CR is 1 + 8M/N. As CR is greater than 1, the technique is not suitable for binary images. It is also a bit time consuming process, though time consumed is for better efficiency only.

1. 4. CONCLUSION

Compression algorithm for images has been shown which can provide good compression for correlated pixels in any series. Various theorems have been then discussed to derive out the range of CR, showing its dependency over bit depth. Thus it has been concluded that the proposed method can be effectively used for higher bit images.

5. REFERENCES

[1] Christopher H. Messom, Gourab Sen Gupta and Serge N. Demidenko "Hough Transform Run Length Encoding for Real Time Image Processing" IEEE transactions on instrumentation and measurement, vol. 56, no. 3, June 2007.

[2] Nikolay Manchev "Parallel algorithm for Run Length Encoding", Proceedings of third international conference on information theory, 2006.

[3] J. Trein, A.Th.Schwarzbacher, B. Hoppe and K.-H. Noffz "A Hardware Implementation of a Run Length Encoding Compression Algorithm with Parallel Inputs", ISSC 2008, Galway, June 18-19.

[4] C.E Shannon and D.A Huffman "Optimizing a Scheme for Run-Length Encoding", Proceedings of IEEE, January 1969.

[5] Hatsukazu Tanaka and Alberto leon- Garcia "Efficient Run-Length Encodings", IEEE Transactions on Information Theory, vol. it-28, no. 6, November 1982.

[6] Chengjie Tu, Jie Liang and Trac D. Tran "Adaptive Runlength Coding", IEEE ICIP 2002.

[7] Salomon.(2004). Data compression Complete Reference, 3ed. New York: Springer.

[8] Bell, T. C., I. H. Witten, and J. G. Cleary (1990) *Text Compression*, Englewood Cliffs, NJ, Prentice-Hall.

[9] Golomb, Solomon W. (1966) "Run-Length Encodings," *IEEE Transactions on Information Theory* **IT-12**(3):399–401.