ISSN 2229-5518

An Improved Approach for Spatial Domain Lossless Image Data Compression Method by Reducing Overhead Bits

Mahmud Hasan, Kamruddin Md. Nur

Abstract— Lossless image compression techniques are used in digital imaging where large amount of data is to be stored without compromising the image quality. The volume of data that can be compressed using lossless image compression schemes is usually much lesser than that of its lossy compression counterparts. Yet, however, lossless compression algorithms are popular in a number of particular image data storage sectors. To meet the increasing demand of large amount of high quality image data storing, numerous algorithms were developed during last few decades featuring lossless image compression and covering various aspects of data compression approaches. Spatial domain lossless image compression methods are popular in most respects since their computational time is comparatively much lesser. In this paper, we focus on a spatial domain image compression technique that uses simple arithmetic operations in order to achieve the specified goal. We revealed that the mentioned algorithm is not always as advantageous as other spatial domain compression systems and often suffers from overhead transmission of unnecessary image data. The thorough investigation over the technique is reported along with the discovered mathematical bound at which the algorithm of interest is failed to achieve the desired target. Finally, to reduce the overhead obtained as a result of algorithmic trouble, an improved mechanism is suggested so that both the transmission time and storage space requirements using this method is facilitated.

Index Terms— Bits Per Pixel (BPP), Block Matrix, Block Processing, Computational Overhead, Inter-Pixel Redundancy, Run Length Coding, Spatial Domain Lossless Image Compression.

1 Introduction

RECENT digital imaging applications have observed copious invention in the field of image compression as the importance of preserving image data is being important day by day. Diverse ideas regarding this issue have been developed, still we suffer from choosing a suitable compression method for industry applications as computational cost of the compression stuff matters. A few standards like JPEG and JPEG-2000 are being used in today's industry applications where achieved compression ratio is important than its relative computational cost [1]. However, due to quality-compression trade-off, these standards fail to provide users with the most desirable image compression criterion- higher compression ratio with higher quality assurance [2].

Lossless image compression techniques, on the other hand, provides us with mentionable compression ratio and unaffected image quality. Such compression methods, that use simple arithmetic calculations in geometric domain or spatial domain [3,4,5], reduce the computational complexity too in a notable extent. Thus, spatial domain lossless image compre - ssion techniques deserve acute significance in digital imaging world. All of such algorithms attempt to reduce the inter-pixel redundancy of an image discovering the fact- since the value of any given pixel can be reasonably predicted from the value of its neighbors, the information carried by individual pixels is relatively small [6].

In this paper, our prime focus is on a spatial domain lossless compression algorithm by Syed & Mehdi [7]. This algorithm performs lossless compression, although, often, it has to suffer from a large unnecessary amount of bits due to computational overhead. We investigated the reason and suggested necessary modification required to improve the algorithm. Comparative results have also been taken into account.

2 Related Study

In digital image compression terminologies, overhead bits refer to the extra amount of bits required by a specific algorithm to compress an image [8]. For example, let us suppose, we have an image compression algorithm that can reduce 500 bits from an image of 1000 bits. It is then regarded that the compression ratio achieved by this algorithm is (1000÷500) for this particular image. Again, using the same algorithm, if another image of 800 bits results in 1000 bits after compression, there presents 200 overhead bits. It is possible since a good number of compression algorithms keep some non-image-information about the image in order to reduce its Bits Per Pixel (bpp). Whenever this non-image-information along with compressed-image-information becomes larger than original-image-information, an overhead occurs. Consequently, an image of n bits needs to be represented by m bits after compression; where m>n.

Preserving non-image-information in order to reduce total

- *Mahmud Hasan is currently teaching at the Dept. of Computer Science & Engineering as a full-time lecturer in Stamford University Bangladesh. E-mail: hasanpoet@gmail.com*
- *Kamruddin Md. Nur is currently teaching at the Dept. of Computer Science & Engineering as Assistant Professor in Stamford University Bangladesh. E-mail: kamruddin.nur@gmail.com*

original-image-information is an age-old technique [9]. A commonly known algorithm for image data compression is Run Length Encoding (RLE) where a stream of same gray level pixels are encoded as (x,y) representing x as pixel-run and y as gray value [2,6]. In this case, x is not an original-image-information, rather, it is non-image-information to finally achieve compressed-image-information. A number of image compression algorithms are developed using such concepts as in [10, 11, 12]. There is another compression algorithm school that does not directly use RLE and preserve non-image-information. Rather, it preserves some non-image-information regarding a local m×n block so that this information helps decide the exact pixel value during decoding [7]. The algorithm we are going to investigate is categorized into this class of algorithms.

3 Focused Algorithm

The algorithm we are examining has been developed by Syed & Mehdi [7]. It requires an image to be divided into a number of m×n blocks where the standard value of m and n is 4. The specific application has freedom to choose m and n other than 4. The algorithm then looks for the maximum and minimum pixel values MAX and MIN within this m×n block and calculates MAX-MIN. The block header preserves 8-bits MIN and a 3-bits coding-information that tells how many bits are required to represent MAX-MIN. Then MIN is subtracted from all pixel values of m×n block and each is encoded by k bits, where k is the number denoted by 3-bits coding-information. Figure 1 illustrates the focused algorithm. The embedding and extraction procedure as given by Syed & Mehdi [7] is shown in the following subsections.

24 34 35 27 64 54 50 53 0 0 0 0 0 0 0 0	MAX-MIN of the Block: 64 24 (8 bits)+7 (3 bits)+ all pixels (16×7 bits) Total Bits Required = 123

Fig. 1. Example Block of Focused Algorithm

3.1 Encoding Steps

The encoding steps of the algorithm proposed by Syed & Mehdi [7] are shown below –

Step1: Select m and n for whole image.

Step2: Take m×n non-overlapping block of image.

Step3: Find the difference of Min and Max value in selected m×n block in X.

Step4: Add 11 bits header (8 bits for Min value of block, and 3 bits dedicated the no. of bits required to represent X value' in Y bits).

Step5: Subtract each pixel from Min value of a block and store in separate Y bits of every pixel in new m×n block.

3.2 Decoding Steps

The decoding steps of the algorithm proposed by Syed & Mehdi [7] are shown below –

Step1: Parse the header and find out block size m and n.

Step2: Find the Min (8 bit) value for each block.

Step3: Parse another 3 bits which represent the no. of (Y) bits required for each pixel value.

Step4: Read next Y bits, add its value to Min and regenerate the actual value of pixel. Repeat this step for all pix - els in a block.

Step5: Repeat the above steps for whole image and regenerate the original image.

4 Overhead Analysis

It should now be clear that according to our focused algorithm any m×n block of an image contains a header of 11 bits, where MIN consists of 8 bits and coding-information comprises the left 3 bits. If these 3 bits denote 100_2 (i.e. 4_{10}), then every pixel of this block is encoded using 4 bits. This technique works efficiently as long as the coding-information remains less than 8 bits. But let us consider a situation where MAX-MIN results in an integer to represent which at least 8 bits are required. Then a typical 4×4 block has to be embedded as 11 bits+16×8 bits, whereas the non-compressed-block was embedded by only 16×8 bits. This situation is possible whenever MAX-MIN≥128. Figure 2 shows a practical phenomenon where such occurrence is illustrated.

156 148 154 159 130 135 140 146 28 34 32 36 21 15 18 18	MAX-MIN of the Block: 144 15 (8 bits)+8 (3 bits)+ all pixels (16×8 bits) Total Bits Required = 139 Overhead Bits = 139-128 = 11

Fig. 2. An Example of Overhead-Block

Although the spatial smoothness of an image is common, it is not guaranteed that at least two pixels of an m×n block cannot differ by a factor of 128 or more. Rather, it happens very frequently. Statistical evidence shows that for gray scale

images, there are at least 5.76% 4×4 blocks where MAX-MIN≥128. For color images, however, the percentage is less-only 2.10%. Whatever the statistical percentage of overhead blocks, surely, for each overhead block, the focused algorithm needs to preserve some extra bits.

5 IMPROVED SUGGESTION

With a view to modifying our focused algorithm, we propose to keep some information regarding the overhead blocks and leave those blocks without embedding. Considering a standard block size 4×4 for 512×512 dimensional images, we find 16,384 blocks that can be treated as a 128×128 dimensional matrix as shown in Figure 3.

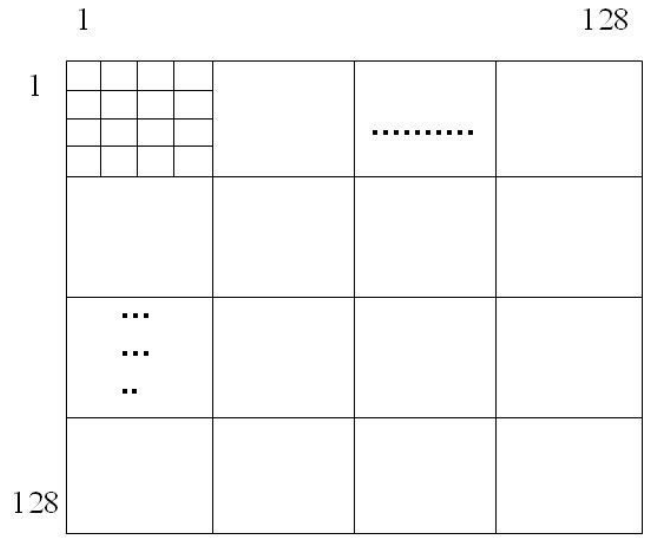

Fig. 3. An 128×128 Matrix, each element is a 4×4 Pixel Block

We require each row of this 128×128 matrix now starts with a 128 bit binary sequence where each 1 indicates an overhead block at that position. Since the overhead blocks are not embedded, 11 bits from each overhead block can be discarded. The CODEC ought now to use a trace variable that will keep checking the 128 bit block-row-header and whenever it finds a 1 in that row-header, it supposes no 11-bits block header for block of that position. For example, if 70th bit of 128 bits block-row-header contains a 1, for 70th block, the decoder does not look for 11-bits block header.

The improved encoding steps are organized as follows –

Step1: Prepend a 128 bit extra header in front of each block-row, all bits are reset.

Step2: Take a m×n non-overlapping block of image as done in focused algorithm (standard size of m and n is 4).

Step3: Find the difference of Min and Max value in selected m×n block in X.

Step4: If Max-Min≥128 i.e. overhead block, set the corresponding bit in 128 bit header. Keep no 11 bit block-header.

Step5: Subtract each pixel from Min value of a block and store in separate Y bits of every pixel in new m×n block.

The improved decoding steps are organized as follows –

Step1: Read first 128 bits, find which are overhead blocks.

Step2: Except the overhead blocks, take 11 bits block-header and follow the decoding steps described in section 3.2.

The comparative performance in the next section of this paper statistically proves that using 128 bits at the beginning of each block-matrix reduces total image information more than the focused algorithm does. However, these 128 bits can be run-length encoded if necessary.

6 COMPARATIVE PERFORMANCE ANALYSIS

TABLE 1
RESULT OF OVERHEAD BLOCK CALCULATION

Test Image	Total 4×4 Blocks	Overhead Blocks	Total Overhead Bits
Baboon	16384	1926	21186
Lena	16384	3319	36509
Cameraman	16384	6598	72578
Iris	16384	2360	25060

A study over 12,82,048 blocks of 4×4 dimension shows that 73,819 blocks cause overheads. In other words, there are 5.76% blocks for which using 11 bits header of the focused algorithm is meaningless. These 11 bits are redundant for each overhead-block resulting in n×11 bits that are non-image-information, where n is the number of overhead-blocks. Table 1 shows a portion of our study for some famous test images.

TABLE 2
RESULT OF OVERHEAD BIT REDUCTION BY PROPOSED MODIFICATION

Test Image	Overhead Bits by Focused Algorithm	Overhead Bits by Proposed Modification	Overhead Bits Reduced
Baboon	21186	16384	4802
Lena	36509	16384	20125
Cameraman	72578	16384	56194
Iris	25060	16384	8676

Our implementation does not preserve 11 bits header for the overhead-blocks as done in focused algorithm. Instead, after dividing a 512×512 image into 128×128 block-matrix, it uses 128 bit long overhead-block-information at the beginning of each of 128 rows. Thus, our improved technique has to use 128×128 bits or 16384 bits instead of 21186, 36509, 72578 and 25060 as overhead bits for the images Baboon, Lena, Cameraman and Iris respectively. A portion of our obtained results is given in Table 2.

As the proposed modification can reduce the overhead bits by a notable extent, the total number of bits after compression by the proposed algorithm is less than that of obtained by focused algorithm. Table 3 shows another comparative study.

TABLE 3
COMPARATIVE PERFORMANCE ANALYSIS

Test Image	Total Bits	Total Bits After Compression by Focused Algorithm	Total Bits After Compression by Proposed Modification
Baboon	2097152	1797568	1792766
Lena	2097152	1282528	1262403
Cameraman	2097152	1181344	1125150

Therefore, if an 8-bit image of 512×512 dimension results in x bits (x≤2097152) after being compressed by the focused algorithm, our statistical evidence proves that, still, x bits contain p overhead bits, where p is statistically greater than 16384. Thus, preserving 16384 bits instead of p bits results in more compressed image data. However, as 16384 bits required by our modification is kept in 128 different parts, each part can be run length coded as mentioned before.

7 CONCLUSION

In this paper, we investigated a novel image compression technique propsed by Syed & Mehdi [7] and found redundant bits inherently preserved by their technique. Then we suggested an improvement over their algorithm which results in more compression ratio as discussed in our comparative performance analysis. Moreover, further research can be conducted in order to reduce the total number of compressed bits by applying run-length coding on the extra 128 bit binary information suggested by our improvement.

REFERENCES

[1] Gregory K. Wallace, "The JPEG Still Picture Compression Standard", *IEEE Transactions on Consumer Electronics*, 1991.

[2] Ralf Steinmetz and Klara Nahrstedt, "Multimedia: Computing, Communications and Applications", 1st Edition, Pearson Education Inc. ISBN: 81-7808-319-1, 2005.

[3] Suneetha Agarwal and Seshagiri Gurram, "Image Compression using Simple Arithmetic Operations", *International Conference on Computational Intelligence and Multimedia Applications (ICCIMA)*, India, pp. 58-62, 2007.

[4] Sunil Kumar Pattanik, K. K. Mahapatra and G. Panda, "A Novel Lossless Image Compression Algorithm using Arithmetic Modulo Operation", *IEEE International Conference on Cybernetics & Intelligence Systems (CIS) and Robotics Automation & Mechatronics (RAM) (CIS-RAM 2006)*, Thailand, pp. 234-238, 2006.

[5] Komal Ramteke and Sunita Rawat, "Lossless Image Compression LOCO-R Algorithm for 16 bit Image", *2nd National Conference on Information and Communication Technology (NCICT)*, pp. 11-14, 2011.

[6] Rafael C. Gonzalez and Richard E. Woods, "Digital Image Processing", 2nd Edition, Pearson Prentice Hall. ISBN: 81-7758-168-6, 2005.

[7] Syed Ali Hassan and Mehdi Hussain, "Spatial Domain Lossless Image Data Compression Method", *International Conference of Information and Communication Technologies*, 2011.

[8] Tinku Acharya and Ajoy K. Ray, "Digital Image Processing: Principles and Applications", John Wiley & Sons, Inc. ISBN: 10 0-471-71998-6, 2005.

[9] M. Nelson and J. L. Gailly, "The Data Compression Book", 2nd ed. New York: M & T Books, 1996.

[10] Al-Wahaib and M. S. KokSheikh Wong, "A Lossless Image Compression Algorithm Using Duplication Run Length Coding", *IEEE Conference on Network Application Protocols and Services*, pp. 245-250, 2010.

[11] J P Cookson and G R Thoma, "X-Ray Image Compression using Run Length Coding", *Journal of Medical Systems*, Volume: 12, Issue: 4, pp. 201-209, 1988.

[12] Samir Kumar Bandyopadhyay, Tuhin Utsab Paul and Avishek Raychoudhury, "Image Compression using Approximate Matching and Run Length", *International Journal of Advanced Computer Science and Applications (IJACA)*, Volume: 2, Issue: 6, pp. 117-121, 2011.s