Compression of Satellite Images Using Lossy and Lossless Coding Techniques

S.Chandravadhana [#1], N.Nithiyanandam [*2]

[#] Electronics and Communication, B.S.Abdur Rahman University
Vandalur, Chennai, Tamilnadu, India.
[1] chandra.vadhu@gamil.com
[*] B.S.Abdur Rahman University
Vandalur, Chennai, Tamilnadu, India.
[2] mail_nithi@yahoo.com

Abstract— **Block truncation coding is one of the simplest compression techniques where the image is divided into blocks and then processed. The traditional method involves computation of a high mean and the low mean to replace the original pixel values. Here the first and the second moments are preserved and the bit rate is a constant value (2 bits per pixel). The major disadvantage is that there are heavy blocking artifacts when the block size increases. Many methods have been evolved in order to improve the compression ratio and also to reduce the bit rate. In this paper an improved block truncation coding algorithm along with an adaptive lossless compression scheme is proposed to improve on the compression ratio and Peak signal to noise ratio. The computational complexity is also further reduced and the blocking artifacts which are inherent in the traditional BTC are also minimized to a great extent.**

Keyword- Block Truncation coding, blocking artifacts, Compression ratio, Peak signal to noise ratio, Bit rate.

I. INTRODUCTION

Block Truncation Coding (BTC) was first proposed by Delp and Mitchell in 1979[1] for lossy image compression. The basic concept of this technique is to divide the original image into non-overlapping blocks, each of which is represented by two different and distinct values. In traditional BTC, the two values also preserve the first- and second-moment characteristics of the original block. When a BTC image is transmitted, the mean and the variance values and the bitmap which stores the arrangement of the two values in each block are required to compute the high and low mean. Although BTC cannot provide high coding gain as other lossy compression techniques, such as JPEG or JPEG2000, the complexity of BTC is much lower than that of these modern techniques, which makes it feasible for less powerful processing kernel, such as ARM and FPGA based applications. In the literature, many approaches have been proposed to improve BTC. One such category involves preserving the moment characteristic of the original image. Halverson. [2] generalized a family of moment-preserving quantizers, by employing moments higher than three. Udpikar and Raina [3] proposed a modified BTC algorithm, which preserves only the first-order moment. The algorithm is optimum in the mean-square sense, and it is also convenient for hardware implementation.

Another category involves improving the image quality and reducing the blocking effect. Kanafani [4] decomposed the image into homogeneous and nonhomogeneous blocks and then compressed them using BTC or vector quantization (VQ). This block classification is achieved by image segmentation using the expectation-maximization (EM) algorithm. The new EM-BTC-VQ algorithm can significantly improve the quality and fidelity of compressed images when compared with BTC or VQ. A problem of BTC is its poor image quality under high compression ratio and low bit rate condition, and some studies have attempted to address this issue. Kamel [5] proposed two modifications on BTC. The first one allows the partitioning of the image into variable block sizes rather than a fixed size. The second modification involves the use of an optimal threshold to quantize the blocks by minimizing the mean square error. Chen and Liu [6] proposed a visual pattern block truncation coding (VPBTC), in which the bitmap is employed to compute the block gradient orientation and match the block pattern.

Another method is the classification of blocks according to the properties of human visual perception. However, most of the improvements described previously increase the complexity substantially. In this work, a new formula is proposed which produces good visual quality for the images and also satisfactory compression ratio.

II. TRADITIONAL BLOCK TRUNCATION CODING

In the traditional BTC the original image is divided into equal size blocks such as 4x4, 8x8, 16x16 and 32x32. The mean, second moment and the variance are calculated using the formula shown in the equations below.

$$\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{1}$$

$$\overline{x^2} = \frac{1}{m}\sum_{i=1}^{m} x_i^2 \tag{2}$$

$$\overline{\sigma}^2 = \left(\overline{x^2} - (\overline{x})^2\right) \tag{3}$$

where $\overline{\sigma}^2$ is the square of the standard deviation and x_i denotes the gray scale value of the original image. The concept of the BTC is to preserve the first and second -moment characteristics of a block when the original block is substituted by its quantization levels. Thus, the following two conditions are maintained:

$$m\bar{x} = (m-q)a + qb \tag{4}$$

$$m\overline{x^2} = (m-q)a^2 + qb^2 \tag{5}$$

where $m = M \times N$, and q denotes the number of pixels with values greater than $\bar{x}$. The low and high mean of the Block Truncated image can be calculated as follows:

$$a = \bar{x} - \sigma\sqrt{\frac{q}{m-q}} \tag{6}$$

$$b = \bar{x} + \sigma\sqrt{\frac{m-q}{q}} \tag{7}$$

where a and b denote the low-mean and high-mean, respectively. Here, m is the total number of pixels and q is the number of pixels having intensity values greater than the threshold. Since BTC is a one-bit quantizer, the mean is employed to threshold the block. The digitized (binary values) result is called bitmap, which is used for recording the arrangement of the two represented values, low-mean and high-mean.

$$y_{i,j} = \begin{cases} b, & if\ h_{i,j} = 1, \\ a, & if\ h_{i,j} = 0, \end{cases} \quad where \quad h_{i,j} = \begin{cases} 1, & if\ x_{i,j} \geq \bar{x} \\ 0, & if\ x_{i,j} < \bar{x} \end{cases} \tag{8}$$

Where $h_{i,j}$ denotes the binary bitmap values of a block and the symbol $y_{i,j}$ denotes the pixel elements of the two tone compressed BTC. Since BTC exhibits annoying blocking artifacts when the block size is increased, an improved form of BTC is proposed, which is described in section III.

III.IMPROVED BLOCK TRUNCATION CODING

In this method the mean and variance of each block is calculated and the high mean and low mean of the blocks are calculated according to the following formulae:

$$a = \bar{x} + \bar{\sigma}\sqrt{\frac{k+m}{q}} \tag{9}$$

$$b = \bar{x} - \bar{\sigma}\sqrt{\frac{k+m}{q}} \tag{10}$$

Where *k* is the block size and varies accordingly.

This method replaces the pixels in the block with *a* and *b* as the high mean and the low mean of each block. The values of *a* and *b* are of high variation so that the images have sharp edges unlike the traditional BTC. The size of the block adaptively plays an important role in determining the compression ratio. As the block size increases the compression ratio also increases as shown in Table I and Table II. The blocking artifacts which are present in the traditional BTC are reduced when the block size increases to 16x16, 32x32, and 64x64.

IV. IMPROVED BLOCK TRUNCATION CODING AND ADAPTIVE LOSSLESS ALGORITHM

The image data is then transferred to the pre-processor which splits the data into blocks of size starting from 4, 8,16,32,64. The bit rate for different block sizes are 2 bpp, 1.125 bpp, 1.031 bpp, 1.007 bpp, 1.001 bpp respectively.

The blocks are then adaptively compressed using various lossless compression [7] techniques. The technique which gives the highest compression ratio is adopted for transmission. The various lossless algorithms are Improved Golomb coding, Run length coding and Huffman coding. Fig 1 shows the block processing of the images.

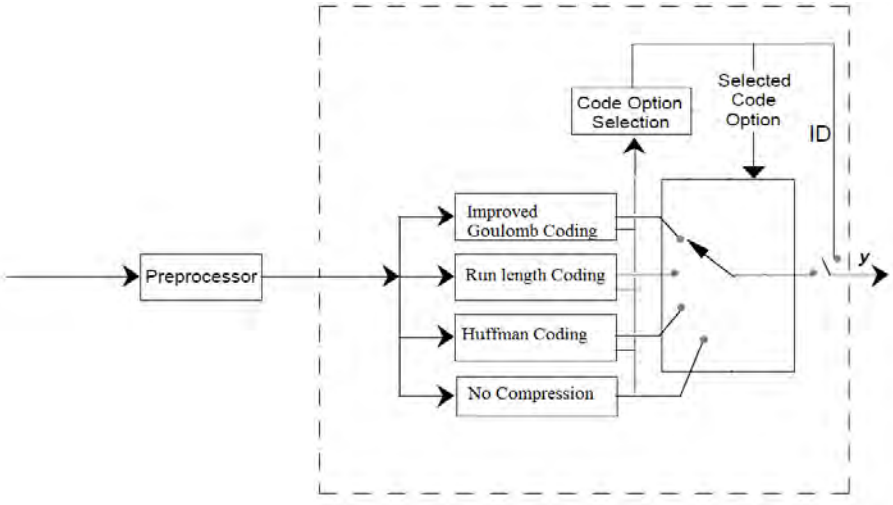


Fig. 1. Adaptive lossless coding

The inputs to the proprocessor are $x_1,x_2,x_3,\ldots x_n$ image blocks of equal size and the output of the preprocessor is δ_1, δ_2, δ_3,….. δ_n are the blocks of data with identification. *Y* is the output of the compressed image. The different types of codings are described below.

V. OPTIMUM GOLOMB CODING

Golomb codes [12] are lossless compression codes when the sequence of uniform data is small unlike large amount of data. Here, it suits well for the binary block truncated codes since the image is divided into blocks and processed. Non Negative integers following a geometric pattern will have a Golomb code as an optimal prefix code, making Golomb coding[7] highly suitable for situations in which the occurrence of small values in the input stream is significantly more likely than large values. This code uses a tunable parameter *M* to divide an input value into two parts:*q*, the result of a division by *M*, and r, the remainder. The quotient is sent in unary coding, followed by the remainder in truncated binary encoding. When *M*=1 Golomb coding is equivalent to unary coding.

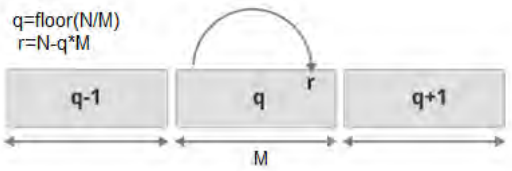


Fig 2: Golomb Code

Golomb codes are those which can be thought of as codes that indicate a number by the position of the *bin* (*q*), and the *offset* within the *bin* (*r*). The above Fig 2 shows the position *q*, and offset *r* for the encoding of integer *N* using Golomb parameter *M*. The following steps show how Golomb code is done. Here instead of choosing the parameter M arbitrarily, it is fixed for optimization and speed.

1. Take any parameter M and fix it as 10. This is an optimum value for fast processing, unlike the usual code which takes any value for M.
2. Keeping N as the integer to be encoded, find the following:
    1. quotient = q = int [N/M]
    2. remainder = r = N modulo M
3. Generate the Code word
    1. The Code format : <Quotient Code><Remainder Code>, where
    2. Quotient Code :

       Write a q-length string of 1 bits Write a 0 bit
    3. Remainder Code :

       Taking M as 10, set $b = log_2(M)$

       If $r < 2^b - M$ code r as plain binary using b-1 bits.

       If $r \geq 2^b - M$ code the number $r + 2^b - M$ in plain binary representation using b bits.

The blocks of high and low mean are given to the Golomb Coder and the output streams of bits are obtained according to the algorithm shown above. The compression ratio is calculated for the reconstructed image in the receiver.

VI. RUN LENGTH CODING

In each block of the image the block is converted to its bit plane. The mean is employed to threshold the block. The digitized (binary values) result is called bitmap, which is used for recording the arrangement of the two represented values, high-mean and low-mean. Run-length encoding (RLE) is a form of data compression technique in which *runs* of data (that is, sequences in which the same data value occurs in many consecutive data elements) are stored as a single data value and count, rather than the original run. This is most useful on data that contains many such runs: for example, simple graphic images such as icons, line drawings, and animations. Here, since we convert the image into its bit plane, the long runs of zeros or ones can be represented by the data value and its count. Note that the data value here is a binary number.

A raster scan and a zig zag scan of the binary values are done and the total runs of 0s and 1s are represented by the number of runs and the binary value.

An alternate method of run length coding maps a run of length r to a prefix of length k (k-1 ones terminated by zero) and a tail of length k (all combinations of k bits), if and only if $\mathbf{2^k - 2 \leq r < 2^{k+1} - 2}$ holds, as shown in Fig 3. This alternate method of run length encoding reduces the number of bits required to be transmitted thereby saving channel bandwidth and also the time of transmission of the data. Table I shows the code for the runs of zeros and ones for integer numbers. Following the procedure in Table I, we can frame the code words for any number of integers.

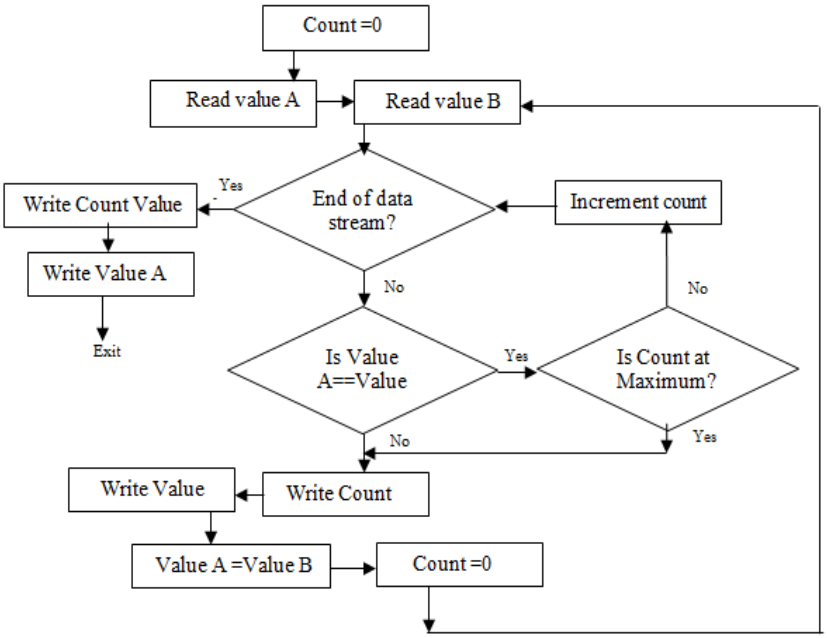


Fig 3: Flow diagram of run length encoding

TABLE I
Code Table for Alternate Method of Run Length Encoding

Run-length r	k	Prefix	Tail	Code word
1 2	1	0	0 1	00 01
3 4 5 6	2	10	00 01 10 11	1000 1001 1010 1011
7 8 9 10 : :	3	110	000 001 010 100	110000 110001 110010 110100

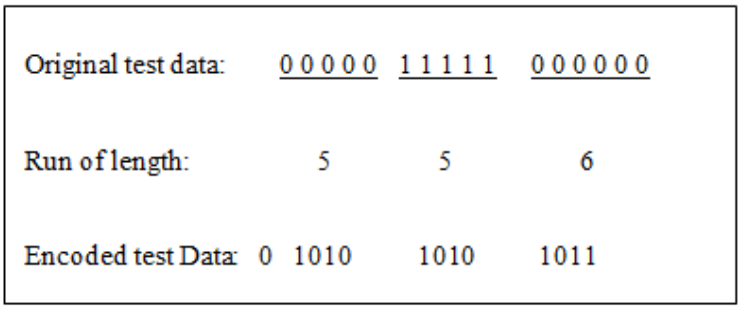


Fig 4: Alternate method of run length encoded data

In this way we can code 16 bits of data to only 13 bits and thereby achieve lossless compression as shown in Figure 4.The optimal scan is taken and used for representation. As the block size increases only very few overheads are required to represent the whole block. Run length coding is a lossless coding [9] suitable for images which have a uniform toning in most parts of the image. Weather images generally have uniform toning in most parts of the image. Also the zig zag scanning improves on the continuous runs of zeros or ones.

VII. HUFFMAN CODING

The compression ratios are compared for the other two methods and only if they are lower than the lossless[11] huffman coding compression, this coding is used. Since the images consist of two tone blocks, the huffman coder waits for all the blocks to enter and codes it by assigning bits to each value. The following are the steps for Huffman Coding.

a) Allow all the blocks to enter the Huffman coder from the Preprocessor. Let the inputs to the huffman coder be the quantized values of the BTC coder, $x_1, x_2, \ldots\ldots x_n$ . .

b) Considering an image of size 1024x1024, the total number of values to be encoded for a 4x4, 8x8, 16x16, 32x32, 64x64 block sizes are 512, 256, 128, 64 and 32 respectively.

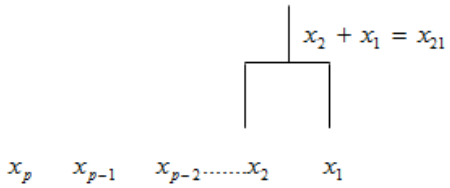
c) Calculate the probability of occurrence of each pixel in the image by using the formula shown in (11).

$$Probability = \frac{Number\ of\ occurences\ of\ the\ pixels}{Total\ number\ of\ pixel\ values\ to\ be\ encoded} \quad (11)$$

d) Arrange the probability of the quantized pixel values in descending order,

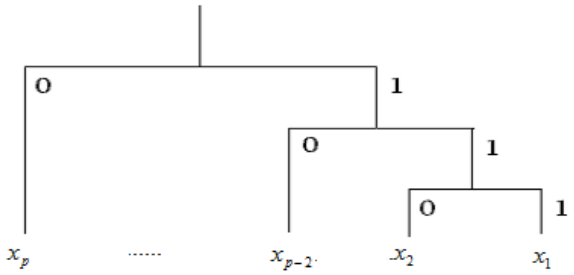
$$x_p, x_{p-1}, x_{p-2}, \ldots\ldots\ldots\ldots x_1$$

e) Select the two lowest pixel values (x), grouped together and sum them. This begins the construction of a "Huffman Tree" structure.



f) Continue the process until the Huffman Tree is completed. The combined probability value should be placed again in the descending order.

g) Each branch of the tree is labeled with either a zero or 1. This way the complete huffman tree is labeled in order to get the variable code length of the quantized pixel values.



h) Tracing down the huffman tree gives the lossless[10] huffman code. This code is a compressed code since variable length is used.

VIII. EXPERIMENTAL RESULTS

In this paper the improvement and hence the novelty made in the compression ratio is that instead of sending the bit plane of the blocks, the mean and the variance in the traditional BTC, the two tone blocks of the improved BTC are sent to the adaptive coder for further compression.

The images for compression are taken from aerial Satellite, launched from United States of America, which reports the weather conditions and aerial view of the continent. The algorithm works with Water Vapour,Visible and Infra Red Images of the satellite. Satisfactory compression ratio and Peak Signal to noise ratio is achieved from improved BTC as shown in the above table.

They are then passed on to the preprocessor for further improving on the compression ratio. The various algorithms for lossless compression are executed and the optimal result is taken. According to the

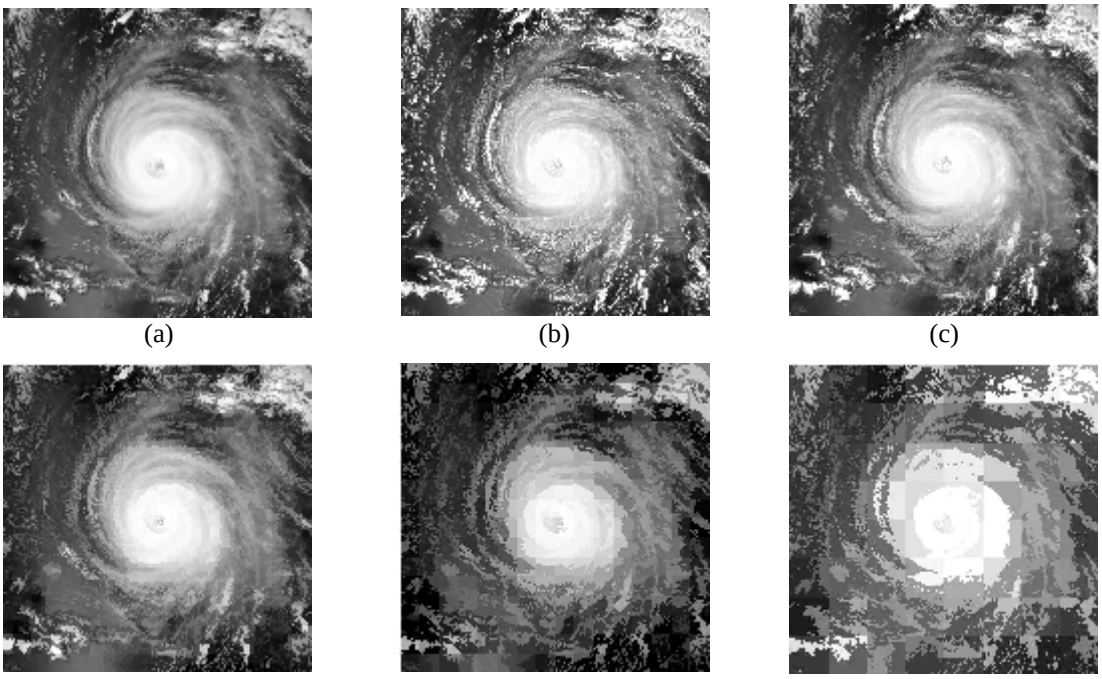
recommendations of the CCSDS (Consulate Committee for Space Data Syatems) a lossy compression along with a lossless compression is employed for satellite on board compression.
The compression ratio(CR) for an image is calculated as represented in (12).

$$CR = \frac{Size\ of\ the\ original\ file}{Size\ of\ the\ compressed\ file} \tag{12}$$

Here, since we are transmitting the digitized image, the size of the file is represented in bits. The quality of an image is measured using various parameters such as Mean Square error (MSE), Peak Signal to Noise Ratio (PSNR), Mean Absolute error (MAE) etc. In this paper the Peak Signal to Noise Ratio is calculated as shown in (13).

$$PSNR = 10\ log_{10}\left[\frac{P\ x\ Q\ X\ 255^2}{\sum_{i=1}^{P}\sum_{j=1}^{Q}\left[\sum_{m,n\epsilon R}\sum_{w_{m,n}}\left(g_{i+m,j+n} - h_{i+m,j+n}\right)\right]^2}\right] \tag{13}$$

The results show that the PSNR values have improved compared to the BTC, when the proposed algorithm is applied to the images. The compression ratio is also satisfactory compared to the Traditional BTC. The results for the CR and PSNR of the traditional BTC for Image 1 is depicted in Table II.



(a) (b) (c)

(d) (e) (f)

Fig 5**:** Image 1: (a) Original Image, (b) – (f) Traditional BTC for block size 4x4, 8x8, 16x16, 32x32, 64x64

TABLE III
Compression Results Using Traditional BTC for Image 1

Block Size	Compression Ratio (CR)	Peak Signal to Noise Ratio (PSNR)
4 x 4	23. 9411	33.45
8 x 8	24.5213	33.51
16 x 16	25.2312	33.72
32 x 32	25.9241	33.98
64 x 64	26.2134	34.21

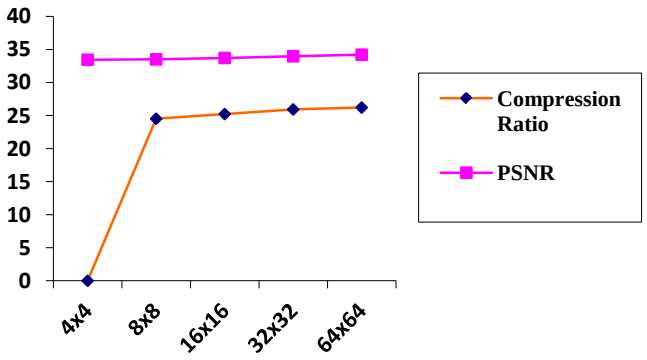


Fig 6: Graph showing the CR and PSNR of Traditional BTC for Image 1.

Fig 7 shows the implementation of improved BTC for Image 2.The graph of the CR and PSNR of the improved BTC for image 2 is also shown in fig 8. We can infer from the graph that the image quality has improved by applying the new and improved BTC. The CR is also satisfactory so that the memory space in the processor is saved.

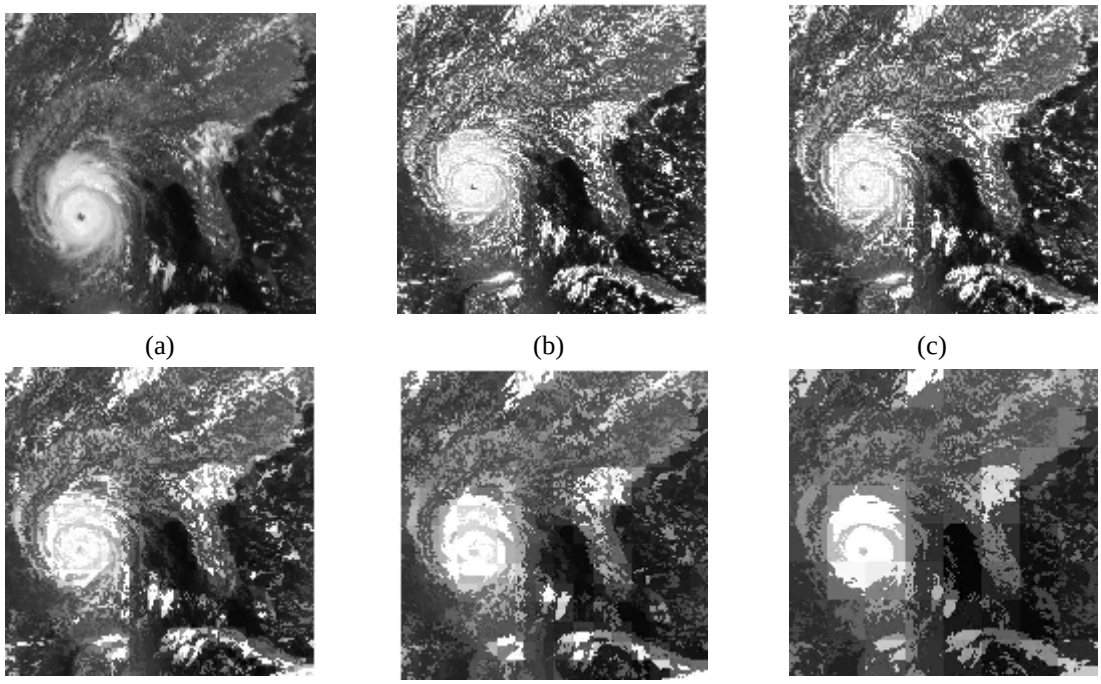


Fig 7: Image 2(Water Vapour) (a) Original Image, (b) – (f) improved BTC with block size 4x4, 8x8, 16x16, 32x32, 64x64.

TABLE IIIII
Compression Results Using Improved BTC for Image 2

Block Size	Compression Ratio (CR)	Peak Signal to Noise Ratio (PSNR)
4 x 4	18.2231	37.65
8 x 8	19.1021	37.54
16 x 16	19.5433	37.98
32 x 32	22.2543	38.54
64 x 64	26.1321	39.67

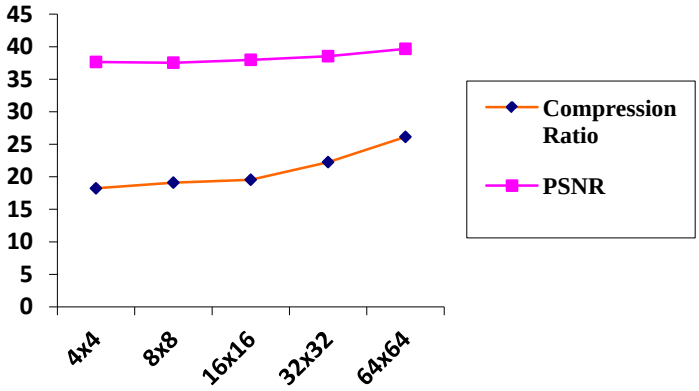


Fig 8**:** Graph showing CR and PSNR for Image 2 with improved BTC

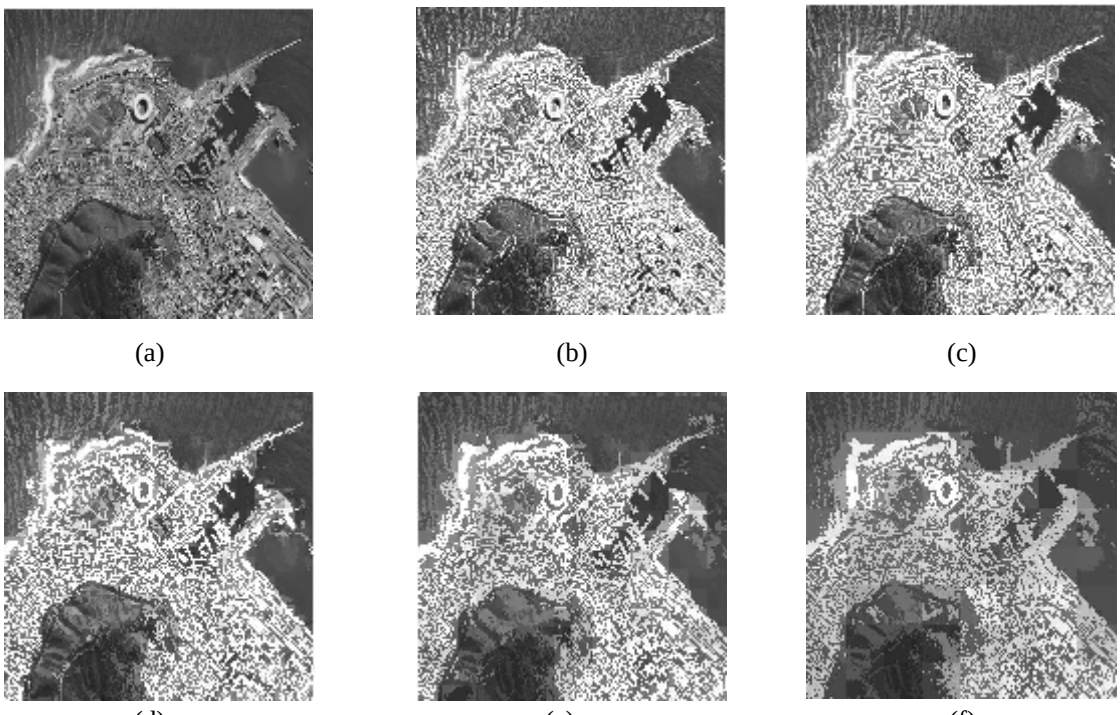


Fig 9: Image 3(Infra-Red) (a) Original Image, (b) – (f) improved BTC with block size 4x4, 8x8, 16x16, 32x32, 64x64.

TABLE IVV
Compression Results using improved BTC for Image 3

Block Size	Compression Ratio (CR)	Peak Signal to Noise Ratio (PSNR)
4 x 4	18.1945	37.56
8 x 8	19.2533	37.63
16 x 16	19.5245	37.87
32 x 32	21.2891	38.66
64 x 64	25.1782	39.65

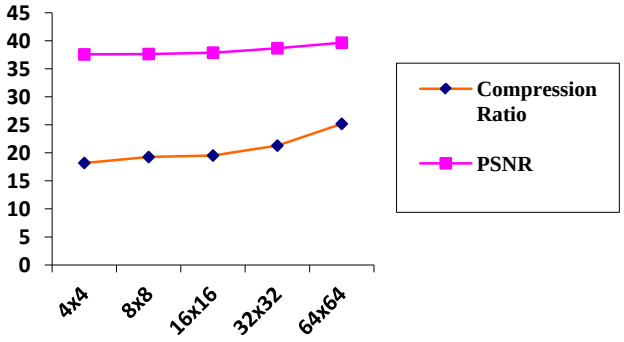


Fig 10**:** Graph showing CR and PSNR for Image 3 with improved BTC

IX. CONCLUSION

The traditional BTC produced severe blocking artifacts when the block size increased for higher compression ratio. Here an improved Block Truncation coding is proposed which substantially reduces the blocking effects when the block size increases as shown in Fig 7 and Fig 9. Also higher compression ratios are obtained for higher block sizes. This lossy compression method is combined with a lossless compression technique for further reducing the bit rate of the image. Results show that the new algorithm of BTC substantially improves the PSNR and satisfactory compression ratio is obtained. Three sample aerial satellite images are taken. The regions of hurricane formation can be clearly demarcated for the images. This algorithm is used for water vapour, Infra-Red and Visible images of the satellite.

ACKNOWLEDGMENT

Authors would like to acknowledge their deep sense of gratitude to all those who supported us for their valuable guidance and for their constant support to carry out this work.

REFERENCES

[1] E. J. Delp and O. R. Mitchell, “Image compression using block truncation coding,” IEEE Trans. Commun., vol. 27, no. 9, pp. 1335–1342, Sep. 1979.
[2] D. R. Halverson, N. C. Griswold, and G. L. Wise, “A generalized block truncation coding algorithm for image compression,” IEEE Trans. Acoust., Speech, Signal Process., vol. ASSP–32, no. 3, pp.664–668, Jun. 1984.
[3] V. Udpikar and J. P. Raina, “Modified algorithm for block truncation coding of monochrome images,” Electron. Letters., vol. 21, no. 20, pp. 900–902, Sep. 1985.
[4] Q. Kanafani, A. Beghdadi, and C. Fookes, “Segmentation-based image compression using BTC-VQ technique,” in Proc. IEEE Int. Conf. Information Science Signal Processing and their Applications, Paris, Jul. 1–4, 2003, vol. 1, pp. 113–116.
[5] M. Kamel, C. T. Sun, and G. Lian, “Image compression by variable block truncation coding with optimal threshold,” IEEE Trans. Signal Process., vol. 39, no. 1, pp. 208–212, Jan. 1991.
[6] L. G. Chen and Y. C. Liu, “A high quality MC-BTC codec for video signal processing,” IEEE Trans. Circuits Syst. Video Technol., vol. 4, no. 1, pp. 92–98, Feb. 1994.
[7] S. Derin Babacan, and Khalid Sayood, “Lossless Hyperspectral-Image Compression Using Context-Based Conditional Average Hongqiang Wang, IEEE Trans: Geoscience and Rmote Sensing, VOL. 45, NO. 12, DECEMBER 2007.
[8] Lei Wang, Jiaji Wu, Licheng Jiao, Li Zhang and Guangming Shi Key ,”Lossy to Lossless image compression based on reversible integer DCT, Laboratory of Intelligent Perception and Image Understanding” of Ministry of Education of China, Institute of Intelligent Information Processing, Xidian University, Xi'an 710071, P.R. China978-1-4244-1764-3/08/$25.00 ©2008 IEEE.
[9] Z. Pan et al, A Lossless Compression Algorithm for SAR Amplitude Imagery Base on Modified Quadtree Coding of Bit Plane, IEEE geoscience and remote sensing letters, Vol.7, No.4, 2010, pp. 723-726.

[10] N. Senthilkumaran, and Dr. J. Suguna," Neural Network Technique for Lossless Image Compression Using X-Ray Images", International Journal of Computer and Electrical Engineering, Vol. 3, No. 2, April, 2011, pp - 1793-8163.
[11] Guo.Z., "Mode- Dependent templates and scan order for H.264/AVC-Based Intra Lossless Coding", IEEE Transactions on Image Processing ,2012,Vol 21,Issue 9, PP-4106-4116.
[12] Din.J,Wei.W,"Adaptive Golomb code for Joint Geometrically Distributed data and its applications in image coding",IEEE Transactions on circuits and systems for Video Technology", Issue 99,PP-1., August 2012.