International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-2 Issue-6, May 2013

Absolute Moment Block Truncation Coding For Color Image Compression

D.Harihara Santosh, U.V.S. Sitarama Varma, K.S.K Chaitanya Varma, Meena Jami, V.V.N.S Dileep

Abstract— In this paper Color image data compression using absolute moment block truncation coding scheme (AMBTC) is implemented. This compression technique reduces the computational complexity and achieves the optaimum minimum mean square error and PSNR. It is an improvised version of BTC, obtained by preserving absolute moments. AMBTC is an encoding technique that preserves the spatial details of digital images while achieving a reasonable compression ratio. The simulation results obtained indicate that both the computational complexity of and the reconstructed image quality obtained using AMBTC algorithm are better than those obtainable with other existing BTC algorithms.

Index Terms—Absolute moment block truncation coding, computational complexity.

I. INTRODUCTION

Image data compression is concerned with minimization of the number of information carrying units used to represent an image. It known that raw digital image occupy a large amount of memory, the amount of memory required creates problems in massive digital image storage for image archiving in a variety of applications [1]. The image compression techniques are categorized into two main classifications namely lossy compression techniques and Lossless compression techniques [2].

Lossy data compression is named for what it does. After one applies lossy data compression to a message, the message can never be recovered exactly as it was before when compressed. When the compressed message is decoded it does not give back the original message. As lossy compression cannot be decoded to yield the exact original message, it is not a good method of compression for critical data, such as textual data. It is most useful for Digitally Sampled Analog Data (DSAD) as it consists mostly of sound, video, graphics, or picture files.

Lossless data compression is also named for what it does. In a lossless data compression file the original message can be exactly decoded. Lossless data compression works by finding repeated patterns in a message and encoding those patterns in an efficient manner.

For this reason, lossless data compression is also referred to as redundancy reduction. Because redundancy reduction is dependent on patterns in the message; it does not work well on random messages. Lossless data compression is ideal for text [3].

Manuscript published on 30 May 2013.
*****Correspondence Author(s)
Mr. D. Hari Hara Santosh obtained his B. Tech. and M. Tech degrees from JNT University, Hyderabad, India.
Mr. U.V.S Sitarama Varma, He is working as Assistant Professor in MVGR College of Engineering, Vizianagaram , AP.India.
Mr. Sri Krishna Chaitanya Varma is currently pursuing his B. Tech. in MVGR College of Engg., JNT University, Kakinada, Andhra Pradesh.India.
Miss Meena Jami, is currently pursuing his B. Tech. in MVGR College of Engg., JNT University, Kakinada, Andhra Pradesh.India.
Mr. V.V.N.S. Dileep is currently pursuing his B. Tech. in MVGR College of Engg., JNT University, Kakinada, Andhra Pradesh.India.

II. BLOCK TRUNCATION CODING

Block truncation coding (BTC) is a simple and fast lossy compression technique for digitized gray scale images. The key idea of BTC is to perform moment preserving (MP) quantization for blocks of pixels so that the quality of the image will remain acceptable and at the same time the demand for the storage space will decrease.

It is a simple and fast algorithm which achieves constant bit rate of 2.0 bits per pixel. The method is however suboptimal. It divides the original images into small sub-images and then using a quantizer, which adapts itself according to the image statistics, to reduce the number of gray. In block truncation coding an image is firstly segmented into n x n blocks of pixels. In the most cases the size is 4 x 4, but we can choose other size as 8 x 8. Secondly, a two level output is chosen for every block and bitmap is also encoded using 1 bit for every pixel[4].

A. Compression Procedure

The image is divided into blocks of typically 4x4 pixels. For each block the Mean and Standard Deviation are calculated, these values change from block to block. These two values define what values the reconstructed or new block will have, in other words the blocks of the BTC compressed image will all have the same mean and standard deviation of the original image [5].

$$\overline{\mathbf{X}} = \frac{\mathbf{1}}{\mathbf{n}} \sum_{\mathbf{i=0}}^{\mathbf{n}} \mathbf{Xi}$$

$$\boldsymbol{\sigma} = \sqrt{\frac{\mathbf{1}}{\mathbf{n}} \sum_{\mathbf{i=1}}^{\mathbf{n}} (\mathbf{xi} - \overline{\mathbf{xi}})\,\mathbf{2}}$$

Where x_i represents the i^{th} pixel value of the image block and n is the total number of pixels in the block. The two values $\overline{x}$ and σ are termed as quantizers of BTC. A two level quantization on the block is where we gain the compression, it is performed as follows:

$$Y(i, j) = 1, \quad x(i, j) > \overline{\mathbf{X}}$$
$$= 0, \quad x(i, j) \leq \overline{\mathbf{X}}$$

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

Where x (i, j) are pixel elements of the original block and y (i, j) are elements of the compressed block. In words this can be explained as: If a pixel value is greater than the mean it is assigned the value "1", otherwise "0". Values equal to the mean can have either a "1" or a "0" depending on the preference of the person implementing the algorithm. By this process each block is reduced to a bit plane. The bit plane along with $\overline{x}$ and σ forms the compressed data. For example a block of 4 x 4 pixels will give a 32 bit compressed data, amounting to 2 bpp.

This 16 bit block is stored or transmitted along with the values of Mean and Standard Deviation. Reconstruction at the decoding side is made with two values "H" and "L" which preserve the mean and the standard deviation. The values of "H" and "L" can be computed as:

$$H = \overline{X} + \sigma \sqrt{\frac{p}{q}}$$

$$L = \overline{X} - \sigma \sqrt{\frac{q}{p}}$$

. Where p and q are the number of 0's and 1's in the compressed bit plane respectively. To reconstruct the image, or create its approximation, elements assigned a 0 are replaced with the "L" value and elements assigned a 1 are replaced with the "H" value. This demonstrates that the algorithm is asymmetric in that the encoder has much more work to do than the decoder. This is because the decoder is simply replacing 1's and 0's with the estimated value whereas the encoder is also required to calculate the mean, standard deviation and the two values to use[6].

$$X\ (i, j) = L,\ y\ (i, j) = 0$$

$$= H,\ y\ (i, j) = 1$$

III. ABSOLUTE MOMENT BLOCK TRUNCATION CODING

In this technique image compression is done using absolute moment block truncation coding. It is an improved version of BTC, preserves absolute moments rather than standard moments, here also a digitized image is divided into blocks of n x n pixels. Each block is quantized in such a way that each resulting block has the same sample mean and the same sample first absolute central moment of each original block[7].

A. Image Compression Using AMBTC

AMBTC has been extensively used in signal compression because of its simple computation and better mean squared error (MSE) performance. It has the advantages of preserving single pixel and edges having low computational complexity. The original algorithm preserves the block mean and the block standard deviation. Other choices of the moments result either in less MSE or less computational complexity.

In AMBTC algorithm similar to BTC there are four separate steps while coding a single block of size n x n. They are quad tree segmentation, bit plane omission, bit plane coding using 32 visual patterns and interpolative bit plane coding[8].

In this work we have compressed the image using AMBTC algorithm.

- Original image is segmented into blocks of size 4x4 or 8x8 for processing.
- Find the mean of each block also the mean of the higher range and lower range.
- Based on these means calculated for each block, bit plane is calculated.
- Image block is reconstructed by replacing the 1's with higher mean and 0's with lower mean.

At the encoder side an image is divided into non-overlapping blocks. The size of each non overlapping block may be (4 x 4) or (8 x 8), etc. Calculate the average gray level of the block (4x4) as (for easy understanding we made the compression for 4 x 4 block. The same procedure is followed for block of any size):

$$\overline{\mathbf{X}} = \frac{1}{16} \sum_{i=1}^{16} \mathbf{X}_i$$

Where X_i represents the i^{th} pixel in the block. Pixels in the image block are then classified into two ranges of values. The upper range is those gray levels which are greater than the block average gray level ($\overline{x}$) and the remaining brought into the lower range. The mean of higher range x_H and the lower range x_L are calculated as:

$$X_H = \frac{1}{k} \sum_{Xi \geq mean}^{n} \boldsymbol{X}_i$$

$$X_L = \frac{1}{k} \sum_{Xi < mean}^{n} \boldsymbol{X}_i$$

Where k is the number of pixels whose gray level is greater than $\overline{x}$. A binary block, denoted by b, is also used to represent the pixels. We can use "1" to represent a pixel whose gray level is greater than or equal to $\overline{x}$ and "0" to represent a pixel whose gray level is less than $\overline{x}$. The encoder writes $\boldsymbol{x}_H$, $\boldsymbol{x}_L$ and $\boldsymbol{x}_b$ to a file.

In the decoder, an image block is reconstructed by replacing the `1's with XH and the '0"s by XL In the AMBTC, we need 16 bits to code the bit plane which is same as in the BTC. But, AMBTC requires less computation than BTC. But the result remains the same[9].

IV. QUAD TREE SEGMENTATION

The quad tree segmentation technique divides the given image in to set of variable sized blocks using a threshold value. Here we use the absolute difference between the higher mean and the lower mean of AMBTC method as controlling value to segment the given image [10].

A. Steps For Quad Tree Segmentation

- Divide the image into non overlapping blocks, suppose the block size to be m_1 x m_1.
- For each block, if the absolute difference between the higher mean and the lower mean is less than mean then apply AMBTC else divide the block into 4 sub images.
- Repeat the above step until the condition gets satisfied.
- Check the above two steps for every block in the image.

The image is divided as shown below depending on the condition that is taken into consideration.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

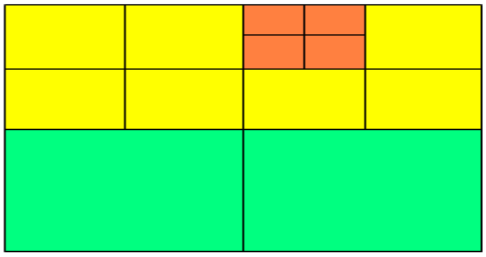


Figure 1: Green color represents 16 x 16 block.
Yellow color represents 8 x 8 block.
Orange color represents 4 x 4 block

V. BIT PLANE OMISSION

In bit plane omission technique, the bit plane obtained in an AMBTC method is omitted in the encoding procedure if the absolute difference between the higher mean $\overline{X}_H$ and the lower mean $\overline{X}_L$ of that block is less than a threshold value only the block mean retained. At the time of decoding the bit plane omitted blocks are replaced by a block filled with the block mean.

The two quantization levels (η_H, η_L) of an image block of 4x4 pixels, can be represented in (1, 0) bit plane, requiring 16 bits. To reduce the storage space, this bit plane can be removed if the two quantized levels (η_H, η_L) differ by a small amount.

$\varepsilon = |\eta_H, - \eta_L| < Th$, where *Th* is an arbitrary constant. In this case, only the mean value of the pixels η is to be preserved to represent the whole block. When h is represented in 8 bits, the storage bits required for this block is only 8 instead of the usual 32 bits[11].

A. *Steps For Bit Plane Omission*

- Get the segmented block from the quad tree segmentation.
- Calculate the higher mean and the lower mean.
- The block that is having the absolute difference between the means greater than threshold is only retained, otherwise the block is omitted.
- At the time decoding omitted blocks are replaced by a block filled with the block mean.

VI. BIT PLANE CODING

A bit plane of a digital discrete signal (such as image or sound) is a set of bits having the same position in the respective binary numbers. For example, for 16-bit data representation there are 16 bit-planes: the first bit-plane contains the set of the most significant bit and the 16th contains the least significant bit.

It is possible to see that the first bit-plane gives the roughest but the most critical approximation of values of a medium, and the higher the number of the bit plane, the less is its contribution to the final stage. Thus, adding bit-plane gives a better approximation[12].

A. *Steps For Bit plane Coding*

- Let I be an image where every pixel value is n-bit long.
- Express every pixel in binary using n bits.
- Form, out of I n binary matrices (called bit planes), where ith matrix consists of the ith bits of the pixels of I.

Example: Let I be the following 2 x 2 image where the pixels are 3 bits long.

101	110
111	011

The corresponding three bit planes are:

1	1
1	0

0	1
1	1

1	0
1	1

Bit plane coding: code the bit planes separately, using RLE (flatten each plane row-wise into a 1D array), Golomb coding, or any other lossy compression technique.

VII. INTERPOLATIVE TECHNIQUE

When there is a correlation among neighboring pixels, there is a correlation among the neighboring bits in the bit plane also. Based on this idea, the alternate bits in the AMBTC bit plane are dropped and the remaining 8 bits are preserved. The dropped bits are then predicted using neighboring bits. In this technique half (8 bits) of the number of the bits in the bit plane of AMBTC is dropped at the time of encoding as in Figure shown. In decoding phase the dropped bits are recovered by taking the arithmetic mean of the adjacent pixel values as given in equations below [13].

1	5	9	13
2	6	10	14
3	7	11	15
5	8	12	16

Figure 2: The pattern of dropping bits. The bold faced bits are dropped

The arithmetic means of the dropped bits are calculated so as to reconstruct the image during decoding as:

$$\hat{x}_i = \frac{1}{3}(x_{i-1} + x_{i+1} + x_{i+4}) \text{ for } i = 2$$

$$\hat{x}_i = \frac{1}{2}(x_{i-1} + x_{i+4}) \text{ for } i = 4$$

$$\hat{x}_i = \frac{1}{3}(x_{i-4} + x_{i+1} + x_{i+4}) \text{ for } i = 5$$

$$\hat{x}_i = \frac{1}{4}(x_{i-4} + x_{i-1} + x_{i+1} + x_{i+4}) \text{ for } i = 7,10$$

$$\hat{x}_i = \frac{1}{3}(x_{i-4} + x_{i-1} + x_{i+4}) \text{ for } i = 12$$

$$\hat{x}_i = \frac{1}{2}(x_{i-4} + x_{i+1}) \text{ for } i = 13$$

$$\hat{x}_i = \frac{1}{3}(x_{i-1} + x_{i+1} + x_{i-4}) \text{ for } i = 15$$

VIII. ENCODING STEPS

Step 1: Divide the given image into a set of non overlapping blocks, say x of size n = 16 x 16 pixels.

Step 2: Compute the block mean$\overline{x}$, lower mean $\overline{x}$L and higher mean $\overline{x}$H for a block.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

Step 3: Fix a threshold value Th1 and if | $\overline{x}$H - $\overline{x}$L |<= Th1encode the block x with the block mean, put '0' as a indicator bit as prefix code for decoding purpose and go to step 14 else continue to step 4.
Step 4: Divide the 16 x 16 block into four non overlapping blocks (xb) of size n = 8 x 8 pixels.
Step 5: Compute the block mean $\overline{x}$b, lower mean $\overline{x}$bL and higher mean $\overline{x}$bH for 8 x 8 block.
Step 6: If | $\overline{x}$bH - $\overline{x}$bL |<= Th2 and encode the block xb with the block mean $\overline{x}$b, put '1' as a indicator bit as prefix code for decoding purpose and go to step 7 for the next block else go to step 8, Th2 is the second threshold value.
Step 7: Repeat the steps 5 and 6 until all the four sub blocks in this block are encoded. If all the blocks are encoded then go to step 14.
Step 8: Divide the 8 x 8 block into four non overlapping blocks (xC) of size n = 4 x 4 Pixels.
Step 9: Compute the block mean $\overline{x}$C, lower mean $\overline{x}$CL and higher mean $\overline{x}$CH for 4 x 4 blocks.
Step 10: If | $\overline{x}$CH - $\overline{x}$CL |<= Th3 and encode the block xC with the block mean $\overline{x}$C, put '01' as a indicator bit as prefix code for decoding purpose and go to step 11 for the next block else go to step 12. Th3 is the third threshold value.
Step 11: Repeat the steps 9 and 10 until all the four sub blocks in this block are encoded. If all the blocks are encoded then go to step 7.
Step 12: Construct the bit plane by taking '1' for the pixels with values greater than or equal to the mean c x and the rest of the pixels are presented by '0'. Encode the bit plane using 32 predefined bit plane pattern of 4 x 4 along with lower mean $\overline{x}$CL and higher mean $\overline{x}$CH, put '10' as a indicator bit as prefix code for decoding purpose and go to step 11. If the bit plane does not match with 32 visual pattern then go to step 11.
Step 13*:* Drop a pattern of bits as shown in Fig. 2 , encode the block by the remaining bits with the lower mean $\overline{x}_{CL}$ and higher mean $\overline{x}_{CH}$, put '11' as a indicator bit as prefix code for decoding purpose and go to step 11.
Step 14*:* Go to step 2 until all the blocks are processed.

IX. DECODING STEPS

Step 1: If the indicator flag is '0' replace the block *x* with the block mean$\overline{x}$.
Step 2: If the indicator flag is '1' replace the block x_b with the block mean $\overline{x}_b$.
Step 3: If the indicator flag is '01' replace the block x_C with the block mean $\overline{x}_C$.
Step 4: If the indicator flag is '10' Get the 32 visual pattern bit plane and decode the bit plane by replacing the 1s by and higher mean $\overline{x}_{CH}$ and the 0s by lower mean $\overline{x}_{CL}$
Step 5: If the indicator flag is '11' block x_C is reconstructed by replacing the 1's by $\overline{x}_{CH}$ and the 0's by $\overline{x}_{CL}$.

X. AMBTC TO COLOR IMAGES

Because of the tri chromatic nature of the human vision. we can assume that the color signal that we want to code lies in a three-dimensional space. Election of the coordinate system to represent this signal is often limited by practical situations. Because of the monitoring system available, we have used the NTSC receiver phosphor primary system in the implementation of color AMBTC. This system defines three primaries R, G, and B such that

$$O < R < 1 ; O < G < 1 ; \ O < B < 1.$$

The relationship between this group of primaries and other coordinate systems is given above .From this point we assume that our gamut of color is limited to their producible colors in the NTSC primary system. Taking into account the boundaries for R, G and B , the solid of all reproducible colors as is a cube such as in figure.

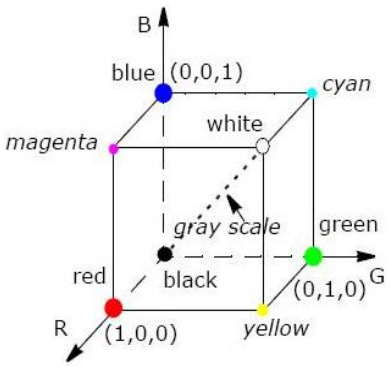


Figure 3: cube of NTSC colors in the R-G-B NTSC space of primitives

A. *Correlation Between Color Planes*

In typical color image data, there exists a high degree of correlation between the planes *R* , *G* , *and B.* It is well known that more efficient coding is achieved when the different output signals are decorrelated. It is therefore convenient to perform some decorrelating operation on the planes *R, G* and *B* before trying to apply AMBTC to the three signals. It is also desired to have an invertible transformation. The ideal decorrelation operation is a Karhunen - Loeve transform that would theoretically achieve total decorrelation between planes. However, such an approach leads *to* lengthy computations of eigen values and eigenvectors and depends on the image statistics. Thus, it is impractical. There exist other transformations that, although they do not achieve optimum results, are close to the optimum case - and are much simpler in practical implementation, We will make use' of one particular transformation. The NTSC *R-G-B* to *Y-I-Q* transformation. This is the method used in the NTSC color TV system. Also it has RGB component space energy compaction properties comparable *to* Karhunen-Loeve for most images[14].

The direct R-G-B to *Y -I-Q* transformation is given by

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.595716 & -0.274453 & -0.321263 \\ 0.211456 & -0.522591 & 0.311135 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

and the inverse Y-I-Q to R-G-B transformation is given by

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0.9563 & 0.6210 \\ 1 & -0.2721 & -0.6474 \\ 1 & -1.1070 & 1.7046 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$$

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

The range of the *Y*, I, and Q signals is

$$O \leq Y \leq 1$$

$$-0.596 \leq I \leq 0.596$$

$$-0.523 < Q < 0.523$$

B. AMBTC Color System

Suppose that we have three color planes *R*, G, *and B* of NTSC primaries for a given image. The goal is to transmit that image achieving compression and having a reconstructed image with an acceptable quality for the human observer. The proposed system diagram is shown in the diagram, the *R*, G, and *B* signals pass through a linear transformation to get the NTSC *Y* ,*I* and Q signals. In order to get a high compression and remembering that the human eye is more sensitive to spatial variations of *Y*, fairly sensitive to spatial variations of *I*, and less to Q, the *I* signal is sub sampled by taking averaging windows of *2* X 2 pixels and the Q signal is sub sampled by taking averaging windows of 4 X 4 pixels. Non integer values could have been used for this decimation; however, they are integers to maintain simplicity in the system. The outputs of this filtering stage are *Y, I, Q* which are coded with three separate AMBTC coders. At the receiver the signals are reconstructed using AMBTC quantizers. In order to keep the sizes correct, signals Y, *I*, Q are obtained by taking bilinear interpolations of two points for *I* and four points for Q in each direction (x and y). Finally, using linear transformation L^{-1} , the reconstructed planes *R, G, B* are obtained.

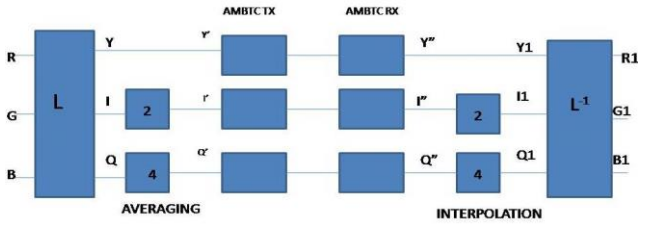


Figure 4: proposed AMBTC color system

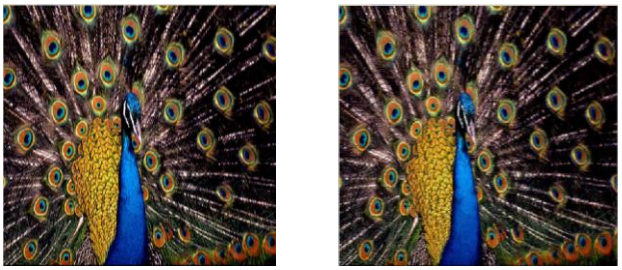
XI. RESULTS

We have to select three threshold values for bit plane omission technique at three different levels of quad tree segmentation for the block size of 16 x 16. As we are using an image of block size of 4 x 4, we have selected only one threshold value. We have applied different combination of threshold values on standard images of size 512 x 512.

From the below tables it is evident that the threshold value of 20 is suitable considering 3 parameters BPP, PSNR, Elapsed time.

To evaluate the performance of our method, we carried out experiments on nine images 5(a) peacock.jpg, 5(b) lena.jpg, 5(c) esb.jpg, 5(d) suv.jpg, 5(e) lighthouse.jpg, 5(f) rose.jpg, 5(g) building.jpg , 5(h) rangoli.jpg, 5(i) monalisa.jpg of size 512 X512 pixels.

We applied our method to each of the 9 images with different spatial and frequency characteristics. Characteristics of test images are evaluated in spatial domain using spatial frequency measure (SFM).

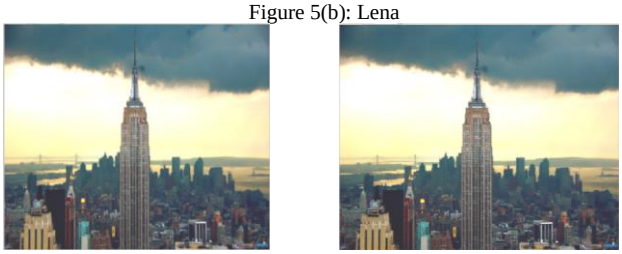


Original Decompressed

Figure 5(a): Peacock

Original Decompressed

Figure 5(b): Lena

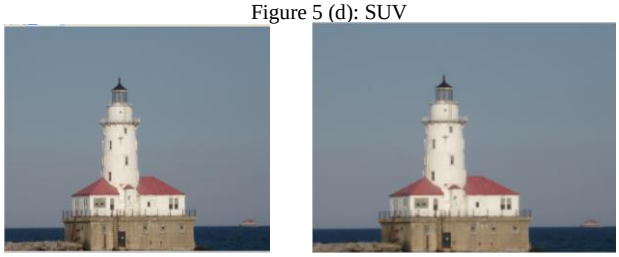


Original Decompressed

Figure 5 (c) :ESB

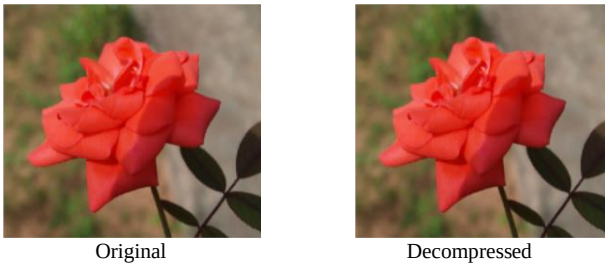
Original Decompressed

Figure 5 (d): SUV



Original Decompressed

Figure 5 (e):Lighthouse



Original Decompressed

Figure 5 (f): Rose

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

Original

Decompressed

Figure 5 (g): Building

Original

Decompressed

Figure 5 (h): Rangoli

Original

Decompressed

Figure 5 (i): Monalisa

Images	Threshold						Execution Time	Threshold						Execution Time
	10							15						
	BPP			PSNR				BPP			PSNR			
	R	G	B	R	G	B		R	G	B	R	G	B	
Peacock	1.2382	1.2438	1.2499	21.6131	21.7216	22.5641	15.51217	1.2023	1.2107	1.2093	21.6035	21.7126	22.5501	14.941133
Lena	0.8465	0.8368	0.809	32.6051	32.718	33.2677	9.517344	0.7635	0.758	0.7284	32.1965	32.3215	32.8115	5.669549
Empire State Building	0.8451	0.8285	0.8109	29.8729	30.3365	31.1942	6.472526	0.8451	0.8282	0.8109	29.8729	30.3365	31.1942	6.472526
SUV Car	1.1244	1.1159	1.1077	26.629	26.7958	26.8137	9.318946	1.0227	1.0071	1.0035	26.5246	26.6827	26.7071	8.307365
Lighthouse	0.6656	0.6676	0.6686	35.3654	35.1592	35.7365	7.155292	0.6448	0.6474	0.6409	35.2007	35.0027	35.4893	5.207648
Building	1.0176	1.0562	1.0402	26.2886	26.2237	25.2781	8.505621	0.9541	0.9946	0.9915	26.2291	26.1699	25.2462	7.929013
Rose	0.6361	0.6457	0.6617	37.5095	37.3762	37.2292	6.759203	0.6162	0.624	0.6258	37.2169	37.0671	36.7645	5.464256
Rangoli	0.9944	1.016	1.0305	26.9722	26.9065	26.8064	9.803747	0.8741	0.9248	0.9291	26.869	26.8262	26.7172	8.560656
Monalisa	0.8428	0.8093	0.7749	34.4309	34.789	36.0526	6.254997	0.7125	0.6917	0.6474	33.5301	33.9058	34.8522	5.065562

TABLE-EXPERIMENTAL VALUES FOR THRESHOLD 10

TABLE-EXPERIMENTAL VALUES FOR THRESHOLD 15

Images	Threshold						Execution Time	Threshold						Execution Time
	20							25						
	BPP			PSNR				BPP			PSNR			
	R	G	B	R	G	B		R	G	B	R	G	B	
Peacock	1.1703	1.1725	1.1592	21.5847	21.6908	22.5154	14.541926	1.1354	1.132	1.1033	21.592	21.6522	22.4491	13.645544
Lena	0.7134	0.7081	0.6876	31.7325	31.8492	32.3894	7.6834	0.6814	0.674	0.658	31.2921	31.3669	31.9275	40904547
Empire State Building	0.7589	0.7484	0.7224	29.6027	30.049	30.8483	5.643994	0.794	0.7859	0.7592	29.7553	30.2307	31.049	6.058577
SUV Car	0.794	0.7859	0.7592	29.7553	30.2307	31.049	6.058577	0.8477	0.8307	0.8291	26.1043	26.2515	26.277	6.631984
Lighthouse	0.6286	0.6316	0.624	34.9509	34.7728	35.2238	5.02152	0.617	0.6204	0.6103	34.6709	34.5235	34.8779	4.940087
Building	0.8995	0.9307	0.9489	26.1312	26.0578	25.1904	7.422714	0.8555	0.8781	0.9036	26.0045	25.9056	25.0907	7.039383
Rose	0.6057	0.6101	0.6098	36.9356	36.7176	36.3745	5.114398	0.5986	0.6029	0.5996	36.6444	36.4316	36.0061	5.222305
Rangoli	0.8059	0.852	0.8505	26.7579	0.267092	26.592	7.68013	0.7609	0.7934	0.7912	26.637	26.5522	26.4309	7.254403
Monalisa	0.6524	0.6383	0.5969	32.8252	33.2275	34.0996	4.495643	0.6204	0.6128	0.5791	32.279	32.7461	33.7021	4.254636

TABLE-EXPERIMENTAL VALUES FOR THRESHOLD 20

TABLE-EXPERIMENTAL VALUES FOR THRESHOLD 25

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

Image	Threshold						Execution Time
	30						
	bpp			PSNR			
	R	G	B	R	G	B	
Peacock	1.0935	1.0856	1.0424	21.4875	21.5842	22.3427	13.645544
Lena	0.6533	0.6496	0.6371	30.7634	30.9006	31.4838	4.672028
Empire State Building	0.794	0.7859	0.7592	29.7553	30.2307	31.049	6.058577
SUV Car	0.7903	0.7764	0.7749	25.8458	26.0021	26.0256	6.083373
Lighthouse	0.6076	0.6092	0.6002	34.3431	34.1581	34.508	4.809776
Building	0.8177	0.8312	0.8603	25.8455	25.7091	24.9437	6.598315
Rose	0.592	0.5943	0.5911	36.2602	35.9587	35.5783	5.167211
Rangoli	0.7321	0.7543	0.7555	26.5191	26.3885	26.2931	6.759346
Monalisa	0.6012	0.5975	0.5703	31.8372	32.348	33.4303	4.171176

Table-experimental values for threshold 30

XII. CONCLUSION

In this paper Color image data compression using absolute moment block truncation coding scheme (AMBTC) is presented. An improvement in BTC can be obtained by preserving absolute moments .This method is called absolute moment block truncation coding AMBTC .Both computational speed and reconstructed image quality are improved by preserving absolute moments instead of standard moments .The new method has the same general characteristics as BTC which include low storage requirements and extremely simple coding and decoding technique . A low computational complexity three stage gray scale image compression scheme using quad tree segmentation is presented .The experimental results show that compression efficiency is increased at the cost of image quality.

REFERENCES

1. Rafael C. Gonzalez ,Richard E. Woods Digital image compression 2nd edition
2. Doaa Mohammed, Fatma Abou-Chadi (Senior Member, IEEE.) Image "Compression Using Block Truncation Coding" ,journal in selected areas of telecommunication (JSAT),Ffebruary edition, 2011.
3. M. Ghanbari "Standard Codecs: Image Compression to Advanced
4. Video Coding" Institution Electrical Engineers , ISBN: 0852967101,
5. 2003 , CHM , 430 pages
6. Pasi Fränti, Olli Nevalainen and Timo Kaukoranta Department of Computer Science, University of Turku Compression of Digital Images by Block Truncation coding : a survey © *The Computer Journal*, 37 (4), 308-332, 1994
7. S. K.Kapde and Mr. S.V.Patil Image compression using BTC and AMBTC(IJARCSEE)Volume 1, Issue 10, December 2012
8. Chen-Kuei Yang, Member, IEEE, Ja-Chen Lin, and Wen-Hsiang Tsai
9. ,Color Image Compression by Moment-Preserving and Block Truncation Coding Techniques, IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. 45, NO. 12, DECEMBER 1997.
10. K.Somasundaram and I.Kaspar Raj ,An Image compression Scheme based on Predictive and Interpolative Absolute Moment Block Truncation Coding , GVIP Journal, Volume 6, Issue 4, December, 2006
11. K.Somasundaram and I.Kaspar Raj ,Low Computational Image Compression Scheme based on Absolute Moment Block Truncation Coding, World Academy of Science, Engineering and Technology 19 2006
12. G. Lu and T. L. Yew, "Image compression using quad tree partitioned iterated function systems," Electronic Letters, VOL. 30, NO.1, pp. 23-24, Jan. 1994.
13. Reversible Image Watermarking using Bit Plane Coding andLifting Wavelet Transform by S. Kurshid Jinna, Dr. L. Ganesan , IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.11, November 2009
14. Yu-Chen Hu, "Low complexity and low bit-rate image compression scheme based on Absolute Moment Block Truncation Coding", Vol. 42 No. 7 (2003) pp 1964-1975.
15. Wen Jan Chen, Shen Chuan Tai ,Bit rate reduction techniques for Absolute Block Truncation Coding, journal for Chinese institute of engineers ,vol. 22,no 5,pp661-667(1999)
16. Y V Ramana, and H Eswaran, "A new algorithm for BTC image bit plane coding," *IEEE Trans on Comm, 32*(10), 1984, pp. 1148-1157.
17. M.D.Lema , and O.R.Mitchell, "Absolute Moment Block Truncation Coding and its Application to Color images, " IEEE Trans. On Communications, Vol. 32, pp. 1148-1157,1984.

AUTHOR PROFILE

Mr. D. Hari Hara Santosh obtained his B. Tech. and M. Tech degrees from JNT University, Hyderabad in the year 2005 and 2010. Presently he is pursuing Ph.D, in Video Processing at JNTU, Hyderabad. He is working as Assistant Professor in MVGR College of Engineering, Vizianagaram, AP. He has 5 publications in both International and National Journals and presented 19 papers at various International and National Conferences. His areas of interests are Image Processing, Speech processing and Video Processing.

Mr. U.V.S Sitarama varma obtained his B. Tech. and MS degrees from Nagarjuna University and Fairleigh Dickinson University, New Jersey, U.S.A in the year 1996 and 2002. He is working as Assistant Professor in MVGR College of Engineering, Vizianagaram , AP. He has 10 years of experience in Information technology and he worked with Siemen IKM German on medical image products like 'Syngo.via'. His areas of interests are Image Processing.

Mr. Sri Krishna Chaitanya Varma k is currently pursuing his B. Tech. in MVGR College of Engg., JNT University, Kakinada, Andhra Pradesh. His areas of interests are Image Processing and Speech processing .He is also a student member of IEEE

Miss Meena Jami is currently pursuing his B. Tech. in MVGR College of Engg., JNT University, Kakinada, Andhra Pradesh. Her areas of interests are Image Processing, Speech processing.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

Mr. V.V.N.S. Dileep is currently pursuing his B. Tech. in MVGR College of Engg., JNT University, Kakinada, Andhra Pradesh. His areas of interests are Image Processing and Speech processing.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)