World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:7, 2008

Low Computational Image Compression Scheme based on Absolute Moment Block Truncation Coding

K.Somasundaram and I.Kaspar Raj

***Abstract*—**In this paper we have proposed three and two stage still gray scale image compressor based on BTC. In our schemes, we have employed a combination of four techniques to reduce the bit rate. They are quad tree segmentation, bit plane omission, bit plane coding using 32 visual patterns and interpolative bit plane coding. The experimental results show that the proposed schemes achieve an average bit rate of 0.46 bits per pixel (bpp) for standard gray scale images with an average PSNR value of 30.25, which is better than the results from the exiting similar methods based on BTC.

***Keywords*—**Bit plane, Block Truncation Coding, Image compression, lossy compression, quad tree segmentation

I. INTRODUCTION

DIGITAL image compression methods help to reduce the space necessary to store or transmit the image data by changing the way these images are represented There are numerous methods for compressing digital image data and each has its own advantages and disadvantages [1]. BTC has been used for many years for compressing digital monochrome images. BTC for monochrome image compression was introduced by Delp and Mitchell [2]. It is a simple and lossy image compression technique. BTC has the advantage of being easy to implement compared to vector quantization [3] and transform coding [4, 5]. The BTC method preserves the block mean and the block standard deviation. In the encoding procedure of BTC the image is first partitioned in to a set of non overlapping blocks, and then the first two statistical moments and the bit plane are computed. In the decoder, each of the encoded image block are reconstructed using the bit plane and the two statistical moments. It achieves 2 bits per pixel (bpp) with low computational complexity.

Lema and Mitchell [6] presented a simple and fast variant of BTC called Absolute Moment Block Truncation Coding (AMBTC). It preserves the higher mean and lower mean of the blocks. However the bit rate achieved is 2 bpp which is same as in the original BTC. In order to reduce the bit rate several techniques have been used to code the statistical moments and the bit plane of BTC.

Since the bit rate of the original BTC is relatively high when compared to other still image compression techniques such as JPEG [4] or JPEG 2000 [5], many modification of BTC have been proposed to further reduce the bit rate. Arce and Gallahar [7] proposed a median filter roots method to code the bit plane and the bit rate is reduced to about 1.38 bpp. Udipikar and Raina [8] introduced BTC image compression using Vector Quantization (VQ) and the bit rate achieved is in the range of 1.0 -1.5 bpp. Zeng and Neuvo [9] have also proposed two BTC methods with VQ schemes. The hybrid BTC / VQ techniques reduce the bit rate, but they need more computation for encoding the code book generation. Ramana and Eswaran [10] introduced a simple strategy to reduce the storage size of bit plane. Chung-Woei Chao et.al.[11] presented a modified block truncation coding algorithm which used block clustering scheme and predefined binary edge patterns. Yung-Gi Wu [12] proposed probability based block truncation image bit plane coding. Yu-Chen Hu [13] presented a modified BTC with predictive technique and bit plane coding with edge pattern. Hence the improvements on BTC are continuing to reduce the low bit rate and computational complexity by keeping the image quality to acceptable limit.

In this paper we propose an image compression scheme based on BTC with low computational complexity. This compressor employs four techniques. These techniques are quad tree segmentation, bit plane omission, bit plane coding using 32 visual patterns and interpolative bit plane coding. From the experimental results, we found that the proposed scheme gives good image quality with low computational complexity and with low bit rate.

In Section II a review of the AMBTC method is given. Section III describes the proposed image compression scheme. The experimental results are discussed in Section IV. Finally we conclude this paper in Section V.

K. Somasundaram is a Professor at the Department of Computer Science and Applications, Gandhigram Rural Institute – Deemed University, Gandhigram, Tanilnadu , India (phone: 91-451-2452371; fax: 2453071 ; e-mail: somasundaramk@yahoo.com) .

I. Kaspar Raj is a Research Scholar at the Department of Computer Science and Applications, Gandhigram Rural Institute – Deemed University, Gandhigram, Tanilnadu , India (phone: 91-98420 17343; e-mail: kasparraj@gmail.com) and is working as system programmer in the same institute.

II. ABSOLUTE MOMENT BLOCK TRUNCATION CODING

In the BTC method, the image is divided into non-overlapping small blocks (normally 4 x 4 pixels). The moments are calculated for each block, i.e., the sample mean $\bar{x}$ and standard deviation σ. The mean $\bar{x}$ standard deviation σ are computed using :

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad , \qquad (1)$$

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x}_i)^2} \quad , \qquad (2)$$

where x_i represent the i^{th} pixel value of the image block and n is the total number of pixels in the block. The two values $\bar{x}$ and σ are termed as quantizers of BTC.

Taking $\bar{x}$ as the threshold value a two-level bit plane is obtained by comparing each pixel value x_i with the threshold. If $x_i < \bar{x}$ then the pixel is represented by '0', otherwise by '1'. By this process each block is reduced to a bit plane. The bit plane along with $\bar{x}$ 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.

In the decoder an image block is reconstructed by replacing by '1' s with *H* and the '0's by *L*, which are given by:

$$H = \bar{x} + \sigma\sqrt{\frac{p}{q}} \quad , \qquad (3)$$

$$L = \bar{x} - \sigma\sqrt{\frac{q}{p}} \quad , \qquad (4)$$

where *p* and *q* are the number of 0's and 1's in the compressed bit plane respectively.

Lema and Mitchell [2] presented a simple and fast variant of BTC, named Absolute Moment BTC (AMBTC) that preserves the higher mean and lower mean of a block. The AMBTC algorithm involves the following steps:

An image is divided into non-overlapping blocks. The size of a block could be (4 x 4) or (8 x 8), etc. Calculate the average gray level of the block (4x4) as :

$$\bar{x} = \frac{1}{16}\sum_{i=1}^{16} x_i \quad , \qquad (5)$$

where x_i represents the i^{th} pixels 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 ($\bar{x}$) and the remaining brought into the lower range. The mean of higher range x_H and the lower range x_L are calculated as :

$$x_H = \frac{1}{k}\sum_{x_i \geq \bar{x}}^{n} x_i \quad , \qquad (6)$$

$$x_L = \frac{1}{16-k}\sum_{x_i < \bar{x}}^{n} x_i \quad , \qquad (7)$$

where k is the number of pixels whose gray level is greater than $\bar{x}$.

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 grater than or equal to $\bar{x}$ and "0" to represent a pixel whose gray level is less than $\bar{x}$. The encoder writes x_H , x_L and *b* to a file. Assume that we use 8 bits to represent x_H , x_L respectively. Then the total number of bits required for a block is 8+8+16 =32 bits. Thus, the bit rate for the AMBTC algorithm is 2 bpp . In the decoder, an image block is reconstructed by replacing the `1' s with x_H and the '0''s by x_L 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

III. PROPOSED SCHEME

The proposed compression scheme makes use of AMBTC, quad tree segmentation, bit plane omission, bit plane coding using 32 predefined visual patterns and interpolative technique. 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. In bit plane omission technique, bit plane of the AMBTC method is omitted in the encoding procedure if the difference between the higher mean and the lower mean is less than a threshold value (***Th***) and only the block mean is retained. At the time of decoding bit plane omitted blocks are replaced by the respective block means. Bit plane coding with visual patterns is the technique to encode bit plane using the predefined 32 visual patterns as in Fig 1. Thus it needs only 5 bits to code the bit plane. The interpolative technique is the method that drops half of the bit plane at the time of encoding and at the time of decoding the dropped bits are recovered by taking the arithmetic mean of the adjacent values. Here it requires only 8 bits to store the bit plane. The detailed steps involved in the compression process are as follows:

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 $\bar{x}$, lower mean $\bar{x}_L$ and higher mean $\bar{x}_H$ for a block

Step 3: Fix a threshold value *Th1* and if $|\bar{x}_H - \bar{x}_L|$ <= *Th1* encode 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 (x_b) of size n = 8 x 8 pixels.

Step 5 : Compute the block mean $\bar{x}_b$, lower mean $\bar{x}_{bL}$ and higher mean $\bar{x}_{bH}$ for 8 x 8 block.

Step 6 : If $\left|\bar{x}_{bH} - \bar{x}_{bL}\right|$ <= *Th2* and encode the block x_b with the block mean $\bar{x}_b$, 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 (x_c) of size n = 4 x 4 Pixels.

Step 9: Compute the block mean $\bar{x}_c$, lower mean $\bar{x}_{cL}$ and higher mean $\bar{x}_{cH}$ for 4 x 4 block

Step 10: If $\left|\bar{x}_{cH} - \bar{x}_{cL}\right|$ <= *Th3* and encode the block x_c with the block mean $\bar{x}_c$, 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 $\bar{x}_c$ and the rest of the pixels are presented by '0'. Encode the bit plane using 32 predefined bit plane pattern of 4 x 4 (given in Fig. 1) along with lower mean $\bar{x}_{cL}$ and higher mean $\bar{x}_{cH}$, 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 $\bar{x}_{cL}$ and higher mean $\bar{x}_{cH}$, , 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.

Decoding steps

If the indicator flag is '0' replace the block x with the block mean $\bar{x}$.

If the indicator flag is '1' replace the block x_b with the block mean $\bar{x}_b$.

If the indicator flag is '01' replace the block x_c with the block mean $\bar{x}_c$.

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 $\bar{x}_{cH}$ and the 0s by lower mean $\bar{x}_{cL}$.

If the indicator flag is '11' block x_c is reconstructed by replacing the 1s by $\bar{x}_{cH}$ and the 0s by $\bar{x}_{cL}$. The dropped bits are estimated by the mean of the adjacent values as follows:

$$
\begin{aligned}
\hat{x}_i &= \frac{1}{3}(x_{i-1} + x_{i+1} + x_{i+4}) \text{ for } i = 2 \\
\hat{x}_i &= \frac{1}{2}(x_{i-1} + x_{i+4}) \text{ for } i = 4 \\
\hat{x}_i &= \frac{1}{3}(x_{i-4} + x_{i+1} + x_{i+4}) \text{ for } i = 5 \\
\hat{x}_i &= \frac{1}{4}(x_{i-4} + x_{i-1} + x_{i+1} + x_{i+4}) \text{ for } i = 7,10 \\
\hat{x}_i &= \frac{1}{3}(x_{i-4} + x_{i-1} + x_{i+4}) \text{ for } i = 12 \\
\hat{x}_i &= \frac{1}{2}(x_{i-4} + x_{i+1}) \text{ for } i = 13 \\
\hat{x}_i &= \frac{1}{3}(x_{i-1} + x_{i+1} + x_{i-4}) \text{ for } i = 15
\end{aligned}
\qquad (8)
$$

1111 1111 1111 0000 0000 0000 1000 1100
0000 1111 1111 0000 0000 1111 1000 1100
0000 0000 1111 0000 1111 1111 1000 1100
0000 0000 0000 1111 1111 1111 1000 1100

1110 0001 0011 0111 1000 1111 1111 0001
1110 0001 0011 0111 0000 1111 1111 0000
1110 0001 0011 0111 0000 1111 1111 0000
1110 0001 0011 0111 0000 0111 1110 0000

1100 1110 1111 1111 0011 0111 1111 1111
1000 1100 1110 1111 0001 0011 0111 1111
0000 1000 1100 1110 0000 0001 0011 0111
0000 0000 0100 1100 0000 0000 0001 0011

0000 0000 1100 1100 0000 0000 0001 0011
0000 1000 1100 1110 0000 0001 0011 0111
1000 1100 1110 1110 0001 0011 0111 1111
1100 1110 1111 1111 0011 0111 1111 1111

Fig. 1 Pre defined 32 bit plane patterns

1	**5**	9	**13**
2	6	**10**	14
3	**7**	11	**15**
4	8	**12**	16

Fig. 2 The pattern of dropping bits. The bold faced bits are dropped

IV. Results and Discussion

While implementing this algorithm we have to select three threshold values for bit plane omission technique at three different levels of quad tree segmentation, if we start our segmentation at the block size 16 X 16. Suppose if we begin the quad tree segmentation at the block size 8 x 8, we have to take two threshold values. To find out optimum threshold values for our image compression scheme, we have applied our image compression schemes with different combination of threshold values on standard eight images of size 512 x 512. The results are given in the Table I and Table II. In Table I case1 refers to our proposed method with quad tree segmentation starting from block size 16 X 16 and the in Table II case-2 refers to our proposed method with quad tree segmentation starting from block size 8 X 8

Using the results given in Table I and Table II we have decided to take *Th1 =10, Th2 = 15, Th3* = 20 as threshold values for three levels in case-1 and 15 for level 1 and 20 for level 2 in case-2 , so as to get a reasonable good quality picture with low bpp compared to other levels.

The Table III gives the bits needed to store an image block of different size for the proposed compression scheme. We used 6 bits for storing higher mean x_H and lower mean x_L instead of 8 bits.

In order to evaluate the performance of our methods, we carried out experiments on eight standard images Lena, Jet, Peppers, Tiffany, Girl, Barb, Boat and Zelda of size 512 X 512 pixels which are given in Fig 3. We applied our methods to each of the above 8 images. For comparison, we also used the AMBTC method and the scheme reported by Yu-Chen Hu [14] (YCH). As mentioned earlier we have taken *Th1 =10, Th2 = 15, Th3* = 20 for three levels of quad tree segmentation in case-1 and Th1 = 15 for level 1 and Th2 = 20 for level 2 as threshold values in case-2. We computed the PSNR values for each of the reconstructed images. The computed PSNR values and bpp values for the reconstructed images are given in Table IV.

From Table IV it can be seen that case-1 gives better *PSNR* values than that of YCH scheme with low bpp. Hence one can obtain better quality image and with better compression when compared to the YCH scheme. We also observe that in case-2 we get better PSNR value than YCH scheme. Case 2 gives better image quality without blocky appearance as the scheme starts with 8 x 8 pixel size. Since the YCH and the case-1 start the quad tree segmentation at 16 x 16 block size, the reconstructed image has blocky appearance. We note that our method ‘s given in case-1 and case-2 have a low bit rate than AMBTC method and the PSNR values of three of the eight standard images are also closer to that of AMBTC method. Therefore appropriate selection of threshold values in this scheme, a compression up to 0.38 bpp can be achievable.

We also observe that the PSNR values of the reconstructed images of girl and jet in case 1 and 2 are very close to that of AMBTC method. Hence the quality of the image is same the as that of AMBTC method for these two images. The girl and jet images are having more uniform segments than other images. So the proposed methods have better compression rate with low computational complexity for gray level images with more uniform areas.

We have also applied our image compression scheme to eight photographs taken for student identity which are given in Fig. 4. Since the photographs have uniform background our method 1 and method 2 achieve compression with low bit rate. Since the photographs are 256 X 256 size, we have taken 10 as threshold value at three levels of quad tree segmentation in case1 and at two levels in case2. We computed the PSNR values for each of the reconstructed images. The computed PSNR values and bpp values for the reconstructed images are given in Table V.

From the Table V it can be seen that our method given in case-1 has a bit rate lower than YCH with acceptable image quality in terms of *PSNR* values. Our Method in case-2 gives reconstructed image without blocky appearance. Since the case-1 starts the quad tree segmentation at 16 x 16 block size, the reconstructed image has blocky appearance. Our method as given in case-2 starts with 8x8 pixel size, can achieve image compression with low computational complexity, for gray scale images, with more uniform areas in the image.

TABLE I
PSNR AND BPP VALUES FOR DIFFERENT THRESHOLD VALUES FOR CASE-1

Threshold	**5,5,5**		**5,10,15**		**5,10,20**		**10,10,10**		**10,15,20**		**15,15,15**		**20,15,10**		**15,10,5**	
Image	**BPP**	**PSNR**	**BPP**	**PSNR**	**BPP**	**PSNR**	**BPP**	**PSNR**	**BPP**	**PSNR**	**BPP**	**PSNR**	**BPP**	**PSNR**	**BPP**	**PSNR**
lena	0.9653	31.92	0.5636	31.24	0.5190	30.91	0.6089	31.36	0.4325	30.55	0.4492	30.53	0.4598	30.18	0.6258	30.99
jet	0.7434	31.60	0.5201	31.18	0.4849	30.92	0.5428	31.21	0.4252	30.65	0.4440	30.71	0.4626	30.58	0.5737	31.07
peppers	1.1503	31.81	0.5593	31.20	0.5195	30.93	0.6145	31.29	0.4164	30.54	0.4283	30.45	0.4513	30.02	0.6594	30.92
tiffany	1.2060	30.16	0.6837	29.67	0.6230	29.42	0.7566	29.81	0.5267	29.17	0.5606	29.21	0.5849	28.96	0.8019	29.61
girl	0.7627	36.62	0.3831	34.76	0.3522	34.21	0.4101	34.83	0.2652	33.29	0.2723	33.06	0.2900	32.38	0.4470	34.02
barb	1.0777	26.68	0.7946	26.52	0.7516	26.43	0.8270	26.55	0.6978	26.33	0.7265	26.35	0.7453	26.27	0.8665	26.48
boat	0.8729	30.22	0.6624	29.91	0.6069	29.63	0.7029	29.99	0.5513	29.47	0.5843	29.58	0.5881	29.37	0.7279	29.86
zelda	1.0848	35.25	0.5161	33.40	0.4770	32.91	0.5701	33.68	0.3551	31.97	0.3625	31.70	0.3847	31.00	0.6651	32.94
Average	0.9823	31.78	0.5854	30.99	0.5418	30.67	0.6291	31.09	0.4588	30.25	0.4785	30.20	0.4958	29.85	0.6709	30.74

TABLE II
PSNR AND BPP VALUES FOR DIFFERENT THRESHOLD VALUES FOR CASE 2

Threshold	**5,5**		**10,10**		**10,15**		**10,20**		**15,10**		**15,15**		**15,20**		**10,5**	
Image	**BPP**	**PSNR**	**BPP**	**PSNR**	**BPP**	**PSNR**	**BPP**	**PSNR**	**BPP**	**PSNR**	**BPP**	**PSNR**	**BPP**	**PSNR**	**BPP**	**PSNR**
lena	0.9785	31.94	0.6484	31.53	0.5722	31.25	0.5275	30.92	0.5570	31.13	0.5054	30.94	0.4661	30.67	0.7442	31.65
jet	0.7780	31.63	0.5991	31.39	0.5472	31.20	0.5117	30.94	0.5463	31.15	0.5083	31.01	0.4774	30.79	0.6590	31.47
peppers	1.1525	31.82	0.6488	31.44	0.5588	31.20	0.5192	30.93	0.5371	31.01	0.4804	30.86	0.4454	30.64	0.7772	31.53
tiffany	1.2046	30.16	0.7829	29.88	0.6814	29.67	0.6207	29.42	0.6713	29.58	0.6043	29.44	0.5497	29.22	0.8966	29.97
girl	0.7889	36.71	0.4665	35.41	0.4059	34.81	0.3750	34.25	0.3892	34.45	0.3447	34.08	0.3163	33.63	0.5689	35.77
barb	1.0846	26.69	0.8489	26.59	0.7981	26.53	0.7550	26.43	0.7966	26.50	0.7586	26.45	0.7176	26.36	0.9182	26.61
boat	0.8987	30.24	0.7484	30.09	0.6834	29.93	0.6277	29.64	0.6849	29.90	0.6376	29.78	0.5891	29.54	0.8055	30.14
zelda	1.0890	35.26	0.5954	33.92	0.5166	33.41	0.4774	32.91	0.4705	32.78	0.4118	32.47	0.3764	32.11	0.7718	34.34
Average	0.9967	31.81	0.6673	31.28	0.5955	31.00	0.5518	30.68	0.5816	30.82	0.5314	30.63	0.4923	30.37	0.7677	31.44

TABLE III
BITS NEEDED TO STORE AN IMAGE BLOCK IN DIFFERENT TECHNIQUES

Size of image block	**Bit length**				
	Indicator	**Block Mean**	**Bit Plane**	**Higher & Lower mean**	**Total Bits**
16 x 16	1	8	-	-	9
8 x 8	2	8	-	-	10
4 X 4 Block mean	2	8	-	-	10
4x4 32 Visual pat	2	-	5	6+6	19
4X4 Interpolative	2	-	8	6+6	22

TABLE IV
PSNR AND BPP VALUES FOR DIFFERENT METHODS ON STANDARD IMAGES

	AMBTC		**Yu-Chen hu**		**Our Method**			
					Case- 1 *Th1*=10,*Th2*=15 *Th3*=20		**Case-2 *Th2*=15 *Th3*=20**	
Image	BPP	PSNR	BPP	PSNR	BPP	PSNR	BPP	PSNR
lena	2.0	33.25	0.4617	29.06	0.4325	30.55	0.4661	30.67
jet	2.0	31.42	0.4589	28.80	0.4252	30.65	0.4774	30.79
peppers	2.0	33.44	0.4367	29.20	0.4164	30.54	0.4454	30.64
tiffany	2.0	31.70	0.5747	27.63	0.5267	29.17	0.5497	29.22
girl	2.0	33.95	0.2737	32.52	0.2652	33.29	0.3163	33.63
barb	2.0	29.87	0.7517	24.50	0.6978	26.33	0.7176	26.36
boat	2.0	31.55	0.6055	27.65	0.5513	29.47	0.5891	29.54
Zelda	2.0	36.74	0.3633	31.09	0.3551	31.97	0.3764	32.11
Average	2.0	32.74	0.4908	28.81	0.4588	30.25	0.4923	30.37

TABLE V
PSNR AND BPP VALUES FOR DIFFERENT METHODS ON SPECIAL IMAGES

Image	**AMBTC**		**Yu-Chen hu**		**Our Method**			
					Case-1 *Th1*=*Th2*=*Th3*=10		**Case-2 *Th2*=*Th3*=10**	
	BPP	**PSNR**	**BPP**	**PSNR**	**BPP**	**PSNR**	**BPP**	**PSNR**
sid1	2.0	37.39	0.3808	32.61	0.3741	33.86	0.4463	34.18
sid2	2.0	38.90	0.2595	33.10	0.2556	34.71	0.3471	35.29
sid3	2.0	33.80	0.4385	29.11	0.4260	30.26	0.4972	30.38
sid4	2.0	34.49	0.5178	29.12	0.5006	30.51	0.5690	30.57
sid5	2.0	38.24	0.2336	33.05	0.2318	34.48	0.3195	34.82
sid6	2.0	35.76	0.3855	30.31	0.3763	32.54	0.4519	32.75
sid7	2.0	38.62	0.2267	33.56	0.2249	34.80	0.3182	35.17
sid8	2.0	38.22	0.2906	33.43	0.2859	34.52	0.3755	34.82
Average	2.0	36.93	0.3416	31.79	0.3344	33.21	0.4156	33.50

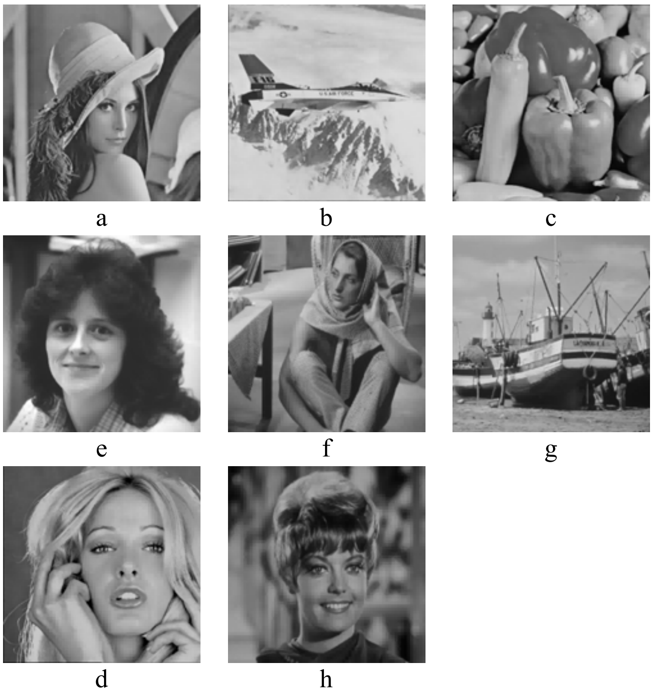

Fig. 3 Standard Images used for our experiment
a) lena b) jet c) peppers d) Tiffany e) girl
f) Barb g) boat h) Zelda

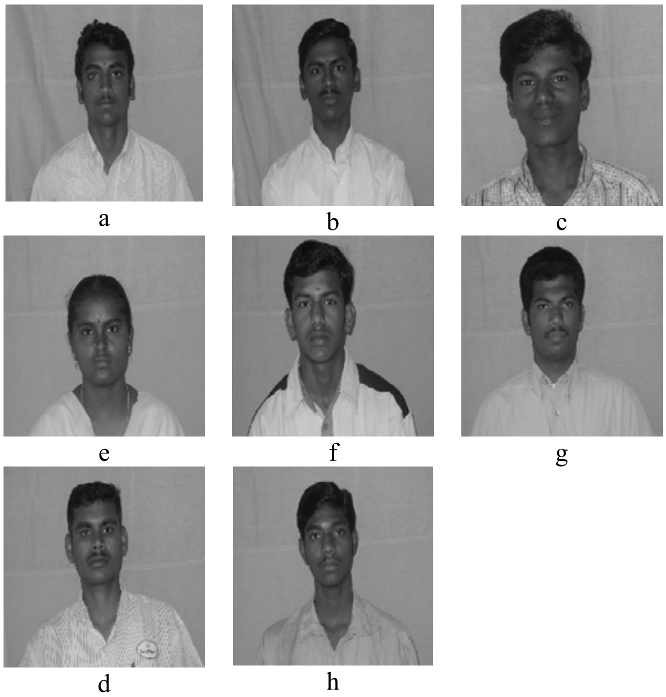

Fig. 4 Photograph Images used for our experiment
a) sid1 b) sid2 c) sid3 d) sid4 e) sid5 f) sid6
g) sid7 h) sid8

V. Conclusion

In this paper we have developed a low computational complexity two stage and three stage gray scale image compression scheme using quad tree segmentation.. The three stage compressor starts with 16x16 pixel block and the two stage compressor starts with 8x8 pixel block. Experimental results, by applying our schemes on standard images, show that an average bit rate of 0.46 bpp with an average PSNR value of 30.25 is for the three stage compressor and 0.4923 bpp with an average PSNR value of 30.37 for two stage compressor which are better in bpp and PSNR than that of YCH can be achieved. Our three stage compressor gives a bit rate of 0.33 bpp at a PSNR value of 33.21 for photos used for ID cards. Our compression scheme may be useful for low cost handheld devices with low computational power which handles images.

References

[1] David Solomon, Data Compression The complete reference 2nd edition, Springer 2001, Newyork.
[2] E.J. Delp, O.R. Mitchell, "Image Compression using Block Truncation Coding", IEEE, Trans. Communications , Vol . 27, pp.1335-1342, September 1979.
[3] N.M. Nasrabadi, R.B. King, Image coding using vector quantization: a review, IEEE Transactions on Communications COM-36(1998) , pp. 957-971.
[4] W.B. Pennebaker, and J.L.Mitchell, JPEG Still Image Data compression Standard., New York, Van Nosttrand Reinhold,1993.
[5] M. Rabbani and R. Joshi, ''An overview of the JPEG 2000 still image compression standard,'' Signal Process. Image Commun. 17, pp. 3–48, 2002.
[6] 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
[7] G.Arce and N.C. Jr.Gallagaher, "BTC image coding using median filter roots", IEEE Transaction on communications, 31, (6), pp. 784-793, 1983.
[8] V.Udpikar, and J.Raina, "BTC image coding using vector Quantization", IEEE Transactions on Communications., Vol. 35, pp 352-56,1987
[9] B.Zend, Y.Neuvo, "Interpolative BTC image coding with Vector Quantization", IEEE Transations on Communications, 41 (1993) 1436-1438.
[10] Y.V.Ramana , and C. Eswaran ," A new algorithm for BTC image bit plane coding " IEEE Trans on Communications. Vol. 43, No.6, pp. 2010-2011, June 1995
[11] Chung-Woei Chao, chaur-Heh Hsieh and Po-Ching Cu, "Image compression using modified Block Truncation Coding algorithm" , Signal Processing Image communication, 12 (1998) 1-11.
[12] Yung-Gi Wu ,"Block Truncation image Bit plane coding , SPOIE, Optical Engineering 41(10) 2476-2478 October 2002
[13] Yu-Chen Hu,"Predictive moment preserving block truncation coding for gray level image compression,Journal of electroninc Imaging, Vol.13(4),2004 pp. 871-877
[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.