Interantional Journal of Soft Computing 2 (1): 118-123, 2007

A New Predictive and Interpolative Image Compression Scheme Based on Block Truncation Coding

K. Somasundaram and I. Kaspar Raj
Department of Computer Science and Applications, Gandhigram Rural Institute-Deemed University Gandhigram-624 302, Tamilnadu, India

Abstract: Block Truncation Coding (BTC) is a fast moment preserving lossy image compression technique. In this paper a new image compression scheme based on BTC is proposed. The proposed image compression scheme contains four techniques. They are prediction technique, bit plane omission technique, bit plane coding using 32 predefined visual patterns and interpolative technique. The experimental results show that the proposed image compression scheme achieves a low bit rate with reasonable good picture quality compared to the recent published work based on BTC. The proposed scheme also has low computational complexity.

Key words: Lossy image compression, block truncation coding, bit plane, visual patterns

INTRODUCTION

Image compression is used to reduce the amount of memory required to store an image without much affecting the quality of an image. It also reduces the time required for images to be sent over the Internet and Intranet. There are several different methods to compress an image file[1]. Block Truncation Coding is one of the simple and fast compression methods and was introduced by Delp and Mitchell[2]. Block truncation coding is a lossy compression method. It works by dividing the image into small sub images and then reducing the number of gray levels in each block. This reduction is performed by a quantizer that adapts to the local image statistics. The BTC method preserves the block mean and standard deviation. In the encoding procedure of BTC the image is first partitioned in to a set of non overlapping blocks. The first two statistical moments for each block are computed. Using the block mean as quantizing level, each pixel in the block are replaced by 1 and 0, resulting in a binary matrix called bit plane. In the decoder, each of the encoded image blocks is reconstructed using the bit plane and the two statistical moments. It achieves 2 bits per pixel (bpp) with low computational complexity. It is easy to implement compared to vector quantization[3] and transform coding[4,5].

Lema and Mitchell presented a simple variant of BTC called Absolute Moment Block Truncation Coding AMBTC[6] . It preserves the higher mean and lower mean of the blocks. However the bit rate achieved with the BTC or AMBTC is 2 bpp. In order to reduce the bit rate several techniques such as median filtering[7], Vector quantization[8], Interpolation[9] and prediction[10] have been used to code the two statistical moments and the bit plane of BTC. Yung-Gi Wu[11] proposed a probability based block truncation image bit plane coding. Yu-Chen Hu[12] presented low bit-rate image compression scheme based on AMBTC. Hence the improvements on BTC are continuing to reduce the bit rate and computational complexity by keeping the image quality to acceptable limit.

In this study we propose an image compression scheme based on AMBTC with low computational complexity. This scheme is a combination of four techniques, prediction technique, bit plane omission, bit plane coding using 32 predefined visual patterns and interpolative bit plane coding. Experimental results show that the proposed scheme gives good image quality with low computational complexity and with low bit rate.

BLOCK TRUNCATION CODING AND AMBTC

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

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

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

Corresponding Author: K. Somasundaram, Department of Computer Science and Applications, Gandhigram Rural Institute-Deemed University Gandhigram-624 302, Tamilnadu, India

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 4×4 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:

$$H = \bar{x} + \sigma\sqrt{\frac{p}{q}} \tag{3}$$

$$L = \bar{x} - \sigma\sqrt{\frac{q}{p}} \tag{4}$$

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 (4×4) or (8×8), etc. The average gray level of the pixels in that block (4×4) is calculated as:

$$\bar{x} = \frac{1}{16}\sum_{i=1}^{16} x_i \tag{5}$$

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:

$$x_H = \frac{1}{k}\sum_{x_i \geq \bar{x}}^{n} x_i \tag{6}$$

$$x_L = \frac{1}{16-k}\sum_{x_i < \bar{x}}^{n} x_i \tag{7}$$

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. If x_H and x_L are represented by 8 bits 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.

PROPOSED SCHEME

The proposed compression scheme makes use of AMBTC, prediction technique, bit plane omission, bit plane coding using 32 predefined visual patterns and interpolative technique.

Prediction technique: This technique makes use of correlation existing among the neighboring image blocks. The prediction technique involves in finding a near similar neighbor upper or left block for the current block and replacing the current block by a code of the matched block. To find similarity of the current block with a neighboring block the squared Euclidean distance between the two blocks is used. The squared Euclidean distance between two blocks x and y is calculated as follows:

$$d(x,y) = \sum_{i=1}^{n}(x_i - y_i)^2 \tag{8}$$

where x_i and y_i are the corresponding the i^{th} pixels of the respective blocks.

Bit plane omission technique: In bit plane omission technique, the bit plane obtained using an AMBTC method is omitted in the encoding procedure if the absolute difference between the higher mean x_H and the lower mean x_L of that block is less than a threshold value and retaining the block mean only. At the time of decoding bit plane omitted blocks are replaced by a block filled with the block mean.

Visual pattern matching: In this technique, the bit plane obtained is matched with 32 visual patterns shown in Fig. 1.

If the bit plane obtained in AMBTC for a block exactly matches with any of the pattern, then the bit plane is replaced with the index of the matched pattern. Thus it needs only 5 bits to code the bit plane.

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	1100	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

Interpolative technique: In this technique half (8 bits) of the bits in the bit plane of AMBTC is dropped at the time of encoding as in Fig. 2. In decoding the dropped bits are recovered by taking the arithmetic mean of the adjacent values as in Eq. 9.

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

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: Ivide the given image into a set of non overlapping blocks, x of size n = 4×4 pixels

Step 2: Ncode the blocks starting from left to right and top to bottom sequence.

Step 1: F the block is in first row or first column then go to step 5.

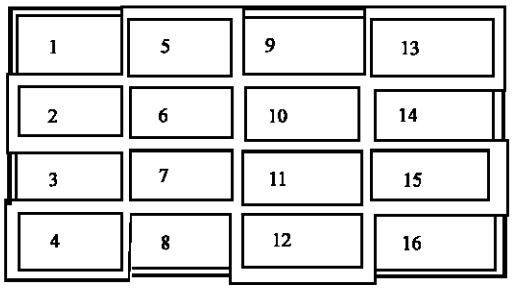

Fig. 2: The pattern of dropping bits. The bold faced bits are dropped

Step 4: o check the current block(cb) with upper block (ub) and lower block (lb) for similarity calculate the squared Euclidean distance

du = d(cb, ub) and dl = d(cb, lb) as in Eq. 8.
Fix threshold for d (x, y) as Th1. Find the dmin = minimum (du, dl).

IF dmin < Th1 THEN
IF upper block = dmin Then
encode the current block as 00
go to step 9
ELSE
encode the current block as 01
go to step 9
End IF
End IF

Step 5: Compute the block mean $\bar{x}$, lower mean $\bar{x}_L$ and higher mean $\bar{x}_H$ for the block

$|\bar{x}_H - \bar{x}_L|$

Step 6: Fix a threshold value Th2. If < = Th2, encode the block x with the block mean $\bar{x}$, put '100' as a indicator bit as prefix code and go to step 9.

Table 1: Bits needed to store an image block in different techniques

	Bit length				
Size of image block	Indicator bits	Block mean	Bit plane	Higher and lower mean	Total bits
Predictive	2	-	-	-	2
Bit plane Omission	3	8	-	-	11
32 Visual pattern	3	-	5	6+6	20
Interpolative	3	-	8	6+6	23

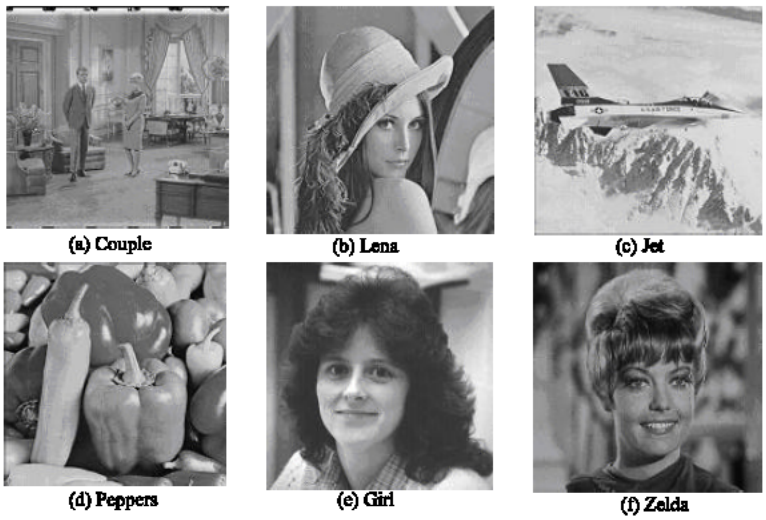

Fig. 3: Standard Images USED FOR experiment

Step 7: Construct the bit plane by taking '1' for the pixels with values greater than or equal to the mean $\bar{x}$ and the rest of the pixels are presented by '0'. Encode the bit plane using the 32 predefined bit plane pattern of 4×4 with lower mean $\bar{x}_L$ and higher mean $\bar{x}_H$, put '101' as a indicator bit as prefix code and go to step 9. If the bit plane does not match with 32 visual pattern then go to step 8 .

Step 8: Drop a pattern of bits as shown in Fig. 2, encode the block by the remaining bits with the lower mean $\bar{x}_L$ and higher mean $\bar{x}_H$, put '110' as a indicator bit as prefix code.

Step 9: Go to step 3 until all the blocks are processed.

Decoding steps

If the indicator flag

= '00' replace the block x with the upper block
= '01' replace the block x with the left block
= '100' replace the block x with the block mean $\bar{x}$.
= '101' Get the 32 visual pattern bit plane and decode the bit plane by replacing the 1s by and higher mean $\bar{x}_H$ and the 0s by lower mean $\bar{x}_L$.
= '110' block x is reconstructed by replacing the 1s by $\bar{x}_H$ and the 0s by $\bar{x}_L$. The dropped bits are estimated by the mean of the adjacent values as in Eq. 9:

This scheme starts with prediction technique and ends with interpolative technique based on BTC and hence named as Predictive and Interpolative BTC (PIBTC) image compression scheme. Under this PIBTC image compression scheme the bits needed to represent the 4x4 block of a given image is given in the Table 1.

Here in the bit plane coding using 32 predefined visual patterns and interpolative technique we code the higher and lower mean of AMBTC in 6 bits.

RESULTS AND DISCUSSION

To evaluate the performance of the proposed PIBTC image compression scheme, we take six standard monochrome images of size 512×512 images "Couple", "Lena", "Jet", "Peppers", "Girl" and "Zelda" which are given in Fig. 3a-f.

To implement this algorithm suitable threshold values for Th1 and Th2 are to be selected. Using the results in[14], we have taken Th2 = 20 for bit plane omission technique. To find out optimum threshold value for prediction technique PIBTC image compression scheme, we have applied our image compression schemes with different combination of threshold values Th1 and Th2 on the six standard images. The results obtained in terms of bpp and PSNR are given in the Table 2. Using the results given in Table 2 we have decided to take Th1 = 400 for prediction

Table 2: PSNR and BPP values for different threshold values

	Th1,Th2 = 400,20		= 500,20		= 600,20		= 700,20		= 800,20	
Images	BPP	PSNR	BPP	PSNR	BPP	PSNR	BPP	PSNR	BPP	PSNR
Couple	0.69	29.06	0.66	29.03	0.64	29.01	0.62	28.98	0.60	28.95
Lena	0.56	30.96	0.54	30.91	0.52	30.87	0.50	30.82	0.49	30.76
Jet	0.50	30.89	0.49	30.85	0.48	30.81	0.47	30.76	0.46	30.71
Peppers	0.56	31.01	0.53	30.95	0.51	30.90	0.48	30.84	0.47	30.77
Girl	0.40	34.33	0.38	34.16	0.36	34.03	0.35	33.92	0.33	33.72
Zelda	0.51	33.14	0.47	33.03	0.45	32.92	0.43	32.79	0.42	32.69
Average	0.54	31.57	0.51	31.49	0.49	31.42	0.48	31.35	0.46	31.27

Table 3: PSNR and BPP values for different methods on standard images

	Ambtc		Ych Th1,Th2=400,3		PIBTC 400,20		PIBTC 150,20		PIBTC 3200,20	
Images	BPP	PSNR	BPP	BPP	BPP	PSNR	BPP	PSNR	BPP	PSNR
Couple	2.0	32.47	0.78	26.27	0.69	29.06	0.80	29.11	0.42	27.84
Lena	2.0	33.25	0.68	28.65	0.56	30.96	0.68	31.07	0.37	29.49
Jet	2.0	31.42	0.56	27.83	0.50	30.89	0.58	31.02	0.38	29.52
Peppers	2.0	33.44	0.79	28.73	0.56	31.01	0.72	31.11	0.35	29.35
Girl	2.0	33.95	0.53	33.85	0.40	34.33	0.51	34.65	0.24	31.03
Zelda	2.0	36.74	0.69	32.61	0.51	33.14	0.66	33.36	0.28	30.39
Average	2.0	33.55	0.67	29.66	0.54	31.57	0.66	31.72	0.34	29.60

Fig. 4: The reconstructed Standard Images using PIBTC

technique and Th2 = 20 as threshold value for bit plane omission technique, so as to get a reasonably good quality picture with low bpp.

We applied PIBTC image compression scheme to each of the 6 images given in Fig. 3. For comparison, we also used the AMBTC method and the scheme of Yu-Chen Hu[13] (YCH). We wanted to compare the performance of our scheme with YCH. Hence we set Th1,Th2 = 150,20 so that the average bpp values are equal and Th1,Th2 = 3200,20 so that the average PSNR values are matched. The computed PSNR and bpp for three sets of (Th1,Th2) are given in Table 3.

We note from Table 3 that PIBTC gives better *PSNR* values and lower bpp than that of YCH scheme. The results for the set Th1, Th2 = 150, 20 gives a result matching the average bpp of our scheme with YCH at 0.66. The resulting average PSNR value of our scheme gives 31.75 while that of YCH is 29.66, an improvement of 7%. The results for the set Th1, Th2 = 3200, 20 gives a matching average PSNR value of our scheme with YCH at 29.6. We note that our scheme gives bpp of 0.34 while that of YCH is 0.67, an improvement of 49%. For any given application an appropriate selection of Th1 and Th2 values will give the required bpp and PSNR. The reconstructed images of PIBTC image compression scheme is given in Fig. 4a-f.

CONCLUSION

A low bit rate image compression scheme by extending the AMBTC scheme has been proposed. This scheme is developed by combining prediction, bit plane

omission, pattern matching and interpolative techniques. Experiments using our scheme on standard images show that an average bit rate of 0.34 giving an average PSNR value of 29.6 is achievable. Our scheme compared to the work of YCH[13] gives an improvement of 7% in image quality while reducing the bit rate to about 50%. An appropriate selection of the two thresholds in this scheme can be used in an application to get the desired bpp and PSNR within the region of its capability.

REFERENCES

1. David Solomon, 2001. Data Compression The Complete Reference 2nd (Edn.), Newyork.
2. Delp, E.J. and O.R. Mitchell, 1979. Image Compression using Block Truncation Coding, IEEE, Trans. Communications, pp: 1335-1342.
3. Nasrabadi, N.M. and R.B. King, 1998. Image coding using vector quantization: A review, IEEE Transactions on ommunications COM-36, pp: 957-971.
4. Pennebaker, B. and J.L. Mitchell, 1993. JPEG still image data compression Standard, New York, Van Nosttrand Reinhold.
5. Rabbani, M. and R. Joshi, 2002. An overview of the JPEG 2000 still image compression standard, Signal Process. Image Commun., pp: 3-48.
6. Lema, M.D. and O.R. Mitchell, 1984. Absolute moment block truncation coding and its application to color images, IEEE Trans. On Communications, pp: 1148-1157.
7. Arce and N.C. Jr. Gallagaher, 1983. BTC image coding using median filter roots, IEEE Transaction on Communications, pp: 784-793.
8. Udpikar and J. Raina, 1987. BTC image coding using vector Quantization, IEEE Transactions on Communications, pp: 352-56.
9. Zend, B. and Y. Neuvo, 1993. Interpolative BTC image coding with Vector Quantization, IEEE Transations on Communications, 41: 1436-1438.
10. Ramana, Y.V. and C. Eswaran, 1995. A new algorithm for BTC image bit plane coding IEEE Trans on Communications. pp: 2010-2011.
11. Yung-Gi Wu, 2002. Block Truncation image Bit plane coding, SPOIE, Optical Engineering, 41: 2476-2478.
12. Yu-Chen Hu, 2003. Low complexity and low bit-rate image compression scheme based on Absolute Moment Block Truncation Coding, pp 1964-1975.
13. Yu-Chen Hu, 2004. Predictive moment preserving block truncation coding for gray level image compression, J. Electronics Imaging, pp: 871-877.
14. Somasundaram K. and I. Kaspar Raj, 2006. Low Computational Image compression scheme based on Absolute Moment Block Truncation Coding, Enformatika Transactions on Enginee. Comp. and Tech., pp: 184-190.