2016

k-d Tree-Segmented Block Truncation Coding for Image Compression

Proceso L. Fernandez Jr
Ateneo de Manila University, pfernandez@ateneo.edu

Ryan Rey M. Daga

k-d Tree-Segmented Block Truncation Coding for Image Compression

Ryan Rey M. Daga[1,2] and Proceso L. Fernandez[1]

[1]*Ateneo de Manila University, Loyola Heights, Katipunan Ave., Quezon City, Philippines*
[2]*University of the Philippines Visayas Tacloban College, Tacloban City, Leyte, Philippines*

Abstract. Block truncation coding (BTC) is a class of image compression algorithms whose main technique is the partitioning of an image into pixel blocks that are then each encoded using a representative set of pixel values. It is commonly used because of its simplicity and low computational complexity. The Quadtree-segmented BTC (QTS-BTC), which utilizes a dynamic hierarchical segmentation technique, is among the most efficient in the BTC class. In this study, we propose a new BTC variant that introduces two ideas: (1) the use of a *k*-d tree for segmentation and (2) the use of a Mean Squared Error (MSE) threshold for dynamically determining the granularity of the blocks. We refer to this new BTC variant as the *k*-d Tree Segmented BTC (KTS-BTC), and we test this against some of the existing BTC variants by running the algorithms on a standard image compression dataset. The results show that the proposed variant yields low bit rates of the compressed images, even outperforming the state-of-the-art QTS-BTC, without a significant reduction in image quality as measured using the Peak Signal-to-Noise Ratio (PSNR). The utilization of *k*-d tree for image segmentation is further shown to have more impact than that of employing the MSE thresholding scheme as a block activity classifier.

1 Introduction

Nowadays, digital images are being used in many different fields. This has resulted to a continuous accumulation of stored and transmitted digital images, as well as an increase in the quality, and hence file size, of a non-trivial fraction of existing digital images. Compression techniques are thus becoming increasingly important because these help reduce the storage requirements and bandwidth costs.

There are several classes of image encoding schemes that have been proposed. These include the vector quantization [1], fractal image compression [2], wavelet image compression [3], and block truncation coding (BTC) [4]. In this study, we focus on the last one.

Block truncation coding was originally proposed by Mitchel et al., in 1979 for grayscale image compression [4]. This technique divides an image into non-overlapping blocks, with each block encoded using a representative set of pixel values. BTC has paved the way for other pixel-based moment-preserving compression techniques. Different methods have been introduced to suit both grayscale and colored images. Absolute moment block truncation coding (AMBTC) [5], generalized moment preserving quantization [6], multiple block partitioning [7] and even the integration of a heuristic such as harmony search [8] have been introduced.

BTC (and its variants) is a popular choice for compression because of its low computational cost, making it suitable for real time multimedia applications. It is generally able to quickly compress an image without much loss of image quality. However, BTC has a higher bit rate for image coding compared to other classes of compression algorithms. To address this issue, some proposed image coding schemes have employed representative patterns to encode bitmaps [9] [10].

Recently, the quadtree segmentation BTC [11] has been proposed to cut down the bit rate requirements of an image while maintaining the image quality. This segmentation technique exploits the spatial similarity among neighboring pixels in the image, and dynamically determines the size of the image segments (i.e., non-overlapping blocks) based on a threshold of difference between two means.

In this study, we propose a new BTC variant that extends the ideas from the quadtree segmentation BTC. In particular, we propose the use of a *k*-d tree (instead of a quadtree) for a more flexible segmentation, and the use of Mean Square Error (MSE), instead of difference of two means, for thresholding. These concepts are discussed in greater detail in later sections.

2 Preliminaries

In this section, we cover the important algorithms that can help in better understanding the quadtree segmentation BTC, from which several concepts in our proposed new algorithm are based.

2.1 Block Truncation Coding

In Block Truncation Coding (BTC), an image is first divided into non-overlapping $m \times m$ blocks. Then for each block, the mean $\bar{x}$ of the pixel values is computed in the standard way,

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

where $n = m^2$.

A corresponding $m \times m$ bit plane1 is generated by encoding each pixel in the block by either 1 or 0 depending on its value relative to $\bar{x}$. Specifically, if the pixel value is greater than or equal to $\bar{x}$, it is coded as 1; otherwise, it is encoded as 0.

The bit values 1 and 0 are mapped to the statistical moments x_H and x_L, respectively, which are computed as follows:

$$x_H = \bar{x} + \sigma\sqrt{\frac{p}{q}} \quad (2)$$

$$x_L = \bar{x} - \sigma\sqrt{\frac{p}{q}} \quad (3)$$

Here, p is the number of 0's in the bit plane, q is the number of 1's, and

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2} \quad (4)$$

The bit rate of the resulting compressed greyscale image depends on the size of all blocks. For the typical 4×4 block size, each of the 16 pixels is encoded using 1 bit on the bit plane, and an overhead of two 8 bits are needed for the x_H and x_L representative values. Thus, 32 bits are used to encode a 16-pixel block, resulting in a bit rate of 2 bits per pixel (bpp).

2.2 Absolute Moment Block Truncation Coding

The Absolute Moment Block Truncation Coding (AMBTC) is a simple and fast variant of BTC that has been proposed for color images [12][5]. A given color image in RGB format is first decomposed into three greyscale images corresponding to the red, green and blue channels.

The encoding for each image channel then follows the BTC technique, except for the computation of the two statistical moments. The formula used in AMBTC are given in Equations 5 and 6.

$$\bar{x}_H = \frac{1}{q}\sum_{x_i \geq \bar{x}} x_i \quad (5)$$

$$\bar{x}_L = \frac{1}{p}\sum_{x_i < \bar{x}} x_i \quad (6)$$

where p and q are still the number of 0's and 1's, respectively, in the generated bit plane.

The resulting bit rate for images compressed using AMBTC is similar to that of the BTC. Using 4×4 blocks, the AMBTC bit requirements is 2 bpp for greyscale images and 6bpp for three-channel color images. Moreover, the AMBTC produces better image quality than BTC, where quality is measured in terms of Peak Signal-to-Noise Ratio (PSNR):

$$PSNR = 10\log_{10}\left(\frac{255^2}{MSE}\right) \quad (7)$$

In the above formula, MSE refers to the Mean Squared Error, which compares each pixel value $x_{i,j}$ at position (i, j) in the original image with the corresponding pixel value $y_{i,j}$ in the reconstructed image. In this study, since all input data images have dimension 512×512, the MSE is computed using the formula

$$MSE = \frac{\sum_{i=0}^{511}\sum_{j=0}^{511}(x_{i,j} - y_{i,j})^2}{(512)(512)} \quad (8)$$

2.3 Quadtree Segmented Block Truncation Coding

The quadtree segmented BTC is a hierarchical segmentation technique that partitions an image into variable-sized square blocks. During the quadtree segmentation process, a series of binary decisions with respect to different threshold values is made. A block type classifier is employed to determine the block activity, and it is usually based on a statistical information [11]. If a block is inactive (e.g., low variance) the segmentation of the given block is terminated, otherwise the block is further divided into four sub-blocks.

Chen et al. recently proposed the quadtree-segmented AMBTC [11]. In this coding scheme, a maximum of three-level quadtree structure, an example of which is shown in Figure 1, is imposed on the segmentation. The procedure of quadtree-segmented AMBTC is as follows:

Step 1: Decompose the color image into three grayscale images using the red, blue and green channels of the image.

Step 2: Partition each grayscale image into image blocks of size 16 ×16.

Step 3: Select an unprocessed 16 ×16 image block, and calculate two quantization levels $\bar{x}_H$ and $\bar{x}_L$ using the formula in AMBTC.

Step 4: If $|\bar{x}_H - \bar{x}_L| \leq THRESHOLD$, then encode this 16 ×16 block using its block mean and proceed to Step 8. Otherwise, subdivide this block into four 8×8 blocks.

Step 5: While there is still an unprocessed 8×8 block, select one such block, and calculate the mean and calculate two quantization levels $\bar{x}_H$ and $\bar{x}_L$. Otherwise, proceed to Step 8.

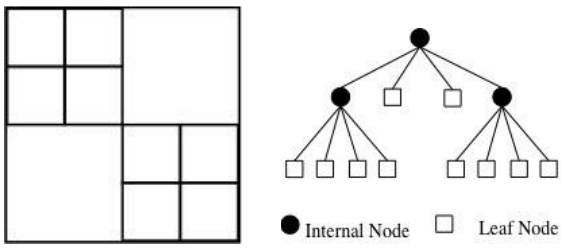
Step 6: If $|\bar{x}_H - \bar{x}_L| \leq THRESHOLD$, then encode this 8 ×8 block using its block mean and proceed to Step 5. Otherwise, subdivide this block into four 4×4 blocks.

Step 7: For each 4×4 image block, calculate the mean and calculate two quantization levels $\bar{x}_H$ and $\bar{x}_L$. If $|\bar{x}_H - \bar{x}_L| \leq THRESHOLD$, then encode this 8 ×8 block using its block mean. Otherwise, encode this block using a 4×4 bit plane. Following the technique in AMBTC. Return to Step 5 after processing all 4×4 blocks.

Step 8: If there is still any 16×16 blocks to be processed, return to Step 3.

We refer to this BTC variants as QTS-BTC in this paper. An example of a block segmentation in QTS-BTC is shown in Fig 1. Here, a 16×16 block is segmented into four 8×8 blocks, two of which are each further subdivided into four 4×4 blocks.

The bit rate in QTS-BTC is typically better than in AMBTC (or BTC) since it is possible to represent an entire block with a single 8-bit number instead of an $m \times m$ bit plane plus two 8-bit quantization values. However, the PSNR values are typically a little bit better in AMBT (BTC).



(b) A segmentation of a 16×16 blocks (a) Corresponding three-level quadtree

Figure 1. Example of a three-level quadtree segmentation

3 Proposed Compression Technique

In this study, we improve the QTS-BTC performance by applying two modifications:

1. The use of a *k*-d tree for the segmentation of a block
2. The use of the inequality MSE ≤ THRESHOLD t decide whether or not to subdivide a block further.

We refer to this new BTC variant as *k*-d tree-segmented BTC, or KTS-BTC. Figure 2 shows the overall process of the proposed image coding scheme.

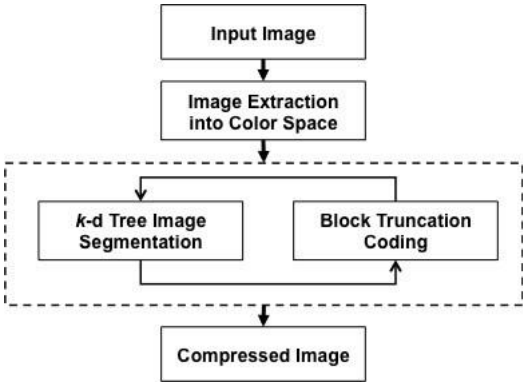


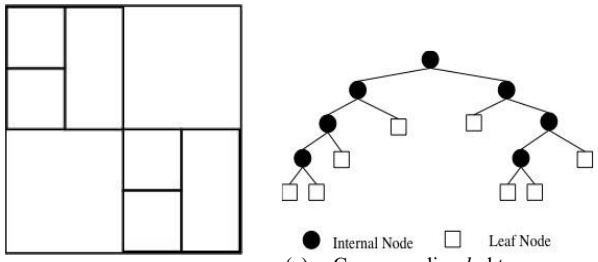
Figure 2. General flow of the proposed compression technique

An input color image is initially decomposed into three greyscale images corresponding to the three color channels, and then each greyscale image is partitioned into non-overlapping 16×16 blocks. For each block, the block mean is computed and the block activity is determined by calculating the MSE of replacing all pixel values by the mean. If a block is inactive (i.e., MSE ≤ THRESHOLD), then the block is encoded using its block mean. Otherwise, it is partitioned into two equally-sized sub-blocks.

The partitioning of a block follows the *k*-d tree partitioning of a 2-dimensional space. That is, the partitioning is alternately done on the x and y axes: a 16×16 block may be divided vertically (producing rectangular 8×16 blocks), then horizontally (producing 8×8 blocks, then vertically), then vertically (rectangular 4×8 blocks), and then finally horizontally (square 4×4 blocks).

The smallest blocks are 4×4 blocks. Each of theses 4×4 blocks is encoded either using the block mean if the block is inactive, or using a 4×4 bit plane following the AMBTC technique if the block is active. This process terminates when all blocks have been encoded.

Figure 3 shows an example of a *k*-d tree segmentation of a block. It should be noted that any block segmentation using the quadtree can be achieved using *k*-d tree segmentation (albeit using more levels in the tree, and thus more bits in the encoding), but the reverse is not true. Since there is possibility to encode an $m \times 2m$ vertical blocks with a single pixel value, which is not allowed in the QTS-BTC, then there is opportunity to have a lower overall bit rate. This, however, comes at the risk of reducing the PSNR. This risk is managed by using an MSE threshold, which may be set to some low value based on the desired PSNR.



(b) A segmentation of 16×16 block (a) Corresponding *k*-d tree

Figure 3. Example of a *k*-d tree segmentation of a 16×16

To encode the resulting variable-sized blocks properly, we use a bit flag 1 to indicate that a block is to be divided further, and 0 otherwise. Table 1 shows the prefixes used for the different block types, as well as the effective bit rates (in bits per pixel) for both greyscale and color images.

The number of bits used in the computation of the bit rates may be determined by careful calculations. For example, an inative 8×8 block uses the prefix 110. The first bit (1) indicates that the main block (16×16) is divided further. This 1-bit overhead cost is, however, shared among four 8×8 blocks. The second bit (1) implies that the 8×16 block is subdivided further, and this 1-bit overhead cost is shared by two. Finally, the last prefix bit (0) indicates that the 8×8 block is inactive, and this 1-bit overhead is shouldered entirely by the 8×8 block. Together with the 8-bit mean, the number of bits required to represent an inactive 8×8 block is this $\frac{1}{4} + \frac{1}{2} + 1 + 8 =$

9.75 bits. The values for the other block types can be similarly analyzed,

Table 1. Effective bit rates (in bpp) of the different block types on 1-channel and on 3-channel images

Block Type	Prefix	Num of Bits	Eff. Bit rate	
			1-ch	3-ch
16×16	0	9	0.035	0.105
8×16	10	9.5	0.074	0.223
8×8	110	9.75	0.152	0.457
4×8	1110	9.875	0.309	0.926
4×4, inactive	11110	9.9375	0.621	1.863
4×4, active	11111	33.937	2.123	6.369

4 Results and Discussion

To test the performance of the proposed KTS-BTC, we run it on a set of standard test images and compared the bit rates and PSNR values of the generated images against those produced by the AMBTC and QTS-BTC. These images, shown in Figure 4, were used in a previous study by Che et al. [11] and were taken from [13].

(c) Airplane (b) House (a) Lena

(f) Peppers (e) Splash (d) Tiffany

Figure 4. Color test images (512×512)

4.1 Compressed Image Results

Figure 5 shows some compressed images using AMBTC, QTS-BTC and the proposed KTS-BTC. There is no noticeable difference among the images, and this is validated by the small differences in the computed PSNR values.

Table 2 details the bit rates of the compressed images produced using these 3 techniques. The threshold values for the KTS-BTC and QTS-BTC used here are both 5. The proposed KTS-BTC yielded the lowest bit rate requirement. It was able to reduce the bit rates of the compressed images produced by AMBTC and QTS-BTC by an average of 22.62% and 14.83%, respectively.

As may be expected, the lower bit rate in KTS-BTC is achieved at the cost of slightly inferior PSNR (see Table 3). However, the difference is quite minimal, e.g., the average PSNR value from the AMBTC was reduced by about 0.53% only.

(a) AMBTC: 32.413 (b) Quadtree: 32.279 (c) k-d Tree: 32.253

Figure 5. Example of compressed images, using different compression techniques, with their corresponding PSNR values

Table 2. Bit rate requirements of color images using different image coding schemes

Image	AMBTC	Quadtree	k-d Tree
Airplane	6	4.175	3.71
House	6	5.103	4.844
Lena	6	5.989	5.293
Peppers	6	6.02	5.696
Splash	6	5.555	3.305
Tiffany	6	5.865	5.009
Average	**6**	**5.451**	**4.643**

Table 3. PSNR (dB) of color images using different image coding schemes

Image	AMBTC	Quadtree	k-d Tree
Airplane	32.413	32.279	32.253
House	30.498	30.480	30.461
Lena	33.109	33.112	32.995
Peppers	32.701	32.709	32.654
Splash	36.158	36.094	35.625
Tiffany	35.308	35.310	35.145
Average	**33.365**	**33.331**	**33.189**

These results indicate that the proposed KTS-BTC compression technique is generally better than QTS-BTC and AMBTC. We investigate further the effect of the two introduced ideas in KTS-BTC.

4.2 Effect of k-d Tree Segmentation

To determine the impact of the use of the k-d tree for segmentation, we implemented the KTS-BTC with a block activity classifier that uses $|\bar{x}_H - \bar{x}_L| \leq THRESHOLD$ to determine when to perform further segmentation.

Table 4 shows that across varying threshold values, the average bit rate is better in KTS-BTC, while the average PSNR value is slightly better in QTS-BTC. Computing the ratio of PSNR to bit rate reveals the superiority of the KTSBTC. Figure 6 shows a graph of this ratio across the possible threshold values.

Table 4. PSNR and BPP at different threshold values

Threshold	QTS-BTC		KTS-BTC	
	PSNR	BPP	PSNR	BPP
5	33.208	4.745	33.189	4.643
10	32.312	2.772	32.228	2.664
15	31.335	1.975	31.191	1.879
20	30.588	1.551	30.217	1.463
25	29.526	1.28	29.307	1.199

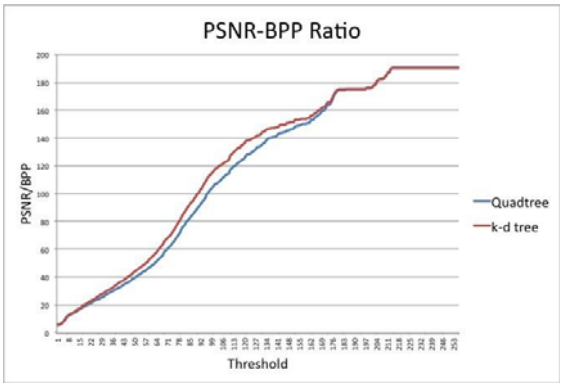


Figure 6. PSNR to BPP ratios at different threshold values using quadtree and *k*-d tree segmentation schemes

4.3 Effect of MSE Thresholding

To determine the impact of using the MSE thresholding, we implemented an MSE-thresholded QTS-BTC. To compare the results of the two QTS-BTC variants properly, it is inappropriate to look at the PSNR to BPP ratio across various threshold values, since the thresholding techniques are different. Instead, we plot the PSNR vs BPP curves. Figure 7 shows that the MSE thresholding is only minimally better.

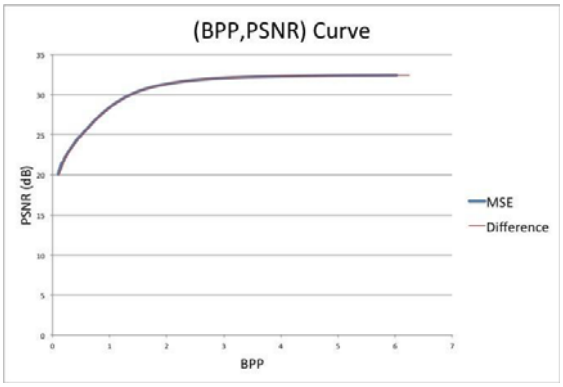


Figure 7: PSNR vs Bit Rate curves of QTS-BTC using the two different thresholding schemes

5 Conclusion

In this paper, we proposed a new BTC compression technique which we refer to as the *k*-d Tree Segmented BTC (KTS-BTC). This variant introduces two ideas: (1) the use of a *k*-d tree for segmentation and (2) the use of MSE for block activity thresholding.

The proposed algorithm was compared against the AMBTC and the state-of-the-art QTS-BTC. The superiority of KTS-BTC was established through actual implementation on a standard dataset. The results showed a significant improvement in bit rates, at the cost of a minimal reduction in PSNR values. A further investigation of the effects of the two introduced ideas showed that the improvement is mainly due to the use of *k*-d trees for segmentation.

Future studies can explore dynamically selecting a better default *k*-d tree-based initial partitioning (horizontal versus vertical), the use of bigger blocks, and the use of other block activity thresholding schemes.

References

1. C.C. Chang, Y.C. Hu, Consumer Electronics, IEEE Transactions on **44**, 1201 (1998)
2. Y. Fisher, Fractals **2**, 347 (1994)
3. B.E. Usevitch, Signal Processing Magazine, IEEE **18**, 22 (2001)
4. E.J. Delp, O.R. Mitchell, Communications, IEEE Transactions on **27**, 1335 (1979)
5. M.D. Lema, O.R. Mitchell, Communications, IEEE Transactions on **32**, 1148 (1984)
6. D. Halverson, N. Griswold, G. Wise, Acoustics, Speech and Signal Processing, IEEE Transactions on **32**, 664 (1984)
7. R.R.M. Daga, P. Fernandez, *Multi-partition Block Truncation Coding for Image Compression*, in *NCITE 2015 Proceedings of the National Conference of Information Technology Education* (PSITE, 2015), pp. 35 – 40
8. R.R.M. Daga, J.P.T. Yusiong, IJCSI International Journal of Computer Science Issues **9** (2012)
9. P. Nasiopoulos, R.K. Ward, D.J. Morse, Communications, IEEE Transactions on **39**, 1245 (1991)
10. B.C. Dhara, B. Chanda, Pattern Recognition **37**, 2131 (2004)
11. W.L. Chen, Y.C. Hu, K.Y. Liu, C.C. Lo, C.H. Wen, International Journal of Signal Processing, Image Processing and Pattern Recognition **7**, 65 (2014)
12. S. Vimala, M. Sathya, K.K. Devi, International Journal of Computer Applications **51**, 48 (2012)
13. *University of southern california - signal and image processing institue image database*, http://sipi.usc.edu/database/, Accessed: 2015-10-20