Dynamic Differential Block Truncation Coding (DDBTC) Using Bit Shift Technique

N. J. Satheesh Kumar, Research Scholar, Department of Computer Science and Research Centre, S. T. Hindu College, Nagercoil, Tamilnadu, India.

Dr. B. Ramakrishnan, Associate Professor, Department of Computer Science and Research Centre, S.T.Hindu College, Nagercoil, Tamilnadu, India.

ABSTRACT:

This paper presents a novel approach for lossless data embedding in Dynamic Differential Block Truncation Coding (DDBTC) compressed images using prediction and bit shifting techniques. DDBTC is a popular method known for its ease of implementation and low CPU cost, making it suitable for real-time processing applications. With the advancement of computer technology in data storage and digital processing, there is a growing demand for efficient image compression techniques to enhance the transmission and storage effectiveness. In this study, we focus on the bit rate, a key parameter of compression techniques that directly impacts efficiency and CPU utilization. By employing bilateral shifts and bitwise operations, we introduce a modified DDBTC scheme that resembles sub-band decomposition. The transformation process involves both right and left bit shifts, effectively reducing the number of bits required to represent the transformed image. This careful compression approach ensures minimal loss of information. Experimental results demonstrate that the proposed system achieves high image quality for the compressed embedded images. By leveraging the capabilities of DDBTC and incorporating prediction and bit shifting techniques, our method effectively enhances data embedding in a lossless manner. The improved compression efficiency and low computational overhead make it well-suited for real-time applications, where quick processing and optimal resource utilization are crucial.

Keywords, BTC, DDBTC, lossless data embedding, bit shifting, compression techniques,

1. INTRODUCTION

The increasing popularity of digital data communication and the rapid advancements in networking technologies have led to a growing demand for efficient transmission and storage of images over the Internet [6]. As a result, image compression techniques have become crucial in optimizing the effectiveness of image communication and storage. Advances in computer technology for mass data storage and digital processing have paved the way for the implementation of advanced data compression techniques. The primary goal of image compression is to convert an image into a space-efficient compressed representation. This not only reduces transmission time but also minimizes storage space requirements, leading to more efficient data transfer and cost savings.

Image compression techniques can be broadly categorized as either lossy or lossless methods. In this context, the bit rate, which represents the average number of bits per stored pixel in the image, plays a significant role as it measures the efficiency of the compression technique. Different applications may have varying requirements and priorities, leading to the selection of compression algorithms with distinct strengths and trade-offs.

Block Truncation Coding (BTC) is a popular technique for compressing grayscale images. It offers a straightforward and computationally efficient approach to reduce the size of image data while maintaining acceptable visual quality [13]. BTC works by dividing the original image into blocks, typically 8 x 8 pixels in size. Within each block, a quantization process is applied to reduce the number of gray levels. The compression process is guided by calculating the mean and standard deviation of each block, which determine the characteristics of the reconstructed or compressed blocks. One notable advantage of BTC is that the mean and standard deviation of the compressed image blocks match those of the original image, allowing for a reasonably accurate representation of the visual content. Due to its simplicity and real-time processing capabilities, BTC has found applications in scenarios where efficient transmission and storage of images are crucial[20].

Dynamic Data Block Truncation Coding (DDBTC) lies in its ability to achieve maximum compression while maintaining rapid visual inspection. Originally developed for color cell compression, DDBTC is particularly suitable for applications where image blocks contain edges or regions with significant intensity variations. By employing a combination of lossy and progressive methods, DDBTC enables efficient visual inspection[10]. This approach involves transmitting an image with reduced resolution initially, allowing for quick display in a small size, and then gradually refining the image quality until achieving perfect reconstruction in a progressive resolution transmission scheme[9]. This feature makes DDBTC beneficial for scenarios that require both high compression rates and improved image quality over time.

2. RELATED WORK

The literature on BTC and its applications demonstrates a range of research efforts focused on different aspects of the technique. These studies encompass aspects such as quantization, modification, comparison with other techniques, and optimization. Further research is necessary to explore advanced variations of BTC and hybrid approaches that leverage recent advancements in compression and coding techniques [12].

Image compression is a fundamental technique in digital image processing that aims to reduce the storage space required for images and facilitate efficient transmission over networks. BTC is a well-known lossy image compression technique that has received significant attention in the literature.

BTC operates by dividing an image into blocks and reducing the number of gray levels within each block using quantization. Several studies have focused on enhancing the quantization process in BTC to improve compression efficiency and image quality. a BTC method based on fuzzy C-means clustering, which demonstrated promising results in terms of achieving better compression performance.performed a comparative analysis of BTC with Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT). They assessed the compression ratio and image quality of each technique, providing insights into their respective strengths and limitations[15].

Researchers have also explored modifications and enhancements to BTC to overcome its limitations and improve its compression performance. Introduced an enhanced BTC method that incorporated additional processing steps [4]. Their approach demonstrated improved compression results, showcasing the potential for optimizing BTC. compared BTC with other techniques, such as Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT), highlighting their respective strengths and limitations. They evaluated the performance of these techniques in terms of compression ratio and image quality[8]18][19]. Comparisons with other compression techniques, such as Huffman coding, have also been investigated. The study conducted a comparative analysis of image compression techniques using BTC and Huffman coding, with the aim of evaluating their compression efficiency and image quality. The researchers sought to gain insights into the advantages and limitations of these techniques [9].

3. OVERVIEW MODEL

Several recent advancements have been made in the literature to improve the basic Block Truncation Coding (BTC) method. BTC is a widely used image compression technique that was initially proposed by Delp and Ritcell in 1979 [3] for grayscale image compression. It has simple encoding/decoding procedures and low computational

complexity, making it suitable for various types of images, including monochrome, moving imagery, color imagery [6], and graphics [2]-[4]. the primary drawback of the BTC scheme is its relatively low compression ratio. To address this limitation, several modifications have been introduced. One such improvement is the Absolute Moment Block Truncation Coding (AMBTC) [5], which was proposed in 1984. AMBTC aims to preserve both the sample mean and the sample first absolute central moment, and it can be applied to grayscale and color image compression. Research has shown that AMBTC provides better reconstructed image quality compared to the original BTC method.

Another variation is the Moment-Preserving Block Truncation Coding (MPBTC) scheme, which preserves the first and second moments of image blocks. MPBTC is also referred to as the traditional BTC algorithm [21]. It has been utilized in multimedia applications, such as hybrid image coding, progressive image transmission, and high bit-rate compression [6]. To further improve the BTC method, a proposed variation called Dynamic Differential Block Truncation Coding (DDBTC) has been introduced. DDBTC is an enhanced version of the traditional BTC algorithm, offering improved compression performance and better image quality. The specific details and processes of DDBTC are not mentioned in the provided information, but it is expected to provide a more comprehensive understanding of the BTC algorithm. the literature highlights various advancements in BTC, such as AMBTC, MPBTC, and the proposed DDBTC, which aim to enhance compression efficiency and reconstructed image quality[6][7][9]18]. These improvements address the limitations of the basic BTC method and make it more suitable for diverse image compression applications.

4. PROPOSED MODEL

In the Block Truncation Coding (BTC) technique, an original image of size M × N is divided into non-overlapping blocks, typically 8 × 8 pixels in size. It is possible to divide the blocks into a different size, denoted as a × b pixels, if needed. Each block is then compressed and stored as a gray-scale image, resulting in a significantly reduced buffered space compared to the original image. The compressed blocks are stored in the reduced buffered space, preserving the visual content of the image while minimizing the storage requirements. This process allows for efficient transmission and storage of the image data while maintaining a reasonable level of visual quality.

In DDBTC, each image block is processed sequentially, following a left-to-right and top-to-bottom order. The objective is to determine the positions of low mean (a) and high mean (b) within the block. The main concept of BTC is to preserve the first and second moments of each block by substituting the original pixel values with their respective high or low means. Let's define m as the total number of pixels in a block (which is equal to the block size, such as 8 × 8).and to the macro blocks (4×4)shown in figure 1. Additionally, let q represent the number of pixels within the block that have values greater than a predefined threshold.

The high mean (b) and low mean (a) can be evaluated as follows:

1. High Mean (b): It is determined by taking the average of the pixel values greater than the predefined threshold within the block. In equation (1)

   $$b = (1/q) * \Sigma(x) \quad (1)$$

   where x represents the pixel values greater than the threshold.

2. Low Mean (a): It is calculated by averaging the pixel values that are not greater than the threshold within the block. Mathematically, it can be expressed as equation (2)

   $$a = (1/(m-q)) * \Sigma(x) \quad (2)$$

   where x represents the pixel values that are not greater than the threshold.

By substituting the original pixel values with either the high mean (b) or the low mean (a) within each block, BTC achieves compression while preserving the statistical characteristics (mean and second-moment) of the original image data.

$$\bar{X} = \frac{1}{k}\sum_{i=1}^{k} x_i \qquad (3)$$

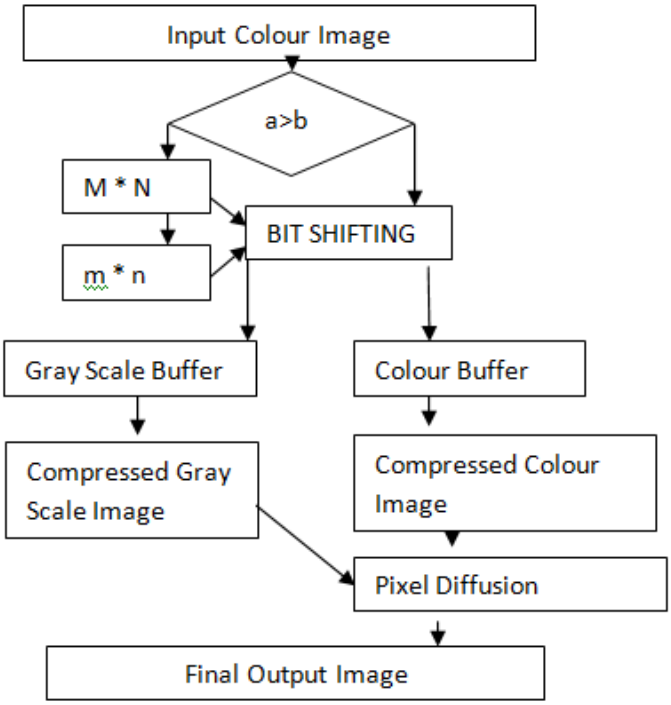


Figure 1.Flow diagram

1. Maximum Value (Xmax): It represents the highest pixel value within the block A and can be calculated by finding the maximum value among all the pixels in the block. Mathematically, it can be expressed as:

   Xmax = max(A) (4)

2. Minimum Value (Xmin): It represents the lowest pixel value within the block A and can be calculated by finding the minimum value among all the pixels in the block. Mathematically, it can be expressed as:

   Xmin = min(A) (5)

Q(X) = { b, if X ≥ (Xmax + Xmin)/2 a, if X < (Xmax + Xmin)/2 (6)

The proposed image shift coding algorithm

Step 1: Input the original image.

Step 2: Divided image blocks one by one M × N pixels.

Step 3: Divided M × N pixels blocks in 8 × 8 pixels .

Step 4: Divided m ×n pixcel macro blocks 4 × 4 pixcels

Step 5: Compute the mean of the block using Eq.(1)- Eq.(4).

Step 6: Find the Max and Min pixel value.

Step 7: Right shift the pixel value -1
Step 8: left shift the pixel value +1
Step 9: if Sum >= threshold value,
Arrange the Xmax and Xmin value in each pixel positions coordinate.
Step 10: end.

Based on the provided information, the process involves converting a color image to a grayscale image using shift operations and specific bit manipulations. Here is a breakdown of the steps involved:

Step 1: Obtain the height and width of the original image.

Step 2: Process the pixels of the image starting from the (0,0) coordinate using M/G/1 queuing. The specific details of the queuing process are not mentioned, so it would require further clarification or information.

Step 3: Convert the color image to grayscale using shift operations. The following operations are performed for each pixel:

- Extract the alpha (α), red (R), green (G), and blue (B) components from the RGB value.
- Right shift the RGB value by 24 bits and perform a bitwise AND with 0xFF to obtain the alpha value.
- Right shift the RGB value by 16 bits and perform a bitwise AND with 0xFF to obtain the red value.
- Right shift the RGB value by 8 bits and perform a bitwise AND with 0xFF to obtain the green value.
- Perform a bitwise AND with 0xFF to obtain the blue value.
- Calculate the new value using the formula: New_Value = (R * 0.299) + (G * 0.587) + (B * 0.114).
- Generate the grayscale value by combining the alpha, new value, and shifting operations: Gray_Value = (α << 24) | (New_Value << 16) | (New_Value << 8) | (New_Value).
- Store the grayscale value in the corresponding pixel position.

Step 4: Repeat the above process for each pixel in the image until reaching the end pixels. And the pixels are given in matrix formate

3	*11*	*33*	*42*	*46*	*14*	*2*	*4*
12	*16*	*22*	*66*	*34*	*86*	*89*	*111*
23	*26*	*35*	*145*	*12*	*78*	*89*	*33*
44	*65*	*3*	*65*	*50*	*101*	*89*	*67*
55	*86*	*85*	*55*	*11*	*98*	*89*	*42*
23	*121*	*5*	*64*	*3*	*12*	*12*	*65*
44	*22*	*45*	*90*	*45*	*34*	*66*	*13*
121	*122*	*23*	*22*	*34*	*12*	*75*	*49*

Figure 2 Matrix 8 * 8

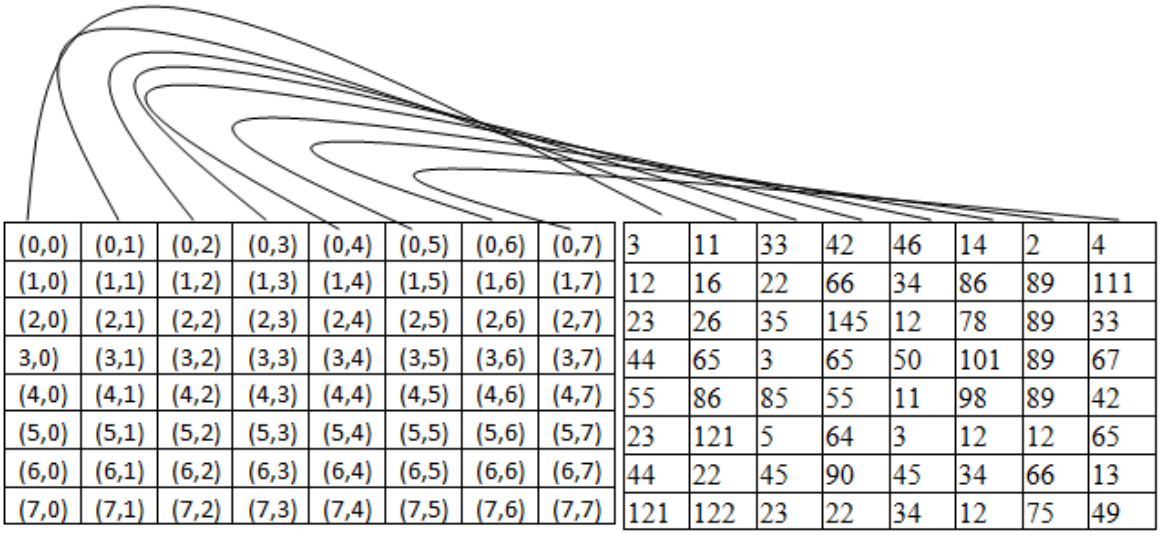
.

3	*3*	*3*	*3*	*3*	*3*	*3*	*3*
3	*3*	*3*	*122*	*3*	*122*	*122*	*122*
3	*3*	*3*	*122*	*3*	*122*	*122*	*3*
3	*122*	*3*	*122*	*122*	*122*	*122*	*122*
3	*122*	*122*	*122*	*3*	*122*	*122*	*3*
3	*122*	*3*	*122*	*3*	*3*	*3*	*122*
3	*3*	*3*	*122*	*3*	*3*	*122*	*3*
122	*122*	*3*	*3*	*3*	*3*	*122*	*3*

Figure 3. Encoded Block

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)	(0,7)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)
3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	(4,6)	(4,7)
(5,0)	(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	(5,6)	(5,7)
(6,0)	(6,1)	(6,2)	(6,3)	(6,4)	(6,5)	(6,6)	(6,7)
(7,0)	(7,1)	(7,2)	(7,3)	(7,4)	(7,5)	(7,6)	(7,7)

Figure 4. Cordinate 8 *8 Matrix



(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)	(0,7)	3	11	33	42	46	14	2	4
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)	12	16	22	66	34	86	89	111
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)	23	26	35	145	12	78	89	33
3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)	44	65	3	65	50	101	89	67
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	(4,6)	(4,7)	55	86	85	55	11	98	89	42
(5,0)	(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	(5,6)	(5,7)	23	121	5	64	3	12	12	65
(6,0)	(6,1)	(6,2)	(6,3)	(6,4)	(6,5)	(6,6)	(6,7)	44	22	45	90	45	34	66	13
(7,0)	(7,1)	(7,2)	(7,3)	(7,4)	(7,5)	(7,6)	(7,7)	121	122	23	22	34	12	75	49

Figure 5. Cordinate with encoded Block

Dot-Diffused Block Truncation Coding (DDBTC) involves the quantization of pixel values within each block and the dot diffusion process. Here is a general outline of the equations involved in DDBTC:

1. Quantization:

Calculate the block mean (mean) and standard deviation (std_dev) for each block.Define a threshold value (T) as a function of mean and std_dev to determine the quantization level. Quantize the pixel values (X) within the block based on the threshold:

- If X >= mean + T * std_dev, quantized_value = high_mean.
- If X < mean + T * std_dev, quantized_value = low_mean.

2. Dot Diffusion:

Determine the quantization error for each pixel within the block. Quantization_error = X - quantized_value. Distribute the quantization error to neighboring blocks using a dot diffusion technique (e.g., Floyd-Steinberg, Error Diffusion). Adjust the neighboring pixel values based on a specific diffusion pattern and weighted coefficients.

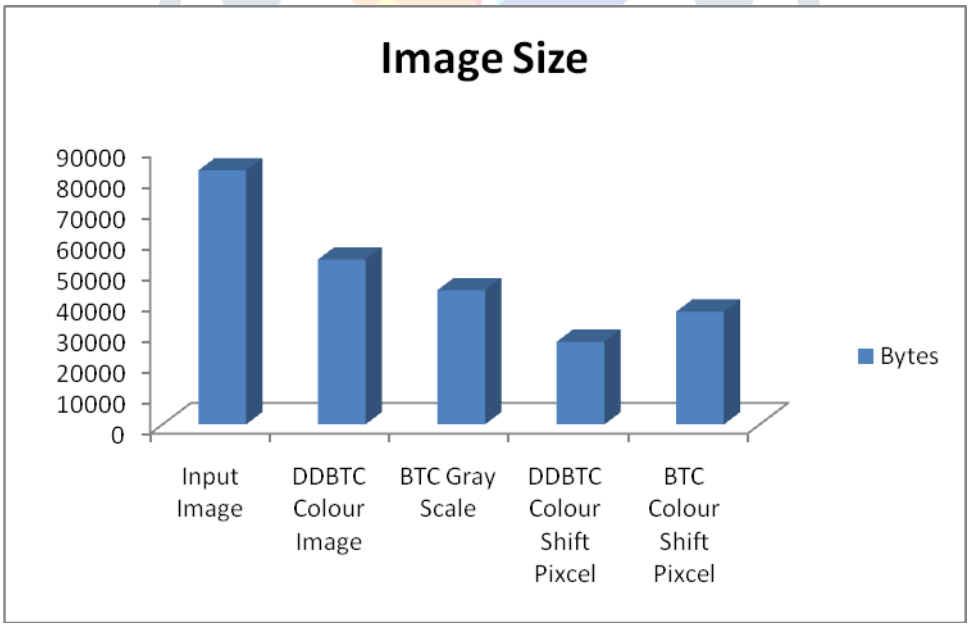
The specific equations for dot diffusion will depend on the chosen diffusion algorithm. Advanced techniques like Floyd-Steinberg or Error Diffusion employ specific coefficients and error propagation patterns to distribute the quantization errors. It's important to note that the exact equations and coefficients used in DDBTC may vary depending on the implementation and the specific requirements of the technique. The outlined equations provide a

general overview of the quantization and dot diffusion steps involved in DDBTC, but the specific details and coefficients may need to be determined based on the chosen implementation and diffusion algorithm.

5. EXPERIMENTAL RESULTS

The paper analyzes and discusses the performance of the proposed DDBTC technique. It likely evaluates the technique based on various factors such as compression ratio, image quality, and processing efficiency. The use of an optimized class matrix and diffused matrix in the evaluation process suggests that these matrices play a role in determining quantization levels and distributing quantization errors during the coding process. According to the paper, the proposed DDBTC method offers high image quality and high processing efficiency. This indicates that the technique is effective in compressing image data while maintaining a satisfactory level of visual fidelity. The details provided suggest that the paper highlights the advantages and performance of DDBTC as a compression technique for image data.

Original Image *DDBTC Compressed Colour* *Gray Scale* *DDBTC Compressed Gray*



6. CONCLUSION

This paper presents a dot-diffused-based BTC image compression technique, DDBTC, which outperforms the existing EDBTC method in terms of image quality and processing efficiency. The proposed method effectively combines the strengths of BTC to achieve excellent image quality. However, there is still potential for future work to develop even better algorithms that can further enhance compression efficiency. The findings of this paper contribute to the field of image compression and provide a foundation for future advancements in the development of more efficient and effective compression techniques.

Reference

1. Bansch, E., & Mikula, K. (1997). "A coarsening finite element strategy in image selective smoothing." Computation and Visualization in Science, 1, 53–61.
2. Bhadouria, V. S., & Dwivedi, R. (2017). "Image compression using block truncation coding." International Journal of Computer Science and Mobile Computing, 6(8), 57-61.
3. Delp, E. J., & Mitchell, O. R. (1979). "Image compression using block truncation coding." IEEE Trans. Commun., 27(9), 1335–1342.
4. Gonzales, R.C., & Woods, R.E. (2008). Digital Image Processing (3rd ed.). Pearson Education.
5. Hu, Y.C. (2003). "Low-complexity and low-bit-rate image compression scheme based on AMBTC." Optical Engineering, 42(7), 1964-1975.
6. Jadhav, S. R., & Patil, P. M. (2017). "Block truncation coding for image compression: A survey." In 2017 International Conference on Communication, Computing, and Electronics Systems (ICCCES) (pp. 661-665). IEEE.
7. Jayalakshmi, G., & Sumathi, C. P. (2018). "Image compression using enhanced block truncation coding." In 2018 International Conference on Recent Trends in Electrical, Control and Communication (RTECC) (pp. 1-5). IEEE.
8. Kaur, M., & Rani, R. (2017). "Performance evaluation of BTC with different block sizes and quantization levels for image compression." International Journal of Computer Applications, 166(1), 10-15.
9. Kuduvalli, G.R., & Rangayyan, R.M. (1992). "Performance analysis of reversible image compression techniques for high-resolution digital teleradiology." IEEE Trans. Med. Imaging, 11, 430-445.
10. Lema, M.D., & Mitchell, O.R. (1984). "Absolute moment block truncation coding and its application to color image." IEEE Transactions on Communications, 32(10), 1148-1157.
11. Ma, K. K., & Manjunath, B. S. (2000). "Texture features and learning similarity." IEEE Transactions on Image Processing, 9(11), 1920-1939.
12. Pratt, W.K. (2001). Digital Image Processing: PIKS Scientific Inside (4th ed.). Wiley-Interscience.
13. Rabbani, M., & Jones, P.W. (1991). "Digital Image Compression Techniques." SPIE Opt. Engpress, Bellingham, Washington.
14. Ramzan, N., & Saba, T. (2016). "An efficient compression algorithm for grayscale images using BTC and Huffman coding." In Proceedings of the International Conference on Frontiers of Information Technology (FIT) (pp. 221-226). IEEE.
15. Roy, A., & Bhattacharya, P. (2018). "Comparative analysis of image compression techniques using BTC and Huffman coding." International Journal of Advanced Research in Computer Science, 9(4), 30-34.
16. Sayood, K. (2000). Introduction to Data Compression (2nd ed.). Morgan Kaufmann.
17. Sayood, K. (2017). Introduction to Data Compression. Morgan Kaufmann.
18. Taubman, D., & Marcellin, M. (2002). JPEG2000: Image Compression Fundamentals, Standards and Practice. Springer.
19. Tripathi, S. C., & Mishra, D. (2016). "A comparative study of different block truncation coding techniques for image compression." In Proceedings of the International Conference on ICT for Sustainable Development (ICT4SD) (pp. 140-146). IEEE.
20. Wallace, G.K. (1992). "The JPEG Still Picture Compression Standard." IEEE Transactions on Consumer Electronics, 38(1), 18-34.
21. Wang, Q., Wang, S., & Hou, M. (2015). "Block truncation coding image compression method based on fuzzy C-means clustering." Journal of Applied Mathematics, 2015, 1-9.