Enhancement Image Compression Using BTC Algorithm

Anurag Gupta
Asst. Professor, CSE Deptt.
Aryan Institute Of Technology
Ghaziabad, India

Sachiv Kumar
Associate Professor, CSE Deptt.
Bhagwati Institute of Tech. & Sc.
Ghaziabad India

Abhishek Raja
Student, M.Tech(CSE)
DITMR,Faridabad
Haryana, India

Abstract—In this research paper Block Truncation Coding (BTC) algorithm is used for image compression. BTC algorithm is a type of lossy image compression techniques for images. It divides the original images into blocks and then uses a quantize to reduce the number of grey level in each block while maintaining the same mean and standard deviation. An improved BTC algorithm names as Enhanced Block Truncation Coding (EBTC) is used to compress the image which gives the improved quality. In EBTC algorithm image is encoded to improve the compression ratio and improve image quality compare to BTC in terms of subject visual quality.

Keywords- image, compression, BTC, AMBTC, correlation, Q level quantizer, mean, standard deviation.

I. INTRODUCTION

Block Truncation Coding (BTC) is an efficient image coding method. In computer graphics we can convert information into images by which we can understand information easily but sometimes images are big. So we can represent a data by reducing a number of bits and correlation between pixels. That is the main reason behind image compression. In image compression, information can be compress when it is redundant.

BTC is one bit adaptive moment-preserving quantizer that preserves certain statistical moments of small blocks of the input image in the quantized output. This algorithm preserves the standard mean and the standard deviation. Various methods have been proposed for image compression such as BTC and absolute moment block truncation coding (AMBTC). AMBTC preserves the higher mean and lower mean of the blocks and use this quantity to quantize output. AMBTC provides better image quality, quiet faster and simple compared to BTC.

II. BTC ALGORITHM

Block Truncation Coding (BTC) is a well-known compression algorithm proposed on March 17, 1977 in the office of O. Robert Mitchell at Purdue University during a conversation between Mitchell and his Ph.D student Edward J. Delp. The basic BTC algorithm is a lossy fixed length compression method that uses a Q level quantizer to quantize a region of image. The quantizer levels are chosen such that a number of the moments of a region in the image are preserved in the quantized output. In its simplest form, the objective of BTC is to preserve the sample mean and standard deviation of a image. The BTC algorithm has following steps:-

1. Image is divided into n x n block. The size of block could be 2x2, 4x4 and 8x8.

2. Assume that a block is containing n pixel with intensity value from p1 to pn.

3. Calculate the mean and variance of each block.

\[
\text{mean} \ p = \frac{1}{n} \sum_{i=1}^{n} p_i
\]

\[
\text{variance} \ p^2 = \frac{1}{n-1} \sum_{i=1}^{n} (p_i - \text{mean})^2
\]

\[\text{deviation} \ \alpha = \sqrt{\text{variance}}\]

4. Calculate three values \( P_{\text{max}} \), \( P_{\text{min}} \) and \( P_{\text{base}} \) for each blocks.

If\( P_{\text{max}} \geq P_{\text{base}} \) then \( P_i = P_{\text{max}} \)

Otherwise \( P_i = P_{\text{min}} \)

5. It is clear that count of \( n^+ \) pixels are greater than base values and \( n^- \) pixels less than its base value \( P_{\text{base}} \).
6. Update the 4 pixel intensity value using this formula:

\[ p_{\text{max}} = p_{\text{max}} + \alpha \sqrt{\frac{n^{+}}{n}} \quad \text{and} \quad p_{\text{min}} = p_{\text{min}} - \alpha \sqrt{\frac{n^{-}}{n}} \]

| Table 1 |
|---|---|---|---|
| 121 | 0 | 12 | 169 |
| 16 | 200 | 56 | 47 |
| 37 | 114 | 247 | 251 |
| 43 | 9 | 7 | 255 |

In this block, calculate three parameters value mean, variance and standard deviation. The calculated value of mean and standard deviation is 98.75 and 92.75 respectively. Here \( n^{+} \) and \( n^{-} \) have 7 and 9 value. Now we find \( p_{\text{max}} \) and \( p_{\text{min}} \) using the formula which mention in algorithm.

\[ p_{\text{max}} = 98.75 + 92.75 \sqrt{\frac{7}{9}} = 204.14 \]

\[ p_{\text{min}} = 98.75 - 92.75 \sqrt{\frac{7}{9}} = 16.78 \]

The rounded value of \( p_{\text{max}} \) and \( p_{\text{min}} \) is 204 and 17. Now put these value in table 2.

| Table 2 |
|---|---|---|---|
| 204 | 17 | 17 | 204 |
| 17 | 204 | 17 | 17 |
| 17 | 17 | 204 | 204 |
| 17 | 17 | 17 | 204 |

It is clear that 4x4 block is compressed into 16 bit only. So now modified block is

| Table 3 |
|---|---|---|---|
| 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 |

### III. AMBTC ALGORITHM

Lema and Mitchell proposed this algorithm which is simple and fast variant of BTC algorithm, named as Absolute Moment Block Truncation Coding (AMBTC) that preserves higher mean and lower mean of block. This algorithm is simple and easy to implement image compression. This algorithm is a lossy fixed length compression method that uses a \( Q \) level quantizer to quantize a given region of the image. This algorithm has following steps:-

Image is divided into non-overlapping blocks. The size of blocks may be 4x4, 8x8.

Calculate the average gray level of the block (4x4) as equations:

Pixels in the image block are then classified into two ranges of values which are the upper range and the lower range. The upper range is those gray levels which are greater than the block average gray level and the remaining are in the lower range. The mean of higher range \( x_{H} \) and the lower range \( x_{L} \) are calculated as:

\[ X_{H} = \frac{\sum x_{H}}{k} \]

\[ X_{L} = \frac{\sum x_{L}}{m-k} \]

where \( k \) is the number of pixels whose gray level is greater than . 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 \( x \) and “0” to represent a pixel whose gray level is less than \( x \). The encoder writes \( x_{H}, x_{L} \) and \( b \) to a file.

AMBTC has many advantages over BTC one advantage is in the case that the quantizer is used to transmit an image from transmitter to a receiver, it is necessary to compute at the transmitter the two quantities, the sample mean and the sample standard deviation for BTC and sample first absolute central moment for AMBTC. When we compare the necessary computation for deviation information, we will find that in case of standard BTC it is necessary to compute a sum of \( m \) values and each of them will be squared while in case of AMBTC it is necessary to compute the sum of these \( m \) values. Since the multiplication time is several times greater than the addition time in most digital processors, thus using AMBTC the total calculation time at the transmitter is significantly reduced.
IV. PROPOSED ALGORITHM

1. Take an image M and separate it into nxn blocks (usually 2x2, 4x4, etc).
2. Prepare a prim_num vector list which holds all prime elements such as:

   \[ \text{Vector_list} \in \{2^0 < \text{prim_num} < 2^n - 1\} \]

   Where n is number of bits used to represent any pixels in the image.
3. Calculate the mean each block.

   \[ \text{Mean}_p = \{\text{mean } b_{i} \neq \text{mean } (M)\} \]

4. Select the two prime numbers prim_num\text{max} and prim_num\text{min} from vector which just greater and smaller of mean of block. The prime variable

   \[ \alpha = (\text{prim_num}\text{max} + \text{prim_num}\text{min}) / 2 \]

5. do the following procedure for each block
6. Calculate the number of pixel n+ , n- and n* that are greater and smaller than threshold value of the block.

   \[ \text{Temp1} = \text{mean} + \alpha \sqrt{ } \]
   \[ \text{Temp2} = \text{mean} - \alpha \sqrt{ } \]

   \[
   \text{For (r=0: r<=n; r++)}
   \text{For (c=0; c<=n; c++)}
   \]

   \text{Case 1:} \quad \text{If } (b_{i}[r][c] = b_{i}[r][0])
   \text{If Pixel intensity value is greater than threshold value}
   \text{Then assign } b_{i}[r][c] = \text{temp1}
   \text{Otherwise } b_{i}[r][c] = \text{temp2}

   \text{Case 2:} \quad \text{if} (b_{i}[r][c+1] = b_{i}[r][c])
   \text{then } (b_{i}[r][c+1]) = b_{i}[r][c])

V. PERFORMANCE ANALYSIS

The above discussed algorithm has been implemented in MATLAB-7.0. In this image compression algorithm, the images “LENA”, shown in Fig.1 (a), and “BABOON” shown in fig.1(b) with size 256 were utilized to test the proposed algorithm.

After all taking an image we make a vector list of pri_num. So vector list for “LENA” image is

\[ \text{Vector_list} \in \{1 < \text{prim_num} < 255\} \]

\[

Fig.1(a) Lena
The Spatial frequencies (SFM) and values computed for the above set of images are given in table.
Test image Lena are images has low details and consequently small SFM. Small value of SFM means that image contains components in low frequency area. Test image Baboon has a lot of details and consequently large SFM. Large value of SFM means that image contains components in high frequency area. It returns that Baboon presents low redundant image, which is difficult for compression. It returns Lena presents high redundant image, which is easy for compression than Baboon.

VI. CONCLUSION

In this paper we proposed an improved compression algorithm for image to reduce the correlation and spatial redundancy between pixels of an image. Our proposed algorithm is useful to maintain the compression ratio and quality of an image. The reconstructed images have a bit rate 1.25 bpp. This corresponds to 85% compression. It has been show that the image compression using EBTC provides better image quality than image compression using BTC at the same bit rate. This algorithm was test on different gray scale images with different sizes.

REFERENCES