IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834, p- ISSN: 2278-8735.
PP 83-91
www.iosrjournals.org

Improved Block Truncation Coding Using Dot Diffusion Algorithm

Jinimol baby[1], Priyanka Udayabhanu[2]
[1]*(ECE Dept , SNGCE , India)*
[2]*(ECE Dept , SNGCE , India)*

Abstract: *Block Truncation Coding, or BTC, is a type of lossy image compression technique for grayscale images .The BTC algorithm uses a two-levels (one-bit) nonparametric quantizes that adapts to local properties of the image. How-ever, its inherent artifacts, blocking effect and false contour, caused by low bit rate configuration are the key problems. To deal with these, an improved BTC, namely dot-diffused BTC (DDBTC), is used. In this paper we improved current method of Block Truncation Coding Using Optimized Dot Diffusion. Dot-diffused BTC image compression technique which can yield excellent image quality and processing speed. But the image Quality is depending on class matrix and diffusion kernel which is calculated by co optimization algorithm. . So in our method we generated class matrix randomly, which avoids co optimization algorithm and gives an opportunity to select better class matrix. And there is no need to compute diffusion kernel for each class matrix, Floyd's and Steinberg filter weights are used as diffusion kernel matrix. we also choose median as one bit quantizer threshold. it is proven that our method is 3 times faster than DDBTC and gives better HPSNR.*

Keywords- *Block truncation coding (BTC), image compression, Dot Diffused Block Truncation Coding, Class matrix, Processing time, HPSNR.*

I. Introduction

With the continuing growth of modern communication technology, demand for image transmission and storage is increasing rapidly. Advanced in computer technology for mass storage and digital processing have paved the way for implementing advanced data compression techniques to improve the efficiency of transmission and storage of images. Applications of data compression are primarily in transmission and storage of information. Typically, a compressed image when decoded to reconstruct its original form will be accompanied by some distortion. The efficiency of a compression algorithm is measured by its data compression ability, the resulting distortion and as well by its implementation complexity. The complexity of data compression algorithms is a particularly important consideration in their hardware implementation image transmission application are in broadcast television, remote sensing via satellite, aircraft, radar, sonar, teleconferencing, computer communication, facsimile transmission.....etc. image storage is required most commonly for education, and business document, medical image used in-patient monitoring system....etc. because of their wide application, data compression and coding schemes have been of great importance in digital image processing application of data compression is also possible in the development of fast algorithms where the number of operation required to implement an algorithm is reduced by working with the compressed data.

Block truncation coding (BTC), which was proposed by Delp and Mitchell [1], is a technique for image compression. The basic concept of this technique is to divide the original image into many non-overlapped blocks, each of which is represented by two distinct values. When a BTC image is transmitted, a pair of values (2 × 8 bits/block) and the corresponding bitmap which addresses the arrangement of the two values in each block (1 bit/pixel) are required. Since a BTC image normally accompanies with annoying false contour and blocking effect, an improved scheme, namely EDBTC was developed to cope with these two issues. The dot diffusion was employed to cooperate with BTC to yield the proposed dot-diffused BTC (DDBTC) image compression technique. This method can be considered as a powerful candidate for use in most of the low power image/video codec system. Comparing with the EDBTC, DDBTC have high image quality. DDBTC can yield excellent image quality and processing speed. But the image Quality is depend on class matrix and diffusion kernel which is calculated by co optimization algorithm. . So in our proposed method we generated class matrix randomly, and there is no need to compute diffusion kernel for each class matrix, Floyd's and Steinberg filter weights are used as diffusion kernel matrix . We also choose median as one bit quantizer threshold. It is proven that our method is 3 times faster than DDBTC and gives better HPSNR as compared to block truncation coding

In this paper, the dot diffusion, i.e., [2] or [3], is employed to cooperate with BTC to yield the proposed dot-diffused BTC (DDBTC) image compression technique. Notably, the diffused matrix and class matrix are co-optimized to obtain even better image quality than that of the EDBTC which has been proved of achieving the best image quality so far, while maintaining the parallelism similar to that of ODBTC. These two properties endorse the proposed DDBTC for efficient compression applications and low power systems, e.g., low end embedded systems or handheld devices for image/video recording. One peculiar example recently we conducted is to embed the proposed scheme into a global positioning system (GPS) device which is used on a car. The low complexity advantage enables the additional coding feature using the rest CPU resource of the GPS device. Thus, the proposed method can be considered as a powerful candidate for use in most of the low power image/video codec system. Comparing with the former work [4], more theoretical analyses/explanations and more comparisons with novel image quality assessment (IQA) are provided for supporting the effectiveness of the proposed scheme.

II. Block Truncation Coding

Most image data compression techniques achieve high data compression ratio. The tradeoff between data compression remains one of the difficult problems. Maintaining high compression ratios with good image quality is possible at a more or less high computational cost. One of the main goals for image data compression is to reduce redundancy in the image block a much as possible. That is, it is very important to represent an image with as few bits as possible while maintaining good image quality. Both compression and decompression algorithms should be simple and efficient. BTC is one of the simple and easy to implement image compression algorithms.

BTC is a technique for image compression called Block. By using a finite Markov chain model, an exact closed form expression for the transient behavior of an **M/M/l** queue. BTC algorithm uses a two-level (one-bit) non parametric quantizer that adapts to local properties **of** the image. The quantizer that shows great promise is one which preserves the local sample moments. This quantizer produces good quality images that appear to be enhanced at data rates of 1.5bits/picture element. No large data storage was required. This technique uses nonparametric quantizer adaptive over local regions of the image. The image will be divided into **4** X 4 pixel blocks, and the quantizer will have 2 levels. If one uses the classical quantization design of Max which minimizes the mean square error, one must know, **Q** *priori,* the probability density function of the pixels in each block. This same knowledge is also required for the absolute error fidelity criteria of Kassam [IO]. Since in general it is not possible to find adequate density function models for typical imagery. Nonparametric quantizes for our coding schemes, that minimizes either mean square error (denoted MSE) or mean absolute error. After dividing the picture into ***n*** X ***n*** blocks *(n* = 4 for our examples), the blocks are coded individually, each into a two level signal. The levels for each block are chosen such that the first two sample moments are preserved. In the presence of many channel errors, BTC was superior to other methods of transform coding and Hybrid coding. The mean square error and mean absolute error measure cannot correlate with photo analyst's evaluations. In some cases, images with large mean square errors were evaluated higher than images with smaller mean square errors. It should be noted that BTC requires a significantly smaller computational load and much less memory.

The proposed DDBTC is an improved version of the traditional BTC algorithm, thus the traditional algorithm will be firstly introduced for a better comprehension. Given an original image of size $P \times Q$, and which is divided into many non-overlapped blocks of size $M \times N$, then each block can be processed independently and eventually represented by two values. The independent processing property yields the additional excellent parallelism advantage. To begin with, the first-, second-moment, and the corresponding variance are obtained by

$$\overline{x} = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} x_{i,j}, \qquad (1)$$

$$\overline{x^2} = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} x_{i,j}^2, \qquad (2)$$

$$\sigma^2 = \overline{x^2} - (\overline{x})^2, \qquad (3)$$

where the variable $x_{i,j}$ denotes the grayscale pixel value in a block. Since BTC is a one-bit quantizer with a threshold x to binarize the block, the block is then replaced by bitmap as defined below

$$h_{i,j} = \begin{cases} 1, & \text{if } x_{i,j} \geq \overline{x} \\ 0, & \text{if } x_{i,j} < \overline{x}' \end{cases} \qquad (4)$$

and the reconstructed result is obtained as

$$y_{i,j} = \begin{cases} b, & \text{if } h_{i,j} = 1 \\ a, & \text{if } h_{i,j} = 0' \end{cases} \qquad (5)$$

where the variable $h_{i,j}$ denotes the bitmap, which is employed to address the arranged positions of low mean (a) and high mean (b). The concept of the BTC is to preserve the first and second-moments of a block when the original value is substituted by its high or low means. Thus, the following two
equations should be maintained.

$$m\overline{h} = (m-q)\,a + qb, \qquad (6)$$
$$m\overline{h}^2 = (m-q)\,a^2 + qb^2, \qquad (7)$$

where $m = M \times N$, and q denotes the number of pixels greater than x. The high and low means can be evaluated as follows

$$a = \overline{x} - \sigma\sqrt{\frac{q}{m-q}}, \qquad (8)$$
$$b = \overline{x} + \sigma\sqrt{\frac{m-q}{q}}. \qquad (9)$$

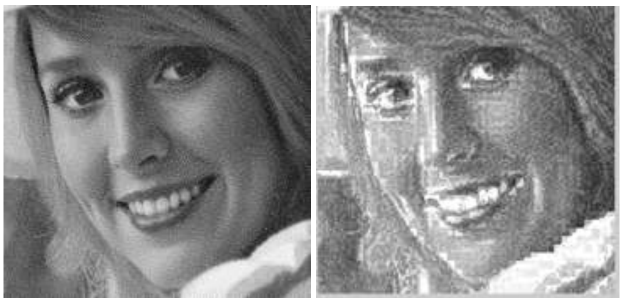
Fig 1. original image and btc image

III. Dot-Diffused Block Truncation Coding

The structure of the proposed DDBTC algorithm is similar to the traditional BTC algorithm, in which the detail comparisons between the two methods are shown in Fig. 2. The variable *mean* can be calculated by (1), and variables *a* and *b* are obtained by (8) and (9), respectively. Notably, the proposed DDBTC totally complies with the flow of the traditional BTC,

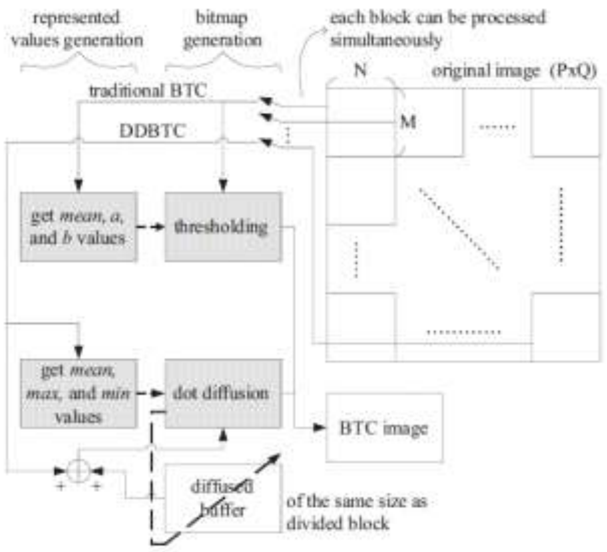
represented values generation
bitmap generation
each block can be processed simultaneously
traditional BTC
DDBTC
N
original image (PxQ)
M
get mean, a, and b values
thresholding
get mean, max, and min values
dot diffusion
BTC image
diffused buffer
of the same size as divided block

Fig. 2. algorithm comparison between the traditional block truncation coding (btc) and the proposed dot-diffused btc (ddbtc).

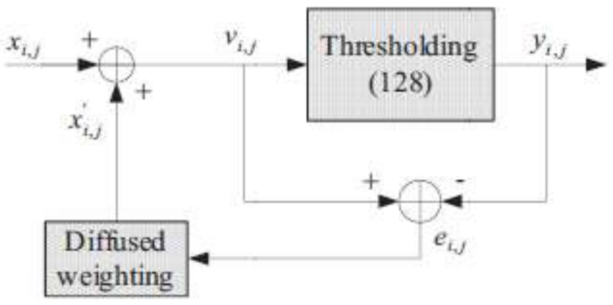

Fig. 3. dot diffusion algorithm

such as parallel processing characteristic and the substitution of the two distinct values in a block. First, the corresponding maximum and minimum are obtained for the proposed method as below:

$$x_{\max} = \max(B), \qquad (10)$$
$$x_{\min} = \min(B), \qquad (11)$$

where the vector *B* denotes a divided original block. The proposed algorithm has two main differences to that of the traditional BTC: 1) The high mean and low mean are replaced by the local maximum *(x*max*)* and minimum *(x*min*)* in a block, because the high dynamic range (*x*max − *x*min) can easily destroy the blocking effect and false contour, and 2) the manner of bitmap generation is replaced by the dot-diffused half toning as detailed below.

Suppose the original image and the divided block are of sizes *P* × *Q* and *M* × *N*, respectively, and each block can be processed independently. For each block, the processing order of pixels is defined by the class matrix as shown in Table I(a) and (c). For example, if the class matrix in Table I(a) is adopted, the original image is divided into blocks of the same size 8 × 8 as that of the class matrix. Each divided block maps to the same class matrix, and all of pixels associated with number zero in the class matrix are processed firstly. Fig.3shows the processing concept and the corresponding equations are given below.

$$v_{i,j} = x_{i,j} + x'_{i,j}, \quad \text{where } x'_{i,j} = \sum_{(m,n)\in R} \frac{e_{i+m,j+n} \times k_{m,n}}{sum}, \tag{12}$$

$$e_{i,j} = v_{i,j} - y_{i,j}, \quad \text{where } y_{i,j} = \begin{cases} 0, & \text{if } v_{i,j} < 128, \\ 255, & \text{if } v_{i,j} \geq 128 \end{cases} \tag{13}$$

where the variable $x_{i,\,j}$ denotes the current input grayscale value, variable $x_{i,\,j}$ denotes the diffused error accumulated from neighboring processed pixels, and variable $v_{i,\,j}$ denotes the modified grayscale output. The variable $y_{i,\,j}$ denotes the binary output in the bitmap, and variable $e_{i,\,j}$ denotes the difference between the modified grayscale output $v_{i,\,j}$ and the binary output $y_{i,\,j}$. The variable *km,n* denotes the diffused weighting, and *R* denotes the support region of diffused weighting, with a suggested size of 3 × 3 as in Knuth’s [2] and Mese-Vaidyanathan’s dot diffusion [3]. The diffused weighting can be represented as:

$$\begin{bmatrix} k_{-1,-1} & k_{-1,0} & k_{-1,1} \\ k_{0,-1} & x & k_{0,1} \\ k_{1,-1} & k_{1,0} & k_{1,1} \end{bmatrix}. \tag{14}$$

The variable *x* denotes the pixel currently being processed. Nota associates to the numbers in the class matrix with a greater value than its own associated value. These are the pixels that have yet to be thresholded. The variable *sum* is the summation of the diffused weights corresponding to those unprocessed pixels.bly, the error can only diffuse to neighboring pixels that :

$$sum = \sum_{m=-1}^{1} \sum_{n=-1}^{1} \begin{cases} k_{m,n}, & \text{if } c_{i+m,j+n} > c_{i,j} \\ 0, & \text{if } c_{i+m,j+n} < c_{i,j}, \end{cases} \tag{15}$$

where the variable *ci, j* denotes the coefficient value in the class matrix. Fig. 5 shows an example which includes four independent blocks split by the thick lines

TABLE I
TWO TRAINED CLASS MATRICES WITH DIFFERENT SIZES, 8 × 8 AND 16 × 16, AND THE CORRESPONDING DIFFUSED MATRICES

(a) 8 × 8 Class Matrix

42	47	46	45	16	13	11	2
61	57	53	8	27	22	9	50
63	58	0	15	26	31	40	30
10	4	17	21	3	44	18	6
14	24	25	7	5	48	52	39
20	28	23	32	38	51	54	60
19	33	36	37	49	43	56	55
12	62	29	35	1	59	41	34

(b) Diffused Matrix for the 8 × 8 Class Matrix

0.27163	1	0.27163
1	x	1
0.27163	1	0.27163

(c) 16 × 16 Class Matrix

6	7	20	10	53	55	66	87	137	142	143	144	172	122	175	164
3	9	23	50	60	51	65	74	130	145	138	148	179	180	214	221
0	14	24	37	67	79	96	116	39	149	162	198	12	146	224	1
15	26	43	28	71	54	128	112	78	159	177	201	208	223	225	242
22	4	48	32	94	98	80	135	157	173	113	182	222	226	227	16
40	85	72	83	104	117	163	133	168	184	200	219	244	237	183	21
47	120	101	105	123	132	170	176	190	202	220	230	245	235	17	41
76	73	127	109	97	134	178	181	206	196	229	231	246	19	42	49
103	99	131	147	169	171	166	203	218	232	243	248	247	33	52	68
108	107	140	102	185	167	204	217	233	106	249	255	44	45	70	69
110	141	88	75	192	205	195	234	241	250	254	38	46	77	5	100
111	158	160	174	119	215	207	240	251	252	253	61	62	93	84	125
151	136	189	199	197	216	236	239	25	31	56	82	92	95	124	114
156	188	191	209	213	228	238	29	36	59	64	91	118	139	115	155
187	194	165	212	2	13	30	35	58	63	90	86	152	129	154	161
193	210	211	8	11	27	34	57	18	89	81	121	126	153	150	186

(d) Diffused Matrix for the 16 × 16 Class Matrix

0.305032	1	0.305032
1	x	1
0.305032	1	0.305032

The class matrix in Table I(a) is employed. In this example, the central position with number 34 is the current processing position, and those numbers in gray represent the processed pixels. The arrows represent the possible diffusing directions. Since the pixels with numbers smaller than 34 are processed, the total number of diffusing directions is four, and thus the variable *sum* = *k*−1,−1 + *k*−1,0 + *k*0,−1 + *k*1,1 in this case. Notably, the error not only can diffuse to the self block, but also can diffuse to its neighboring blocks.

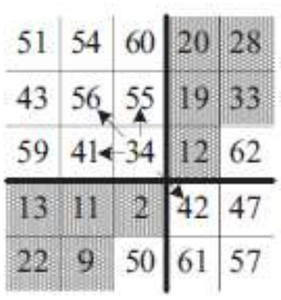

Fig. 4. demonstrated example of diffusion between blocks using the class matrix in Table I(a).

Since the dot diffusion in DDBTC is different from the traditional one, such as the pixel depth of the represented values, the threshold and the two replaced values in (13) are modified as below

$$e_{i,j} = v_{i,j} - y_{i,j}, \quad \text{where } y_{i,j} = \begin{cases} x_{\min}, & \text{if } v_{i,j} < \bar{x} \\ x_{\max}, & \text{if } v_{i,j} \geq \bar{x}' \end{cases} \quad (16)$$

where the variables x, xmin, and xmax are defined in (1), (10), and (11), respectively. For the current stage, DDBTC cannot provide better image quality than that of EDBTC for the following two reasons: 1) The class matrix and the diffused matrix employed in traditional dot diffusion [2], [3] are designed for two-tone output, while the DDBTC generates multi-tone output when the bitmap is replaced with the maximum and minimum values of the block. 2) The threshold employed in the traditional dot diffusion is a fixed number 128. This is different from that used in DDBTC, in which adaptive mean values are employed for each block. Consequently, we proposed a co-optimization procedure for the class matrix and diffused matrix to further improve image quality. Herein, the human-visual peak signal-to-noise ratio (HPSNR; named HVS-PSNR in [5]) is adopted as the cost function for IQA as defined in (17)

$$\text{HPSNR} = 10\log_{10} \times \frac{P \times Q \times 255^2}{\sum_{i=1}^{P}\sum_{j=1}^{Q}\left[\sum_{(m,n)\in R} w_{m,n}\left(x_{i+m,j+n} - y_{i+m,j+n}\right)\right]^2}, \tag{17}$$

where $x_{i,j}$ and $y_{i,j}$ denote the original image and reconstructed image of size $P \times Q$, respectively; the variable *wm,n* denotes the filter to simulate the low-pass property of HVS. To define this filter, the Gaussian filter with two parameters defined in [5], the standard deviation 1.3 and support region R of size 9 × 9, is adopted.

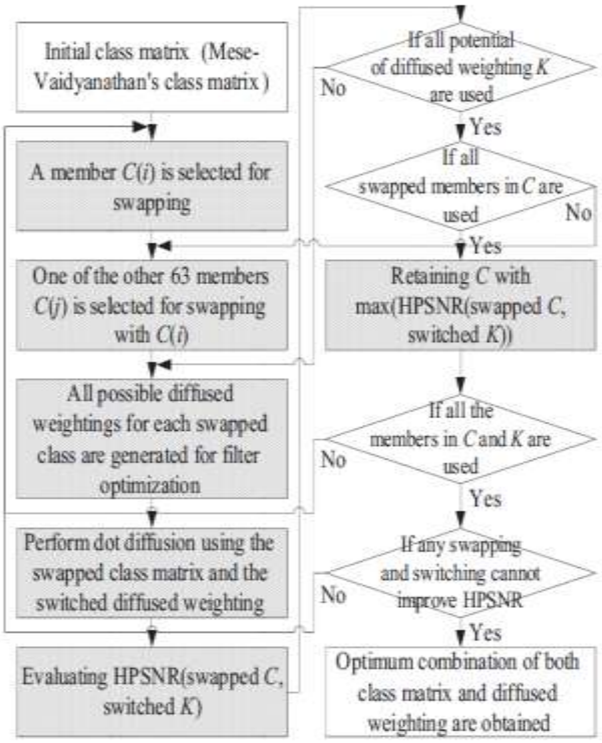
Initial class matrix (Mese-Vaidyanathan's class matrix)
A member C(i) is selected for swapping
One of the other 63 members C(j) is selected for swapping with C(i)
All possible diffused weightings for each swapped class are generated for filter optimization
Perform dot diffusion using the swapped class matrix and the switched diffused weighting
Evaluating HPSNR(swapped C, switched K)
If all potential of diffused weighting K are used
No
Yes
If all swapped members in C are used
No
Yes
Retaining C with max(HPSNR(swapped C, switched K))
If all the members in C and K are used
No
Yes
If any swapping and switching cannot improve HPSNR
No
Yes
Optimum combination of both class matrix and diffused weighting are obtained

Fig. 5. class matrix and diffused matrix co-optimization algorithm.

According to the experimental results, slight modification of the parameter will not significantly change the results. Fig.5 shows the procedure of the co-optimization algorithm, and the detailed steps are explained below.

Step 1. The Mese-Vaidyanathan's class matrix [3] is adopted as the initial Class matrix (C) in this optimization.

Step 2. Suppose the coefficients in the class matrix are collocated as a 1D sequence. Successively swap each member *C(i)* in the class matrix with one of the other 63 members *C(j)* (suppose the size of the class matrix is 8 × 8), where *i* _= *j* .

Step 3. Generate all potential diffused weightings *km,n K* by adjusting 10−6 within a range of 0 to 1. During the generation of diffused weighting, the nearest vertical and horizontal weights are fixed as 1, and the other four diagonals are kept at the same value.

Step 4. Evaluate the average HPSNR using (6) with eight natural grayscale images of size 512 × 512, Lena, Mandrill, Peppers, Milk, Tiffany, Airplane, Lake, and Shuttle, with swapped class matrices (Step 2) and switched diffused weightings (Step 3).

Step 5. The successive combination of both swapped class matrices and switched diffused weightings is capable of achieving the highest DDBTC image quality by max(HPSNR (swapped *C*, switched *K*). These are then employed as the new class matrix and diffused matrix candidate.

Step 6. Select another member *C(i)*, and perform Steps 2 to 5.

Step 7. If all swapped class matrices and switched diffused weightings cannot improve HPSNR, then terminate this optimization. Otherwise, perform Steps 2 to 6.

Table I shows the two optimized class matrices of different sizes, 8 × 8 and 16 × 16, and their corresponding diffused matrices. For the sake of the co-optimization treatment, the proposed DDBTC can achieve even better image quality than that of the EDBTC and ODBTC. Notably, the bigger size in class matrix leads to less parallel advantage. For example, the whole bitmap of an image can be obtained in 64 unit times when a class matrix of size 8 × 8 is employed, while when the size of the class matrix grows to 16 × 16, then 256 unit times are required to obtain the whole bitmap of an image. Hence, there is a tradeoff between the size of the class matrix and processing efficiency.

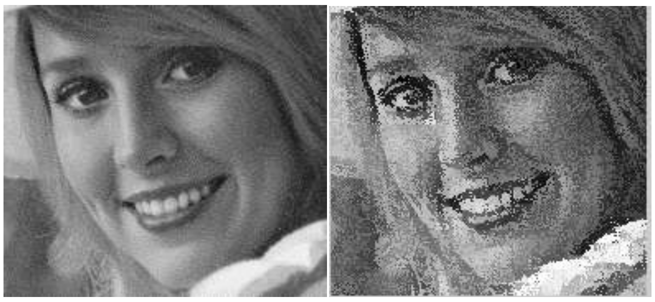
Fig 6. original image and DDBTC image

IV. Conclusion

This paper presented a dot-diffused-based BTC image compression technique which can yield excellent image quality (even superior to that of the EDBTC_F), processing speed (faster than that of the EDBTC_F about 696× for block of size 8×8, and around 164× for block of size 16×16), and artifact free results (inherent blocking effect and false contour artifacts of the traditional BTC) simultaneously. The performance can be attributed to the use of the inherent parallelism of the dot diffusion and the proposed co-optimization procedure over the class matrix and diffused matrix. As documented in the experimental results, the proposed DDBTC is superior to EDBTC in terms of image quality and processing efficiency, and has much better image quality than that of the ODBTC. Thus, the proposed DDBTC has important values and impacts in prospective highly efficient or low powerless compression communication and related applications. Although the proposed DDBTC provides satisfactory image quality, future work can be put to develop a better optimization algorithm to avoid class matrix and diffused matrix being trapped in local optimal states.

Acknowledgements

I would like to thank Asst.Prof. Priyanka Udayabhanu for her valuable guidance. I also like to thank entire ECE department for their cooperation.

References

[1] E. J. Delp and O. R. Mitchell, "*Image compression using block truncation coding," IEEE Trans. Commun., vol. 27, no. 9,pp. 1335–1342, Sep. 1979.*

[2] D. E. Knuth, "*Digital halftones by dot diffusion," ACM Trans. Graph., vol. 6, no. 4, pp. 245–273, Oct. 1987.*

[3] M. Mese and P. P. Vaidyanathan, "*Optimized halftoning using dot diffusion and methods for inverse halftoning," IEEE Trans. Image Processing, vol. 9, no. 4, pp. 691–709, Apr. 2000.*

[4] J. M. Guo and Y. F. Liu, *"Improved block truncation coding using optimized dot diffusion," in Proc. IEEE Int. Symp. Circuits Syst., May–Jun. 2010, pp. 2634–2637.*

[5] J. M. Guo and Y. F. Liu, "*Joint compression/watermarking scheme using majority-parity guidance and halftoning-based block truncation coding," IEEE Trans. Image Process., vol. 19, no. 8, pp. 2056–2069, Aug. 2010.*

[6] P. Li and J. P. Allebach, "*Block interlaced pinwheel error diffusion," J. Electron. Imag., vol. 14, no. 2, p. 023007, Apr.–Jun. 2005.*

[7] D. R. Halverson, N. C. Griswold, and G. L. Wise, "*A generalized block truncation coding algorithm for image compression," IEEE Trans.Acoust., Speech, Signal Process., vol. 32, no. 3, pp. 664–668, Jun. 1984.*

[8] V. Udpikar and J. P. Raina, "*Modified algorithm for block truncation coding of monochrome images," Electron. Lett., vol. 21, no. 20,pp. 900–902, Sep. 1985.*

[9] Q. Kanafani, A. Beghdadi, and C. Fookes, "*Segmentation-based image compression using BTC-VQ technique," in Proc. Int. Symp. Signal Process. Appl., vol. 1. Jul. 2003, pp. 113–116.*

[10] S. Horbelt and J. Crowcroft, "*A hybrid BTC/ADCT video codec simulation bench," in Proc. 7th Int. Workshop Packet Video, Mar. 1996, pp. 18–19.*

[11] M. Kamel, C. T. Sun, and G. Lian, *"Image compression by variable block truncation coding with optimal threshold," IEEE Trans. Signal Process., vol. 39, no. 1, pp. 208–212, Jan. 1991.*

[12] L. G. Chen and Y. C. Liu, *"A high quality MC-BTC codec for video signal processing," IEEE Trans. Circuits Syst. Video Technol., vol. 4, no. 1, pp. 92–98, Feb. 1994.*

[13] Ulichney, *Digital Halftoning*. Cambridge, MA, USA: MIT Press, 1987.

[14] D. J. Lieberman and J. P. Allebach, "*A dual interpretation for direct binary search and its implications for tone reproduction and texture quality," IEEE Trans. Image Process., vol. 9, no. 11, pp. 1950–1963, Nov. 2000.*

[15] J. M. Guo, "*Improved block truncation coding using modified error diffusion," Electron. Lett., vol. 44, no. 7, pp. 462–464, Mar. 2008.*

[16] R. W. Floyd and L. Steinberg, "*An adaptive algorithm for spatial gray scale," in Int. Symp. Soc. Inf. Display, Dig., 1975, pp. 36–37.*

[17] J. F. Jarvis, C. N. Judice, and W. H. Ninke, "*A survey of techniques for the display of continuous-tone pictures on bilevel displays," in Proc. Comp. Graph. Image Process., vol. 5. 1976, pp. 13–40.*

[18] P. Stucki, *"MECCA: A multiple-error correcting computation algorithm for bilevel image hardcopy reproduction," IBM Res. Lab., Zurich, Switzerland, Res. Rep. RZ1060, 1981.*

[19] J. N. Shiau and Z. Fan, *"A set of easily implementable coefficients in error diffusion with reduced worm artifacts," Proc. SPIE, vol. 2658, pp. 222–225, Jan. 1996.*