Color image compression based on block truncation coding using pattern fitting principle

Bibhas Chandra Dhara[a,*], Bhabatosh Chanda[b]

[a]Department of Information Technology, Jadavpur University, Salt Lake Campus, Kolkata 700098, India
[b]Electronics and Communication Sciences Unit, Indian Statistical Institute, Kolkata 700108, India

Received 17 May 2006; received in revised form 10 November 2006; accepted 19 December 2006

Abstract

This paper describes a color image compression technique based on block truncation coding using pattern fitting (BTC-PF). High degree of correlation between the RGB planes of a color image is reduced by transforming them to $O_1O_2O_3$ planes. Each O_i plane ($1 \leqslant i \leqslant 3$) is then encoded using BTC-PF method. Size of the pattern book and the block size are selected based on the information content of the corresponding plane. The result of the proposed method is compared with that of several BTC based methods and the former is found superior. Though this method is a spatial domain technique, it is also compared with JPEG compression method, which is one of most popular frequency domain techniques. It is found that the performance of the proposed method is a little inferior to that of the JPEG in terms of quality of the reconstructed image. Decoding time is another important criterion where the compressed image is decoded frequently for various purposes. As the proposed method requires negligible decoding time compared to JPEG, the former is preferred over the latter in those cases.

Keywords: Block truncation coding; Color image compression; Image fidelity index; Pattern fitting; Reversible color transformation

1. Introduction

The objective of an image compression technique is to represent an image with smaller number of bits without introducing appreciable degradation of visual quality of decompressed image. These two goals are mutually conflict in nature. In true-color digital image, each color component (R,G,B) is usually quantized with 8-bits, so a color is specified by 24 bits per pixel. To transmit or store a color image huge bandwidth or storage space is required. To reduce storage space or transmission cost image compression is the solution. In gray-scale image there is a high correlation between neighbor pixels. In color image, in addition to this, there is also a high correlation between the color components. The straightforward way to compress color image is to compress each of the color components separately using gray-scale compression technique. However, this class of method does not use the correlation between the color planes. Another alternative is to convert RGB to less correlated YIQ, YUV, YC_bC_r, etc. and then to apply gray-scale image compression method on each of these components. In all these Y** representation of a color image, Y stands for the intensity at a pixel and other two components contain the color information. As human visual system cannot distinguish all those 16 million colors represented by 24 bits, to compress a color image the color components are compressed more than the corresponding luminance component.

In addition to high compression ratio and good quality of the reconstructed image, it is desired that the compression technique should satisfy four characteristics. First, since decoding (or reconstruction) is done more frequently than the compression (which is usually done once), reconstruction method should be computationally as less expensive as possible. Second, image processing techniques may be applied directly on the compressed image itself, and the image features may be extracted directly from it. Third, progressive transmission of compressed image must be allowed so that partially transmitted image may be viewed and transmission of undesired image may be terminated. And fourth, quality and compression

ratio may be controlled. Though transform domain compression techniques (e.g. DCT, wavelet-based methods) are usually superior to the spatial domain techniques in terms of compression ratio and quality, the latter techniques are well ahead compared to the former ones if these additional requirements are considered. Block truncation coding and vector quantization are two widely used spatial domain compression techniques. Block truncation coding (BTC) algorithm developed by Delp and Mitchell [1] is a simple block-based spatial domain image compression technique for gray-scale image. This is a lossy method and this method preserves lower order moments of gray level. The absolute moment BTC [2], upper and lower mean BTC [3], adaptive BTC [4] are proposed to improve the quality. There are several other modifications of BTC method to improve the quality. A prediction-based BTC method which gives a good image quality at low bit rate with little extra computational cost in encoding/decoding procedure is proposed in Ref. [5]. A further reduction in bit rate is achieved by pattern fitting [6]. All the above-mentioned BTC methods consider only two levels in bit pattern. Three level BTC algorithm [7,8] are also used to improve the quality of reconstructed images.

These variations of BTC techniques can be applied separately on each of the color planes of a color image. As there is a high degree of correlation among the color planes, this can be exploited to reduce the bit rate further. A single-bit-map BTC (SBBTC) method, developed by Wu and Coll [9], exploits the inter-color correlation by using a single bit map to quantize all three color planes. Here two color vector $C_1 = (R_1, G_1, B_1)$ and $C_2 = (R_2, G_2, B_2)$ and a bit pattern are needed to reconstruct the block. In SBBTC method bit rate is 4 bits/pixel if block size is 4×4, while the bit rate for color BTC (CBTC), i.e., BTC method applied to each color plane separately, is 6 bits/pixel. Kurita and Otsu [10] suggested another SBBTC method which generates the single bit-map by employing an adaptive one-bit vector quantization based on the principle score of each pixels in the block. The SBBTC method proposed by Tai et al. [11] uses Hopfield neural network (HNN) to generate the single-bit-map of a block. Yang et al. [12] suggests another single bit map method, called CICMPBTC method, based on moment preserving principle. However, depending on the uniformity of color components a block is defined as k-spectral (with $0 \leqslant k \leqslant 3$) if 3-$k$ of the three color components (R,G,B) of the block are nearly uniform. If a component is uniform then mean of that component is used in both C_1 and C_2. Using spectral index, number of components in C_2 may vary from 0 to 3, which improves the compression efficiency. Again CICMPBTC method is modified to CICMPBTC* considering: (i) quantization of each color component from 8 bit to 5 bit, and (ii) by selecting one from a set of predefined 64 bit maps. Similar idea of using predefined set of bit-maps is also used in Ref. [13].

It is obvious that applying BTC to each color plane results in high quality reconstructed image, but the bit rate is also high. A much lower bit rate with a little sacrifice in visual quality can be achieved by block truncation coding using pattern fitting (BTC-PF) [6]. The results of BTC-PF on "Lena", a gray-scale image of size 512×512, is PSNR $= 31.59$ db with 0.64 bpp, while that of conventional BTC [1] method is 32.89 db with 2 bpp. Block truncation based algorithms are basically used for coding the still images, but these methods can also be used as a part in video compression and multimedia communication techniques [14]. In this study, a color image compression method based on BTC-PF is proposed. In this method an image is transformed from RGB to $O_1O_2O_3$ system. Each O_i $(1 \leqslant i \leqslant 3)$ plane is compressed using BTC-PF method with different parameter values. Compression ratio is enhanced by considering some blocks as smooth (determined by the variance in intensity) and by quincunx sampling method. Entropy coding lowers bit rate further. The remainder of this paper is organized as follows. In Section 2, basic strategy of BTC-PF method is described. Proposed method for color image compression using BTC-PF is presented in Section 3. Experimental results and comparative study are shown in Section 4. Finally, concluding remarks are given in Section 5.

2. BTC-PF method

In BTC method, an image to be compressed is divided into non-overlapping blocks of size $n \times n$. Usually n is taken to be integer power of 2. Compression is achieved by representing each block by Q different gray values corresponding to a Q-level pattern of size $n \times n$ which is obtained by a partitioning based on block statistics. In conventional BTC [1] the number of level Q is 2. Hence, in BTC method for each image block one Q-level pattern of size $n \times n$ and Q different gray values are required to reconstruct the image block. In BTC-PF method, instead of determining the Q-level pattern based on the block statistics, it is selected from a set of, say, M, predefined Q-level patterns. The pattern should satisfy some image quality in the best way with respect to the candidate block. Thus, at the time of reconstruction of a block, the index of selected pattern and Q gray values are sufficient. As the method selects a pattern from a small set of predefined patterns to represent the block, the quality of the reconstructed image is, in general, little lower than that of the conventional BTC. However, this little sacrifice in PSNR earns a huge gain in bit rate. In BTC-PF method there are two basic steps: (i) selection of best pattern and (ii) determining Q gray values. Thus, in essence, BTC-PF method quantizes the intensity pattern of a block in terms of a predefined pattern and Q gray values ($Q \ll n^2$).

2.1. Selection of best fit pattern

The method of selection of best pattern for an image block B is as follows. For an image block B, let pixels are $x_1, x_2, \ldots, x_{n^2}$ and the corresponding pixel intensities $f(x_i)$. Available patterns in the given pattern book are, say, $P_1, P_2, \ldots, P_M$ of size $n \times n$ and levels present in each of the patterns are represented by t where $0 \leqslant t \leqslant Q-1$, i.e., any pattern $P_j = p_{j0} \cup p_{j1} \cup \cdots \cup p_{j(Q-1)}$ $(j \in \{1, 2, \ldots, M\})$ such that $p_{js} \cap p_{jt} = \phi$ if $s \neq t$ and p_{jt} is the collection of pixel coordinates having level t in P_j. In other words, $p_{jt} = \{x_i | P_j(x_i) = t\}$. Here image block B is fit to these patterns P_j, $j = 1, 2, \ldots, M$, in least-square-error sense and the pattern, which fits best, is selected. If image

block B is tried to fit with the pattern P_j, then square-error due to this is

$$e_j = \sum_{t=0}^{Q-1} e_{jt}, \tag{1}$$

where

$$e_{jt} = \sum_{x_i \in p_{jt}} (f(x_i) - \mu_t)^2, \tag{2}$$

$$\mu_t = \frac{1}{|p_{jt}|} \sum_{x_i \in p_{jt}} f(x_i), \tag{3}$$

where $|p_{jt}|$ is the cardinality of the set p_{jt}, i.e., the number of pixels having level t in the jth pattern. Finally, index I of the best fit pattern is the one for which square-error is minimum, i.e.,

$$I = \arg\left\{ \min_{j \in 1,2,\ldots,N} \{e_j\} \right\}. \tag{4}$$

Now if $\max\{\mu_t\} - \min\{\mu_t\}$ of the best fit pattern P_I is less than a predefined threshold, the corresponding image block is treated as smooth block. A smooth block is represented by an additional index not considered in the pattern set. To reconstruct the smooth block, only over all block mean μ is required, where $\mu = \sum_{t=0}^{Q-1} (\mu_t * |p_{It}|)/n^2$. Otherwise, index I of the selected pattern and corresponding Q means $\{\mu_t : t = 0, 1, \ldots, Q-1\}$ are necessary to reconstruct the block. Thus, compared to the conventional BTC, BTC-PF method quantizes the bit patterns by P_js and the gray values present in the block by μ_ts.

In this method to represent the Q-level pattern only $\log_2 M$ bits are required rather than $\lceil \log_2 Q \rceil n^2$ bits, where $\lceil x \rceil$ represents smallest integer greater than x. Usually, $\log_2 M \ll \lceil \log_2 Q \rceil n^2$ which leads to significant compression. Here Q-level patterns may be designed manually according to expected nature of edges and lines in the image blocks or by some suitable clustering technique that fits the present application. In the present work we have adopted latter approach, which is described in the next section.

2.2. Design of pattern book

Designing the pattern book for BTC-PF system is the most crucial task. In fact, quality of reconstructed image as well as the compression ratio is highly dependent on the pattern book. To generate the pattern book for encoding a particular color plane a collection of large number of such planes is first extracted from the color images. The selected plane of each image is divided into non-overlapping $n \times n$ blocks. The intensity of each block is transformed in such a way that average intensity of the block becomes zero and the variance becomes unity. Such transformation is needed to ensure that all the blocks having similar intensity pattern must belong to same cluster.

Then a clustering algorithm is applied on all these blocks for M number of classes. In this work we have used *K-means algorithm* [15] for the purpose. It is well known that the final clusters produced by K-means algorithm depend on the initialization of cluster centers. Here we have used M data points selected by the *maximin distance algorithm* [15] as the initial cluster centers to maximize initial inter-class distance.

Intensity $f(x_i)$ $(i = 1, 2, \ldots, n^2)$ of each of these M prototypes corresponding to M cluster is then thresholded to obtain a Q-level pattern. This requires $(Q-1)$ threshold values $\{\tau_t : 1 \leqslant t \leqslant Q-1\}$ which are obtained through exhaustive search minimizing the error

$$E = \sum_{t=0}^{Q-1} \sum_{\tau_t < f(x_i) \leqslant \tau_{t+1}} (f(x_i) - \mu_{t+1})^2, \tag{5}$$

where

$$\mu_{t+1} = \frac{1}{|\tau_t < f(x_i) \leqslant \tau_{t+1}|} \sum_{\tau_t < f(x_i) \leqslant \tau_{t+1}} f(x_i), \tag{6}$$

where $\tau_0 = \min\{f(x_i)\} - 1$ and $\tau_Q = \max\{f(x_i)\}$. It should be noted that though integer numbers are used as levels in the pattern, they are just labels and have no arithmetic value.

3. Proposed method

In a color image high correlation exists among R, G and B planes, so a high compression can be achieved by exploiting the inter-plane as well as the spatial correlations and the code redundancies. In the proposed method the inter-plane redundancy is reduced by converting RGB to a less correlated triplet. The spatial redundancy is reduced by block quantization using BTC-PF method (as described in Section 2) and the code redundancy by entropy coding using, say, Huffman code. Let us call the proposed method as CBTC-PF method.

3.1. Image encoding

The proposed color image compression method is composed of the above-mentioned steps as detailed below. To reduce the inter-plane redundancy following color transformation is applied.

3.1.1. RGB to $O_1O_2O_3$ conversion

There are standard color conversions like RGB to YIQ, YUV, YC_bC_r, etc. where YIQ, etc. have less inter-correlation among the triplet than RGB. Among these triplets, YIQ is the most popular in color image transmission and is used in color TV broadcasting. However, since the data in RGB as well as transformed domain is considered to be integer, the afore-mentioned transformations are, in a sense, lossy transformations. So in CBTC-PF method RGB to $O_1O_2O_3$ transformation [16,17] is used which is known to be lossless. It is experimentally found that better compression (i.e., higher PSNR and lower bpp) can be achieved using $O_1O_2O_3$ compared to YIQ (see Section 4). RGB to $O_1O_2O_3$ conversion is given by

$$O_1 = \left\lfloor \frac{R+G+B}{3} + 0.5 \right\rfloor, \tag{7}$$

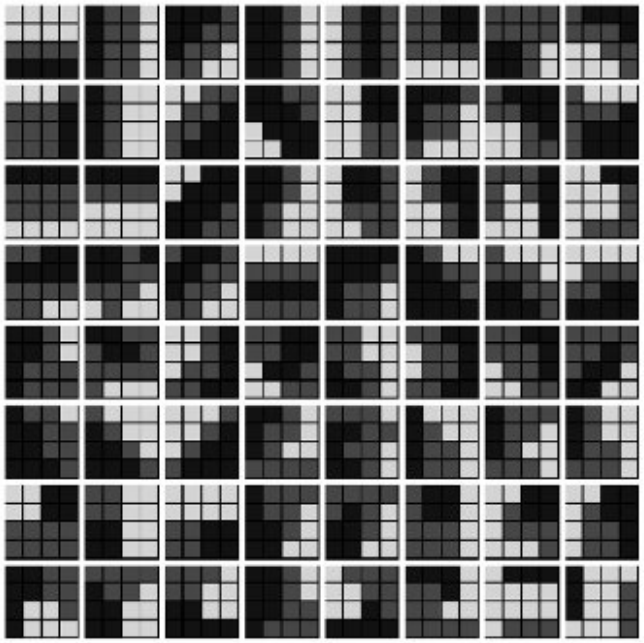

Fig. 1. 3-Level patterns used to define the gray-level pattern of image block of O_1 plane.

$$O_2 = \left\lfloor \frac{R-B}{2} + 0.5 \right\rfloor, \tag{8}$$

$$O_3 = B - 2G + R \tag{9}$$

and the corresponding inverse transformation is

$$B = O_1 - O_2 + \left\lfloor \frac{O_3}{2} + 0.5 \right\rfloor - \left\lfloor \frac{O_3}{3} + 0.5 \right\rfloor, \tag{10}$$

$$G = O_1 - \left\lfloor \frac{O_3}{3} + 0.5 \right\rfloor, \tag{11}$$

$$R = O_1 + O_2 + O_3 - \left\lfloor \frac{O_3}{2} + 0.5 \right\rfloor - \left\lfloor \frac{O_3}{3} + 0.5 \right\rfloor, \tag{12}$$

where $\lfloor x \rfloor$ stands for largest integer not exceeding x. O_1 stands for intensity or luminance component, while O_2 and O_3 together represent chrominance at each pixel. Among these O_1 contains major information followed by O_2 and O_3. For example, when "Lena" image is transformed from RGB to $O_1O_2O_3$, the entropy of O_1 plane is 5.05, and that of O_2 and O_3 planes are 4.21 and 3.39, respectively. This suggests allotment of more bits for O_1 plane compared to O_2 and O_3. This requirement is reflected in setting the parameters of pattern book. To define pattern book for O_1 plane, parameters Q, n and M are set to 3, 4 and 64, respectively; whereas, for image planes O_2 and O_3 parameters are set to 2, 4 and 16. Accordingly, the pattern book used for O_1 is shown in Fig. 1 and another pattern book for O_2 and O_3 planes is shown in Fig. 2.

3.1.2. Block quantization: O_1 plane

To encode the O_1 plane of a color image the plane is divided into 4×4 (i.e., $n = 4$) non-overlapping blocks. Being the intensity of the pixels O_1 values are spatially correlated. To exploit

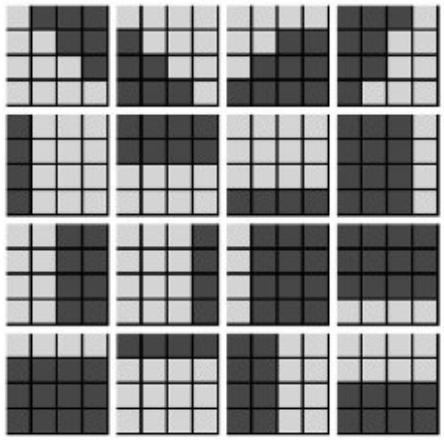

Fig. 2. 2-Level patterns used to define the gray-levels of 4×4 sub-blocks generated from O_2 and O_3 planes.

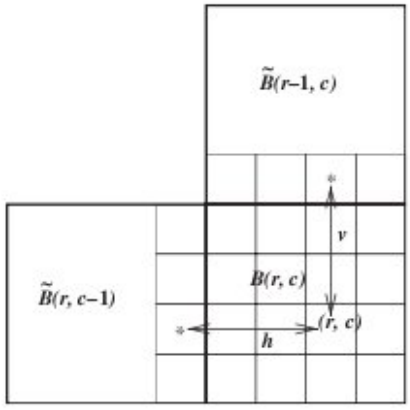

Fig. 3. Estimation of $B(r, c)$ using information from vertical and horizontal '*'-marked pixels of neighboring blocks.

this spatial redundancy, an image block $B(r, c)$ is first estimated as $\hat{B}(r, c)$ using information from already reconstructed neighboring blocks $\tilde{B}(r, c-1)$ and $\tilde{B}(r-1, c)$ (see Fig. 3) as

$$\hat{B}(r, c) = \frac{v}{v+h} val_h + \frac{h}{v+h} val_v, \tag{13}$$

where v (respectively, h) is the distance between (r, c) and '*' on the same column (respectively, row), and val_v and val_h are the values at the vertical and horizontal '*'-marked position as shown in Fig. 3. The estimation error is

$$B_e(r, c) = B(r, c) - \hat{B}(r, c). \tag{14}$$

The residual block $B_e(r, c)$ thus obtained is actually coded by the BTC-PF method. It should be noted that the blocks on the first row and first column of the image plane are estimated using the left or the top block only as the case may be. The top-left block is quantized straightaway using the same pattern book. Since the residual block $B_e(r, c)$ contains a significant number of zero values and some +ve and −ve values, to determine the pattern book for the residual blocks of the O_1 plane we have chosen $Q = 3$. Secondly, if the number of patterns in the pattern book increases the quality of the reconstructed image increases

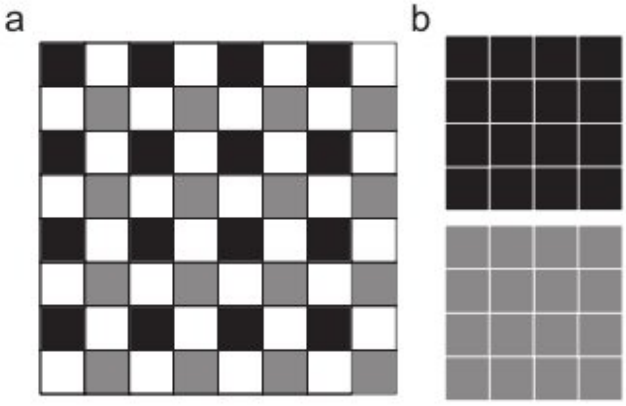

Fig. 4. Two 4×4 sub-blocks derived from 8×8 block by Quincunx sampling method: (a) a block of size 8×8, (b) two blocks of size 4×4 are formed with the pixels taken from alternate row and column marked by different color in (a).

but the bits per pixels increases, and vice versa. So to make a compromise we have chosen $M = 64$. The pattern used in our experiment is shown in Fig. 1.

3.2. Block quantization: O_2 and O_3 plane

Both O_2 and O_3 represent the chrominance components of original pixel. Both the planes contain less information than O_1 plane. So these two planes are compressed with lower bit rate. Here, each plane is divided into 8×8 (say $B_8(r, c)$) non-overlapping blocks. If the variance of $B_8(r, c)$ is less than certain threshold (T) block is considered smooth and only block mean (μ) is required to reconstruct the block. Otherwise, $B_8(r, c)$ is split into two sub-blocks $B_{41}(r, c)$ and $B_{42}(r, c)$ by quincunx sampling as shown in Fig. 4. Fig. 4(a) shows the original 8×8 block where sampled pixels are shaded with black or gray. Black pixels form the 4×4 block $B_{41}(r, c)$ and gray pixels $B_{42}(r, c)$. Each sub-block is encoded by BTC-PF with $Q = 2$ by using a pattern book of 16 patterns that are shown in Fig. 2.

3.3. Data encoding

The encoded string obtained from the above steps contains only the index of the selected pattern and different mean values. For example, for a block of O_1 plane either $(0, \mu)$ [for smooth block] or (I, μ_1, μ_2, μ_3) [for non-smooth block] are sent to the decoder to reconstruct the block. The blocks of O_1 plane encoded by BTC-PF are basically error blocks and so the pixel values of the blocks vary from -255 to $+255$. BTC-PF does not provide any ordering among μ_1, μ_2 and μ_3, and equal number of bits has to be spent to represent these values. But, if μ_1, μ_2 and μ_3 are arranged as μ'_1, μ'_2 and μ'_3 such that $\mu'_1 \leqslant \mu'_2 \leqslant \mu'_3$ then (μ_1, μ_2, μ_3) can be represented as $(\mu'_1, \mu'_2 - \mu'_1, \mu'_3 - \mu'_2)$ without any loss. Then a better compression can be achieved by entropy coding. The mapping from (μ_1, μ_2, μ_3) to (μ'_1, μ'_2, μ'_3) may be defined by using some extra bits, called *ordering index* (OI) denoted by I_0. An example of ordering index and the corresponding mapping from (μ_1, μ_2, μ_3) to (μ'_1, μ'_2, μ'_3) are given in the first three columns of Table 1. Hence, for a block of O_1 plane the encoder gives out either $(0, \mu)$ or $(I, I_0, \mu'_1, \mu'_2 - \mu'_1, \mu'_3 - \mu'_2)$.

In case of O_2 and O_3 planes, for any 8×8 block, either $(0,\mu)$ [for smooth block] or $(I_1, I_2, \mu_{11}, \mu_{12}, \mu_{21}, \mu_{22})$ [for non-smooth block] are given out by the block quantizer. Allowing a little error, instead of $(I_1, I_2, \mu_{11}, \mu_{12}, \mu_{21}, \mu_{22})$, $(I_1, I_2, \mu''_{11}, \mu''_{12}, \mu''_{21}, \mu''_{22})$ may be used [6] where $\mu''_{11} = \lfloor(\mu_{11} + \mu_{12})/2\rfloor$, $\mu''_{12} = \lfloor(\mu_{11} - \mu_{12})/2\rfloor$, $\mu''_{21} = \lfloor(\mu_{21} + \mu_{22})/2\rfloor$ and $\mu''_{22} = \lfloor(\mu_{21} - \mu_{22})/2\rfloor$. Hence, for a 8×8 block of O_2 and O_3 planes, either $(0, \mu)$ or $(I_1, I_2, \mu''_{11}, \mu''_{12}, \mu''_{21}, \mu''_{22})$ are given out by the encoder. To achieve higher compression, these μ values as well as the pattern indices (i.e., I) are coded by entropy coding. In our experiment we have adopted Huffman coding scheme using separate codebooks for separate items.

3.4. Image decoding

The encoding string for a block of a O_1 plane is either $(0,\mu)$ or $(I, I_0, \mu'_1, \mu'_2 - \mu'_1, \mu'_3 - \mu'_2)$. For a block of O_2 and O_3 planes, it is either $(0,\mu)$ or $(I_1, I_2, \mu''_{11}, \mu''_{12}, \mu''_{21}, \mu''_{22})$. These encoding strings indicate that the decoding methods of O_1 plane and O_2 (O_3) planes are similar but not exactly same. Now we discuss the decoding methods for O_1 and $O_2(O_3)$ planes separately.

In case of O_1 plane, if the first component of the encoding string of a block is 0, then the block is reconstructed by block mean (μ) only. Otherwise, the I_0 value and μ'_1, μ'_2 and μ'_3 are used to obtain μ_1, μ_2 and μ_3 using the mapping defined in the last column of Table 1. Finally, the block of the O_1 plane is reconstructed based on the index I of the selected pattern and these μ values. This particular block is either an original block (if it is the top-left one) or a residual block. In case of residual

Table 1
Mapping between (μ_1, μ_2, μ_3) and (μ'_1, μ'_2, μ'_3) using OI

OI	Relation between μ_1, μ_2, μ_3	Mapping from μ_i to μ'_j	Mapping from μ'_j to μ_j
00	$\mu_1 \leqslant \mu_2 \leqslant \mu_3$	$\mu'_1 = \mu_1,\ \mu'_2 = \mu_2,\ \mu'_3 = \mu_3$	$\mu_1 = \mu'_1,\ \mu_2 = \mu'_2,\ \mu_3 = \mu'_3$
01	$\mu_1 \leqslant \mu_3 \leqslant \mu_2$	$\mu'_1 = \mu_1,\ \mu'_2 = \mu_3,\ \mu'_3 = \mu_2$	$\mu_1 = \mu'_1,\ \mu_2 = \mu'_3,\ \mu_3 = \mu'_2$
100	$\mu_2 \leqslant \mu_1 \leqslant \mu_3$	$\mu'_1 = \mu_2,\ \mu'_2 = \mu_1,\ \mu'_3 = \mu_3$	$\mu_1 = \mu'_2,\ \mu_2 = \mu'_1,\ \mu_3 = \mu'_3$
101	$\mu_2 \leqslant \mu_3 \leqslant \mu_1$	$\mu'_1 = \mu_2,\ \mu'_2 = \mu_3,\ \mu'_3 = \mu_1$	$\mu_1 = \mu'_3,\ \mu_2 = \mu'_1,\ \mu_3 = \mu'_2$
110	$\mu_3 \leqslant \mu_1 \leqslant \mu_2$	$\mu'_1 = \mu_3,\ \mu'_2 = \mu_1,\ \mu'_3 = \mu_2$	$\mu_1 = \mu'_2,\ \mu_2 = \mu'_3,\ \mu_3 = \mu'_1$
111	$\mu_3 \leqslant \mu_2 \leqslant \mu_1$	$\mu'_1 = \mu_3,\ \mu'_2 = \mu_2,\ \mu'_3 = \mu_1$	$\mu_1 = \mu'_3,\ \mu_2 = \mu'_2,\ \mu_3 = \mu'_1$

we first estimate the block using Eq. (14) and then the intensity block is reconstructed by adding residue to the estimated block.

The decoding method of O_2 and O_3 plane is little bit different from O_1 plane. Like O_1 plane, if the first component of encoding string is 0 then the block (B_8) is reconstructed straightaway using the block mean (μ) only. Otherwise, using (I_1, μ''_{11}, μ''_{12}) and (I_2, μ''_{21}, μ''_{22}) two blocks B_{41} and B_{42} are reconstructed following the BTC-PF method [6]. Finally, B_8 is reconstructed by filling the alternate pixels in a manner that is reverse of quincunx sampling followed by interpolation of unfilled pixels (i.e., white pixels shown in Fig. 4(a)) using nearest known pixel values.

It may be recalled that we claim CBTC-PF method incur low decoding cost. To justify the claim let us analyze CBTC-PF decoding algorithm to determine its computational cost as follows. To reconstruct image plane in CBTC-PF method, for each block, corresponding index of the selected pattern and μ values are required. For O_1 plane, to reconstruct a 4×4 block ($\tilde{B}$) following steps are needed.

- Reconstruction of error block (B'_e), depending on index value, requires at most two additions.
- Estimation of block ($\hat{B}$) from neighbors $\tilde{B}$ (as shown in Fig. 3) requires 32 multiplication and 16 additions. However, since there are only 11 different weights and 255 different values (*val*) (see Eq. (13)) are possible, in actual implementation table look up method is used and so only 16 additions are required.
- Finally, for $\tilde{B} = \hat{B} + B'_e$, 16 additions are required.

For O_2 and O_3 plane, to reconstruct a 8×8 block ($\tilde{B}$) of a plane following steps are performed.

- If the block is smooth no operation is needed.
- For non-smooth 8×8 block:
  - Two 4×4 blocks are reconstructed each of which needs two additions only.
  - Remaining 32 pixel values are determined by interpolation. Let each interpolation needs C number of computation (where C equals to at most three additions and two bit-shift operations). Thus the total is 32C.

Thus, in CBTC-PF method to reconstruct a 16×16 color block, in worst case, 576 additions and 256C (for interpolation) operations are required.

4. Experimental results

In this section, the result of proposed method (CBTC-PF) is presented and compared with five other well-known techniques. These methods are applied on a number of 24-bit color images of different sizes. Eight of these images are shown in Fig. 5. The size of "Lena", "Peppers", and "Airplane" is 512×512 and "Couple", "House", "Zelda" and "Girl" is of 256×256 and "Lab" is of 480×480. Images of different sizes are considered in the experiment as the detail density, i.e., detail per unit area, varies with the image size. Usually smaller images have higher detail density and the performance of the compression technique is inferior for the images with higher detail density. The sizes of the pattern books for O_1 and $O_2(O_3)$ planes are 64 and 16, respectively. The quality and bit rate can be controlled by changing the threshold value T on block variance and the size of pattern book. However, here we have varied only the value of T and the corresponding effects on O_1 and $O_2(O_3)$ planes are reported in Tables 2 and 3, respectively. These results suggest to set $T = 2$ for O_1 plane and $T = 6$ for O_2 and (O_3) planes. Let us recall that we have claimed in Section 3.1.1 that better performance can be achieved with $O_1O_2O_3$ system compared to YIQ, etc. This may be supported by Table 4 where performance of $O_1O_2O_3$ and YIQ are compared in terms of PSNR and bpp with same values of the parameters.

Table 5 presents the comparison of results of CBTC-PF, conventional color BTC (CBTC), single-bit-map BTC (SBBTC) [9,11], CICMPBTC* [12] and JPEG [18,19]. Here compression ratio is measured in terms of bpp and the image quality in terms of PSNR and visual fidelity index [20]. The bpp and PSNR may be defined, respectively, as

$$\text{bpp} = \frac{\text{size of compressed color image in bits}}{\text{number of pixels}} \tag{15}$$

and

$$\text{PSNR} = 10 * \log_{10} \frac{255^2 * 3}{\text{MSE}(R) + \text{MSE}(G) + \text{MSE}(B)}. \tag{16}$$

Image fidelity metric (*FI*) [20] quantifies the appearance of a processed image relative to the original image as perceived by human observer. This metric is defined as a combination of three factors: loss of correlation, luminance distortion, and contrast distortion. Let $X = \{x_i | i = 1, 2, \ldots, N\}$ and $Y = \{y_i | i = 1, 2, \ldots, N\}$ be the original and processed image data. The fidelity index (*FI*) is then given by

$$FI = \frac{\sigma_{xy}}{\sigma_x \sigma_y} \frac{2\overline{x}\,\overline{y}}{\overline{x}^2 + \overline{y}^2} \frac{2\sigma_x \sigma_y}{\sigma_x^2 + \sigma_y^2}, \tag{17}$$

where

$$\overline{x} = \frac{1}{N} \sum_{i=1}^{N} x_i, \quad \overline{y} = \frac{1}{N} \sum_{i=1}^{N} y_i,$$

$$\sigma_x^2 = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \overline{x})^2, \quad \sigma_y^2 = \frac{1}{N-1} \sum_{i=1}^{N} (y_i - \overline{y})^2,$$

$$\sigma_{xy} = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \overline{x})(y_i - \overline{y}).$$

The dynamic range of *FI* is $[-1, 1]$. The maximum value 1 occurs only when the processed image is identical with the original image and -1 when $y_i = 2\overline{x} - x_i$. The first component of Eq. (17) is the correlation coefficient between X and Y, which has dynamic range $[-1, 1]$. The second component measures the closeness of the mean luminance of both images and lies in $[0, 1]$. The last component represents how similar is the contrast of both the images and again lies in $[0, 1]$. To characterize an

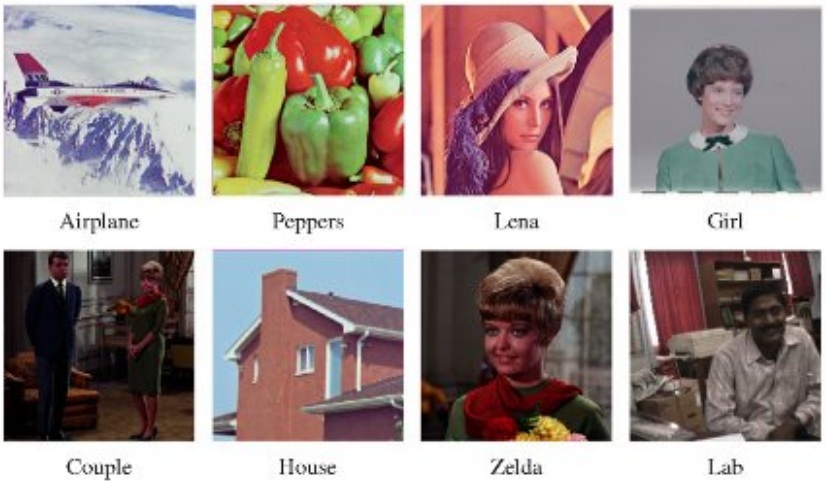

Fig. 5. Some of the original images used in the experiment.

Table 2
The effect of different threshold values (T) on block variance of O_1 plane

Images	$T=0$		$T=1$		$T=2$		$T=3$	
	No. of smooth blocks	PSNR	No. of smooth blocks	PSNR	No. of smooth blocks	PSNR	No. of smooth blocks	PSNR
Total number of blocks 16384								
Lena	12	35.25	520	35.23	6810	34.90	10 937	34.21
Peppers	2	35.50	43	35.50	3420	34.94	9200	33.55
Airplane	656	34.80	3897	33.97	9275	31.86	11 143	31.18
Total number of blocks 4096								
Couple	251	35.61	1011	35.51	1983	34.86	2694	33.84
House	186	35.19	1093	35.11	1455	34.91	2423	32.67
Zelda	20	34.78	325	34.77	1734	34.40	2578	33.77
Girl	404	37.05	2254	36.92	3106	36.53	3365	36.00
Total number of blocks 14400								
Lab	225	38.63	3395	38.39	8371	36.96	10 935	35.16
Average	2.73%	35.85	20.41%	35.68	49.44%	34.92	69.03%	33.80

image, these properties are measured locally over the image and finally combine them into single value called FI. Here, a window of size 8×8 is considered for local measurement, which slides over the entire image. Thus

$$FI = \frac{1}{L}\sum_{j=1}^{L} FI_j,$$

where L is the number of possible positions of the window and FI_j represents the fidelity index for the jth location of the window. To obtain the fidelity index of a color image, this method can be applied to each color plane individually, and combine them through a (weighted) mean. Suppose a color image is defined in $l\alpha\beta$ system, then the color fidelity index FI_{color} may be defined as

$$FI_{color} = \sqrt{w_l FI_l^2 + w_\alpha FI_\alpha^2 + w_\beta FI_\beta^2}, \quad (18)$$

where w_l, w_α, and w_β represents the fidelity factors. In our experiment we set $w_l = w_\alpha = w_\beta = \frac{1}{9}$.

To evaluate the performance of both CBTC and SBBTC methods block size is set to 4×4. In CBTC method, conventional BTC [1] is employed on each of the three color planes separately. The bit rate in CBTC method is constant and is 6 bpp. In practice, CBTC is not a popular method. It is presented here only to give an idea of quality of color image compressed directly by BTC. In the simplest implementation of SBBTC method, to define the bit map for the color blocks, K-means clustering algorithm with $K = 2$ is employed on 16 vectors corresponding to 16 pixels of a 4×4 block. Each vector has three components. Thus, in SBBTC method, for any 4×4 block $(16 + 2 \times 8 \times 3)$ bits are required to reconstruct the color block. A modified SBBTC method [11] using vector quantized color vectors achieves PSNR values 30.03 and 29.90 db for "Lena" and "Airplane" images, respectively, for a fixed bpp 2.0. The quality (in terms of PSNR as well as

Table 3
The effect of different threshold values (T) on block variance of O_2 and O_3 plane

Images	Plane	$T=3$		$T=4$		$T=5$		$T=6$		$T=7$	
		No. of smooth blocks	PSNR	No. of smooth blocks	PSNR	No. of smooth blocks	PSNR	No. of smooth blocks	PSNR	No. of smooth blocks	PSNR
Total number of blocks 4096											
Lena	O_2	1313	35.60	2408	35.44	2810	35.29	3133	35.14	3396	34.93
	O_3	2997	39.54	3408	39.28	3625	39.04	3770	38.80	3859	38.59
Peppers	O_2	495	33.56	1367	33.53	2379	33.43	2827	33.30	3065	33.17
	O_3	1892	34.43	2400	34.36	2767	34.25	3012	34.11	3174	33.98
Airplane	O_2	2467	37.40	2765	37.29	3029	37.11	3218	36.90	3359	36.67
	O_3	2859	38.73	3132	38.57	3311	38.39	3496	38.09	3663	37.72
Total number of blocks 1024											
Couple	O_2	432	36.23	617	36.04	732	35.80	818	35.52	864	35.32
	O_3	858	41.75	931	41.47	962	41.24	982	41.01	997	40.72
House	O_2	372	36.38	629	36.20	731	36.03	782	35.88	820	35.69
	O_3	726	37.92	788	37.80	843	37.58	889	37.26	930	36.92
Zelda	O_2	343	34.33	487	34.26	654	34.11	744	33.95	806	33.81
	O_3	712	38.89	837	38.71	895	38.47	931	38.24	958	38.01
Girl	O_2	825	42.34	870	42.05	920	41.47	956	40.93	979	40.49
	O_3	938	46.74	967	46.16	1004	44.89	1015	44.40	1019	44.16
Total number of blocks 3600											
Lab	O_2	2141	37.28	2748	36.94	3065	36.64	3242	36.40	3346	36.19
	O_3	3107	40.89	3315	40.65	3433	40.40	3484	40.16	3514	40.14
Average		61.59%	38.25	73.99%	38.05	81.85%	37.76	86.49%	37.51	89.59%	37.28

Table 4
Comparative results of proposed method in $O_1O_2O_3$ and YIQ domains

Images	$O_1O_2O_3$		YIQ	
	PSNR	bpp	PSNR	bpp
Image size 512 × 512				
Airplane	30.36	1.04	30.16	1.08
Peppers	30.15	1.50	29.63	1.60
Lena	31.93	1.17	31.63	1.20
Image size 256 × 256				
Girl	35.13	0.60	35.12	0.65
Couple	32.44	1.00	32.24	1.05
House	31.79	1.20	31.35	1.31
Zelda	31.31	1.12	31.17	1.17
Image size 480 × 480				
Lab	33.64	0.93	33.48	0.95
Average	32.09	1.07	31.84	1.13

FI) of the reconstructed images of CBTC-PF method is little lower than that of the CBTC and SBBTC methods, but the former attains a huge compression (approximately 6 times and 4 times, respectively) compared to that of the latter ones. Compared to CICMPBTC and CICMPBTC* methods the proposed is method is superior in terms of both PSNR and bpp and comparable in terms of *FI*. The reconstructed images of the CBTC-PF method are shown in Fig. 6. It was mentioned earlier (see Section 1) that transform (frequency) domain methods are usually superior to the spatial domain methods in terms of PSNR and bpp. JPEG now-a-days is the most popular frequency domain compression technique employing DCT. From the results reported in Table 5, the performance of CBTC-PF and JPEG are comparable though, strictly speaking, JPEG is little superior to CBTC-PF. On an average CBTC-PF achieves PSNR 32.09 db, 1.07 bpp with $FI = 0.6202$; while these figures for JPEG are 32.85 db, 1.02 bpp and 0.6594. It may be noted that *FI* for CBTC-PF, CICMPBTC and JPEG are less than that of CBTC and SBBTC. This is because of the first term of Eq. (17) which stands for correlation between original X and the reconstructed image Y. The value of the first term is high if $X(i) < X(j)$ implies $Y(i) \leqslant Y(j)$, which is true for BTC and CBTC but not for other cases. However, though not always, this is true for most of the pixels in SBBTC. So in case of CBTC-PF, CICMPBTC and JPEG, the lower value of correlation reduces the overall value of *FI*.

It should also be noted that CBTC-PF and JPEG belong to two different classes of compression techniques. In JPEG method dequantization and inverse DCT (IDCT) are the major steps for decoding. To analyse the computational cost during decoding time of JPEG method, here we only consider IDCT operation. Straightforward implementation of DCT on n data requires n^2 multiplications and $n(n-1)$ additions. However, because of popularity of DCT, various fast algorithms for DCT are proposed. Sung et al. [21] have shown that the quality of reconstructed image in JPEG method depends on the type of fast DCT algorithm used. Loeffler et al. [22] proposed one of the fastest methods which takes 11 multiplications and 29

Table 5
Experimental results of CBTC-PF method and some other methods on test images

Images	CBTC-PF			CBTC			SBBTC			CICMPBTC			CICMPBTC*			JPEG		
	PSNR	bpp	FI	PSNR	bpp	FI	PSNR	bpp	FI	PSNR	bpp	FI	PSNR	bpp	FI	PSNR	bpp	FI
Image size 512 × 512																		
Airplane	30.36	1.04	0.6068	32.12	6	0.8815	32.40	4	0.8133	30.60	2.65	0.6409	28.05	1.63	0.6521	31.46	0.90	0.6601
Peppers	30.15	1.50	0.6021	32.38	6	0.8758	31.91	4	0.7388	28.83	3.34	0.6338	26.86	1.98	0.5837	30.47	1.47	0.6244
Lena	31.93	1.17	0.6295	32.79	6	0.8827	32.63	4	0.7870	31.89	3.06	0.6848	29.15	1.83	0.6360	32.76	1.03	0.6564
Image size 256 × 256																		
Girl	35.13	0.60	0.4114	34.60	6	0.8480	34.79	4	0.7188	34.17	2.26	0.4400	30.32	1.42	0.5871	36.85	0.62	0.4834
Couple	32.44	1.00	0.6919	33.08	6	0.8917	33.21	4	0.8372	32.33	3.06	0.7414	29.70	1.83	0.7063	33.02	0.94	0.7367
House	31.79	1.20	0.5742	32.28	6	0.8622	32.30	4	0.7279	29.17	3.05	0.6247	27.55	1.83	0.6318	31.34	1.24	0.5894
Zelda	31.31	1.12	0.6763	32.37	6	0.8887	32.16	4	0.8038	31.01	3.12	0.7060	28.64	1.87	0.6794	32.06	1.00	0.6983
Image size 480 × 480																		
Lab	33.64	0.93	0.7690	33.75	6	0.9168	32.92	4	0.8745	32.08	2.97	0.7943	30.32	1.79	0.7708	34.84	0.94	0.8268
Average	32.09	1.07	0.6202	32.87	6	0.8809	32.79	4	0.7876	31.26	2.94	0.6582	28.82	1.77	0.6559	32.85	1.02	0.6594

Fig. 6. The reconstructed images due to proposed method.

additions for 1-D 8-point DCT. Hence, for a 16 × 16 color block (four 8 × 8 O_1-blocks, one 8 × 8 O_2-block and one 8 × 8 O_3-block) 1056 multiplications and 2784 additions are required. To generate a 16 × 16 $O_2(O_3)$ block, from a reconstructed 8 × 8 $O_2(O_3)$ block, interpolation method is used. Hence, 384C operations are required for a 16 × 16 color block. In Section 3.4 we have shown that, in worst case, CBTC-PF method requires 576 additions and 256C operations to reconstruct a color block of same size. Moreover, in CBTC-PF method, setting $T = 6$ gives more than 86% smooth blocks of O_2 and O_3 planes (see Table 3) and setting $T = 2$ gives almost 50% smooth blocks of O_1 plane (see Table 2). Hence, the decoding time for CBTC-PF method is practically negligible compared to that for JPEG.

5. Conclusions

A color image compression technique based on BTC with pattern fitting is presented. In the proposed CBTC-PF method, to reduce inter-plane, redundancy, (R,G,B) is transformed to ($O_1O_2O_3$) using lossless transformation as it is seen that the algorithm performs better in O_1,O_2,O_3 domain compared to other well-known color domain, e.g., YIQ. We have selected the size of the blocks, size of pattern books and the number of levels in pattern depending on the entropy of O_i ($1 \leqslant i \leqslant 3$) plane. The performance of the proposed method is compared with Color BTC, single-bit-map BTC, CICMPBTC* in terms of bpp, PSNR and fidelity index (*FI*). It is found that the proposed method is superior to the said methods. The performance

of CBTC-PF method is little inferior than JPEG in terms of PSNR, though bit-rate and *FI* is more or less same. The computational cost at the decoding phase of CBTC-PF method is negligible compared to that of JPEG, which is one of the requirements for the image that are frequently downloaded or decoded. Progressive transmission is possible as low resolution image can be obtained from only the block means available in the compressed data stream of proposed method. Various image processing and feature extraction algorithm can be applied directly on the compressed data as the spatial correspondence is retained in it. For example, image segmentation by gray level thresholding by computing μ values with threshold, edge detection [23], histogram computation based on μ and block pattern, contrast intensification by gray level mapping, etc. can be implemented straightaway.

Acknowledgment

We gratefully acknowledge the source of images and JPEG code used in our experiment. The images are taken from http://sipi.usc.edu/database/. We use the JPEG software of The Independent JPEG Group and which can be found at ftp://ftp.uu.net/graphics/jpeg/.

References

[1] E.J. Delp, O.R. Mitchell, Image compression using block truncation coding, IEEE Trans. Commun. 27 (1979) 1335–1342.
[2] M.D. Lema, O.R. Mitchell, Absolute moment block truncation coding and applications to color images, IEEE Trans. Commun. 32 (1984) 1148–1157.
[3] V.R. Udpikar, J.P. Raina, A modified algorithm for block truncation coding of monochrome images, Electron. Lett. 21 (1987) 900–902.
[4] L. Hui, An adaptive block truncation coding algorithm for image compression, in: Proceedings of ICASSP90, vol. 4, 1990, pp. 2233–2236.
[5] Y.C. Hu, Predictive moment preserving block truncation coding for gray-level image compression, J. Electron. Imaging 13 (2004) 871–877.
[6] B.C. Dhara, B. Chanda, Block truncation coding using pattern fitting, Pattern Recognition 37 (2004) 2131–2139.
[7] N. Efrati, H. Licztin, H.B. Mitchell, Classified block truncation coding-vector quantization: an edge sensitive image compression algorithm, Signal Process.: Image Commun. 3 (1991) 275–283.
[8] I. Mor, Y. Swissa, H.B. Mitchell, A fast nearly optimum equispaced 3-level block truncation coding, Signal Process.: Image Commun. 6 (1994) 397–404.
[9] Y. Wu, D.C. Coll, Single bit map block truncation coding for color image, IEEE Trans. Commun. 35 (1987) 352–356.
[10] T. Kurtia, N. Otsu, A method of block truncation coding for color image compression, IEEE Trans. Commun. 41 (1993) 1270–1274.
[11] S.C. Tai, Y.C. Lin, J.F. Lin, Single bitmap block truncation coding of color image using a hopfield neural network, Inform. Sci. 103 (1997) 211–218.
[12] C.K. Yang, J.C. Lin, W.H. Tsai, Color image compression by moment-preserving and block truncation coding techniques, IEEE Trans. Commun. 45 (1997) 1513–1516.
[13] S.I. Olsen, Block truncation and planar image coding, Pattern Recognition Lett. 21 (2000) 1141–1148.
[14] A.K. Al-Asmari, A new video compression algorithm for different videoconferencing standards, Int. J. Network Manage. 13 (2003) 3–10.
[15] J.T. Tou, R.C. Gonzalez, Pattern Recognition Principles, Addison-Wesley, Reading, MA, 1974.
[16] K. Komatsu, K. Sezaki, Reversible transform coding of images, in: Proceedings of SPIE Visual Communications and Image Progressing, vol. 2727, 1996, pp. 1094–1103.
[17] T. Nakachi, T. Fujii, J. Suzuki, Lossless and near-lossless compression of still color images, in: Proceedings of ICIP99, vol. 1, 1999, pp. 453–457.
[18] W.B. Pennebaker, J.L. Mitcell, JPEG: Still Image Data Compression Standard, Van Nostrand Reinhold, New York, 1993.
[19] G.K. Wallace, The jpeg still picture compression standard, IEEE Trans. Consumer Electron. 38 (1992) xviii–xxxiv.
[20] A. Toet, M.P. Lucassen, A new universal colour image fidelity metric, Displays 24 (2003) 197–207.
[21] C.C. Sung, S.J. Ruan, B.Y. Lin, M.C. Shie, Quality and power efficient architecture for the discrete cosine transform, IEICE Trans. Fundam. E88-A (2005) 3500–3506.
[22] C. Loeffler, A. Lightenberg, G.S. Moschytz, Practical fast 1-d DCT algorithms with 11-multiplications, in: Proceedings of ICASSP, May, 1989, pp. 988–991.
[23] Y.C. Hu, C.C. Chang, Edge detection using block truncation coding, Int. J. Pattern Recognition Artif. Intell. 17 (2003) 951–966.

About the Author—BIBHAS CHANDRA DHARA was born in 1976. He received B.Sc. (Hons) in Mathematics from University of Calcutta in 1997. He received B.Tech in Computer Science & Engineering from University of Calcutta in 2000. He received M.Tech in Computer Science from Indian Statistical Institute in 2002. Currently, he is working as a Lecturer in Department of Information Technology, Jadavpur University, Kolkata, India. His research area and interest include Image Processing and Video Compression.

About the Author—BHABATOSH CHANDA was born in 1957. He received B.E. in Electronics and Telecommunication Engineering and Ph.D. in Electrical Engineering from University of Calcutta in 1979 and 1988, respectively. He received 'Young Scientist Medal' of Indian National Science Academy in 1989 and 'Computer Engineering Division Medal' of the Institution of Engineers (India) in 1998. He is also a recipient of UN fellowship and, UNESCO-INRIA fellowship. He is a Fellow of Institute of Electronics and Telecommunication Engineers and the National Academy of Science, India. He also received Shri Hari Om Prerit Vikram Sarabhai Research Awards in 2001. He has published more than 70 technical articles. His research interest includes Image Processing, Pattern Recognition, Computer Vision and Mathematical Morphology, Video Compression, and the application domain includes document page image, satellite image, biomedical image and indoor scenes. He has been working as a Professor at Indian Statistical Institute, Kolkata, India since 1996.