Image and Video Compression using Block Truncation Coding and Pattern Fitting for Fast Decoding

Bibhas Chandra Dhara
Department of Information Technology
Jadavpur University, Salt Lake Campus
Kolkata 700098
India

Thesis submitted to Indian Statistical Institute
for partial fulfillment of the requirements for
the degree of Doctor of Philosophy

To My Parents

Abstract

In the present era of multimedia, the requirement of image/video storage and transmission for video conferencing, image and video retrieval, video playback, etc. are increasing exponentially. As a result, the need for better compression technology is always in demand. The limited bandwidth of internet also asks for transmission of desired objects only. Progressive image transmission provides such facilities, where transmission is done in steps and the transmission of undesired image can be stopped at an early stage. Modern applications, in addition to high compression ratio, also demand for efficient encoding and decoding processes, so that computational constraint of many real-time applications is satisfied. In some applications, multimedia objects are compressed just once but are reconstructed frequently and most of the times the compressed objects are available on the local storage device. In such cases, the compression methods should be such that efficient reconstruction of the objects must be possible along with high quality. This thesis concentrates on image and video coding that leads to fast decoding. Such techniques are suitable for image search and retrieval and video playback.

In this thesis, first of all, a novel grayscale image compression method (BTC-PF) based on block truncation coding (BTC) and vector quantization (VQ) is proposed. This method inherits the advantages of both BTC and VQ. The BTC-PF method has some controlling parameters through which we can control the bit-rate and quality. The BTC-PF method is also successfully employed to handle low bit-rate coding.

The BTC-PF method is extended to color image compression. Here, the RGB model is transformed to $O_1O_2O_3$ model through a lossless transformation. The BTC-PF method is then used to compress the O_i planes independently.

The BTC-PF method is modified to achieve the progressive transmission of grayscale

and color images. In this respect, the patternbook which is used by BTC-PF is organized as full search progressive transmission tree (FSPTT). Finally, the residual image due to the progressive transmission is encoded by the context-based adaptive lossless image coding (CALIC) to achieve a lossless coding.

In video coding, the BTC-PF method is employed for intra-frame and residual frame coding. The residual frames are usually obtained by reducing inter-frame redundancy (or temporal redundancy) through motion estimation, which is one of the most time consuming modules in a video compression system. Here also, we have proposed a fast motion estimation scheme to obtain the residual frames on which a modified version of BTC-PF method is applied.

The performance of proposed methods including grayscale image compression, color image compression, progressive image transmission and video coding are quite good. As the decoding methods are mainly based on table look up, decoding module of these methods are faster than that of the state-of-the-art techniques. The effectiveness of the proposed schemes is established by comparing the performance with that of the existing methods.

Acknowledgment

I am deeply indebted to my advisor Prof. Bhabatosh Chanda, Indian Statistical Institute, for his continuing support, constructive criticism, feedback and guidance which made this effort to fruition.

I would like to thank Prof. S. Chattopadhyay, Mr. U. Ray, Mr. U. K. Roy, Mrs. P. Bhaumik and Mr. C. Maiti of Department of Information Technology, Jadavpur University for their generosity and inspiration.

Special thanks to Prof. D. P. Mukherjee and Dr. P. Pal of Indian Statistical Institute and Dr. S. K. Saha, Department of Computer Science, Jadavpur University, for their alround help and support.

All the members of Department of Information Technology, Jadavpur University, deserve special thanks as they have provided a lot of logistic support.

The author wishes to thank all the members of the ECSU, Indian Statistical Institute.

I must mention the continuous encouragement obtained from Susanta-da, Nilanjan, Bidyut, Gautam, Sarit, Suman, Partha-da, Pradip, Sitansu, Muni, Arindam and many others which helped me to overcome the hurdles and also their delightful company that kept me afresh.

I would like to express my gratitude to my parents and other family members. I also would like to express my deep gratitude to my maternal uncle Mr. Ganesh Koley who always encourages me in my study since childhood.

(Bibhas Chandra Dhara)

Contents

List of Figures

List of Tables

List of Abbreviations

BTC	Block Truncation Coding
VQ	Vector Quantization
BTC-PF	Block Truncation Coding using Pattern Fitting
CALIC	Context-based Adaptive Lossless Image Coding
FSPTT	Full Search Progressive Transmission Tree
ME	Motion Estimation
bpp	bit per pixel
CR	Compression Ratio
PSNR	Peak Signal to Noise Ratio
FI	Fidelity Index
HBTC	Hierarchical BTC
2BTC-PF	BTC-PF with 2 level patterns
2BTC-PFG	2BTC-PF with patternbook generated using Geometrical structure
2BTC-PFC	2BTC-PF with patternbook generated using Clustering method
CIBTC-VQ	Classified Interpolative Block Truncation Coding and Vector Quantization
Adaptive D/I	block truncation coding with Adaptive Decimation and Interpolation
LBTC-PF	Low bit-rate compression using BTC-PF
QS	Quincunx Sub-sampling
SBBTC	Single-bit-map BTC
CBTC	Color BTC

CICMPBTC	Color Image Compression by Moment-Preserving and BTC technique
CICMPBTC*	modified version of CICMPBTC
CBTC-PF	Color image compression using BTC-PF method
PIT	Progressive Image Transmission
PBTC	Progressive image transmission based on BTC
BPM	Bit Plane Method
CBMCPBTC	Common Bit Map Color image Progressive transmission using BTC
GBN	Guessing By Neighbor method
PBTC-PF	Progressive image transmission method using BTC-PF
Palgo	Progressive algorithm
LPBTC-PF	Lossless Progressive scheme using BTC-PF
QuadSegment	Quadtree Segmentation based progressive transmission
CPBTC-PF	Color image Progressive transmission scheme based on BTC-PF
BMA	Block Matching Algorithm
SW	Search Window
HBTC-PF	Hierarchical Block Truncation Coding using Pattern Fitting
BDM	Block Distortion Measure
SQP	SQuare Pattern
CP	Cross Pattern
OS	Octagon Search
SAD	Sum of Absolute Difference
DS	Diamond Search
ETSS	Efficient Three Step Search
HEXBS	Hexagon Search
EHEXBS	Enhance HEXxagon Search
ISC	Initial Search Center
MOS	Modified Octagon Search

Chapter 1

Introduction

1.1 Background

In today's information-driven society various multimedia items, e.g. graphics, image and video are generated, manipulated and transmitted in digital form. The volume of this digital information is huge and a large amount of storage space is required. The transmission of such information through limited bandwidth channel is time consuming also. The amount of data transmitted through the Internet grows heavily every year, and a large fraction of this data is image. The usefulness of digital images in communicating information is well appreciated; however, the cost of storing and transmitting images is much higher than the storage and transmission of text. Although a series of still images forms a video sequence, but these still images are treated in a different way in comparison to the video (image frames). In response to the demand, almost in every year, the capacity of the storage devices as well as the Internet bandwidth are increased. Still, it is not sufficient to meet the current requirement. Under such circumstances, image/video compression technique has come

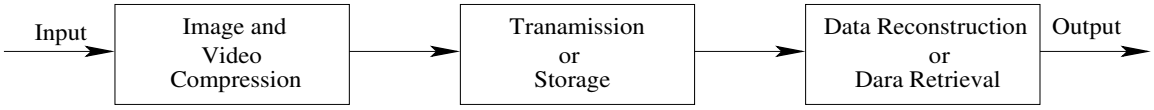


Figure 1.1: Image and video compression for transmission and storage.

up as a most effective solution.

An image compression method represents an image in a more compact way. The compression techniques increase the effective volume of storage space allowing more data to be stored in the storage device. In addition, the compression methods increase the effective bandwidth of the transmission medium and higher volume of data can be transmitted over the same transmission channel. Thus, image and video compression refers to a process in which the amount of data used to represent an image or a video is reduced to meet the bit-rate requirement (not more than the maximum available bit-rate). On the other hand, the quality of the reconstructed image or video must satisfy the requirement of certain application, and the computational complexity should also be affordable for the application. Fig. 1.1 shows the functionality of image and video compression for transmission and storage. The required quality of the reconstructed image and video is application dependant. For example, in medical science and some other scientific measurement, the reconstructed image and video has to be an exact replica of the original image and video. In these application fields only lossless compression methods are used. However, in applications like motion pictures, videophony etc. a certain amount of information loss is acceptable. This type of compression technique is called lossy compression.

1.2 Feasibility of Image and Video Compression

Image and video information have different types of redundancies, such as *inter-pixel redundancy, psychovisual redundancy, code redundancy,* and *spectral redundancy.* The amount of compression depends on the extent of exploitation of such redundancies.

- **Inter-pixel redundancy** signifies the correlation between pixels within an image frame or between the pixels of a group of successive frames (in a video sequence). These types of redundancies are known as spatial redundancy and temporal redundancy, respectively. **Spatial redundancy** implies that the intensity value of a pixel can be guessed from that of the neighboring ones. Exploiting the spatial redundancy, data size to represent an image can be reduced. There are several methods in literature like, predictive coding method (also known as differential coding method) and its variations from different angles [136], which are designed to exploit the spatial redundancy. **Temporal redundancy** is concerned with the statistical correlation between the pixels in successive frames in a video sequence. So, a frame may be predicted from its preceding and/or succeeding frames along the temporal direction. This technique is referred to as inter-frame predictive technique. Motion estimation (ME) based coding techniques [139] are widely used in video coding to exploit the temporal redundancy.

- **Code redundancy** is present in almost all images. Usually, in a digital image the number of bits are used for representing the intensity at each pixel is constant regardless the intensity value and the frequency of occurrence of the intensity. The most popular method for reducing code redundancy is by employing variable length code such as Huffman code [70] and arithmetic code [169]. The variable length coding method reduces the average bit-rate.

- **Spectral redundancy** is related to the correlations between the color planes (R, G, B) of an image. In a straightforward manner, a color image can be compressed by employing a grayscale compression technique independently to each of the color planes. But, higher compression may be achieved by mapping the correlated color planes into uncorrelated ones. De-correlation of the color planes can be done pixel-wise by Karhunan-Loeve transform (KLT) [124, 130]. However, KLT is image dependent, i.e., the transform matrix needs to be computed for each image and transmitted along with the coded image. The linear transformations of the RGB space to luminance and chrominance (LC) spaces are popular. Some widely used LC spaces are YCbCr, YIQ, YUV [42]. These LC representations consist of a luminance component and two chrominance components. LC spaces are popular because (i) the luminance component represents the intensity of the image and can be represented as a grayscale image, and (ii) the chrominance components represent the color information of the image. An important aspect of these color spaces is that the human eye has lower sensitivity to high frequency components of the chrominance range. Therefore, chrominance components are sub-sampled and compressed more aggressively. All the above LC transformations are irreversible due to the effect of finite precision arithmetic. Hence, these transformations are not suitable for lossless compression. In lossy system, this error (due to transformation) is invisible because of quantization.

- **Psychovisual redundancy** originates from the characteristics of the human vision system (HVS) [56]. In the HVS, all visual information are not perceived equally; some information may be more important than others. This implies that if we apply fewer data to represent less important visual information, perception is not affected. In this sense, some visual information is psychovisually redundant and elimination of this type redundancy leads to compression. However, the present work does not exploit psychovisual redundancy.

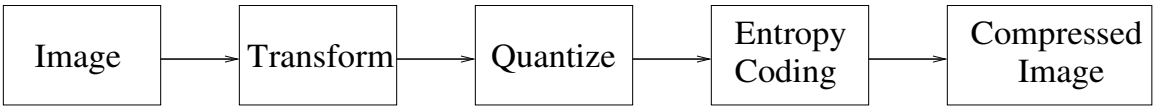


Figure 1.2: Transform based compression

1.3 Compression Techniques

Different types of redundancies in image and video information help to compress them. Video coding methods exploit the temporal redundancy (by motion estimation) and use the still image compression method to reduce the spatial redundancy. Compression methods may be classified in two groups: transform domain methods and spatial domain methods. Compression techniques may also be classified as lossy/lossless.

1.3.1 Transform domain methods

The performance of transform domain methods is, in general, superior to that of spatial domain methods in terms of compression ratio and quality of the reconstructed image. Currently used compression standards like, JPEG (DCT based) [120, 159] and JPEG2000 (wavelet based) [147, 27, 140, 43] are transform based methods. In transform domain methods, images are first transformed (using, say, DCT or wavelet transform) from spatial domain to frequency domain, then coefficients are quantized followed by the entropy coding of the quantized coefficients as shown in Fig. 1.2.

In traditional transform coding, Fourier transform or discrete cosine transform are used. The main problem of these transformations is the block size to handle both smooth and active area. The smooth area information is localized in a more compact way in frequency domain; whereas, in the active area it is more localized in spatial domain. So, it is very difficult to obtain a compromise between these using the traditional transformation. The wavelet representation of the image data provides a good

trade-off between spatial and frequency domain representation. All steps of wavelet-based compression are invertible, except the quantization step. Several algorithms have been developed to exploit this. Embedded algorithms, like embedded zerotree wavelet (EZW) [138] and its refined version, i.e., set partitioning in hierarchical trees (SPIHT) [135] are frequently used in coding technique. These processes take care of quantization and produce embedded bit-stream as the output. One advantage of embedded coding is that the encoder can terminate the encoding process at any time (in order to meet a target transmission rate). JPEG2000 uses wavelet transform as its core technique because it provides not only coding efficiency, but also good spatial and quality scalable functionality. Wavelet transform is also adopted in video coding [104, 53, 94].

1.3.2 Spatial domain methods

Spatial domain compression methods (specially the decoding part) are computationally much less intensive compared to the transform domain methods. Popular spatial domain image compression techniques are the vector quantization (VQ) [136], block truncation coding (BTC) [30] and 1-D/2-D run length coding [136, 8]. Other methods include block matching coding [142], quadtree coding [137], context-based coding [90, 108], predictive coding (e.g., Differential pulse code modulation (DPCM) [136]), bit-plane coding [56] and fractal based coding [77, 115, 151]. In bit-plane method, an image (gray or color) is considered as series of binary images and each binary image is compressed via binary compression method. All other methods mentioned above are based on the observation that the value of any randomly selected pixel in the image depends on the values of its neighbors. In block matching method, there are two buffers, namely, the search buffer and the look-ahead buffer. Each block in the look-ahead buffer is matched with all the blocks in the search buffer and the best matched block is selected. This method is an extension of LZ77 [182]. Context-based

method compresses an image by scanning it pixel by pixel, examining the context of every pixel, and then assigning a probability to it depending on how many times the same context was seen in the past. DPCM methods calculate the difference d_{ij}= x_{ij} - x_{ij-1} between two consecutive pixels and encode d_{ij}. The first pixel value x_{i0} is either encoded separately or written on the compressed stream in raw format. The quadtree method scans the image region-wise and looks for region consisting of more or less same value. This type of methods produces a tree, whose size depends on the content (smoothness/contrast variation) of the image. The method starts with a single node representing the entire image. If the (sub)-image represented by the current node is non-uniform, then it is divided into 4 quadrants, which are represented by the 4 children of the current node; otherwise, the current node is saved as leaf node with corresponding intensity value. Fractal based image compression (FIC) technique [77, 115, 151] makes use of iterated function system to achieve high compression ratio. The main characteristics of FIC approach relies on the assumption that image redundancy can be efficiently exploited through block-wise self-transformability.

1.3.3 Lossless/Lossy coding

Image and video compression are crucial steps in multimedia applications, and the quality has to be maintained up to a certain satisfactory level. Again, in some applications like, medical analysis, scientific measurement and legal issues, the reconstructed image has to be an exact replica of the original image. Thus, depending on the requirement, a compression method may be lossy/lossless. All the compression methods (either spatial domain based or transform domain based) is either lossy or lossless depending on the coding scheme. In lossless compression, there is a limit beyond which the compression may not be achieved. However, in lossy compression, there is a direct relation between the amount of compression and the distortion incurred. Normally, lossy compression provides higher compression than lossless compression. Examples

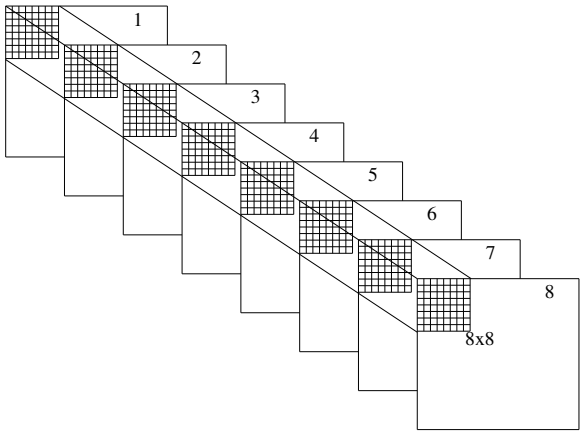


Figure 1.3: 3-D DCT of $8 \times 8 \times 8$.

of lossless coding methods are Run length coding, Contour coding, Huffman coding, Lempel-Ziv-Welch coding [121, 167], and arithmetic coding. Lossy coding methods include Vector Quantization (VQ), block truncation coding, predictive coding, transform based coding (JPEG and JPEG2000) and fractal based coding.

1.3.4 Video coding

Still image compression based video compression techniques can be classified into two categories. The first category is straightforward where each frame is compressed separately. One of the simplest video compression algorithms is called 'motion JPEG' [168], and it involves independent coding of the frames of a video sequence using JPEG method. But, in the second group, still image compression methods are used in generalized form. For example, generalized DCT applied to video coding by extending 2-D DCT to 3-D DCT. Instead of applying 8×8 DCT to a block of single frame, $8 \times 8 \times 8$ DCT (shown in Fig. 1.3) can be applied to a video sequence. That is, 8×8 co-located blocks of 8 successive frames are coded together with the 3-D DCT [139]. Similar concept is proposed in [62], where 3-D BTC ($4 \times 4 \times 3$) is used to compress video sequence. Currently, the video coding methods use a hybrid coding

method [59, 75, 76, 141, 74, 9], where a predictive coding method is used to reduce the redundancy (spatial redundancy by intra-frame prediction and temporal redundancy by inter-frame prediction), and finally the residual frame is coded by standard image compression method. Wavelet transform is also adopted in video coding [104, 53, 94].

1.4 Performance measurement

As discussed earlier, image and video compression algorithms are used mainly to represent image and video in a more compact way. This saves the storage space and reduces bandwidth requirement during transmission. At the same time, the quality of the reconstructed image and video is also an important factor. Suppose $f(x, y)$ is an original image of size $M \times N$ and each pixel value is represented using b bits. Thus, MNb bits are required to represent the image f. Now if K bits can preserve the desired information of that image and $K \ll MNb$, it is said that the compression is achieved. Suppose, the image is reconstructed from those K bits and $g(x, y)$ denotes the reconstructed image. If $g(x, y) = f(x, y)$ for all (x, y), then it is termed as a lossless compression, otherwise it is a lossy compression. Performance of a compression method is measured in terms of f, g, MNb and K. In some applications, the computational complexity of the coding and decoding methods is also an important issue.

1.4.1 Compression ratio

In order to evaluate the performance of a compression method, one of the two major criteria is bit-rate, which is expressed in terms of bit per pixel (bpp). It is defined as the average number of bits required by the compressed image for each pixel of the

image. Thus

$$bpp = \frac{\text{size of compressed image in bits}}{\text{no of pixels in the image}} = \frac{K}{MN} \tag{1.1}$$

Sometimes, the amount of compression is also defined as compression ratio (CR), where

$$CR = \frac{\text{size of original image}}{\text{size of compressed image}} = \frac{MNb}{K} \tag{1.2}$$

Another criterion is the quality of the reconstructed image. The amount of compression, the quality and the computation time are closely interrelated and one of them may be more important than others depending on the specific application. A compression algorithm can be thought of as means to offer the best trade off among these three. The measurement of visual quality of the reconstructed image or video is very important. There are two types of quality (visual) assessments: the subjective assessment, and the objective assessment. Each of these has its own merits and demerits.

1.4.2 Subjective Quality Measurement

The subjective visual quality measurement plays an important role in visual communication. In subjective quality assessment, a number of reconstructed images (frames of video) are given and the observers are requested to rank the images. The average of these ranks indicates the subjective quality of the scheme. The process is time consuming, laborious and expensive. Subjective quality measurement is non-repetitive, that is, for the same set of images the ranking given by an observer may change time-to-time.

1.4.3 Objective Quality Measurement

The goal of objective quality assessment research is to suggest quality metrics that can quantify the quality of image and video automatically. Objective quality assessment techniques in literature are mainly error-based methods [165]. These methods use pixel based difference metrics like, mean square error (MSE), root MSE, mean absolute error (MAE), signal to noise ratio, and peak signal to noise ratio (PSNR). Let $f(x,y)_{M\times N}$ and $g(x,y)_{M\times N}$ be the original and reconstructed images, respectively. The MSE between f and g is defined as

$$MSE = \frac{1}{MN}\sum_{x=1}^{M}\sum_{y=1}^{N}\{f(x,y) - g(x,y)\}^2 \tag{1.3}$$

For 8-bit grayscale image PSNR is defined as

$$PSNR = 10log_{10}\frac{255^2}{MSE}\ dB \tag{1.4}$$

These error-based methods are appealing because they are simple to calculate, have a physical meaning, and are mathematically convenient in the context of optimization. However, they may not correlate well with the perceived visual quality [44, 52]. The PSNR value for color image is defined by the same Eq. (1.4), but MSE may be defined as

$$MSE = \frac{MSE_R + MSE_G + MSE_B}{3} \tag{1.5}$$

where MSE_R, MSE_G and MSE_B are the MSE of red, green and blue planes, respectively.

Recently, a new notion [163, 164, 165] is used in measuring the image quality. It says: *The main function of the human eyes is to extract structural information from the viewing field, and the human visual system is highly adapted for this purpose. Therefore, a measurement of structural distortion should be a good approximation of perceived image distortion.* Based on this notion, a universal quality metric (also

known as fidelity index) [163] is defined. Image fidelity index (FI) [163] 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 reconstructed/processed image data. The fidelity index (FI) is then given by:

$$FI = \left(\frac{\sigma_{xy}}{\sigma_x \sigma_y}\right)\left(\frac{2\overline{x}\ \overline{y}}{\overline{x}^2 + \overline{y}^2}\right)\left(\frac{2\sigma_x \sigma_y}{\sigma_x^2 + \sigma_y^2}\right) \tag{1.6}$$

where,

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

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

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

The dynamic range of FI is [-1, 1]. The maximum value 1 occurs only when the reconstructed/processed image is identical with the original image and -1 when $y_i = 2\overline{x}\text{-}x_i$. The first component of Eq. (1.6) 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 image, the fidelity indexes FI_j are measured locally over small region of the image and finally are combined into a single value called FI. Here, a window of size 8×8 is considered for local measurement, which slides (row-by-row and column-by-column) over the entire image. Thus

$$FI = \frac{1}{L}\sum_{j=1}^{L} FI_j \tag{1.7}$$

where L is the number of possible positions of the window and FI_j represents the fidelity index at the j^{th} position. Most of the image distortion metrics in the literature

apply to the grayscale images. But, the universal metric, FI [163] has been extended for color images [152]. To obtain the fidelity index of a color image, the fidelity index of each color plane is calculated, and they are combined through a (weighted) mean [152]. Suppose a color image is defined in $l\alpha\beta$ system, then the color fidelity index FI_{color} may be defined as:

$$FI_{color} = \sqrt{w_l FI_l^2 + w_\alpha FI_\alpha^2 + w_\beta FI_\beta^2} \tag{1.8}$$

where w_l, w_α, and w_β represents the fidelity factors. In our experiment we set $w_l = w_\alpha = w_\beta = \frac{1}{9}$. The major drawback of the universal metric lies in its time complexity. In this respect PSNR has its strength, since it is fast and easy to implement. This makes PSNR the most popular measure.

1.5 Objective of the Thesis

Image and video compression refers to a process in which the amount of data used to represent an image or a video is reduced to meet the bit-rate requirement. The quality of the reconstructed image or video should satisfy the requirement for intended applications and the computational complexity should also be affordable for the application. A number of compression methods are available in the literature, but still this is an open research area. Standard compression methods, like JEPG and MPEG, are widely used. These are transform domain methods. Sometimes computational complexities of standard methods are beyond the affordable limit of certain applications. Dissatisfaction with the capabilities of standard algorithms for specific application has kept the search for new methods alive.

The compression methods offer a trade-off between the quality, bit-rate and computational cost. Whenever, we talk about an image compression method, its performance should be good enough for both grayscale and color images. In general, the decod-

ing or reconstruction of the image/video from the compressed data stream is done several times (may be at different resolution also), but the compression is done just only once. For example, in applications like, image search and retrieval, video-on-demand and video playback, the item has to be decoded frequently depending on the request; whereas it is encoded once by an off-line process. So the decoding method (reconstruction of the image/video) should be fast enough to meet the requirement of real-time display.

The transmission of digital information should be cost effective too. Consider the situation that after completion of the transmission it is found that the transmitted item is not the desired one, and then it is a total waste of the bandwidth and time. If the decision can be taken at very early stage whether to continue or abort the transmission, then both time and bandwidth can be saved. The progressive image transmission method, first, transmits the initial sketch and then step-by-step transmits more data and improve the quality of the received image. Using progressive transmission scheme, the transmission of undesired image can be aborted at the early stage. Multimedia transmission thus suggests that the progressive transmission, and consequently, progressive coding should be considered as a mandatory feature of any image coding method.

For certain application fields like, medical science, lossless compression is very much important. So, the capability of achieving the lossless compression is one of the important characteristics of an image coding technique. Thus, the compression algorithm must have certain tunable parameters through which the trade-off between the quality and application specific compression ratio can be achieved. Present requirement of image data manipulation also suggests that the computation of image features, like histogram computation, region extraction, edge detection, etc. should be done directly on the compressed version. Last but not the least, a successful image compression method may also find its application in intra-frame or I-frame coding as

well as residual frame coding in a video compression scheme.

Given this scenario, the objective of this thesis is to propose a compression technique having the following features.

- It should be good enough for both grayscale and color images.
- The methodology should have low computational cost and decoding must be simple and fast enough to support real-time display.
- The scheme should have features to support progressive transmission.
- The proposed scheme may have extendibility to achieve lossless coding method.
- It can be tuned to obtain desired trade-off between the quality and compression ratio.
- It should have applicability in video coding.

1.6 Contribution of the Thesis

The primary objective is to develop an image compression method that should support fast decoding, must facilitate the progressive transmission, and perform satisfactorily for both grayscale and color images as well as video. Whenever, we talk of fast decoding, the obvious choice is the spatial domain based compression method as the inverse transformation, required in frequency domain techniques, is omitted here. The inverse transform is the major time consuming module in the decoding part of any transform domain compression scheme. Computation of image feature in the compressed image is much easier if the compression method is a spatial domain method. Two widely used spatial domain based image compression techniques are

block truncation coding (BTC) and vector quantization (VQ). BTC method is well-known for good quality, but compression ratio is low; on the other hand, VQ offers high compression ratio, but the quality of reconstructed image is poor. These two methodologies have formed the basis of the proposed scheme.

- In this thesis, we have proposed a novel spatial domain based image coding method, called block truncation coding using pattern fitting (BTC-PF) [32, 33] which combines the benefits of both BTC and VQ coding methods and is applied to grayscale images using two-level patterns. The method is named as 2BTC-PF.

- The BTC-PF method is modified to achieve low bit-rate image coding and referred to as LBTC-PF [34].

- The BTC-PF method is successfully extended for the color image compression method called CBTC-PF [37].

- The coding methodology of BTC-PF method is extended for the progressive coding, considering a tree structure for the patternbook. The extended version of BTC-PF method, namely PBTC-PF and CPBTC-PF, is employed for progressive transmission of both grayscale and color images [38, 39], respectively.

- The BTC-PF method is also employed in video coding. The method is used to encode the I-frame as well as the residual frames [40]. In this connection, a fast motion estimation method [36] is also proposed to obtain the residual frame by reducing temporal redundancy.

1.7 Thesis Organization

In this thesis, first of all we have proposed image and video compression methods based on the principle of the BTC and VQ methods. The proposed method provides good quality as well as high compression ratio and at the same time its decoding cost is negligible compared to that of JPEG and JPEG2000. The organization of the thesis is as follows. In chapter 2, the past works on BTC and VQ are reviewed and the fundamental idea of the BTC-PF method is presented. The grayscale image compression using BTC-PF method is discussed in chapter 3 and its extension to low bit-rate image compression is also discussed. Chapter 4 is focused on the color image compression method. Progressive transmission scheme based on BTC-PF is discussed in chapter 5. This chapter describes both lossy and lossless compression of grayscale and color images. Chapter 6 is focused on video coding method. In this chapter a fast motion estimation method, I-frame coding method and the residual frame coding method are discussed. Concluding remarks and the future scope of the research is presented in chapter 7.

1.7.1 Data used in experimentation

In order to study the performance of the proposed methods (for grayscale image compression, color image compression, progressive image transmission, and video coding), we have carried out the experiment on a large number of images and video sequences. We have taken the test images and video sequences from different sources [2, 3, 1]. To present and compare the performance of grayscale image compression methods we use a set of 12 different images: 'Airplane', 'Boat', 'Couple', 'Lake', 'Lena', 'Man', 'Peppers', 'Zelda', 'Tiffany', 'Goldhill', 'Splash', and 'Baby'. These images are shown in Fig. 1.4. To present and compare the performance of the color image compression

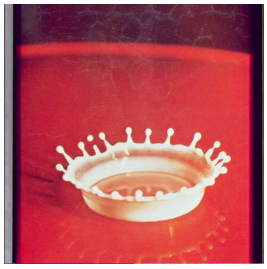
Figure 1.4: Collection of grayscale images used to evaluate the performances.

methods, we have used a set of 10 images: 'Lena', 'Peppers', 'Airplane', 'Couple', 'House', 'Zelda', 'Girl', 'Lab', 'Splash', and 'Tiffany' which are shown in Fig. 1.5. The performance of the proposed motion estimation algorithm and the video coding method using the proposed ME algorithm and BTC-PF method are evaluated on a number of video sequences: 'Carphone', 'Claire', 'Mother Daughter', 'Coastguard', 'Container', 'Hall Monitor', 'Susie', 'Tennis', 'Football', 'Foreman', 'Miss America', 'Mobile', 'News', 'Stefan' and 'Tempete'. Fig. 1.6 shows three successive frames of each of these sequences.

Airplane

Lena

Peppers



Splash

Tiffany

Couple

House

Zelda

Girl

Lab

Figure 1.5: Sample color images used in the experiment.

Figure 1.6: Three successive frames from various test video sequences.

Chapter 2

Block Truncation Coding using Pattern Fitting

2.1 Introduction

In this chapter, a spatial domain based compression method is proposed to fulfill the requirements mentioned in section 1.5. Block truncation coding (BTC) and vector quantization (VQ) are two widely used spatial domain based coding techniques. BTC method results in good quality, but suffers from high bit-rate. On the other hand, in VQ, quality is not as good as BTC, but the bit-rate is low. The encoder of VQ selects the best codeword from the codebook. This process is time consuming. However, the decoder is based only on table look-up method and, hence, is faster. In this chapter, an image compression method, namely block truncation coding using pattern fitting (BTC-PF), based on the amalgamation of BTC and VQ is proposed. In the next chapter we describe the application of this method to graylevel image compression. Then this method is extended to color image compression, progressive

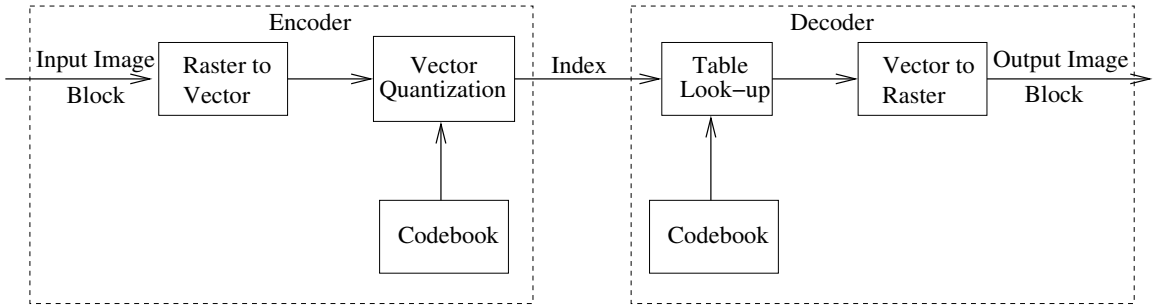


Figure 2.1: Basic block diagram of VQ.

image transmission, and video coding. These extended methods are presented in subsequent chapters. The proposed method (BTC-PF) is primarily a modification of block truncation coding (BTC) using the pattern fitting which is similar to VQ concept. Before elaborating the proposed method, a brief study on VQ and BTC is given in section 2.2 and section 2.3 respectively. The fundamental concepts of the proposed BTC-PF method are introduced in section 2.4.

2.2 Vector Quantization

A vector quantization, Q, is a many-to-one mapping from a set of K-dimensional vectors V=$\{v_1,\ldots,v_M\}$, into a set W ($\subset V$):

$$Q : V \longrightarrow W \tag{2.1}$$

where W=$\{w_1,\ldots,w_N\}$ is the set of K-dimensional reconstruction vectors called a codebook, and $w_i \in W$ is called a codevector or codeword. Each codeword w_i is identified by the index i. The quantizer assigns an input vector $v_m \in V$ to a codeword $w_i \in W$ (represented by the index i) such that distortion $d(v_m, w_i) \leq d(v_m, w_j)$, $\forall w_j \in W$. To reconstruct, the index i is used to look up the codebook and corresponding w_i is used to represent and/or generate v_m.

Image data is first partitioned into a set of blocks. Each of such blocks is arranged

in a vector according to certain order, say, lexicographic order. The quantization (or coding) step involves searching for each input vector v_i, the closest codeword w_j, in the codebook. Finally, the index j of the selected codeword is coded and transmitted to the decoder. At the decoder, the index is decoded and the corresponding vector is picked up from the same codebook by table look-up operation. The basic block diagram of VQ is shown in Fig. 2.1. The performance and the complexity of VQ depend on the codebook used to encode the vectors and on the quantization process, which include distortion measure and search technique.

2.2.1 Codebook Generation

The performance of VQ depends on the codebook (not only on codewords of the codebook, but also on the size of the codebook). Development of a good codebook is the key step of VQ. The codebook should satisfy the following criteria:

1. The input vector space is partitioned into a given number of regions. The number of regions is decided according to the allowed bit-rate.

2. Each region is represented by a codeword. The codeword is the statistical mean (with respect to the distortion metric) of the vectors within the region. For example, if distortion metric is MSE, then statistical mean is nothing but the mean (average) vector.

To generate a codebook, a large number of vectors from various images are used to form a training set and the training set is used to generate a codebook using, say, clustering algorithm (LBG [99], K-means [153]). The codewords of the codebook generated through clustering technique depend on the initial seeds.

2.2.2 Quantization

The quantization process in VQ involves the selection of best-matched codeword, w_j, for the input vector v_i. In this selection, distortion metric ($d(v_i, w_j)$) could be L_1- or L_2-norm based or some other distortion metric. If L_2-norm is used as the distortion metric, then the distortion $d(v_i, w_j)$ between input vector v_i =$(v_{i1}, v_{i2}, \ldots, v_{iK})$, and codeword $w_j \in W$ is defined as

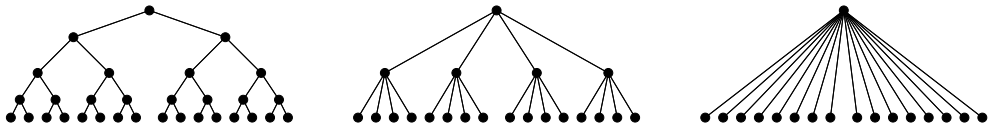
$$d(v_i, w_j) = \sum_{t=1}^{K} (v_{it} - w_{jt})^2 \tag{2.2}$$

We ignore the square-root operation as it will not have any bearing on the result. Similarly, L_1-norm based distortion metric is given as

$$d(v_i, w_j) = \sum_{t=1}^{K} |v_{it} - w_{jt}| \tag{2.3}$$

To search the codeword in the codebook, the most straightforward technique is the full search. The codeword w_m for which the distortion [given in Eq. (2.2) or Eq. (2.3)] is minimum is selected.

The full search gives optimum solution with respect to the given codebook and the metric, but it is computationally expensive. VQ is simple in implementation, but its encoder is computationally very expensive. Quality of the reconstructed image also depends on the codebook. There is exponential relation between the bit-rate and the size of the codebook (also the encoding time complexity). There are several fast search methods for VQ in literature [112, 68, 105, 117, 20, 16, 12]. Such methods fall into two categories. Methods in the first category decrease the search time at the cost of the quality. They search a subset of the codebook, and the optimum result is not ensured. Tree-structured VQ (TSVQ) [16, 12] is one such algorithm. The other category, called fast full search, reduces the search time and at the same time gives full-search equivalent results [68, 105, 117, 20]. Mainly two approaches are adopted to speed up the full search process. (i) A 'best-so-far' threshold is developed. During



(a) (b) (c)

Figure 2.2: (a) Binary tree search quantization, (b) quadtree search quantization, (c) full search quantization.

the process of evaluation of $d(v_i, w_j)$ (given in Eq. (2.2) or Eq. (2.3)) if partial sum reaches the threshold then search is stopped and this codeword will no longer be a candidate. (ii) The distortion metric is simplified and accordingly some data are pre-computed and stored in a table (as an offline process) [118, 105]. VQ method may also be categorized according to the domain on which it is applied. Some groups are spatial VQ, transform VQ, and subband VQ etc. A detail review may be found in [114].

2.2.3 Structured VQ

So far, we consider only the unconstrained VQ (no restriction is put on the codebook organization). But this is limited to codebooks of small size and vectors of low dimensions. Otherwise, search complexity will be prohibitive. In structural constrained VQ methods, the codebook is organized in a structured manner. This means that the codewords are distributed in a restricted manner in the space. The VQ methods, like tree-structured VQ (TSVQ) [16, 12], mean-residual VQ (M/RVQ) [4], multi-stage VQ (MSVQ) [81, 63], classified VQ (CVQ) [129], predictive VQ (PVQ) [61], finite state VQ (FSVQ) [7, 10], Lattice VQ (LVQ) [79, 84], entropy constrained VQ (ECVQ), hierarchical VQ (HVQ) [113], etc. do not employ (conceptually) the full search as they intend for reducing the time complexity in the quantization step. For example, TSVQ organizes the codebook like a tree. Starting with the root, at each level input

vector is compared with all the children of the current codeword and only one of them is chosen. The process is continued until a leaf of the tree is reached. Fig. 2.2 shows different structures of the codebooks. Full-search VQ is a special case of TSVQ, where level of the tree is two and all the codewords are children of the root node of the tree (Fig. 2.2(c)). A codebook may have some vectors with same intensity pattern, but they differ in terms of their mean value. All the vectors having same pattern can be represented by a common vector along with the corresponding mean. In this particular case, removing the mean of each vector the size of the codebook can be reduced. Then for each input vector, the resulting residual vector, obtained by subtracting sample mean from the input vector, is vector quantized and the mean of the input vector is scalar quantized. This concept forms a basis of the mean-shape VQ (M/SVQ) [4, 11] and the mean-residual VQ (M/RVQ) [4]. Multi stage VQ (MSVQ) [81, 63] cascades of a number of VQ steps. The first level VQ is performed on the input vector. Then second-level vector quantization is on the residual vector, and so on. Low bit-rate VQ coded image suffers from the edge distortion, as edges cannot be reproduced perfectly by a small sized codebook. In another approach the vectors are classified and for each class a separate codebook is used. Each input vector is coded by the appropriate quantizer. The computational complexity is greatly reduced since only a sub codebook is checked for an input vector. This concept is known as classified VQ (CVQ) [129]. PVQ [61] is a straightforward extension of the scalar predictive quantization (DPCM) to vector data. Since the consecutive image blocks, i.e., the vectors are statistically dependent, a prediction of the incoming vector is made based on the previously coded vector and, finally, the residual vector is vector quantized. Finite state VQ (FSVQ) [7, 10], is represented by a finite state machine. Like CVQ, FSVQ also has multiple codebooks; only difference with the CVQ is that no side information is sent to the decoder to inform which of the codebooks to use. This information is carried out by the next-state rule. The basic VQ is the fixed rate VQ, i.e., the indices of codewords are represented by fixed-length codes.

The variable-length coding of indices results a lower average bit-rate. In this coding, the codewords selected infrequently are assigned the higher-length indices, while the codewords that are used frequently are assigned shorter-length indices. This approach is called the entropy constrained VQ (ECVQ). A special type of structured VQ technique is Lattice VQ (LVQ) [79, 84]. Here, the codewords are some regular points of the Euclidean space and these points (or codewords) can be defined by a matrix and which is known as the generating matrix. Theoretically, the size of a codebook can be infinite; however, there is no need to store these codewords (the codewords can be generated by the generating matrix) and by exploring the observations that a regular pattern in the codewords exist, the search time is reduced. So far, we consider the fixed block size vector quantization; however, the variable block size quantization is also possible and that reduces the bit-rate significantly. Hierarchical VQ (HVQ) [113] uses this concept, where the image blocks are partitioned in a quadtree manner and the leaf blocks are vector quantized. Here, the sizes of the leaf nodes may vary and for each possible size of the leaf nodes there is a codebook.

2.3 Block Truncation Coding

Block truncation coding (BTC) is a simple, fast, lossy and fixed length compression technique for grayscale images. This is a block-adaptive binary encoding scheme based on moment preserving quantization. The concept is introduced by Delp and Mitchell [30]. In BTC method, an image is divided into $n \times n$ blocks (in general n=4) and each block is coded separately. Graylevels of each block is quantized by Q level quantizer and these quantizer levels are chosen such that a few low order moments are preserved in the quantized output. In the simplest form of BTC, the first two moments are preserved and blocks are represented by two quantization levels. By incorporating additional constraints, higher order moments can be preserved. Suppose k ($=n^2$) be

the number of pixels in a block and also suppose $f(\mathbf{x}_i)$, $\mathbf{x}_i \in C$ are the gray values of the pixels in a block of the original image where C represents the set of coordinates of pixels in the block, i.e., $C = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\}$. The first two sample moments m_1 and m_2 are given by

$$m_1 = \frac{1}{k}\sum_{i=1}^{k} f(\mathbf{x}_i) \tag{2.4}$$

$$m_2 = \frac{1}{k}\sum_{i=1}^{k} f^2(\mathbf{x}_i) \tag{2.5}$$

m_1 is the sample mean and the sample variance σ^2 of image block is given by

$$\sigma^2 = m_2 - m_1^2 \tag{2.6}$$

Then a two level quantization is performed on the block. The pixels with intensity greater than the quantization threshold are quantized to value b, and the other pixels are quantized to value a. Here, the sample mean m_1 is set as the quantization threshold and suppose the quantization partition of C with respect to the threshold m_1 gives two sets of pixels C_0 and C_1, such that $C = C_0 \cup C_1$ and $C_0 \cap C_1 = \emptyset$ where $C_0 = \{\mathbf{x'}_1, \mathbf{x'}_2, \ldots, \mathbf{x'}_{k'}\}$ and $C_1 = \{\mathbf{x''}_1, \mathbf{x''}_2, \ldots, \mathbf{x''}_{k-k'}\}$. The quantization partition of C can be represented by a binary-pattern P. The binary pattern or bit-plane P is defined as

$$P(\mathbf{x}_i) = \begin{cases} 0 & \text{if } f(\mathbf{x}_i) \leq m_1 \\ 1 & \text{if } f(\mathbf{x}_i) > m_1 \end{cases} \tag{2.7}$$

The partition of C is defined as $C_0 = \{\mathbf{x}_i | P(\mathbf{x}_i) = 0\}$ and $C_1 = \{\mathbf{x}_i | P(\mathbf{x}_i) = 1\}$. The quantization levels a and b are used to preserve first two sample moments. Then

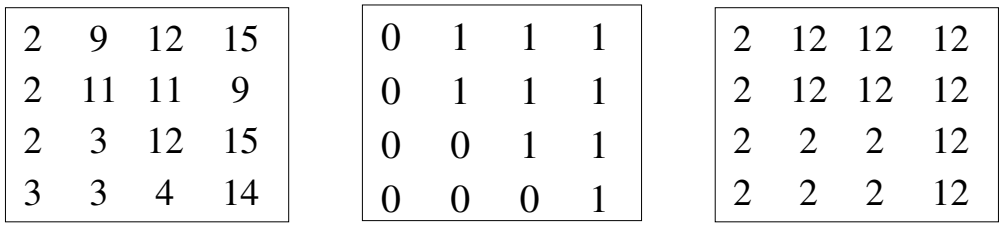
$$km_1 = k'a + (k - k')b \tag{2.8}$$

$$km_2 = k'a^2 + (k - k')b^2 \tag{2.9}$$

and solving for a and b yields

$$a = m_1 - \sigma\sqrt{\frac{k - k'}{k'}} \tag{2.10}$$

$$b = m_1 + \sigma\sqrt{\frac{k'}{k - k'}} \tag{2.11}$$



(a) (b) (c)

Figure 2.3: An example of BTC: (a) original block, $m_1 = 7.94$, $\sigma = 4.91$, (b) corresponding bit-pattern, k' =7, (c) reconstructed block with a=2 and b=12.

Thus a compressed image block is represented by the triplet (a, b, P). For 8-bit grayscale image, the straightforward BTC needs $\frac{8+8+16}{16} = 2$ bits per pixel. An example of BTC method is shown in Fig. 2.3. The steps for encoding a block by BTC are:

- Perform quantization (determination of threshold, and quantization levels)
- Coding of the quantization levels (i.e., a and b)
- Coding of the bit-pattern P

BTC method is simple and results into high quality reconstructed image, but the compression ratio is low. In literature, there are several modifications of BTC to improve the performance. A brief survey on BTC is given in [48]. A detail comparison of the various basic BTC algorithms has been reported in [106]. The original BTC [30] uses threshold $T_h = m_1$ and preserves the first and second order moments. This method can be extended to preserve third order moments also. To preserve third order moment, the T_h is determined such a way that number of pixels greater than T_h is given by

$$q = (k - k') = \frac{k}{2}(1 + G\sqrt{\frac{1}{G^2 + 4}}) \tag{2.12}$$

where

$$m_3 = \frac{1}{k}\sum_{i=1}^{k} f^3(\mathbf{x}_i) \tag{2.13}$$

and

$$G = \frac{3m_1m_2 - m_3 - 2m_1^3}{\sigma^3}, \qquad \sigma \neq 0 \tag{2.14}$$

A generalized method to preserve r-, $2r$- and $3r$-th moments is presented in [60]. The absolute moment BTC (AMBTC) [93] preserves sample mean (m_1) and first order absolute central moment α, where

$$\alpha = \frac{1}{k}\sum_{i=1}^{k} |f(x_i) - m_1| \tag{2.15}$$

and corresponding quantization levels are

$$a = m_1 - \frac{\alpha k}{2k'} \tag{2.16}$$

$$b = m_1 + \frac{\alpha k}{2(k-k')} \tag{2.17}$$

If the quantization levels are set as m_{low} (lower mean) and m_{high} (higher mean), then MSE is reduced. The means, m_{low} and m_{high} are defined as follows.

$$m_{low} = \frac{1}{k'}\sum_{f(x_i)\leq m_1} f(x_i) \tag{2.18}$$

$$m_{high} = \frac{1}{k-k'}\sum_{f(x_i)> m_1} f(x_i) \tag{2.19}$$

AMBTC is faster than BTC, as no square root operation is required. There are different methods using various quantization techniques available in the literature. Neural network based quantization method is proposed in [128], where there is no direct quantization threshold; instead, Hopfield Neural Network (HNN) outputs the quantization level a or b for each pixel. Another important concept is the optimal quantization. The MSE distortion metric can be minimized (i) by selecting the threshold T_h as $\frac{a+b}{2}$ for given quantization levels, or (ii) by choosing the quantization

levels as m_{low} and m_{high} (defined by Eqs. (2.18) and (2.19)) for a given threshold T_h. These two concepts are used to define an iterative method [45] where iteration starts with threshold $T_h = m_1$ or quantization levels $a = x_{min}$ and $b = x_{max}$. Similarly, the median of the subset of pixels with respect to T_h can be used to minimize the mean absolute error (MAE) criterion.

In BTC, a block is represented either by (m_1, σ, P) or (a, b, P). The simplest representation of (a, b) or (m_1, σ) requires 8 + 8 bits for 8-bit grayscale images as these values $\in$[0, 255]. Fewer amounts of bits can be used to encode the parameters by scalar quantization of those separately. For example, say, $b \in [0, 127]$ and $a \in [0, 63]$, and then to encode (a, b) 6 + 7 =13 bits are required. Joint quantization method [62] reduces the bits requirement of (m_1, σ) to 10 bits. In [157], a modified AMBTC method is proposed, where a block is represented by (m_1, m_{low}, P). VQ is applied to the pair (m_1, m_{low}), and 8-bits are used to encode this pair. In predicted VQ method [47], the predictive errors (e_a, e_b) are encoded by VQ. Discrete cosines transform (DCT) can be used to compress the pair (a, b) [172]. In DCT based coding method, two sub-sample images one for a-values and other for b-values (or $(b - a)$-values) are formed and they are encoded. A linear predictor and scalar quantizer are included in differential pulse code modulation (DPCM) for coding the quantization levels a and b [69]. All the above mentioned coding methods for (a, b) or (m_1, σ) are mainly lossy methods. However, entropy-coding methods (Huffman, Arithmetic coding, etc.) can be used for the purpose of lossless coding.

In the bit-stream of BTC method, bit-plane shares a significant amount of space. Such requirements can be reduced. Normally, lossless coding for bit-plane is acceptable. However, considering a lossy coding method for bit-plane a large compression is possible at the cost of the quality. In the literature, there are several methods to reduce the size of the bit-plane [88, 162]. One simple method [107, 111] is to omit the bit-plane for smooth/nearly smooth blocks and these blocks are represented by

the corresponding block average (m_1). The smoothness of a block can be determined by thresholding on the variance [107], or on the intensity range of the block [111]. Another type of methods [107, 178], where bit requirement is reduced by partial coding (some bit-positions are ignored) of the bit-plane. At the time of reconstruction, the missing bits are determined by some pre-defined logical expressions [107]. Major drawback of the partial bit-plane coding is that some missing pixel positions may estimated wrong bit value and hence, the reconstructed image may contain pixels where an a-value has been changed to b-value and vice-versa. Hence, it fails for large contrast image block. In [107], an adaptive bit-plane coding method is proposed, where no bit-plane (for smooth/nearly smooth block), full bit-plane coding (for large contrast block), and partial bit-plane coding (for relatively low contrast block) are employed depending on the blocks. To reduce the error in partial bit-plane coding technique, the interpolation technique may be used to reconstruct the image [178, 88, 162]. Here, at the decoding phase, the partial block is reconstructed using available bits and then missing pixel values are interpolated on the basis of existing values.

Several VQ based BTC (BTC-VQ) approaches [156, 172, 69, 18, 176] have been proposed for reducing the volume of the bit-plane. In these methods, a codebook of binary vectors is available. For each image block, instead of transmitting the bit-plane, index of the code vector closest to bit-plane is transmitted. Generally, Hamming distance is used to measure the closeness of the vectors. One of the advantages to use Hamming distance is that it can be implemented by look-up table. An adaptive BTC-VQ is proposed in [166], where codebooks of different sizes are used to encode the bit-plane. The selection of the codebook depends on the block contrast which is defined as the difference between two quantization levels b and a (i.e., $b-a$). The main drawback in BTC-VQ methods is that the reconstructed image often suffers from staircase effect especially in the high contrast areas. A modified version of BTC-VQ, namely the classified VQ [45] uses two codebooks to avoid the staircase effect. The first codebook contains a set of 2-level code vectors and the

second codebook contains a set of 3-level code vectors. These codebooks are used for low contrast and high contrast blocks, respectively. In [116], a set of codebooks of different sizes are used. The selection of the codebook is made according to the desired compression ratio such that reconstruction is optimum.

The BTC method and all its variations discussed here are fixed size BTC (fBTC), i.e., the block size is fixed. The main drawback of fBTC is that it cannot adapt to the local statistics of the image. It also introduces artifacts within the regions that contain the edges. Data representation of fBTC in smooth region is inefficient [82, 133, 111]. One obvious solution is to encode the similar block by same information. In [23], a clustering technique is used to merge the similar blocks into a cluster and then cluster center is encoded by BTC based method which uses a small set of predefined binary edge patterns. The method presented in [64] tries to find out a suitable neighboring block, which is already encoded, to represent the current block. If a block is found within a pre-defined range, then the current block is represented by that neighbor block; otherwise current block is represented by the block mean or encoded by BTC. To overcome the drawbacks of the fBTC methods, another variation is to use the variable block sizes. These strategies are grouped as variable BTC (vBTC) [82] and hierarchical BTC (HBTC) [133, 111]. The vBTC or HBTC starts with a large block. Then, the activity of each block is tested and if it is found to be high active block, it is decomposed into a number of sub-blocks of equal size. The process is recursively applied to each sub-block until either current sub-block has low activity or it is a minimum size block. This hierarchical structure of the decomposition can be efficiently implemented by using quadtree, horizontal-vertical binary tree, and binary space partitioning tree [137]. Among these data structures quadtree is the most popular one. All the sub-blocks, which are not decomposed, can be encoded by any method discussed earlier. Some examples are hierarchical BTC-VQ compression system [46] and quadtree-segmented coding using VQ and BTC [65].

The quality of the images, which are encoded by BTC method, is usually good. However, some raggedness may appear if there is a sharp contrast in a block. To overcome this problem multi-level (3 or 4-level) BTC is used. A 3-level quantization method along with the original BTC is proposed in [54]. Methods proposed in [31, 45] use 3-level quantizer to encode the edge block. The non-edge blocks are encoded by 2-level quantizer. A 4-level quantizer is proposed in [26] without considering extra parameters, i.e., all four parameters are derived from a and b. The idea of the multi-level quantization is extended in [158]. Here, 1-, 2-, 4- and 8-level quantizers are used. Different optimum/nearly optimum methods are introduced in [173, 109, 87, 15]. In [173, 87] the quantization levels are determined to reduce mean absolute error. A fast equi-spaced 3-level quantization algorithm is proposed in [109] to achieve nearly optimum mean square error. Optimization of 2-, 3- and 4-levels quantization are considered in [15]. To obtain the optimum quantization, dynamic range tuning (DRT) is used. Various optimized BTCs are combined and an adaptive multi-level coding scheme is proposed.

All the methods for encoding the bit-plane are lossy. Entropy based coding of bit-plane [47] is lossless. In lossless coding of bit-plane, Markov model can be used to predict the bit-value and then arithmetic coding can be used. However, achievement due to such coding of bit-plane is very less. For example, for 'Lena' image only 10% bit-rate can be saved. A method proposed in [174], which saves 26% bit-rate for 'Lena' image using probability distribution of each bit-plane.

One important feature of BTC method is that the decoders are efficient. Second, some of the features of the images can be computed directly from the compressed data. These characteristics of BTC justify the use of BTC though its performance (bit-rate, quality) is poor compared to that of transform domain compression method.

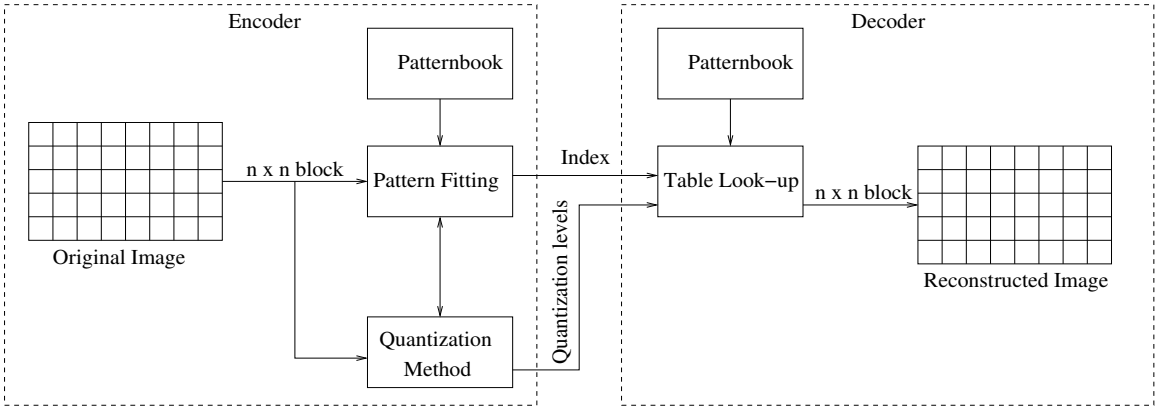


Figure 2.4: A schematic diagram of BTC-PF method.

2.4 BTC-PF method

The encoding method of VQ is time consuming, whereas its decoding method uses table look-up method and is very fast. This method results in higher compression ratio, though quality of the reconstructed image is usually not as good as BTC. BTC is a simple and fast method, which enables high quality reconstruction but bit-rate is also high. Comparatively, the encoder of BTC is faster than that of VQ, while its decoder is little slower. A compromise between these two methods gives a fast decoder, maintains good quality for reconstructed image with moderate bit-rate. Again, this hybrid method can also be used in image feature extraction. That means the compressed data due to this method can directly be used to compute image features like, edge [66, 28], histogram [127], and so on. In BTC [30], a block is represented by (a, b, P), where a and b are quantization levels for reconstruction and P represents the intensity pattern within the block. The pattern P shares a significant amount of space in the bit-stream of the BTC method. In VQ based BTC method [156, 172, 69, 18, 176] there is a binary codebook and instead of transmitting actual bit-pattern, the index of the closest bit-pattern (from the codebook) is transmitted. It may wrongly assign a and b values in some positions.

In the proposed BTC-PF method, for each image block of size $n \times n$, the pattern P is selected from a set of, say, M predefined patterns called patternbook. Each pattern represents Q intensity levels of the block and size is same as that of the block (i.e., $n \times n$). For an image block B, the best fit pattern P is found from the patternbook. Then Q quantization levels are determined from the B using the intensity pattern defined by the selected pattern P. At the time of reconstruction of a block, the index of the selected pattern and Q different graylevels are required. As the method selects a pattern from a set of predefined patterns to represent the block, the quality of the reconstructed image is, in general, little lower than that of conventional BTC. However, this little sacrifice in PSNR earns a huge gain in bit-rate. The block diagram of the BTC-PF method is shown in Fig. 2.4. Like VQ methods, the performance of BTC-PF also depends on the patternbook. Hence, the design of patternbook is a crucial step. The encoder of BTC-PF thus consists of: (i) pattern fitting (selection of the best pattern) and (ii) the quantization method, i.e., determining the Q graylevels. Here also, the decoder uses the table look-up method.

2.4.1 Selection of Pattern

The method of selection of the best fit pattern for an image block B of size $n \times n$ is as follows. For an image block B, let the pixels coordinates are $x_1, x_2, \ldots, x_{n^2}$ and the corresponding pixel intensities are $f(x_i)$. Available patterns are, say, $P_1, P_2, \ldots, P_M$ of size $n \times n$ and the levels present in a pattern are represented by t where $1 \leq t \leq Q$. Thus, any pattern is represented as $P_j = p_{j1} \cup p_{j2} \cup \ldots \cup p_{jQ}$, $j = 1, 2, \ldots, M$ such that $p_{js} \cap p_{jt} = \emptyset$ 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, we try to fit the image block B to these patterns P_j, $j = 1, 2, \ldots, M$. The one with least square-error is chosen as the best fit. In the selection process, the total square-error between image block B and

the pattern P_j is

$$e_j = \sum_{t=1}^{Q} e_{jt} \tag{2.20}$$

where

$$e_{jt} = \sum_{x_i \in p_{jt}} (f(x_i) - \mu_{jt})^2 \tag{2.21}$$

$$\mu_{jt} = \frac{1}{|p_{jt}|} \sum_{x_i \in p_{jt}} f(x_i) \tag{2.22}$$

where $|p_{jt}|$ is the cardinality of the set p_{jt}, i.e., the number of pixels having level t in the j^{th} pattern P_j. Finally, the best fit pattern is the one for which square-error is minimum, and its index I is obtained as

$$I \ = \ arg\{\min_{j \in 1,2,\ldots,M} \{e_j\}\} \tag{2.23}$$

It may be noted that the index I conveys the partition of the image block whereas in multilevel BTC [31, 45, 26] to define the partition of the pixels of a block we have to calculate the threshold values and correspondingly the pattern itself.

2.4.2 Quantization levels

The pattern selection process of BTC-PF method returns the best matched pattern for the current block. Hence, the partition of the block becomes known. The quantization levels can be determined so that a number of moments remain preserved in the reconstructed block. For example, in case of 2 level BTC-PF to preserve first order and second order moments, Eqs. (2.10) and (2.11) can be used. In other way, mean values μ_{It}, $t \in 1, 2, \ldots, Q$, defined in Eq. (2.22), can be used as the quantization levels, where I is the index of the best match pattern.

2.4.3 Design of Patternbook

The Design of the patternbook for BTC-PF system is one of the most important tasks. In fact, the quality of reconstructed image is dependant on the patternbook. One option is to generate the patternbook manually by considering spatial homogeneity as well as the geometrical orientation of the edges and lines. These patterns are designed heuristically based on extensive observation and experimentation guided by intuition. This technique is a laborious, and not free from the biasness of the designer. The other alternative is to develop a well-devised method to avoid human interference. In such methods, to generate a patternbook, a number of images are considered. These images are divided into $n \times n$ blocks and subsequently corresponding to each block a n^2-dimensional vector is generated. Then a clustering algorithm (e.g., LBG [99], K-means [153]) is applied on all these vectors to produce M classes. The intensity of each block, prior to clustering, is transformed so 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 the same cluster. Intensity $f(x_i)$ (i=1,2,...,n^2) of each of these M prototypes corresponding to M cluster is then thresholded to obtain a Q-level pattern. This requires the (Q-1) threshold values $\{\tau_t : 1 \leq t \leq Q-1\}$ that provide minimum error. To obtain those, $\{\tau_t\}$s are exhaustively varied over the intensity range. The error function to be minimized is as follows.

$$E = \sum_{t=0}^{Q-1} \sum_{\tau_t < f(x_i) \leq \tau_{t+1}} (f(x_i) - \mu_{t+1})^2 \tag{2.24}$$

where,

$$\mu_{t+1} = \frac{1}{|\tau_t < f(x_i) \leq \tau_{t+1}|} \sum_{\tau_t < f(x_i) \leq \tau_{t+1}} f(x_i) \tag{2.25}$$

where τ_0 = -1 and τ_Q = maximum intensity, for example $\tau_Q = 2^8 - 1$ for 8-bit image. It may be noted that although the threshold selection process is computation intensive, it is a part of off-line processing.

2.4.4 Performance Control

The BTC-PF method encodes an image block by a Q-level pattern and Q graylevels. It renders a good compromise between the quality and the bit-rate. The quality of reconstructed image depends on the patternbook, and we can achieve higher quality by considering larger patternbook or by increasing the number of levels of the patterns. This enhancement in quality is achieved at the cost of bit-rate. Low bit-rate coding, on the other hand, can be achieved by considering the smaller patternbook and by reducing the levels of the patterns. Again, we can control the performance of the BTC-PF by varying the size of the image block. Larger the image block, lower is the quality and higher is the compression ratio. The selection of pattern in the encoding method of BTC-PF is the major time consuming module and this is proportional to the size of the block, size of the patternbook and the number of levels in the patterns. Hence, in real-time applications, the computational cost can be controlled by proper selection of patternbook and levels of the patterns. However, it is important to note that decoding time is almost invariant with respect to these factors. Another important observation is that, if the difference between the successive quantized levels is less than some threshold, then the block may be assumed to be a smooth one and only a graylevel may be used to code that block. In that case, the compression ratio is increased drastically. So, the value of threshold can also control the performance and as expected. Higher is the threshold, lower is the quality and higher is the compression ratio and vice versa. With appropriate selection of the patternbook (size of the patternbook, levels of the patterns), size of the block and the threshold, the BTC-PF method may offer a good trade-off among the quality, bit-rate and this complexity. In this thesis, we have shown this in subsequent chapters, where we have used different patternbooks with different image block sizes and threshold values.

2.5 Conclusion

In this chapter, we present the fundamental concept of the BTC-PF method combining the concept of the BTC and VQ methods. The BTC-PF method adopts the merits of both the BTC and VQ methods. In BTC-VQ method [156, 172, 69, 18, 176] there is a chance that, during reconstruction, the value 'a' is assigned to a pixel which should have the value 'b' and vice versa. Our method eliminates this possibility. In subsequent chapters we will apply the proposed method to image and video compression.

Chapter 3

Grayscale Image Compression using BTC-PF

3.1 Introduction

This chapter focuses on the grayscale image compression using BTC-PF technique introduced in the last chapter. This BTC-PF method is a hybrid of the BTC method and VQ method as both of them lead to fast decoding. The BTC method results in high quality, but low compression ratio. On the other hand, VQ based compression methods result in poor quality with high compression ratio. In BTC-PF method, we are able to combine the advantages of both BTC and VQ methods except one, i.e., the encoding time. BTC-PF method is employed in image compression with different requirements. First of all, we devise an image compression algorithm based on BTC-PF method and employ it on the grayscale images. Subsequently, the technique is extended for low bit-rate compression. To evaluate the performance, 2-level patterns are used. Hence, two quantization levels are required to reconstruct the blocks. In

this experiment, the contrast and bias are used to represent the quantization levels and these two values help greatly to reduce the bit-rate. The proposed BTC-PF method out performs other spatial domain based methods. In section 3.2, the details of the grayscale compression method and its performance are discussed. The method to achieve low bit-rate is presented in section 3.3. Finally, the conclusion is drawn in section 3.4.

3.2 Image compression based on BTC-PF

BTC-PF method represents a block by I (the index of the selected pattern) and Q quantization levels. In this compression scheme, 2-level BTC-PF (2BTC-PF) method is used. In BTC [30] method, two level quantization is used and the quantization levels are determined in such a way that first two moments are preserved. In 2BTC-PF method, to maintain the uniformity we have also tried to preserve the first two moments in determining the quantization levels. Suppose, the pattern selection method of BTC-PF returns I as the index for a block; where $P_I = C_0 \cup C_1$ and $C_0 \cap C_1 = \emptyset$ where $C_0 = \{\mathbf{x'}_1, \mathbf{x'}_2, \ldots, \mathbf{x'}_{k'}\}$ and $C_1 = \{\mathbf{x''}_1, \mathbf{x''}_2, \ldots, \mathbf{x''}_{k-k'}\}$. Then $\{f(\mathbf{x}_i)|\mathbf{x}_i \in C_0\}$ defines the candidates of a quantization bin and $\{f(\mathbf{x}_i)|\mathbf{x}_i \in C_1\}$ that of the other bin. Without loosing generality, let us assume that the pixels of the set C_0 are marked as 0 and those in C_1 are marked as 1. Thus, the partition can be represented as a bit-pattern. Two graylevels $A - d$ (for the pixels marked as 0) and $A + d$ (for the pixels marked as 1) are used to represent the blocks. In order to maintain the moment preserving conditions, it is required to satisfy the following equations:

$$km_1 = k'(A-d) + (k-k')(A+d) \tag{3.1}$$

$$km_2 = k'(A-d)^2 + (k-k')(A+d)^2 \tag{3.2}$$

Solving for A and d we get

$$A = m_1 + \frac{\sigma(2k' - k)}{2\sqrt{k'(k - k')}} \tag{3.3}$$

$$d = \frac{\sigma k}{2\sqrt{k'(k - k')}} \tag{3.4}$$

Hence, intensity $\hat{f}(\mathbf{x}_i)$ of the pixels of the corresponding block in the reconstructed image is given by

$$\hat{f}(\mathbf{x}_i) = \begin{cases} A + d & \text{if } \mathbf{x}_i \in C_1 \\ A - d & \text{if } \mathbf{x}_i \in C_0 \end{cases} \tag{3.5}$$

It is clear that $a = A - d$ and $b = A + d$, where a and b are the quantization levels for partition denoted by I and these are determined by Eqs. (2.10) and (2.11).

3.2.1 Encoding of a Block

In BTC or AMBTC method, the quantization levels are stored either as (a, b) or $(a, b - a)$. In our 2BTC-PF method, the quantization levels are $A - d$ and $A + d$. To reconstruct a block, A and d are to be stored/transmitted. It gives advantage over usual scheme of storing a and b in the context of coding. The relationship between (A, d) and (b, a) is as follows:

$$A = \frac{b + a}{2} \tag{3.6}$$

$$d = \frac{b - a}{2} \tag{3.7}$$

The values A and d represent the bias (low frequency component) and contrast (high frequency component) in a block respectively. In view of the wavelet theory [102, 29], A may be considered as the response of the scaling function or the low resolution representation of the block and d is the response of wavelets. The Eqs. (2.10), (2.11) and (3.7) reveal that d is always non-negative and the standard deviation of d is

smaller than that of b and a. It leads to a higher compression for d in case of entropy coding. Another important observation is that d-value closer to 0 means the graylevels within the block can reliably be represented only by A. This leads to approximate d as d' which can be defined as

$$d' = \begin{cases} 0 & \text{if } d \leq d_{th} \\ d - d_{th} & \text{otherwise} \end{cases} \tag{3.8}$$

where, d_{th} is threshold of low value. Thus, to reconstruct a block d', A and the index (I) of the selected pattern are to be stored/transmitted. Again, if d' is zero, then no pattern index is required, and this situation can be handled by examining the d' value. This leads to further reduction in bit-rate.

The values of A do not usually cover the dynamic range of a grayscale image. The range of values also varies from image to image. So we transform the values of A to cover the range from 0 to $2^l - 1$, where l is the largest integer not exceeding $log_2(A_{max} - A_{min})$. Hence, the transformed value is defined as

$$A' = \frac{(2^l - 1)(A - A_{min})}{(A_{max} - A_{min})} \tag{3.9}$$

It may be noted that this needs transmission of A_{min} and A_{max} once at the beginning. As mentioned earlier that A is bias or average intensity of the block and due to spatial homogeneity it has stronger correlation with that of its neighboring blocks compared to a or b as used in conventional BTC. So, for further reduction of bpp, A' is coded through predictive coding. Fig. 3.1 shows the blocks that are used in prediction of A'. Suppose A' values of the blocks numbered as 1, 2, 3, 4, and C are A'_1, A'_2, A'_3, A'_4 and A'_c, respectively and further assume that these values are known. Now, assuming a linear model the prediction error is obtained as:

$$\triangle A' = A'_c - \sum_{i=1}^{4} w_i A'_i \tag{3.10}$$

where, w_1, w_2, w_3, w_4 are weights and are estimated considering a large number of

Used for Prediction 2	Used for Prediction 3	Used for Prediction 4
Used for Prediction 1	Current Block C	

Figure 3.1: Blocks used in Prediction of A' in current block.

images such that minimum $\triangle A'$ is achieved on an average. It is evident that the variance of $\triangle A'$ is much less than that of A'. Hence, through entropy coding $\triangle A'$ can be represented by less number of bits than that is needed to represent A' directly. It may be noted that the prediction coding does not introduce any further error in the reconstructed image. Thus, the encoding process is summarized as follows:

Step 1 Partition the image into a number of 4×4 non-overlapping blocks (B_4).

Step 2 For each block, B_4, do the following

Step 2.1 Select the best match pattern [see section 2.4.1], denoted by the index I

Step 2.2 Compute A and d using Eqs. (3.3) and (3.4).

Step 2.3 If $d < d_{th}$ then d'=0; else $d' = d$ - d_{th}

Step 2.4 Compute A' following the Eq. (3.9).

Step 2.5 Compute $\triangle A'$, using linear model given in Eq. (3.10).

Step 3 Encode d', $\triangle A'$ and I and transmit these information in said order (i.e., first d' followed by $\triangle A'$ and finally I (if any)).

		ng_1	ng_2		
	$*$	p_1	p_2	$*$	
ng_8	p_8	$\tilde{A}$	$\tilde{A}$	p_3	ng_3
ng_7	p_7	$\tilde{A}$	$\tilde{A}$	p_4	ng_4
	$*$	p_6	p_5	$*$	
		ng_6	ng_5		

Figure 3.2: Decoded pixel intensity of Block 4×4, if $\tilde{d} = 0$

3.2.2 Decoding of a block

The encoded bit-stream contains d', $\triangle A'$ and I (if any) block wise. The decoder first obtain these values and then compute $\tilde{d}$ the approximation of d' as

$$\tilde{d} = \begin{cases} 0 & \text{if } d' = 0 \\ d' + d_{th} & \text{if } d' > 0 \end{cases} \tag{3.11}$$

Similarly $\tilde{A}$, approximation of A' can also be computed.

$$\tilde{A} = \frac{A'_c(A_{max} - A_{min})}{2^l - 1} + A_{min} \tag{3.12}$$

where

$$A'_c = \triangle A' + \sum_{i=1}^{4} w_i A'_i$$

If a block is smooth ($\tilde{d} = 0$), the intensity of all pixels is straightway set to $\tilde{A}$. Let us call this conventional concept as CONM (conventional method). Alternatively, instead of setting all pixels to $\tilde{A}$, pixels of only central 2×2 block (see Fig. 3.2) may be set to $\tilde{A}$ and in the second pass intensity of pixel p_i, for $i = 1, 2, \ldots, 8$ is determined by linear interpolation using horizontal and vertical set of data depending on context. In the next pass the intensity of pixels marked with '$*$' are computed by

bi-linear interpolation. Let us call this interpolative method as INTM. It is expected that INTM should give higher PSNR as well as smoother visual appearance than the method CONM. This is experimentally verified on six synthetic images as shown in Fig 3.3, where intensities are generated as sine wave of various frequency and orientations. Table 3.1 shows the comparative performance of these two methods. Now, if $\tilde{d} \neq 0$ and then the block is reconstructed with the quantization levels $\tilde{A} - \tilde{d}$ and $\tilde{A} + \tilde{d}$, where the quantization partition (or bit-pattern) of the block is represented by the index I. Thus, the decoding method is described as follows.

Step 1 From encoded bit-stream obtain d', $\triangle A'$, and I (depending on d').

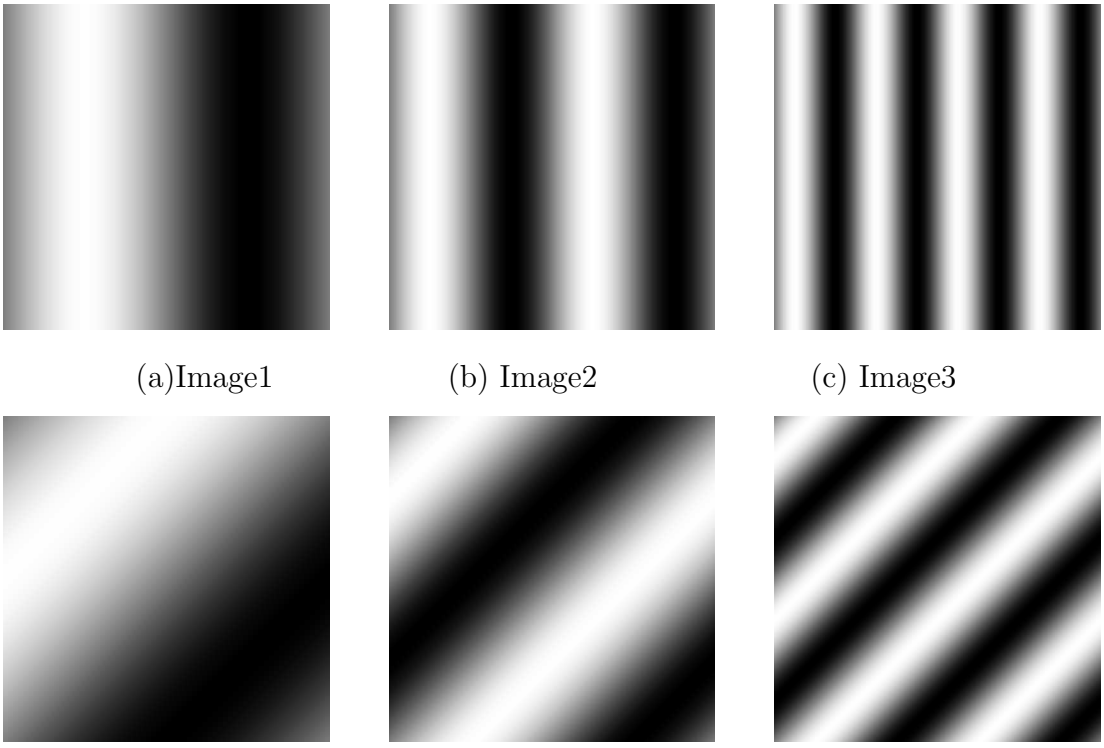
Step 2 Compute $\tilde{d}$, $\tilde{A}$ using Eqs. (3.11) and (3.12).

Step 3 If $\tilde{d} \neq 0$, reconstruct the block by taking pixel values as $\tilde{A} - \tilde{d}$ and $\tilde{A} + \tilde{d}$ based on 0/1 value of the pattern represented by the index I.

Step 4 Else use INTM method to reconstruct the block.

3.2.3 Performance of 2BTC-PF

To evaluate the performance of our 2BTC-PF method, a number of 8 bit images (given in Fig. 1.4) are used as the test image. Here, the nature of d', A' and $\triangle A'$ are studied on several images, and according to the average distribution, the weights w_1, w_2, w_3, w_4 are determined (used in Eq. (3.10)) and two Huffman codebooks for d' and $\triangle A'$ are developed. This is done to make a robust system for a wide variety of images. To encode an index value 6 bits are used without entropy coding. Eq. (3.8) shows that higher the value of d_{th} more number of blocks would be treated as smooth and hence both the quality and bit-rate become low. Thus, a trade-off between the quality and bit-rate is required. Experimentally we choose the value



(a)Image1 (b) Image2 (c) Image3

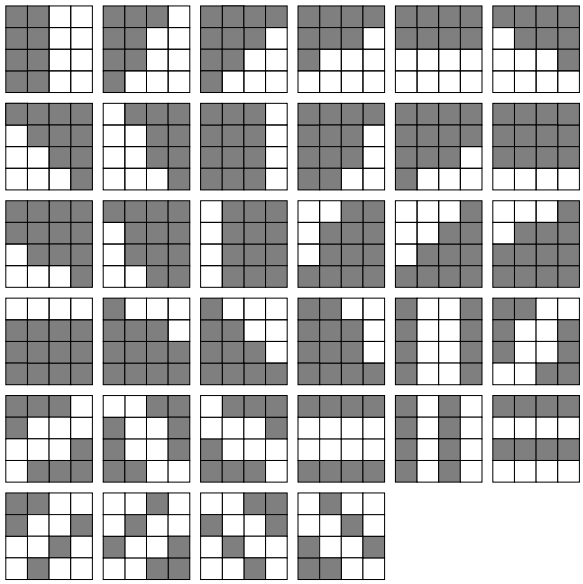
(d)Image4 (e) Image5 (f) Image6

Figure 3.3: Synthetic images used to compare INTM and CONM.

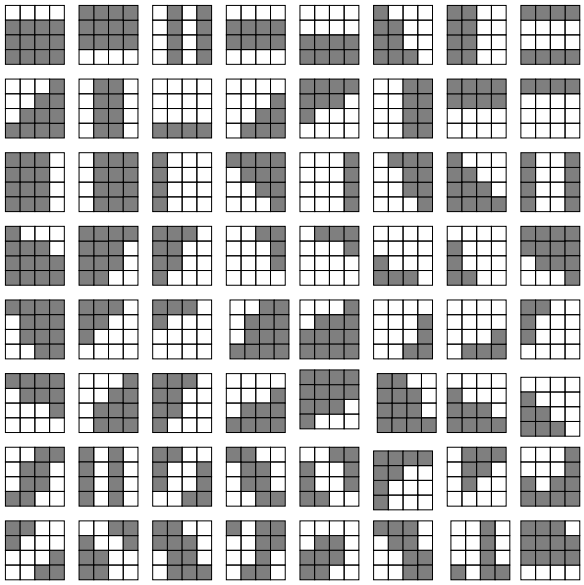
Table 3.1: Comparison of results of INTM and CONM methods on the set of synthetic images shown in Fig. 3.3.

Image	PSNR	
	INTM	CONM
Image1	47.29	44.17
Image2	45.53	39.64
Image3	40.47	38.57
Image4	48.26	45.93
Image5	46.95	42.08
Image6	41.60	38.96

of d_{th} as 4. In BTC-PF method, there are three main issues: (i) selection of best pattern, (ii) determination of quantization levels, and (iii) design methodology of the patternbook. These are discussed in section 2.4.1 - 2.4.3. To select the patterns, in 2BTC-PF method we have used the scheme described in section 2.4.1. We determine the quantization levels A and d, so that certain moments of the blocks remain preserved. The quantization method is described in section 3.2. Section 2.4.3 highlights the design process of a patternbook which are a) the clustering algorithm based b) manual generation. In this experiment two separate patternbooks are used, one is designed manually (shown in Fig. 3.4(a)) and the other one is generated by clustering algorithm (shown in Fig. 3.4(b)). The size of each patternbook is 64. The first patternbook contains altogether 64 patterns which include all the patterns of Fig. 3.4(a) and the complement of first thirty patterns of the same. To design the second patternbook, Fig. 3.4(b), first of all the input image blocks are processed so that their mean is reduced to zero and variance becomes unity. Then these modified vectors are fed as input to the K-means algorithm. Finally, the clustered centers are thresholded at zero to obtain the binary-patterns. Depending on the patternbook used, the 2BTC-PF method is called as 2BTC-PFG (when the patternbook is generated manually by considering **geometrical orientation of edges**) and 2BTC-PFC (using patternbook generated by **clustering method**). Here, the performance is evaluated in terms of bpp (Eq. (1.1)), PSNR (Eq. (1.4)) and fidelity index (FI) (Eq. (1.7)). The proposed encoding method (see section 3.2.1) is refined step-by-step to achieve more compression, obviously at the cost of quality. The step-by-step effect of 2BTC-PFG on 'Lena' image is shown in Fig. 3.5. In all the steps, bpp is calculated considering 6 bits for index (if any). In the conventional BTC method the quality of the reconstructed image is given by PSNR=32.89 dB and the output image is shown in Fig. 3.5(a). The refinement using the patternbook (given in Fig. 3.4(a)), is shown in Fig. 3.5(b) - 3.5(e). We obtain PSNR=31.79 dB and bpp=1.375 with A and d. Here, we consider 8 bits for A and d. Let us denote this step as PF-A-d (Fig. 3.5(b)).



(a)



(b)

Figure 3.4: Set of two-level patterns (also viewed as bit-patterns) used to fit with graylevels of image block: (a) considering the geometrical orientation of edges, (b) applying clustering algorithm (K-means).

(a) (b) (c)

(d) (e)

Figure 3.5: The step-by-step effect of 2BTC-PFG on 'Lena': (a) BTC (PSNR=32.89 dB, bpp=2.00), (b) PF-A-d (PSNR= 31.79 dB, bpp= 1.375) (c) PF-A-d' (PSNR= 31.61 dB, bpp= 0.88) (d) CONM (PSNR= 31.57 dB, bpp= 0.64) (e) INTM (PSNR= 31.59 dB, bpp= 0.64).

With A and d' we obtain PSNR=31.61 dB and bpp= 0.88. Here, entropy coding is used for d' and A is of size 8 bits (see Fig. 3.5(c)). Let us denote this step by PF-A-d'. The approximation A' and d' produces PSNR=31.57 dB, bpp=0.64. Here, predictive entropy coding is deployed for A', for d' entropy coding is used and CONM method is used for $d' = 0$ (see Fig. 3.5(d)). The INTM method improve the quality at the same bit-rate, PSNR=31.59 dB, bpp=0.64, shown in Fig. 3.5(e). The effect of the patternbooks (i.e., 2BTC-PFG and 2BTC-PFC) on 'Lena' image are given in Table 3.2. The visual appearance of the reconstructed images in case of 2BTC-PFG

Table 3.2: The effect of the patternbooks (shown in Fig. 3.4) on the image 'Lena'.

Steps	2BTC-PFG		2BTC-PFC	
	PSNR	bpp	PSNR	bpp
BTC	32.89	2.00	32.89	2.00
PF-A-d	31.79	1.375	31.82	1.375
PF-A-d'	31.61	0.88	31.63	0.89
CONM	31.57	0.64	31.60	0.64
INTM	31.59	0.64	31.62	0.64

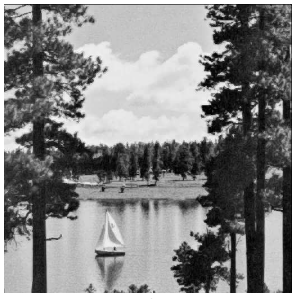
method is shown in Fig. 3.6.

The result of the 2BTC-PF method is compared with JPEG and two other modified version of BTC methods, namely CIBTC-VQ [88] and Adaptive D/I [162]. In CIBTC-VQ method, a block is classified as either a low-detail block or a middle-detail block or a high-detail block. To determine the class of a block, thresholds Th_1 and Th_2 are applied on the block variance. If it is low detail block (applying Th_1), then the block is represented by the block mean. A middle-detail or high-detail block, determined by threshold Th_2, is encoded by 2-level BTC. The reconstruction levels are quantized by VQ, and separate codebooks are used for middle-detail blocks and high-detail blocks. Finally, sub-sampling method is applied on the bit-plane. The every-other-row and every-other-column sub-sampling is used for middle-detail block and quincunx sub-sampling method is applied for high-detail block. The Adaptive D/I method is based on the gradient decimation method. In this method, for each block the horizontal and vertical gradients are computed. Accordingly, blocks are classified into one of the four groups: Smooth block, horizontal edge block, vertical edge block, and complex block based on two threshold values Thr_1 and Thr_2. For each class different decimation method is applied and, finally, the sub-sampled block

Airplane
(30.52, 0.60, 0.6557)

Boat
(30.03, 0.70, 0.6715)

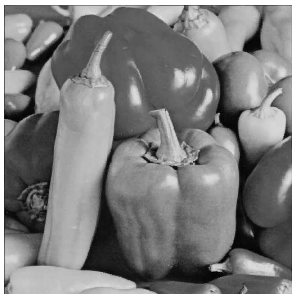
Couple
(31.65, 0.75, 0.7693)



Lake
(26.16, 0.94, 0.7150)

Lena
(31.59, 0.64, 0.6965)

Man
(29.96, 0.80, 0.7666)

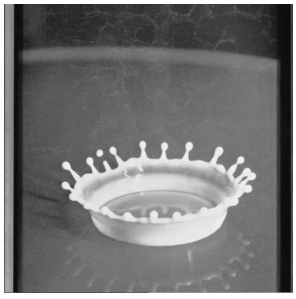


Peppers
(31.69, 0.68, 0.6803)

Zelda
(34.89, 0.60, 0.7654)

Tiffany
(33.80, 0.57, 0.6589)

Goldhill
(30.80, 0.84, 0.7840)



Splash
(35.35, 0.45, 0.5784)

Baby
(31.96, 0.80, 0.7981)

Figure 3.6: The Reconstructed images using 2BTC-PFG, values inside the bracket represent the PSNR, bpp and FI respectively. For example, 'Airplane' with PSNR=30.52 dB, bpp=0.60 and FI=0.6557.

is coded by AMBTC or by the block mean. To reduce the bit-rate further the block means and the difference between the mean and the lower reconstruction level, are encoded with predictive entropy coding. The comparative results are reported in Table 3.3. In the experimentation of CIBTC-VQ [88], the threshold $Th1$ is set to 10 and $Th2$ to 50 as suggested in the article and two bits are used for identifying the class of a block. In Adaptive D/I method Thr_1 and Thr_2 have been set to 6 and 5 respectively to achieve the PSNR close to what was reported in the original paper [162] and bit-rate has been measured for both the cases with and without predictive entropy coding. The result in Table 3.3 shows that the performance of the proposed method (2BTC-PF) is same for both the patternbooks. The average result of the 2BTC-PF method is quite impressive compared to other methods. The 2BTC-PF method is better than CIBTC-VQ method in all respect. The Adaptive D/I method gives same quality (PSNR and FI) as 2BTC-PF, but bit-rate is much higher. The table 3.3 also reveals thatthe proposed 2BTC-PF method is inferior to JPEG interms of PSNR, FI and bpp (we have used the JPEG code of the Independent JPEG Group [58]). However, the decoding cost of 2BTC-PF method is negligible compared to that of JPEG. A detail analysis of the cost of the decoding methods is given in chapter 4. Hence, our method is more suitable for intended applications, where fast decoding is essential. Comparing the performance of all these methods we conclude that our proposed method offers a good trade-off between quality and bit-rate, and requires negligible decoding time.

Table 3.3: Results of 2BTC-PF method and its comparison with other BTC based methods.

Images	2BTC-PF						CIBTC-VQ			Adaptive D/I				JPEG[1]		
	2BTC-PFG			2BTC-PFC												
	PSNR	bpp	FI	PSNR	bpp	FI	PSNR	bpp	FI	PSNR	bpp		FI	PSNR	bpp	FI
											without predictive	with predictive				
Airplane	30.52	0.60	0.6557	30.56	0.62	0.6577	30.15	0.76	0.6822	31.58	1.30	1.15	0.7060	39.96	0.60	0.7112
Boat	30.03	0.70	0.6715	30.06	0.70	0.6725	29.48	0.80	0.6632	29.86	1.41	1.21	0.7034	38.64	0.64	0.7288
Couple	31.65	0.75	0.7693	31.64	0.75	0.7685	30.61	0.82	0.7363	32.42	1.52	1.19	0.7967	38.84	0.65	0.8000
Lake	26.16	0.94	0.7150	26.18	0.94	0.7128	25.70	0.89	0.6602	27.87	1.94	1.70	0.8124	34.92	0.88	0.7193
Lena	31.59	0.64	0.6965	31.62	0.64	0.6958	30.92	0.78	0.6928	31.78	1.42	1.13	0.7347	39.87	0.55	0.7266
Man	29.96	0.80	0.7666	29.96	0.80	0.7670	29.33	0.84	0.7500	31.20	1.59	1.32	0.8086	37.51	0.74	0.8034
Peppers	31.69	0.68	0.6803	31.66	0.68	0.6787	30.76	0.79	0.6448	32.69	1.60	1.25	0.7308	39.96	0.56	0.6817
Zelda	34.89	0.60	0.7654	34.96	0.60	0.7662	33.57	0.76	0.7597	34.04	1.39	1.04	0.7944	41.86	0.46	0.7942
Tiffany	33.80	0.57	0.6589	33.78	0.58	0.6579	31.58	0.75	0.6303	32.70	1.30	1.03	0.7024	39.78	0.48	0.6765
Goldhill	30.80	0.84	0.7840	30.80	0.84	0.7837	29.85	0.86	0.7454	30.03	1.71	1.37	0.8199	37.66	0.72	0.8052
Splash	35.35	0.45	0.5784	35.33	0.46	0.5782	34.25	0.71	0.6005	35.70	1.14	0.88	0.6319	42.44	0.41	0.6200
Baby	31.96	0.80	0.7981	31.99	0.81	0.7989	30.84	0.60	0.7795	32.47	1.56	1.28	0.8220	39.39	0.71	0.8267
Average	31.54	0.70	0.7116	31.54	0.71	0.7115	30.59	0.78	0.6954	31.86	1.49	1.22	0.7526	39.22	0.62	0.7411

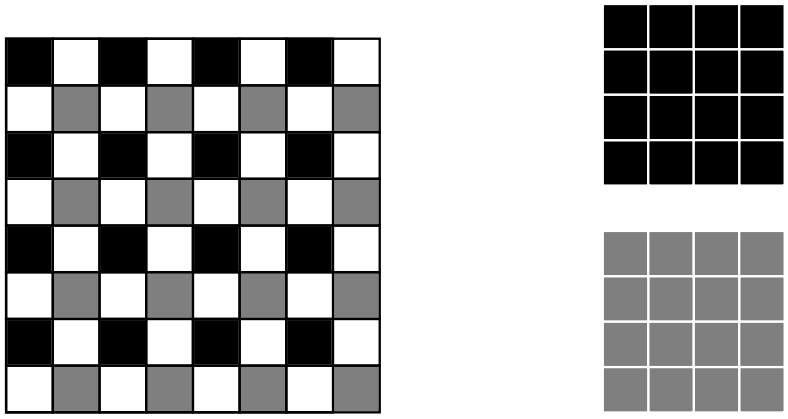
[1]JPEG is transform domain method whereas others are spatial domain methods. Though JPEG is superior its decoding time is many times higher than that of the proposed method.

3.3 Low bit-rate compression using BTC-PF

In real-time applications like video conferencing, higher compression is needed and quality is not so important as long as it is within an acceptable limit. In such case, the 2BTC-PF method is not suitable. Here, a low bit-rate method (LBTC-PF) is proposed. It is a modified version of BTC-PF and is more suitable than 2BTC-PF method for the category of application mentioned above. In LBTC-PF method, the images are first partitioned into 8×8 (B_8) non-overlapping blocks. From each B_8 block, two 4×4 (B_4) sub-blocks B_{41} and B_{42} are generated by quincunx sub-sampling (QS) [177, 178] is as shown in Fig. 3.7. Each B_4 is encoded by BTC-PF method and finally, B_8 is reconstructed by interpolation method. In low bit-rate coding method, the prime objective is to achieve a higher degree of compression maintaining an acceptable quality. To increase the compression ratio, sub-sampling method is used. Here, QS is chosen as the sampling method. In place of QS, one may use other sub-sampling technique also. Finally, the lower-rate coding scheme must maintain an acceptable quality along with the higher degree of compression. We have also tried with every-other-row and every-other-column (EREC) [177, 178] sub-sampling method. It is observed that EREC gives higher compression with respect to QS, but quality wise it falls below. Hence, we have chosen QS. The detail of the coding scheme is described hereafter.

3.3.1 Encoding the block

In LBTC-PF method, each B_4 block is encoded by 2-level BTC-PF method. In this coding technique, the quantization levels are A (Eq. (3.3)) and d (Eq. (3.4)). Hence, for each B_8 the triplets (A_1, d_1, I_1) and (A_2, d_2, I_2) are required for the reconstruction. Here d is also approximated to d' (Eq. (3.8)) for better compression. Also the



(a) (b)

Figure 3.7: 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).

correlation between A_1-values of neighboring B_8s and that between A_1 and A_2 within the same B_8 are exploited. Like A-value of the previous section [see section 3.2.1], the A_1 value of each B_8 is predicted from neighboring A_1's. The same linear model (given in Eq. (3.10)) is used to obtain the predicted error $\triangle A_1$. As, there is also a strong correlation between A_1 and A_2 of each block B_8, the difference coding, i.e., $A_1 - A_2$, is used in place of A_2. Thus, the encoding bit-stream of LBTC-PF method contains $(\triangle A_1, A_1 - A_2, d_1', d_2', I_1, I_2)$. The first four components are coded by entropy coding. To improve the correlations between A_1's and that between A_1 and A_2, a low pass filter (neighborhood averaging) is employed prior to quincunx sub-sampling. The smoothing of the block further reduces the deviation in d_i values. For coding, a 2-level patternbook of size 256 is used (shown in Fig. 3.8). To encode, the indices of the blocks, index graphs are used. The method of index coding using index graph is as follows.

In this method the image is partitioned into 8×8 (B_8) blocks. Let us denote ij^{th} B_8 by B_8^{ij} and its sub-blocks are denoted by B_{41}^{ij} and B_{42}^{ij}. Let X_k^{ij}, for k= 1,2, be

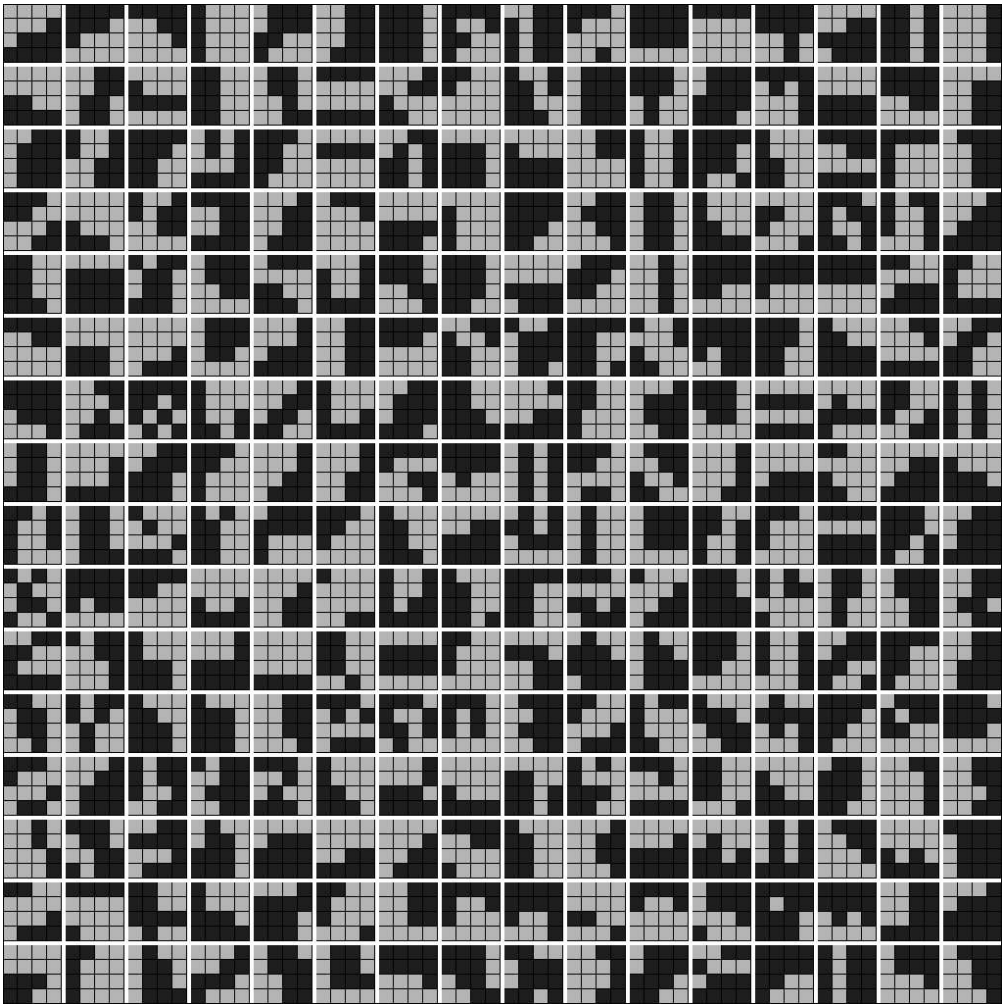


Figure 3.8: The binary patternbook used in LBTC-PF method to encode B_4 blocks.

the random variable for the index of the sub-block B_{4k}^{ij}. Using the joint frequency distribution of $X_1^{ij}X_2^{ij}$ and $X_1^{ij}X_1^{ij+1}$ two index graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are defined. V_1 is same as V_2 ($|V_1| = |V_2| = 257$). The vertices of G_1 and G_2 represents the indices of bit-patterns in patternbook and one auxiliary index for the blocks with $d' = 0$. If d'=0 then the auxiliary index is actually not transmitted. It may be noted that, if higher the degree of the vertices, lot of combinations will come up in the joint frequency distribution. Thus, more bits are required for representation. If vertex v_j of v_iv_j is not adjacent to v_i then we represent v_j by a 'bit-sequence' (indicates non-

adjacency) followed by the binary representation of v_j. If v_j is adjacent to v_i and let it is k^{th} adjacent vertex. Then v_j is represented by the binary representation of k as v_i is already encoded. However, if we consider lower degree then rate of non-adjacency becomes higher and then 'bit-sequence' followed by binary representation consumes more bits. Thus, for optimal representation, degree is to be selected judiciously and we have experimentally chosen the degree ≤ 31. Hence, to represent the adjacency 5 bits (binary representation of 1 to 31) are required and the '00000' is can be used as the 'bit-sequence'.

Now, we use the index graph to encode the indices. First of all, find the best-fit bit-pattern by linear search method on original patternbook. Now consider x_1^{ij}, x_2^{ij} and x_1^{ij+1} are the indices for the sub-blocks B_{41}^{ij}, B_{42}^{ij} and B_{41}^{ij+1} respectively. Further, assume that x_1^{ij} already coded. Now, x_2^{ij} is coded using G_1 as follows.

Step 1 If $d_2^{'ij}$ is non-zero

Step 1.a If x_1^{ij} is adjacent of x_2^{ij}

Step 1.a.1 Let x_2^{ij} is k^{th} (≤ 31) adjacent of x_1^{ij} then the encoding string of x_2^{ij} is the binary representation of k.

Step 1.b Otherwise (if step 1.a or step 1.a.1 fails) the encoding string of x_2^{ij} is '00000' followed by the binary representation of x_2^{ij}.

Step 2 If $d_2^{'ij}$ is zero, no bit-stream.

Similarly, x_1^{ij+1} is encoded using G_2.

3.3.2 Decoding the block

The encoded bit-stream of LBTC-PF method is the block wise collection of $(\triangle A_1, A_1 - A_2, d_1', d_2', I_1, I_2)$. Depending on d' the indices I_1 or I_2 may or may not be present. For a block the decoding scheme applied in all blocks B_8, is as follows.

Step 1

Step 1.a The value of $\triangle A_1$, $A_1 - A_2$, d_1', and d_2' are obtained by table look-up method

Step 1.b Compute A_1 and A_2.

Step 2

Step 2.a If d_1' is zero, then I_1=0, i.e., an auxiliary index.

Step 2.b If $d_1' \neq 0$

Step 2.b.i If the index bit-stream for I_1 contains '00000', then I_1 is the integer equivalent of the following 8 bits.

Step 2.b.ii Else

Step 2.b.ii.α Consider the graph G_1, the I_1 of the previous B_8 block (I, say), t (the integer equivalent of the current five bits).

Step 2.b.ii.β The t^{th} children of node I of G_1 is the I_1 of the current B_8 block.

Step 3

Step 3.a If d_2' is zero, then I_2=0, i.e., an auxiliary index.

Step 3.b If $d_2' \neq 0$

Step 3.b.i If the index bit-stream for I_2 contains '00000', then I_2 is the integer equivalent of the following 8 bits.

Step 3.b.ii Else

Step 3.b.ii.α Consider the graph G_2, the I_1 of the current block, t (the integer equivalent of the current five bits).

Step 3.b.ii.β The t^{th} children of node I_1 of G_2 is the I_2 of the current B_8 block.

Step 4 Now (A_1, d_1', I_1) and (A_2, d_2', I_2) are available and reconstruct the block B_{41} and B_{42}.

Step 5 Reconstruct the block B_8

Step 5.a From reconstructed B_{41} and B_{42}, assign the value to the pixels given by QS.

Step 5.b Interpolation technique is used to assign the value to the missing pixels.

3.3.3 Performance of LBTC-PF method

Like 2BTC-PF method, the performance of LBTC-PF method is also measured in terms of PSNR (Eq. (1.4)), bpp (Eq. (1.1)) and FI (Eq. (1.7)). The results are evaluated on the collection of grayscale images shown in Fig. 1.4. The step-by-step refinement of the LBTC-PF method for 'Lena' image is given in Table 3.4, and the results on 'Lena' is shown in Fig. 3.9. The steps in Table 3.4 are: (i) BTC, the conventional BTC method is used to encode the image; (ii) QS-BTC, each B_4 sub-blocks of B_8 is encoded by conventional BTC; (iii) QS-PF, the B_4s are encoded by BTC-PF method and to calculate the bpp, the predictive entropy coding of A_1, entropy coding of A_1 -A_2, and d-values are used, and the indices are coded straightway; (iv) In QS-PF-d'-IG, first of all d is approximated to d' (by Eq.(3.8)), the index graph (IG) is used to encode the indices and remaining information are encoded by entropy

Table 3.4: Results of the different steps of LBTC-PF method.

Steps	Average PSNR	Average bpp
BTC	32.34	2.00
QS-BTC	29.49	1.00
QS-PF	28.40	0.54
QS-PF-$d^{'}$-IG (LBTC-PF)	28.29	0.43

coding (like previous step). This index graph concept leads to low bit per pixel and also preserved the quality. Finally, the result of LBTC-PF method is compared with CIBTC-VQ, Adaptive D/I, and 2BTC-PF. The comparative results are shown in Table 3.5. The result reveals that the LBTC-PF method gives a better trade-off among the performance metrics.

(a) (b) (c) (d)

Figure 3.9: Step-by-step results of LBTC-PF method on 'Lena': (a) BTC (PSNR = 32.89 dB, bpp = 2), (b) QS-BTC (PSNR = 30.36 dB, bpp = 1), (c) QS-PF (PSNR = 29.56 dB, bpp = 0.53), (d) QS-PF-$d^{'}$-IG (PSNR = 29.43 dB, bpp = 0.39).

Table 3.5: Comparative results of LBTC-PF method with other BTC based methods.

Images	LBTC-PF			2BTC-PF[2]			CIBTC-VQ			Adaptive D/I			
	PSNR	bpp	FI	PSNR	bpp	FI	PSNR	bpp	FI	PSNR	bpp		FI
											without predictive	with predictive	
Airplane	27.69	0.38	0.4996	30.52	0.60	0.6557	30.15	0.76	0.6822	31.58	1.30	1.15	0.7060
Boat	27.76	0.41	0.5333	30.03	0.70	0.6715	29.48	0.80	0.6632	29.86	1.41	1.21	0.7034
Couple	29.36	0.42	0.6030	31.65	0.75	0.7693	30.61	0.82	0.7363	32.42	1.52	1.19	0.7967
Lake	23.96	0.50	0.5297	26.16	0.94	0.7150	25.70	0.89	0.6602	27.87	1.94	1.70	0.8124
Lena	29.43	0.39	0.5574	31.59	0.64	0.6965	30.92	0.78	0.6928	31.78	1.42	1.13	0.7347
Man	27.82	0.45	0.6074	29.96	0.80	0.7666	29.33	0.84	0.7500	31.20	1.59	1.32	0.8086
Peppers	29.02	0.42	0.5336	31.69	0.68	0.6803	30.76	0.79	0.6448	32.69	1.60	1.25	0.7308
Zelda	32.54	0.40	0.6542	34.89	0.60	0.7654	33.57	0.76	0.7597	34.04	1.39	1.04	0.7944
Tiffany	31.31	0.35	0.4995	33.80	0.57	0.6589	31.58	0.75	0.6303	32.70	1.30	1.03	0.7024
Goldhill	28.74	0.47	0.6193	30.80	0.84	0.7840	29.85	0.86	0.7454	30.03	1.71	1.37	0.8199
Splash	32.59	0.31	0.4166	35.35	0.45	0.5784	34.25	0.71	0.6005	35.70	1.14	0.88	0.6319
Baby	29.09	0.47	0.6582	31.96	0.80	0.7981	30.84	0.60	0.7795	32.47	1.56	1.28	0.8220
Average	29.11	0.42	0.5593	31.54	0.70	0.7116	30.59	0.78	0.6954	31.86	1.49	1.22	0.7526

[2]The result of 2BTC-PF are generated using the patternbook given in Fig. 2(a).

3.4 Conclusion

In this chapter, BTC-PF based grayscale image compression is presented. In grayscale compression method two level patterns are used to encode the image blocks. To evaluate the performance, two different patternbooks are used and the result shows that their performances are comparable. These two patternbooks (2BTC-PF) are generated by two different approaches: (i) considering geometrical orientations of the edges and line, and (ii) by applying clustering algorithm. The concept of BTC-PF is extended for low bit-rate coding of the images. To achieve the low bit-rate, quincunx sub-sampling method is used to 8×8 block and then BTC-PF method is employed. These methods (2BTC-PF and LBTC-PF) are proposed to achieve high compression keeping PSNR as high as possible. At the same time, the FI of the reconstructed images are quite good and comparable with other methods. Here representative levels are computed in such a way that they denote the bias and contrast of the block. One advantage of this kind of representation of levels is that as the bias components of neighboring blocks are strongly correlated, the predictive coding can successfully be used. Secondly, as low contrast component indicates that the block is smooth, in that case no bit-pattern is needed resulting into more compression. The performances (mainly bit-rate with a little sacrifice in quality) of both the methods can be improved by considering the blocks of variable size. Here, we have dealt with grayscale images only. But, the concept of BTC-PF method can be further extended for the compression of color images and it is presented in the next chapter.

Chapter 4

Color Image Compression using BTC-PF

4.1 Introduction

In this chapter, a color image compression technique based on BTC-PF method is proposed. 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. Thus, to transmit or to store a color image, huge bandwidth or storage space is required. To reduce storage space or transmission cost, image compression technique is employed. In grayscale image there is a correlation between neighbor pixels, whereas in color image additionally there is also high correlation between color components (spectral redundancy). The straightforward way to compress color image is to compress each of the three color components separately using similar algorithm used for grayscale image compression. However, this approach does not exploit the correlation between the color planes. An alternative is to convert from RGB model to other models like YIQ, YUV, YCbCr.

Then apply grayscale image compression method on each component. This is done to improve the performance significantly because the correlation among the components of YIQ etc. is much less than that of RGB components. In these Y** representation of a color image, Y contains the intensity value of the pixels (luminance component) and the other two contain color information (chrominance component). As human visual system cannot distinguish all colors in a true-color image, chrominance components are compressed more than the corresponding luminance component.

In chapter 2, different variations of BTC and VQ based methods have been discussed. These techniques can be applied separately on each color plane. As there is high degree of correlation among the three color planes [42], this can be utilized to reduce the bit-rate further. A single-bit-map BTC (SBBTC) method [171], exploits the inter-color correlation by using single bit-map to quantize all the 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) (BTC method to all the color planes) is 6 bits/pixel. Kurita and Otsu [89] have suggested another variation of SBBTC method which generates the single bit-map by employing an adaptive one-bit vector quantization using the principle score of each pixels in the block. Tai et al [148] describe a SBBTC method with Hopfield Neural Network (HNN), where HNN is used to generate the single-bit-map of blocks. Yang et. al. [175] describe a color image compression method based on BTC and moment preserving principle (CICMPBTC). This method also uses single bit-map for all three color components. 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. For example, if the red component is uniform then using the spectral index the reconstructed vectors may be represented as C_1=(R_1, G_1, B_1) and C_2=(G_2, B_2). Again CICMPBTC method is modified to CICMPBTC* considering: (i) quantization of each color component from 8 bit to 5 bit, and (ii) by selecting the bit-map from a

set of predefined 64 bit-maps. Similar idea of using predefined set of bit-maps is used in [116].

It is obvious that applying BTC to each color plane, a good quality can be maintained but bit-rate will become high. A better compression can be achieved with a little lower visual quality by BTC-PF method. The results of 2BTC-PF on 'Lena', a grayscale image of size 512×512, is PSNR = 31.59 dB with 0.64 bpp, while that of BTC [30] method is 32.89 dB with 2 bpp.

In color image high spectral correlation exists between R, G and B planes, so a better compression can be achieved by also exploiting this inter-plane redundancy along with spatial redundancy and code redundancy. To do so, in our method, first of all an image is transformed from the triplet RGB to a less correlated triplet $O_1O_2O_3$, to reduce the inter-plane redundancy. Then, the spatial redundancy is removed by BTC-PF method and this is followed by entropy coding to reduce code redundancy. Let us call this method as color block truncation coding with pattern fitting (CBTC-PF) method. The visual quality index of the reconstructed images are measured in terms of PSNR as well as image fidelity metric [152], which quantifies how a processed color image appears relative to the original image, as perceived by human observer.

The remaining part of this chapter is organized as follows. The color transformation is presented in section 4.2. The image encoding scheme is discussed in section 4.3, and section 4.4 is focused on the image decoding technique. Experimental results and comparative study are shown in section 4.5. Finally, conclusions are made in section 4.6.

4.2 RGB to $O_1O_2O_3$ conversion

There are standard color image conversion scheme 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 aforementioned transformations are, in a sense, lossy transformations. So in the CBTC-PF method we have used the transformation from RGB to $O_1O_2O_3$ transformation [86, 110] 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 4.5). RGB to $O_1O_2O_3$ transformation is given by

$$O_1 = \lfloor \frac{R+G+B}{3} + 0.5 \rfloor \tag{4.1}$$

$$O_2 = \lfloor \frac{R-B}{2} + 0.5 \rfloor \tag{4.2}$$

$$O_3 = B - 2G + R \tag{4.3}$$

and the corresponding inverse transformation is

$$B = O_1 - O_2 + \lfloor \frac{O_3}{2} + 0.5 \rfloor - \lfloor \frac{O_3}{3} + 0.5 \rfloor \tag{4.4}$$

$$G = O_1 - \lfloor \frac{O_3}{3} + 0.5 \rfloor \tag{4.5}$$

$$R = O_1 + O_2 + O_3 - \lfloor \frac{O_3}{2} + 0.5 \rfloor - \lfloor \frac{O_3}{3} + 0.5 \rfloor \tag{4.6}$$

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.

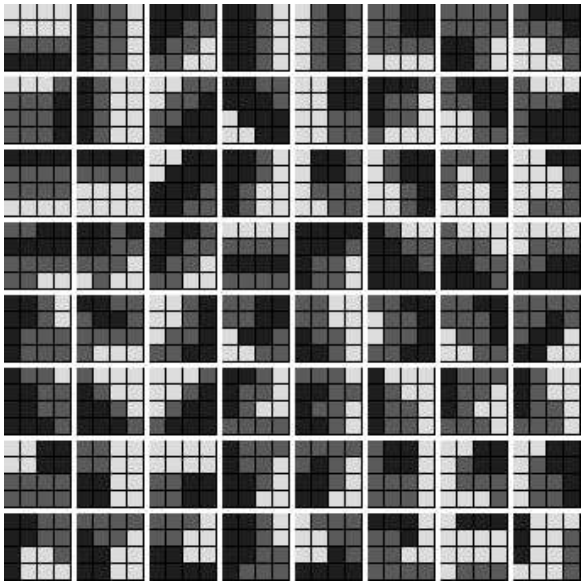


Figure 4.1: 3-level patterns used to define the graylevel pattern of image block of O_1 plane.

4.3 Image Encoding

In $O_1O_2O_3$ domain, O_1 contains major information and is 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 that more bits may be allotted for O_1 plane in comparison to O_2 and O_3 planes. This requirement is reflected in setting the parameters of patternbook. In the BTC-PF method, we have used three parameters Q (number of levels of the pattern), n (pattern size) and M (the size of the patternbook). In our experiment, to generate the patternbook for O_1 plane, the parameters Q, n and M are set to 3, 4 and 64, respectively; whereas for image planes O_2 and O_3 the parameters are set to 2, 4 and 16. Accordingly, the patternbook obtained for O_1 is shown in Fig. 4.1 and Fig. 4.2 shows the patternbook for O_2 and O_3 planes. The encoding method for the color images using CBTC-PF is shown in Fig. 4.3, and the detailed coding techniques are presented in the following sections.

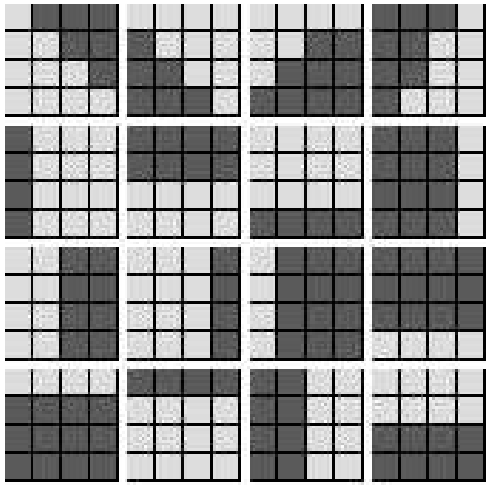


Figure 4.2: 2-level patterns: used to define the graylevels of 4×4 sub-blocks generated from O_2 and O_3 planes.

4.3.1 Coding Scheme: 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. O_1 being the intensity of the pixels, these values are spatially correlated. By exploiting the spatial redudancy, we may estimate the intensity within an image block from its negihboring blocks. In that case the residue depicts the local variation in intensity or graylevel due to the slope of the object surface and the incident intensity distribution. Patternbook is also generated using these residual graylevel distributions over the block. Simplest estimate may be the block mean [4] or the intensity plane [96]. But, these are some kind of global estimation and do not give always good estimation. For better results, we have adopted pixel-wise estimation. In our work, an image block $B(r, c)$ is first estimated as $\hat{B}(r, c)$ using information available in the encoded blocks $\tilde{B}(r, c-1)$ and $\tilde{B}(r-1, c)$ [see Fig. 4.4] as:

$$\hat{B}(r, c) = \frac{v}{v+h} val_h + \frac{h}{v+h} val_v \tag{4.7}$$

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

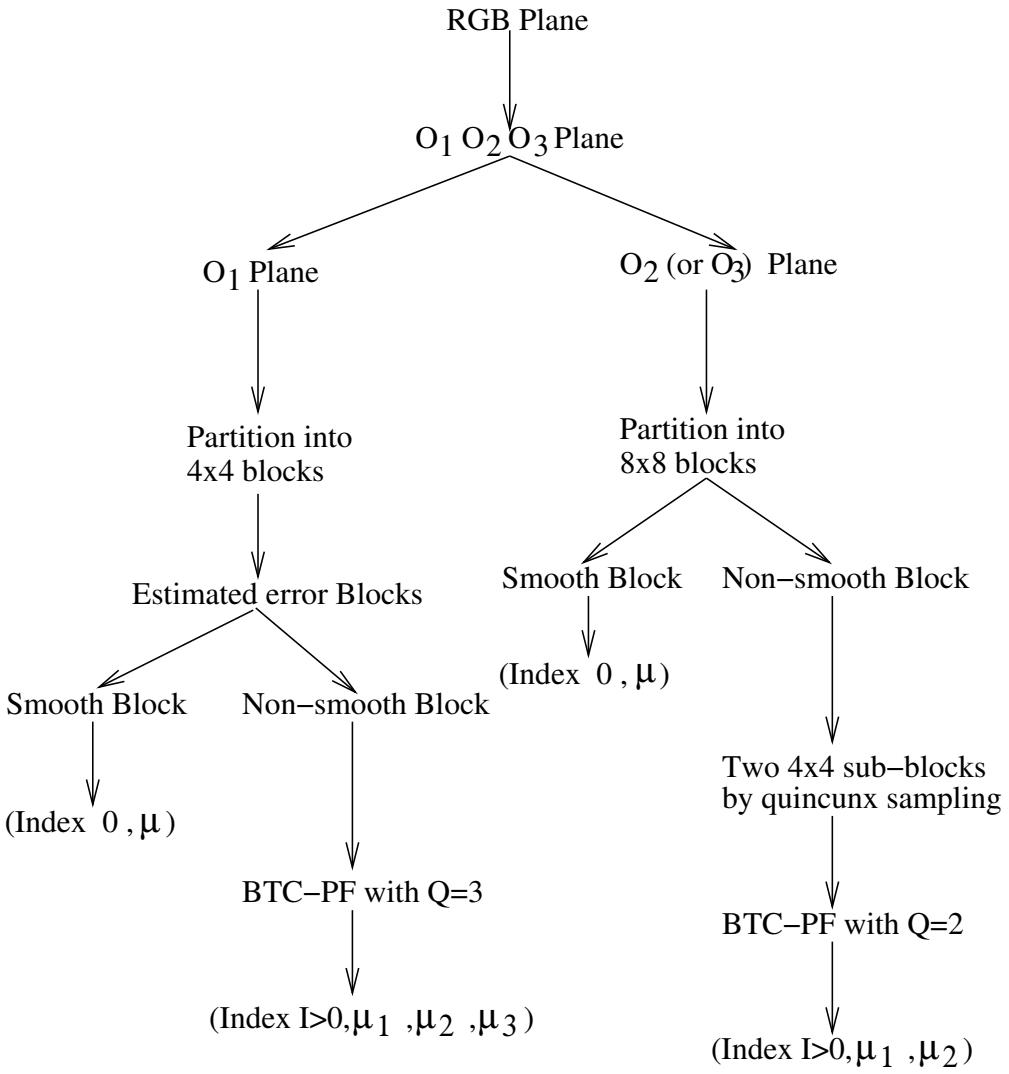


Figure 4.3: Pictorial representation of the image encoding process using CBTC-PF method.

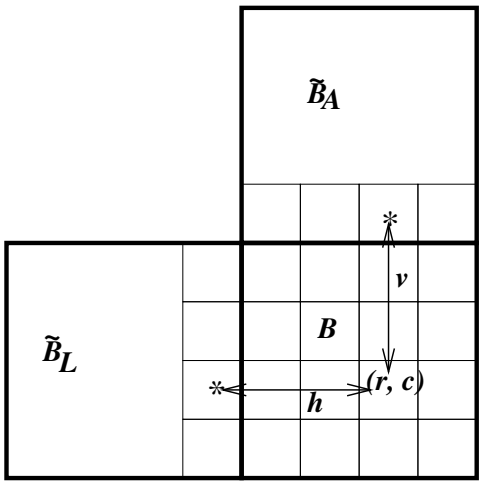


Figure 4.4: Estimation of $x_i \in B(r, c)$ using information from vertical and horizontal '*'-marked positions of neighbor blocks.

'*'-marked position as shown in Fig. 4.4. The estimation error is

$$B_e(r, c) = B(r, c) - \hat{B}(r, c) \tag{4.8}$$

The residual block $B_e(r, c)$ thus obtained is actually coded by the BTC-PF method, where the quantization levels are the mean values of the corresponding partitions of the block defined by the selected pattern. 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 patternbook. Since, the residual block $B_e(r, c)$ contains a significant number of zero values and some positive and negative values, we have chosen $Q = 3$ to determine the patternbook for the residual blocks of the O_1 plane. Secondly, if the number of patterns in the patternbook increases then the quality of the reconstructed image improves 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. 4.1.

4.3.2 Coding Scheme: 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 can be 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), then the block is considered as smooth block 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 sub-sampling (as explained in section 3.3). If we consider high value of T then it results more smooth block. Hence, both the quality and bit-rate becomes lower. So, we set the threshold (T) experimentally compromising between quality and bit-rate. Each sub-block is then encoded by BTC-PF with Q=2 and a patternbook of 16 patterns as shown in Fig. 4.2 is used. The quantization levels are the mean values of the block partitions, as defined by the selected pattern.

4.3.3 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, μ) [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 $\mu_1^{'}$, $\mu_2^{'}$ and $\mu_3^{'}$ such that $\mu_1^{'} \leq \mu_2^{'} \leq \mu_3^{'}$ 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 $(\mu_1^{'}, \mu_2^{'}, \mu_3^{'})$ may be defined by using some extra bits, called *ordering index* (OI) denoted by I_o. An example of ordering

Table 4.1: Mapping between (μ_1, μ_2, μ_3) and $(\mu_1^{'}, \mu_2^{'}, \mu_3^{'})$ using OI

OI I_o	Relation between μ_1, μ_2, μ_3	Mapping from μ_i to $\mu_j^{'}$	Mapping from $\mu_i^{'}$ to μ_j
00	$\mu_1 \leq \mu_2 \leq \mu_3$	$\mu_1^{'} = \mu_1,\ \mu_2^{'} = \mu_2,\ \mu_3^{'} = \mu_3$	$\mu_1 = \mu_1^{'},\ \mu_2 = \mu_2^{'},\ \mu_3 = \mu_3^{'}$
01	$\mu_1 \leq \mu_3 \leq \mu_2$	$\mu_1^{'} = \mu_1,\ \mu_2^{'} = \mu_3,\ \mu_3^{'} = \mu_2$	$\mu_1 = \mu_1^{'},\ \mu_2 = \mu_3^{'},\ \mu_3 = \mu_2^{'}$
100	$\mu_2 \leq \mu_1 \leq \mu_3$	$\mu_1^{'} = \mu_2,\ \mu_2^{'} = \mu_1,\ \mu_3^{'} = \mu_3$	$\mu_1 = \mu_2^{'},\ \mu_2 = \mu_1^{'},\ \mu_3 = \mu_3^{'}$
101	$\mu_2 \leq \mu_3 \leq \mu_1$	$\mu_1^{'} = \mu_2,\ \mu_2^{'} = \mu_3,\ \mu_3^{'} = \mu_1$	$\mu_1 = \mu_3^{'},\ \mu_2 = \mu_1^{'},\ \mu_3 = \mu_2^{'}$
110	$\mu_3 \leq \mu_1 \leq \mu_2$	$\mu_1^{'} = \mu_3,\ \mu_2^{'} = \mu_1,\ \mu_3^{'} = \mu_2$	$\mu_1 = \mu_2^{'},\ \mu_2 = \mu_3^{'},\ \mu_3 = \mu_1^{'}$
111	$\mu_3 \leq \mu_2 \leq \mu_1$	$\mu_1^{'} = \mu_3,\ \mu_2^{'} = \mu_2,\ \mu_3^{'} = \mu_1$	$\mu_1 = \mu_3^{'},\ \mu_2 = \mu_2^{'},\ \mu_3 = \mu_1^{'}$

index and the corresponding mapping from (μ_1, μ_2, μ_3) to $(\mu_1^{'}, \mu_2^{'}, \mu_3^{'})$ are given in the first three columns of Table 4.1. Hence, for a block of O_1 plane the encoder gives out either $(0, \mu)$ or $(I,\ I_o,\ \mu_1^{'},\ \mu_2^{'} - \mu_1^{'},\ \mu_3^{'} - \mu_2^{'})$.

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 [33] where $\mu_{11}^{''} = \lfloor \frac{\mu_{11}+\mu_{12}}{2} \rfloor$, $\mu_{12}^{''} = \lfloor \frac{\mu_{11}-\mu_{12}}{2} \rfloor$, $\mu_{21}^{''} = \lfloor \frac{\mu_{21}+\mu_{22}}{2} \rfloor$ and $\mu_{22}^{''} = \lfloor \frac{\mu_{21}-\mu_{22}}{2} \rfloor$. 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 separately. In our experiment, we have adopted Huffman coding scheme.

4.4 Image Decoding

The encoding string for a block of a O_1 plane is either (0,μ) or (I, I_o, $\mu_1^{'}$, $\mu_2^{'}$-$\mu_1^{'}$, $\mu_3^{'}$-$\mu_2^{'}$). For a block of O_2 and O_3 planes, it is either (0,μ) 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 μ_1, μ_2 and μ_3 are obtained from I_o, $\mu_1^{'}$, $\mu_2^{'}$ and $\mu_3^{'}$ by using the mapping defined in the last column of Table 4.1. Finally, the block (B_e) of the O_1 plane is reconstructed ($B_e^{\prime}$) 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 (explained in section 4.3.1). In case, it is residual block, we first estimate the block using Eq. (4.8) 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, $\mu_{11}^{''}$, $\mu_{12}^{''}$) and (I_2, $\mu_{21}^{''}$, $\mu_{22}^{''}$) two blocks B_{41} and B_{42} are reconstructed following the BTC-PF method [33]. Finally, B_8 is reconstructed by filling the alternate pixels in a manner that is reverse of quincunx sampling. Unfilled pixel (i.e., white pixels shown in Fig. 3.7(a)) values are obtained by linear interpolation based on nearest known pixel values.

It may be recalled that we claim CBTC-PF method incurs 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 the index value requires at most two additions.
- Estimation of block ($\hat{B}$) [by Eq. (4.7)] from neighbors $\tilde{B}$ (as shown in Fig. 4.4) requires 32 multiplication and 16 additions. However, since there are only 11 different weights and 256 different values (*val*) [see Eq. (4.7)] 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 2 additions only.
  - Remaining 32 pixel values are determined by linear interpolation. Let each interpolation needs C number of computation (where C equals to at most 3 additions and two bit-shift operations). Thus, the total is 32C. In worst case, 32C means 96 addition and 64 bit-shift operations.

Thus, in CBTC-PF method to reconstruct a 16×16 color block, in worst case, 576 addition and 256C (for interpolation) operations are required. On the other hand, JPEG required 1312 multiplications, 2784 additions and 384C operations to reconstruct a 16×16 block (as will be shown in the next section).

4.5 Experimental Results

In this section, the result of CBTC-PF method 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 shown in Fig. 1.5. The sizes of the patternbooks 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 patternbook. 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 Table 4.2 and Table 4.3, 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 subsection 4.2 that better performance can be achieved with $O_1O_2O_3$ system compared to YIQ etc. This may be supported by the Table 4.4 where performance of $O_1O_2O_3$ and YIQ are compared in terms of PSNR and bpp with same values of the parameters.

Table 4.5 presents the comparison of results of CBTC-PF, conventional color BTC (CBTC), single-bit-map BTC (SBBTC) [171, 148], CICMPBTC* [175] and JPEG [120, 159]. Here compression ratio is measured in terms of bpp and the image quality in terms of PSNR and visual fidelity index (FI) [152]. The bpp, PSNR and FI are defined by Eq.(1.1), Eq. (1.4) and Eq. (1.8).

To evaluate the performance of both CBTC and SBBTC methods block size is set to 4×4. In CBTC method, conventional BTC [30] 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 $(16 + 2 \times 8 \times 3)$

Table 4.2: 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
Image Size 512×512, Total Number of 4×4 Blocks 16384								
Airplane	656	34.80	3897	33.97	9275	31.86	11143	31.18
Lena	12	35.25	520	35.23	6810	34.90	10937	34.21
Peppers	2	35.50	43	35.50	3420	34.94	9200	33.55
Splash	286	39.49	2691	37.82	11452	34.22	14356	33.20
Tiffany	72	37.37	1897	36.73	9722	33.50	12283	32.53
Image Size 256×256, Total Number of 4×4 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
Image Size 480×480, Total Number of 4×4 Blocks 14400								
Lab	225	38.63	3395	38.39	8371	36.96	10935	35.16
Average	2.69%	36.37	21.07%	36.00	52.77%	34.71	71.38%	33.61

Table 4.3: 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
Image Size 512×512, Total Number of 8×8 Blocks 4096											
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
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
Splash	O_2	2814	35.81	3035	35.72	3192	35.60	3296	35.47	3358	35.36
	O_3	3015	38.68	3293	38.49	3422	38.30	3535	38.08	3618	37.85
Tiffany	O_2	1821	35.50	2447	35.36	2884	35.19	3199	34.98	3434	34.75
	O_3	2063	35.98	2735	35.83	3197	35.50	3485	35.48	3687	35.31
Image Size 256×256, Total Number of 8×8 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
Image Size 480×480, Total Number of 8×8 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		63.17%	37.90	74.96%	37.71	82.43%	37.44	86.93%	37.21	89.92%	37.00

Table 4.4: Comparative results of CBTC-PF 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
Lena	31.93	1.17	31.63	1.20
Peppers	30.15	1.50	29.63	1.60
Splash	31.61	0.85	31.14	1.00
Tiffany	30.53	0.98	30.09	1.13
Image Size 256×256				
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
Girl	35.13	0.60	35.12	0.65
Image Size 480×480				
Lab	33.64	0.93	33.48	0.95
Average	31.89	1.04	31.60	1.12

bits are required to reconstruct the color block. A modified SBBTC method [148] using vector quantized color vectors achieves PSNR values 30.03 dB and 29.90 dB for 'Lena' and 'Airplane' images respectively for a fixed bpp 2.0. A variation of CICMPBTC* [175] achieves SNR values 29.14 dB, 28.60 dB and 28.54 dB for 'Lena', 'Airplane' and 'Peppers' images respectively and corresponding bpp are 1.95, 1.77 and 2.09. Thus, from these data as well as the experimental results presented in Table 4.5, it is evident that the proposed method is significantly superior in comparison to the known methods of this class of compression techniques that intend to achieve lower bpp than conventional BTC. The quality (in terms of PSNR as well as 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 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. 4.5.

It was mentioned earlier [see section 4.1] 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. We compare the performance of CBTC-PF with JPEG (we use the JPEG code of The Independent JPEG Group [58]). From the results reported in Table 4.5, the performance of CBTC-PF and JPEG are comparable. But, strictly speaking, JPEG is little superior to CBTC-PF. On an average CBTC-PF achieves PSNR 31.89 dB, 1.04 bpp with FI=0.5977; while these figures for JPEG are 32.63 dB, 1.00 bpp and 0.6419. 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. (1.6) which stands for correlation between original image X and the reconstructed image Y. The value of this term is high if for any two pixels $X(i) < X(j)$ implies $Y(i) \leq Y(j)$, which is true for BTC and CBTC but not for other cases. However, though not always, this

Airplane
(30.36, 1.04, 0.6068)

Lena
(31.93, 1.17, 0.6295)

Peppers
(30.15, 1.50, 0.6021)

Splash
(31.61, 0.85, 0.4684)

Tiffany
(30.53, 0.98, 05476)

Couple
(32.44, 1.00, 0.6919)

House
(31.79, 1.20, 0.5742)

Zelda
(31.31, 1.12, 0.6763)

Girl
(35.13, 0.60, 0.4114)

Lab
(33.64, 0.93, 0.7690)

Figure 4.5: The Reconstructed images due to CBTC-PF method and the values inside the bracket represent the PSNR, bpp and FI respectively. For example, 'Airplane' with PSNR=30.36 dB, bpp=1.04 and FI=0.6068.

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 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 analyze the computational cost of decoding part of JPEG method, here we only consider IDCT operation. Straight forward implementation of IDCT (same as DCT also) on n data requires n^2 multiplications and $n(n-1)$ additions. But, various fast algorithms for DCT already have been proposed [145, 101]. Loeffler et.al [101] have proposed one of the fastest algorithms which takes 11 multiplications and 29 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. Number of operations required to generate a 16×16 O_2 (O_3) block from a reconstructed 8×8 O_2 (O_3) blocks (after IDCT) by interpolation is 384C. Dequantization requires $16 \times 16 = 256$ multiplications. Hence, decoding part of JPEG compression (based on DCT algorithm in [101]) requires, in worst case, 1312 multiplications, 2784 additions and 384C operations; on the other hand, in section 4.4 we have shown that, in worst case, CBTC-PF method requires only 576 additions and 256C operations to reconstruct a color block of same size. Moreover, in CBTC-PF method, setting T=2 gives more than 52% smooth blocks of O_1 plane [see Table 4.2] and setting T=6 gives almost 87% smooth blocks of O_2 and O_3 planes [see Table 4.3]. The reconstruction of smooth block requires no operation and thus, high values of T will reduce the cost further. Hence, the decoding time for CBTC-PF method is practically negligible compared to that for JPEG. The result in Table 4.5 reveals that our method (CBTC-PF) offers a good trade-off compare to the other methods.

Table 4.5: 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
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
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
Splash	31.61	0.85	0.4684	35.91	6	0.8652	36.07	4	0.7585	32.20	2.44	0.5764	29.15	1.50	0.5568	32.79	0.88	0.5540
Tiffany	30.53	0.98	0.5476	35.03	6	0.8802	33.45	4	0.7496	30.50	2.80	0.6196	27.45	1.72	0.6211	30.71	1.01	0.5899
Image Size 256 × 256																		
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
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
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	31.89	1.04	0.5977	33.43	6	0.8792	33.18	4	0.7809	31.27	2.88	0.6462	28.72	1.74	0.6425	32.63	1.00	0.6419

†CICMPBTC* is a modified version CICMPBTC method.

4.6 Conclusions

A color image compression technique based on BTC with pattern fitting is presented in this chapter. In the proposed CBTC-PF method, to reduce inter-plane redundancy, (RGB) is transformed to ($O_1O_2O_3$) using lossless transformation as it has been observed that the algorithm performs better in $O_1O_2O_3$ domain compared to other well-known color domain, e.g., YIQ. We have selected the size of the blocks, size of patternbooks and the number of levels in pattern depending on the entropy of O_i ($1 \leq i \leq 3$) 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 lower than JPEG in terms of PSNR, though bit-rate and FI is more or less same. The computational cost at the decoding phase of the proposed method is negligible compared to that of JPEG, which is one of the requirements for the image that are frequently downloaded or decoded.

Thus, we have proposed BTC-PF scheme for grayscale image compression and it is extended to CBTC-PF to work with color images. In the context of transmission, compression methodology should support progressive transmission. In progressive transmission, images are transmitted in step-by-step and after decoding the transmitted image user can decide to continue or abort the transmission. Thereby, transmission wastage can be reduced. The BTC-PF/CBTC-PF method can also be extended to incorporate the progressive transmission capability and it will be detailed in the next chapter.

Chapter 5

Progressive Image Transmission using BTC-PF

5.1 Introduction

In the context of information sharing and communication through Internet, bandwidth is a major issue. Due to the restriction on bandwidth multimedia transmission becomes slow. Particularly during search, transmission of large images takes huge time. Furthermore, after transmission if it is found that the image is not the desired one then it results into huge wastage of time and bandwidth. Progressive image transmission (PIT) provides a solution for this. Initially, PIT method transmits the most significant information to the receiver so that the receiver can generate a rough sketch of the original image. From the initial sketch, the receiver can decide whether it is the desired image or not and accordingly may continue or abort the transmission. If the transmission continues, in the subsequent steps less significant information are transmitted to the receiver to improve the quality of the reconstructed image. A flow

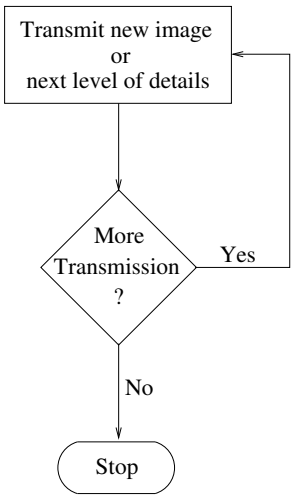


Figure 5.1: A flow diagram of progressive image transmission.

diagram for PIT is shown in Fig. 5.1. The advantages of PIT are:

1. users can grasp a rough image quickly without waiting for a long time to get the complete high quality image, and
2. users can quickly decide whether to continue with transmission or not. Thus, the discontinuation saves bits and time wasted otherwise.

Based on certain features in progressive image coding technique, they can be classified as follows:

- **Quality progressive (SNR progressive):** In this scheme, the image is reconstructed in full resolution (i.e., in full size). Intensity/Color information per pixel is transmitted progressively and thus by decoding more and more bits, a higher quality image is obtained.
- **Resolution progressive:** Here, the decoder reconstructs a lower resolution image just by using initial portion of the bit-stream. Then the difference to higher resolution is received and the image is refined step-by-step.

- **Component progressive:** In such scheme, luminance component of a color image is encoded first which is then followed by the chrominance components.

Tzou [155] has presented a thorough review and comparisons of some PIT methods. The PIT methods can be classified into *transform domain methods, pyramid-structure methods*, and *spatial domain methods*.

Transform domain PIT methods may be divided into different sub-classes depending on the transformation applied. Some of them [120, 25] are based on DCT, while some others are based on wavelet transformation. In JPEG 2000 [147, 27, 140, 43] wavelet based method is used to transmit images progressively. Transform domain PIT methods can also be classified depending on the order in which transform coefficients are transmitted. The techniques of embedded image coding using zerotrees (EZW) [138] and set partitioning in hierarchical trees (SPIHT) [135, 154, 149] transmit wavelet coefficients in different passes according to their significance and are used efficiently in PIT.

In the pyramidal approach, different levels of the pyramid structure correspond to successive approximation of an image. Accordingly, information of the pyramid structure is transmitted to the receiver from top to bottom. Different pyramid structure based PIT methods are reviewed in [55]. Among these methods, reduced-difference based structure gives the best result. Some methods based on Laplacian pyramid are presented in [6, 126].

Among the spatial domain methods, bit plane method (BPM) [155] is the simplest and straightforward one. In this method, the transmitter transmits one bit for each pixel in each iteration (or step), and the transmission of bits starts from most significant bit (MSB) to least significant bit (LSB). The major problems of BPM method are (i) the quality of image reconstructed from first few bit-planes may be very poor,

and (ii) the bit-rate is very high. One of the reasons behind the low quality of the reconstructed image in the initial level(s) of BPM lies in the fact that the reconstruction ability of each plane is fixed irrespective of actual pixel intensities. An improved BPM (IBPM) [22] has been proposed to enhance the image quality in the initial stage, but this method requires more data to be transmitted to the receiver. In IBPM, the reconstruction levels are the mean values of the pixels and these mean values are transmitted to the receiver along with the bit-plane. Guessing by neighbor method (GBN) [21] is proposed to reduce the bit-rate by exploiting the spatial correlations between neighbor pixels. A quadtree based PIT method is proposed in [67] where the images are decomposed into blocks and at the receiver end the blocks are reconstructed by block means. In [17], a block truncation coding based progressive scheme (PBTC) has been proposed where image blocks are first encoded by BTC [30] and then all the pixel positions represented by 0 and 1 are separately encoded by BTC again. It is repeated until all pixels have either same value or having only one pixel. Vector quantization (VQ) based method can also be used to reduce the bit-rate of PIT. A tree-structured vector quantization (TSVQ) based PIT scheme is proposed in [155], where in each subsequent step the best codeword is selected from the descendents of previously selected codeword. Multistage vector quantization (MSVQ) based schemes are proposed in [161, 72]. In index sampling methods (ISM-VQ) [19], first the image is compressed by VQ and then certain sampling technique (e.g., diagonal sampling technique, central sampling technique) is applied to select some indices and these are transmitted to the receiver. At the receiver end, corresponding blocks are reconstructed and remaining blocks are estimated from neighbor blocks. Another VQ based method using full-search progressive transmission tree (FSPTT) is presented in [131]. In such tree, the terminal nodes are generated by clustering method and all leaf nodes are at the same level. These leaf nodes are considered as final codewords. The internal nodes, considered as temporary codewords, are derived from the child nodes using bottom-up approach. A block is encoded through an exhaustive search

of the terminal nodes of the FSPTT. The indices of the selected codewords are then transmitted progressively, and internal codewords are used to reconstruct the block gradually. Recently, a multi-segment image coding scheme [119] is proposed and it consists of two basic stages: i) the lossy encoder, and ii) the residual data classifier with entropy coding.

Almost all the methods mentioned above are applicable for PIT of grayscale images. However, these methods can be used for progressive transmission of color images. Two approaches may adopt for this purpose. In the first approach, grayscale PIT technique is employed to each color planes (R, G, B) separately and then information is transmitted from each plane with equal importance. For example, BPM and PBTC methods can be extended and let us call these methods as CBPM and CPBTC, respectively. The expected bit-rate of these methods is very high. To reduce the bit-rate of CPBTC method, a method using common bit-map for each of three color blocks (CBMCPBTC) is presented in [24]. In other approach, first transform an image from RGB to another domain, say, XYZ. Then certain PIT method is employed on them and transmits more data from visually sensitive plane compared to other planes. These transmissions of data must be in interleaving fashion according to importance. GBN method addresses this concept for color image transmission.

In this chapter, PIT scheme using BTC-PF method (PBTC-PF) is introduced. In PBTC-PF method, we try to hybridize resolution progressive and SNR progressive methods. First, we discuss the basic concepts of the PBTC-PF method in section 5.2 and then we extend the scheme for lossless transmission of grayscale image. Section 5.3 is focused on the lossless transmission scheme. The PBTC-PF method is subsequently extended for color image transmission which is presented in section 5.4. The concluding remarks on PIT are made in section 5.5.

5.2 Fundamentals of PBTC-PF

In PBTC-PF method, like BTC-PF, images are partitioned into $n \times n$ (n=4) blocks and each block is encoded by BTC-PF method (see section 2.4). In this technique, 2-level 4×4 binary patterns are used to encode the image blocks. The patternbook is organized like a full search progressive transmission tree (FSPTT) and the leaf nodes of the tree form the codebook used by the BTC-PF method. A portion of FSPTT structure is shown in Fig. 5.2, where at each level we have a subset of patterns present at its child level. At root level we have an empty set of patterns and consider each block as a smooth block making the system resolution progressive. Index of a node given in Fig. 5.2 is the relative index of the node with respect to the parent node. The actual index of any node (pattern), other than the root, is its relative index followed by the actual index of its parent (see Fig. 5.3). The tree has four levels. The root node is at the first level, and it has eight children, which represents horizontal edge, vertical edge, and diagonal edges (45^0 and 135^0) and their complements, at the second level. The third and fourth levels have 32 and 128 nodes altogether respectively in the form of a quadtree. All the intermediate patterns of FSPTT (except root) are also present at the next level. Parent node is one of the child nodes. The child node which has the minimum average Hamming distance from the other nodes is consider as the parent of these nodes. Here, the terminal nodes are generated by clustering algorithm. An example of BTC-PF with FSPTT is shown in Fig. 5.3. The size of the actual patternbook used in BTC-PF method is 128 (where last 64 patterns are the complements of the first 64 patterns). The BTC-PF method returns index I, and mean graylevels μ_1, μ_2 ($> \mu_1$). Instead of μ_1 and μ_2 two levels A and d defined as $A = \frac{\mu_1+\mu_2}{2}$ and $d = \frac{\mu_2-\mu_1}{2}$ are transmitted to the receiver. The block is reconstructed with index I and graylevels $A-d$ and $A+d$. Normally, the range of d is significantly less than those of μ_1 or μ_2, and hence less than that of A. This guides us to transmit A first, then d. Thus, relatively higher quality at the initial stage(s)

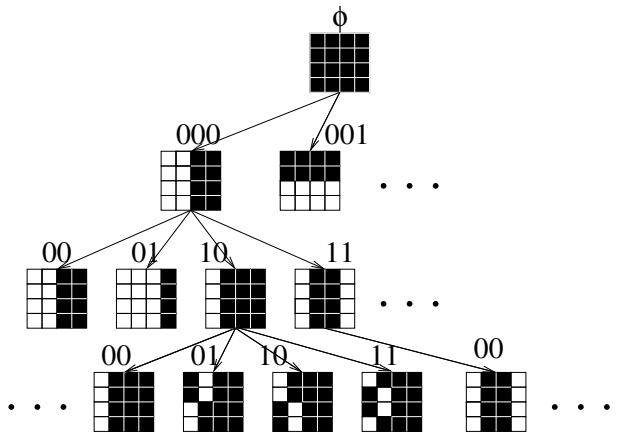


Figure 5.2: A partial structure of FSPTT used in BTC-PF.

155 202 215 203	201 201 201 201	171 171 201 201	171 201 201 201	171 201 201 201
168 193 210 204	201 201 201 201	171 171 201 201	171 201 201 201	171 201 201 201
183 180 205 208	201 201 201 201	171 171 201 201	171 201 201 201	201 171 201 201
199 180 200 205	201 201 201 201	171 171 201 201	171 201 201 201	201 171 201 201
(a)	(b)	(c)	(d)	(e)

Figure 5.3: Illustration of BTC-PF using FSPTT (a) Original image block, BTC-PF returns A=186, d=15 and I=1010000 (middle most pattern of the last level) of Fig. 5.2. Step-by-step refinement of the reconstructed block, (b) I=ϕ, MSE= 303.25, (c) I= 000, MSE= 198.25, (d) I= 10000, MSE= 157.00, and (e) I=1010000, MSE= 74.50.

could be achieved. In PIT, the root codeword of Fig. 5.2 is the initial codeword for the blocks of size 4×4. In FSPTT, the index of the leaf node is transmitted step-by-step. Hence, in PBTC-PF method, the grayvalues A, d and the index I are transmitted in interleaved fashion (hybrid of SNR progressive and resolution progressive). The detail of the progressive transmission scheme is as follows and executes this for each image block.

Progressive Algorithm (Palgo)

Step 1 Transmit A-value in two steps where $A_q = \lfloor A/F \rfloor$ (i.e., most significant bits) are transmitted first followed by $A_r = A\%F$ (i.e., least significant bits) for some fixed value F ($=2^k$). This two-step transmission saves bits if the transmission needs to be aborted after the initial sketch. The detail of the transmission is as follows.

1.a Transmit $\mathbf{A_q}$ to the receiver and then reconstruct the block with the value $A_q \times F$ and root pattern of FSPTT (Fig. 5.2) as the current pattern.

1.b Transmit A_r and refine the reconstructed using the same pattern and $A = A_q \times F + A_r$.

Step 2 Transmit three bits [there are eight nodes at level 2] corresponding to the intermediate pattern (P_{2i}) at level 2 (assuming root at level 1), along the path from root to the selected leaf node of Fig. 5.2.

Step 3 Transmit d value (like transmission of A) in two steps as $d_q = \lfloor d/F_1 \rfloor$ and $d_r = d\%F_1$ for some other fixed value F_1 ($=2^t, t < k$).

3.a Transmit d_q to the receiver and then reconstruct the block using P_{2i}, A, and $d_q \times F_1$.

3.b Transmit d_r, and refine the reconstructed block with P_{2i}, A, and $d{=}d_q{\times}F_1 + d_r$.

Step 4 Two more bits [each node at level 2 has four children] are transmitted to represent the pattern (P_{3j}) at level 3, along the same path. Reconstruct the block with P_{3j}, A and d.

Step 5 Finally last **two bits** [each node at level 3 has four children] are transmitted to represent the actual pattern (P_{4l}) at leaf level, selected by BTC-PF, and refine the block with P_{4l}, A and d.

Now, the **Progressive Algorithm (Palgo)** is explained with an example shown in Fig. 5.4. Original image block is shown in Fig. 5.4(a). The BTC-PF method returns I=1010000, A=186, and d=15. Suppose F=16, and F_1=8. Step 1 of **Palgo** results in A_q=186/16=11 and A_r=186%16=10. Now A_q=11 is sent to the receiver and the uniform block is reconstructed using grayvalue 176 (=11 $\times$ 16). The result is shown in Fig. 5.4(b). Then A_r=10 is transmitted and the reconstructed block is refined to have gary-value 186 (176+10) as shown in Fig. 5.4(c). In the Step 2 of **Palgo** three bits representing intermediate pattern I=000 of level 2 is sent. Step 3 of **Palgo** gives d_q=15/8=1 and d_r=15%8=7. Transmit d_q=1 and reconstruct block with intermediate pattern (represented by the I=000) and graylevels 178 and 194 as shown in Fig. 5.4(d). Then transmit d_r=7, the refined graylevels are 171 and 201. The reconstructed block is shown in Fig. 5.4(e). Then, transmit two more bits (10) to represent intermediate pattern, I=10000, of level 3 of Fig. 5.2 as the current pattern and the block is reconstructed with graylevels 171, 201 and the new pattern (step 4). This block is shown in Fig. 5.4(f). Finally, as mentioned in step 5, send last two bits (10) to represent the actual pattern (selected by BTC-PF, I=1010000). The block is refined with same graylevels using the current pattern and obtained as shown in Fig. 5.4(g).

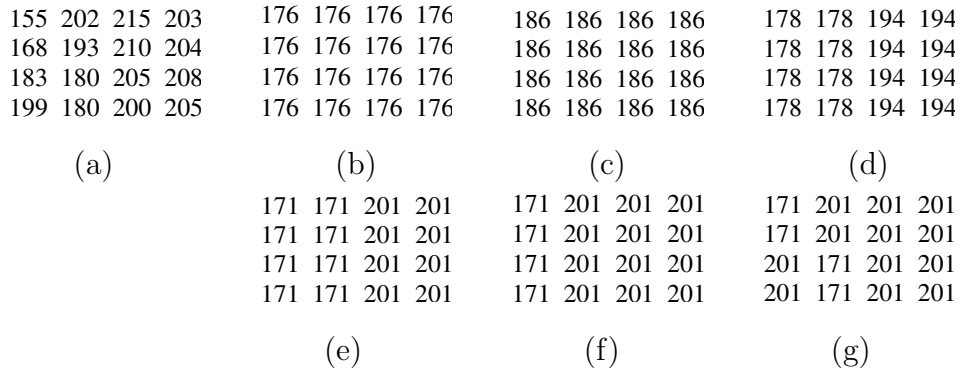


Figure 5.4: Illustration of PBTC-PF with F=16 and F_1=8: (a) Original image block, BTC-PF returns A=186, d=15, I=1010000 of Fig. 5.2, (b)I=ϕ, MSE=597.00, (c) I=ϕ, MSE=329.50, (d) I=000, MSE=203.50, (e) I=000, MSE=198.25, (f) I=10000, MSE=157.00, (g) I=1010000, MSE=74.50.

5.3 Lossless progressive transmission

We extend our PBTC-PF scheme to make it lossless. This technique is a novel combination of lossy and lossless coding techniques, call this method as LPBTC-PF. This particular technique has two phases: the phase-I uses the PIT concept described in section 5.2. In phase-II, a lossless technique is applied on the residual image obtained from first phase. For this purpose, context-based adaptive lossless image coding (CALIC) [171] is used. The coding scheme used here is shown in Fig. 5.5. At the beginning stage(s), this method gives a good quality image with a low bit-rate compared to other spatial domain methods. The performance of LPBTC-PF is comparable with SPIHT and JPEG2000.

5.3.1 CALIC

CALIC [171] is a sequential coding scheme which operates in two modes: binary mode and continuous-tone mode. Selection of the mode depends on the context of

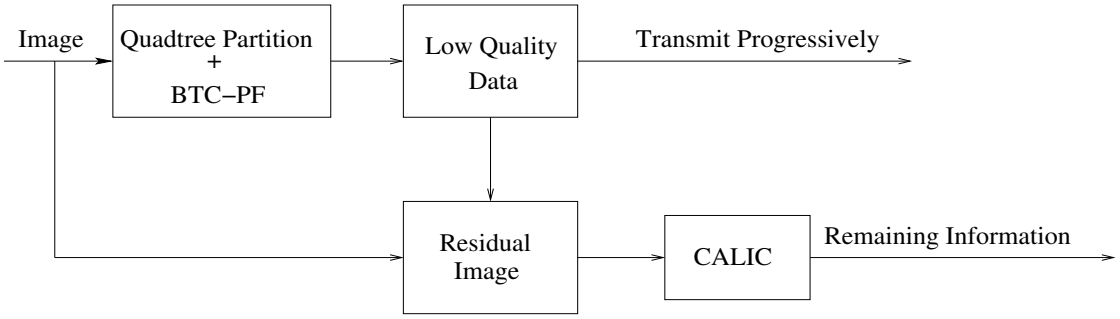


Figure 5.5: The proposed lossless encoder.

the current pixel and the selection process is completely transparent to the user. The continuous-tone mode employs prediction, context determination, context modeling of the prediction error and entropy coding of prediction error. The gradient adjusted predictor (GAP) is used for prediction. In this prediction method, gradients along horizontal and vertical directions are calculated. These values are used to detect the magnitude and orientation of edges and necessary adjustment is made to get improved performance in the presence of local edges. The predicted value is further adjusted via an error feedback loop. The performance of GAP can be improved through context modeling. The selected context can incorporate structures like texture to improve the coding efficiency. In CALIC, the sign-flipped predictor error is entropy coded using eight estimated conditional probabilities in eight different contexts.

CALIC operates in binary mode, for instances, where local data has no more than two intensity values, black and white. Binary mode of CALIC uses context-based adaptive ternary entropy coder to encode values (black and white) and escape symbols.

5.3.2 Phase-I: Quadtree Partition and BTC-PF Coding

In this phase, first an image is divided into a number of non-overlapping blocks of size 16×16. Based on smoothness (defined on variance) of the blocks, each 16×16 block

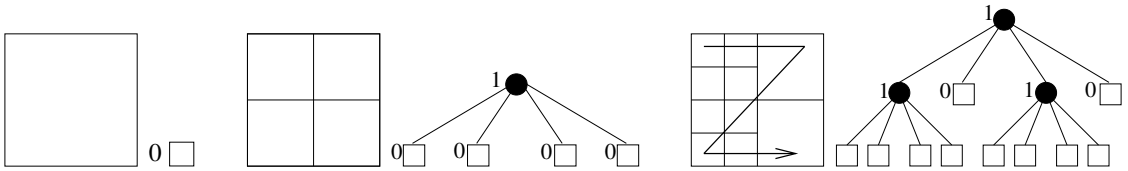


Figure 5.6: Examples of different quadtree partition of a 16×16 block: (a) 'partition-string' is '0', (b) 'partition-string' is '10000', (c) 'partition-string' is '11010'.

is then decomposed as a quadtree till block size reach at 4×4. Some examples of quadtree partition of 16×16 blocks and the corresponding encoding strings are shown in Fig. 5.6. Bit '0' is used to represent a smooth block (applying thresholding on the block variance) and '1' is used to represent a non-smooth block. These quadtrees can have at most three levels. The partition of a block is represented by 'partition-string' which is either a single bit string '0' or a 5-bit string (started with 1). Fig. 5.6 shows that the size of a leaf block can be of size 16×16, 8×8 or 4×4. From now on, we use the term block to denote the leaf block. The blocks of sizes 16×16 or 8×8 are encoded by the corresponding block mean and blocks of size 4×4 are encoded by BTC-PF method. Thus, a 4×4 block is represented by (A, d, I) [see section 5.2]. Since BTC-PF is a lossy compression method, to build a lossless transmission system using BTC-PF we need to transmit the residual image $f_r = f - f_c$, where f is the original image and f_c is the reconstructed one at the end of this phase.

5.3.3 Phase-II: Residual Image Coding

In this phase, the residual image obtained from phase-I (see Fig. 5.5) is encoded to achieve the lossless coding. The residual image intensity has a low variance compared to the original image. It is well known that the entropy coding is more appropriate when entropy of the signal is low or the distribution (curve) of the signal is highly

peaked (i.e., variance is low). Thus it is appropriate to encode residual images [119]. For example, the average standard deviation and entropy of the original images used in our experiment are 45.624 and 7.190, respectively, whereas those of the residual images are 6.260 and 4.520, respectively. For better performance, in our experiment we have used Context-Based Adaptive Lossless Image Coding (CALIC) [171] to encode residual images. The compressed residual data (f_r) is transmitted to the receiver end and added with f_c which gives the original image f.

5.3.4 Progressive transmission

In spatial domain based PIT methods, like BPM, IBPM, GBN etc., to draw the initial sketch of an image it requires at least 1 bit per pixel. The proposed method is a low bit-rate scheme and consists of two phases. The data encoded in phase-I consists of the string to represents the quadtree decomposition of blocks, graylevel and index I (if any). As described earlier, a block can be of size 16×16, 8×8 or 4×4. Only the block mean is used to represent the blocks of sizes 16×16 and 8×8. A 4×4 block passes through BTC-PF and the triplet $(A,\ d,\ I)$ is used to represent the block. In PIT, at first, the string representing the decomposition of all the blocks is transmitted. Then other information (e.g., graylevels, index) is sent progressively. In the BTC-PF method the value of A corresponding to any 4×4 block is very close to its block mean. For simplicity, we denote the mean of the blocks of sizes 16×16 and 8×8 also by A. Hence, for a block either A or $(A,\ d,\ I)$ has to be transmitted. All the A values of an image is transmitted following the step 1 of **Palgo**. The transmission of bit-stream up to step 1.a of **Palgo** including the bit-stream which represents the decomposition of the blocks is considered as the **stage 1** of the LPBTC-PF. Similarly, steps 1.b, 3.a, 3.b, 4 and 5 of **Palgo** are considered as the **stages 2, 3, 4, 5** and **6** of LPBTC-PF respectively. At the end of stage 6, the reconstructed image obtained is of lower quality than the original. So we need to transmit the residual image $f_r{=}f - f_c$

to make the system lossless, where f is the original image and f_c is the compressed image obtained at the end of stage 6. In the phase-II, the f_r is compressed using CALIC algorithm and is transmitted to receiver. This step is considered as the **stage 7** of the proposed method.

5.3.5 Performance of the LPBTC-PF method

In our experiment, a number of 8-bit grayscale images (shown in Fig. 1.4) are used. The performances are evaluated in terms of bpp (Eq. (1.1)) and PSNR (Eq. (1.4)). The stage-by-stage performances of the proposed method (LPBTC-PF) on the test images are reported in Table 5.1 in terms of PSNR and cumulative bit-rate (Cbpp). The average performance of the phase-I of the proposed method is compared with some spatial domain based PIT methods, which are reported in Table 5.2. Proposed method is compared with the bit-plane method (BPM) [155], the progressive BTC (PBTC) [17], the guessing by neighbor method (GBN) [21] and the method based on quadtree segmentation (QuadSegment) [67].

Results in Table 5.2 shows that phase-I of LPBTC-PF method maintains reasonably good quality (PSNR) with lower bit-rate. And bpp is always better than other spatial domain methods. After phase-I (i.e., lossy compression) of our scheme we obtain PSNR=31.326 dB with 0.677 bpp where as that of QuadSegment method [67] are 31.835 dB and 0.912 bpp, respectively. The decoding module of the proposed scheme is computationally less expensive with respect to transform domain methods. The average performance of our method is also compared with sub-band coding methods like SPIHT and JPEG2000. The comparative results are given in Table 5.3. The stage-by-stage results of sub-band methods in Table 5.3 are taken from [67]. Though bit-rate of the proposed method is little higher than the sub-band methods, the reconstruction time of the proposed method is negligible as the proposed method

Table 5.1: The stage-by-stage performance of the proposed PIT method in terms of PSNR and cumulative bit-rate (Cbpp).

Images		Stages						
		Phase-I						Phase-II
		1	2	3	4	5	6	7
Airplane	PSNR	22.989	24.316	24.801	26.116	29.200	30.757	∞
	Cbpp	0.131	0.249	0.404	0.481	0.532	0.583	4.100
Boat	PSNR	23.331	24.653	25.217	26.648	28.990	30.208	∞
	Cbpp	0.156	0.298	0.491	0.587	0.651	0.715	4.659
Couple	PSNR	24.440	26.252	26.852	27.803	30.496	31.452	∞
	Cbpp	0.191	0.343	0.547	0.649	0.717	0.785	4.779
Lena	PSNR	24.817	26.629	27.227	28.687	29.715	31.279	∞
	Cbpp	0.133	0.252	0.404	0.480	0.530	0.580	4.432
Lake	PSNR	22.491	23.502	24.019	25.335	27.165	28.692	∞
	Cbpp	0.171	0.327	0.539	0.645	0.715	0.785	5.371
Man	PSNR	23.822	25.136	25.686	26.853	28.363	30.056	∞
	Cbpp	0.186	0.355	0.587	0.703	0.780	0.857	4.986
Peppers	PSNR	23.758	25.323	25.787	26.952	30.089	31.265	∞
	Cbpp	0.141	0.267	0.426	0.505	0.558	0.611	4.845
Zelda	PSNR	26.380	29.679	30.372	31.290	32.311	32.902	∞
	Cbpp	0.132	0.250	0.390	0.460	0.506	0.552	4.225
Tiffany	PSNR	25.657	28.470	28.964	29.412	31.503	32.721	∞
	Cbpp	0.118	0.224	0.352	0.416	0.458	0.500	4.218
Goldhill	PSNR	24.437	26.131	26.730	27.758	29.655	30.865	∞
	Cbpp	0.192	0.370	0.616	0.739	0.821	0.903	5.274
Splash	PSNR	25.375	27.742	28.171	29.327	33.228	34.016	∞
	Cbpp	0.089	0.168	0.256	0.300	0.329	0.358	3.715
Baby	PSNR	23.950	25.554	26.185	27.568	30.040	31.702	∞
	Cbpp	0.194	0.371	0.616	0.738	0.819	0.900	4.826
Average	PSNR	24.287	26.116	26.668	27.812	30.063	31.326	∞
	Cbpp	0.153	0.290	0.469	0.559	0.618	0.677	4.619

requires only a few shifting, addition and table look-up operations (see chapters 3 and 4).

The multi-segment coding method [119] which also uses the same concept, i.e., lossy coding followed by lossless coding of the residual, reports that overall bit-rate for image 'Lena' are 4.700 bpp and 4.870 bpp depending on the segmentation and tiling methods. In case of the proposed lossless coding method it is 4.432 bpp. The average bit-rate of the LPBTC-PF method is 4.619 bpp (see Table 5.1) and that value of the CALIC[1], SPIHT[2] and JPEG2000[3] methods is 4.215 bpp, 4.174 bpp and 4.262, respectively. The experimental result shows that our method is highly applicable for low bit-rate applications compared to other spatial domain methods. This method can also be used in the applications requiring lossless images (like medical imaging, remote sensing). As the overall bit-rate and also initial bit-rates are lower compared to others with good qualities, the proposed method is suitable for multimedia browsing as stated in the objective. The visual quality of the output images produced at different stages of the proposed method on 'Lena' is shown in Fig. 5.7.

[1]the binary code of CALIC which we have used is available at [170].

[2]we have used the SPIHT code available at [134].

[3]we have used the Kakadu [146] system for JPEG2000 coding.

Table 5.2: Average comparative results of various spatial domain PIT methods.

Methods		Stages					
		1	2	3	4	5	6
BPM	PSNR	17.491	23.165	29.123	34.727	40.711	46.372
	Cbpp	1	2	3	4	5	6
PBTC-16	PSNR	28.391	34.420	40.206	46.181	52.756	61.066
	Cbpp	1.0625	2.1246	3.2336	4.324	5.2846	6.0259
GBN[4]	PSNR	21.04	22.78	31.08	34.83	38.88	46.38
	Cbpp	1.00	1.65	2.65	3.43	4.43	5.41
QuadSegment[5]	PSNR	25.198	27.177	28.420	29.605	31.017	31.835
	Cbpp	0.145	0.219	0.309	0.436	0.683	0.912
PBTC-PF	PSNR	24.287	26.116	26.668	27.812	30.063	31.326
	Cbpp	0.153	0.290	0.469	0.559	0.618	0.677

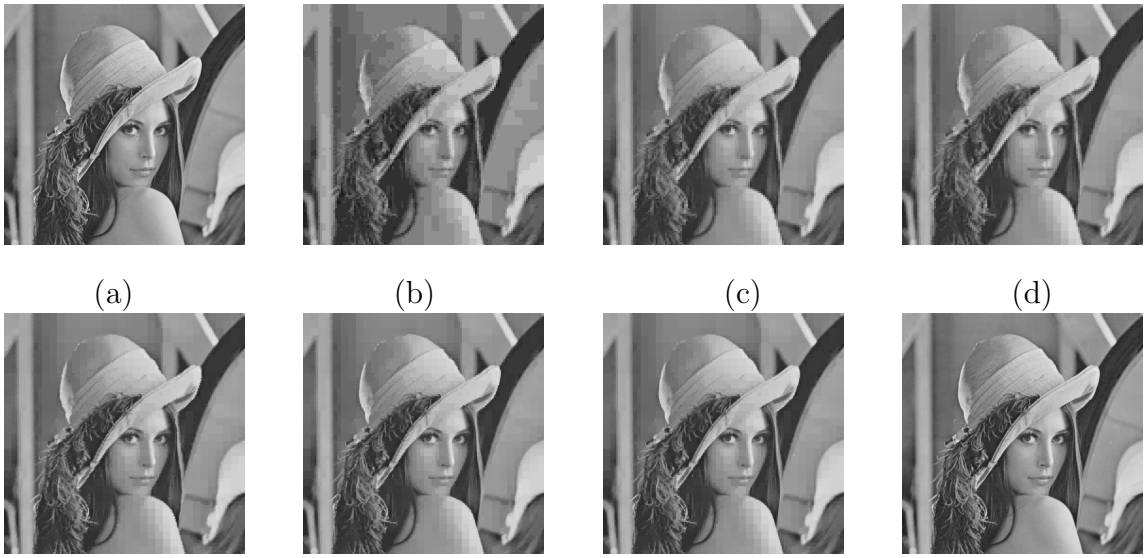
[4]Results are taken from [21]

[5]Theses results are from [67] and Cbpp is without considering relative addressing technique.

Table 5.3: Comparisons of proposed method with sub-band coding methods.

Coding Method		Stages								
		1	2	3	4	5	6	7	8	9
SPIHT[5]	PSNR	21.723	23.683	25.998	28.520	31.480	34.867	38.718	43.028	47.539
	Cbpp	0.086	0.116	0.158	0.239	0.410	0.731	1.281	2.158	3.333
JPEG2000[5]	PSNR	28.674	31.672	33.814	35.644	36.985	38.194	38.869	39.100	
	Cbpp	0.052	0.114	0.188	0.284	0.383	0.492	0.567	0.615	
LPBTC-PF	PSNR	24.287	26.116	26.668	27.812	30.063	31.326	∞		
	Cbpp	0.153	0.290	0.469	0.559	0.618	0.677	4.619		

[5]Theses results are from [67] and Cbpp is without considering relative addressing technique.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 5.7: (a) Original Image, (b)-(g) Reconstructed images in Phase I: (b) Stage 1: PSNR= 24.817 dB, Cbpp = 0.133, (c) Stage 2: PSNR= 26.629 dB, Cbpp = 0.252, (d) Stage 3: PSNR= 27.227 dB, Cbpp = 0.404, (e) Stage 4: PSNR= 28.687 dB, Cbpp = 0.480, (f) Stage 5: PSNR= 29.715 dB, Cbpp = 0.530, (g) Stage 6: PSNR= 31.279 dB, Cbpp = 0.580, (h) Reconstructed images in Phase II: PSNR=∞, Cbpp = 4.432.

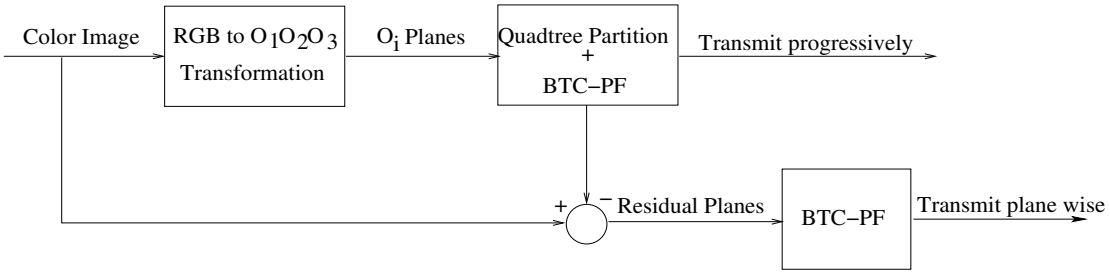


Figure 5.8: A block diagram of the proposed color image coding scheme.

5.4 Color image progressive transmission

In section 5.2, the fundamentals of the PBTC-PF method has been discussed for grayscale images. In this section, the concept is extended for color images. In this method, first of all the color image (RGB) is transformed into $O_1O_2O_3$ domain using a reversible transformation. The details of this transformation can be seen in section 4.2. The proposed BTC-PF based color PIT method (CPBTC-PF) also consists of two phases. In the first phase, the image planes O_i are coded in lossy manner and are transmitted progressively in an inter-leaved format. In the second phase, the residual planes ($\hat{O}_i$) resulted from O_i and corresponding reconstructed planes are coded and transmitted. A blcok diagram of the proposed CPBTC-PF method is shown in Fig. 5.8.

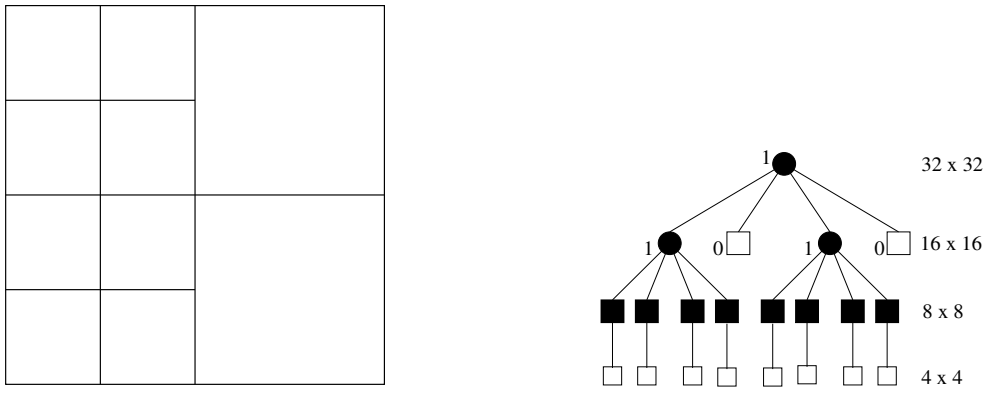
5.4.1 Phase-I coding

First, the color images (RGB) are transformed into $O_1O_2O_3$ domain and then these color planes are coded separately. In chapter 4, it has been mentioned that O_1 is the luminance component of the image and remaining two represents the chrominance of the image. Here, all the image planes are hierarchically partitioned and then coded accordingly. The size of the blocks depends on the visual sensitivity of the image

planes and partition conditions are also changed. The O_1 plane is encoded by the concept similar to that used to encode the grayscale image, explained in section 5.3.2. Hence, to represent a block of O_1 plane, either A or (A, d, I) is required. The coding technique of $O_2(O_3)$ is as follows.

Coding Scheme: O_2 and O_3 plane

Both O_2 and O_3 planes represent the chrominance components and contain less information compared to O_1 plane. Comparatively, lesser amount of bits are spent for $O_2(O_3)$ plane. The coding scheme of O_2 and O_3 planes are similar to that of O_1 plane. Like O_1 plane, these two planes are also first divided into blocks of size 32×32. Then each of them is gradually decomposed until 8×8 blocks are reached. All blocks of different sizes are initially coded only by block mean (A). To achieve the better quality, from each 8×8 block one 4×4 block is generated using averaging over 2×2 window. An example of quadtree partition of 32×32 block and the corresponding 'partition-string' is shown in Fig. 5.9. Then these 4×4 blocks are encoded by BTC-PF, which returns the quantization levels μ_1 and μ_2 ($> \mu_1$), and index I. An 8×8 block is reconstructed from (μ_1, μ_2, I), where a 2×2 sub-block is filled according to the current pattern. It is obvious that μ_2 is greater than the mean of 8×8 block and μ_1 is less than that. In PIT, for a 32×32 or 16×16 block, the A value is transmitted. For a 8×8 block, first transmit A of the block and then transmit $(A - \mu_1 (= \delta_1), \mu_2 - A (= \delta_2), I)$. When only A of 8×8 block is available, then the block is reconstructed using A. Next, when (δ_1, δ_2, I) becomes available, then the block is refined using (μ_1, μ_2, I).



(a) (b)

Figure 5.9: Example of the quadtree partition: (a) 32×32 block (B_{32}), (b) corresponding quadtree, 2×2 sub-sampling at leaf nodes. The 'partition-string' is 11010.

5.4.2 Phase-II coding

The second phase is used to achieve even better quality. In this phase, the residual planes $\hat{O}_i$ are obtained from O_i and their corresponding reconstructed planes. All three residual planes are encoded by BTC-PF method considering the eight patterns at level 2 (assuming root at level 1) of the FSPTT (see Fig. 5.2) as the patternbook.

5.4.3 Progressive transmission

The position of the blocks of different sizes is transmitted by the bit-streams representing the quadtree partition. Then the block mean (A) of 16×16 and 8×8 blocks of O_1 plane, and the same for 32×32, 16×16 and 8×8 blocks of O_2 and O_3 planes are transmitted along with A value of 4×4 blocks of O_1 plane. The transmission of bit-stream up to step 1.a (for all A values) of **Palgo**, including the bit-stream for the decompositions of the planes, is the **stage 1** of the proposed method (CPBTC-PF). Similarly, steps 1.b, 2.a, 2.b, 3, 4 of **Palgo** are assumed as the **stages 2, 3, 4, 5 and**

6, respectively (**stage 2 - stage 6**, to transmit the information of O_1 plane). Then the bit-streams consisting of δ_1, δ_2, and seven bits representing the selected patterns (as FSPTT given in Fig. 5.2 is used) for O_2 and O_3 planes are transmitted. This is considered as the **stage 7** of the CPBTC-PF method. The encoded information resulted from the residual planes $\hat{O}_1$, $\hat{O}_2$ and $\hat{O}_3$ are transmitted in order. These steps are considered as **stages 8, 9 and 10** of the CPBTC-PF method.

5.4.4 Performance of CPBTC-PF method

In our experiment, number of color images, shown in Fig. 1.5, are used. The performance of the proposed method is evaluated in terms of bit-rate (cumulative bit per pixel (Cbpp)) and PSNR. The visual quality of the proposed method at different stages are shown in Fig. 5.10. The stage by stage performance of the CPBTC-PF method is reported in Table 5.4. In phase-I (up to stage 7) all bit-rates are calculated straight way on raw data, but in phase-II (stages 7 -9) entropy coding is used to encode the informations.

Average results of different spatial domain PIT methods and the CPBTC-PF method are compared in Table 5.5. CPBTC-PF method is compared with CBPM, CPBTC, CBMCPBTC and GBN methods. Table 5.5 reveals that the average quality of reconstructed images just after the very first step of the proposed method is good enough to infer about the overall content of the image consuming least number of bits compared to other methods. At the first stage bit-rate of CBPM, CPBTC and GBN methods is greater or equal to 3 bpp and quality of reconstructed image are 18.106 dB, 27.828 dB, and 20.83 dB, respectively. For CBMCPBTC method those values are 1.19 bpp and 24.99 dB, but the CPBTC-PF method gives quality 22.131 dB with a bit-rate of 0.161 bpp only. Hence, after the initial sketch of the image, if it is found that the image being transmitted is not the desired one then the transmission can

Figure 5.10: Reconstructed images by CPBTC-PF method at different stages: (a) Stage 1: PSNR=22.597 dB, Cbpp=0.159, (b) Stage 2: PSNR=25.774 dB, Cbpp =0.301, (c) Stage 3: PSNR=26.208 dB, Cbpp=0.446, (d) Stage 4: PSNR=27.122 dB, Cbpp=0.518, (e) Stage 5: PSNR=27.745 dB, Cbpp=0.566, (f) Stage 6: PSNR=28.592 dB, Cbpp=0.614, (g) Stage 7: PSNR=28.978 dB, Cbpp=0.676, (h) Stage 8: PSNR=30.302 dB, Cbpp=1.029, (i) Stage 9: PSNR=31.128 dB, Cbpp=1.339, (j) Stage 10: PSNR=32.185 dB, Cbpp=1.593.

be terminated. Thus, CPBTC-PF method saves maximum time and bandwidth with respect to other methods. The table also shows that both PSNR and Cbpp of other methods increase in large steps, whereas in CPBTC-PF method these increments are in small steps. It may be noted that to terminate the transmission at a point the currently ongoing stage is to be completed. Low Cbpp denotes that the volume of transmission in intermediate stages is less for CPBTC-PF. Thus, the waiting time is less for the CPBTC-PF method compared to the other methods.

Table 5.4: The stage by stage performance of CPBTC-PF method in terms of PSNR and cumulative bit-rate (Cbpp), $*$ indicates the results with entropy coding.

Images		Stages									
		1	2	3	4	5	6	7	8*	9*	10*
Airplane	PSNR	21.314	23.865	24.246	25.174	27.217	27.935	28.798	29.962	30.389	31.384
	Cbpp	0.164	0.311	0.460	0.534	0.583	0.632	0.720	1.054	1.279	1.521
Lena	PSNR	22.597	25.774	26.208	27.122	27.745	28.592	28.978	30.302	31.128	32.185
	Cbpp	0.159	0.301	0.446	0.518	0.566	0.614	0.676	1.029	1.339	1.593
Peppers	PSNR	21.538	24.009	24.308	24.986	25.913	26.223	27.807	28.567	29.066	29.854
	Cbpp	0.200	0.379	0.531	0.607	0.657	0.707	0.886	1.240	1.539	1.801
Splash	PSNR	22.521	25.671	25.933	26.584	27.973	28.206	30.025	30.906	31.707	32.647
	Cbpp	0.126	0.238	0.318	0.358	0.384	0.410	0.527	0.822	1.061	1.298
Tiffany	PSNR	22.285	25.532	25.756	26.042	27.567	27.894	28.650	29.138	29.688	30.818
	Cbpp	0.139	0.261	0.370	0.424	0.460	0.496	0.568	0.933	1.232	1.544
Couple	PSNR	22.815	25.759	26.212	26.864	28.574	29.220	29.536	30.827	31.609	32.359
	Cbpp	0.167	0.316	0.484	0.568	0.624	0.680	0.719	1.090	1.379	1.600
House	PSNR	21.637	23.623	23.957	25.002	26.624	27.093	28.186	29.071	29.518	30.684
	Cbpp	0.153	0.289	0.414	0.476	0.517	0.558	0.650	0.982	1.252	1.520
Zelda	PSNR	22.316	25.508	25.945	26.896	27.528	28.243	28.881	30.229	30.910	31.547
	Cbpp	0.188	0.357	0.538	0.628	0.688	0.748	0.832	1.207	1.498	1.733
Girl	PSNR	22.128	27.197	27.623	27.921	30.892	31.551	31.639	32.568	33.618	34.523
	Cbpp	0.104	0.196	0.295	0.344	0.377	0.410	0.416	0.646	0.866	1.051
Lab	PSNR	22.163	26.474	27.117	28.766	29.567	29.894	30.231	32.527	34.088	34.822
	Cbpp	0.177	0.336	0.525	0.619	0.682	0.745	0.767	1.116	1.446	1.645
Average	PSNR	22.131	25.341	25.731	26.536	27.960	28.485	29.273	30.410	31.172	32.082
	Cbpp	0.161	0.298	0.438	0.508	0.554	0.600	0.676	1.012	1.289	1.531

Table 5.5: Comparative results of proposed method with some other methods.

Methods		Stages									
		1	2	3	4	5	6	7	8	9	10
CBPM	PSNR	18.106	22.387	28.466	34.475	40.490	46.277	51.253			
	Cbpp	3	6	9	12	15	18	21			
CPBTC-16	PSNR	27.828	33.929	39.737	45.729	52.328	60.569				
	Cbpp	3.187	6.296	9.468	12.577	15.307	17.440				
CBMCPBTC	PSNR	24.99	28.34	30.67	32.04	33.01	34.32				
	Cbpp	1.19	2.38	3.76	5.51	8.00	11.97				
GBN	PSNR	20.83	22.40	31.38	34.46	39.71	46.40				
	Cbpp	3.00	4.82	7.82	10.03	13.03	15.79				
CPBTC-PF	PSNR	22.131	25.341	25.731	26.536	27.960	28.485	29.273	30.410	31.172	32.082
	Cbpp	0.161	0.298	0.438	0.508	0.554	0.600	0.676	1.012	1.289	1.531

5.5 Conclusions

In this chapter, BTC-PF method is successfully employed for progressive image transmission. BTC-PF method is tuned to devise the desired method which is a hybrid of the resolution progressive and the SNR progressive transmission scheme. At the very beginning of the transmission, the image is reconstructed by considering each 4×4 block as a smooth block. The quality of the reconstructed image is refined in successive steps both in terms of intensity and resolution. The selection of the grayvalues 'A' and 'd' helps in progressive transmission. Organizing the patternbook like FSPTT also facilitates the progressive transmission. First, a grayscale image transmission method is presented and then it is extended to color image transmission and lossless transmission methods. For lossless transmission, a lossless coding method CALIC is used as the second phase of the coding. Experimental result reveals that the proposed lossless transmission, that is PIT followed by CALIC, requires slightly more bits than that if CALIC transmits the original image in conventional way. For example, the proposed method requires about 9.5% bits extra compared to CALIC,

if the entire image is transmitted up to the end. However, if the image transmission is terminated immediately, as it happens in case of image search and download where multiple image transmission is involved, the overall gain of the proposed method is significant over CALIC.

The color images are first transformed to $O_1O_2O_3$ domain by a lossless transformation. In the actual implementation, the image is divided into a number of blocks and each block is decomposed like a quadtree based on the smoothness criterion. This decomposition process results in variable size block coding and helps to achieve low bit-rate with an acceptable quality at the initial stage. The experimental results shows that the proposed method gives better results than other spatial domain based methods. The lossless color image progressive transmission method is similar to that of gray scale image as describe in section 5.3.4. That means, lossless transmission can be achieved through PIT followed by the transmission of residual O_1, O_2 and O_3 planes by CALIC. The performance of the proposed method as well as its comparison with SPIHT and JPEG2000 are proportional to that of garyscale case, so are not repeated here. The performance of lossless coding is also comparable with that of the coding schemes like JPEG2000, SPIHT, and CALIC. So far, we have proposed the methodology to work with the still images (both, grayscale and color images) which has been further extended to cater progressive transmission. The schemes can also be deployed for video coding. In video coding method, the I-frame and the residual frames are encoded by still image compression. Thus, proposed techniques can also be extended to do so. Moreover, in video, the concept of motion comes into the picture. Hence, a motion estimation scheme becomes important to generate the residual frames. In the next chapter, all these issues related to the video compression are presented.

Chapter 6

Video Compression using BTC-PF

6.1 Introduction

A vast area of information exchange/handling applications involves the transmission and storage of video, which demands for higher efficiency in video coding. Raw video is a sequence of image frames. In television application, the size of each frame is 720×480 and the sequences have 30 frames per second. For 24 bits images, the required transmission rate is 248 Mbps [13]. The limited bandwidth of networks and storage media insist a clear need of effective video compression. Video signals contain redundant information and this redundancy may be in various forms like spatial redundancy (i.e., correlations between the pixels in the same frame), temporal redundancy (i.e., correlation between video frames) and code redundancy (i.e., statistical redundancy that is present in the bit-stream). Because of these redundancies, it is possible to compress the video signal for transmission at a manageable data rate without introducing significant degradation. Video compression standards MPEG-x or ITU-H.26x [59, 75, 76, 141, 74, 9] have several mechanisms to exploit this

redundant information. Video encoding system includes different modules to remove intra-frame and inter-frame redundancies, and to encode the difference frame. The Fig. 6.1 represents the standard video encoder/decoder. The most important feature of any video coding technique is the predictive coding, and thus only a residual frame needs to be transmitted. If the pixels are predicted from the other pixels of the same frame, the spatial redundancy is reduced and it is called intra-frame prediction. As an alternative, pixels can be predicted from the pixels of the other frames of the video sequence. It is called inter-frame prediction; this reduced the temporal redundancy. Inter-frame prediction is also known as motion compensation and block matching algorithm (BMA) is widely used for this purpose. Residual frame coding is also an important module in video coding methods. The residual frame coding of standard video methods (MPEG-x, H.26x) includes transform based coding technique (DCT), quantization and entropy coding. For real-time applications, like video conferencing, both the encoding and decoding complexity are important. However, in the applications like video playback, the compression is done once but it is decoded several times. In such situation, we may consider the encoding as an off-line process, but the decoding is on-line. Hence, decoding should be fast. In fact, it should be fast enough to accommodate higher resolution frames, if required. In this chapter we present a video coding method that needs negligible time for decoding. For intra-frame coding exploiting spatial redundancy, a BTC-PF based method is used and section 6.2 is focused on that. To reduce the time complexity of the encoder a fast BMA algorithm is proposed in section 6.3. The major time consuming module of the standard video decoding methods is the inverse transformation. A block diagram of the standard video coding/decoding method is shown in Fig. 6.1. In chapter 4, it is shown that the decoding complexity of BTC-PF method is negligible compared to DCT based compression method. With an aim towards fast decoding, a hierarchical block truncation coding using pattern fitting (HBTC-PF), a modified version of BTC-PF to encode the residual frames is proposed in section 6.4. The complete video coding method is

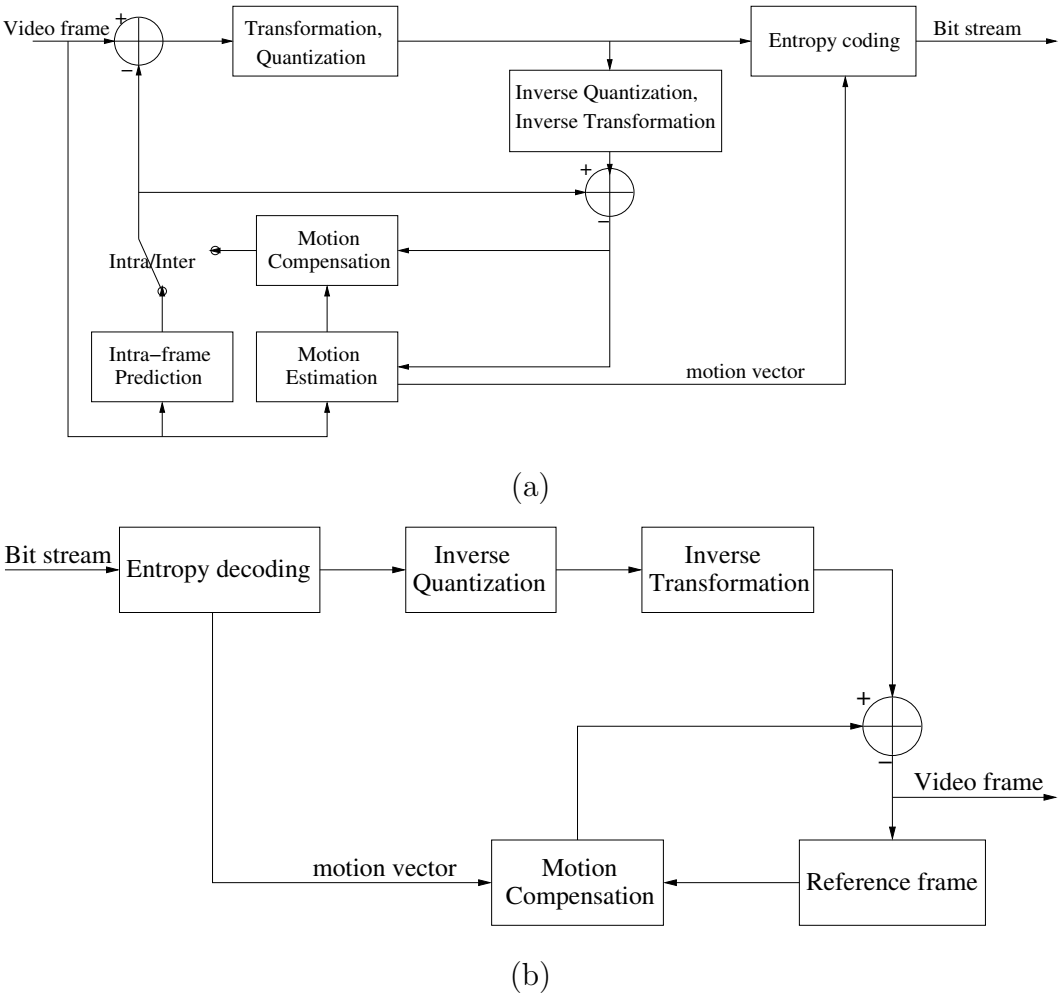


Figure 6.1: Standard video coding model: (a) encoder, (b) decoder.

presented in section 6.5. Section 6.6 discusses the performance of the proposed video coding method. Finally, it is concluded in section 6.7.

6.2 Intra-frame prediction

In each frame of a video sequence, there is a spatial correlation. It is possible to obtain a reasonable estimate of the values of pixels based on a set of neighboring pixels. For this purpose, any well-accepted image compression method can be used to exploit

the spatial redundancy. For example, 'motion JPEG' [168] may be used to encode the frames of a video sequence. The frames which undergo intra-frame prediction are denoted by I-frame. In I-frame prediction, the image (frame) is partitioned into a number of 4×4 blocks. To exploit the spatial redundancy, an image block B is first estimated as $\hat{B}$ using information from already reconstructed neighboring blocks $\tilde{B_L}$ and $\tilde{B_A}$ [see Fig. 4.4] as done by [37]:

$$\hat{B}(r,c) = \frac{v}{v+h} val_h + \frac{h}{v+h} val_v \tag{6.1}$$

where v (respectively, h) is the distance between (r,c) and '*' on the same column (respectively, row), and val_v (val_h) are the values at the corresponding vertical (horizontal) '*'-marked position, respectively, as shown in Fig. 4.4. Implementation of Eq. (6.1) normally requires 32 multiplications and 16 additions per 4×4 block. However, using table look-up method [37], the number of computation can be reduced to 16 additions. But, it requires a huge additional space. The estimated value of a pixel (r,c) depends only on val_v and val_h. It would be a better estimation if we can accommodate more pixels of $\tilde{B_L}$ and $\tilde{B_A}$, and in our implementation we have tried to use the values of the last row of $\tilde{B_A}$ and last column of $\tilde{B_L}$. Thus, we adopt a simplified recursive estimation scheme as:

$$\hat{B}(r,c) = \frac{\hat{B}(r-1,c) + \hat{B}(r,c-1)}{2} \tag{6.2}$$

where we also treat $\tilde{B}$ blocks as $\hat{B}$ for prediction. To implement Eq. (6.2) only 16 additions and 16 bit-shift operations per block are required without any requirement of additional space. The residual block $B_e(r,c) = B(r,c) - \hat{B}(r,c)$ is coded by the BTC-PF method, where the quantization levels are the mean values of the corresponding partitions of the block defined by the selected pattern. Since the residual block $B_e(r,c)$ contains a significant number of zero values and some positive and negative values, to encode the residual blocks we have chosen $Q = 3$. The details of encoding of the residual block by BTC-PF method is given in section 4.3.1.

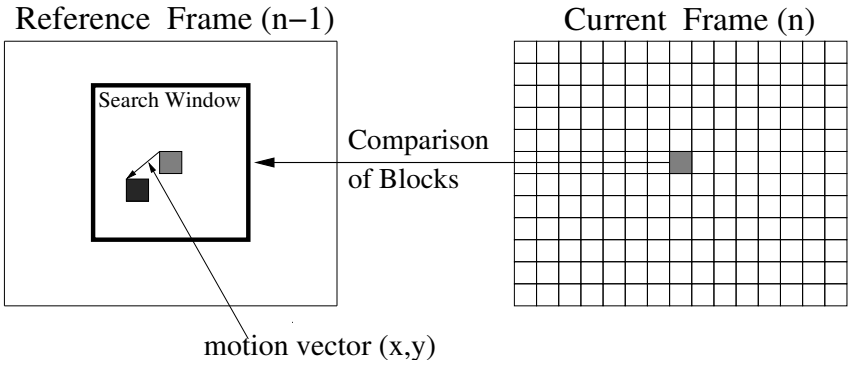


Figure 6.2: Block matching concept.

6.3 Motion estimation

The inter-frame prediction method intends to de-correlate the pixels, by removing the temporal redundancy and is based on the estimation of motion between successive frames. Video coding standards estimate motion by matching block of pixels between frames, which offers a good compromise between temporal redundancy reduction and computational cost. In BMA, current frame is divided into $N \times N$ (N=16) block of pixels, commonly referred to as macroblock. The objective of the motion estimation (ME) process is to find a best match for each macroblock in the reference frame. The exhaustive search over the entire reference frame gives the optimum match; however, this is impractical because of time complexity. Instead, the search is restricted to $[-p, p] \times [-p, p]$ search window (SW) around the location of the current block. A macroblock is referred to as (k, l), the top-left corner of the block. If $(k + x, l + y)$ be the location of the best match in reference frame then (x, y) denotes the motion vector (MV). Fig. 6.2 represents the block matching concept. In BMA, there are many choices for block distortion measure (BDM), such as mean-square-error (MSE), sum of absolute difference (SAD), mean of absolute difference (MAD), etc. Among these criterion, SAD is the most popular one because it requires less computation and produces similar performance as the MSE [122]. Let the intensity of the ij^{th} pixel of

the current frame (n^{th} frame) is represented by $f_n(i,j)$, then the SAD between the block (k,l) of the current frame and the block $(k+x, l+y)$ of the previous frame is given by

$$SAD_{(k,l)}(x,y) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |f_n(k+i, l+j) - f_{n-1}(k+x+i, l+y+j)| \tag{6.3}$$

The motion vector $mv(k,l)$ of the block (k,l) is given by

$$mv(k,l) = arg(\min_{(x,y)\in SW} \{SAD_{(k,l)}(x,y)\}) \tag{6.4}$$

The full search (FS) algorithm is the simplest block matching algorithm, which searches all possible points in SW. The advantage of FS is that it can find global optimum solution, but the computational cost for this algorithm is very high. The total complexity of ME process depends on (i) the number of search points, (ii) the number of pixels of the block considered to compute BDM, and (iii) the cost of BDM. Accordingly, the fast block matching algorithms belong to one of the following groups.

(i) The algorithms that produce the same result as the FS but with fewer calculations called the fast full search [14, 57, 5, 98]. A simple fast full search algorithm stops the distortion calculation at a search point once the 'best-so-far' threshold is reached [160, 97, 49].

(ii) The algorithms that reduce the complexity of BDM achieved so by omitting some pixels of the candidate block. A simple approach is to consider the pixels of alternate rows and columns of the macroblock. A number of different low complexity criterion such as pixel difference classification (PDC) and difference pixel count (DPC) are also reported [13].

(iii) The algorithms that use fewer search points. These are most common and are faster than those of other groups. The number of search points can be predetermined and fixed as it has been done in three-step-search (TSS) [85] or it can vary according to search strategy as in 2-D log search [78], cross-search (CS) [50], four step search (4SS)

[123], the new three step search (NTSS) [95], efficient three-step search (ETSS) [80], block-based gradient descent search (BBGDS) [100], diamond search (DS) [150, 181], hexagonal search (HEXBS) [180], enhance hexagon search (EHEXBS) [179] etc. A hybrid method presented in [35], this method combining TSS and 4SS and gives a good result. The search strategy of these algorithms can be summarized by following steps:

Step 1 Evaluate the block matching error at some positions surrounding the center of the search window (SW).

Step 2 Compare the matching error and find the wining point for next step.

Step 3 Steps 1 and 2 are repeated either for a fixed number of iterations or until the algorithm converges at a minimum (may be a local minimum).

All the above fast search methods uses blocks of fixed size in ME purpose. Variable block size may also be used [71, 83]. All these algorithms assume a unimodal cost surface, i.e., *the block matching error decreases monotonically as the search point moves closer to the global minimum (respect to search window)*, and therefore that local minimum becomes the global minimum. This assumption may not be valid always. However, these algorithms drastically reduce the number of search points. If the initial search center (ISC), the location from where the search process starts be closer to the global optimal point, the time complexity of motion estimation is reduced. There is a strong correlation between the motion vectors of the neighboring blocks (spatial and temporal). So these vectors can be used to predict the ISC. Fig. 6.3 shows the blocks that are normally used for prediction, 'E' is the current block and others are neighboring blocks. In literature, there are several methods for MV prediction. H.263 and H.264 use median prediction, i.e., it takes the median values of corresponding components of MV's from the neighboring blocks as the component of the predicted MV. Mean or weighted mean can also be used to predict the MV [35].

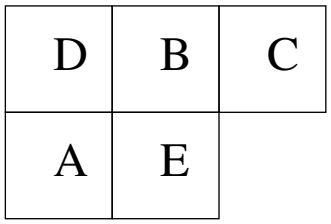


Figure 6.3: The location of current block (E) and its neighbor blocks.

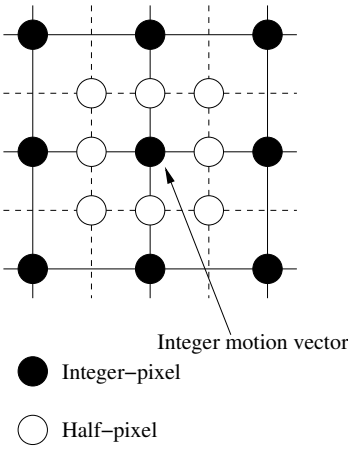
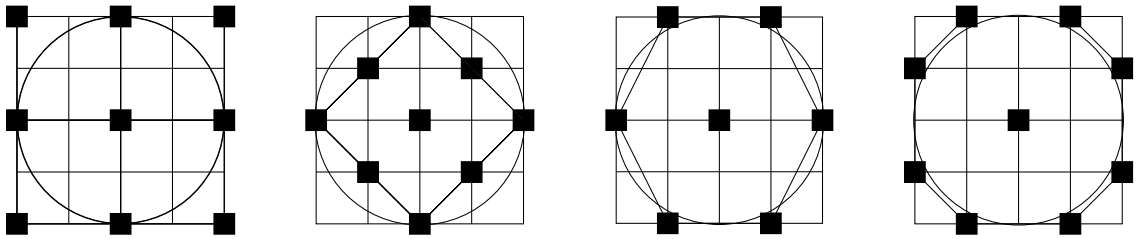


Figure 6.4: Search points for half-pixel motion estimation.

A spatio-temporal prediction concept is proposed in [73, 35]. The hybrid method [35], not only predicts the motion vector but also determines the size of the search area according to the amplitude of the predicted vector and fixed up the search direction. In this discussion, it is assumed that the number of reference frame is one. H.264 [9] successfully uses the multiple reference frames to obtain an MV.

All the above BMAs return integer-pixel motion vector. However, in both natural and synthetic video sequences, the true frame-to-frame displacement of moving objects is rarely an integer number [51]. In order to exploit the temporal redundancy efficiently, ME should be performed over a search area with greater pixel resolution, where MVs may point to blocks placed at half, quarter or one-eighth pixel locations of the search area. Any existing BMA can be used to estimate MVs with sub-pixel accuracy (SPA) at the sub-pixel level. This technique degrades seriously due to the

excessive requirement of storage and computation complexity. These constraints have forced the video standards (H.263, MPEG-2, H.264) to adopt ME with SPA as optional and extended part of the standard. Usually, a two-step search approach is preferred for estimation of MVs with SAP [13]. In the first step ME is performed at integer pixel level. Then in the second step, the search area surrounding the selected integer block is interpolated into a higher resolution and the integer-pixel MV is refined into sub-pixel accuracy. If we consider half-pixel range then in the search area there are eight half-pixels positions (shown in Fig. 6.4). For half-pixel motion estimation (HME), it becomes meaningful to reduce the computational complexity of HME. A fast HME is presented in [91], which examines five search points instead of eight. A paraboloid prediction method [41] examines only three search points and the linear model based prediction method [92] reduces the number of search point to 2.21 on average. In [143], different models are proposed for sub-pixel motion estimation. The method avoids the sub-pixel interpolation and subsequent second step search. Instead, five different models are defined and the parameters are determined using the motion-compensated error of the neighboring pixels of the integer-pixel motion vector. Then the motion-compensated error at sub-pixel level is estimated. In these ME algorithms, previous/next frame in the sequence may be used as reference to predict the motion of current frame. When, the reference frame appears before (previous one) this is forward prediction and the current frame is called predictive frame (P-frame). Another type of inter-prediction is the Bi-prediction (B-frame), which supports forward and backward prediction for inter-prediction. The H.264 allows multiple reference frames, which include previously encoded frame as well as future frame. The residual frame is then compressed using some transform domain (usually DCT or wavelet transform) methods. As we have seen that the decoding time due to BTC-PF method is negligible in comparison to that of the transform domain compression methods. In this chapter, we evaluate the performance of the BTC-PF method in terms of quality and bit-rate for video compression. Here, the BTC-PF



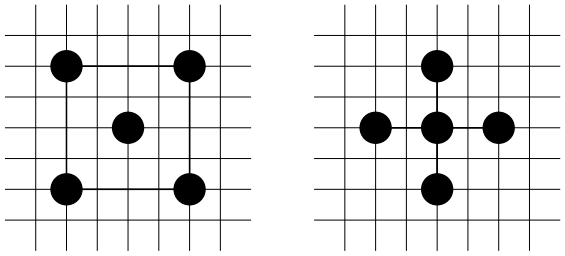
(a) (b) (c) (d)

Figure 6.5: Distribution of search points over circular region: (a) Square grid pattern, (b) Diamond pattern, (c) Hexagonal pattern and (d) Octagonal pattern.

method is employed to encode the I-frame and residual frames. To track the motions in video sequence, we have proposed an octagonal search method. The details of the octagon based search method and BTC-PF method based encoding techniques are presented in the following sections.

6.3.1 Octagonal Search method

The fast search methods use some selected search points to find out the MV and these methods are considered a unimodal error surface. On the basis of this assumption, no search method should have the biasness over the direction and the search length to trap the motion. Hence, the search pattern have to be isotropic (i.e., circular) with respect to current search point. For example, TSS, 4SS use square pattern which is a circle in terms of chess-board distance [132]; DS and similar search algorithms use diamond pattern, which is a circle in terms of city-block distance [132]; and HEXBS and EHEXBS use hexagonal pattern, which is again an approximation of a circle over hexagonal grid. Different search patterns approximating a circular region are shown in Fig. 6.5. From the patterns, it is clear that among the patterns octagon has the closest resemblance with the circular search pattern. An octagonal pattern



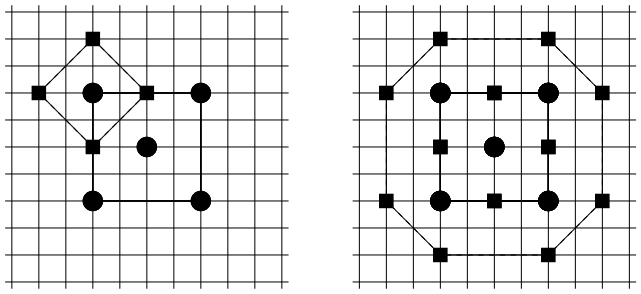
(a) (b)

Figure 6.6: Two basic patterns used in the proposed method with $R = 2$: (a) Square pattern (SQP) (b) Cross pattern (CP).

can be generated by dilating a square pattern by a cross pattern. In this method, two basic patterns: square pattern (SQP) and cross pattern (CP) [Fig. 6.6] are used alternatively in subsequent steps, starting with say, square pattern. Here, the pattern size (R) is defined as: 1) for SQP, R = half of the horizontal (or vertical) distance between any two boundary points. 2) for CP, R = distance between the center point and any boundary point. Let SQP is considered first and then CP is dilated to each boundary point of the SQP. Then in dilated pattern, the boundary points form an octagon with a 3×3 structure at the center. The situation is demonstrated in Fig. 6.7. The octagonal search (OS) method uses three schemes to achieve a good result.

1. **Initial Search Center (ISC) Prediction:** To estimate the motion vector (MV) of the current block either a spatial prediction or a temporal prediction or a hybrid method (combination of two) is used. In the proposed scheme, hybrid prediction is used. To predict the ISC, the motion vectors of the spatio-temporal neighbor blocks are used. The blocks whose motion vectors are used to determine the ISC are shown in Fig. 6.8. From the motion vectors (mv_x^l, mv_y^l), (mv_x^a, mv_y^a), and (mv_x^c, mv_y^c) for which SAD value is minimum is being set as ISC.

2. **Octagonal Search:** The search starts at ISC point with SQP. From these



(a) (b)

Figure 6.7: Octagon Pattern, SQP followed by CP: (a) CP employed to left-top boundary point of SQP, (b) Octagon Pattern with 3×3 grid structure of SPs at center.

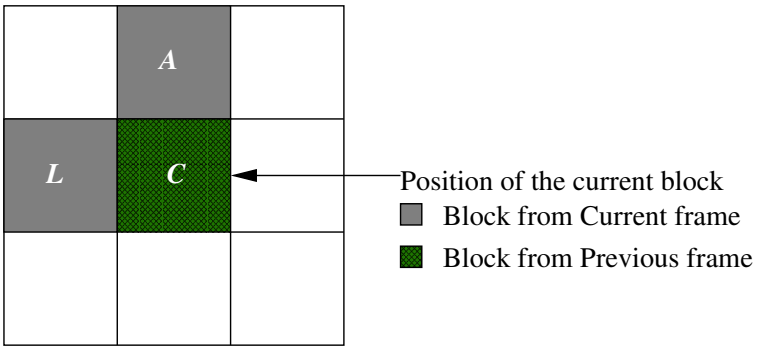


Figure 6.8: The motion vectors (mv_x^l, mv_y^l) and (mv_x^a, mv_y^a) of spatial neighbor blocks 'L' and 'A', and motion vector (mv_x^c, mv_y^c) of temporal (co-located) neighbor block 'C' are used to determine ISC.

search points, the point for which SAD value is minimum is the wining point for the next step of the search process. In the next step, CP is used centering the wining point. Then, the search is continued with SQP and CP alternately in subsequent steps. In Fig. 6.7(a), it has been shown that, in the first step of search procedure the wining point is the top-left point of SQP and CP is employed at that point.

3. **Search steps:** In this method, the size of the initial SQP is two. In any step, if the wining point is the center point of the current pattern, the size of the pattern for next step will be decremented by one. The search will be stopped when the size of the pattern become zero. In this method, number of search steps is not restricted, in order to find a motion vector.

The Octagonal search (OS) method is summarized as follows:

1. Compute SADs at three points: (mv_x^l, mv_y^l), (mv_x^a, mv_y^a), and (mv_x^c, mv_y^c). The point with minimum SAD is set as ISC.

2. Start the search procedure with SQP and R = 2 at ISC.

3. Find the wining point for the next step. If the wining point is the center of the current pattern, R is decremented by one.

4. If R is zero, stop else consider the other pattern (e.g., if SQP is the current pattern CP will be the pattern for next step, and vice versa) at the wining point and go to step 3.

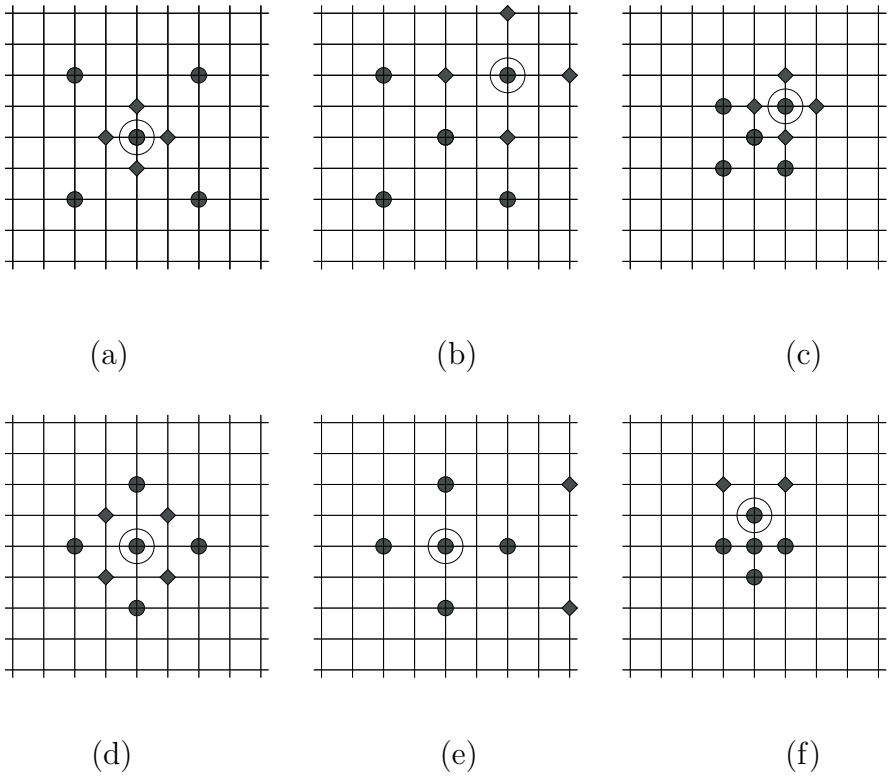
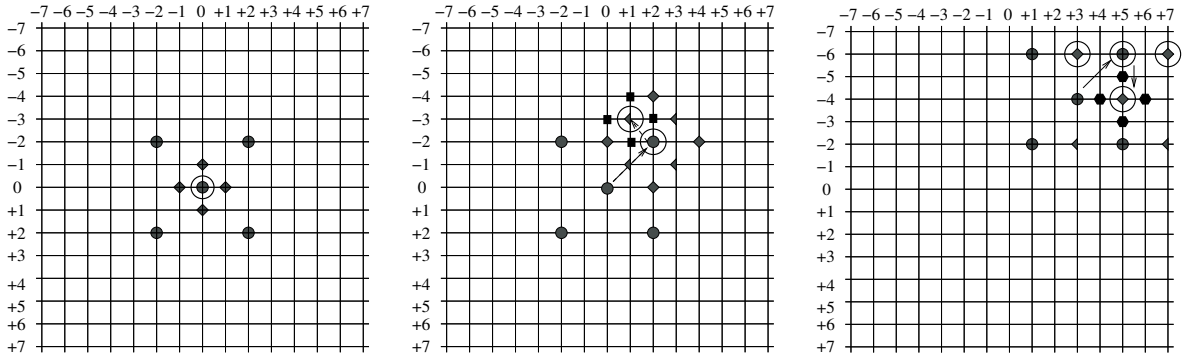


Figure 6.9: Illustrates different possibilities in two successive search steps. (a) - (c) SQP is current pattern, and (d) - (f) CP is the current pattern. (a) - (d) four search points have to be checked in the next step, (e) - (f) two search points have to be checked in the next step.

6.3.2 Complexity of OS method

For fast BMAs, computational complexity can be measured in terms of number of search points required to find a motion vector. Here, we also examine this. In OS method to predict ISC, three motion vectors are considered and we may need to calculate BDM at three points in the worst case or in a single point at best case. During the search, numbers of SPs have to be checked at next step depending on the current search pattern. Different possibilities are explained below:

- Current search pattern is SQP: Three different situations are possible, which



(a) (b) (c)

Figure 6.10: Examples of OS method. Encircled SPs represent duplicate computation of BDM. a) ISC = (0,0), MV(0,0). (b) ISC = (0,0), MV(+1,-3). (c) ISC = (+3,-4), MV(+5,-4).

are shown in Fig. 6.9(a) - 6.9(c). In all of the cases four search points have to be checked in the next step.

- Current search pattern is CP: All three situations are shown in Fig. 6.9(d) - 6.9(f). In first case, four points have to be checked and in the other two cases, two points have to be checked.

All the search points to be checked at a particular step may not be new. Some of them may have been already checked in previous steps. Thus, to avoid the redundancy in computation of BDM, a binary table is used to store checked/unchecked status of the reference blocks. Fig. 6.10 illustrates different search path to find the motion vector (MV), encircled search points are used in more than one search steps.

6.3.3 Performance of OS method

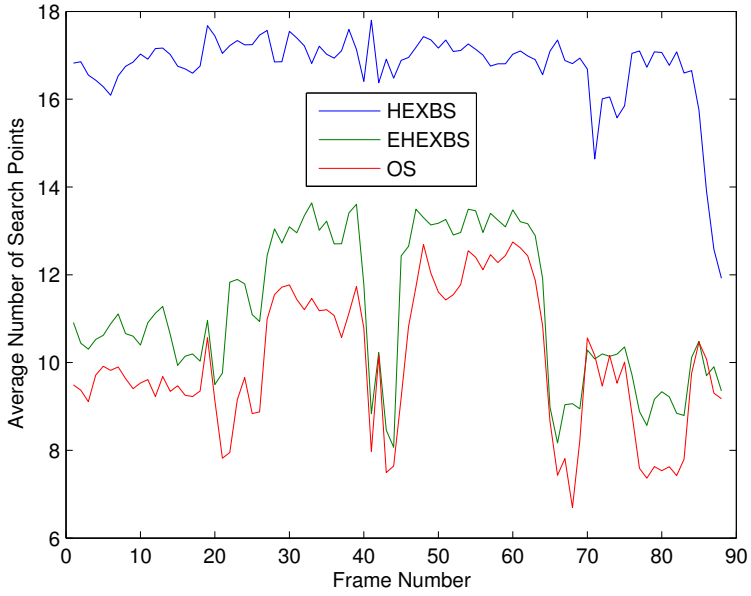
The experiment is conducted on the luminance component of different video sequences mentioned in section 1.6. These video sequences have different frame size and different number of frames. They also consist of different degrees and types of motion content. The performance of OS method is compared against other block matching algorithms like DS, HEXBS, EHEXBS, and ETSS. In all implementations of different BMAs, SAD is used as BDM, block size is taken as 16×16 and search window $(SW) = 15 \times 15$. The OS method is compared with others in terms of two aspects: (i) number of search points per block, and (ii) the PSNR between estimated frame (reconstructed frame, using MV and reference frame) and original frame. The results of various methods are summarized in Table 6.1 and Table 6.2. In our implementations of all BMAs, a binary table is used to avoid the multiple counts of a SP. From the experimental results, it is clear that the proposed OS method is not only faster than other methods; it also gives better estimation as revealed by the PSNR value. For example, to trap the large motion of 'Football', OS and EHEXBS methods takes 10.01 and 11.19 SPs per block and PSNRs are 24.52 dB and 24.45 dB respectively. Similarly, in small motion sequence, e.g., 'Claire', these two methods takes 7.86 and 8.34 SPs and gives 42.80 dB and 42.75 dB as the PSNR. On an average, OS method saves 4.5% SPs compared to EHEXBS and average quality of EHEXBS and OS are 32.42 dB and 32.50 dB respectively. Thus, it is clear that proposed OS method has the capabilities to trap both large motion and small motion activity with equal efficiency. Fig. 6.11(a) and Fig. 6.12(a) plot the number of search points per block using 'Football' and 'Stefan' sequences respectively on a frame-by-frame basis. Fig. 6.11(b) and Fig. 6.12(b) show the frame wise comparison of PSNR using different BMAs for these two sequences. From the experimental results, it is established that OS is fastest one compared to other state-of-the-art BMAs.

Table 6.1: Number of search points during motion estimation with respect to original reference frame using different methods.

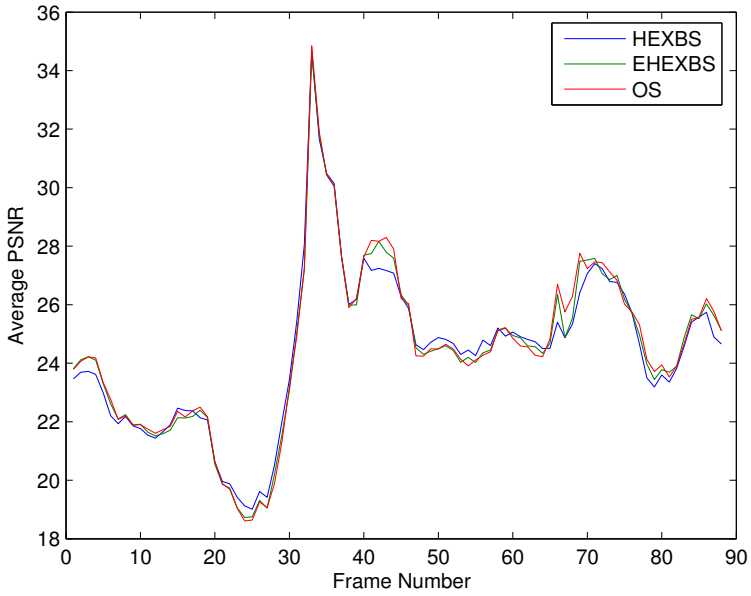
Sequence	DS	ETSS	HEXBS	EHEXBS	OS
Carphone	14.44	15.11	11.22	09.54	09.55
Clarie	11.45	11.53	09.66	08.34	07.86
Coastguard	16.31	17.29	12.78	08.92	08.63
Container	12.34	12.41	10.38	08.66	08.51
Miss America	16.50	15.93	12.16	10.29	10.16
Mobile	13.29	14.34	10.72	09.25	08.61
Mother & Daughter	11.82	11.97	09.84	08.47	08.12
Susie	12.78	13.26	09.96	09.73	09.30
Tennis	14.68	15.22	11.83	09.55	08.84
Hall Monitor	12.83	13.20	10.47	09.09	08.90
Football	24.41	23.98	16.75	11.19	10.01
Foreman	15.90	16.59	12.35	09.80	09.42
News	12.52	12.65	10.51	09.03	08.58
Stefan	16.75	17.52	13.00	09.36	08.73
Tempete	12.78	15.76	10.60	08.98	08.71
Average	14.59	15.12	11.48	9.35	8.93

Table 6.2: PSNR obtained from motion estimation with respect to original reference frame using different methods.

Sequence	DS	ETSS	HEXBS	EHEXBS	OS
Carphone	29.81	29.73	29.53	29.58	29.66
Clarie	42.79	42.79	42.76	42.75	42.80
Coastguard	30.06	30.05	30.00	30.08	30.09
Container	37.81	37.84	37.81	37.82	37.83
Miss America	37.36	37.60	36.91	37.52	37.53
Mobile	23.83	23.86	23.65	23.77	23.86
Mother & Daughter	40.79	40.79	40.69	40.77	40.79
Susie	34.41	34.37	33.82	34.41	34.42
Tennis	28.38	28.25	27.87	28.22	28.44
Hall Monitor	34.27	34.25	34.20	34.19	34.22
Football	24.55	24.56	24.40	24.45	24.51
Foreman	33.06	32.89	32.31	32.72	33.05
News	37.72	37.70	37.58	37.64	37.69
Stefan	24.46	24.54	24.33	25.05	25.24
Tempete	27.36	27.39	27.33	27.30	27.29
Average	32.44	32.44	32.21	32.42	32.50

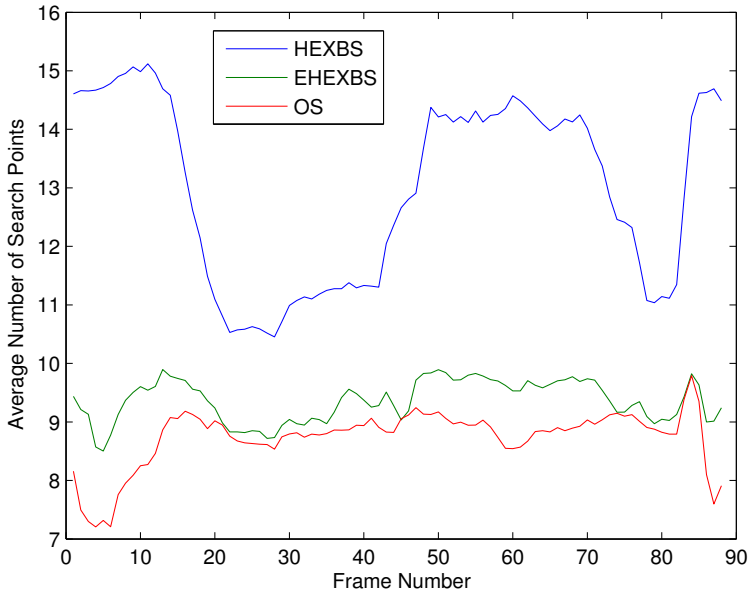


(a)

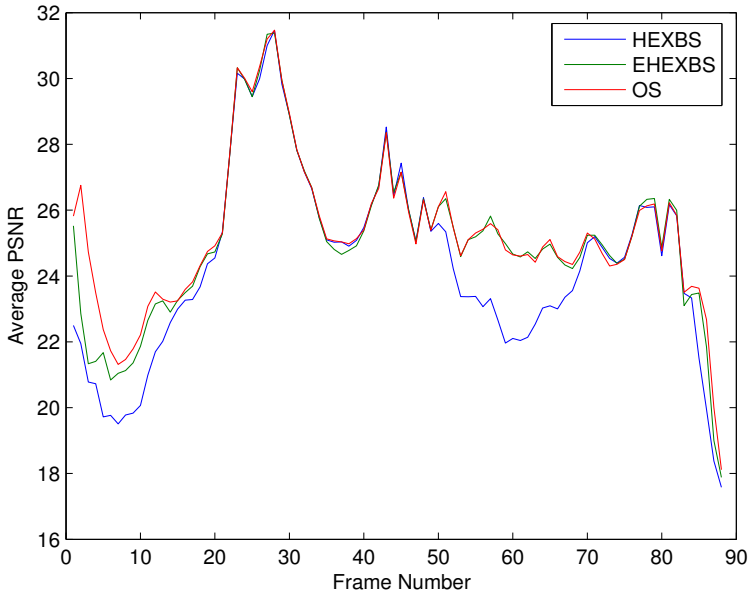


(b)

Figure 6.11: Frame wise performance comparison between different BMA on sequence 'Football' by (a) number of search points per block and (b) PSNR per frame.



(a)



(b)

Figure 6.12: Frame wise performance comparison between different BMA on sequence 'Stefan' by (a) number of search points per block and (b) PSNR per frame.

6.4 Difference frame coding

The video standards include frequency transformation, quantization, and entropy coding for residual frame coding. Discrete cosine transform (DCT) is most widely used because its performance is close to KLT and it is possible to have efficient hardware and software implementation of DCT based coding. Video compression standard like H.261/2, MPEG-2,-4 part 2 [74] uses 8×8 DCT to encode the difference frames. H.264/AVC uses 4×4 integer approximation of DCT. Video standards like H.261/2, MPEG-1/2 use adaptive quantization step where as in H.261 a flat quantization matrix is used. The final component of the video compression standard is the entropy coding. H.264 uses a number of techniques for entropy coding: Golomb codes, context adaptive variable length coding (CAVLC), and context adaptive binary arithmetic coding (CABAC) [125]. For application like video playback, the decoder has to be fast enough to reconstruct the frames from transmitted data; but the most time consuming module of the decoder is the inverse transformation. A BTC-PF based method is employed to encode the residual frames and it results into a faster decoding (see section 4.3.1). The residual frame is obtained after the motion compensation. Hence, it is expected that some regions of the residual frame will contain low and smooth values and some other regions have high contrast values. To achieve better performance, a hierarchical BTC-PF (HBTC-PF) method is introduced. In HBTC-PF method, first the difference frame is partitioned into 16×16 blocks (B_{16}). Then according to the block intensity range, either the block is represented by the block mean or the block is partitioned into equal sized (8×8) sub-blocks. Then, the decomposition of the sub-block is repeated (with some variations) until the size becomes 4×4. In case of high contrast block is estimated from the neighboring blocks and the corresponding error coded. Hierarchical BTC-PF method (HBTC-PF) is employed to encode the residual frames and the method is as given below:

Step 1 The difference frame is first partitioned into 16×16 blocks (B_{16}).

Step 2 For each B_{16} do the following

Step 2.1 Compute the block intensity range (R_{16}).

If $R_{16} \leq Th$, then the block is represented by the block mean

else B_{16} is decomposed into four 8×8 blocks (B_8).

For each B_8 block do the following

Step 2.1.1 Compute the block intensity range (R_8).

If $R_8 \leq Th$, then the block is represented by the block mean

else if $Th < R_8 \leq 2Th$ then a 4×4 block is generated from 8×8 by sub-sampling by 2, and the 4×4 block is then coded by BTC-PF method

else (i.e., $R_8 > 2Th$) four 4×4 blocks (B_4) are constructed.

For each B_4 do the following

Step 2.1.1.a Compute the block intensity range (R_4)

if $R_4 \leq Th$ the block is represented by the block mean

else if $Th < R_4 \leq 2Th$ then the block coded by BTC-PF method

else (i.e. $R_4 > 2Th$) values of the block is estimated from neighbor blocks and corresponding error is quantized and transmitted.

In HBTC-PF method, each B_{16} is partitioned hierarchically and finally coded. To decompose the blocks here we have applied the threshold (Th) on the block intensity range. If we consider large value of Th then this results more smooth block and hence lower quality along with lower bit-rate. So, we set the value of Th experimentally compromising between PSNR and bpp. A bit-stream (called as 'partition-string'), which represents the partitions of B_{16}, has to be sent to the decoder. For block B_{16}, there are two possibilities (see step 2.1); it can be represented either by '0' or '1'. For B_8, three possibilities (see step 2.1.1) and '0', '10', and '11' are used

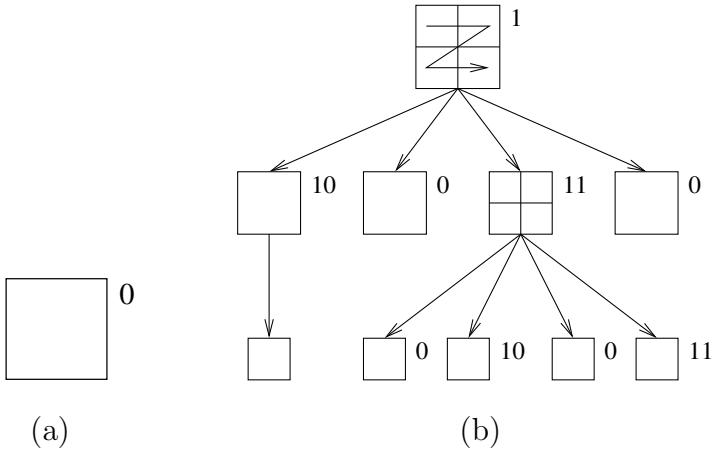


Figure 6.13: The hierarchical partition of B_{16}: (a) smooth block and 'partition-string' is '0', (b) non-smooth block and 'partition-string' is '1100110010011'.

to represent encoding methods by 'block mean', 'sub-sampling followed by BTC-PF', and 'partitioned into 4×4 blocks' respectively. A 4×4 block is encoded by 'block mean', 'BTC-PF' or 'estimation and quantization' (see step 2.1.1a) and these possibilities are represented by '0', '10', and '11' respectively. There are different situations when a B_{16} is partitioned. Some of the possible partitions are shown in Fig. 6.13. Fig. 6.13(a) shows a smooth B_{16} block and an example of partition of a non-smooth B_{16} is shown in Fig. 6.13(b). The decomposition of Fig. 6.13(b) can be represented by the 'partition-string' 1100110010011. The 'partition-string' is defined by considering the level-wise bit-strings of the partition.

6.5 Proposed video coding method

The standard video coding techniques include inter-prediction method, intra-prediction method and residual coding technique. Proposed video coding method (see Fig. 6.14) has three main modules similar to the video coding standard: (i) inter-pixel redundancy removal, (ii) hierarchical BTC-PF method to encode the difference frames (see

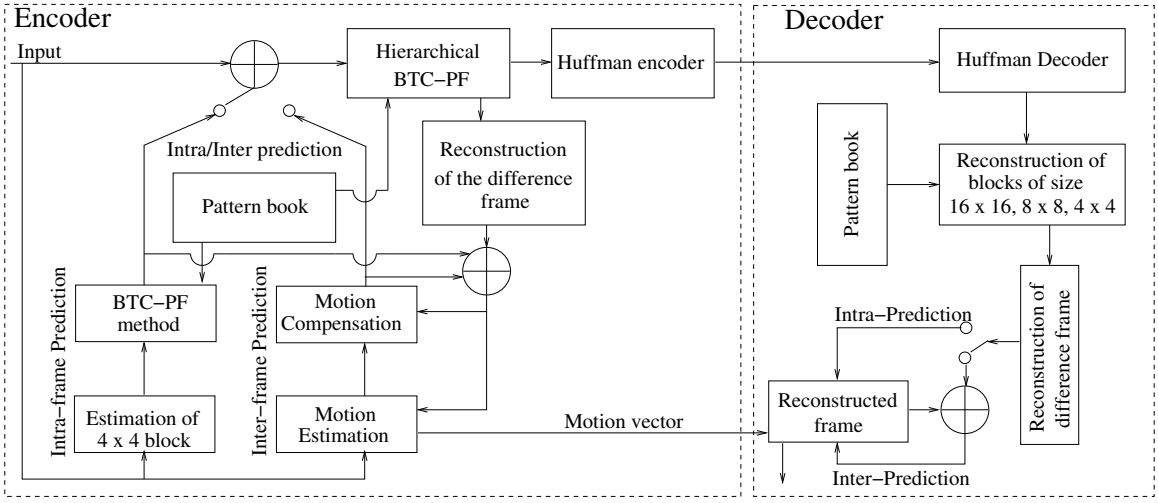


Figure 6.14: Block diagram of the proposed video coding technique.

section 6.4) and (iii) entropy coding for information obtained from other modules. The inter-pixel redundancy includes intra-prediction and inter-prediction. The intra-prediction method has been discussed in section 6.2 and it is similar to the encoding of the O_1 plane of the color image (for details, see section 4.3.1). To reduce the temporal redundancy, the octagon based search method is used. This method is presented in the section 6.3.1. The performance of the proposed video coding method is compared with that of H.264. For implementation of H.264, we have used the source code of H.264, which is available at [144]. In this implementation, the ME method is a hybrid concept of the different ME algorithms like, HEXBS, DS, CS, etc. and some other extensions, like a full search over 3×3 window etc. In our implementation we have used only P-frame [see section 6.3] prediction and better the motion estimation lower is the bit-rate for difference frame coding. To achieve a better result in terms of quality and efficiency we employ a modified octagonal search (MOS) method as described below.

Step 1 Predict the initial search center: For the prediction of motion vector, motion vectors of the spatio-temporal neighbor blocks are used (as shown in Fig. 6.8).

In [36] the motion vector for which the BDM of the current block was minimum was taken as the ISC. To improve the performance[1], in the present work, the motion vector of the current block is predicted by taking component-wise median of the motion vectors of the above-mentioned blocks (as shown in Fig. 6.8). Finally, the ISC (initial search center) is selected from the predicted vector and the (0,0) vector whichever gives the minimum distortion for the current macroblock.

Step 2 Determine the initial search pattern: If the ISC suggests predominantly horizontal or vertical motion, the cross pattern (CP) is selected as the initial search pattern; else the square pattern (SQP) is selected. The initial size of the pattern is 2.

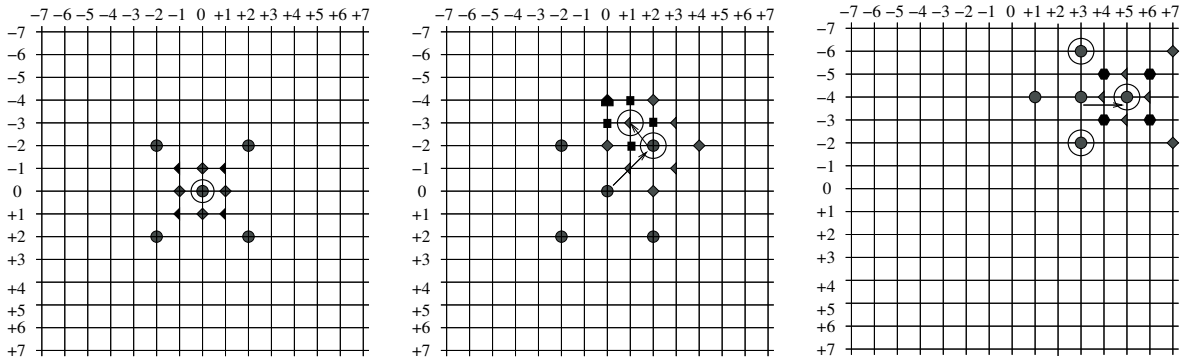
Step 3 Search for the motion vector: Compute the BDMs at the search points given by the current pattern. The point, which gives the minimum distortion, is the wining point and serves as the search center for the next step. If wining point is the center of the current pattern the size of the pattern is decreased by 1.

Step 3a If the **size of the pattern is greater than zero**, consider the other pattern centering at the winning point and goto Step 3.

Step 3b If the size of the pattern becomes zero **for the first time**, then reset pattern size to 1 and consider the other pattern at the wining point and goto Step 3; else stop.

Thus in this modified octagonal search method, the ISC may be the predicted vector or (0,0) vector and the initial search pattern is determined dynamically. Searching method considers some additional search points (see step 3b) to detect motion more precisely compared to one given in [36]. The proposed prediction of ISC reduces search points and the additional search points improve the performance. An illustration of

[1]This is verified experimentally.



(a) (b) (c)

Figure 6.15: Illustration of modified octagonal search method, encircled SPs represents duplicate computation of BDM. a) ISC = (0,0), MV(0,0). (b) ISC = (0,0), MV(+1,-3). (c) ISC = (+3,-4), MV(+5,-4).

different sequence of the modified octagonal search method is shown in Fig. 6.15. The changes (e.g., number of search points or selected search points etc.) in modified octagonal method can be visualized by comparing the Fig. 6.10 and Fig. 6.15.

Entropy coding technique is a lossless coding method and used to reduce the statistical redundancy. In the proposed method, inter-prediction module returns motion vector for each macroblock; and HBTC-PF method (on residual frames) outputs Q graylevels and the index of the selected pattern for each block (of size 4×4). In addition, in the HBTC-PF method, B_{16} block is partitioned hierarchically and coded. In H.264, to achieve greater compression context-based entropy coding is used. In our method, we have used Huffman coding technique to encode all the information except the motion vector. The motion vectors are encoded in straightway.

Table 6.3: Comparative performance of H.264 ME and proposed MOS algorithms.

Video sequence	H.264 ME algo			MOS		
	PSNR	bpp	ME time (in sec.)	PSNR	bpp	ME time (in sec.)
Carphone	35.83	0.317	1.673	35.85	0.309	0.919
Claire	38.70	0.070	1.151	38.71	0.070	0.789
Coastguard	33.57	0.508	1.709	33.57	0.462	1.001
Container	35.46	0.107	1.114	35.46	0.107	0.830
Miss America	39.06	0.078	0.622	39.02	0.078	0.653
Mobile	32.33	1.363	1.732	32.33	1.358	0.884
Mother Daughter	36.29	0.105	1.377	36.28	0.103	0.827
Susie	36.31	0.191	0.849	36.34	0.191	0.435
Tennis	33.59	0.435	1.473	33.59	0.434	1.419
Hall Monitor	37.37	0.097	3.768	37.38	0.096	3.408
Football	36.72	0.386	2.501	36.76	0.402	1.826
Foreman	35.93	0.234	1.596	35.90	0.241	1.352
News	37.38	0.099	1.255	37.37	0.098	1.217
Stefan	34.52	0.719	1.575	34.53	0.738	1.311
Tempete	33.89	0.789	4.979	33.88	0.797	3.697
Average	35.80	0.366	1.825	35.80	0.365	1.371

```
------------------------------ JM 10.1 (FREXt) ------------------------------
YUV Format                        : YUV 4:2:0
Frames to be encoded I-P/B        : 300/0
PicInterlace / MbInterlace        : 0/0
Transform8x8Mode                  : 1
Freq. for encoded bitstream       : 30
Hadamard transform                : Not used
Error robustness                  : Off
Search range                      : 16
Total number of references        : 1
References for P slices           : 1
Sequence type                     : IPPP (QP: I 28, P 28)
Entropy coding method             : CAVLC
Profile/Level IDC                 : (100,40)
Motion Estimation Scheme          : HEX
Search range restrictions         : none
RD-optimized mode decision        : not used
Data Partitioning Mode            : 1 partition
Output File Format                : H.264 Bit Stream File Format
Residue Color Transform           : not used
------------------------------------------------------------------------------
xit JM 10 (FREXt) encoder ver 10.1
```

Figure 6.16: Coding environment in H.264 platform.

6.6 Performance of the video coding method

In the proposed method, we have introduced two tools for video coding: a fast motion estimation method and a BTC-PF based method for coding the intra-frame as well as difference frames which leads to a very fast decoding method. The performance of the proposed video coding method is compared with H.264. To do so, we use the source code of H.264 which is available at [144]. In this implementation, we have used IPPP (only one I-frame (first frame) and remaining are P-frames) structure to reduce the inter-pixel redundancy. We set the parameters of the reference software to change the coding environment (e.g., IPPP is the coding structure, block size is 16×16, no multiple reference, no sub-pixel motion estimation, etc.) similar to the proposed method. The used coding environment of H.264 is shown in Fig. 6.16. The comparative results of the proposed method, i.e., the modified octagonal search method and the motion estimation method used in the reference software is reported

in Table 6.3. In this testing, our ME algorithm is used in H.264 environment. It is clear that the proposed ME method (MOS) have comparable performance in terms of PSNR (quality) and bpp (bit-rate). Again, MOS is faster than the method used in H.264 [144]. In Table 6.4, the performance of the proposed video coding method is compared with that of H.264 and is found little inferior. The main reason is that the proposed video coding method uses HBTC-PF in place of transform based coding. It is well established that the performance of the transform based image compression methods is superior over that of spatial domain based methods and almost all the video coding standards use the transform based compression technique. In the proposed method, we have purposefully used BTC-PF method as our main target to develop a coding method which would lead to a very efficient decoding technique. The complexity of the decoder of any transform based approach is same as that of encoder because the complexity of both forward and inverse transform is same. BTC-PF method is asymmetric in that sense, i.e., the time complexity of the decoder is negligible compared to the encoder. The decoder requires only table look-up and a few operations; the basic steps of the proposed decoder are shown in the right part of the Fig. 6.14. The decoding technique of transform based method using DCT normally uses 8×8 inverse DCT, which requires 176 multiplications and 464 additions; on the other hand, decoder of BTC-PF based method requires no multiplication and only a few additions on an average. The details of the computation of using DCT are given in [37]. A 4×4 integer approximation of DCT is used in H.264, and the inverse transformation in that case needs 64 additions and 16 shifting operations [103]. Computational requirement of the proposed method may be described as follows. HBTC-PF partitions the blocks B_{16} hierarchically based on the intensity range and a block is coded by

- block mean; and no operation is required to reconstruct.
- BTC-PF method; in this experiment a 3-level BTC-PF method is employed

which returns index I, and graylevels μ_1, μ_2, and μ_3. With some extra bits, called 'order index', the graylevels are ordered as $\mu_1^{'} \leq \mu_2^{'} \leq \mu_3^{'}$ and then coded as $\mu_1^{'}, \mu_2^{'} - \mu_1^{'}, \mu_3^{'} - \mu_2^{'}$. Hence, 2 additions are required for reconstruction.

- quantized prediction error; median of three neighbors is used as the predicted value and then prediction error is quantized. So for each pixel 3 comparisons, one shifting operation and one addition are required for reconstruction.

In the proposed method, it is estimated that 15% (approx.) blocks are coded by prediction method and 20% (approx.) blocks are coded by BTC-PF. All these calculations are based on blocks of size 4×4. The complexity of the proposed decoder is summarized in the Table 6.5. Table 6.5 indicates how the proposed decoder is faster than that of standard method(s).

6.7 Conclusions

The standard video coding methods use the transform coding technique to encode the residual frames. Decoding of compressed data resulted through such method takes significant time for inverse transformation. Decoding of compressed data, on the other hand, is usually much faster if compression is done by some spatial domain method. In this chapter, a BTC-PF based method is used to encode both I-frame and the residual frames of video sequence. The residual frames are obtained through motion estimation, the major time consuming module in video coding method. So octagonal pattern based fast motion estimation is introduced and its performance (speed in particular) is found to be better than other motion estimation methods. We have also compared the performance of the MOS method in H.264 platform. The result shows that they are comparable in terms of quality and bit-rate but the proposed method is much faster. The performance of the proposed video coding

Table 6.4: Comparisons of proposed video coding and H.264. For both cases modified octagonal search (MOS) method is used for ME.

Video sequence	Proposed video coding		H.264	
	PSNR	bpp	PSNR	bpp
Carphone	35.49	0.516	35.85	0.309
Claire	38.42	0.183	38.71	0.070
Coastguard	33.10	0.692	33.57	0.462
Container	35.57	0.283	35.46	0.107
Miss America	38.62	0.182	39.02	0.078
Mobile	33.88	1.593	32.33	1.358
Mother Daughter	35.73	0.271	36.28	0.103
Susie	35.49	0.340	36.34	0.191
Tennis	33.57	0.761	33.59	0.434
Hall Monitor	37.03	0.242	37.38	0.096
Football	34.12	0.768	36.76	0.402
Foreman	35.40	0.431	35.90	0.241
News	36.88	0.258	37.37	0.098
Stefan	35.02	1.144	34.53	0.738
Tempete	34.23	1.141	33.88	0.797
Average	35.50	0.587	35.80	0.366

Table 6.5: Average complexity of the decoder at block level.

Block Size	Video Standard				Proposed method			
	Mult	Add	Comp	Shift	Mult	Add	Comp	Shift
4×4[2]	0	64	0	16	0	2.80	7.20	2.40
8×8[3]	176	464	0	0	0	11.20	28.80	9.60

method (BTC-PF method for I-frame and MOS followed by HBTC-PF for P-frame) is also compared with H.264 and found that quality is nearly same, but bit-rate of our method is little higher than H.264. In H.264, there are several features which are to be set by the user. Hence, coding selection and preparation of the environment is a complex task. However, in our method the motion vector is coded straightway and other data (e.g., graylevels, indices of the selected pattern, etc.) are encoded by Huffman coding technique. To improve the bit-rate further, we need to search for efficient coding technique.

[2]The complexity of video standard according to integer approximation of DCT used in H.264 [103].

[3]Complexity of other video standard, considering fast DCT [101].

Chapter 7

Conclusions

The objective of this thesis is to develop an image compression method for which the decoder would be very efficient. Such method is suitable in situations where image or video is compressed once but decoded frequently, e.g., image retrieval, video-on-demand and video playback. It is clear that the decoding time due to spatial domain based compression is much less than that of the sub-band compression techniques. Two widely used spatial domain compression techniques are block truncation coding (BTC) and vector quantization (VQ). BTC method results in good quality image with high bit-rate, while the VQ is well known for low bit-rate but produces poor quality image. This thesis proposes a hybrid compression method using the concept of BTC and VQ to achieve a good compromise between bit- rate and quality. The proposed compression method, namely the block truncation coding using pattern fitting (BTC-PF), inherits the advantages of both BTC and VQ methods. In BTC-PF method (see section 2.4), there is a Q-level patternbook and, for each image block, the best pattern is selected from it. Then the quantization level is determined. Hence, given the patternbook, to represent a block, Q grayvalues and the index of the selected pattern are required. From the index value, we obtain the pattern of the block (by

table look-up method) and a few simple operations (like, addition, subtraction, and shifting) are required to obtain the values to be assigned to the pixels. The number of operations is much less than what is required in the transform coding method. The BTC-PF method is employed to compress grayscale images, color images as well as the residual frames of video.

In the grayscale compression (see chapter 3) we use 2-level patterns to partition the image blocks. In the proposed 2BTC-PF method, images are partitioned into 4×4 blocks and each block is encoded independently. The reconstruction levels are $A - d$ and $A + d$, which resemble the bias (A) and contrast (d) of the block. This representation helps to reduce the bit-rate further. If the contrast is low, then the block is represented by 'A' (i.e., by block mean) only and no bit-pattern is required. The bias A of neighboring blocks has strong correlation and this suggests the predictive coding of A. The performance of the 2BTC-PF method is found superior to other BTC based method.

Though this method is inferior to JPEG, a transform domain method, it is well ahead to the transform domain method in terms of computational cost and is suitable for intended. The 2BTC-PF method is further extended to low bit-rate image coding. In this method (LBTC-PF) the images are partitioned into 8×8 blocks then from each 8×8 block, two 4×4 blocks are generated by quincunx sub-sampling method. Each 4×4 block is then encoded by 2BTC-PF method. During reconstruction, the interpolation technique is employed. In 2BTC-PF method, the indices are encoded in straightway, whereas in LBTC-PF method the index graph (IG) is used for the same purpose. The performance of LBTC-PF is also compared with other BTC based methods and the former is found to give better results. Compared to the result of 2BTC-PF, the LBTC-PF produces much higher compression ratio at very little sacrifice in quality. Computation cost of both the methods are comparable. The quality of reconstructed image due to our methods can be increased by considering

larger patternbook, obviously at the expense of bits. Secondly, during coding, we have used the fixed size block. The performance (bit-rate) of the proposed methods can be improved by considering the variable block size. The variable size coding can be accomplished through the quadtree structure. We use the variable block size coding in the subsequent methods.

BTC-PF based color image compression method (CBTC-PF) is presented in chapter 4. In this method, the spectral redundancy between the color planes are reduced by transforming RGB to $O_1O_2O_3$. This transformation has some advantages over the standard transformation like, YIQ, YCbCr and HVS. For example, this transformation requires integer arithmetic only and, thus, is lossless. The performance of CBTC-PF in $O_1O_2O_3$ domain is better compared to that of YIQ. The O_1 plane represents the luminance component and O_2, O_3 the chrominance component of the image. The coding of O_1 is done in two steps. In the first step the image blocks are estimated from the reconstructed neighbor blocks. In the next step the residue image is coded by BTC-PF using variable block size. Here, the image blocks are decomposed like a quadtree on the basis of block variance. We start with larger block and decompose in quadtree manner based on the smoothness judged against a threshold. Changing the threshold value, with a fixed patternbook, we can control the quality of the reconstructed image and bit rate. For O_2 and O_3 planes we spent less bit as they are not visually so important and for each of these planes the threshold value is higher than that of O_1 plane. For O_1 plane 3-level patternbook is used, whereas for O_2 and O_3 planes 2-level patternbook is used and the size of the patternbook is also smaller. Depending on the importance, we can change the patternbook (i.e., the number of levels and the size of the patternbook) and control the performance. The performance of the CBTC-PF method is better than that of the BTC based methods. Though the performance of the proposed method is little lower than JPEG, the computational cost of CBTC-PF method is negligible compared to that of JPEG.

The BTC-PF method provides a good solution to the progressive image transmission. The proposed method, i.e., progressive block truncation coding using pattern fitting (PBTC-PF) is a low cost method and its decoder is also faster. Here, the patternbook is organized like a full search progressive transmission tree (FSPTT), where the leaf nodes contain the patterns from the patternbook used by the BTC-PF method. In the implementation of PBTC-PF, we consider the multilevel decomposition of the image blocks. The grayvalues 'A', 'd' and the FSPTT structure of patternbook helps greatly in progressive transmission. The lossless transmission is also a requirement in some applications like, transmission of medical images. BTC-PF method is a lossy coding technique; hence, to provide lossless transmission facility, a lossless coding technique has to be incorporated after the PBTC-PF. In this work, we have employed CALIC, a lossless coding technique, on the residual image as the second phase of PIT. Experimental result reveals that the proposed lossless transmission, that is PIT followed by CALIC, requires slightly more bits than that if CALIC transmits the original image in conventional way. For example, the proposed method requires about 9.5% bits extra compared to CALIC, if the entire image is transmitted up to the end. However, if the image transmission is terminated immediately, as it happens in case of image search and download where multiple image transmission is involved, the overall gain of the proposed method is significant over CALIC. For the progressive transmission of color images, first the images are transformed into $O_1O_2O_3$ domain (explained earlier). Then each plane is coded by PBTC-PF with quadtree structure. Finally, the information transmitted in an interleaved manner according to the priority of the planes to ensure the color appearance of the reconstructed image. The proposed color image progressive transmission method outperforms some recently proposed spatial domain methods.

BTC-PF is also used to develop a video coding method, where to encode both the I-frames as well as the P-frames (more specifically, residual frames) the BTC-PF method is employed. In video coding method, the residual frame is resulted after

the reduction of temporal redundancy (by motion compensation). The I-frame is encoded by the same method used to encode O_1 plane of the color images. To encode the residual frame a hierarchical BTC-PF (HBTC-PF) method is used. The motion estimation is the most time consuming module of any video coding method. We propose an octagon based search method, which is faster than the currently used other motion estimation methods and produces the result of similar quality. The performance of the proposed motion estimation method is evaluated in H.264 environment. The result shows that its performance (in terms of quality and bit-rate) is same and the estimation method is faster. The performance of the proposed complete video coding method is also compared with H.264. The result shows that the performance of the proposed method is little poorer than that of H.264 as expected, since the latter is a transform domain method. However, as the decoder of our method is mainly based on table look-up method, its computational cost is negligible compared to that of H.264. The bit-rate of the proposed method may be improved further by incorporating some efficient symbol coding technique.

7.1 Future scope of work

This thesis is work on the BTC-PF method. In BTC-PF method a patternbook is used to encode an image. To represent a block either the mean graylevel of the block (for low contrast block) or Q different graylevels are used. In this method, the high contrast or middle contrast blocks are encoded by same patternbook, sometimes this may deteriorate the performance. The performance can be improved by considering different patternbook for different types of block like classified VQ. We have used one patternbook for O_1 and another patternbook for O_2 and O_3. Performance may be improved by using different patternbooks for O_2 and O_3. Performance of BTC-PF method highly depends on the patternbook. So generation of patternbook is a critical

task. In this work we have generated patternbook (i) manually based on intensive observation of intensity pattern (representing various structural features), and (ii) by employing K-means clustering algorithm on the intensity block. Better patternbook may be generated by using more advanced clustering methods and/or using other image features in addition to intensity.

In PIT, we have considered FSPTT structure of the patternbook. The intermediate patterns are also at the leaf level. An intermediate pattern is one among its children which gives minimum Hamming distance from the other patterns. Hence, the patterns at some intermediate levels are not so good enough. We may achieve better results at the intermediate levels, if we use separate patternbooks for each level. This structure would increase the search (encoding) time. However, we may reduce the search time, if we are able to design the patternbooks in such a way that a pattern at one level gives a list of patterns that have to be checked at the next level. This organization of the patterns (considering all the patternbooks) reduces the search time and is expected to give better result. This changed organization of the patterns can no longer be a tree, now this converts into a directed graph like the one in finite state VQ.

In BTC-PF method the selected pattern helps to know the intensity pattern of the reconstructed block. The change in intensity of the pixels can be used to region segmentation and edge detection. So from the compressed data stream these features can be computed. The histogram of the compressed image can be easily computed based on the pattern and the graylevels. Hence, the BTC-PF method can be used in histogram based applications like search and retrieval to reduce the volume of the data space. Some global features may be extracted from BTC-PF compressed image. Image watermarking is an important solution for copyright protection, ownership identification, etc. The simplest spatial domain method is to modify the least significant bit to embed a binary signature or message. The BTC-PF method returns the index and Q graylevels. So we can use the BTC-PF method to embed a message

into an image (in compressed domain) either by modifying the index or the graylevels without affecting the visual quality.

Motion estimation is not major concern of the thesis; we used it in video coding. In motion estimation we have used integer-pixel motion estimation and. For this purpose sum of absolute difference (SAD) is used as the error metric to find the best matched block. In the proposed fast ME method we have employed the single initial point to start the search process and also have used a single reference frame. The performance of the ME method can be improved by considering multiple initial points, multiple reference frames, sub-pixel accuracy, and variable block size. The computation of SAD is the computationally intensive task and this can be reduced either by considering simple metric or by reducing the block size. We can reduce the cost by adopting the multiresolution approach. In the multiresolution approach the candidate block is compared with the reference blocks at low resolution (i.e., consider some pixels of the blocks) and then some blocks (say, 50% of possible search blocks) are selected. Then these blocks are compared with the candidate block subsequently at higher resolution, and the process is repeated. We stop when the blocks are at full resolution.

The above discussion may be summarized as follows. We can improve or control the performance of BTC-PF by changing the threshold value, by considering different patternbooks, by changing the size of the codebook, etc. We can employ multiresolution search method and sub-pixel search method in motion estimation to achieve a good result. We can use BTC-PF method in compressed domain applications where feature extraction is essential. We can also use the BTC-PF method for copyright protection, identification of the owner, etc. Our work is in progress in these areas we are looking forward to achieve better results or solve these problems.

Bibliography

[1] Center for image processing research (cipr) video sequences available at. *http://www.cipr.rpi.edu/resource/sequences/index.html*.

[2] The usc-sipi image database available at. *http://sipi.usc.edu/database/*.

[3] Yuv video sequences available at. *http://trace.eas.asu.edu/yuv/index.html*.

[4] H. Abut. Image compression using non-adaptive spatial vector quantization. In *Proc. of the* 16^{th} *Asilomar conference Circuits, Systems and Computers*, pages 55–61, 1982.

[5] T. G. Ahn, Y. H. Moon, and J. H. Kim. Fast full search motion estimation based on multilevel successive elimination algorithm. *IEEE Transactions on Circuits and Systems for Video Technology*, 14:1265 – 1269, 2004.

[6] B. Aiazzi, L. Alparone, and S. Baronti. A reduced laplacian pyramid for lossless and progressive image transmission. *IEEE Transactions on Communications*, 44:18 – 22, 1996.

[7] R. Aravind and A. Gersho. Low-rate image coding with finite-state vector quantization. In *Proc. of the IEEE international conference on Acoustic, Speech, and Signal Processing*, pages 137 – 140, 1986.

[8] R. B. Arps. Binary image compression. In *Image Transmission Techniques*, pages 219 – 276. Academic Press, W. K. Pratt (Ed.), New York, 1979.

[9] Joint Video Specification (ITU-T Rec. H.264 ISO/IEC 14496-10 AVC). Joint Committee Draft, Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, document JVT-G050r1.doc . 2003.

[10] R. Baker and H. H. Shen. A finite-state vector quantizer for low-rate image sequence coding. In *Proc. of the IEEE international conference on Acoustic, Speech, and Signal Processing*, pages 760 – 763, 1987.

[11] R. L. Baker and R. M. Gray. Differential vector quantization of achromatic imagery. In *Proc. of the international Picture Coding Symposium*, pages 105 – 106, 1983.

[12] U. Bayazit and W. A. Pearlman. Variable-length constrained-storage tree-structured vector quantization. *IEEE Transactions on Image Processing*, 8:321 – 331, 1999.

[13] V. Bhaskaran and K. Konstantinides. *Image and Video Compression Standards: Algorithms and Architectures, 2nd ed.* Kluwer Academic Publishers, 1999.

[14] M. Brunig and W. Niehsen. Fast full search block matching. *IEEE Transactions on Circuits and Systems for Video Technology*, 11:241 – 247, 2001.

[15] K. W. Chan and K. L. Chan. Optimisation of multi-level block truncation coding. *Signal Processing: Image Communication*, 16:445 – 459, 2001.

[16] C. C. Chang and T. S. Chen. New tree-structured vector quantization with closed-coupled multipath searching method. *Optical Engineering*, 36:1713 – 1720, 1997.

[17] C. C. Chang, H. C. Hsia, and T. S. Chen. A progressive image transmission scheme based on block truncation coding. In *LNCS Vol 2105*, pages 383–397, 2001.

[18] C. C. Chang and Y. C. Hu. Hybrid image compression methods based on vector quantization and block truncation coding. *Optical Engineering*, 38:591 – 598, 1999.

[19] C. C. Chang, Jr. C. Jau, and T. S. Chen. A fast reconstruction method for transmitting images progressively. *IEEE Transactions on Consumer Electronics*, 44:1225 – 1233, 1998.

[20] C. C. Chang and I. C. Lin. Novell full-search schemes for speeding up image coding using vector quantization. *Real Time Imaging*, 10:95 – 102, 2004.

[21] C. C. Chang, T. K. Shih, and I. C. Lin. Guessing by neighbors: an efficient reconstruction method for transmitting image progressively. *The visual computer*, 19:342–353, 2003.

[22] C. C. Chang, F. C. Shine, and T. S. Chen. A new scheme of progressive image transmission based on bit-plane method. In *Proc. of the* 5^{th} *Asia pacific conference on communications and fourth optoelectronics and communications conference*, pages 892 – 895, 1999.

[23] C. W. Chao, C. H. Hsiesh, and P. C. Lu. Image compression using modified block truncation coding algorithm. *Signal Processing: Image Communication*, 12:1 – 11, 1998.

[24] C. C. Chen and M. N. Wu. A color image progressive transmission method by common bit map block truncation coding approach. In *Proc. of the international conference on Communication Technology*, pages 1774–1778, 2003.

[25] T. S. Chen and C. Y. Lin. A new imrpovement of jpeg progressive image transmission using weight table of quantized dct coefficient bits. In *Proc. of the* 3^{rd} *IEEE pacific rim conference on multimedia*, pages 720–728, 2002.

[26] S. C. Cheng and W. H. Tsai. Image compression using adaptive multilevel block truncation coding. *Journals of Visual Communications and Image Representations*, 4:225 – 241, 1993.

[27] A. Christopoulos, A. Skodras, and T. Ebrahimi. Jpeg2000 still image coding system: an overview. *IEEE Transactions on Consumer Electronics*, 46:1103–1127, 2000.

[28] P. C. Cosman, K. L. Oehler, E. A. Riskin, and R. M. Gray. Using vector quantization for image processing. *Proc. of IEEE*, 81:1326 – 1341, 1993.

[29] G. M. Davis and A. Nosratinia. Wavelet-based image coding: an overview. *Applied and Computational Control, Signals, and Circuits,*, 1:205 – 269, 1998.

[30] E. J. Delp and O. R. Mitchell. Image compression using block truncation coding. *IEEE Transactions on Communications*, 27:1335 – 1342, 1979.

[31] J. Dewitte and J. Ronsin. Original block coding scheme for low bit rate image transmission. *Signal Processing II: Theories and Applications*, 1983.

[32] B. C. Dhara and B. Chanda. Image compression algorithm based on btc and pattern fitting method. In *Proc. of the* 5^{th} *international conference on Advances in Pattern Recognition*, pages 299 – 302, 2003.

[33] B. C. Dhara and B. Chanda. Block truncation coding using pattern fitting. *Pattern Recognition*, 37:2131 – 2139, 2004.

[34] B. C. Dhara and B. Chanda. A modified btc using quincunx subsampling and pattern fitting for very low bpp. *Proc. of the* 4^{th} *Indian conference on Computer Vision, Graphics and Image Processing*, pages 563 – 568, 2004.

[35] B. C. Dhara and B. Chanda. Video motion estimation using prediction based hybrid approach. In *Proc. of the* 17^{th} *international conference on Pattern Recognition*, volume 4, pages 737 – 740, 2004.

[36] B. C. Dhara and B. Chanda. Block motion estimation using predicted partial octagonal search. In *Proc. of the IET international conference on Visual Information Engineering*, pages 277 – 282, 2006.

[37] B. C. Dhara and B. Chanda. Color image compression based on block truncation coding using pattern fitting principle. *Pattern Recognition*, 40:2408 – 2417, 2007.

[38] B. C. Dhara and B. Chanda. A lossless progressive image transmission scheme using btc-pf and calic. *Communicated to Pattern Recognition*, 2007.

[39] B. C. Dhara and B. Chanda. Progressive transmission scheme for color images using btc-pf method. In *Proc. of the* 6^{th} *international conference on Advances in Pattern Recognition*, pages 180 – 185, 2007.

[40] B. C. Dhara and B. Chanda. A video coding technique using octagonal motion search and btc-pf method for fast recosntruction. *Communicated to Signal Image Video Processing*, 2007.

[41] C. Du and Y. He. A comparative study of motion estimation for low bit rate video coding. *SPIE Proc. Visual Communications and Image Processing*, 4067:1239 – 1249, 2000.

[42] S. Dunn. Digital color. *http://davis.wpi.edu/~matt/courses/color*.

[43] T. Ebrahimi, C. A. Christopoulos, and D. T. Lee. Special issue introduction, jpeg2000. *Signal Processing: Image Communication*, 17:1–144, 2002.

[44] M. P. Eckert and A. P. Bradley. Perceptual quality metrics applied to still image compression. *Signal Processing*, 70:177 – 200, 1998.

[45] N. Efrati, H. Liciztin, and H. B. Mitchell. Classified block truncation coding-vector quantization: An edge sensitive image compression algorithm. *Signal Processing: Image communication*, 3:275 – 283, 1991.

[46] P. Franti, T. Kaukoranta, and O. Nevalainen. On the design of a hierarchical btc-vq compression system. *Signal Processing: Image Communication*, 8:551 – 562, 1996.

[47] P. Franti and O. Nevalainen. Block truncation coding with entropy coding. *IEEE Transactions on Communications*, 43:1677–1685, 1995.

[48] P. Franti, O. Nevalainen, and T. Kaukoranta. Compression of digital images by block truncation coding: A survey. *The Computer Journal*, 37:308–332, 1994.

[49] A. Q. Gao, C. J. Duanmu, and C. R. Zou. A multilevel successive elimination algorithm for block matching motion estimation. *IEEE Transactions on Image Processing*, 9:501 – 504, 2000.

[50] M. Ghanbari. The cross-search algorithm for motion estimation. *IEEE Transactions on Communications*, 38:950–953, 1990.

[51] B. Girod. Motion-compensating prediction with fractional-pel accuracy. *IEEE Transactions on Communications*, 41:604 – 612, 1993.

[52] B. Girod. What's wrong with mean squared error? *Visual Factors of Electronic Image Communications*, pages 207 – 220, the MIT press, 1993.

[53] D. Gleich, P. Planinsic, and Z. Cucej. Low bitrate video coding using wavelet transform. In *Proc. of the EURASIP conference on Video/Image Processing and Multimedia Communications*, pages 369 – 374, 2003.

[54] T. Goeddel and S. Bass. A two dimensional-quantizer for coding of digital imagery. *IEEE Transactions on Communications*, 29:60 – 67, 1981.

[55] M. Goldberg and L. Wang. Comparative performance of pyramid data structures for progressive image transmission. *IEEE Transactions on Communications*, 39:540 – 548, 1991.

[56] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Prentice Hall , 2002.

[57] C. Grecos, A. Saparon, and V. Chouliaras. Three novel low complexity scanning orders for mpeg-2 full search motion estimation. *Real Time Imaging*, 10:53 – 65, 2004.

[58] Independent JPEG Group. The independent jpeg group's free jpeg software available at. *ftp://ftp.uu.net/graphics/jpeg/*.

[59] ITU-T Recommendation H.261. Video codec for audiovisual services at p×64 kbits/s. 1993.

[60] D. Halversion, N. Griswold, and G. Wise. A generalized block truncation coding algorithm for image compression. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32:664 – 668, 1984.

[61] A. Haoui and D. Messerschmitt. Predictive vector quantization. In *Proc. of the IEEE international conference on Acoustic, Speech, and Signal Processing*, pages 420 – 423, 1984.

[62] D. Healy and O. R. Mitchell. Digital video bandwidth compression using block truncation coding. *IEEE Transactions on Communications*, 29:1809 – 1817, 1981.

[63] C. H. Hsieh, W. Y. Shao, and M. H. Jing. Image compression based on multi-stage vector quantization. *Journal of Visual Communication and Image Representation*, 11:374 – 384, 2000.

[64] Y. C. Hu. Improved moment preserving block truncation coding for image compression. *Electronics Letters*, 39:1377 – 1379, 2003.

[65] Y. C. Hu and C. C. Chang. Quadtree-segmented image coding schemes using vector quantization and block truncation coding. *Optical Enigineering*, 39:464 – 471, 2000.

[66] Y. C. Hu and C. C. Chang. Edge detection using block truncation coding. *International Journal of Pattern Recognition and Artificial Intelligence*, 17:951 – 966, 2003.

[67] Y. C. Hu and J. H. Jiang. Low-complexity progressive image transmission scheme based on quadtree segmentation. *Real Time Imaing*, 11:59 – 70, 2005.

[68] C. M. Huang, Q. Bi, G. S. Stiles, and R. W. Harris. Fast full search equivalent encoding algorithms for image compression using vector quantization. *IEEE Transactions on Image Processing*, 1:413 – 416, 1992.

[69] C. S. Huang and Y. Lin. Hybrid block truncation coding. *IEEE Signal Processing Letters*, 4:328 – 330, 1997.

[70] D. Huffman. A method for the construction of minimum redundancy codes. *Proceeding of the IRE*, 40:1098 – 1101, 1952.

[71] K. C. Hui, W. C. Siu, and Y. L. Chan. Fast motion estimation of arbitrarily shaped video objects in mpeg-4. *Signal Processing: Image Communication*, 18:33 – 50, 2003.

[72] W. J. Hwang and B. Y. Ye. Storage and entropy-constrained multi-stage vector quantization and its application to progressive image transmission. *IEEE Transactions on Consumer Electronics*, 43:17 – 23, 1997.

[73] I. Ismaeil, A. Docef, F. Kossentini, and R. Ward. Efficient motion estimation using spatial and temporal motion vector prediction. In *Proc. of the international conference on Image Processing*, pages 70 – 74, 1999.

[74] ISO/IEC 14496-2 (MPEG-4 Video) . Information technology: Coding of natural/visual objects, part 2: Visual. 1999.

[75] ISO/IEC CD 11172-2 (MPEG-1 Video) . Information technology - coding of moving pictures and associated audio for digital storage media at up to about 1.5 mbits/s: Video. 1993.

[76] ISO/IEC CD 13818-2 ITU-T H.262 (MPEG-2 Video) . Information technology - generic coding of moving pictures and associated audio information: Video. 1995.

[77] A. E. Jacquin. Image coding based on a fractal theory of iterated contractive image transformations. *IEEE Transactions on Image Processing*, 1:18 – 30, 1992.

[78] J. Jain and A. Jain. Displacement measurement and its application in interframe image coding. *IEEE Transactions on Communications*, 29:1799–1808, 1981.

[79] D. G. Jeong and J. D. Gibson. Image coding with uniform and piecewise-uniform vector quantizers. *IEEE Transactions on Image Processing*, 4:140 – 146, 1995.

[80] X. Jing and L. P. Chau. An efficient three-step search algorithm for block motion estimation. *IEEE Transactions on Multimedia*, 6:435 – 438, 2004.

[81] B. H. Juang and A. H. Gray. Multiple stage vector quantization for speech coding. In *Proc. of the IEEE international conference on Acoustic, Speech, and Signal Processing*, volume 1, pages 597 – 600, 1982.

[82] M. Kamel, C. T. Sun, and L. Guan. Image compression by variable block truncation coding with optimal threshold. *IEEE Transactions on Signal Processing*, 39:208 – 212, 1991.

[83] N. A. Khan, S. Masud, and a. Ahmad. A variable block size motion estimation algorithm for real-time h.264 video coding. *Signal Processing: Image Communication*, 21:306 – 315, 2006.

[84] W. H. Kim, Y. H. Hu, and T. Q. Nguyan. Wavelet-based image coder with entropy-constrained lattice vector quantizer (eclvq). *IEEE Transactions on Circuit System-II: Analog and Digital Signal Processing*, 45:1015 – 1030, 1998.

[85] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro. Motion compensated interframe coding for video conferencing. In *Proc. Nat. Telecommun. conference*, pages G.5.3.1–G.5.3.5, 1981.

[86] K. Komatsu and K. Sezaki. Reversible transform coding of images. *Proc. of the SPIE Visual Communications and Image Processing*, 2727:1094 – 1103, 1996.

[87] C. H. Kuo and C. F. Chen. Nearly optimum multilevel block truncation coding based on a mean absolute error criterion. *IEEE Signal Processing Letters*, 3:269 – 271, 1996.

[88] C. H. Kuo, C. F. Chen, and W. Hsia. A compression algorithm based on classified interpolative block truncation coding and vector quantization. *Journal of Information Science and Engineering*, 15:1–9, 1999.

[89] T. Kurtia and N. Otsu. A method of block truncation coding for color image compression. *IEEE Transactions on Communications*, 41:1270 – 1274, 1993.

[90] G. Langdon and J. Rissanen. Compression of black white images with arithmetic coding. *IEEE Transactions on Communications*, 29:858 – 867, 1981.

[91] K. H. Lee, J. H. Choi, B. K. Lee, and D. G. Kim. Fast two-step half-pixel accuracy motion vector prediction. *Electronics Letters*, 36:625 – 627, 2000.

[92] Y. G. Lee, J. H. Lee, and J. B. Ra. Fast half-pixel motion estimation based on directional search and a linear model. *Visual Communications and Image Processing*, Proc. of the SPIE 5150:1513 – 1520, 2003.

[93] M. D. Lema and O. R. Mitchell. Absolute moment block truncation coding and its application to color images. *IEEE Transactions on Communications*, 32:1148 – 1157, 1984.

[94] R. Leonardi and J. R. Ohm. Wavelet video coding - an overview (iso/iec jtc1/sc29/wg11 w7824). 2006.

[95] R. Li, B. Zeng, and M. Liou. A new three-step search algorithm for block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 4:438–442, 1994.

[96] S. Z. Li and A. K. Jain. *Handbook of Face Recognition*. Springer,New York, 2005.

[97] W. Li and E. Salari. Successive elimination algorithm for motion estimation. *IEEE Transactions on Image Processing*, 4:105 – 107, 1995.

[98] C. Lim, H. S. Kang, T. Y. Kim, and K. Y. Yoo. A fast full search algorithm for variable block based motion estimation of h.264. In *ISVC 2005*, pages 710–717. LNCS 3805, 2005.

[99] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28:84 – 95, 1980.

[100] L. K. Liu and E. Feig. A block-based gradient descent search algorithm for block motion estimation in video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 6:419–422, 1996.

[101] C. Loeffler, A. Lightenberg, and G. S. Moschytz. Practical fast 1-d dct algorithms with 11-multiplications. *Proc. of the IEEE international conference on Acoustic, Speech, and Signal Processing*, pages 988 – 991, May, 1989.

[102] S. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press , New York, 1998.

[103] H. S. Malvar, A. Hallapuro, M. Harczewicz, and L. Kerofsky. Low-complexity transformation and quantization in h.264/avc. *IEEE Transactions on Circuits and Systems for Video Technology*, 13:598 – 603, 2003.

[104] D. Marpe and H. L. Cycon. Very low bit rate video coding using wavelet-based techniques. *IEEE Transcations on Circuits and Systems for Video Technology*, 9:85 – 94, 1999.

[105] J. Mielikainen. A novel full-search vector quantization algorithm based on the law of cosines. *IEEE Signal Processing Letters*, 9:175 – 176, 2002.

[106] H. B. Mitchell, N. Zilverberg, and M. Avraham. A comparison of different block truncation coding algorithms for image compression. *Signal Processing : Image Communication*, 6:77 – 82, 1994.

[107] O. R. Mitchell and E. J. Delp. Multilevel graphics representation using block truncation coding. *Proceedings of the IEEE*, 68:868–873, 1980.

[108] A. Moffat. Two-level context based compression of binary images. In *Proc. of the Data Compression*, pages 382 – 391. IEEE Computer Society Press, 1991.

[109] I. Mor, Y. Swissa, and H. B. Mitchell. A fast nearly optimum equi-spaced 3-level block truncation coding algorithm. *Signal Processing: Image Communication*, 6:397 – 404, 1994.

[110] T. Nakachi, T. Fujii, and J. Suzuki. Lossless and near-lossless compression of still color images. *Proc. of the international conference on Image Processing*, 1:453 – 457, 1999.

[111] P. Nasiopoulos, R. K. Ward, and D. J. Morse. Adaptive compression coding. *IEEE Transactions on Communications*, 39:1245 – 1254, 1991.

[112] M. N. Nasrabadi and R. A. King. Image coding using vector quantization: A review. *IEEE Transactions on Communications*, 36:957 – 971, 1988.

[113] N. M. Nasrabadi. Use of vector quantizer in image coding. In *Proc. of the IEEE international conference on Acoustic, Speech, and Signal Processing*, pages 125 – 128, 1985.

[114] N. M. Nasrabadi and R. A. King. Image coding using vector quantization: a review. *IEEE Transactions on Communications*, 36:957 – 971, 1988.

[115] G. E. Oien and S. Lepsoy. Fractal-based image coding with fast decoder convergence. *Signal Processing*, 40:105 – 117, 1994.

[116] S. I. Olsen. Block truncation and planar image coding. *Pattern Recognition Letters*, 21:1141–1148, 2000.

[117] Z. Pan, K. Kotani, and T. Ohmi. An improved full-search-equivalent vector quantization method using the low of cosines. *IEEE Signal Processing Letters*, 11:247 – 250, 2004.

[118] H. Park and V. K. Prasana. Modular vlsi architectures for real-time full-search-based vector quantization. *IEEE Transactions on Circuits and Systems for Video Technology*, 3:309 – 317, 1993.

[119] G. Pavlidis, A. Tsompanopoulos, N. Papamarkos, and C. Chamzas. A multi-segment image coding and transmission scheme. *Signal Processing*, 85:1827 – 1844, 2005.

[120] W. B. Pennebaker and J. L. Mitchell. *JPEG: still image data compression standard*. New York: Van Nostrand Reinhold, 1993.

[121] D. Phillips. Lzw data compression. *The Computer Application Journal, Circuit Cellar Inc.*, 27:36 – 48, 1992.

[122] L. M. Po and C. K. Cheung. A new center-based orthogonal search algorithm for fast motion estimation. *Proc. of the IEEE TENCON Digital Signal Processing Applications*, 2:874–877, 1996.

[123] L. M. Po and W. C. Ma. A novel four-step search algorithm for fast block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 6:313–317, 1996.

[124] W. Pratt. *Digital Image Processing*. John Wiley and Sons, New York, 1978.

[125] A. Puri, X. Chen, and A. Luthra. Video coding using the h.264/mpeg-4 avc compression standard. *Signal Processing: Image Communication*, 19:793 – 849, 2004.

[126] G. Qiu. A progressively predictive image pyramid for efficient lossless coding. *IEEE Transactions on Image Processing*, 8:109 – 115, 1999.

[127] G. Qiu. Color image index using btc. *IEEE Transactions on Image Processing*, 12:93 – 101, 2003.

[128] G. Qiu, M. R. Verley, and T. J. Terrell. Improved block truncation coding using hopfield neural network. *Electronics Letters*, 27:1924 – 1926, 1991.

[129] B. Ramamurthi and A. Gersho. Image vector quantization with a perceptually-based cell classifier. In *Proc. of the international conference on Acoustic, Speech, and Signal Processing*, volume 9, pages 698 – 701, 1984.

[130] K. R. Rao and P. C. Yip Eds. *The Transform and Data Compression Handbook*. CRC Press , Boca Raton, FL, 2001.

[131] E. A. Riskin, R. Lander, R. Y. Wang, and L. E. Atlas. Index assignment for progressive transmission of full-search vector quantization. *IEEE Transactions on Image Processing*, 3:307 – 312, 1994.

[132] A. Rosenfeld and A. C. Kak. *Digital Picture Processing, 2nd ed., Chapter 9*. Academic Press , New York, 1982.

[133] J. U. Roy and N. M. Nasrabadi. Hierarchical block truncation coding. *Optical Engineering*, 30:551 – 556, 1991.

[134] A. Said and W. A. Pearlman. Spiht image compression programs. *http://www.cipr.rpi.edu/research/SPIHT/spiht3.html*.

[135] A. Said and W. A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and System for Video Techniques*, 6:243– 250, 1996.

[136] D. Salomon. *Data Compression: The Complete Reference*. Springer-Verlag, New York, 2000.

[137] H. Samet. *Application of spatial data structures: Computer Graphics, Image Processing and GIS*. Addison Wesley, Reading , MA, 1990.

[138] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41:3445 – 3462, 1993.

[139] Y. Q. Shi and H. Sun. *Image and video compression for multimedia engineering: fundamentals, algorithms, and standards*. CRC Press , USA, 1999.

[140] A. Skodras, A. Christopoulos, and T. Ebrahimi. Jpeg2000: the upcoming still image compression standard. *Pattern Recognition Letters*, 22:1337–1345, 2001.

[141] Standardization Sector of ITU . Video coding for low bitrate communication. Draft ITU-T Rec. H.263 Mar. 1996.

[142] J. A. Storer and H. Helfgott. Lossless image compression by block matching. *The Computer Journal*, 40:137 – 145, 1997.

[143] J. W. Suh and J. Jeong. Fast sub-pixel motion estimation techniques having lower computational complexity. *IEEE Transactions on Consumer Electronics*, 50:968 – 973, 2004.

[144] K. Suhring. H.264/avc reference software. *http://iphome.hhi.de/suehring/tml/*.

[145] C. C. Sung, S. J. Ruan, B. Y. Lin, and M. C. Shie. Quality and power efficient architecture for the discrete cosine transform. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science*, E88-A:3500 – 3507, 2005.

[146] D. Tabuman. Kakadu software - a comprehensive framework for jpeg2000. *http://www.kakadusoftware.com/*.

[147] D. Tabuman and M. W. Marcelin. *JPEG 2000: image compression fundamentals, standards and practices*. Kluwer Academic Publishers, 2001.

[148] S. C. Tai, Y. C. Lin, and J. F. Lin. Single bit-map block truncation coding of color image using a hopfield neural network. *Information Sciences: an International Journal*, 103:211 – 228, 1997.

[149] P. Y. Tasi, Y. C. Hu, and C. C. Chang. A progressive secret reveal system based on spiht image transmission. *Signal Processing: Image Communication*, 19:285 – 297, 2004.

[150] J. Y. Tham, S. Ranganath, M. Ranganath, and A. A. Kassim. A novel unrestricted center-based diamond search algorithm for block motion estimation.

IEEE Transactions on Circuits and Systems for Video Technology, 8:369 – 377, 1998.

[151] L. Thomas and F. Deravi. Region-based fractal image compression using heuristic search. *IEEE Transactions on Image Processing*, 4:832 – 838, 1995.

[152] A. Toet and M. P. Lucassen. A new universal colour image fidelity metric. *Displays*, 24:197 – 207, 2003.

[153] J. T. Tou and R. C. Gonzalez. *Pattern Recognition Principles*. Addison-Wesely , London, 1974.

[154] C. L. Tung, T. S. Chen, W. H. A. Wang, and S. T. Yeh. A new improvement of spiht progressive image transmission. In *Proc. of the* 5^{th} *IEEE international symposium on multimedia software engineering*, pages 180–187, 2003.

[155] K. H. Tzou. Progressive image transmission: a review and comparison of techniques. *Optical Engineering*, 26:581–589, 1987.

[156] V. Udpikar and J. Raina. Btc image coding using vector quantization. *IEEE Transactions on Communications*, 35:352 – 356, 1987.

[157] V. R. Udpikar and J. P. Raina. A modified algorithm for block truncation coding of monochrome images. *Electronics Letter*, 21:900 – 902, 1985.

[158] T. J. Uhl. Adaptive picture data compression using block truncation coding. *Signal Processing II: Theories and Applications*, 1983.

[159] G. K. Wallace. The jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38:xviii – xxxiv, 1992.

[160] H. S. Wang and R. M. Mersereau. Fast algorithms for estimation of motion vectors. *IEEE Transactions on Image Processing*, 8:435 – 438, 1999.

[161] L. Wang and M. Goldberg. Lossless progressive image transmission by residual error vector quantization. *IEE Proceedings*, 135:421 – 430, 1988.

[162] Y. K. Wang and G. F. Tu. Block truncation coding with adaptive decimation and interpolation. In *Proc. SPIE international conference on Visual Communication and Image Prrocessing*, volume 4067, pages 430–437, 2000.

[163] Z. Wang and A. C. Bovik. A universal image quality index. *IEEE Signal Processing Letters*, 9:81 – 84, 2002.

[164] Z. Wang, A. C. Bovik, and L Lu. Why is image quality assessment so difficult? In *Proc. of the IEEE international conference on Acoustic, Speech, and Signal Processing*, pages 3313 – 3316, 2002.

[165] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13:600 – 612, 2004.

[166] A. Weitzman and H. B. Mitchell. An adaptive btc-vq image compression algorithm operating at a fixed bit-rate. *Signal Processing : Image Communication*, 5:287 – 294, 1993.

[167] T. Welch. A technique for high-performance data compression. *IEEE Computer*, 17:8 – 19, 1984.

[168] R. Westwater and B. Furht ND B. Furht. *Real-time video compression: Techniques and Algorithms*. Kluwer Academic , Norwall, MA, 1997.

[169] I. H. Witten, M. N. Radford, and J. G. Cleary. Arithmetic coding for data compression. *Communications of ACM*, 60:520 – 540, 1987.

[170] X. Wu. Context-based adaptive lossless image coder (huffman coding). *ftp://ftp.csd.uwo.ca/pub/from_wu/v.huff/*.

[171] X. Wu and N. Memon. Context-based, adaptive, lossless image coding. *IEEE Transactions on Communications*, 45:437–444, 1997.

[172] Y. Wu and D. C. Coll. Btc-vq-dct hybrid coding of digital images. *IEEE Transactions on Communications*, 39:1283 – 1287, 1991.

[173] Y. Wu and D. C. Coll. Multilevel block truncation coding using minimax error criterion for high-fidelity compression of digital images. *IEEE Transactions on Communications*, 41:1179 – 1191, 1993.

[174] Y. G. Wu. Block truncation image bit plane coding. *Optical Engineering*, 41:2476 – 2478, 2002.

[175] C. K. Yang, J. C. Lin, and W. H. Tsai. Color image compression by moment-preserving and block truncation coding techniques. *IEEE Transactions on Communications*, 45:1513 – 1516, 1997.

[176] C. K. Yang and W. H. Tasi. Improving block truncation coding by line and edge information and adaptive bit plane selection for gray-scale image compression. *Pattern Recognition Letters*, 16:67 – 75, 1995.

[177] B. Zeng. Two interpolative btc image coding schemes. *IEEE Transactions on Communications*, 27:1126 – 1128, 1991.

[178] B. Zeng and Y. Neuvo. Interpolative btc image coding with vector quantization. *IEEE Transactions on Communications*, 41:1436 – 1438, 1993.

[179] C. Zhu, X. Lin, L. Chau, and L. M Po. Enhance hexagonal search for fast block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 14:1210 – 1214, 2004.

[180] C. Zhu, X. Lin, and L. P. Chau. Hexagon-based search pattern for fast block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 12:349 – 355, 2002.

[181] S. Zhu and K. K. Ma. A new diamond search algorithm for fast block-matching motion estimation. *IEEE Transactions on Image Processing*, 9:287 – 290, 2000.

[182] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23:337 – 343, 1977.

List of Publications Related to this Thesis

- B. C. Dhara, and B. Chanda. Image Compression Algorithm Based on BTC and Pattern Fitting Method. In *Proc. of the* 5^{th} *International Conference on Advances in Pattern Recognition*, pages 299–302, 2003.

- B. C. Dhara, and B. Chanda. Block truncation coding using pattern fitting. *Pattern Recognition*, 37:2131–2139, 2004.

- B. C. Dhara, and B. Chanda. A Modified BTC using quincunx subsampling and pattern fitting for very low bpp. In *Proc. of the* 4^{th} *Indian Conference on Computer Vision, Graphics and Image Processing*, pages 563–568, 2004.

- B. C. Dhara, and B. Chanda. Block Motion Estimation Using Predicted Partial Octagonal Search. In *the Proc. of the IET International conference on Visual Information Engineering (VIE 2006)*, pages 277–282, 2006.

- B. C. Dhara, and B. Chanda. Progressive Transmission Scheme for Color Images using BTC-PF method. In *Proc. of the* 6^{th} *International Conference on Advances in Pattern Recognition*, pages 180–185, 2007.

- B. C. Dhara, and B. Chanda. Color Image Compression Based on Block Truncation Coding using Pattern Fitting Principle. *Pattern Recognition*, 40:2408–2417, 2007.

- B. C. Dhara, and B. Chanda. A lossless progressive image transmission scheme using BTC-PF and CALIC. *Communicated to Pattern Recognition*,2007.

- B. C. Dhara, and B. Chanda. A video coding technique using octagonal motion search and BTC-PF method for fast reconstruction. *Communicated to Signal Image and Video Processing*, 2007.

Addendum

to

Image and Video Compression using Block Truncation Coding and Pattern Fitting for Fast Decoding

Bibhas Chandra Dhara

Department of Information Technology

Jadavpur University, Salt Lake Campus

Kolkata 700098

India

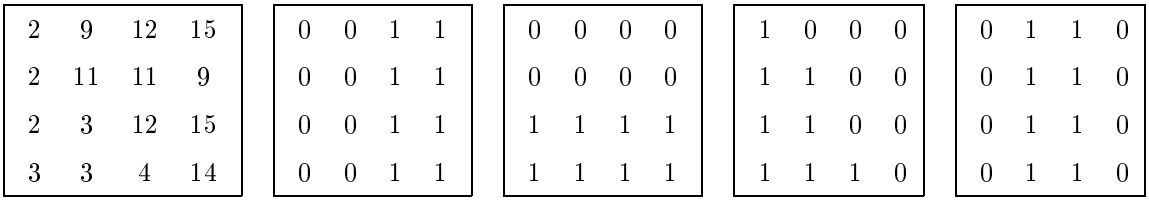
Addendum to "Image and Video Compression using Block Truncation Coding and Pattern Fitting for Fast Decoding"

(i) The following text is to be read at the end of subsection 2.4.1, i.e., after line 12 of first paragraph in page 37.

The pattern selection process is illustrated with an example as shown in Fig. 2.5. Let us consider an image block of size 4×4 (i.e., n=4) as shown in Fig. 2.5(a) and, say, eight binary patterns, $P_1, P_2, \ldots, P_8$, (i.e., Q=2 and M=8 as shown in Figs. 2.5(b)-(i)). These patterns are used to encode the given block. As a result of fitting the first pattern (P_1, Fig. 2.5(b)), the μ values are μ_0=4.375 and μ_1=11.50, respectively and corresponding errors e_{10} and e_{11} are 87.875 and 94.00, respectively. The total error, e_1, is 181.875. Similarly, the total errors due to the other patterns (P_2,P_3,...,P_8) are e_2=370.875, e_3=104.375, e_4=384.375, e_5=374.375, e_6=384.875, e_7=374.375 and e_8=374.375, respectively. Thus, the best pattern to represent the given block is P_3 (i.e., I=3), which incurs the minimum error 104.375.

(ii) The following text is to be read at the end of subsection 2.4.2, i.e., after the last line of page 37.

The quantization process is described using the same image block and the same patterns as given in Fig. 2.5. In the example, number of level is 2 and the best pattern selected for the given image block is P_3. To preserve first and second order moments Eqs. (2.10) and (2.11) are used and the corresponding quantization levels are a=3 and b=13. Please note that the complement of P_3 (i.e., '0' is replaced by '1' and '1' by '0') could also be a best pattern (as error with respect to P_3 and its complement is same) and then also a= 3 and b=13. As an alternative, if the mean values $(\mu_{3t} | t \in 1, 2, \ldots, Q)$ obtained in course of fitting the pattern P_3 (the best pattern) are used as the reconstruction levels (Eq. 2.22) then the reconstruction levels are 4 and 12. The quantization levels are obtained after the round-off process. The reconstructed blocks according to different quantization techniques are shown in Fig. 2.6 and the error due to respective reconstruction levels is also reported.



(a) Original block

2	9	12	15
2	11	11	9
2	3	12	15
3	3	4	14

(b) P_1, μ_0=4.375, μ_1=11.5, e_1=181.875

0	0	1	1
0	0	1	1
0	0	1	1
0	0	1	1

(c) P_2, μ_0=8.875, μ_1=7.0, e_2=370.875

0	0	0	0
0	0	0	0
1	1	1	1
1	1	1	1

(d) P_3, μ_0=3.75, μ_1=12.125, e_3=104.375

1	0	0	0
1	1	0	0
1	1	0	0
1	1	1	0

(e) P_4, μ_0=7.75 μ_1=8.125, e_4=384.375

0	1	1	0
0	1	1	0
0	1	1	0
0	1	1	0

(f) P_5, μ_0=7.125, μ_1=8.75, e_5=374.375

0	0	1	1
0	1	1	0
0	1	1	0
1	1	0	0

(g) P_6, μ_0=8.0, μ_1=7.875, e_6=384.875

1	1	1	1
1	1	1	0
1	0	0	0
0	0	0	0

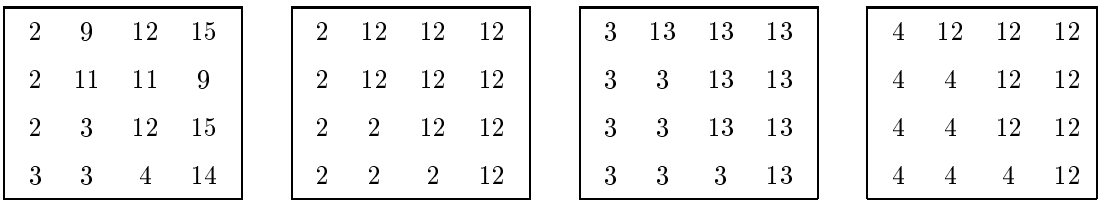
(h) P_7, μ_0=7.75, μ_1=8.125, e_7=384.375

0	0	0	0
1	1	1	1
1	1	1	1
0	0	0	0

(i) P_8, μ_0=7.125, μ_1=8.75, e_8=374.375

1	0	0	0
1	1	1	0
0	1	1	1
0	0	0	1

Figure 2.5: An illustration of the pattern selection process. (a) Original image block, (b)-(i) the set of binary patterns $\{P_1, P_2,...,P_8\}$. The values of the mean (μ) and the error (e) due to fitting the patterns with the block (a) are given below the corresponding pattern. P_3 is selected as the representative pattern for the given block as it incurs minimum error.



(a)

2	9	12	15
2	11	11	9
2	3	12	15
3	3	4	14

(b)

2	12	12	12
2	12	12	12
2	2	12	12
2	2	2	12

(c)

3	13	13	13
3	3	13	13
3	3	13	13
3	3	3	13

(d)

4	12	12	12
4	4	12	12
4	4	12	12
4	4	4	12

Figure 2.6: (a) Original block, (b)-(d) reconstructed blocks: (b) by conventional BTC method, error=49, (c) by fitting P_3 and using Eqs. (2.10) and (2.11), error=115, (d) by fitting P_3 and using Eq. (2.22), error=105.

145	055	048	088
116	101	039	067
076	114	046	046
070	135	095	048

(a)

137	090	061	033
090	054	065	045
097	059	059	047
082	120	053	046

(b)

070	076	091	146
092	060	061	085
108	106	050	058
071	091	055	054

(c)

153	107	074	146
109	043	040	109
072	037	029	051
052	033	033	028

(d)

182	120	080	092
147	126	119	040
117	104	098	058
041	120	088	033

(e)

073	090	102	066
049	054	069	065
072	052	043	066
048	038	047	056

(f)

071	119	119	070
118	168	180	100
089	173	199	175
056	169	193	186

(g)

056	145	173	066
052	082	166	158
087	048	092	163
115	048	077	099

(h)

(A) Set of original image blocks

1.89	-0.75	-0.95	0.21
1.04	0.60	-1.22	-0.39
-0.13	0.98	-1.01	-1.01
-0.31	1.59	0.42	-0.95

(a)

2.26	0.65	-0.34	-1.31
0.65	-0.59	-0.21	-0.90
0.89	-0.41	-0.41	-0.83
0.37	1.68	-0.62	-0.86

(b)

-0.37	-0.14	0.44	2.58
0.48	-0.76	-0.72	0.20
1.10	1.02	-1.15	-0.84
-0.33	0.44	-0.96	-0.99

(c)

1.97	0.88	0.10	1.80
0.93	-0.63	-0.70	0.93
0.05	-0.77	-0.96	-0.44
-0.42	-0.87	-0.87	-0.98

(d)

2.06	0.54	-0.43	-0.14
1.20	0.69	0.51	-1.41
0.47	0.15	0.00	-0.97
-1.39	0.54	-0.24	-1.58

(e)

0.64	1.63	2.33	0.24
-0.74	-0.45	0.41	0.18
0.58	-0.57	-1.09	0.24
-0.80	-1.39	-0.86	-0.34

(f)

-1.33	-0.35	-0.35	-1.35
-0.37	0.64	0.88	-0.74
-0.97	0.74	1.27	0.78
-1.64	0.66	1.15	1.00

(g)

-1.00	0.95	1.56	-0.78
-1.09	-0.43	1.41	1.23
-0.32	-1.17	-0.21	1.34
0.29	-1.17	-0.54	-0.05

(h)

(B) Set of normalized blocks

Figure 2.7: The set of blocks used to design patternbook.

1.56	0.23	-0.23	0.63
0.86	-0.13	-0.46	-0.31
0.47	0.19	-0.71	-0.82
-0.41	0.67	-0.45	-1.08

-0.56	0.74	1.18	-0.63
-0.73	-0.08	0.90	0.22
-0.23	-0.33	-0.01	0.79
-0.71	-0.63	-0.08	0.20

(a) Class prototypes (by K-means)

1	1	0	1
1	0	0	0
1	1	0	0
0	1	0	0

0	1	1	0
0	0	1	1
0	0	0	1
0	0	0	1

(b) Binary patterns obtained by thresholding at 0

1	1	0	1
1	0	0	0
1	1	0	0
0	1	0	0

0	1	1	0
0	0	1	0
0	0	0	1
0	0	0	0

(c) Binary patterns obtained by optimal thresholding

2	1	0	1
1	0	0	0
1	1	0	0
0	1	0	0

0	2	2	0
0	1	2	1
1	0	1	2
0	0	1	1

(d) Three level patterns obtained by optimal thresholding

Figure 2.8: Class prototypes (by K-means on the blocks shown in Fig. 2.7(B)) and different patterns based on different thresholding techniques.

(iii) The following text is to be read at the end of subsection 2.4.3, i.e., after the last line of page 38.

The patternbook design process is illustrated using the following example. Here, we consider 8 image blocks of size 4×4 (see Fig. 2.7(A)). Then each of these blocks is normalized to obtain zero mean and unit variance block by simple operation, $B^{nor}(i,j) = \frac{B^{org}(i,j) - mean(B^{org})}{std.dev(B^{org})}$ where B^{org} and B^{nor} are original block and normalized block, respectively. The normalized data set is given in Fig. 2.7(B). K-means algorithm is applied on these data to obtain two clusters (i.e., K=2) and the resultant cluster-centers or the prototypes are shown in the Fig. 2.8(a). As the data set (i.e., image blocks) is normalized, zero mean prototypes are expected. So the binary pattern vectors may be obtained from these prototypes by theresholding at zero and the resultant patterns are shown in Fig. 2.8(b). The binary patterns generated from the same prototypes with respect to the optimal threshold are shown in Fig. 2.8(c). It may be noted that the binary pattern with respect to zero and that with respect to the optimal threshold may not be the same. The Figs. 2.8(b) and 2.8(c) reveal this situation. The three level patterns from the same prototypes obtained by optimal thresholding are given in Fig. 2.8(d).