ON ANALYSIS OF COMPRESSED VISUAL INFORMATION AND ITS APPLICATION

Thesis submitted for the degree of

Doctor of Philosophy (Technology)

in

Computer Science and Engineering

by

BHASKAR DEY

Department of Computer Science and Engineering

University of Calcutta

2017

To Lord Krishna, without whose divine providence, the thesis would never have made light of the day

Declaration of Authorship

This is to declare that the thesis entitled “On Analysis Of Compressed Visual Information And Its Application” submitted by Mr. Bhaskar Dey who got his name registered on 11.06.2014 for the award of Ph. D. (in Computer Science & Engineering) degree of University of Calcutta, is absolutely based upon his own work carried out at Center for Soft Computing Research, Indian Statistical Institute under the supervision of Prof. Malay Kumar Kundu, and that neither this thesis nor any part of it has been previously submitted for any degree/diploma or any other academic award anywhere.

……...

(BHASKAR DEY)

Acknowledgements

First and foremost, I would like to acknowledge the tireless and prompt help of my supervisor, Prof. Malay Kumar Kundu. I gratefully acknowledge his valuable conceptual suggestions during the research works. Whenever I struggled, his generous support always came at just the right time. Prof. Kundu has allowed me complete *freedom* to define and explore my own directions in research. While this proved a bit difficult and somewhat bewildering to begin with, I have come to appreciate the wisdom of his way—it encouraged me to think for myself, something that is unfortunately all too easy to avoid as an undergraduate student but absolutely necessary for a research scholar. It is my privilege to work under his supervision, being possessed of profound knowledge and humble personality.

I want to thank the senior faculty members of the Center for Soft Computing Research, Indian Statistical Institute (ISI) – Prof. S. K. Pal, Prof. A. Ghosh for their constructive suggestions and comments. I am also grateful to Dr. K. Ghosh, Dr. S. S. Ray, and Dr. Saurabh Das for useful discussions. I would never forget the company I had from my fellow colleagues, both current and former, for their helpful discussions and support specially, Mr. S. Kundu, Dr. A. Ganivada, Mrs. D. Bhunia (Chakraborty), Mr. J. K. Pal, Mr. S. Bandyopadhyay, Mr. S. J. Choudhury, Ms. S. Das, Ms. C. Chatterjee, Dr. M. Chowdhury, Dr. S. Das, and Mr. J. Mondal. I express my sincere thanks to all the office staffs of the Center for Soft-Computing Research, ISI, for providing me with all kinds of facilities whenever I needed.

I am also grateful to the members of the Ph.D. committee of the Department of Computer Science and Engineering, University of Calcutta. I thank my parents and relatives, who taught me the values of hard work and patience. I would like to share this moment of happiness with all of them.

……...

(BHASKAR DEY)

Author's Publications/Patents

- Journals

1. B. Dey and M. K. Kundu, "Robust background subtraction for network surveillance in H. 264 streaming video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 10, pp. 1695-1703, 2013.
2. B. Dey and M. K. Kundu, "Enhanced Macroblock Features for Dynamic Background Modeling in H.264/AVC Video Encoded at Low-Bitrate," *IEEE Transactions on Circuits and Systems for Video Technology*, (Accepted).
3. B. Dey and M.K. Kundu, "Efficient Foreground Extraction from HEVC Compressed Video for Application to Real-Time Analysis of Surveillance 'Big' Data," *IEEE Transactions on Image Processing*, vol. 24, no. 11, pp. 3574-3585, Nov. 2015.
4. B. Dey and M.K. Kundu, "Turning Video into Traffic Data - An Application to Urban Intersection Analysis Using Convolution Neural Network," *IET Image Processing*, (communicated).

- Patents Granted/filed

B. Dey and M. K. Kundu, "Foreground Motion Detection In Compressed Video Data"

i. India Patent Application #IN-591/KOL/2015, filed on May 27, 2015 (priority filing)
ii. US Patent Application #US-14/807,853, Patent #US 9,172,828 B2 granted on 18/07/2017
iii. PCT Patent Application #PCT/IB2016/052344, Publication #WO/2016/189404 on 01/12/2016 (counterpart filing)

Abstract

The computational approach to video analysis is important to a large number of applications in diverse disciplines, which includes traffic monitoring, visual tracking, surveillance, video-annotation and summarization, as well as actions and gesture recognition. The first stage of analysis originates from detecting changes due to object-motion/activity in the scene. Indeed, for many applications, the very fact that something is moving makes it of 'interest' while anything else can be ignored. In such cases, it is common for regions of interest (ROI) to be categorized as foreground while the remaining part of the scene as background. The simplest approach, commonly referred to as background subtraction or moving object segmentation, involves subtracting the current frame from a model of its background. It may be noted that digital images and video sequences today are both stored and transmitted in a compressed form. However, most stat-of-the-art methods are prohibitively complex as they operate in the spatial domain at the pixel level, i.e., in the un-compressed form. Hence, they require the entire video to be decoded to be able to process it. On the other hand, there are newly developed techniques in the compressed domain which reuse features that are already present in the encoded form. These methods, although are computationally efficient, suffer from segmentation accuracy. In this thesis, the related methodology, both on pixel and compressed domain are first explored. In this thesis, new methods for moving object segmentation from compressed-domain codecs, such as H.264 (Baseline and Main/High profiles) and HEVC codecs are presented. The methods exhibit high speed and robustness against static as well as dynamic backgrounds. Extensive comparisons with the state-of-the-art methods are carried out, to evaluate qualitatively as well as quantitatively, the performance of the proposed solutions. The aforementioned novel developments are suitable for inclusion as application in a video-based surveillance system. The proposed methods have been used, in conjunction with deep-learning architectures, in traffic intersection analysis of real-life data.

Contents

List of Figures

List of Tables

List of Abbreviations

AVC	Advanced Video Coding
ANN	Artificial Neural Network
BGS	Background Subtraction
CABAC	Context Adaptive Binary Arithmetic Coding
CAVLC	Context Adaptive Variable Length Coding
CB	Coding Block
CBR	Constant Bit Rate
CCTV	Closed-Circuit Television
CNN	Convolutional Neural Network
CQT	Coding Quadtree
CTB	Coded Tree Block
CTU	Coding Tree Unit
DCT	Discrete Cosine Transform
DST	Discrete Sine Transform
EKF	Extended Kalman Filter
FIFO	Fist-In First-Out
fps	Frames Per Second
GMM	Gaussian Mixture Model
GOP	Group of Picture
HEVC	High Efficiency Video Coding
ICA	Independent Component Analysis
IDCT	Inverse Discrete Cosine Transform
IEC	International Electrotechnical Commission
IIR	Infinite Impulse Response
IRT	Incremental Rank Tensor
ISO	International Organization for Standardization
JCT-VC	Joint Collaborative Team on Video Coding
KDE	Kernel Density Estimation
LoPP	Locality Preserving Projections
LPCA	Local Principle Component Analysis
LUT	Lookup Table
MAP-MRP	Maximum A Posteriori Markov Random Field
MATD	Mean of Absolute Transform Differences
McFIS	Most Common Frame of a Scene
MOG	Mixture of Gaussians
MOGG	Mixture of General Gaussians
MPEG	Moving Picture Experts Group
MRF	Markov Random Field
PB	Prediction Block
PCA	Principal Component Analysis
pdf	Probability Density Function

PSMF	Probabilistic Spatio-temporal Macroblock Filter
PU	Prediction Unit
RDO	Rate-Distortion Optimization
RGB	Red-Green-Blue
RQT	Residual Quadtree
ROI	Region of Interest
RTDENN	Real-Time Dynamic Ellipsoidal Neural Networks
SC-SOBS	Spatially Coherent Self-Organizing Background Subtraction
SGD	Stochastic Gradient Descent
SL-PCA	Subspace learning using Principal Component Analysis
SNMVM	Sum of Normalized Motion Vector Magnitudes
SOBS	Self-Organizing Background Subtraction
SONN	Self-Organizing Neural Network
ST-DBF	Spatio-Temporal De-correlated Block Features
STMF	Spatio-Temporal Median Filter
SVDD	Support Vector Data Description
SVM	Support Vector Machine
SVR	Support Vector Regression
TB	Transform Block
UKF	Unscented Kalman Filter
VBR	Variable Bit Rate
VCEG	Video Coding Experts Group
VIVDS	Video Image Vehicle Detection System

Chapter 1

Introduction and Scope of the Thesis

1.1 Introduction

The computational approach to video analysis is important for a large number of applications in diverse disciplines, which includes monitoring traffic and vehicles [1]-[3], tracking [4] - especially of people [5], animals [6], surveillance [7] systems which include airport, law enforcement, shopping malls etc., video-annotation [8], summarization [9], as well as recognition of human action [10], [11] and gestures [12], [13]. The *first stage of analysis* originates with detection of *motion* activities, i.e., identification of changes in the scene. Indeed, for many applications, the fact that something is moving makes it of 'interest', while the remaining part of the scene may be ignored. In such cases, it is customary for regions of interest (ROIs) to be categorized as the *foreground* while the remaining part of the scene as the *background*. The most common approach, called *background subtraction*, involves subtracting the current frame from a model of its background.

Moving object segmentation problem stage consists in taking an image, usually from a video sequence, and dividing it into two complimentary sets of pixels. The first set contains the pixels which correspond to foreground objects while the second or the complimentary set contains the background pixels [14]. It is difficult to specify an absolute standard with respect to what should be identified as foreground and what should be marked as background. This is because of the fact that the notion of foreground and background is somewhat application specific [15]. For the present thesis, any moving objects of interest is considered as the foreground and anything as the background. However, when there is a random motion in the background due to waving tree branches, ripples, fountain etc, it is considered as the background which is dynamic in nature, i.e., *dynamic background*. In general, algorithms that classify foreground from the background are called background subtraction (BGS) methods. Such methods typically use a model of the background and its pixel-wise difference from the current frame of the sequence. The pixels with significant difference are identified as foreground. The detail of such approach is discussed in the following section.

1.2 Background Subtraction: General overview

Fig 1.1 shows a typical framework of a BGS algorithm. The simplest form of BGS is to acquire a single representative frame and use it as the background model for future

processing. New input frames are compared directly to the background image and foreground pixels are classified by a deviation from the model:

$$mask(x,y) = \begin{cases} foreground & \text{if } diff(x,y) \geq threshold \\ background & \text{otherwise} \end{cases} \quad (1.1)$$

where *mask* is the output image, *diff* is the absolute difference between the current frame and the background frame, while (x, y) denotes the pixel intensity at a given spatial location.

As stated in [15], it is difficult to specify a 'gold-standard' definition of what a background subtraction algorithm should detect as foreground. This is because the notion of the foreground changes over variations of data and applications types. However, a majority of applications require foreground detection algorithms to address three key issues. Firstly, it should be robust against changes in illumination and avoid detecting shadows cast by moving objects. Secondly, it should also be robust in a situation where the background is under random motion, e.g. swaying trees, shimmering waves, fountain,

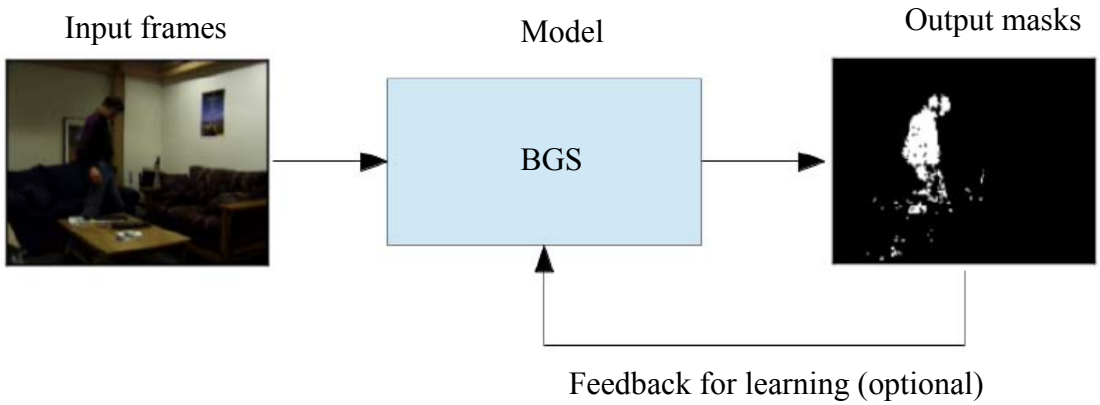


Fig. 1.1 A typical BGS System

camera jitter etc. Last, but by no means least, the foreground detection rate should be fast enough to support real-time implementation constraints.

Fig. 1.2 shows an overview of a typical background subtraction method (1) Background initialization using *N* frames to obtain the first background image without the moving objects. (2) Then, the moving object detection is made through the foreground detection that consists in classifying pixels as foreground or background by comparing the background image and the current frame. (3) Background maintenance to update the background image over time. The steps (2) and (3) are performed repeatedly as time

progresses. These actual steps are discussed in detail, in the following sub-sections.

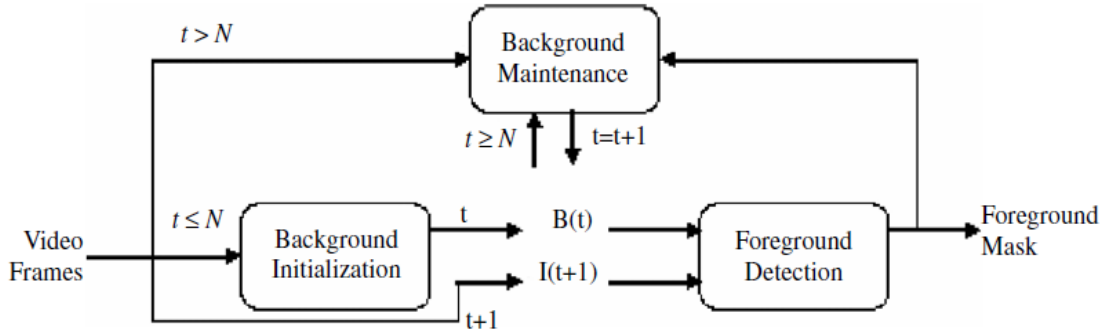


Fig. 1.2 Background Subtraction Process. N is the number of frames that is used for the background initialization. B_t and I_t are the background and the current image at time t, respectively.

1.2.1 Background Initialization

Background initialization refers to the initialization of the model. In the current literature, in contrast to background model representation and maintenance, the initialization of the background model was only marginally investigated. The main reason is that it is often assumed that initialization can be achieved by exploiting some clean frames at the beginning of the sequence. Naturally, this assumption is rarely encountered in real-life scenarios, because of continuous presence of noise, background clutter, illumination changes, etc. It is common for the background model to be initialized using the first frame or a background model over a set of training frames. The set of frames may or may not contain foreground objects. The main challenge here, is to obtain a background model from a set of training frames without containing foreground objects. Practically, some algorithms use: (1) batch mode training of (say N) frames (consecutive or otherwise) [14], (2) incremental training with known N frames, and (3) progressive training with an indefinite number of frames, which generates partial backgrounds and continues until a complete background image is obtained [15, 16]. Furthermore, initialization algorithms also depend on the number of modes and the complexity of their background models [17].

1.2.2 Background Subtraction

In this stage, the frame *diff*(x, y) is obtained by computing the absolute difference between the current frame I_t and the corresponding background frame B_t. As described in equation (1.1), the pixel at (x,y) in I_t is considered as foreground if *diff*(x, y) exceeds a desired threshold.

1.2.3 Background Maintenance

Background maintenance refers to the mechanism that adapts the background model to the changes in the scene occurring over time. This learning process has to be achieved online and so the algorithm has to be an incremental one. The important aspects of this step are the following:

Maintenance schemes: In the literature, three maintenance schemes are mentioned so far, *viz*, the blind, the selective, and the fuzzy adaptive schemes. The blind background-maintenance scheme updates all the pixels with the same rules which is usually an Infinite Impulse Response (IIR) filter:

$$B_{t+1}(x,y) = (1-\alpha)B_t(x,y) + \alpha I_t(x,y) \tag{1.2}$$

where α is the learning rate which is a constant in [0, 1]. B_t and I_t are the background and the current image at time t, respectively. The main disadvantage of the blind scheme is that the pixels classified as foreground are also used in the computation of the new background thereby generating errors in the background image. To solve this problem, some authors suggested to use a selective maintenance scheme that consists in updating the new background image with different learning rates depending on the classification of a pixel into foreground or background:

$$B_{t+1}(x,y) = (1-\alpha)B_t(x,y) + \alpha I_t(x,y) \quad \text{if (x,y) is background} \tag{1.3}$$

$$B_{t+1}(x,y) = (1-\beta)B_t(x,y) + \beta I_t(x,y) \quad \text{if (x,y) is foreground} \tag{1.4}$$

Here, the idea is to adapt a pixel classified as background very quickly and a pixel classified as foreground very slowly. For this reason, $\beta \ll \alpha$ and usually $\beta = 0$. So the Eq. (1.4) becomes:

$$B_{t+1}(x,y) = B_t(x,y) \tag{1.5}$$

But the problem is that an erroneous classification may result in an incorrect background model, which subsequently remains unchanged. This problem can be addressed by a fuzzy adaptive scheme [18] which takes into account the uncertainty of the classification. This can be achieved by graduating the update rule using the result of the foreground detection.

Learning rate: The learning rate determines the speed of the adaptation to the scene changes. It can be (1) fixed, or dynamically adjusted by (2) a statistical or (3) a fuzzy method [19–22]. It may be noted that the learning rate determines the speed of adaptation to illumination changes. The learning rate also deals with different challenges which have

different temporal characteristics. To decouple between the adaptation mechanism and the incorporation mechanism, some authors [23, 24] used a set of counter parameters which represents the number of times a pixel is classified as a foreground pixel. When this number is larger than a threshold, the pixel is classified as background. This gives a time limit on how long a pixel can be considered as a foreground pixel.

Frequency of the update: The aim of this step is to determine is to update the background only when needed. This may be done for every new frame but in absence of any significant changes, pixels are not required to be updated [25, 26].

1.2.4 Foreground detection

Foreground detection consists of labeling pixels as either as background or as foreground. This is a mainly classification task.

1.2.5 Choice of the picture's element

In general, the smallest unit of picture's element considered for moving object detection in video sequence is a pixel [27]. However, in certain applications the smallest unit may be a collection of pixels called block [28] or a cluster [29]. The size of the element determines the robustness to noise, and the precision. A pixel-based method gives a pixel-based precision but it is less robust to noise than block-based or cluster based methods.

1.2.6. Choice of the features

In the literature, the most commonly reported features [30] for background subtraction are color features, edge, stereo, motion and texture. These features may be classified as spectral features (color intensity), spatial features (edge, texture) and temporal features (motion). These features play important role which allow us to handle various difficult situations such as illumination changes, motion changes, structure background changes etc. Color features are very discriminative but they have several limitations in the event of illumination change, camouflage and shadow. Edge features handle the local illumination changes. Texture is used to adapt the background model to illumination changes and shadows. If more than one feature is used, the relative importance has to be determined while designing the operator [31, 32]. The operator may be of crisp operator type, statistical type or fuzzy type.

While developing a background model, one should select carefully the features in

order to handle the perturbation in the background [33], which may arise due to a poor-quality image source, camera jitter, camera automatic adjustments, time of the day, light switch, bootstrapping, camouflage, foreground aperture, moved background objects, inserted background objects, multi-modal background, waking foreground object, sleeping foreground object, shadows, etc. Other challenges may occur with changing environment due to illumination changes, dynamic backgrounds, etc.

1.3 Basic Models: Pixel-Based methods

A lot of literature has been directed to the issue of constructing background models. Most existing approaches operate in the spatial (or pixel) domain, and follow the same philosophy that a background model is independently estimated for each spatial location through a series of pixel values (gray or color) at temporal axis. These are **pixel-based** methods. The different background representation models can be classified in the following categories: basic models, statistical models, cluster models, neural network models and estimation models. A brief review of each categories of BGS methods is summarized in the following sections.

In the category of basic models case, the background is modeled using an average [34], a median [35] or a histogram analysis over time [36]. Once the model is computed, pixels of the current image are classified as foreground by thresholding the difference between the background image and the current frame as follows:

$$d(I_t(x,y), B_{t-1}(x,y)) > T \tag{1.6}$$

where T is a constant threshold, $I_t(x, y)$, $B_{t-1}(x,y)$ are the current image at time t and the background image at time $t-1$ respectively. Function $d(.,.)$ is a distance measure which is usually the absolute difference between the current and the background images.

1.4 Statistical Models

The statistical models offer more robustness to illumination changes and dynamic backgrounds. The statistical background models can be classified in the following categories, viz., Gaussian models, support vector models and subspace learning models.

Gaussian models: The simplest way to represent the background is to assume that the history over time of pixel's intensity values can be modeled by a Gaussian. Following this idea, Wren et al. [37] have proposed to use a single Gaussian distribution to model the

background image. Kim et al. [38] generalized the single Gaussian method using single general Gaussian to alleviate the constraint of a strict Gaussian. However, a unimodal model cannot handle dynamic backgrounds when there are waving trees, ripples, fountain etc. To solve this problem, the Mixture of Gaussians (MOG) or Gaussian Mixture Model (GMM) [17] has been used to model dynamic backgrounds. Many improvements of the MOG were developed which are more robust and adaptive to the critical situations. For example, Porikli and Tuzel [39] defined each pixel as layers of 3D multivariate Gaussian distribution. Each layer corresponds to a different appearance of the pixel. Using a Bayesian approach, the probability distributions of mean and variance are estimated instead of the mean and variance themselves. In an improvement proposed by Alvar et al. [40], a Real-Time Dynamic Ellipsoidal Neural Network (RTDENN) was used to improve the background maintenance in case of MOG. Later, Allili et al. [41] proposed the mixture of general Gaussians (MOGG) to alleviate the constraint of a strict Gaussian. However, the MOG and MOGG present several disadvantages. For example, a background having fast variations cannot be accurately modeled with just a few Gaussians (usually 3 to 5), which causes problems for sensitive detection. So, a non-parametric technique [42] was proposed to estimate background probabilities at each pixel from many recent samples over time using Kernel density estimation (KDE). This method, however, is very time consuming. Hence, several improvements of the KDE method were proposed. For example, Sheikh and Shah [43] modeled the background using a KDE method over a joint domain-range representation of image pixels. In [44], The background and foreground models are used competitively in an MAP-MRF (Maximum A Posteriori Markov Random Field) decision framework. Furthermore, a sequential KDE algorithm is also presented in [45].

Support vector models: The second category uses more sophisticated statistical models such as support vector machine (SVM) [46], support vector regression (SVR) [47] and support-vector data description (SVDD) [48]. Lin et al. [46, 49] proposed a method to initialize the background model using a probabilistic SVM. In this method, SVM classification is applied for all pixels of each training frame by computing the output probabilities. Based on these results, newly found pixels are evaluated and determined if they should be incorporated into the background model. The background initialization continues until there are no more new background pixels to be considered. The commonly used features are the optical flow value and inter-frame difference. In a similar way, Wang

et al. [50, 51] used a separate SVR to model each background pixel as a function of intensity. The background initialization is made using a batch algorithm, and background maintenance is achieved with an online SVR algorithm [53]. Tavakkoli et al. [48, 52] proposed to label pixels in into foreground and background classes using SVDD. Unlike parametric and non-parametric density estimation techniques, the background model is not based on the probability function of the background / foreground. Indeed, it is an analytical description of the decision boundary between background and foreground classes. So, the model accuracy is not bounded to the accuracy of the estimated probability density functions. Therefore, the memory requirements are less than those of non-parametric techniques. This is because in those methods, pixel values for all background training frames is required to be stored to regenerate the probability of pixels in new frames. For the initialization, the process is made offline while maintenance is done using an on-line method. For the foreground detection, pixels are only compared with the support vectors, which are practically much fewer than the number of frames in the temporal window. Furthermore, SVDD explicitly models the decision boundary of the known class. So, this results in fewer parameter tunings and automatic classification. The background initialization was improved by a genetic approach [54], and the background maintenance is made by an incremental SVDD algorithm [55, 56].

Subspace learning models: The third category employs subspace learning methods. Subspace learning using Principal Component Analysis (SL-PCA) [57] is applied on a set of known images to construct a background model, which is represented by the mean image. This is obtained from the projection matrix comprising the first few significant eigenvectors of Principal Component Analysis (PCA). In this way, foreground segmentation is accomplished by computing the difference between the input image and its reconstruction. However, this model presents several limitations: (1) The size of the foreground objects must be small, and they should not appear in the same location during a long period in the training sequence [58-60]; (2) For the background maintenance; it is computationally intensive to perform model updating using the batch mode PCA. Moreover, without a mechanism of robust analysis, the outliers or foreground objects may be incorporated erroneously into the background model. Some robust incremental mechanisms have been developed in [61-63]; (3) The application of this model is mostly limited to gray-scale images since the integration of multi-channel data is not straightforward. It involves a much higher-dimensional space and generally causes

additional difficulty to manage the data space. Recently, Han and Jain [64] proposed an efficient algorithm using a weighted incremental 2-Dimensional Principal Component Analysis. The proposed algorithm was applied to 3-channel (Red-Green-Blue, i.e. RGB) and 4-channel (RGB + infra-red) data. Results show noticeable improvements in the presence of multimodal background and shadows; (4) lastly, the background model representation in PCA based methods is not multi-modal so various illumination changes cannot be handled correctly.

Recently, Dong et al. [65] proposed to use a multi-space learning method to handle different illumination changes. The feature space is organized into clusters, which represent different lighting conditions. A Local Principle Component Analysis (LPCA) method is used to learn separately eigen-subspaces for each cluster. When a new image arrives, the algorithm selects the learned subspace which shares the nearest lighting condition. The result of [65] shows that the LPCA algorithm outperforms the original PCA, especially under the condition of sudden illumination change. Recently, other reconstructive subspace models were also used. Yamazaki et al. [66] and Tsai et al. [67] used Independent Component Analysis (ICA) for background modelling. In another method, Bucak et al. [68] proposed an incremental non-negative matrix factorization to reduce the dimension. In order to incorporate the spatial information, Li et al. [69] used an Incremental Rank-(R1, R2, R3) Tensor (IRT). Recently, Krishna et al. [70] used the Locality Preserving Projections (LoPP) is a classical linear technique that projects the data along the directions of maximal variance. It also preserves the neighborhood structure of the observation matrix. Furthermore, LoPP shares many of the data representation properties of nonlinear techniques, which are interesting to separate background and foreground.

1.5 Cluster Models

Cluster models suppose that each pixel in the frame can be temporally represented by clusters. The clustering approaches are mainly K-means algorithms [71], Codebooks [72] or basic sequential clustering methods [73].

K-means models: Butler et al. [71] proposed an algorithm that assigns a group of clusters to each pixel in the frame. The background initialization is achieved off-line. The clusters are ordered according to the likelihood of the background model, which are adapted

according to lighting variations. Incoming pixels are matched against the corresponding cluster group and are classified according to whether the matching cluster is considered part of the background or not. To improve the robustness, Duan et al. [74] proposed to use a genetic K-means algorithm. The idea is to alleviate the disadvantages of the traditional K-means algorithm, which has random and locality aspects causing lack of global optimization.

Codebook models: Kim et al. [72] proposed to model the background using a codebook model. For each pixel, a codebook is constructed which consists of one or more codewords. The samples at each pixel are clustered into a set of codewords based on a color distortion metric and brightness bounds. The clusters represented by codewords do not necessarily correspond to single Gaussian or other parametric distributions. Instead, the background is encoded on a pixel-by-pixel basis. Detection involves testing the difference of codewords of the current image and the background model. An incoming pixel is classified as background if (1) the color distortion to some codewords is less than the detection threshold, and (2) its brightness lies within the brightness range of that codeword. Otherwise, it is classified as foreground. This original algorithm has been improved in several ways. For example, Kim et al. [75] presented a modified algorithm which has layered modeling/detection and adaptive codebook updating. Sigari and Fatih [76] proposed a two-layer codebook model. The first layer is the main codebook; the second one is the cache codebook. Both layer contain some codewords related to a same pixel. Main codebook models the current background images and cache codebook. is used to model new background images during input sequence. In order to be robust to illumination changes, other improvements concerned the color models which can be used instead of the cone cylinder model such as the hybrid-cone cylinder model in [77], and the spherical model in [78]. Other modifications in this regard are block based approach [79, 80], hierarchical approach [81] or multi-scale approach [82] to reach real-time requirements.

Basic sequential clustering: Another clustering approach was developed in [83]. It is based on the assumption that the foreground objects would appear throughout the sequence. Initially, pixel intensities are classified based with an on-line clustering model. Then, cluster centers and appearance probabilities of each cluster are calculated. Finally, a single or multi-intensity cluster with the appearance probability greater than a threshold is identified as the background pixel. A modified version proposed in [84] adds a second

step as a merging procedure to classify classes. When a cluster deviates and becomes close to one another, the two clusters are fused in one cluster. Xiao and Zhang [85] improved this approach with a two-threshold sequential clustering algorithm. The avoidance of quick deviation of clusters to themselves is improved by the second threshold based on the creation of a new cluster. Recently, Benalia and Ait-Aoudia [86] proposed to address the problem of cluster deviation without the use of the two thresholds. The algorithm consists in saving the first value of the cluster, when it is created, in another cluster center. Then, the current cluster value is compared to its past value after every updating operation of the cluster value for deviation control. If the deviation is important and bigger than a threshold, a new cluster is created from the past one and takes the weight of the current one. To optimize the used memory, the old clusters with no update are deleted based on the assumption that a background cluster is updated with a large time frequency.

1.6 Neural Network Models

In this case, the background information is represented in terms of the weights of a neural network, which is trained on a set of frames. The network learns how to classify each pixel as background or foreground. The main approaches can be classified in the following model categories:

General Regression Neural Network: Culibrk et al. [87, 88] proposed to use a neural network architecture to form an unsupervised Bayesian classifier for background modeling. The constructed classifier efficiently handles the segmentation in natural sequences with complex background motion and changes in illumination. The weights allow us to develop a background model, which is updated to reflect the variations in the background. This algorithm may be parallelized on a sub-pixel level and designed to enable efficient hardware implementation.

Multivalued Neural Network: Luque et al. [89] used a foreground detection method based on the use of a multivalued discrete neural network. The multivalued neural network model is used to eliminate some of the deficiencies of the MOG algorithm. One of the advantages of using a multivalued neural network for foreground detection is that all process units (neurons) compute the solution in parallel. Another advantage of the multivalued neural model is its ability to represent non-numerical classes or states. This

can be very useful when dealing with foreground detection problems, in which pixel states are usually defined in terms of qualitative labels such as foreground, background and shadow.

Competitive Neural Network: Luque et al. [90] used competitive neural network based approach on adaptive neighborhoods to construct the background model. The weights and the adaptive neighborhood of the neurons are used to model the background. This algorithm is developed for parallel processing on the pixel level and designed for hardware implementation to achieve real-time processing.

Dipolar Competitive Neural Network: Luque et al. [91] improved the previous approach [90] by applying a dipolar competitive neural network, which is used to classify the pixels as background or foreground. The dipolar representation is designed to deal with the problem of estimating the directionality of data at a low-computational cost. The dipolar competitive neural network achieved better results in terms of precision and false-positive rate, whereas the false-negative rate is, at least, comparable to the competitive neural network.

Self-Organizing Neural Network: Maddalena and Petrosino [92] proposed a self-organizing neural network approach for learning motion patterns in the HSV color space, which is used to construct a background model. This algorithm is known as Self-Organizing Background Subtraction (SOBS). This detects moving objects by using the background model through a map of motion and stationary patterns. Furthermore, an update of the neural network is used to make the network structure much simpler and the training step much more efficient. Recently, Maddalena and Petrosino [93] improved the SOBS method by introducing spatial coherence into the background update procedure. This lead to the so-called Spatially Coherent Self-Organizing Background Subtraction (SC-SOBS) algorithm, which provides further robustness against false detections.

Hierarchical Self Organizing Neural Network: The Self Organizing Neural Network (SONN) has some limitations related to their fixed network structure in terms of i) the flexibility of the number of neurons; and 2) lack of representation of hierarchical relations in an architecture. To address both limitations, Palomo et al. [94] proposed a growing hierarchical neural network. This neural network model has a hierarchical structure divided

into layers, where each layer is composed of different single SONNs with adaptive structures. The adaptive structures are determined during the unsupervised learning process in accordance with input data. Experimental results show good performances, especially in the case of illumination changes.

1.7 Estimation Models

In estimation-models category the background model is obtained using a filter. Any pixel of the current image that deviates significantly from its predicted value is declared foreground. This filter may one of the following types, which are discussed below:

Wiener filter: Toyama et al. [33] proposed an algorithm called Wallflower, which makes probabilistic predictions about a background pixel value in the next image using a one-step Wiener filter. The Wiener filter is a linear predictor based on the recent history of pixel values. Any pixel in the current image that deviates significantly from its predicted value is declared foreground. The Wiener filter works well for periodically changing pixels; while for randomly changing pixel values, it produces a larger value of the threshold used in foreground detection. The main advantage of the Wiener filter is that it reduces the uncertainty of a pixel value by taking into account how it varies with time. The shortcoming of the method becomes evident when a moving object corrupts the history of pixel values. To solve this, Toyama et al. [33] kept a history of the predicted values at each pixel as well as the history of actual values. For every new pixel, they computed two predictions based on the actual history and the predicted history. If either prediction is within a tolerance level, the pixel is considered as background. To handle adaptation, the prediction coefficients are recomputed for every new frame. Furthermore, the authors incorporated a frame-level algorithm to deal with global illumination changes.

Kalman filter: Karmann et al. [95] proposed a background estimation algorithm based on the Kalman filter. It is an optimal estimator of the state of processes. It is based on the assumption that (1) they can be modeled by a linear system, (2) the measurement and the process noise are white, and have zero mean Gaussian distributions. Under these conditions, the Kalman filter provides an optimal estimate of the state of the process. In [95], the system state corresponds to the background image, and the measurements to the input gray levels. The system input term is set to zero, and the temporal distributions of

the background intensities are considered constant. All unexpected changes are described by random noise, which by hypothesis, is a zero mean Gaussian variable. In this approach, gradual illumination changes can be captured only by the random noise term, which has to vary in time accordingly. Another limitation of the method is that sudden illumination changes cause intensity variations that are considered as foreground pixels and cannot be correctly managed. To address these limitations, Boninsegna and Bozzoli [97] introduced a new term to model the intensity variations caused by gradual illumination changes. In addition, Messelodi et al. [98] developed a module that measured such global changes, and used this information as an external input to the system. Other improvements in this category of methods are use of the texture features instead of the intensity features [98–101], and a local region feature rather than a pixel feature [102]. Wang et al. [103] proposed to use an extension of the Kalman filter to nonlinear systems, called Unscented Kalman Filter (UKF) to address more dynamic backgrounds, abrupt illumination changes, and camera jitter.

Correntropy filter: The Kalman filter gives the optimal solution to the estimation problem when all the processes are Gaussian random processes, but it offers a sub-optimal behavior in non-Gaussian settings. This happens in some challenging situations met in video-surveillance. So, Cinar and Principe [104] proposed a Correntropy filter that extracts higher order information from the sequence. The information theoretic cost function is based on the similarity measure Correntropy. The Correntropy filter copes with salt and pepper noise which is not Gaussian.

Chebychev filter: Chang et al. [96] used a Chebychev filter to model the background. The objective is to update the background under changes in illumination while utilizing low memory as well as low computational resources. The drastic changes can be detected if there is a large discrepancy between the estimated background and the current frame for several frames in the sequence. Only two frames are kept in memory to represent the filter. Furthermore, the intensity channel is utilized to generate the background image.

1.8 Disadvantages of Pixel-based Approaches

Large amount of video data generated every day is stored in the compressed form for archiving, distribution, and streaming purposes. The increasing popularity of digital

surveillance cameras with has necessitated the development of fast and efficient video analysis tools that require fewer computations in real-time. Conventional pixel-based video analysis systems require decoding of huge number of frames per unit time. It involves higher computational complexity and storage for full decoding of all frames. This is a bottleneck for real-time performance, even though they offer satisfactory performance in terms of analysis.

On the contrary, video analysis in the compressed domain requires a nominal computational resource as well as lower bandwidth/storage requirements. It may be noted that, bit rate of uncompressed video is huge compared to its compressed-domain counterpart. Compressed-domain analysis for moving objects requires only partial decoding of encoded information such as motion vectors, transform coefficients, quantization parameters, macroblock partition modes, etc. Moreover, the availability of hardware codecs for the existing video compression standards such as MPEG-1/2, H.264/AVC (Advanced Video Coding) [105], and HEVC (High-Efficiency Efficiency Video Coding) [106] have made it possible to analyze their performances in real-time.

1.9 Objective and Contribution of the Thesis

Given the limitations of pixel-based approaches, the main objective of this thesis is to study and propose different aspects of foreground segmentation methods in popular compressed domain formats, such as H.264 (Baseline and High profile) and HEVC compressed video under real-world constraints. The methods are tested in a practical traffic surveillance scenario. This was motivated by the fact that the information required for foreground motion/activity is partly embedded into the compressed form. So, if by some way or the other, the information in compressed domain can be extracted then the overall computational burden of foreground extraction may be greatly reduced making the way for real-time analysis applicable to real time streaming video. Hence, different encoding constraints (bit rate, encoding parameters) as well as challenging surveillance scenarios (dynamic background, moving shadows) are explored during the Ph.D research work. The main emphasis is given to the development of methods for feature extraction in the compressed-domain, which would enable faster analysis of encoded video. In-depth research of the above issues leads to the development of some novel techniques in the compressed domain, which could be integrated as a pre-processing utility in a real-time surveillance system.

The main contributions of the thesis are described in four contributory chapters as follows:
1) Development of a highly efficient block-level feature extraction technique for compressed video encoded in H.264 Baseline profile. The main aim is to have background segmented in real-time. In H.264 compressed videos, each frame is composed of several macroblocks. Moving object segmentation from video using information of each decoded pixel, is computationally expensive task. Hence, in view of above, two macroblock features are defined [107] using partially encoded information of each macroblock. The features with a higher value correspond to object motion, which enable us to select probable macroblocks corresponding to moving objects or foreground motion. In a secondary stage, the block-level segmentation is further refined using the decoded pixel information of only the selected macroblocks. The novelty of the approach is that using encoded information of macroblocks, computation and processing related to pixels of the background are substantially reduced.

2) It may also be noted that the Baseline profile of H.264 can provide a limited number basic parameters only (unidirectional motion vectors, uniform quantization parameters for color components – Cb and Cr). Compared to the Baseline profile, High/Main profile uses a larger set of parameters (such as bi-directional and weighted prediction, separate quantization parameters, etc) to encode a video with similar quality using less storage/bandwidth. Compared to the Baseline profile, the motion information of macroblocks in H.264 Main/High profile are distributed over larger set of parameters. Thus, using the set of parameters available for the Baseline are inadequate to describe the motion information of macroblocks for video encoded in H.264 Main/High profile completely. Experimentally, it is observed that the segmentation performance using the macroblock features described in the previous methods drastically deteriorate with compression bitrate. In view of that, a new method [108] is suggested for H.264 Main/High profile which is robust (in terms of segmentation accuracy) under moderate to very low bitrate compression.

3) The new HEVC [106] standard for video compression, promises up to 50% bit rate savings compared against the best of compression schemes available today. This is a futuristic standard and its encoding and decoding methods are processor intensive. Hence, it is currently deployed in applications where high-end computing facilities can be

afforded. Unlike macroblocks in H.264 (Baseline, Main, and High profiles), the block-coding unit in HEVC is known as the Coding Tree Units (CTU) and its encoding structure is hierarchical in nature. To extract the motion information of CTU blocks, spatio-temporal features are proposed in [109]. The extracted CTU features are later used to segment foreground objects from HEVC compressed video. This is one of the first methods in the literature on segmentation of moving objects from HEVC compressed video.

4) The background subtraction masks obtained earlier by the above methods are used in a real-life traffic surveillance application. The basic objective was to obtain real-time information about the number of vehicles on various road segments of a traffic intersection. The data was captured by video cameras placed at different traffic intersections in the city of Kolkata. The results obtained for moving object segmentation from compressed video (chapter 4) was taken as input. The segmented patches were extracted, labeled and trained using a deep-layered architecture of convolutional neural networks into two broad categories, viz. vehicles and non-vehicles. In the testing sequence vehicles are detected tracked between the source and destination using the Extended Kalman-filter and total number of vehicles are noted. The application has been reported in [110].

To summarize, the present thesis comprises of seven chapters, four of which describe novel contributions. Chapter-wise organization of the current thesis is depicted in Fig. 1.4 and is described below:

- Chapter 2: Preliminaries on Different Video Coding Standards

  This chapter contains a brief overview of video compression in general. Thereafter, it is followed by a description of the compression process used in H.264 (Baseline, Main/ High) and HEVC. The advantages of different codec along with their profiles are highlighted in brief. A profile more or less defines what the encoder can use when compressing the input video – and there are lots of H.264 features that the encoder can enable. Which ones it's allowed to enable is defined by the profile. Profiles ensure compatibility between devices having different decoding capabilities. With profiles, the encoder and decoder agree on a feature set that they can both handle.
- Chapter 3: Feature Extraction for Background Subtraction from Video Compressed in H.264 Baseline Profile

  Based on the analysis, it has been found that most related techniques that work in H.264 compressed domains are based solely on the motion vector field, while ignoring

the residual coded information. Motion vectors, however, do not necessarily correspond to true object motion in a video scene, and such related techniques end up wrongly classifying the dynamic motion of background pixels (eg. waving tree branches, fountain, ripples, camera jitter, etc.) as the foreground. To overcome the above limitations, two features are proposed; one based on the motion vector field, and the other based on transform coefficients [107].

- Chapter 4: Enhanced Macroblock Feature Extraction Under Low Bitrate Encoding Constraints of H.264 Main/High Profile

  In order to meet low bitrate requirements of a practical surveillance scenario, the compression algorithm in the Main/High Profile of H.264 uses a larger set of encoding parameters and features that are not available in the baseline profile. Enhanced macroblock features are proposed [108] using quantization step-sizes of individual coefficients in a macroblock as well as partition-sizes of predicted sub-macroblocks. This is particularly important as the parameters vary widely at constrained low bitrates.

- Chapter 5: Foreground Extraction from HEVC Compressed Video using CTU Features

  In today's application areas of video surveillance, such as parking lot, hotel, harbor, airport, etc., there is a growing demand on detailed capturing, and at the same time, a need for faster decoding and bandwidth/storage requirements. The introduction and development of HEVC standard brought extensive possibilities to the industry by addressing problems such as shortage of bandwidth, improving transmission efficiency. Being the most bandwidth-efficient codec today, HEVC will likely be a very popular choice for video content delivery in the coming decade. In applications where minimizing bandwidth is not the highest priority, HEVC can still be used to improve video quality significantly at the same bitrate as H.264/AVC. In literature, until recently there were no well-known algorithms for video content interpretation and analysis using features derived specifically from HEVC coded video. In this chapter, a scheme for segmenting foreground regions from HEVC compressed sequence is presented [109].

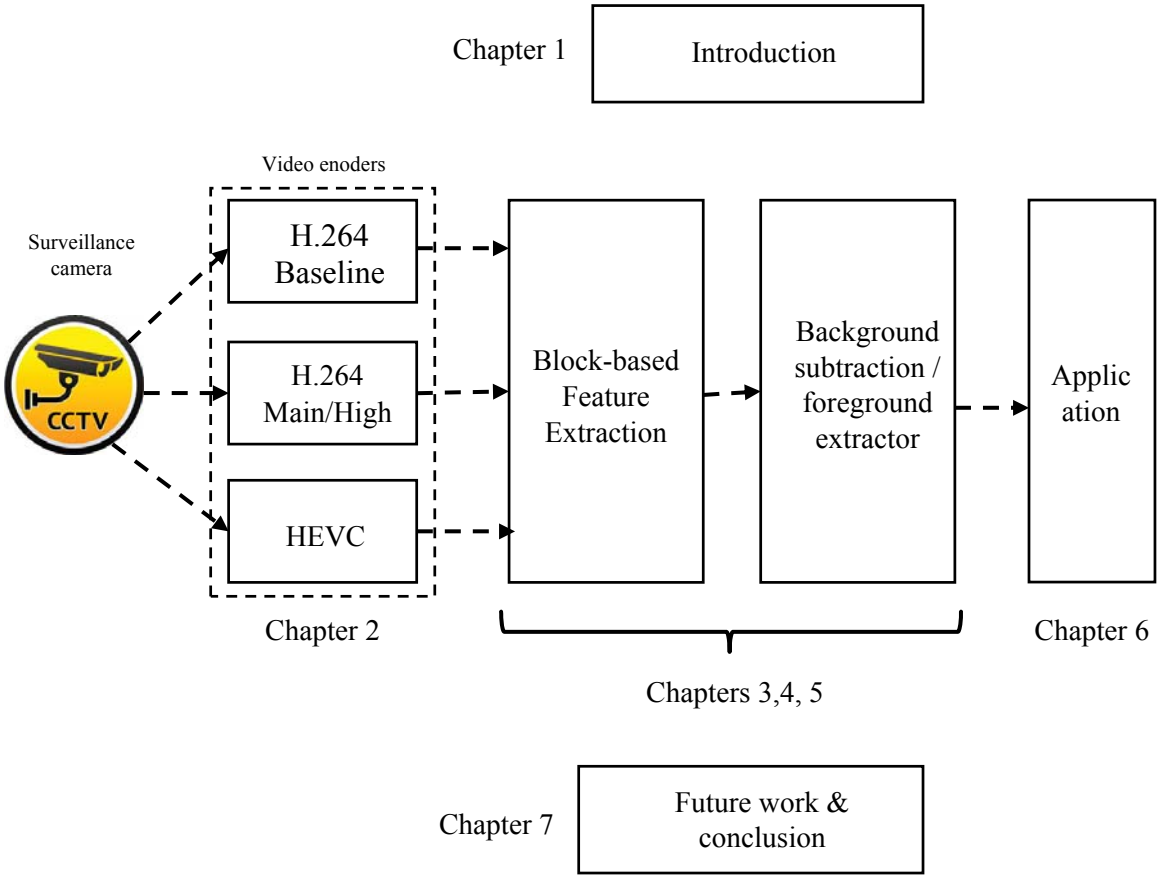


Fig. 1.3 Thesis outline: A pictorial representation

- Chapter 6: An Application to Traffic Sequence Analysis

In modern intelligent transportation systems, the video-image vehicle-detection system (VIVDS) is gradually becoming one of the popular methods at signalized traffic intersection due to its convenient installation and rich information content provided. In order to understand, investigate and optimize the operation of transportation networks in middle-scale cities and metropolitans, the urban transportation planning industry actively engages in traffic monitoring programs, which involve in traffic data and traffic information collection at given intersections or main roads [110]. Preliminary study shows that the VIVDS for the traffic monitoring can reduce the costs for the traffic data collection, traffic scene analysis and potentially provide more accurate and useful traffic information for the supporting or transportation service systems. Besides, VIVDS has recently become one of the key parts in the intersection decision support system, which is developed to enhance the driver's ability to pass the intersections. For archival and retrieval, the traffic sequences are usually stored in H.264/HEVC compression formats.

- Chapter 7: An Application to Traffic Sequence Analysis

Finally, it is tried to highlight the possible novel applications of the research studies conducted during the dissertation development, and how it could be extended/improved so that the compressed-domain paradigm could be applied in other fields.

Chapter 2

Preliminaries on Different Video Coding Standards

2.1 Video Compression Standards

Video Compression is the term used to define a method for reducing the data used to encode digital video content. This reduction in data, translates to benefits, such as smaller storage requirements and lower transmission bandwidth requirements. Motion JPEG (M-JPEG) is one of the earliest video compression formats. Originally developed by JPEG (Joint Photographic Experts Group) for multimedia applications, M-JPEG is used by many digital video capture devices. In M-JPEG, each video frame is separately compressed as a single JPEG image.

Moving Picture Experts Group or **MPEG** is a working group of the ISO (International Organization for Standardization), which develops major standards for digital video compression. The term also refers to the family of digital video compression standards and file formats developed by the group. MPEG algorithms compress data to form small bits that can be easily transmitted. MPEG achieves its high compression rate by storing only the changes from one frame to another, instead of each individual frame. MPEG uses a type of *lossy* compression scheme, since some data is lost during the compression process. However, the loss of data is generally imperceptible to the human eye.

The major MPEG standards include the following;

1) MPEG-1: The most common implementations of the MPEG-1 standard provide a video resolution of 352-by-240 at 30 frames per second (fps). This produces video quality slightly below the quality of conventional VCR videos.
2) MPEG-2: Offers resolutions of 720x480 and 1280x720 at 60 fps, with full CD-quality audio. This is sufficient for all the major TV standards, including NTSC, and even HDTV. MPEG-2 is used by DVD-ROMs. MPEG-2 can compress a 2-hour video into a few gigabytes. While decompressing an MPEG-2 data stream requires only modest computing power, encoding video in MPEG-2 format requires significantly more processing power. An attempt to develop MPEG-3 was taken for HDTV but later abandoned in place of using MPEG-2.
3) MPEG-4: A graphics and video compression algorithm standard that is based on MPEG-1 and MPEG-2 and Apple QuickTime technology. Wavelet-based MPEG-4 files are smaller than JPEG or QuickTime files, so they are designed to transmit video and images over a narrower bandwidth and can mix video with text, graphics and 2-D and 3-D animation layers. MPEG-4 was standardized in October 1998 in

the ISO/IEC document 14496. The recent developments of the standards are Advanced Video Coding (AVC)/H.264 and High Efficiency Video Coding (HEVC). The H.264 supports three encoding profiles, viz. Baseline, Main and High. These profiles are developed to support various cost and performance goals.

4) MPEG-7: Formally called the Multimedia Content Description Interface, MPEG-7 provides a tool set for completely describing multimedia content. MPEG-7 is designed to be generic and not targeted to a specific application.
5) MPEG-21: Includes a Rights Expression Language (REL) and a Rights Data Dictionary. Unlike other MPEG standards that describe compression coding methods, MPEG-21 describes a standard that defines the description of content and also processes for accessing, searching, storing and protecting the copyrights of content.

Today, H.264 (all profiles) and HEVC are by far the most popular compression formats used in storage and transmission of video. H.264 is the result of a joint project between the ITU-T's Video Coding Experts Group and the ISO/IEC Moving Picture Experts Group (MPEG). H.264 has already been introduced in new electronic gadgets such as mobile phones and digital video players, and has gained fast acceptance by end users. Service providers such as online video storage and telecommunications companies have also adopted H.264 as their premier coding format.

Video compression technologies are about reducing and removing redundant video data so that a digital video file can be effectively sent over a network and stored on computer disks. With efficient compression techniques, a significant reduction in file size can be achieved with little or no adverse effect on the visual quality. The video quality, however, can be affected if the file size is further lowered by raising the compression level for a given compression technique.

Different compression technologies, both proprietary and industry standards, are available. Most network video vendors today use standard compression techniques. Standards are important in ensuring compatibility and interoperability. They are particularly relevant to video compression since video may be used for different purposes and media. Furthermore, the video needs to be viewable many years from the recording date.

A pair of algorithms that works together for video encoding and decoding is called a codec (encoder/decoder). Video codecs of different standards are normally not compatible with each other; that is, video content that is compressed using one standard

cannot be decompressed with a different standard. For instance, an MPEG-2 decoder will not work with an MPEG-4 encoder. This is simply because one algorithm cannot correctly decode the output from another algorithm but it is possible to implement many different algorithms in the same software or hardware, which would then enable multiple formats to coexist on the same system. Even within a particular video compression codec, there are various levels of compression that can be applied (so called as profiles); and the more aggressive the compression, the greater the savings in storage space and transmission bandwidth, but the lower the quality of the compressed video (as manifested in visual artefacts such as blockings, pixelated edges, blurring, rings – that appear in the video) and the greater the computing power required. Examples of other video compression formats are H.261, MPEG-1, MPEG-2, MPEG-4 Visual, Theora, Dirac, RealVideo RV40, VP8, HEVC etc. However, standards such as H.264 and HEVC are considered here, which are most widely used and promising open source video compression techniques today.

2.2 Redundancy

The basic aim of compression is to reduce redundancy in data and use a smaller representation such that when it is decompressed, there is no visual difference between the original and the decompressed format. This can be achieved by eliminating various types of redundancy that exist in the pixel intensity values. In general, three basic redundancies exist in digital images/video, which are as follows.

Psycho-visual Redundancy: It is a redundancy corresponding to different sensitivities to all image signals by human eyes. Therefore, eliminating some less important information in our visual processing may be acceptable.

Inter-pixel Redundancy: It is a redundancy corresponding to statistical correlation between neighboring pixels. It is a common experience that neighboring pixels have highly similar intensities within the same frame (spatial-redundancy) and those in the same position in adjacent frames (temporal redundancy).

Coding Redundancy: The uncompressed image usually is coded with each pixel by a fixed length. For example, an image with 256 grey scales is represented by an array of 8-bit integers. Using some variable length code schemes such as Huffman coding and arithmetic coding may produce compression.

Different compression standards utilize different methods of reducing data redundancy, and hence, results differ in bit rate and quality. Compression algorithms fall into two types:

image compression and video compression. Image compression uses intra-frame, i.e., within the same frame, coding technology. Data is reduced within an image frame simply by removing redundant information that may not be noticeable to the human eye. Motion JPEG is an example of such a compression standard. Images in a Motion JPEG sequence is coded or compressed as individual JPEG images.

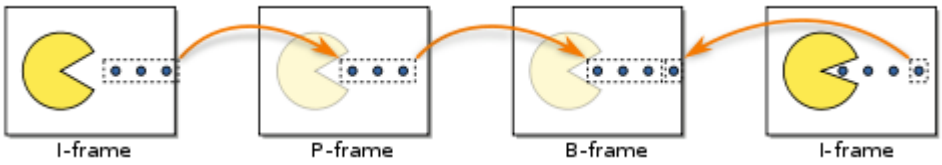


Fig. 2.1 A sequence of video frames, consisting of two keyframes (I), one forward-predicted frame (P) and one bi-directionally predicted frame (B). Arrows indicate the references to frames

Depending on the encoding profile, different types of frames such as I-frames, P-frames and B-frames, may be used by an encoder. An **I-frame**, or intra frame, is a self-contained frame that can be independently decoded without any reference to other images. The first image in a video sequence is always an I-frame. I-frames are needed as starting points for new viewers or resynchronization points if the transmitted bit stream is damaged. I-frames can be used to implement fast-forward, rewind and other random access functions. An encoder will automatically insert I-frames at regular intervals or on demand if new clients are expected to join in viewing a stream. The drawback of I-frames is that they consume much more bits, but on the other hand, they do not generate many artifacts. A **P-frame**, which stands for predictive inter frame, makes references to parts of earlier I and/or P frame(s) to code the frame. P-frames usually require fewer bits than I-frames, but a drawback is that they are very sensitive to transmission errors because of the complex dependency on earlier P and I reference frames. A **B-frame**, or bi-predictive inter frame, is a frame that makes references to both an earlier reference frame and a future frame. Fig. 2.1 illustrates the relation between the frame types.

When a video decoder restores a video by decoding the bit stream frame by frame, decoding must always start with an I-frame. P-frames and B-frames, if used, must be decoded together with the reference frame(s). In the H.264 baseline profile, only I- and P-frames are used.

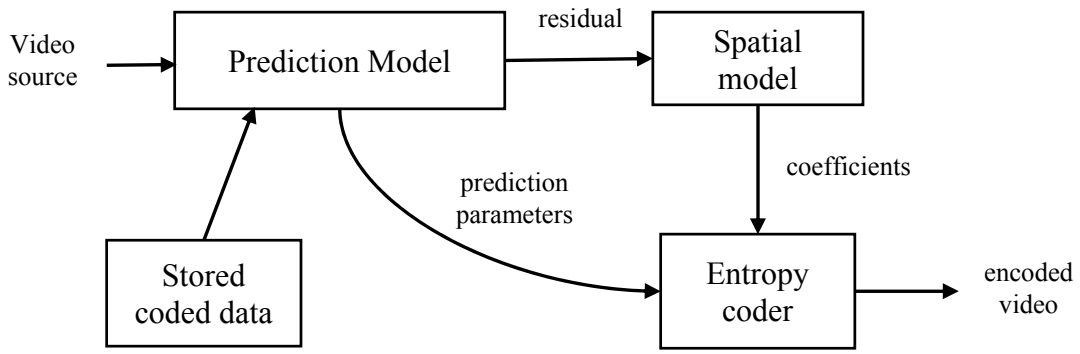


Fig. 2.2 Video Compression block diagram

2.3 Video Compression

Video compression (Figure 2.2) consists of three main functional units: a prediction model, a spatial model and an entropy encoder. The input to the prediction model is an uncompressed 'raw' video sequence. The prediction model attempts to reduce the redundancies by exploiting the similarities between neighboring video frames, typically by constructing a prediction of the current video frame or block of video data. In H.264/AVC, the prediction is formed from data in the current frame or in one or more previous and/or future frames. It is created by spatial extrapolation from neighboring image samples, intra-prediction, or by compensating for differences between the frames, motion compensated or inter-prediction. The output of the prediction model is a **residual** frame, created by subtracting the prediction from the actual current frame, and a set of model parameters indicating the intra prediction type or describing how the motion was compensated.

The residual frame forms the input to the spatial model which makes use of similarities between local samples in the residual frame to reduce spatial redundancy. In most compression schemes this is carried out by applying a transform to the residual samples and quantizing the results. The transform converts the samples into another domain in which they are represented by transform coefficients. The coefficients are floating point values. The coefficients are quantized to remove insignificant values, leaving a small number of significant coefficients that provide a more compact representation of the residual frame. The output of the spatial model is a set of quantized transform coefficients.

The parameters of the prediction model, *i.e.* intra prediction mode(s) or inter prediction mode(s) and motion vectors, and the spatial model, i.e. coefficients, are

compressed by the entropy encoder. This removes the redundancies in the video data, for example representing commonly occurring vectors and coefficients by short binary codes. The entropy encoder produces a compressed bit stream or file that may be transmitted and/or stored. A compressed sequence consists of coded prediction parameters, coded residual coefficients and header information.

2.4 H.264 family of Video Coding Standards

2.4.1 Introduction

H.264, also known as MPEG-4 Part 10/AVC for Advanced Video Coding, is the most popular MPEG standard for video encoding. H.264 is expected to become the video standard of choice in the coming years. This is because an H.264 encoder can, without compromising image quality, reduce the size of a digital video file by more than 80% compared with the Motion JPEG format and as much as 50% more than with the MPEG-4 standard. This means that much less network bandwidth and storage space are required for a video file. Or seen another way, much higher video quality can be achieved for a given bit rate.

H.264 was jointly defined by standardization organizations in the telecommunications (International Telecommunication Union's Video Coding Experts Group) and information technology industries (ISO/IEC Moving Picture Experts Group), and is expected to be more widely adopted than previous standards. In the video surveillance industry, H.264 will most likely find the quickest applications where there are demands for high frame rates and high resolution, such as in the surveillance of highways, shopping malls, airports and casinos, etc. This is where the economies of reduced bandwidth and storage needs will deliver the biggest savings.

Video signals in H.264 typically use YCbCr color space with 4:2:0 sampling (instead of the red-green-blue color space). This separates a color representation into three components called Y, Cb, and Cr. The Y component is also called luma, and represents brightness. The two chroma components Cb and Cr represent the extent to which the color deviates from gray toward blue and red, respectively.

In H.264, each frame of a video is represented in units of a **macroblock** (16x16 displayed pixels). Apart from macroblocks, H.264 introduces a new concept called **slices** — segments of a frame consisting of one or more macroblocks and having a maximum size as that of the frame. Video compression is achieved by a prediction of the macroblock

based on previously-coded data, either form the current frame, i.e., **intra-prediction** or from other frames that have already been coded and transmitted, i.e., **inter-prediction**. The encoder subtracts the prediction from the current macroblock to form a **residual**. Finding a suitable inter-prediction is often described as **motion estimation**. Subtracting an inter-prediction from the current macroblock is called **motion compensation**. The prediction methods supported by H.264 are more flexible than in previous standards, enabling predictions and hence efficient video compression. Intra-prediction uses 16x16 and 4x4 block sizes to predict the macroblock from surrounding, previously coded pixels with the same frame. Inter-prediction uses a range of block-sizes (from 16x16 down to 4x4) to predict pixels in the current frame from similar regions in previously-coded frames.

As discussed earlier, the residual transform process is followed by quantization. Quantization reduces the precision of the transform coefficients according to a quantization parameter (QP). Typically, the result is a block in which most or all of the coefficients are zero, with a few non-zero coefficients. Setting QP to a high value means that more coefficients are set to zero, resulting in high compression at the expense of poor decoded image quality. Setting QP to a low value means that more non-zero coefficients remain after quantization, resulting in better decoded image quality but lower compression.

Finally, these values and parameters (syntax elements) are converted into binary codes using **context adaptive variable length coding** (CAVLC) and/or **context adaptive binary arithmetic coding** (CABAC). Each of these encoding methods produces an efficient, compact binary representation of the information. The encoded bitstream can then be stored and/or transmitted.

In the following, detailed discussion is presented regarding macroblock prediction and partitions, the transform and quantization processes in H.264/AVC.

2.4.2 Macroblock prediction

Much of the performance gain compared with previous standards is due to H.264/AVC's efficient prediction methods. As aforementioned, a macroblock is basic coding unit of H.264. For every macroblock, a prediction is created, an attempt to duplicate the information contained in the macroblock using previously coded data, and subtracted from the macroblock to form a residual.

Figure 2.3 shows the prediction sources for three macroblocks, an I-macroblock, a P-Macroblock and a B-macroblock. An I-macroblock (I macroblock) is predicted using

intra prediction from neighboring samples in the current frame. A P macroblock is predicted from samples in a previously-coded frame which may be before or after the current picture in display order, i.e. a 'past' or a 'future' frame. Different rectangular sections (partitions) in a P macroblock may be predicted from different reference frames. Each partition in a B macroblock is predicted from samples in one or two previously-coded frames, for example, one 'past' and one 'future' as shown in the figure.

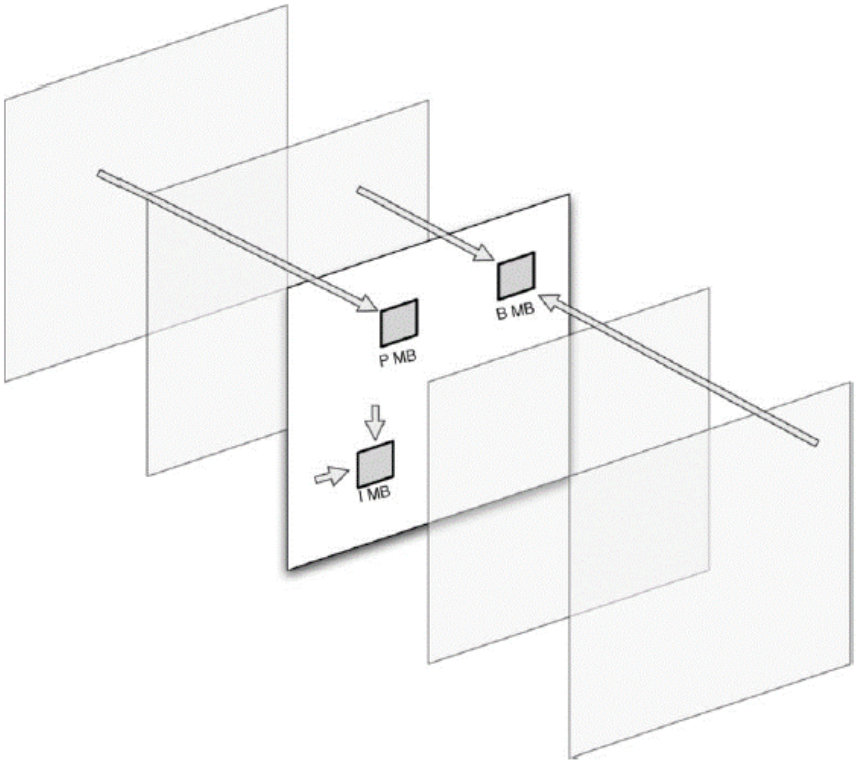


Fig. 2.3 Example of macroblock types and prediction sources

An intra (I) macroblock is coded without referring to any data outside the current slice. I macroblocks may occur in any slice type. Every macroblock in an I slice is an I macroblock. I macroblocks are coded using intra prediction, *i.e*. prediction from previously-coded data in the same slice. In an intra macroblock, there are three choices of intra prediction block size for the luma component, namely 16 × 16, 8 × 8 or 4 × 4.

In case of inter-prediction, each 16×16 P or B-macroblock may be predicted using a range of block sizes. The macroblock is split into one, two or four macroblock partitions: either

(a) one 16x16 macroblock partition (covering the whole macroblock),

(b) two 8x16 partitions,

(c) two 16x8 partitions or

(d) four 8x8 partitions.

If 8x8 partition size is chosen, then each 8x8 block of luma samples and associated chroma samples, a sub-macroblock, is split into one, two or four sub-macroblock partitions): one 8x8, two 4x8, two 8x4 or four 4x4 sub-macroblock partitions.

2.4.3 Forward Transform and Quantization in H.264/AVC

As mentioned earlier, the basic 4x4 transform used in H.264 is a scaled approximation of the Discrete Cosine Transform (DCT). It may be noted that the Baseline profile supports only 4x4 transform. The transform and quantization processes are structured such that computational complexity is minimized. This is achieved by reorganizing the processes into a core part and a scaling part. Consider a block of pixel data that is processed by a two-dimensional DCT followed by quantization (dividing by a quantization step size, Q_{step}, then rounding the result) (Figure 2.4a). Rearrange the DCT process into a core transform (C_f) and a scaling matrix (S_f) (Figure 2.4b). Scale the quantization process by a constant (2^{15}) and compensate by dividing and rounding the final result (Figure 2.4c). Combining S_f and the quantization process into M_f (Figure 2.4d), we get:

$$M_f \approx \frac{2^{15} S_f}{Q_{step}} \tag{2.1}$$

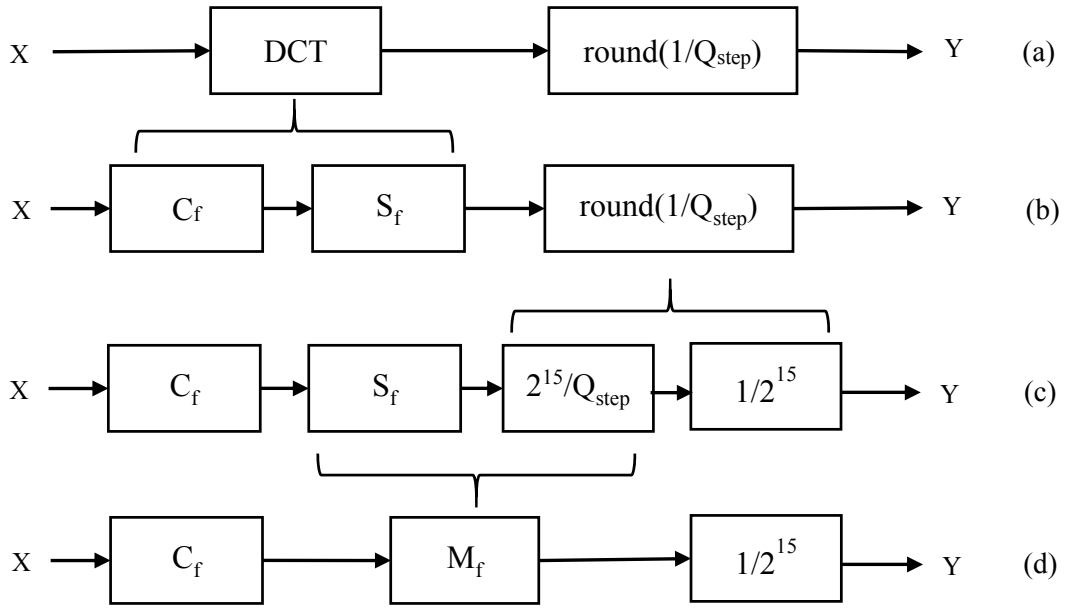


Fig. 2.4 Development of the forward transform and the quantization process

2.4.4 Rescaling and the Inverse Transform Process

During the decoding process the re-scaling operation is followed by a two-dimensional

inverse DCT (IDCT) (Figure 2.5a). In practice, however, the whole process is split into a core transform (C_i) and a scaling operations (S_i) (Figure 2.5b). Scale the re-scaling process by a constant (2^6) and compensate by dividing and rounding the final result (Figure 2.5c). Combine the re-scaling process and S_i into V_i (Figure 2.5d), where:

$$V_i = 2^6 S_i Q_{step} \tag{2.2}$$

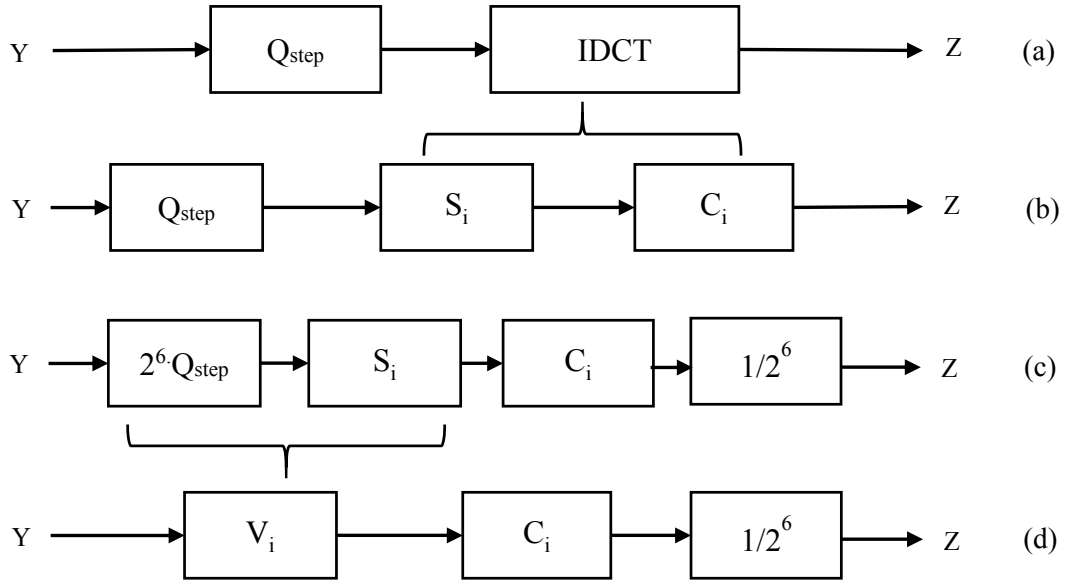


Fig. 2.5 Development of the rescaling and inverse transform process

2.4.5 Development of C_f and S_f (4x4 blocks)

Considering a 4x4 two-dimensional DCT of a block of pixels *X*, we have

$$Y = A \cdot X \cdot A^T \tag{2.3}$$

where · indicates matrix multiplication, and

$$A = \begin{bmatrix} 1/2 & 1/2 & 1/2 & 1/2 \\ \frac{1}{\sqrt{2}}\cos(\pi/8) & \frac{1}{\sqrt{2}}\cos(3\pi/8) & -\frac{1}{\sqrt{2}}\cos(3\pi/8) & -\frac{1}{\sqrt{2}}\cos(\pi/8) \\ 1/2 & -1/2 & -1/2 & 1/2 \\ \frac{1}{\sqrt{2}}\cos(3\pi/8) & -\frac{1}{\sqrt{2}}\cos(\pi/8) & \frac{1}{\sqrt{2}}\cos(\pi/8) & -\frac{1}{\sqrt{2}}\cos(3\pi/8) \end{bmatrix}$$

The rows of A are orthogonal and have unit norms, i.e., the rows are ortho-normal. Calculation of Y on a processor requires approximation of the irrational numbers $\frac{1}{\sqrt{2}}\cos(3\pi/8)$ and $\frac{1}{\sqrt{2}}\cos(\pi/8)$. A fixed-point approximation is equivalent to scaling each row of A and rounding to the nearest integer. Choosing a particular approximation (multiply by 2.5 and round) gives C_{f4} as

$$C_{f4} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}$$

This approximation is chosen to minimize the complexity of implementing the transform (multiplication by C_{f4} requires only additions and binary shifts) whilst maintaining good compression performance. The rows of C_{f4} have different norms. To restore the orthonormal property of the original matrix A, each row of A is multiplied by the reciprocal of the corresponding row. Therefore, a new matrix A_1 is obtained as

$$A_1 = C_f \bullet R_f \tag{2.4}$$

where

$$R_f = \begin{bmatrix} 1/2 & 1/2 & 1/2 & 1/2 \\ 1/\sqrt{10} & 1/\sqrt{10} & 1/\sqrt{10} & 1/\sqrt{10} \\ 1/2 & 1/2 & 1/2 & 1/2 \\ 1/\sqrt{10} & 1/\sqrt{10} & 1/\sqrt{10} & 1/\sqrt{10} \end{bmatrix},$$

and denotes element-by-element multiplication (Hadamard-Schur product). Note that the new matrix A_1 is orthonormal.

The two-dimensional transform (Equation 2.3) becomes:

$$Y = A_1 \cdot X \cdot A_1^T = [C_f \bullet R_f] \cdot X \cdot [C_f^T \bullet R_f^T] \tag{2.5}$$

Rearranging terms, we get

$$\begin{aligned} Y &= [C_f \cdot X \cdot C_f^T] \bullet [R_f \bullet R_f^T] \\ &= [C_f \cdot X \cdot C_f^T] \bullet S_f \end{aligned} \tag{2.6}$$

where

$$S_f = R_f \bullet R_f^T = \begin{bmatrix} 1/4 & 1/2\sqrt{10} & 1/4 & 1/2\sqrt{10} \\ 1/2\sqrt{10} & 1/10 & 1/2\sqrt{10} & 1/10 \\ 1/4 & 1/2\sqrt{10} & 1/4 & 1/2\sqrt{10} \\ 1/2\sqrt{10} & 1/10 & 1/2\sqrt{10} & 1/10 \end{bmatrix}.$$

2.4.6 Development of C_i and S_i (4x4 blocks)

Considering a 4x4 two-dimensional IDCT of a block of pixels *Y*, we have

$$Z = A^T \cdot Y \cdot A \tag{2.7}$$

where

$$A = \begin{bmatrix} 1/2 & 1/2 & 1/2 & 1/2 \\ \frac{1}{\sqrt{2}}\cos(\pi/8) & \frac{1}{\sqrt{2}}\cos(3\pi/8) & -\frac{1}{\sqrt{2}}\cos(3\pi/8) & -\frac{1}{\sqrt{2}}\cos(\pi/8) \\ 1/2 & -1/2 & -1/2 & 1/2 \\ \frac{1}{\sqrt{2}}\cos(3\pi/8) & -\frac{1}{\sqrt{2}}\cos(\pi/8) & \frac{1}{\sqrt{2}}\cos(\pi/8) & -\frac{1}{\sqrt{2}}\cos(3\pi/8) \end{bmatrix}$$

as before. By scaling each row of A and rounding to the nearest 0.5, C_i is obtained.

$$C_i = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1/2 & -1/2 & -1 \\ 1 & -1 & -1 & 1 \\ 1/2 & -1 & 1 & -1/2 \end{bmatrix}$$

The rows of A are orthogonal but non-unit norms. To restore the orthonormal property of the original matrix A, each row of A is multiplied by the reciprocal of the corresponding row. Therefore, the new matrix A_2 is obtained as

$$A_2 = C_i \bullet R_i \tag{2.8}$$

where

$$R_i = \begin{bmatrix} 1/2 & 1/2 & 1/2 & 1/2 \\ \sqrt{2/5} & \sqrt{2/5} & \sqrt{2/5} & \sqrt{2/5} \\ 1/2 & 1/2 & 1/2 & 1/2 \\ \sqrt{2/5} & \sqrt{2/5} & \sqrt{2/5} & \sqrt{2/5} \end{bmatrix}.$$

The two-dimensional inverse transform (Equation 2.7) becomes:

$$Z = A_2^T \cdot Y \cdot A_2 = [C_i^T \bullet R_i^T] \cdot Y \cdot [C_i \bullet R_i] \tag{2.9}$$

Rearranging terms, we get

$$\begin{aligned} Z &= C_i^T \cdot [Y \bullet R_i^T \bullet R_i] \cdot C_i \\ &= C_i^T \cdot [Y \bullet S_i] \cdot C_i \end{aligned} \tag{2.10}$$

where

$$S_f = R_i^T \bullet R_i = \begin{bmatrix} 1/4 & 1/\sqrt{10} & 1/4 & 1/\sqrt{10} \\ 1/\sqrt{10} & 2/5 & 1/\sqrt{10} & 2/5 \\ 1/4 & 1/\sqrt{10} & 1/4 & 1/\sqrt{10} \\ 1/\sqrt{10} & 2/5 & 1/\sqrt{10} & 2/5 \end{bmatrix}.$$

The core inverse transform C_i and the rescaling matrix V_i are defined in the H.264 standard. Hence, derivation of V_i and M_f are shown.

Table 2.1 Estimated Q_{step} (4 × 4 blocks) = V_{i4} / ($S_i \cdot 2^6$), element-by-element division

QP	~Q_{step} (n = 0)	~Q_{step} (n = 1)	~Q_{step} (n = 2)
0	0.6750	0.6250	0.6423
1	0.6875	0.7031	0.6917
2	0.8125	0.7812	0.7906
3	0.8750	0.8984	0.8894
4	1.0000	0.9766	0.9882
5	1.1250	1.1328	1.1364
6	1.2500	1.2500	1.2847
…	…		
12	2.5000	2.5000	2.5694
…	…		
18	5.0000	5.0000	5.1387
…	…		
48	160	160	164.4384
…	…		
51	224	234	227.6840

Table 2.2 Estimated Q_{step} (4 × 4 blocks) = V_{i4} / ($S_{i4} \cdot 2^6$), element-by-element division

QP	$Q_{step} \cdot 2^6$	V_i = round($S_i \cdot$ $Q_{step} \cdot 2^6$)
0	40	$\begin{bmatrix} 10 & 13 & 10 & 13 \\ 13 & 16 & 13 & 16 \\ 10 & 13 & 10 & 13 \\ 13 & 16 & 13 & 16 \end{bmatrix}$
1	44.898	$\begin{bmatrix} 11 & 14 & 11 & 14 \\ 14 & 18 & 14 & 18 \\ 11 & 14 & 11 & 14 \\ 14 & 18 & 14 & 18 \end{bmatrix}$
2	50.397	$\begin{bmatrix} 13 & 16 & 13 & 16 \\ 16 & 20 & 16 & 20 \\ 13 & 16 & 13 & 16 \\ 16 & 20 & 16 & 20 \end{bmatrix}$
3	56.569	$\begin{bmatrix} 14 & 18 & 14 & 18 \\ 18 & 23 & 18 & 23 \\ 14 & 18 & 14 & 18 \\ 18 & 23 & 18 & 23 \end{bmatrix}$
4	63.496	$\begin{bmatrix} 16 & 20 & 16 & 20 \\ 20 & 25 & 20 & 25 \\ 16 & 20 & 16 & 20 \\ 20 & 25 & 20 & 25 \end{bmatrix}$
5	71.272	$\begin{bmatrix} 18 & 23 & 18 & 23 \\ 23 & 29 & 23 & 29 \\ 18 & 23 & 18 & 23 \\ 23 & 29 & 23 & 29 \end{bmatrix}$

2.4.7 Development of V_i

From Equation 2,

$$V_i = 2^6 S_i Q_{step} \tag{2.11}$$

H.264 supports a range of quantization step sizes Q_{step}. The precise step sizes are not defined in the standard, rather the scaling matrix V_i is specified. Q_{step} values corresponding to the entries in V_i are shown in Table 2.1.

The ratio between successive Q_{step} values is chosen to be $\sqrt[6]{2} \approx 1.2246$, so that Q_{step} doubles in size when QP increases by 6. Any value of Q_{step} can be derived from the first

six values in the table (QP0 – QP5) as follows:

$$Q_{step}(QP) = Q_{step}(QP \bmod 6) \times 2^{\lfloor QP/6 \rfloor} \tag{2.12}$$

The values in the matrix V_i depend on Q_{step} (hence QP) and on the scaling factor matrix S_i. These are shown for QP 0 to 5 in Table 2.2.

Table 2.3 Table ***v*** defined in the H.264 standard

QP	***v*** (r, 0): $\mathbf{V_{i4}}$ positions (0,0), (0,2), (2,0), (2,2)	***v*** (r, 1): $\mathbf{V_{i4}}$ positions (1,1), (1,3), (3,1), (3,3)	***v*** (r, 2): Remaining $\mathbf{V_{i4}}$ positions
0	10	16	13
1	11	18	14
2	13	20	16
3	14	23	18
4	16	25	20
5	18	29	23

It may be noted that there are only three unique values in each matrix V_i. These three values are defined as a table of values ***v*** (Table 2.3) in the H.264 standard, for QP=0 to QP=5. Therefore, V_i may be denoted as:

$$V_{i4} = v(QP, n) \tag{2.13}$$

where v (r,n) is the element at row r, column n of v. For larger values of QP (QP>5), the rows of v are indexed by QP%6 and then multiply by $2^{\lfloor QP/6 \rfloor}$. In general:

$$V_{i4} = v(QP \bmod 6, n) \cdot 2^{\lfloor QP/6 \rfloor} \tag{2.14}$$

Thus, the complete inverse transform and scaling process (for 4x4 blocks in macroblocks excluding 16x16-Intra mode) becomes:

$$Z = \text{round}\left([C_{i4}^T] \cdot [Y \bullet v(QP \bmod 6, n) \cdot 2^{\lfloor QP/6 \rfloor}] \cdot [C_{i4}] \cdot \frac{1}{2^6} \right) \tag{2.15}$$

2.4.8 Derivation of M_{f4}

Combining equations – (2.1) and (2.2) – for 4x4 blocks, we get

$$M_{f4} \approx \frac{S_{i4} \cdot S_{f4} \cdot 2^{21}}{V_{i4}} = \text{round}\left(\frac{S_{i4} \cdot S_{f4} \cdot 2^{21}}{V_{i4}} \right) \tag{2.16}$$

S_{i4}, S_{f4} are known and V_{i4} is defined in the standard. The numerator of M_{f4} is given as

$$S_{i4} \bullet S_{f4} \cdot 2^{21} = \begin{bmatrix} 131072 & 104857.6 & 131072 & 104857.6 \\ 104857.6 & 83886.1 & 104857.6 & 83886.1 \\ 131072 & 104857.6 & 131072 & 104857.6 \\ 104857.6 & 83886.1 & 104857.6 & 83886.1 \end{bmatrix}. \tag{2.17}$$

The entries of matrix M_{f4} are based on matrix m. Via the relation (2.16), m it is related to v and obtained as shown in Table 2.4.

Table 2.4 Table of ***m***

QP	***v*** (r, 0): $\mathbf{V_{i4}}$ positions (0,0), (0,2), (2,0), (2,2)	***v*** (r, 1): $\mathbf{V_{i4}}$ positions (1,1), (1,3), (3,1), (3,3)	***v*** (r, 2): Remaining $\mathbf{V_{i4}}$ positions	***m*** (r, 0): $\mathbf{M_{f4}}$ positions (0,0), (0,2), (2,0), (2,2)	***m*** (r, 1): $\mathbf{M_{f4}}$ positions (1,1), (1,3), (3,1), (3,3)	***m*** (r, 2): Remaining $\mathbf{M_{f4}}$ positions
0	10	16	13	13107	5243	8066
1	11	18	14	11916	4660	7490
2	13	20	16	10082	4194	6554
3	14	23	18	9362	3647	5825
4	16	25	20	8192	3355	5243
5	18	29	23	7282	2893	4559

Hence for QP values from 0 to 5 can be obtained, M_f can be obtained from m as

$$M_{f4} = \begin{bmatrix} m(QP,0) & m(QP,2) & m(QP,0) & m(QP,2) \\ m(QP,2) & m(QP,1) & m(QP,2) & m(QP,1) \\ m(QP,0) & m(QP,2) & m(QP,0) & m(QP,2) \\ m(QP,2) & m(QP,1) & m(QP,2) & m(QP,1) \end{bmatrix}. \tag{2.18}$$

Denote this as:

$$M_{f4} = m(QP,n) \tag{2.19}$$

Where m(r,n) is the entry at row r, column n of matrix m.

For QP>5, index the row of m by (QP mod 6) and then divide by $2^{\lfloor QP/6 \rfloor}$. In general:

$$M_{f4} = m(QP \bmod 6, n) \cdot 2^{\lfloor QP/6 \rfloor} \tag{2.20}$$

The complete forward transform, scaling and quantization process (for 4x4 blocks and for

Table 2.5 Estimated Q_{step} (8×8 blocks) = V_{i8} / ($S_{i8} \cdot 2^8$), element-by-element division

QP	V_{i8} = round($S_{i8} \cdot$ $Q_{step} \cdot 2^8$)
0	$\begin{bmatrix} 20 & 19 & 25 & 19 & \cdots \\ 19 & 18 & 24 & 18 & \\ 25 & 24 & 32 & 24 & \\ 19 & 18 & 24 & 16 & \\ \cdots & & & & \cdots \end{bmatrix}$
1	$\begin{bmatrix} 22 & 21 & 28 & 21 & \cdots \\ 21 & 19 & 26 & 19 & \\ 28 & 26 & 35 & 26 & \\ 21 & 19 & 26 & 19 & \\ \cdots & & & & \cdots \end{bmatrix}$
2	$\begin{bmatrix} 26 & 24 & 33 & 24 & \cdots \\ 24 & 23 & 31 & 23 & \\ 33 & 31 & 42 & 31 & \\ 24 & 23 & 31 & 23 & \\ \cdots & & & & \cdots \end{bmatrix}$
3	$\begin{bmatrix} 28 & 26 & 35 & 26 & \cdots \\ 26 & 25 & 33 & 25 & \\ 35 & 33 & 45 & 33 & \\ 26 & 25 & 33 & 25 & \\ \cdots & & & & \cdots \end{bmatrix}$
4	$\begin{bmatrix} 32 & 30 & 40 & 30 & \cdots \\ 30 & 28 & 38 & 28 & \\ 40 & 38 & 51 & 38 & \\ 30 & 28 & 38 & 28 & \\ \cdots & & & & \cdots \end{bmatrix}$
5	$\begin{bmatrix} 36 & 34 & 46 & 34 & \cdots \\ 34 & 32 & 43 & 32 & \\ 46 & 43 & 58 & 43 & \\ 34 & 32 & 43 & 32 & \\ \cdots & & & & \cdots \end{bmatrix}$

modes excluding 16x16-Intra prediction) becomes

$$Y = \text{round}([C_f] \cdot [X] \cdot [C_f^T] \cdot m(QP \bmod 6, n)) \cdot \frac{1}{2^{15+\lfloor QP/6 \rfloor}} \tag{2.21}$$

A method for background subtraction on video sequences encoded under H.264 Baseline profile settings is discussed in Chapter 3. The Main and High profiles additionally support transform and quantization for 8x8 blocks. This is discussed in the following section.

2.4.9 Transform and Quantization for 8x8 blocks

The High and Main profile of H.264 supports adaptive macroblock-level switching between 8x8 and 4x4 transform block size for the luminance component. The forward and inverse 8×8 transforms are developed in a similar way to the 4×4 integer transforms but with the following differences:

1) The core transform C_{f8}, C_{i8} is an 8-point integer transform that is numerically similar to an 8-point scaled DCT but cannot be produced exactly by scaling and rounding an 8-point DCT matrix, whereas the 4-point integer transform can be produced by scaling and rounding a 4-point DCT matrix.

$$C_{f8} = \begin{bmatrix} 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\ 12 & 10 & 6 & 3 & -3 & -6 & -10 & -12 \\ 8 & 4 & -4 & -8 & -8 & -4 & 4 & 8 \\ 10 & -3 & -12 & -6 & 6 & 12 & 3 & -10 \\ 8 & -8 & -8 & 8 & 8 & -8 & -8 & 8 \\ 6 & -12 & 3 & 10 & -10 & -3 & 12 & -6 \\ 4 & -8 & 8 & -4 & -4 & 8 & -8 & 4 \\ 3 & -6 & 10 & -12 & 12 & -10 & 6 & -3 \end{bmatrix}$$

2) The 8×8 transform processes have a larger dynamic range than the 4×4 processes. The forward scaling factor is $1/2^{22}$ and the inverse scaling factor is 1/28.

Similar to the case of 4x4 blocks, the matrix R_{f8} is formed by replacing the elements of C_{f8} by the reciprocal of corresponding norms, i.e.,

$$R_{f8} = \begin{bmatrix} 1/\sqrt{512} & 1/\sqrt{512} & 1/\sqrt{512} & 1/\sqrt{512} & \\ 1/\sqrt{578} & 1/\sqrt{578} & 1/\sqrt{578} & 1/\sqrt{578} & \\ 1/\sqrt{320} & 1/\sqrt{320} & 1/\sqrt{320} & 1/\sqrt{320} & \cdots \\ 1/\sqrt{578} & 1/\sqrt{578} & 1/\sqrt{578} & 1/\sqrt{578} & \\ & & \cdots & & \cdots \end{bmatrix}$$

The first quadrant is shown; the remaining quadrants are identical. The two-dimensional forward transform is obtained by applying C_{f8} to the rows and the columns of the 8×8 input

block. Hence the combined scaling matrix S_{f8} is computed as

$$S_{f8} = R_{f8} \bullet R_{f8}^{T} = \begin{bmatrix} 1/512 & 1/544 & 1/128\sqrt{10} & 1/544 & \cdots \\ 1/544 & 1/578 & 1/136\sqrt{10} & 1/578 & \\ 1/128\sqrt{10} & 1/136\sqrt{10} & 1/320 & 1/136\sqrt{10} & \\ 1/544 & 1/578 & 1/136\sqrt{10} & 1/578 & \\ \cdots & & & & \cdots \end{bmatrix}$$

In case of 8x8 inverse transform, the rescaling matrix V_{i8}, specified in the H.264 standard. It incorporates transform normalization S_{i8} and inverse quantization or rescaling. V_{i8} is approximately related to S_{i8} and Q_{step}, the quantizer step size QP, as follows:

$$V_{i8} \approx S_{i8} \cdot Q_{step} \cdot 2^{8} \tag{2.22}$$

To ensure consistent decoding behavior, V_{i8} is defined by the standard as a function of QP. Table 2.5 lists V_{i8} for the first six values of QP. Only the top-left quadrant of V is shown, the remaining three quadrants are identical. There are six unique values in each matrix V_{i8}. These are defined as a table ***v*** in the standard, for QP = 0 to QP = 5 as follows.

Table 2.6 The relation between QP and positions in table ***v*** as defined in the H.264 standard

QP	V_{m0}	V_{m1}	V_{m2}	V_{m3}	V_{m4}	V_{m5}
0	20	18	32	19	25	24
1	22	19	35	21	28	26
2	26	23	42	24	33	31
3	28	25	45	26	35	33
4	32	28	51	30	40	38
5	36	32	58	34	46	43

The positions v_{mr} map to the following positions in the matrix Vi8:

$$V_{i8} = \begin{bmatrix} v_{m0} & v_{m3} & v_{m4} & v_{m3} & v_{m0} & v_{m3} & v_{m4} & v_{m3} \\ v_{m3} & v_{m1} & v_{m5} & v_{m1} & v_{m3} & v_{m1} & v_{m5} & v_{m1} \\ v_{m4} & v_{m5} & v_{m2} & v_{m5} & v_{m4} & v_{m5} & v_{m2} & v_{m5} \\ v_{m3} & v_{m1} & v_{m5} & v_{m1} & v_{m3} & v_{m1} & v_{m5} & v_{m1} \\ v_{m0} & v_{m3} & v_{m4} & v_{m3} & v_{m0} & v_{m3} & v_{m4} & v_{m3} \\ v_{m3} & v_{m1} & v_{m5} & v_{m1} & v_{m3} & v_{m1} & v_{m5} & v_{m1} \\ v_{m4} & v_{m5} & v_{m2} & v_{m5} & v_{m4} & v_{m5} & v_{m2} & v_{m5} \\ v_{m3} & v_{m1} & v_{m5} & v_{m1} & v_{m3} & v_{m1} & v_{m5} & v_{m1} \end{bmatrix}$$

For a given QP, V_{i8} is obtained as:

$$V_{i8} = v(QP \bmod 6, n) \cdot 2^{\lfloor QP/6 \rfloor} \tag{2.23}$$

Where v(r,n) is the element of V_{i8} at *r*th row and *n*th column. V_{i8} and hence effective Qstep doubles as QP increases by 6. The effective quantizer step size Qstep can be estimated as given in Table 2.7.

Table 2.7 Estimated Q_{step} (8x8 blocks)

QP	Estimated $Q_{step} = V_{i8} / (S_{i8} \cdot 2^8)$, element by element division
0	0.6250
1	0.6875
2	0.8125
3	0.8750
4	1.0000
5	0.1250

To summarize, the forward quantization and scaling matrix M_{f8} is derived from S_{f8}, S_{i8} and V_{i8} as follows:

$$M_{f8} = \text{round}\left(\frac{S_{i8} \bullet S_{f8} \cdot 2^{30}}{V_{i8}} \right) \tag{2.24}$$

A method for background subtraction on video sequences encoded under low bit rate constraints in H.264 Main/High profile is discussed in Chapter 4.

2.5 High Efficiency Video Codec (HEVC) or H.265

2.5.1 Introduction

High Efficiency Video Coding (HEVC), also known as H.265, is a new video compression standard, developed by the Joint Collaborative Team on Video Coding (JCT-VC). The JCT-VC brings together image and video encoding experts from around the world, producing a single standard that is approved by two standards bodies;

1. ITU-T Study Group 16 – Video Coding Experts Group (VCEG) – publishes the H.265 standard as ITU-T H.265; and
2. ISO/IEC JTC 1/SC 29/WG 11 Motion Picture Experts Group (MPEG) – publishes the HEVC standard as ISO/IEC 23008-2.

The initial version of the H.265/HEVC standard was ratified in January, 2013. HEVC was developed with the goal of providing twice the compression efficiency of the previous standard, H.264 / AVC. Although compression efficiency results vary depending on the type of content and the encoder settings, at typical consumer video distribution bit rates HEVC is typically able to compress video twice as efficiently as AVC. End-users can take advantage of improved compression efficiency in one of two ways (or some combination of both);

1. At an identical level of visual quality, HEVC enables video to be compressed to a file that is about half the size (or half the bit rate) of AVC, or

2. When compressed to the same file size or bit rate as AVC, HEVC delivers significantly better visual quality.

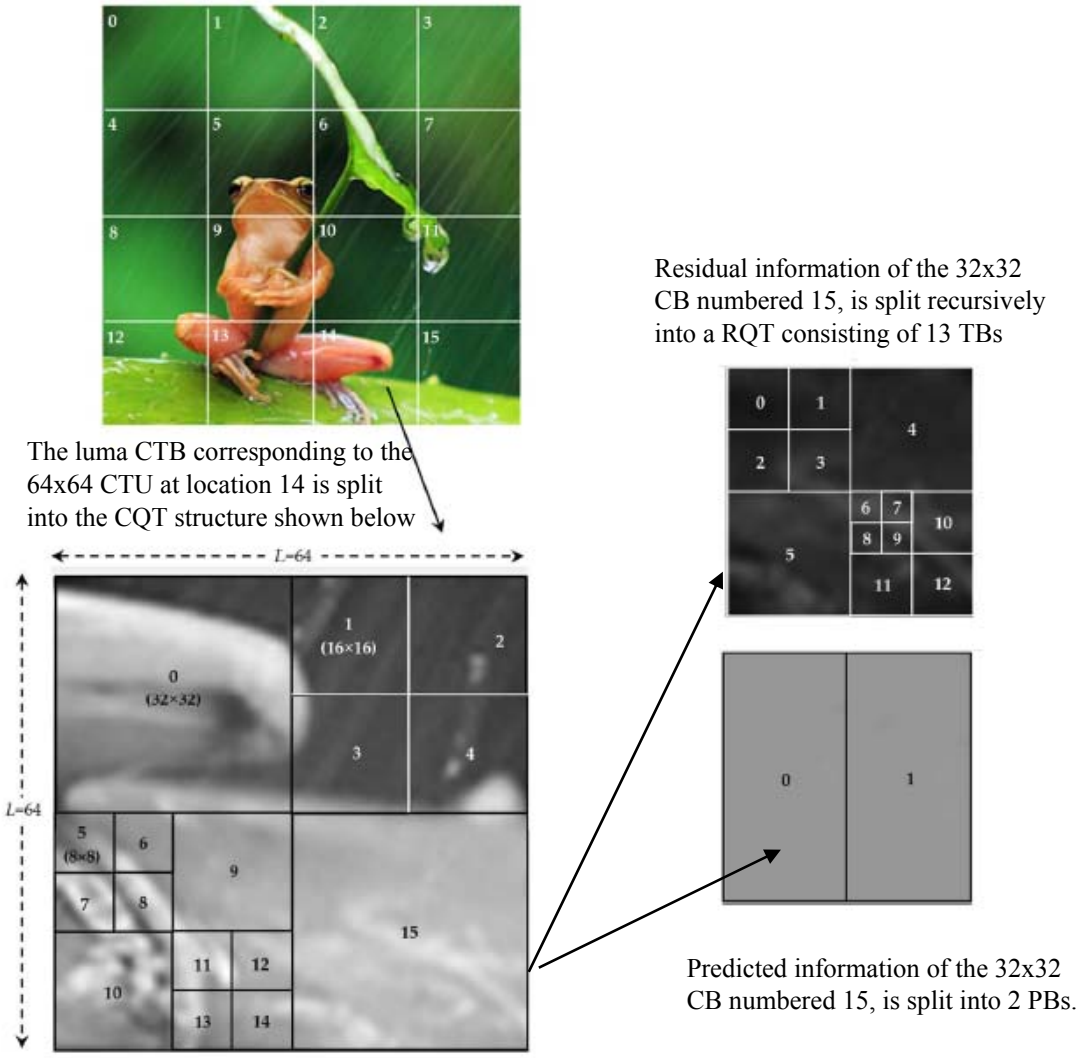


Fig. 2.6 An example of a 256 × 256 image partitioned into sixteen 64 × 64 CTUs (top image). The luma CTB corresponding to CTU at location 14 is split into a CQT structure. The CB number 15 of the CQT is shown to be coded using a predicted information consisting of 2 PBs and the prediction residual, which is split into an RQT consisting of 13 TBs.

Similar to H.264/AVC, HEVC also uses the YCbCr color space with 4:2:0 sampling structure. The sparse sampling of the chroma components, which enables relatively less data to be coded without noticeable difference. A compressed HEVC video consists of a sequence frames, each of which is split into non-overlapping blocks called coding tree unit (CTU). The CTU is the fundamental coding unit of compression, which maintains information for each color component in structures known as the coded tree block (CTB). A luma CTB covers a rectangular picture area of L × L samples of the luma component and the corresponding chroma CTBs cover L/2 × L/2 samples of each of the

two chroma components. The value of L (heretofore used to denote the size of luma CTBs) for a given sequence is fixed by the encoder and signaled by a sequence parameter which may be either equal to 16, 32, or 64. Each CTB can be split recursively into a quadtree structure, all the way down to 8×8 (in units of luma samples) regions. The quadtree structure is known as the coding quadtree (CQT). So, for the example shown in Fig. 1 (top), the 64×64 luma CTB corresponding to the CTU at location 14 is shown to consist of two 32×32, six 16×16, and eight 8×8 regions. These regions are called coding blocks (CBs). The spatial and the temporal redundancies respectively of a given CB are reduced by splitting it into blocks that were predicted from previously coded blocks within the same frame (called intra-prediction), as well as from the neighboring frames (called inter-prediction). The blocks are called prediction blocks (PBs). Inter-prediction of a PB is a temporal de-correlation technique by which one or two suitable reference blocks are selected. The reference blocks are indicated with offsets relative to the current block, in both horizontal and vertical directions. The prediction information of a PB is indicated as a motion vector corresponding to the offsets, and a reference frame index pointing to the referenced frame.

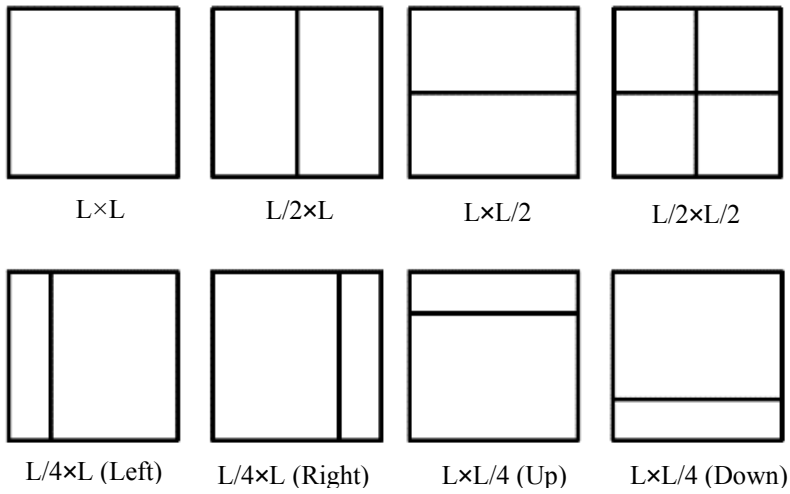


Fig. 2.7 Modes for splitting a CB into PBs, subject to certain size constraints. For intra-predicted CBs, only L×L and L/2×L/2 are supported

It may be noted that the inter-predicted CBs it is specified whether the luma and chroma CBs are split into one, two, or four PBs. The splitting into four PBs is allowed only when the CB size is equal to the minimum allowed CB size, using an equivalent type of splitting as could otherwise be performed at the CB level of the design rather than at the PB level. When a CB is split into four PBs, each PB covers a quadrant of the CB. When a

CB is split into two PBs, six types of this splitting are possible. The partitioning possibilities for inter-predicted CBs are depicted in Fig. 3. The upper partitions illustrate the cases of not splitting the CB of size L×L, of splitting the CB into two PBs of size L×L/2 or L/2×L, or splitting it into four PBs of size L/2×L/2. The lower four partition types in Fig. 3 are referred to as asymmetric motion partitioning, and are only allowed when L is 16 or larger. One PB of the asymmetric partition has the height or width L/4 and width or height M, respectively, and the other PB fills the rest of the CB by having a height or width of 3L/4 and width or height L. Each inter-predicted PB is assigned one or two motion vectors and reference picture indices. To minimize worst-case memory bandwidth, PBs of luma size 4×4 are not allowed for interpicture prediction, and PBs of luma sizes 4×8 and 8×4 are restricted to uni-predictive coding. The inter-prediction process is further described as follows. The luma and chroma PBs, together with the associated prediction syntax, form the prediction unit (PU).

The error in intra/inter-prediction of a CB, i.e., the residual, still contains correlation between spatially neighboring pixels. Therefore, the residual is further split into a quadtree structure, called the residual quadtree (RQT), into transform blocks of suitable sizes ranging from 4 × 4 to 32 × 32 samples. The TBs are subjected to transform coding and quantization. The purpose of transform coding is to decompose a batch of correlated signal samples into a set of uncorrelated spectral coefficients, with energy concentrated in as few coefficients as possible. This compaction of energy permits a prioritization of the coefficients, with the more energetic ones receiving a greater allocation of encoding bits. The resulting transform coefficients are quantized. This results in rounding of coefficient values to some acceptable unit of precision specified by a quantization parameter. The output of a quantizer is typically a sparse array of quantized coefficient levels, mainly containing zeros. Finally, the non-zero coefficient levels are entropy coded. In chapter 5, a novel approach for background subtraction in bitstreams encoded in the HEVC is proposed.

Chapter 3

Feature Extraction for Background Subtraction from Compressed Video in H.264 Baseline Profile

3.1 Introduction

The baseline profile of H.264/AVC is the simplest and most commonly used among all the profiles. It was primarily targeted for devices with low computing capability. Besides the interoperability among a wide range of devices, it is also faster to encode and decode, i.e., lower latency. Latency is the total time it takes to encode, transmit, decode and display the video at the destination. Interactive video applications require that latency be extremely small. However, in terms of compression the baseline profile far less efficient compared to other profiles, *i.e.*, Main and High. In the previous chapter, detailed discussion was presented regarding the preliminaries related to macroblock prediction, transform coding and quantization for compression of uncompressed video signal to H.264 baseline profile.

Although there exist several techniques for moving object segmentation in MPEG domain, algorithms that work on H.264 compressed video are relatively few. Thilak and Cruesere [111] presented a system to track targets in H.264 video, which relied heavily on the prior knowledge of the size of the target. Zeng et al. [112] proposed an algorithm to segment moving objects from the sparse motion vector field using block based Markov random field (MRF). The limitation of this approach is that it is only applicable to video sequences with stationary background. Liu et al. [113] used complex binary partition tree to segment normalized motion vector field into motion-homogeneous regions. The complexity of the algorithm increases drastically with a noisy motion vector field. Solana-Cipres et al. [114] incorporated fuzzy linguistic concepts that use motion vector and macroblock decision modes. Fei and Zhu [115] introduced mean shift clustering using normalized motion vector field and partitioned block size for moving object segmentation. This algorithm fails to detect objects with slow or intermittent motion. W. You et al. [116] proposed a new probabilistic spatio-temporal macroblock filtering (PSMF) and linear motion interpolation for object tracking in P-frames. The linearity assumption holds good for group of picture (GOP) sizes less than ten frames and slow moving targets. Most related techniques that work in H.264 compressed domain are based solely on the motion vector field. Motion vectors, however, do not necessarily correspond to true object motion and such related techniques end up wrongly classifying the dynamic components (eg. waving tree branches, fountain, ripples, camera jitter etc.) as foreground. Poppe et al. [117] proposed an alternative technique that relied on the total number of bits required to encode

a given macroblock. This technique fails to detect slow-moving objects.

In this chapter, a novel feature vector is proposed that effectively describes macroblock data in H.264 Baseline video. The temporal statistics of feature vectors are used to select a set of potential macroblocks that are occupied fully or partially by moving objects. From the set of coarsely localized candidate macroblocks, foreground pixels are detected by comparing corresponding pixel colors pair-wise with a background model. The proposed method is embedded into the decoding process and obtains pixel-level segmentation at the cost of a negligible overhead. A common drawback of the foreground extraction approaches on H.264 is the assumption of a fixed quantization parameter (QP) for all macroblocks which limit them to variable bit rate (VBR) applications with uncontrolled bit rate. However, in most streaming video applications, a predetermined constant output bit rate is desired. These applications, referred to as constant bit rate (CBR) applications, ensure a target bit rate by carefully selecting a different QP for each macroblock. Unlike the related approaches, our algorithm allows each macroblock to have a different QP in consideration of the stringent bandwidth requirements of a practical surveillance scenario. Secondly, filtering is performed on the aggregate result of macroblock-level object segmentation obtained jointly from the motion vector field and the residual data rather than applying separate filtering procedures on either or both of the features in isolation. This helps us to reduce computation while relying more on the consensus between the macroblock features.

As discussed earlier, an encoded video sequence information in which all redundancies have been minimized. In other words, all incremental differences which occur from one frame to the next are encoded in the form of motion vectors and transform coefficients. Our proposed method entails quantification of the encoded information as macroblock features. Two macroblock features are identified; the first one corresponds to the mean absolute value of the transform coefficients of a macroblock; while the other corresponds to the sum of motion vector magnitudes. In the following sections, computation of macroblock features is described. The algorithm for background segmentation is described later.

3.2 Macroblock feature #1: Mean of Absolute Transform Differences (MATD)

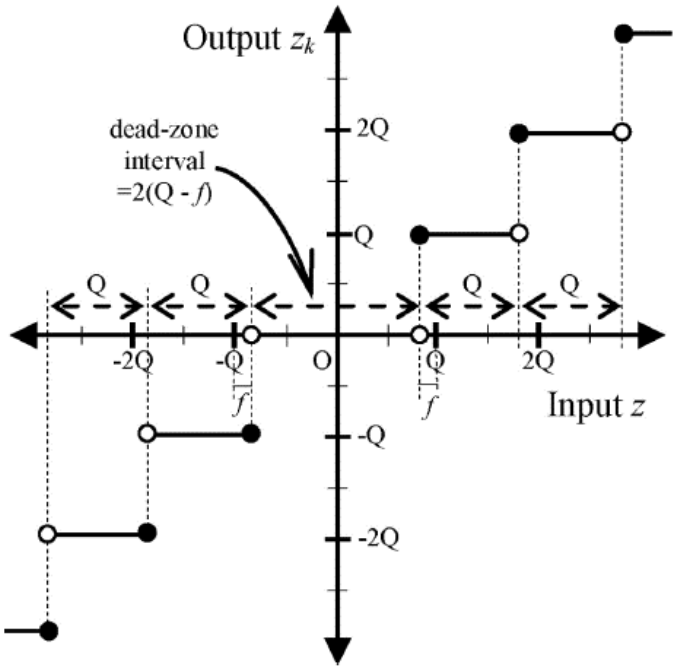


Fig. 3.1 Relation between the input coefficient z and the reconstructed output z_k for quantization step size Q and rounding offset *f*.

As the name indicates, MATD or Mean of absolute transform differences is the mean absolute value of the sum of quantized DCT coefficients in a coded macroblock. MATD is formulated using the statistical properties of DCT coefficients. In literature, it is found to be most appropriate and convenient to model the distribution of DCT coefficients by Laplacian distributions [125]–[127]. A Laplacian probability density function (pdf) is given by

$$p(z) = \frac{1}{2b}\exp\left(-|z|/b\right), z \in \mathbb{R} \tag{3.1}$$

where $b > 0$ is the Laplacian parameter which defines the width of the pdf and z being the value of a given DCT coefficient.

$$k = \lfloor (|z| + f)/Q \rfloor \mathrm{sgn}(z), \tag{3.2}$$

where k ∈ Z represents the quantization level or index that is actually transmitted by the source encoder, Q>0 is the uniform quantization step-size except the dead-zone interval, and $f \in [0, Q/2]$, is the rounding offset that controls the width of the dead-zone [128]. Typically, f is set to be Q/6 for P-frames in an effort to approximate the distribution of DCT coefficients in a quantization interval. The decoder process reconstructs the quantized output z_k of the input coefficient z as

$$z_k = kQ. \tag{3.3}$$

Fig. 3.1 illustrates the relationship between z and z_k. Formally, it may be verified that the value z of any input coefficient is mapped to kQ in the quantization process, such that

1) k=0, if $z \in (-Q+f, Q-f)$;
2) $k=-1, -2, -3, \ldots$ if $z \in \left(\left(k-1+\frac{f}{Q}\right)Q, \left(k+\frac{f}{Q}\right)Q\right]$;
3) $k=1, 2, 3, \ldots$ if $z \in \left[\left(k-\frac{f}{Q}\right)Q, \left(k+1-\frac{f}{Q}\right)Q\right)$.

Let $P(z_k)$ be the probability that z is mapped to z_k. Using equation (3.1) and substituting for f (=Q/6) in the limiting expressions of above intervals, we get

$$P(z_k)=\begin{cases}\int_{\left(k-\frac{5}{6}\right)Q}^{\left(k+\frac{1}{6}\right)Q} p(z)dz=\exp\left(-2r(1-3k)\right)\sinh 3r & \text{for } k<0\\ \int_{-\frac{5}{6}Q}^{\frac{5}{6}Q} p(z)dz=1-\exp\left(-5r\right) & \text{for } k=0\\ \int_{\left(k-\frac{1}{6}\right)Q}^{\left(k+\frac{5}{6}\right)Q} p(z)dz=\exp\left(-2r(1+3k)\right)\sinh 3r & \text{for } k>0\end{cases} \tag{3.4}$$

where

$$r=Q/6b. \tag{3.5}$$

The H.264 standard does not specify Q directly for each coefficient separately but rather uses a quantization parameter QP, whose relationship to Q for each 4×4 block of DCT coefficients of a macroblock is given by the quantization matrix:

$$Q=\begin{bmatrix} q(QP,0) & q(QP,2) & q(QP,0) & q(QP,2)\\ q(QP,2) & q(QP,1) & q(QP,2) & q(QP,1)\\ q(QP,0) & q(QP,2) & q(QP,0) & q(QP,2)\\ q(QP,2) & q(QP,1) & q(QP,2) & q(QP,1)\end{bmatrix}, \tag{3.6}$$

where

$$q(QP,n)=m\left(\text{mod}(QP,6),n\right)2^{\lfloor QP/6\rfloor} \tag{3.7}$$

m being a scalar multiplier as defined earlier in Table-2.4. It is noted in (3.6) that exactly 1/2 of the transformed coefficients are quantized with a step size equal to q(QP,2), and the remaining with q(QP,0) and q(QP,1) equally. Thus, r can take only three possible values $\frac{q(QP,0)}{6b}$, $\frac{q(QP,1)}{6b}$, and $\frac{q(QP,2)}{6b}$ for given values of QP and b.

Given the fact that the quantized coefficients $\{z_k\}$ of a macroblock (both I- and P- types) are entropy coded, the lower bound on the average bit rate (bits/coefficient) may be expressed as

$$H=-\sum_{k=-\infty}^{+\infty} P(z_k)\log_2 P(z_k).$$

Using (3.4)-(3.6), the above expression simplifies to

$$H(r)=\frac{\exp(-2r)}{\sinh 3r\ln 4}\left(6r+\left(1-\exp(-6r)\right)\left(2r-\ln\sinh 3r\right)\right)-\left(1-\exp(-5r)\right)\log_2\left(1-\exp(-5r)\right). \tag{3.8}$$

Total bits *B* required in coding the DCT coefficients of a macroblock may be computed as the product of H and the total number of DCT coefficients. For sequences encoded in the YCbCr 4:2:0 baseline format, a macroblock is represented by 256 luminance (Y), 64 red chrominance (Cr) and 64 blue chrominance (Cb) samples, giving a total of 384 samples. Therefore, we have

$$\begin{aligned} B &= 384\left(\tfrac{1}{4}H\left(\tfrac{q(QP,0)}{6b}\right)+\tfrac{1}{4}H\left(\tfrac{q(QP,1)}{6b}\right)+\tfrac{1}{2}H\left(\tfrac{q(QP,2)}{6b}\right)\right) \\ &= 96H\left(\tfrac{q(QP,0)}{6b}\right)+96H\left(\tfrac{q(QP,1)}{6b}\right)+192H\left(\tfrac{q(QP,2)}{6b}\right). \end{aligned} \tag{3.9}$$

Table-2.4 is used to express quantization step sizes $\frac{q(QP,1)}{6b}$ and $\frac{q(QP,2)}{6b}$ as scalar multiples of $\frac{q(QP,0)}{6b}$, so that the right-hand side of (3.9) may be expressed in one unknown. For known values of B and QP (obtained from the macroblock header), numerical methods are used to evaluate as it is difficult to derive an exact closed form solution by analytical means. The statistical prediction of MATD for the current macroblock is formulated as the mean of absolute values of z_k.

$$MATD=\sum_{k=-\infty}^{\infty}|z_k|P(z_k)=2\sum_{k=1}^{\infty}|z_k|P(z_k) \tag{3.10}$$

Using (3.4)-(3.6), the above expression of predicted MATD simplifies to the form shown in (3.10)

$$MATD=\frac{q(QP,0)\exp\left(-\frac{q(QP,0)}{3b}\right)}{8\sinh\left(\frac{q(QP,0)}{2b}\right)}+\frac{q(QP,1)\exp\left(-\frac{q(QP,1)}{3b}\right)}{8\sinh\left(\frac{q(QP,1)}{2b}\right)}+\frac{q(QP,2)\exp\left(-\frac{q(QP,2)}{3b}\right)}{4\sinh\left(\frac{q(QP,2)}{2b}\right)}. \tag{3.11}$$

Substituting the values of $\frac{q(QP,0)}{6b}$ and its scalar multiples $\frac{q(QP,1)}{6b}$ and $\frac{q(QP,2)}{6b}$ in (3.10), finally MATD is obtained. Using the fact that B and QP can only assume non-negative integer values from a limited range, we construct a lookup table containing pre-computed values of MATD (indexed by B and QP). This enables us to bypass entropy decoding and de-quantization of individual coefficients which would otherwise be required to compute MATD directly, i.e., actual MATD. The correlation between the actual MATD and the predicted MATD is explored later in the experimental section.

3.3 Sum of Normalized Motion Vector Magnitudes (SNMVM)

SNMVM represents the motion compensated information associated with the motion vectors of a macroblock. As mentioned in chapter 2, two new coding characteristics are introduced in H.264 motion compensation: variable partition size and multiple reference frames. Each inter coded macroblock is predicted using a range of block sizes. Accordingly, the macroblock is split into one, two or four macroblock partitions. If the partition size is chosen as 8×8, then each 8×8 block, heretofore a sub-macroblock, is split into one, two or four sub-macroblock partitions (either one 8×8, two 4×8, two 8×4 or four 4×4 sub-macroblock partitions).

Each partition or sub-macroblock partition of a P-macroblock has a motion vector (mvx_i, mvy_i) pointing to an area of the same size in a reference frame, which is used to predict the current (say ith) partition. Each partition in a given macroblock may be predicted from different reference frame(s). However, the sub-macroblock partitions within an 8×8 sub-macroblock share the same reference frame. Let us denote the reference index of the ith partition of a macroblock as f_i. A P-frame having frame number t is predicted from a list of previous frames where reference index 0 denotes frame (t-1), reference index 1 denotes frame (t-2), and so on. In order to obtain uniformity among the partitions we normalize each of them by f_i and assign fixed weights according to the ratio of the macroblock area it represents. The weight w_i of the ith partition is defined as the fraction of the partition size contributing to the total macroblock area, i.e., 256. Assuming a total of p partitions in the current macroblock, the computation process of SNMVM is described in Fig. 3.2. It may be noted that the value of SNMVM for an I-macroblock in P-frame is taken as zero, as they do not contain any motion compensated information. The computation in this step involves four multiplications / divisions and four additions for each partition, the total number of partitions being no more than 16 for any given macroblock.

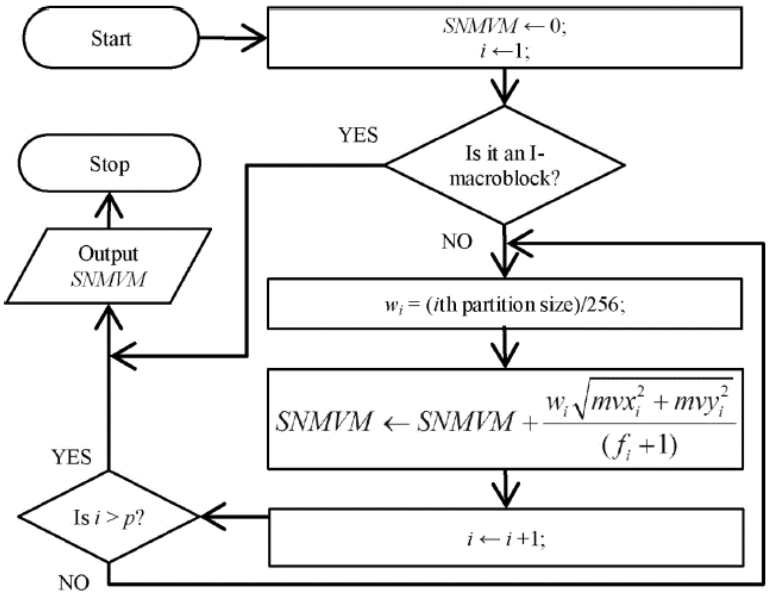


Fig. 3.2 Flowchart for computation of SNMVM.

3.4 The Macroblock feature vector

Formally, let T be the total number of macroblocks in each frame. macroblocks are numerically addressed using a macroblock index $idx \in \{0,1,\ldots,T-1\}$ in raster-scan order starting with zero for the macroblock at the top-left hand corner of a frame. A P-frame macroblock having frame number t at location idx is described by a vector

$$\vec{p}_{t,idx} = [x, y]^T, \tag{3.12}$$

where components $x \geq 0$ and $y \geq 0$ are the values of MATD and SNMVM respectively for the given macroblock.

3.5 Initialization and Incremental Update of Covariance

In the proposed approach, static and dynamic components present at different locations in a scene background are modeled with a temporally weighted local covariance $idx \in \{0,1,\ldots,T\text{-}1\}$. Feature vectors computed for each idx are stored in T corresponding fist-in first-out (FIFO) buffers, each having a predefined capacity M. A new macroblock vector is inserted at the rear. However, if the buffer is full, the vector at the front, being the least relevant in the temporal order of vectors in the buffer, is removed to make room for a new one. Corresponding to each position $j \in \{1(\text{front}),2,3,\ldots, M(\text{rear})\}$ in a buffer containing a vector, predefined weight $W_j=j/M$ is assigned in order of temporal relevance. Without loss of generality, let us assume that buffer[idx] is in a state in which all n vectors from the

sequence $\{[x_i, y_i]^T\}_{i=1}^{n}$ are inserted, with the nth vector currently at the rear. If $n > M$, this would have resulted in the removal of previous $(n-M)$ vectors. Let σ_x^2 and σ_y^2 denote respective weighted variances of $\{x_i\}_{i=\max(1,n-M+1)}^{n}$ and $\{y_i\}_{i=\max(1,n-M+1)}^{n}$ currently accumulated in the buffer. Also, let σ_{xy} denote the covariance between the same set of values. Accordingly, the required covariance matrix Σ_{idx} is expressed as in (3.12).

$$\sum_{idx} = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix}. \tag{3.13}$$

Weighted variance σ_x^2 is defined as

$$\sigma_x^2 = \frac{\sum_{i=\max(1,n-M+1)}^{n} W_{M-n+i}\left(x_i - \bar{x}\right)^2}{\sum_{i=\max(1,n-M+1)}^{n} W_{M-n+i}} = \left(\overline{x^2} - \bar{x}^2\right). \tag{3.14}$$

where

$$\bar{x} = \sum_{i=\max(1,n-M+1)}^{n} W_{M-n+i} x_i \Big/ \sum_{i=\max(1,n-M+1)}^{n} W_{M-n+i} \tag{3.15}$$

and

$$\overline{x^2} = \sum_{i=\max(1,n-M+1)}^{n} W_{M-n+i} x_i^2 \Big/ \sum_{i=\max(1,n-M+1)}^{n} W_{M-n+i}. \tag{3.16}$$

Similarly, we have

$$\sigma_y^2 = \left(\overline{y^2} - \bar{y}^2\right). \tag{3.17}$$

and

$$\sigma_{xy} = \left(\overline{xy} - \bar{x}\bar{y}\right). \tag{3.18}$$

where

$$\bar{y} = \sum_{i=\max(1,n-M+1)}^{n} W_{M-n+i} y_i \Big/ \sum_{i=\max(1,n-M+1)}^{n} W_{M-n+i}, \tag{3.19}$$

and

$$\overline{xy} = \sum_{i=\max(1,n-M+1)}^{n} W_{M-n+i} x_i y_i \Big/ \sum_{i=\max(1,n-M+1)}^{n} W_{M-n+i}. \tag{3.20}$$

Direct computation of Σ_{idx} using (3.13), (3.16) and (3.17) following each insertion would be computationally prohibitive and inefficient as most of the samples in the buffer remain unaltered between subsequent insertions. Hence, recursive counterparts of (3.14), (3.15), (3.18), (3.19), and (3.20) are formulated in order to facilitate online update of Σ_{idx}. The recursive update of $\bar{x}$ is considered as follows. Let $[S_x]_{n-1}$ denote the sum of x-

components in the buffer prior to insertion of the nth vector. If $1 < n \leq M$, insertion of the nth vector at the rear causes all other entries in the buffer to be shifted by one place towards the front. Otherwise ($n > M$), the same insertion process will additionally cause the (n-M)th vector to be removed from the front. In either case, the insertion operation causes the existing sum of x-components to decrease by $([S_x]_{n-1}/M)$ and increase by x_n. The adjusted sum is finally normalized by the sum $[W]_n$ of predefined weights to obtain $\overline{x}$. Equations (3.21) and (3.22) summarize the initialization and recursive update process of Σ_{idx} for the nth insertion.

$$\begin{bmatrix} \overline{x^a y^b} \\ S_{x^a y^b} \end{bmatrix}_n = \begin{cases} \left[x_1^a y_1^b,\ x_1^a y_1^b \right]^{\mathrm{T}}; & \text{if } n = 1 \\ \begin{bmatrix} \dfrac{\overline{x^a y^b} W - (S_{x^a y^b}/M) + x_n^a y_n^b}{W + W_{M-n+1}} \\ S_{x^a y^b} + x_n^a y_n^b \end{bmatrix}_{n-1}; & \text{if } 1 < n \leq M \\ \begin{bmatrix} \dfrac{\overline{x^a y^b} W - (S_{x^a y^b}/M) + x_n^a y_n^b}{W} \\ S_{x^a y^b} - x_{n-M}^a y_{n-M}^b + x_n^a y_n^b \end{bmatrix}_{n-1}; & \text{if } n > M. \end{cases} \tag{3.21}$$

where ordered pair (a,b) ∈ {(1,0), (2,0), (0,1), (0,2), (1,1)}. Similarly, the recursive counterparts of (3.15), (3.18), (3.19) and (3.20) may be obtained from (3.21) by substituting for (a,b) each of the ordered pairs (2,0), (0,1), (0,2), and (1,1) respectively. The update procedures of $\overline{x}$, $\overline{y}$, $\overline{x^2}$, $\overline{y^2}$, and $\overline{xy}$ are followed by an adjustment of $[W]_n$ as

$$[W]_n = \begin{cases} 1; & \text{if } n = 1 \\ [W + W_{M-n+1}]_{n-1}; & \text{if } 1 < n \leq M \\ [W]_{n-1}; & \text{if } n > M. \end{cases} \tag{3.22}$$

It may be noted that, the overall process of updating Σ_{idx} requires no more than 14 multiplications, ten divisions, and 21 additions.

3.6 Proposed Method

The proposed method operates at two levels of granularity as follows:

1. performing a coarse macroblock-level segmentation of each frame by selecting of a set of potential macroblocks that are occupied fully or partially by parts of a moving object; and
2. performing a finer pixel-level segmentation of the selected macroblocks by eliminating pixels that are similar to the corresponding background model.

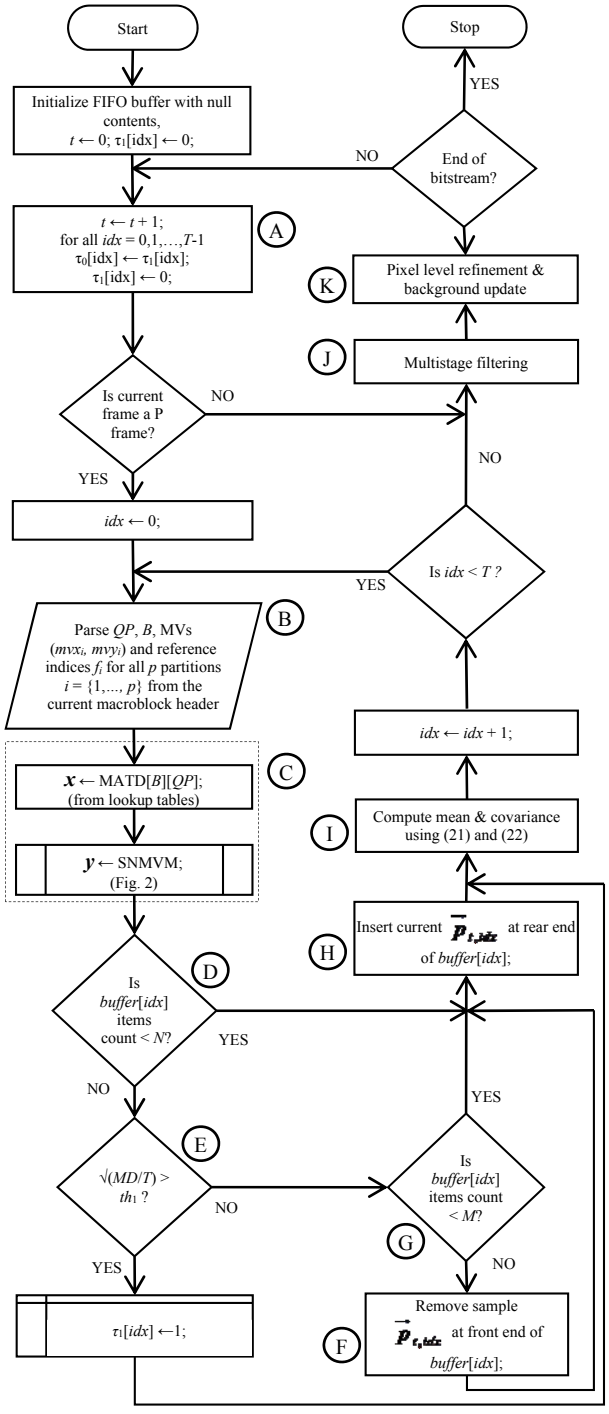


Fig. 3.3 Flowchart of the proposed method.

The flowchart of the proposed method is shown in Fig. 3.3. In order to compute temporal covariance of each macroblock in a frame, T identical FIFO buffers (indexed by *idx*) are used. The macroblock-level binary segmentation of the current and the previous frame are stored in arrays $\tau_1[0: T-1]$ and $\tau_0[0: T-1]$ respectively. Before parsing a new frame, the contents of τ_1 are copied to τ_0 while those of τ_1 are initialized to zero (block A).

As usual, frame numbers are denoted by *t*. Parameters B, QP and motion vectors, which are parsed from the current macroblock header (block B) are used to compute its feature vector representation as described in sections 3.1 and 3.2 (block C).

Using the first *N* (*N*<*M*, say 100) frames, a median background frame is computed (explained later). At the same time, feature vectors *idx* ∈ {0,1,…,*T*−1} are inserted (block-

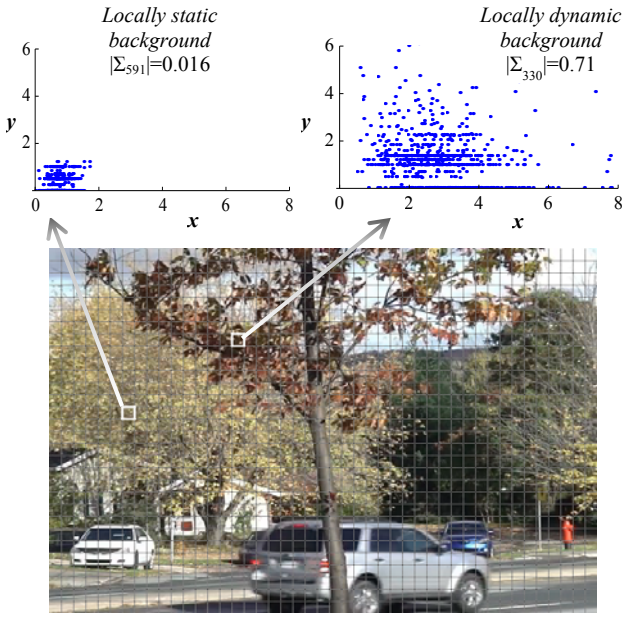


Fig. 3.4 Modeling background variance in the proposed feature space.

H) into the corresponding buffers, i.e., buffer[*idx*]. Temporally weighted mean and covariance Σ_{idx} are computed online (block I) using (3.21) and (3.22). Buffered feature vectors for a given macroblocks are characteristics of locally varying background model. If the total number of vectors in a buffer at given *idx* reaches *N* (block D), any incoming vector $\vec{p}_{t,idx}$ is considered as a candidate for further insertion, based on a criteria set as

$$\tau_1[idx] = \begin{cases} 1 \text{ (foreground candidate);} & \text{if } \sqrt{MD/T} > th_1 \\ 0 \text{ (background);} & \text{otherwise} \end{cases} \tag{3.23}$$

where $MD = \left(\vec{p}_{t,idx} - \vec{\mu}_{idx}\right)^T \Sigma_{idx}^{-1} \left(\vec{p}_{t,idx} - \vec{\mu}_{idx}\right)$ is the Mahalanobis distance and th_1=0.225 being a predetermined threshold. If $\tau_1[idx]$=0, is inserted into buffer[*idx*]; otherwise, the current macroblock is selected as one of the probable candidates expected to contain part(s) of a foreground object. Should the buffer be already full (block G), the vector at the front is removed (block F) prior to the insertion of (block H) at the rear. The coarse macroblock-level segmentation so obtained, is filtered (block J) using the procedure described in Section 3.7. Pixels constituting the set of filtered macroblocks are further used

to obtain precise pixel-level segmentation (block K), details of which are presented in section 3.9.

Fig. 3.4 shows the scatter plots of the buffered feature vectors corresponding to different macroblock locations (highlighted in white) at 330 and 591 for the frame t=1896 from the Fall sequence. The location at 591 depicts stationary background. As shown in the corresponding scatter diagram, this is modeled by a highly dense cluster with low $|\Sigma_{591}|$. The same sequence also portrays highly dynamic component (waving tree branches) in the background at location 330, which is characterized by a sparsely distributed scatter with a higher value of $|\Sigma_{330}|$.

3.7 Multi-stage filtering for noisy macroblocks

In (3.23), the macroblock-level binary decision $\tau_1[\text{idx}]$ for all $\text{idx} \in \{0,1,\ldots,T-1\}$ may erroneously misclassify some macroblocks as foreground that do not correspond to true object motion, i.e., false positives and vice versa, i.e., false negatives. In order to reduce misclassification of macroblocks, multi-staged filter banks are used in which the output of a 3×3×2 spatio-temporal median filter (STMF) is cascaded to a 3×3 Gaussian filter. For frames of width u macroblocks and height of v macroblocks as shown in Fig. 3.5, the spatio-temporal support for the candidate macroblock at idx in frame t is highlighted with shaded blocks. The output of the STMF is computed as the median of $\tau_0[idx]$, $\tau_1[idx]$, $\tau_1[idx-1]$, $\tau_1[idx+1]$, $\tau_1[idx-u]$, and $\tau_1[idx+u]$.

The STMF often erroneously removes macroblocks containing boundaries of slow-moving objects. In addition, the output of STMF may occasionally contain small holes or gaps in the segmented regions that correspond to very large moving objects (comprising a half of the entire frame or more). Consequently, Gaussian filter is applied on the STMF output to obtain a smooth segmentation mask at the macroblock-level.

It may be noted that an optimized algorithm for computation the median (of six binary values) would require no more than five comparisons. The symmetric Gaussian filter additionally requires three multiplications and eight additions.

3.8 Color Space and Comparison technique used

In the proposed method, a low-complexity color comparison technique is used to determine if two given colors are similar or different. Considering a given pair of pixels

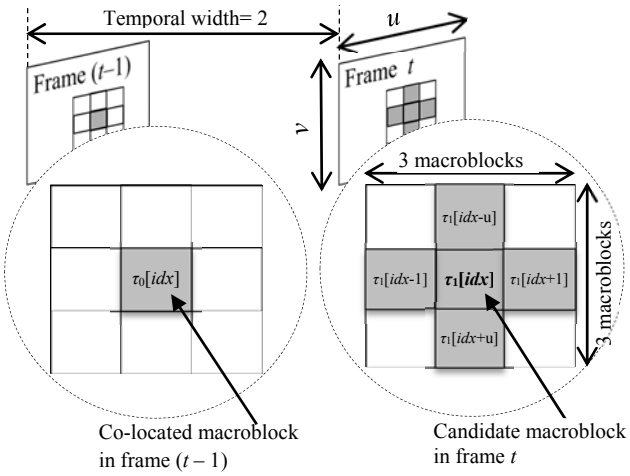


Fig. 3.5 Illustration of 3×3×2 support for the STMF (enlarged insets).

with YCbCr color co-ordinates $X_F \equiv (Y_F, Cb_F, Cr_F)$ and $X_B \equiv (Y_B, Cb_B, Cr_B)$ respectively from the current and the background frame, let luminance differential $\Delta Y = |Y_F - Y_B|$ and chrominance differential $\Delta C = |Cb_F - Cb_B| + |Cr_F - Cr_B|$. Decision thresholds $t_Y = C_1 + C_2|\Sigma_{idx}|$ for luminance and $t_C = C_3 + C_4|\Sigma_{idx}|$ for chrominance are used which scale linearly with $|\Sigma_{idx}|$ for a given macroblock. Constants $C_1=18$, $C_2=81$, $C_3=0.5$, and $C_4=4.2$ are empirically set for dynamic background sequences. X_F and X_B are considered different and the pixel corresponding to X_F is classified as foreground if $\Delta Y > t_Y$ and $\Delta C > t_C$. This is in contrast to most existing techniques where the similarity of pixels is determined based on the Euclidean (or straight-line) distance between a given pair of points in RGB color space. It is obvious that the proposed color comparison involves fewer computations (using additions and subtractions only) than those of Euclidean distance based techniques. This method benefits from the inherent advantage of YCbCr color space, *i.e.*, decoupling of the luminance (or brightness) and the chrominance (or color) signals which are perceived independently by the human visual system. It may be noted that the CIELUV color space is widely known to be robust to illumination variations. Hence, using a similar method, we obtained results using both the YCbCr and the CIELUV (or LUV for short) color space.

3.9 Background Model Initialization & Update

In order to obtain foreground-background classification of the pixels of an input frame, a background model is initialized and continually updated in order to incorporate gradual changes in the scene. For initialization of the background model, the first *N* frames of the input sequence are divided into ten equal-sized groups, from each of which one frame is

randomly picked or sub-sampled for computation of a temporal median frame. This frame is used as the initial background model corresponding to frame t=N+1. For subsequent frames (i.e., $t > N$), pixels constituting the set of (filtered) candidate macroblocks in the current frame are compared with the corresponding pixels in the background model. If the pixels are found to be different (as discussed in the previous section), the corresponding pixel in the current frame is labeled as foreground. The remaining pixels in the frame constitute the background. This produces a precise pixel-level segmentation of the current frame. If the number of pixels labeled as foreground exceeds ~20% of the total pixels in a given macroblock, the pixel-level comparison process is repeated for the co-located macroblock in the following frame. This typically enforces inter-frame continuity of object segmentation masks. All macroblocks in a frame corresponding to which $\tau_1[idx]$=0 and $\sqrt{(MD/T)} \in (0.02, 0.12)$, represent the background. Pixels constituting such macroblocks are used to update the corresponding pixel $B_t(x, y)$ in the existing background using (3.24). The number of such macroblocks in a given frame, say β, is practically very small.

$$B_{t+1}(x,y) = \alpha I_t(x,y) + (1-\alpha) B_t(x,y) \text{ for } t > N \tag{3.24}$$

where $I_t(x, y)$ is the pixel's intensity value in frame t; α=0.08 is a predefined learning rate that determines the tradeoff between stability and quick update.

3.10 Results & Discussion

Following the discussion in section 3.2, we provide statistical comparison of the actual MATD against the predicted MATD values of all P-frame macroblocks selected at regular intervals from Traffic sequence. The sequence was encoded in VBR as well as constant bit rate (CBR) modes as reported in Fig. 3.6. The values of actual MATD are found to be greater than the corresponding predicted MATD values, owing to fact that the latter is modeled using the entropy criterion, which is the theoretical lower bound on the average bitrate. It is observed that the actual MATD and the predicted MATD values are very highly correlated (correlation coefficient $\rho > 0.98$). In (3.23), we used the Mahalanobis distance, which is invariant under arbitrary non-singular linear transformations of the form shown in Fig. 3.6 (with regression equations). Thus, predicted MATD qualifies for a convenient surrogate to actual MATD insofar as the discriminative aspect of MATD is concerned.

The proposed algorithm was implemented in C and integrated into H.264/AVC macroblock decoding module of Libavcodec an open source audio/video codec library that

is available as a part of the FFmpeg [129] project. We evaluate our approach on the entire benchmark dataset provided for the Change Detection challenge (CD.net) 2012 [130] and the Background Models Challenge (BMC) dataset. As the proposed method uses pixel information to achieve pixel-level accuracy in the final stage of segmentation, we consider it fair and obvious to compare our results with those of proven state-of-the-art pixel-based approaches [42], [131]-[134]. All 31 sequences of the dataset were encoded in CBR mode with a target bit rate of 1024kb/s. The encoder configuration was set as follows: Baseline profile with YCbCr 4:2:0 (progressive format) 8-bit chroma subsampling, GOP size varying in [1, 250], rate-distortion optimization enabled, and the range of motion vectors (using hexagon-based search) was [-16…16] × [-16…16] for three reference frames. The rate of decoding frames was fixed at 25 frames per second (fps).

The background subtraction masks of the proposed method for a few selected frames are shown in Fig. 3.7 to enable qualitative evaluation against the specified ground-truth masks. Results obtained using both YCbCr and CIELUV (LUV) color spaces for the

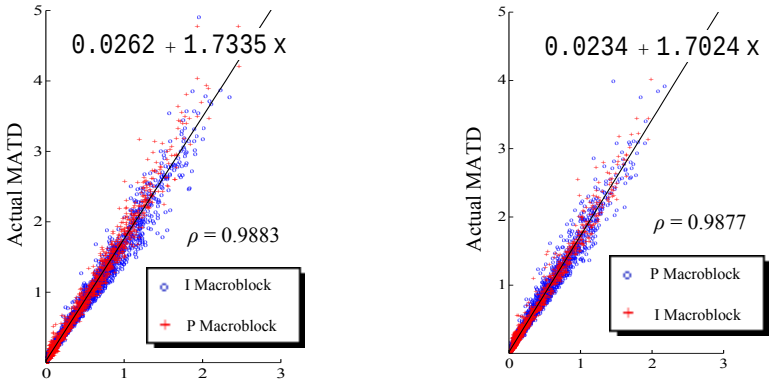


Fig. 3.6 Scatterplot of Actual MATD vs. Predicted MATD (with regression lines by the method of least squares). (a) VBR with fixed QP=25. (b) CBR at 1024kb/s.

sequences encoded under CBR constraints are shown. For quantitative evaluation, a set of seven evaluation metrics defined in [130] such as

$$\text{Recall (Re)} = \#\text{TP}/\left(\#\text{TP} + \#\text{FN}\right), \tag{3.25}$$

$$\text{Precision (Pr)} = \#\text{TP}/\left(\#\text{TP} + \#\text{FP}\right), \tag{3.26}$$

$$\text{F-measure} = 2 \times \text{Pr} \times \text{Re}/\left(\text{Pr} + \text{Re}\right), \tag{3.27}$$

and average processing speed (in frames per second). The notations #TP, #FP, and #FN are the total number of true positives, false positives, and false negatives (in terms of pixels) respectively. An exhaustive comparison of the proposed method with those of [42],

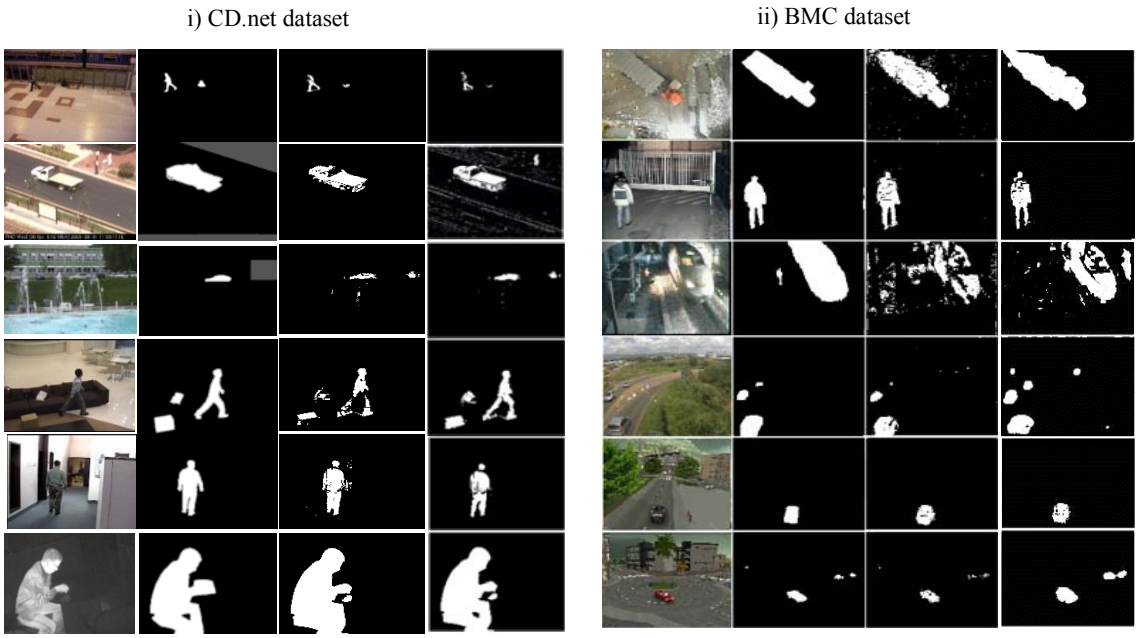


Fig. 3.7 Each row shows one example from a sequence selected form the dataset. The left column shows segmentation results from the CD.net dataset, while those in the right are from the BMC dataset. The sequences from the CD.net dataset (top to bottom) are highway, boulevard, fountain01, sofa, cubicle, and library. The sequences from the BMC dataset (top to bottom) are Video002, Video004, Video007, Video008 (from Real Video category), 512, and 522 (synthetic video category).

[131]-[134] (applied on original input frames prior to encoding with default parameters defined in each work) is summarized in Table-3.2. Subscripts indicate rank of the corresponding figures in the indicated evaluation category. It may be noted that all evaluation metrics were given equal weightage for computation of average rank.

It may be noted that every codec can deliver a varying degree of output video quality for a given set of input frames. Any degradation of visual data introduced by "lossy" compression will inevitably remain visible through any further processing of the content. Notwithstanding the encoding options which considerably affect the performance of a compressed-domain algorithm, the proposed method delivers better overall performance even when pitted against state-of-the-art pixel-based techniques.

Background segmentation is but one component of a potentially complex computer vision system. Therefore, in addition to being accurate, a successful technique must consume as few CPU cycles and as little memory as possible. An algorithm that segments perfectly but computationally expensive is useless because insufficient processing resources will remain to do anything useful with its results in real-time. Most

notable aspect, in this regard, is the comparison of average processing speeds in Table-3.2. The computing speeds were recorded for videos with 720×420 resolution on a personal computer powered by Intel Core i7-2600 3.40 GHz CPU with 16GB RAM (no dedicated hardware or graphics processing unit used). It is evident, that the proposed method runs significantly faster in comparison to any of the reported state of the art techniques.

The computation per macroblock involved in each step of the proposed method, costs up to a constant factor. Consequently, the complexity of the overall process is $O(T + c_1\beta + c_2 \sum_{idx=0}^{T-1} \tau_1[idx])$ where $\sum_{idx=0}^{T-1} \tau_1[idx]$ denotes the number of candidate macroblocks that require pixel-level processing, T is the total number of macroblocks per frame, and c_1, c_2 are constants. Arguably, the running time scales linearly with T, incurring only a negligible overhead $\kappa = O(c_1\beta + c_2 \sum_{idx=0}^{T-1} \tau_1[idx]) \leq T$ in addition to the regular decoding cost claimed by each frame. The proposed method is aptly built for real-time network streaming applications in consideration of variable / constant bit rate options under practical bandwidth constraints. It also proved to be robust to a diverse set of real-world (non-synthetic) surveillance sequences.

Although H.264 Baseline is very popular video coding standard used by the industry, but there is a continued attempt in order to reduce the bandwidth/storage requirements for efficient low cost transmission of the video of same quality. As a result of it, more efficient Main and High profiles are being used. Therefore, in Chapter 4, we propose enhanced macroblock features for the application of background subtraction in H.264 Main/High profile.

Table 3.1 Quantitative evaluation (Results of all categories combined)

Method	Avg Recall		Avg. F-Measure		Avg. Precision		Avg. speed (fps)	**Average Rank**
	CD.net	BMC	CD.net	BMC	CD.net	BMC		
Proposed (LUV)	0.5953(4)	0.6787(4)	0.6526(3)	0.7678(1)	0.7884(3)	0.9054(1)	401.7(2)	**2.5714**
Proposed (YCbCr)	0.5899(5)	0.7067(3)	0.6484(5)	0.7604(2)	0.7986(2)	0.8916(2)	443.1(1)	**2.8571**
SC-SOBS [131]	0.8016(1)	0.7316(2)	0.7283(1)	0.7593(3)	0.7316(4)	0.7916(3)	7.3(6)	2.8571
ViBe [132]	0.6960(3)	0.7686(1)	0.6485(4)	0.6968(5)	0.6950(5)	0.7665(4)	121.2(3)	3.5714
KDE [42]	0.7442(2)	0.6612(5)	0.6719(2)	0.6976(4)	0.6843(6)	0.7178(6)	8.8(5)	4.2857
GMM [133]	0.5072(6)	0.5679(6)	0.5904(6)	0.6254(6)	0.8228(1)	0.7521(5)	30.1(4)	4.8571

****Major portion of this Chapter taken from the following publication**

B. Dey and M. K. Kundu, "Robust background subtraction for network surveillance in H.264 streaming video," IEEE Transactions on Circuits and Systems for Video Technology, vol. 23, no. 10, pp. 1695-1703, 2013.

Chapter 4

Enhanced Macroblock Feature Extraction Under Low-Bitrate Encoding Constraints of H.264 Main/High Profile

4.1 Introduction

As mentioned in the previous chapter, H.264 Baseline is a well-established standard for video surveillance. However, its Main and High-profile implementations, have the unique capabilities that pack more visual detail into a given bitrate. The range of quantization step sizes and other parameters vary depending on the frame content and low bitrate constraints. When the parameters vary over a large range it poses challenges for identification of moving objects/targets accurately from the bitstream. Therefore, in order to accommodate very low bitrate constraint, the current chapter is focused on background modeling for compressed domain H.264 (Main and High profile) video streams and this has got quite different approach for feature extraction from the model discussed in the earlier chapter. To this end, enhanced macroblock features are proposed. Experimental results showing comparison on bitstreams encoded at strikingly low bitrates are obtained over a diverse set of standardized surveillance sequences.

The significant contributions of the chapter are discussed as follows:

1. It is known that video encoders can choose to improve image quality in any encoding situation where decisions are made that affect both file size and quality simultaneously. The classical method of making encoding decisions is for the encoder to choose the result which yields the highest quality output image for a given bandwidth. Under low and constrained bandwidth, the choice that the encoder makes, although require more bits in complex segments of an image still suffer from quality benefit (as determined by quantization) compared to less complex areas. As a consequence, the methods [117], [107] which depend solely on the number of encoding bits fail to detect motion/activity when the bitrate is severely constrained and rate-distortion optimization is enabled. Therefore, in the current work, enhanced macroblock features are proposed which use both encoding bits as well as the quantization step-sizes of individual coefficients in a macroblock. This is particularly important as the quantization parameters vary widely between simple and complex sections of an encoded image at lower bitrates;
2. The proposed method is a two-stage hybrid segmentation where macroblocks covering probable foreground regions are first identified. A novel thresholding technique is also used to select the candidate macroblocks corresponding to foreground objects.
3. In the second stage, pixels constituting the selected macroblocks are classified into

foreground or background by comparing their counterparts with those of a background model. It may be noted that the presence of shadows usually causes false classification, leading to various unwanted behavior such as object shape distortion and object merging. Therefore, instead of using the native YCbCr color space, pixel differences are estimated using the CIELUV color space because of its perceptual uniformity of color differences. Experimental results [135] on real-life surveillance videos have shown that the CIELUV color space is the most efficient in modeling cast shadows.

4.2 Enhanced Macroblock Features

As mentioned earlier, the existing macroblocks features depend only on the number of encoding bits and hence fail to detect motion/activity when the bitrate is severely constrained and rate-distortion optimization is enabled. Therefore, in this section enhanced macroblock features are proposed which takes, both - the encoding bits as well as the effective quantization step sizes of individual coefficients. The proposed method is described later in section 4.3.

In compressed video, motion vectors are used to predict temporally correlated patches of image from neighboring frames, while the residual error, i.e., the difference between the predicted and the target patch, is subjected to signal transforms to reduce the spatial redundancy. Put otherwise, the motion vectors as well as the transformed coefficients register incremental changes occurring in a scene. Using a compressed video analyzer, it was verified that the macroblock regions corresponding to moving objects contain motion vectors and transform coefficients of significantly larger magnitudes than those corresponding to the scene background. Hence, the *signal energies* associated with the quantized transform coefficients and the motion vectors of a given macroblock, heretofore designated as F_1 and F_2 respectively, are adopted as the enhanced macroblock features for moving object segmentation.

A total of K macroblocks are assumed in each frame, which are addressed numerically using a macroblock index $idx \in \{1,2,\ldots, K\text{-}1\}$ in the raster-scan order starting with idx=0 for the macroblock at the top-left-hand corner. Formally, the feature vector corresponding to the macroblock at location idx in frame t is denoted as the pattern (or feature) vector

$$\vec{F}_{t,idx} = (F_1, \quad F_2)^T,$$

where F_1, F_2 $\mathbb{R}\geq 0$, the set on non-negative reals.

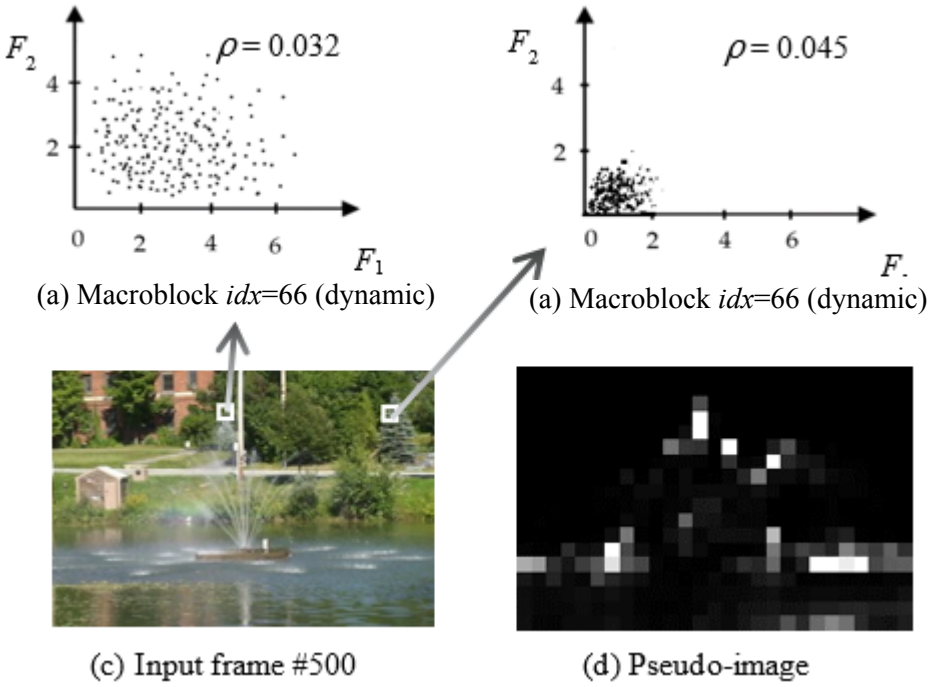


Fig. 4.1 Scatterplots shown on the top row correspond to two different macroblocks depicting (a) dynamic content and (b) static content. Bottom row shows (c) an input frame selected from fountain02 sequence with macroblocks demarcated in white, and (d) the corresponding pseudo-image of $|\Sigma_{idx}|$ scaled to [0, 255].

It may be noted that the determinant of covariance $|\Sigma_{idx}|$ between F_1 and F_2 for all macroblocks at location *idx*, reflects the measure of randomness or 'dynamic' entity present at that location in the scene. To illustrate this fact, the (temporal) mean μ_{idx} and the covariance Σ_{idx} for all $idx \in \{1,2,\ldots, K\text{-}1\}$ were computed from a set of initial 500 frames of containing randomness at the background. Fig. 4.1 (top row) shows the scatterplots of F_1-F_2 feature space for macroblocks at static as well as dynamic locations depicting fountain at the background. A higher $|\Sigma_{idx}|$ value (Fig. 4.1-a) was observed for the macroblock region corresponding to 'dynamic' location compared to that of a 'static' location (Fig. 4.1-b). The correlation coefficient (ρ) for both the locations, however, are very close to zero, illustrating the fact that F1 and F2 are statistically uncorrelated. The second row in Fig. 4.1 shows the input frame (Fig. 4.1-c) and the corresponding grayscale pseudo-image (Fig. 4.1-d) obtained by scaling $\{\Sigma_{idx}\}_{idx=0}^{K-1}$ to [0, 255]. In the pseudo-image, the location of macroblocks with higher $|\Sigma_{idx}|$, i.e., brighter blocks, strongly correlate with the presence of dynamic content while those having a lower value of $|\Sigma_{idx}|$, *i.e.*, darker blocks, correspond to static areas of the scene.

The macroblock feature F_1, as introduced above, is computed as signal energy

associated with its quantized transform coefficients. Computation of F_1 using decoded transform coefficients is computationally challenging. Therefore, F_1 is statistically predicted using the number of transform coding bits, and the quantization parameters QPC (with C=Y for the luma component, and C=Cb, Cr for the chroma components) of a macroblock. These values of the parameters are parsed, in real-time, from the H.264 bitstream coding syntax.

It may here be noted that the H.264/AVC standard adopted multiple transform block sizes (4×4 and 8×8) to improve the coding efficiency. The transforms were designed as approximations to the inverse discrete cosine transform with emphasis on minimizing the number of arithmetic operations. These transforms had large variations of the norm of the basis vectors. As a consequence of this, non-flat default quantization matrices were specified to compensate for different norms of the basis vectors [136]. However, uniform quantization was assumed in [107], which causes its performance to suffer significantly at constrained low bitrates. Therefore, we derive independently, the relation between quantization step size Q and quantization parameter QP^C effective for each coefficient in 4×4 and 8×8 transform blocks, and use it to derive the expression of F_1.

4.2.1 Relation between QP^C and Q in a 4x4 coefficient block

The transform process in H.264 is implemented in conjunction with quantization, to enable a division-free process involving 16-bit addition, multiplication, and binary-shift operations on integers. Considering a 2-dimensional DCT of a 4×4 block of input signal *X*, the integer approximation used is the matrix product

$$Y = A \cdot X \cdot A^{T}, \tag{4.1}$$

where *A* denotes the core-transform matrix [136]. Since the basis vectors of A have different norms, Y is normalized using the matrix

$$E = \mathrm{norm}(A) \circ \mathrm{norm}(A)^{T}. \tag{4.2}$$

where,

1) Function normr(A) returns a matrix of the same size as input A, with the output matrix having each element equal to the reciprocal of the norm of the corresponding row vector in A; and
2) denotes the Hadamard product of two matrices

Substituting the value of A in (4.2), we have

$$E = \begin{bmatrix} 1/4 & 1/2\sqrt{10} & 1/4 & 1/2\sqrt{10} \\ 1/2\sqrt{10} & 1/10 & 1/2\sqrt{10} & 1/10 \\ 1/4 & 1/2\sqrt{10} & 1/4 & 1/2\sqrt{10} \\ 1/2\sqrt{10} & 1/10 & 1/2\sqrt{10} & 1/10 \end{bmatrix},$$

Using E, the required process that yields the resultant block Z of normalized coefficients is

$$Z = Y \circ E, \tag{4.3}$$

As mentioned earlier, the above process is implemented in conjunction with quantization as [136]

$$\begin{aligned} k_{i,j} &= \text{round}(Z_{i,j}/Q_{i,j}) \\ &= \text{round}(Y_{i,j}E_{i,j}/Q_{i,j}) \quad [\because Z = Y \circ E] \\ &= \text{round}(Y_{i,j}S_{i,j}/2^{L}) \\ &= \text{sgn}(Y_{i,j})((|Y_{i,j}|S_{i,j} + f) \gg L), \end{aligned} \tag{4.4}$$

where subscripts $i, j = 0, \ldots, 3$ are denote the element at the $(i+1)$th row and the $(j+1)$th column of the corresponding 4×4 matrix, and $L = 15 + \lfloor QP^{C}/6 \rfloor$. S is specified by the quantization matrix M as

$$S_{i,j} = \begin{cases} M_{m,0} & \text{for } (i \bmod 2, j \bmod 2) = (0,0), \\ M_{m,1} & \text{for } (i \bmod 2, j \bmod 2) = (1,1), \\ M_{m,2} & \text{otherwise;} \end{cases}$$

where $m = \text{QP}^{\text{C}} \bmod 6$, and

$$M = \begin{bmatrix} 13107 & 11916 & 10082 & 9362 & 8192 & 7282 \\ 5243 & 4660 & 4194 & 3647 & 3355 & 2893 \\ 8066 & 7490 & 6554 & 5825 & 5243 & 4559 \end{bmatrix}^{\text{T}}.$$

From (4.4), we have

$$E_{i,j}/Q_{i,j} = S_{i,j}/2^{L},$$

which on rearrangement of terms gives

$$Q_{i,j} = (E_{i,j}/S_{i,j})2^{L}. \tag{4.5}$$

Finally, substituting the values of L, $E_{\text{i,j}}$, and $S_{\text{i,j}}$ in (4.5), the required relation between QP^{C} and Q for 4×4 coefficient block (say Q4) is obtained as

$$Q4^{\text{C}}_{i,j} = \begin{cases} R_{m,0} & \text{for } (i \bmod 2, j \bmod 2) = (0,0), \\ R_{m,1} & \text{for } (i \bmod 2, j \bmod 2) = (1,1), \\ R_{m,2} & \text{otherwise;} \end{cases} \tag{4.6}$$

where

$$R = 2^{\left\lfloor \frac{QP^C}{6} \right\rfloor} \begin{bmatrix} \frac{2^{13}}{13107} & \frac{2^{11}}{2979} & \frac{2^{12}}{5041} & \frac{2^{12}}{4681} & 1 & \frac{2^{12}}{3641} \\ \frac{2^{14}}{26215} & \frac{2^{12}}{5825} & \frac{2^{13}}{10485} & \frac{2^{14}}{18235} & \frac{2^{14}}{16775} & \frac{2^{14}}{14465} \\ \frac{2^{13}}{4033\sqrt{10}} & \frac{2^{13}}{3745\sqrt{10}} & \frac{2^{13}}{3277\sqrt{10}} & \frac{2^{14}}{5825\sqrt{10}} & \frac{2^{14}}{5243\sqrt{10}} & \frac{2^{14}}{4559\sqrt{10}} \end{bmatrix}^T,$$

and superscript C = Y, Cb, Cr depending on the color component associated with the input block X.

4.2.2 Relation between QP^C and Q in an 8x8 coefficient block

The adaptive 8×8 transform is optionally supported in High profiles. When selected for coding a macroblock, the 8×8 transform is applied to the luma signal only, while the chroma components are subjected to 4×4 transforms as usual. The core integer transform of an 8×8 block of input signal X is performed according to (4.2), where

$$A = \begin{bmatrix} 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\ 12 & 10 & 6 & 3 & -3 & -6 & -10 & -12 \\ 8 & 4 & -4 & -8 & -8 & -4 & 4 & 8 \\ 10 & -3 & -12 & -6 & 6 & 12 & 3 & -10 \\ 8 & -8 & -8 & 8 & 8 & -8 & -8 & 8 \\ 6 & -12 & 3 & 10 & 10 & -3 & 12 & -6 \\ 4 & -8 & 8 & -4 & -4 & 8 & -8 & 4 \\ 3 & -6 & 10 & -12 & 12 & -10 & 6 & -3 \end{bmatrix} \cdot 1/8$$

as specified by [137]. Substituting the value of A in (4.2), the matrix E for 8×8 block transforms is obtained as

$$E = \begin{bmatrix} 1/8 & 2/17 & 1/2\sqrt{10} & 2/17 & \\ 2/17 & 32/289 & 8/17\sqrt{10} & 32/289 & \\ 1/2\sqrt{10} & 8/17\sqrt{10} & 1/5 & 8/17\sqrt{10} & \cdots \\ 2/17 & 32/289 & 8/17\sqrt{10} & 32/289 & \\ & & \cdots & & \cdots \end{bmatrix}_{8\times 8}.$$

Only the top-left quadrant is shown, while the remaining three quadrants are identical. The quantization of an 8×8 block of transform coefficients is implemented according to equation 4.5 with the exception that $L = 16 + \lfloor QP^C/6 \rfloor$,

$$S_{i,j} = \begin{cases} M_{m,0} & \text{for } (i \bmod 4, j \bmod 4) = (0,0), \\ M_{m,1} & \text{for } (i \bmod 2, j \bmod 2) = (1,1), \\ M_{m,2} & \text{for } (i \bmod 4, j \bmod 4) = (2,2), \\ M_{m,3} & \text{for } (i \bmod 4, j \bmod 2) = (0,1) \\ & \text{or } (i \bmod 2, j \bmod 4) = (1,0), \\ M_{m,4} & \text{for } (i \bmod 4, j \bmod 4) \in \{(0,2),(2,0)\}, \\ M_{m,5} & \text{otherwise;} \end{cases}$$

where subscripts i, j = 0,…,7 denote the element at (i+1)th row and (j+1)th column of the corresponding 8×8 matrix, and

$$M = \begin{bmatrix} 13107 & 11428 & 20972 & 12222 & 16777 & 15481 \\ 11916 & 10826 & 19174 & 11058 & 14980 & 14290 \\ 10082 & 8943 & 15978 & 9675 & 12710 & 11985 \\ 9362 & 8228 & 14913 & 8931 & 11984 & 11259 \\ 8192 & 7346 & 13159 & 7740 & 10486 & 9777 \\ 7282 & 6428 & 11570 & 6830 & 9118 & 8640 \end{bmatrix}.$$

Finally, substituting the values of L, $E_{i,j}$ and $S_{i,j}$ in (4.5), the required relation between QP and Q for an 8×8 coefficient block (say Q8) is obtained as

$$Q8^{C}_{i,j} = \begin{cases} R_{m,0} & \text{for } (i \bmod 4, j \bmod 4) = (0,0), \\ R_{m,1} & \text{for } (i \bmod 2, j \bmod 2) = (1,1), \\ R_{m,2} & \text{for } (i \bmod 4, j \bmod 4) = (2,2), \\ R_{m,3} & \text{for } (i \bmod 4, j \bmod 2) = (0,1) \\ & \text{or } (i \bmod 2, j \bmod 4) = (1,0), \\ R_{m,4} & \text{for } (i \bmod 4, j \bmod 4) \in \{(0,2),(2,0)\} \\ R_{m,5} & \text{otherwise;} \end{cases} \tag{4.7}$$

where

$$R = 2^{\left\lfloor \frac{QP^{C}}{6} \right\rfloor} \begin{bmatrix} \frac{2^{13}}{13107} & \frac{2^{19}}{825673} & \frac{2^{14}}{26215} & \frac{2^{16}}{103887} & \frac{2^{15}}{16777\sqrt{10}} & \frac{2^{19}}{263177\sqrt{10}} \\ \frac{2^{11}}{2979} & \frac{2^{20}}{1564357} & \frac{2^{15}}{47935} & \frac{2^{16}}{93993} & \frac{2^{13}}{3745\sqrt{10}} & \frac{2^{18}}{121465\sqrt{10}} \\ \frac{2^{12}}{5041} & \frac{2^{21}}{2584527} & \frac{2^{15}}{39945} & \frac{2^{17}}{164475} & \frac{2^{14}}{6355\sqrt{10}} & \frac{2^{19}}{203745\sqrt{10}} \\ \frac{2^{12}}{4681} & \frac{2^{19}}{594473} & \frac{2^{16}}{74565} & \frac{2^{17}}{151827} & \frac{2^{11}}{749\sqrt{10}} & \frac{2^{19}}{191403\sqrt{10}} \\ 1 & \frac{2^{20}}{1061497} & \frac{2^{16}}{65795} & \frac{2^{15}}{32895} & \frac{2^{14}}{5243\sqrt{10}} & \frac{2^{19}}{166209\sqrt{10}} \\ \frac{2^{12}}{3641} & \frac{2^{19}}{464423} & \frac{2^{15}}{28925} & \frac{2^{16}}{58055} & \frac{2^{14}}{4559\sqrt{10}} & \frac{2^{13}}{2295\sqrt{10}} \end{bmatrix},$$

and superscript C = Y for the luma component only.

4.2.3 Computation of F_1 for a given macroblock

The spatial-to-frequency transforms applied in H.264 is an integer approximation of the 2-dimentional DCT using only integer operations. The DCT coefficients are best modeled in literature [126] as a zero-mean distribution with a Laplacian pdf, i.e.,

$$p(z) = \frac{1}{2b}\exp\left(-|z|/b\right), z \in \mathbb{R} \tag{4.8}$$

where z is the value of a given DCT coefficient, and b>0 is a parameter that determines the coefficient variance. Quantization of an input coefficient z is performed as

$$k = \text{round}(z/Q) = \text{sgn}(z)\left\lfloor \left(|z| + f\right)/Q \right\rfloor, \tag{4.9}$$

where k represents the coefficient level that is encoded by the source encoder, Q>0 is the uniform quantization step-size, and f is a rounding offset with a value equal to Q/3 or Q/6 accordingly as the macroblock is intra- or inter-predicted [135]. It may be verified that all values of z in the open interval $(-Q+f, Q-f)$ are mapped to k=0. Similarly the quantization levels are mapped to k = 1, 2, 3,… for z $[(kQ-f), (kQ+Q-f))$ and $k = -1, -2, -3,...$ for z $((kQ-Q+f), (kQ+f)]$. Therefore, the probability $P_f(z_k)$ that a random input coefficient z is mapped to level k via the quantization process (4.11) is analytically expressed as

$$P_f(z_k) = \begin{cases} \int_{(kQ-Q+f)}^{(kQ+f)} p(z)dz = \dfrac{e^{kQ/b}}{2}(e^{Q/b}-1)e^{-(Q-f)/b}; & \text{for } k<0 \\ \int_{(Q-f)}^{(-Q+f)} p(z)dz = 1-e^{-(Q-f)/b}; & \text{for } k=0 \\ \int_{(kQ-f)}^{(kQ+Q-f)} p(z)dz = \dfrac{e^{-kQ/b}}{2}(e^{Q/b}-1)e^{-(Q-f)/b}; & \text{for } k>0 \end{cases} \tag{4.10}$$

where e = exp(1).

The inverse operation involving the reconstruction of coefficient value z_k corresponding to level k is given by

$$z_k = kQ. \tag{4.11}$$

Given b and Q for a particular coefficient position in a transform block (TB), the signal energy $F_{1,\text{coeff}}$ associated with it is given as

$$F_{1,\text{coeff}}(Q,b) = \sum_{k=-\infty}^{\infty} z_k^2 P_f(z_k) = \frac{Q^2 e^{f/b}(1+e^{Q/b})}{(e^{Q/b}-1)^2}. \tag{4.12}$$

It is theoretically based on the fact that energy of a discrete-time signal $x(n)$, which is assumed to be composed of components z_k with probability $P_f(z_k)$, is defined as $\sum_{n=-\infty}^{\infty}|x(n)|^2$. The values of Q for 4×4 (=Q4) and 8×8 (=Q8) TBs are obtained from (4.6) and (4.7), while the parameter b is evaluated by equating the number of bits (say bits) consumed in coding the transform coefficients of a macroblock with its analytical

expression obtained using entropy. Recall that entropy, heretofore denoted as H_f, is a lower bound on the average number of bits required to encode a given coefficient. Using (4.10), H_f (in bits/coefficient) may be expressed as

$$\begin{aligned} H_f(Q,b) &= -\sum_{k=-\infty}^{+\infty} P_f(z_k)\log_2 P_f(z_k) \\ &= -(1-\mathrm{e}^{-(Q-f)/b})\log_2(1-\mathrm{e}^{-(Q-f)/b}) - \frac{\mathrm{e}^{-(Q-f)/b}}{\ln 2}\left(\ln\left(\frac{\mathrm{e}^{Q/b}-1}{2}\right) - \frac{Q-f}{b} - \frac{Q\mathrm{e}^{Q/b}}{b(\mathrm{e}^{Q/b}-1)}\right), \end{aligned} \tag{4.13}$$

which is a function of Q and b.

Based on the TB sizes adopted in H.264, we formulate F_1 under two cases as follows:

1) *When 4×4 transforms are used for all color components*

Considering a macroblock to be encoded using the popular YCbCr 4:2:0 color format, there exist sixteen 4×4 TBs of the luma component and four 4×4 blocks of each of the chroma components. The value of Q for a particular coefficient vary depending on its position in the transform block, and hence the value of H_f. Therefore, we express bits as the sum of all H_fs corresponding to each coefficient of a macroblock, i.e.

$$bits = 16\sum_{i=0}^{3}\sum_{j=0}^{3} H_f(Q4_{i,j}^{\mathrm{Y}}, b) + 4\sum_{C=Cb,Cr}\sum_{i=0}^{3}\sum_{j=0}^{3} H_f(Q4_{i,j}^{C}, b). \tag{4.14}$$

In (4.14), the coefficients of a given macroblock are modeled with a single source distribution; hence, b – the distribution parameter, is constant for all TBs. The values of *bits*, B, QP^{Y}, QP^{Cb}, and QP^{Cr}, which are parsed from the macroblock-layer syntax in the input bitstream are used to compute $Q4_{i,j}^{Y}$, $Q4_{i,j}^{Cb}$ and $Q4_{i,j}^{Cr}$ by substituting C with Y, Cb, and Cr respectively in (4.6). The remaining parameter b is evaluated using standard numerical methods [137] for finding the root of non-linear equation in one unknown.

The energy associated with each 4×4 TB of a given macroblock is computed as the average of the value given by (4.12), i.e.,

$$\frac{1}{16}\sum_{i=0}^{3}\sum_{j=0}^{3} F_{1,\mathrm{coeff}}(Q4_{i,j}^{\mathrm{C}}, b);\ \mathrm{C} = \mathrm{Y}, \mathrm{Cb}, \mathrm{Cr}.$$

Finally, F_1 due to the entire macroblock is computed as the weighted average of the values obtained for individual color components. Thus, we have

$$F_1 = \frac{1}{24}\sum_{i=0}^{3}\sum_{j=0}^{3} F_{1,\mathrm{coeff}}(Q4_{i,j}^{\mathrm{Y}}, b) + \frac{1}{96}\sum_{C=Cb,Cr}\sum_{i=0}^{3}\sum_{j=0}^{3} F_{1,\mathrm{coeff}}(Q4_{i,j}^{\mathrm{C}}, b). \tag{4.15}$$

The normalization factors/weights are based on the fact that in 4:2:0 sampling, the luma coefficients comprise a ratio of exactly 2/3 of the total number of coefficients of a macroblock, while that for the chroma components being equal to 1/6 each. It may be noted that F_1 is 'quantization-normalized' as its computation takes into account the widely varying quantization step sizes applied to individual coefficients at constrained bitrates.

2) *When 8×8 transforms are used for the luma component*

It may be noted that, encoded bitstreams using 8×8 transforms for the luma component (High profiles only) use 4×4 transforms for the chroma components. Thus, four transform blocks of each color component (but varying sizes) comprise a given macroblock. Therefore, we express *bits* as

$$bits = 4\left[\sum_{i=0}^{7}\sum_{j=0}^{7} H_f(Q8_{i,j}^{\mathrm{Y}}, b) + \sum_{C=Cb,Cr}\sum_{i=0}^{3}\sum_{j=0}^{3} H_f(Q4_{i,j}^{C}, b)\right]. \tag{4.16}$$

The values of $Q8_{i,j}^{Y}$ are computed using (4.7), while those of $Q4_{i,j}^{Cb}$ and $Q4_{i,j}^{Cr}$ are obtained by replacing C with Cb and Cr respectively in (4.6). The unknown constant *b* is evaluated using numerical techniques for solving non-linear equations in one unknown. Using (4.12), the energy associated with an 8×8 luma TB is given as while those contributed by the chroma blocks are obtained as described for 4×4 blocks (case-1).

$$\frac{1}{64}\sum_{i=0}^{7}\sum_{j=0}^{7} F_{1,\mathrm{coeff}}(Q8_{i,j}^{\mathrm{Y}}, b),$$

Finally, F_1 for a given macroblock is expressed as the normalized sum

$$F_1 = \frac{1}{96}\left[\sum_{i=0}^{7}\sum_{j=0}^{7} F_{1,\mathrm{coeff}}(Q8_{i,j}^{\mathrm{Y}}, b) + \sum_{C=Cb,Cr}\sum_{i=0}^{3}\sum_{j=0}^{3} F_{1,\mathrm{coeff}}(Q4_{i,j}^{\mathrm{C}}, b)\right]. \tag{4.17}$$

4.2.4 Implementation considerations for computation of F_1

The statistical prediction of F_1 using (4.15) and (4.17) for a given macroblock depends only on bits, QP^{Y}, QP^{Cb}, and QP^{Cr}, the values of which are non-negative integers in the range specified by the H.264 standard. The maximum number of bits allowed in the macroblock-layer (for bitstreams using 8-bit color sampling) is specified as 3200. Furthermore, $QP^{Y} \in \{0, 1,...,51\}$, while QP^{Cb} and QP^{Cr} are determined from Table-4.1 based on the index denoted as qP_I. The value of qP_I is obtained as [138]

$$qP_I = \min(\max(0, QP^Y + qP_{Offset}), 51),$$

where $qP_{Offset} \in \{-12, -11,...,11,12\}$ is a value specified by syntax elements *chroma_qp_index_offset* (for Cb) and *second_chroma_qp_index_offset* (for Cr) in the Picture Parameter Set (PPS). As a consequence, the number of distinct input combinations of (*bits*, QP^Y, QP^{Cb}, QP^{Cr}) possible is no more than 3200×17807. In order to reduce computations at run-time, pre-computed values of F_1 for all input combinations are indexed in lookup tables. Thus, the computation required for statistical prediction of F_1 is completely replaced with simple lookup operations.

Table 4.1 QP^C as a function of qP_I

qP_I	QP^C	qP_I	QP^C	qP_I	QP^C
< 30	$=qP_I$	37	34	45	38
30	29	38	35	46	38
31	30	39	35	47	38
32	31	40	36	48	39
33	32	41	36	49	39
34	32	42	37	50	39
35	33	43	37	51	39
36	34	44	37		

It may be noted that the proposed formulation of F_1 is based on transform coefficient statistics using only partially decoded parameters of a macroblock such as bits, QP^Y, QP^{Cb}, and QP^{Cr} rather than actual coefficients decoded from compressed video. Considering the fact that signal processing transforms output exactly the same number of coefficients as the input pixels, the statistical formulation helps us to evaluate F_1 with significantly less computation as compared to that of pixel-based methods.

4.2.5 Computation of F_2 for a given Macroblock

F_2 is the signal energy associated with motion vectors of a macroblock. It quantifies the localized motion content of a macroblock partition in the current frame with respect to previously coded frames. An inter-predicted macroblock is composed of one of more partitions predicted from a set of previously coded reference frames. The prediction information is indicated by a set of motion vectors and corresponding reference frame indices. Each motion vector associated with a macroblock partition is predicted either uni-directionally (from one) or bi-directionally (from two) reference frame(s). F_2 is computed

as the normalized/weighted sum of squares of the motion vector magnitudes of a macroblock. Accordingly, the weight or normalization factor associated with each motion vector is determined on the basis of 1) the ratio of the partition size to the total macroblock area it represents; and 2) the reference frame index, considering the temporal distance between the current and the referenced frame.

As mentioned earlier, a macroblock may be split into one, two or four partitions using either one 16×16 partition (covering the whole macroblock); two 8×16 partitions; two 16×8 partitions; or four 8×8 partitions. In case of 8×8 blocks, each block is split into one, two or four sub-partitions (either one 8×8, two 4×8, two 8×4 or four 4×4 sub-partitions). Let

$$W_i = \text{Size of the } i\text{th partition} \big/ 256$$

denote the weight due to the ith partition of a macroblock. Furthermore, corresponding to the ith partition, let

$$d_i = \begin{cases} 1; & \text{for uni-directional prediction} \\ 2; & \text{otherwise} \end{cases}$$

Corresponding to each motion vector (x_{ij}, y_{ij}) denoting the prediction of the ith partition in the jth direction (i.e., either forward or backward), let us denote the reference frame index as s_{ij} (where reference index 0 denotes frame one in the past/future, reference index 1 denotes frame two in the past/future, and so on). It is easy to check that the weights of all motion vectors of a macroblock sum up to unity. Assuming a total of p (maximum value being 16) partitions in a given inter-predicted macroblock, F_2 is computed as

$$F_2 = \sum_{i=1}^{p} \frac{W_i}{d_i} \sum_{j=1}^{d_i} \left[(x_{ij}^2 + y_{ij}^2) \big/ (s_{ij} + 1) \right]. \tag{4.18}$$

It may be noted that:

1) The value of F_2 for intra-predicted macroblocks is taken as zero as they do not encode any motion compensated information;
2) A macroblock may be split into a maximum of 16 partitions with each having two motion vectors in both directions. Therefore, computation of F_2 for any given macroblock requires no greater than 16×2×5 multiplications/divisions and 16×2×2 additions.

4.3 The Proposed Method

This section is divided into subsections involving background-model initialization,

segmentation, and update.

4.3.1 Background model initialization

Based on the proposed macroblock features, a 2D Gaussian pdf is fitted using the most-recent *n* frames. In order to avoid fitting the model from scratch for every incoming frame, a running (or on-line cumulative) estimate is computed. The background model parameters corresponding to each macroblock having location *idx* in frame *t*, i.e., the background mean $\vec{\mu}_{t,idx}$ and the co-variance $\Sigma_{t,idx}$ are recursively computed as

$$\vec{\mu}_{t,idx} = \begin{cases} \vec{F}_{t,idx} & \text{for } t = 1 \\ \eta\vec{F}_{t,idx} + (1-\eta)\vec{\mu}_{t-1,idx} & \text{for } t \geq 2 \end{cases} \tag{4.19}$$

and,

$$\Sigma_{t,idx} = \begin{cases} \text{Diag}(0,0) & \text{for } t = 1 \\ \eta\left(\vec{F}_{t,idx} - \vec{\mu}_{t,idx}\right)\left(\vec{F}_{t,idx} - \vec{\mu}_{t,idx}\right)^T & \\ \quad +(1-\eta)\Sigma_{t-1,idx} & \text{for } t \geq 2 \end{cases} \tag{4.20}$$

where, η=0.01 is an empirically chosen parameter that determines the tradeoff between stability and quickly updates.

The background frame is initialized using the first *N*=250 frames from the input sequence. Although the set of initial frames that are used for background-model learning is ideally supposed to contain only background information, in practice, however, it is difficult to get real-world sequence not occupied by foreground objects. Hence, not a single frame is appropriate to be used as the background frame. To address this problem, a concept of the most common frame of a scene (McFIS) [139] has been developed for video coding using dynamic background modeling. In this paper, a recently-modified version of the McFIS generation algorithm [140] for the decoded frames has been used as the initial background frame. We initialized the parameters in this implementation as follows: maximum number of models for a pixel *K*=3, learning rate α=0.1, weight ω=0.001, and variance σ=30 as used in [140].

4.3.2 Segmentation

Beginning with frame *t*=*N*+1, the segmentation process operates via a two-stage coarse-to-fine process as follows.

1) Macroblock selection: At the coarse scale, block-level segmentation of each frame is performed by selecting a set of macroblocks potentially containing foreground

regions. The selected macroblocks are those that correspond to $D_t > \alpha$, where $D_t = \sqrt{\left(\vec{F}_{t,idx} - \vec{\mu}_{t,idx}\right)^T \Sigma_{t,idx}^{-1} \left(\vec{F}_{t,idx} - \vec{\mu}_{t,idx}\right)}$ is the Mahalanobis distance of a given macroblock measured in units of $\left|\Sigma_{t,idx}\right|$ from $\vec{\mu}_{t,idx}$, and α = 2.8 being a predetermined threshold.

2) Pixel-level refinement: Since real object boundaries rarely follow block boundaries, the macroblock-level segmentation is further refined by eliminating pixels from selected macroblocks that are similar to the co-located pixels in the background model. Similarity between a pair of color coordinates $X_C \equiv (L_C, U_C, V_C)$ and $X_B \equiv (L_B, U_B, V_B)$ in the CIELUV color space is ascertained if the distance

$$\Delta\left(X_C, X_B\right) = \left|\frac{L_C - L_B}{\sigma_L}\right| + \left|\frac{U_C - U_B}{\sigma_U}\right| + \left|\frac{V_C - V_B}{\sigma_V}\right| < \tau,$$

where σ_L, σ_U, and σ_V respectively denote the standard deviations of L, U, and V components of the corresponding background pixel. Furthermore, τ=3.4 denotes a decision threshold chosen empirically for defining the fluctuations in luminosity and chromaticity. The value of Δ is a low-complexity approximation of the standardized Euclidean distance between X_C and X_B. The rationale behind the choice of CIELUV space is that, unlike RGB or YCbCr, it exhibits perceptual uniformity, i.e., for the same distance Δ computed between any two pairs of color coordinates, equal color differences are perceived by the human visual system. Experiments on real-life surveillance videos with varying illumination conditions have shown the CIELUV color space to be the most efficient. As brightness information (L component) is separated from chrominance (U and V components), the space is affected less by shadows.

4.3.3 Background Model Update

Let C_t and B_t denote the input frame at t and the corresponding background frame respectively. We adopt a selective update policy where only pixels enclosed by the macroblocks that are not selected in the macroblock selection stage are used to update the current background B_t to B_{t+1} as

$$B_{t+1}(x,y) = \eta C_t(x,y) + \left(1-\eta\right) B_t(x,y), \text{ for } t > N \tag{4.21}$$

where η = 0.01 is the rate of update as defined earlier.

It may be noted that the proposed model essentially translates to a bivariate normal

distribution $\mathcal{N}(\vec{\mu}_{t,idx}, \Sigma_{t,idx}) = (2\pi)^{-1}|\Sigma_{t,idx}|^{-1/2} \exp(-D_t^2/2)$ representing the background at the macroblock-level in F_1–F_2 feature space. Taking $\Sigma_{t,idx} = \text{Diag}(\sigma_1^2, \sigma_2^2)$ and $\vec{\mu}_{t,idx} = (\mu_1, \mu_2)^T$, the value of decision threshold α which correspond to 99%-prediction interval (for our background model predictions) is computed as

$$\int_{\mu_2-\alpha\sigma_2}^{\mu_2+\alpha\sigma_2} \int_{\mu_1-\alpha\sigma_1}^{\mu_1+\alpha\sigma_1} \mathcal{N}\left(\vec{\mu}_{t,idx}, \Sigma_{t,idx}\right) dF_1 dF_2 = \text{erf}^2\left(\frac{\alpha}{\sqrt{2}}\right) = 0.99. \quad (4.22)$$

Solving (4.22) for α, we get α = 2.8.

4.4 Results and Discussion

In this section, the relation between statistically predicted values of F_1 and those that are actually computed using decoded coefficients are analyzed. Subsequently, quantitative and qualitative comparisons are demonstrated on standard sequences to highlight the efficacy of the proposed method at low bitrate.

4.4.1 Comparison between the predicted and the actual values of F_1

The predicted values of F_1 are validated experimentally against their actual counterparts computed directly from the decoded coefficients. Fig. 4.2 illustrates the comparison between actual and predicted values of F_1 for a static (*idx*=76) as well as a dynamic location (*idx*=66) of *fountain02* sequence. It is observed that the predicted values are slightly lower in magnitude compared to those which are computed directly from the decoded coefficients. This is a consequence of the fact that prediction of F_1 depends on bits, whose lower bound on the average bitrate was modeled using the entropy measure. Furthermore, the predicted value and its actual counterpart are found to be linearly correlated. As a matter of fact, the correlation coefficient was found to be greater than 0.96 for all individual sequences. Please note that the discrepancy between the predicted and the actual values, however, do not affect the macroblock selection process described earlier, because the Mahalanobis distance D is invariant under arbitrary linear transformations of the feature space. The average computation time for the predicted and the actual F_1 were also noted as 0.638 nsec and 396.7 nsec. Moreover, the average processing speeds in frames per second (fps) using the actual F_1 were found to be 14.1 fps and 12.8 fps for videos encoded at 1000 KB/s and 200 KB/s respectively. For comparison, the average processing speeds using the predicted values of F_1 are mentioned in Table-4.2. The overall memory requirement of the proposed method involving the actual F_1 and the

predicted F_1 was analyzed to be of the order O(N) and O($N + \varepsilon$) respectively, where N is the number of macroblocks per frame and constant ε = 434.74 MB, being the memory overhead due to look up tables. It may be also noted that O(N+ε) is O(N); since ε is constant, i.e., the space complexity of the methods using the predicted and the actual values of F_1 are arguably the same. Fig. 4.3 illustrates the comparison the segmentation results obtained with the predicted and actual values of F_1. It may be observed that there is negligible difference between the segmentation results obtained with the predicted and the actual value of F_1. By using the predicted value of F_1, as evidenced from the results, we are able to reduce the computation time significantly without affecting the overall segmentation process.

4.4.2 Evaluation of segmentation results

For comparison of segmentation results, the proposed method has been evaluated against the benchmark Changedetection.net (CD.net) dataset [130]. This dataset, unlike any other publicly available, addresses the major key challenges, especially dynamic background, accompanying a real-world surveillance scenario and includes accurate ground truth masks for each frame. The proposed method was implemented in C and patched into the original H.264 decoding module of FFmpeg [129]. The source was built on 64-bit Windows platform.

Fig. 4.4 illustrates the qualitative (visual) comparison of segmentation results of nine selected sequences, viz. *PETS2006*, *traffic*, *boulevard*, *fountain01*, *fall*, *canoe*, *sofa*, *cubicle*, and *peopleInShade* encoded at 200KB/s. The sequences depict irregular camera jitter, dynamic background and moving cast shadows in the background, which are typical of a real-world surveillance scenario. Due to space constraints, segmentation results from two state-of-the-art methods [107] and [141] are shown in Fig. 4.4. Comparison of the results with the provided ground-truth masks (on row 2) shows that the proposed method is best able to model real-world surveillance scenarios, e.g., static background (*PETS2006*), mild camera jitter (*traffic* and *boulevard*), dynamic background (*fountain01*, *fall*, and *canoe*), and moving cast shadow (*sofa*, *cubicle*, and *peopleInShade*).

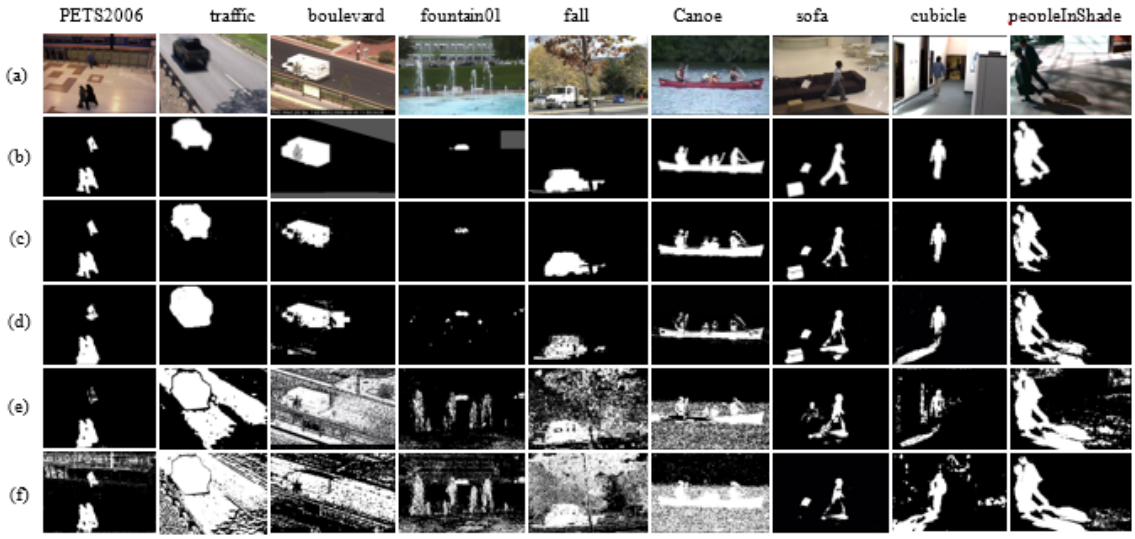


Fig. 4.2 Qualitative results are shown column-wise with at least one sequence selected from each category of the dataset [130]. Starting from the top, the first row (a) illustrates selected image frames. Row (b) illustrates the corresponding ground-truth masks. Row (c) and Row (d) illustrate results obtained with the proposed method using the CIELUV space and the native YCbCr space respectively. Row (e) and Row (f) demonstrate the corresponding masks obtained with [107] and [141] respectively.

In order to compare the achieved results quantitatively, we consider the performance metrics, as reported in Table-4.2. The metrics adopted for quantitative evaluation includes recall, precision, f-measure, and average processing speed. Please note all processing speeds are reported for videos having a resolution of 720×420 pixels on a personal computer powered by Intel Core i7-2600 3.40 GHz CPU and 16GB RAM. The parenthesized figures appearing on the left of each performance score in Table-4.2 indicate the rank of an algorithm in the corresponding evaluation category.

In order to demonstrate the proposed method for a wide range of bitrates, all sequences of the dataset [130] were encoded to H.264 High profile at target bitrates of 200KB/s (low bitrate) and 1000KB/s (high bitrate). It may be noted that no fixed quantization parameter was adopted for encoding the macroblocks, i.e., for each encoded macroblock in a frame the encoder was free to choose a different quantization parameter depending on the complexity of the scene and the target bitrate. The encoder configuration was set as follows: H.264 High profile YCbCr 4:2:0 progressive format with 8-bit color sampling, rate-distortion optimization (RDO) was enabled, and the motion vector range was set to [-16…16]. The streaming rate was fixed at 25 frames per second (fps). Segmentation results were obtained using CIELUV as well as the native YCbCr color space. The evaluation in Table-4.2 is based on the average rank computed over the individual performance metrics discussed above. It is observed that the proposed

method (both CIELUV and YCbCr) obtained better results as compared to the state-of-the-art compressed domain methods [107], [117], and [141]. The proposed method has also been compared with [131], which is a recent pixel-based method. Fig. 5 further demonstrates the impact of low bitrate encoding on the segmentation performance (accessed in terms of F-measure) on two sequences selected from the dataset. Sequences fall and canoe (depicting dynamic background) were encoded with target bitrates of 200 KB/s, 400 KB/s, 600 KB/s, 800 KB/s, and 1000 KB/s. For reference, we have also included segmentation result. It is observed that the proposed method performed consistently better when compared with other state-of-the-art, methods, especially at lower bitrates. Evidently, with the enhanced macroblock features, the proposed method is better able to model the background dynamics under variable quantization schemes adopted at a wide range of bitrates.

On a final note, one key issue for any successful algorithm is its computational complexity. For the proposed method, we used pre-computed lookup tables to replace any run-time computation associated with F_1, while the maximum number of operations involved with F_2, D for each macroblock remains constant. Consequently, the running time of the proposed method reduces to O(N), which is linear in terms of the number of macroblocks/frame. It may be noted that background segmentation is but one component

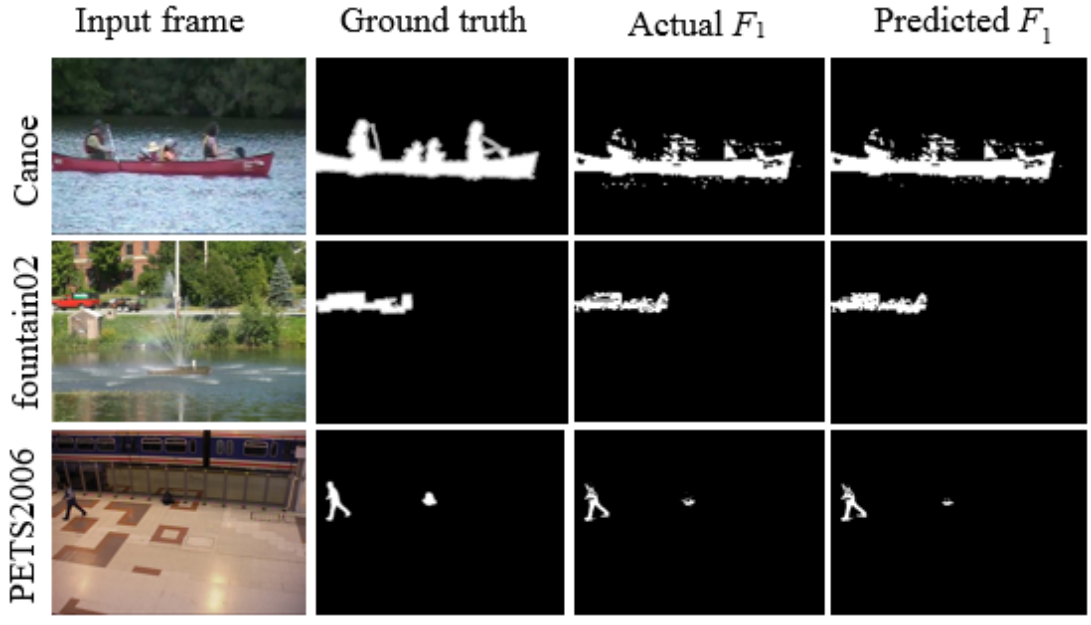


Fig. 4.3 Comparison of the final segmentation results of three sequences using the predicted and the actual values of the proposed feature F_1. No significant difference was registered between the segmentation results obtained with the predicted and the actual value of F_1.

of a potentially complex computer vision system. The problem of foreground-background

separation is only fundamental, and its results are utilized in conjunction with real-world vision problems such as tracking, surveillance and gesture and event detection in real-time, etc. There is, therefore, a huge requirement of high-speed computing techniques as the demand for real-time analysis of multiple surveillance-feeds grows over a constrained-bandwidth network.

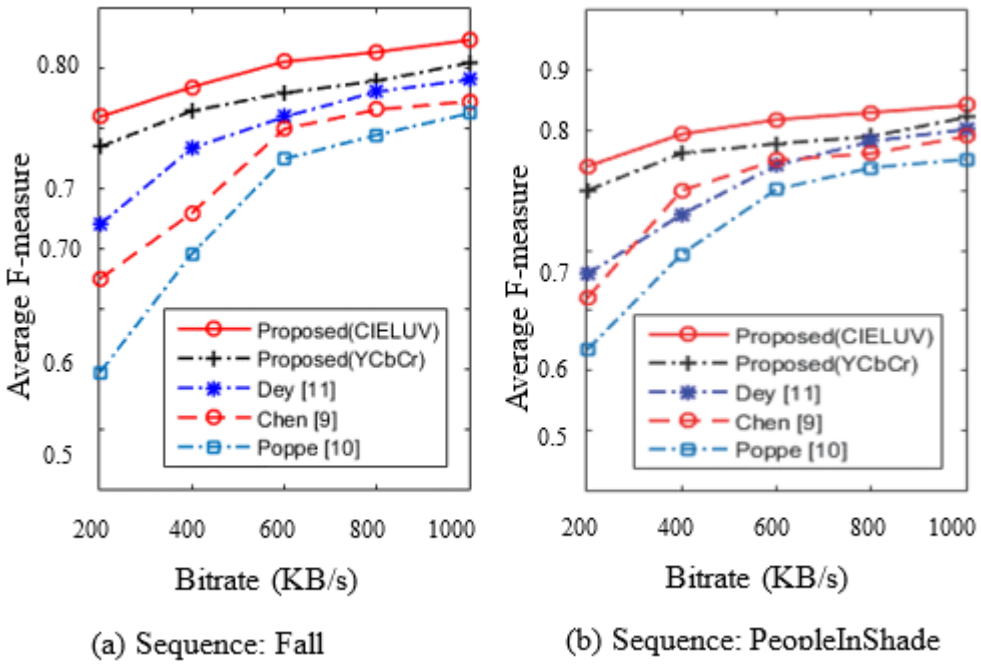


Fig. 4.4 Impact of video encoding bitrates on the segmentation performance of various algorithms.

Background modeling is one of the key techniques for automatic video analysis, especially in the domain of video surveillance. In this paper, we proposed novel block-based features for dynamic background modeling to work directly on low bit rate encoded videos. Experimentally, the method has been tested on H.264 High profile, a premier codec used for streaming of surveillance-grade high-definition video at constrained bit rates. The method clearly outperforms the current benchmark in terms of segmentation performance, while consuming significantly less CPU time. Considering the bandwidth crunch which mounts a huge challenge to deliver high-quality video, the proposed method is geared to tap into the benefits of superior compression that H.264 has to offer.

Although H.264 Main and High profiles could achieve high compression quality at low bitrate. However, research tried to push the limit further to achieve higher compression for high definition images at lower bandwidth. As a result, the new HEVC coding standard was proposed by Video Coding Expert Group (VCEG). Successor to H.264, HEVC is said to double the data compression ratio compared to the former at the

same level of video quality. It is targeted at next-generation communication and other application including HDTV display. In view of this, a new application for background subtraction using the CTU block features is presented in Chapter 5.

Table 4.2 Overall Quantitative Comparison on H.264 encoded sequences of ChangeDetection.net [130] Data set

Method	Average Recall		Average Precision		Average F-Measure		Average speed (fps)		Avg. Rank
	1000 KB/s	200 KB/s	1000 KB/s	200 KB/s	1000 KB/s	200 KB/s	1000 KB/s	200 KB/s	
Proposed (Luv)	**0.8202(1)**	**0.7470(2)**	**0.8411(1)**	**0.7932(1)**	**0.8271(1)**	**0.7579(1)**	417.8(3)	435.2(3)	**1.6250**
Proposed (YCbCr)	0.8116(2)	0.7189(3)	0.8097(2)	0.7276(3)	0.8058(2)	0.7161(3)	**436.1(1)**	**453.6(1)**	2.1250
Madalena [131]	0.8016(3)	0.8016(1)	0.7316(4)	0.7316(2)	0.7283(3)	0.7283(2)	7.3(6)	7.3(6)	3.3750
Dey et. al [107]	0.7117(5)	0.6433(5)	0.7998(3)	0.7135(4)	0.6856(4)	0.6758(5)	431.8(2)	444.3(2)	3.7500
Chen et. al [141]	0.7691(4)	0.7076(4)	0.7305(5)	0.6689(5)	0.6513(5)	0.6782(4)	115.4(5)	133.4(5)	4.8750
Poppe et. al [117]	0.7016(6)	0.6379(6)	0.6651(6)	0.5497(6)	0.6322(6)	0.5938(6)	138.7(4)	151.4(4)	5.5000

****Major portion of this Chapter taken from the following publication**

B. Dey and M. K. Kundu, "Enhanced Macroblock Features for Dynamic Background Modeling in H.264/AVC Video Encoded at Low-Bitrate," IEEE Transactions on Circuits and Systems for Video Technology, (Accepted).

Chapter 5

Foreground Extraction from HEVC Compressed Video using CTU Features

5.1 Introduction

While surveillance video is the biggest source of unstructured Big Data today, the emergence of High-Efficiency Video Coding (HEVC) standard is poised to have a huge role in lowering the costs associated with transmission and storage. Among the benefits of HEVC over the legacy H.264/AVC, is a staggering 40 percent or more bitrate reduction at the same visual quality. Given the bandwidth limitations, video data is compressed essentially by removing spatial and temporal correlations that exist in its uncompressed form. This causes compressed data, which is already de-correlated, to serve as a vital resource for machine learning with significantly fewer samples for training. With notable contributions [119]–[124] having made for further reducing the bandwidth required for storage/transmission of surveillance video, lots of media applications and products are currently pursuing the HEVC support. In literature, until recently there were no well-known algorithms for video content interpretation and analysis using features derived specifically from HEVC coded video. Off late, a novel research effort toward derivation of visual content using HEVC bitstream semantics [124] has been reported.

In this chapter, an efficient approach to foreground extraction / segmentation is proposed using novel spatio-temporal de-correlated block features (ST-DBF) extracted directly from HEVC compressed video. Most related techniques, in contrast, work on uncompressed images claiming significant storage and computational resources not only for the decoding process prior to initialization, but also for the feature selection/extraction and background modeling stage following it. The proposed approach has been qualitatively and quantitatively evaluated against several other state-of-the-art methods.

A low-complexity technique for foreground segmentation is proposed in this chapter. The method which relies on ST-DBF extracted from HEVC coded bitstream semantics. Real-time performances with accuracy comparable to those of pixel-based methods are targeted. The major contributions of the proposed method are summarized as follows:

1) A set of two features that sufficiently describe block coding units of HEVC compressed video is proposed. Video compression, in general, entails transformation of statistically correlated (or redundant) visual signals into highly de-correlated bitstream. As a matter of fact, the lesser the correlation between input signals, the higher is the compression achieved. The new HEVC coding standard convincingly outperforms its predecessors, *i.e.*, H.264/AVC, MPEG-4 Visual, etc.

in terms of compression efficiency. Consequently, HEVC coded video provides the best source of de-correlated data; the features derived from which enables faster machine learning using significantly fewer samples for training. It's important to mention here, that unlike the proposed ST-DBF, uncompressed natural images are typically high-dimensional data containing mostly redundant features. This pose severe computational challenges for pixel-based methods while learning and adapting to background models in real-time;

2) Unlike most pixel-based methods that require significant storage and computational overhead for decoding a compressed video prior to initialization, we propose its efficient reuse for the development of a block-based background model.
3) Offline or batch processing for initialization and update of background model parameters are completely replaced with an online process by using proposed recursive formulations.

5.2 The CTU features

As aforementioned, the frames in HEVC are coded in units of CTU. In HEVC compressed video, motion vectors and transform coefficients of a CTU registers only incremental changes occurring between adjacent frames. Major changes correspond to moving objects, while others are due to non-stationary elements of the background. It was verified, using a video codec analyzer, that the CTUs enclosing parts of moving objects contained motion vectors and transform coefficients of significantly higher energy than those normally corresponding to the scene background. Hence, the energy associated with the prediction information (*i.e.*, motion vectors, reference indices) and the prediction error/residual (i.e., transform coefficient levels) of a CTU are adopted as indicators or features of a potential foreground activity/motion in a coded video sequence. The features associated with the prediction and the residual are, hereafter, denoted by non-negative random variables y and x respectively.

Without loss of generality, we assume C CTUs per encoded picture or frame. CTUs in a given picture are addressed/indexed numerically using a location parameter $idx \in \{0, 1, ..., C-1\}$ in the raster-scan order starting with $idx=0$ for the CTU at the top-left-hand corner. Specifically, the features corresponding to the CTU located at idx in frame t are denoted as the 2×1 pattern vector

$$\vec{F}_{t,idx} = (x_{t,idx}, \quad y_{t,idx})^T$$

where $x_{t,idx}$, $y_{t,idx}$ are particular instances of the variables x and y respectively. Computation of $x_{t,idx}$ and $y_{t,idx}$ are described in the following sections.

5.2.1 Computation of CTU feature $x_{t,idx}$

We describe $x_{t,idx}$ as the energy associated with residual transform coefficients the CTU located at idx in frame t. As discussed earlier, the residuals obtained by subtracting the predicted pixels from those of the target image blocks are subjected to transform coding and quantization. HEVC specifies integer approximations of the DCT for transform sizes 32x32, 16x16, 8x8, and 4x4 [142]. Alternatively, the discrete sine transform (DST) is used only for 4x4 intra-predicted luma TBs. In the following, we propose a statistical formulation of $x_{t,idx}$ based on the following CB information extracted from the RQT: 1) the quantization parameter QP; and 2) the number of bits B consumed in coding the residual transform coefficients for each TB.

The two-dimensional auto-regressive models of the first order, i.e., 2-D AR(1), have extensively been used to represent natural images owing to its capability to provide robust estimations for the intensity of pixels by a small number of parameters. The residues practically have a zero mean value; hence, we model a residual TB as stationary 2-D AR(1) signal source X of the form

$$X(i,j) = \rho_h X(i-1,j) + \rho_v X(i,j-1) - \rho_h \rho_v X(i-1,j-1) + \xi(i,j), \tag{5.1}$$

where ξ is a zero-mean white noise with unit variance, and ρ_h, ρ_v denote the first-order horizontal and vertical correlation coefficients respectively. Under stationary conditions, the 2-D correlation function is separable and may be expressed as [143]

$$R(i,j) = \sigma_X^2 \rho_h^{|i|} \rho_v^{|j|}, \tag{5.2}$$

where

$$\sigma_X^2 = 1/[(1-\rho_h^2)(1-\rho_v^2)].$$

Considering a generic $N \times N$ residual TB, the resultant block of transformed coefficients Z may be specified as a matrix product

$$Z = AXA^T, \tag{5.3}$$

where A is the $N \times N$ orthogonal transformation basis matrix. For DCT (Type-2), the (i, j)th element of A is

$$A(i,j)=\begin{cases} 1/\sqrt{N} & i=0, 0\le j<N \\ \sqrt{2/N}\cos\left(i\pi(2j+1)/2N\right) & 1\le i<N, 0\le j<N, \end{cases} \tag{5.4}$$

while the same for DST (Type-7) being

$$A(i,j)=\frac{2}{\sqrt{2N+1}}\sin\frac{(2i+1)(j+1)\pi}{2N+1} \quad 0\le i,j<N. \tag{5.5}$$

The statistical distributions of transform coefficients are best modelled in literature [126] as a zero mean Laplacian pdf, i.e.,

$$f(z)=\frac{1}{2\lambda}\exp\left(-|z|/\lambda\right), z\in\mathbb{R} \tag{5.6}$$

where z is the coefficient value of a particular frequency component, and $\lambda>0$ being a parameter that determines the coefficient variance, *i.e.*, $2\lambda^2$. Under the condition (2), the coefficient variance of the (u, v)th component of a TB, say $\sigma_Z^2(u,v)$,, may be given as [144]

$$\sigma_Z^2(u,v)=\sigma_X^2\left[ARA^T\right]_{u,u}\left[ARA^T\right]_{v,v}, \tag{5.7}$$

where $[\cdot]_{(u,u)}$ is the (u, u)th element of the argument matrix. Therefore, we express the probability distribution of the coefficients of Z as a weighted mixture of N^2 Laplacian densities (5.6), each corresponding to individual frequency component as

$$f_{TB}^N(z)=\frac{1}{\sqrt{2}N^2}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}\frac{1}{\sigma_Z(u,v)}\exp\left(-\frac{\sqrt{2}}{\sigma_Z(u,v)}|z|\right). \tag{5.8}$$

As discussed earlier, the transform coefficients are quantized, with the degree of quantization signaled by the RQT using QP ∈ {0, 1, 2, ..., 51}. In general terms, quantization of an input coefficient z is performed according to

$$k=\text{round}(z/Q)=\text{sgn}(z)\left\lfloor\left(|z|+f\right)/Q\right\rfloor, \tag{5.9}$$

where k ∈ {0, ±1, ±2, . . .} is the mapped quantization level for all values of z in the corresponding quantization interval, Q represents the width of quantization intervals determined by N, QP as [142]

$$Q=2^{21+\lfloor QP/6\rfloor-\log_2 N}, \tag{5.10}$$

and f is a rounding offset parameter equal to Q/3 and Q/6 for the intra-predicted and the inter-predicted CBs respectively. From (5.9), it is easy to check that all values of z in the interval $(-Q+f, Q-f)$ are mapped to k=0. Similarly, for k = −1, −2, −3, … the respective quantization intervals are $((kQ-Q+f), (kQ+f)]$, while those for k = 1, 2, 3, ... being equal to

[($kQ-f$), ($kQ+Q-f$)). Therefore, the probability $P(z_k)$ that a random input coefficient of Z is mapped to level k, may be analytically expressed as the definite integral of (5.8) over the kth quantization interval

$$P(z_k)=\begin{cases}\int_{(kQ-Q+f)}^{(kQ+f)} f_{TB}^{N}(z)dz=\dfrac{1}{2N^2}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}\exp\left(\frac{\sqrt{2}(kQ+f)}{\sigma_Z(u,v)}\right)\left(1-\exp\left(-\frac{\sqrt{2}Q}{\sigma_Z(u,v)}\right)\right) & \text{for } k<0\\ \int_{(-Q+f)}^{(Q-f)} f_{TB}^{N}(z)dz=\dfrac{1}{N^2}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}\left(1-\exp\left(-\frac{\sqrt{2}(Q-f)}{\sigma_Z(u,v)}\right)\right) & \text{for } k=0.\\ \int_{(kQ-f)}^{(kQ+Q-f)} f_{TB}^{N}(z)dz=\dfrac{1}{2N^2}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}\exp\left(\frac{\sqrt{2}(-kQ+f)}{\sigma_Z(u,v)}\right)\left(1-\exp\left(-\frac{\sqrt{2}Q}{\sigma_Z(u,v)}\right)\right) & \text{for } k>0\end{cases} \tag{5.11}$$

Using the level information k, the decoding process simply reconstructs the quantized coefficients z_k as

$$z_k = kQ. \tag{5.12}$$

Please note that a given k, the computation process of z_k and $P(z_k)$ requires only N and QP (which is related to Q) be known. Using (5.11) and (5.12), the energy x_{TB} associated with the coefficients of an $N\times N$ TB may be given as

$$x_{TB}(N,QP)=\sum_{k=-\infty}^{\infty} z_k^2 P(z_k) = 2\sum_{k=1}^{\infty} z_k^2 P(z_k),$$

which on simplification yields

$$x_{TB}(N,QP)=\frac{Q^2}{N^2}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}\frac{\left(1+\exp\left(-\frac{\sqrt{2}Q}{\sigma_Z(u,v)}\right)\right)\exp\left(-\frac{\sqrt{2}(Q-f)}{\sigma_Z(u,v)}\right)}{\left(1-\exp\left(-\frac{\sqrt{2}Q}{\sigma_Z(u,v)}\right)\right)^2}. \tag{5.13}$$

While the values of N and QP directly from the RQT, $\sigma_Z(u,v)$ is indirectly dependent on ρ_h, ρ_v via (5.7) and (5.2). We assume, without loss of generality, that the residual inter-pixel correlations are the same in both horizontal and vertical directions, i.e., $\rho_h = \rho_v = \rho$ (say). Therefore, the value of ρ is estimated by equating an expression for the number of bits B consumed in coding the residual TB with its actual value obtained from the RQT during decoding.

We formulate an expression of B as follows. Recall that the lower bound on the average number of bits required to encode a quantized coefficient may be given by the (discrete) entropy measure

$$H(Z)=-\sum_{k=-\infty}^{\infty} P(z_k)\log_2(P(z_k)). \tag{5.14}$$

The entropy of a mixture density, in general, cannot be obtained in a closed form due to

its inherent difficulty in evaluating the logarithm of a sum of exponential functions. However, we observe that the entropy $H(Z)$ is a concave function of the probability distribution of a randomly picked coefficient of Z, which is considered to be a mixture of N^2 Laplacian distributed variables of equal weightage, say $z_{0,0}, z_{0,1}, \ldots, z_{N-1,N-1}$, *i.e.*,

$$Z = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} \frac{1}{N^2} z_{u,v}.$$

Therefore, by Jensen's inequality, we have

$$H(Z) = H\left(\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} \frac{1}{N^2} z_{u,v}\right) \geq \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} \frac{1}{N^2} H(z_{u,v}) = H_{approx}(Z),$$

which is a lower bound on $H(Z)$. Simplification of $H_{approx}(Z)$ yields

$$H_{approx}(Z) = \frac{1}{N^2}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} H(z_{u,v}), \tag{5.15}$$

where

$$\begin{aligned} H\left(z_{u,v}\right) = &-\left(1-\exp\left(-\tfrac{\sqrt{2}(Q-f)}{\sigma_Z(u,v)}\right)\right)\log_2\left(1-\exp\left(-\tfrac{\sqrt{2}(Q-f)}{\sigma_Z(u,v)}\right)\right) \\ &-\frac{\sqrt{2}\exp\left(-\tfrac{\sqrt{2}(Q-f)}{\sigma_Z(u,v)}\right)}{\sigma_Z(u,v)\ln 2}\left(f-\frac{Q}{1-\exp\left(-\tfrac{\sqrt{2}Q)}{\sigma_Z(u,v)}\right)}\right) \\ &-\exp\left(-\tfrac{\sqrt{2}(Q-f)}{\sigma_Z(u,v)}\right)\left(\log_2\left(1-\exp\left(-\tfrac{\sqrt{2}Q}{\sigma_Z(u,v)}\right)\right)-1\right). \end{aligned}$$

The expression for the number of transform coding bits B for an $N\times N$ TB is given by

$$B = N^2 H_{approx}(Z) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} H(z_{u,v}). \tag{5.16}$$

Substituting the values of B, Q, and f in (5.16), the unknown parameter ρ unknown parameter is evaluated using numerical methods. As discussed earlier, the value of ρ is used to compute $\sigma_Z(u,v)$, which in turn is substituted in (5.13) to evaluate x_{TB}. Given the fact that B and QP assume only discrete non-negative values from a limited range, we enumerate all possible combinations of (B, QP) for all $N \in \{4,8,16,32\}$ and construct lookup tables (LUTs) containing precomputed values of x_{TB}. The use of LUTs, therefore, replaces the complexity of repetitive numerical computations at run-time with only a few lookup operations.

We assume p TBs of sizes $N_1\times N_1$, $N_2\times N_2$, ..., $N_p\times N_p$ comprising a given CB. Therefore, the energy x_{CB} of the CB with known QP (which is fixed for all constituent TBs) and size $M\times M$ may be expressed as the weighted sum of the energy associated with

individual TBs, i.e.,

$$x_{CB}(M,QP)=\frac{1}{M^2}\sum_{i=1}^{p}N_i^2 x_{TB}(N_i,QP). \tag{5.17}$$

A normalization factor or weight (N_i^2/M^2) is associated with the ith TB in proportion to the CB area it represents. Finally, assuming q CBs of sizes $M_1\times M_1$, $M_2\times M_2$, ..., $M_q\times M_q$ and respective quantization parameters QP_1, QP_2, …, QP_q comprising a given CTU, we have

$$x_{t,idx}=\frac{1}{L^2}\sum_{i=1}^{q}r_i M_i^2 x_{CB}(M_i,QP_i), \tag{5.18}$$

where r_i is a multiplier equal to 2/3 or 1/6 accordingly as pixels of the ith CB are of luma (Y) or chroma components (Cb or Cr) respectively. The multiplier ensures that the weights sum up to unity, i.e., $\sum_{i=1}^{q}(r_i M_i^2/L^2)=1$.

It may be noted that the proposed formulation of $x_{t,idx}$ is based on the transform domain statistics using only coded information such as B, QP, N, f for each PB/TB rather than actual coefficients decoded from a compressed video. Considering the fact that discrete trigonometric transforms output exactly the same number of coefficients as input pixels, the statistical formulation (5.18) helps us evaluate $x_{t,idx}$ without involving a complexity equivalent to those of the pixel-based methods. In fact, the proposed formulation of F_1 replaces all run-time computations with a few lookup operations. In Section 5.4.1, we compare the predicted values of $x_{t,idx}$ with those that are computed directly from the decoded coefficients.

5.2.2 Computation of CTU feature $y_{t,idx}$

Unlike $x_{t,idx}$ which quantifies the energy associated with residual or prediction error, $y_{t,idx}$ is the energy associated with CTU prediction. The prediction mode is signaled at the CB-level as being intra or inter. Each luma and chroma CBs constituting a CTU is split into one, two, or four PBs. It may be noted that the PBs are essentially rectangular (instead of being a square) in the case when a CB is split into two partitions. The intra-prediction mode of a CB implies that the constituent PBs were predicted from previously coded samples from one (uni-directional, i.e., either forward or backward) or two (bi-directional – both forward as well as backward) reference frames.

It may be noted that the motion vector magnitudes corresponding to a given PB encode the predicted part of local incremental changes occurring in a scene. Therefore, the prediction

signal energy of a CB is computed as the weighted sum of the squared motion vector magnitudes of all constituent PBs, where the weights represent the portion (in terms of sample size) of the CB accounted for by each PB. Formally, we denote $(h_{i,j}, v_{i,j})$ as the motion vector corresponding to the *i*th PB in the *j*th predicted direction (either forward or backward), where $h_{i,j}$ and $v_{i,j}$ represent the offsets in the horizontal and the vertical directions respectively. Let $s_{i,j}$ be the reference frame index corresponding to $(h_{i,j}, v_{i,j})$. The value of $s_{i,j}$ implies that the current frame and the frame being referenced are at a temporal distance of $(s_{i,j} + 1)$, between which a change due to motion/activity must have occurred. Hence, the individual energy term associated with the squared motion vector magnitudes, i.e., $(h_{i,j}^2 + v_{i,j}^2)$ is normalized by $(s_{i,j} + 1)$. Furthermore, let d_i be equal to 1 or 2 accordingly as the *i*th PB is uni-directionally or bi-directionally predicted. We assume a total of p PBs of rectangular sizes $W_1 \times H_1$, $W_2 \times H_2$, ..., $W_p \times H_p$ constituting a given M×M CB. Therefore, the energy y_{CB} associated with the CB is given by

$$y_{CB}(M) = \frac{1}{M^2} \sum_{i=1}^{p} \frac{W_i H_i}{d_i} \sum_{j=1}^{d_i} \left((h_{i,j}^2 + v_{i,j}^2) / (s_{i,j} + 1) \right), \tag{5.19}$$

where weights $(W_i H_i / M^2)$ and $(1 / d_i)$ are respectively due to the sample size and the number of motion vectors associated with the *i*th PB.

Finally, assuming q CBs of sizes $M_1 \times M_1$, $M_2 \times M_2$, ..., $M_q \times M_q$ comprising the luma CTB (since the same prediction information used for the luma components is used by the chroma components as well) of a given CTU, we have

$$y_{t,idx} = \frac{1}{L^2} \sum_{i=1}^{q} M_i^2 y_{CB}(M_i). \tag{5.20}$$

It may be noted that the minimum PB size is 8×4 or 4×8 for unidirectional prediction and 8×8 for bi-directional prediction, which accounts for a maximum of 128 motion vectors for a 64×64 CTU. Additionally, in (5.19), a maximum of five multiplications / divisions and two additions is required for each motion vector. Therefore, computation of $y_{t,idx}$ requires no more than 640 multiplications / divisions and 256 additions. The information related to PB sizes, motion vectors, and reference indices that are required to compute $y_{t,idx}$ is parsed from the CQT.

5.3 The Proposed method

As highlighted in the introductory section, the proposed foreground segmentation method

involve binary decisions at two levels of granularity as follows:

1. Performing a coarse block-level segmentation of each frame by selecting of a set of potential CTUs that are occupied fully or partially by parts of moving objects;
2. Performing a finer pixel-level segmentation by eliminating pixels from the selected CTUs that are similar (in intensity) to the corresponding background model.

Therefore, our coarse-to-fine approach requires background models to be initialized both at the CTU and the pixel level as follows. The set of initial T (say, T=200) frames of a sequence are used for unsupervised training of background model parameters. For pixel-based model initialization, the temporal median of a set of frames selected at regular intervals is computed. Although the initial sequence is ideally supposed to contain only background information, in practice however, it is difficult to get real-world sequences devoid of foreground appearances. It may be noted that the median has a breakdown value ½, i.e., it can tolerate upto 50% outliers. Therefore, the median is chosen as a robust statistic that proves useful in filtering out foreground objects appearing in the initial frames. Simultaneously, a block-based background model is initialized by computing the mean $\vec{\mu}_{idx}$ and co-variance Σ_{idx} for the cluster of CTU patterns $\vec{F}_{t,idx}$ observed over the training history $t = \{1, 2, ..., T\}$. The parameters $\vec{\mu}_{idx}$ and Σ_{idx} are computed for all $idx \in \{0, 1, ..., C-1\}$, where C is the number of clusters or CTUs per frame. Computation of $\vec{\mu}_{idx}$ and Σ_{idx} is implemented on-line via a recursive process (described later), which enables us to update the model parameters with only a few additions/multiplications for every incoming pattern.

Following the training phase, the first natural step in segmenting a new frame (with $t > T$) is to compare CTU patterns for similarity with the corresponding background cluster. Similarity is ascertained if the Mahalanobis distance D measured between the incoming pattern $\vec{F}_{t,idx}$ and its corresponding cluster (background) centroid $\vec{\mu}_{idx}$ falls below a desired threshold α. For a given frame t, let S_t be the subset of CTUs satisfying

$$D = \sqrt{\left(\vec{F}_{t,idx} - \vec{\mu}_{idx}\right)^T \Sigma_{idx}^{-1} \left(\vec{F}_{t,idx} - \vec{\mu}_{idx}\right)} > \alpha, \tag{5.21}$$

for all $idx \in \{0, 1, ..., C-1\}$. The value of α is adaptively selected, as discussed later in the section. The CTUs in S_t, comprise a coarse segmented region of the foreground in frame t. Since real object boundaries rarely follow block boundaries, pixel-level refinement of the CTUs $\in S_t$ is performed by eliminating pixels from the segmented region that are similar to the co-located pixels in the background frame. Considering a given pair of pixels with

YCbCr color coordinates $X_F \equiv (Y_F, Cb_F, Cr_F)$ and $X_B \equiv (Y_B, Cb_B, Cr_B)$ respectively from the segmented region and the background frame, let luminance differential $\Delta Y = |Y_F - Y_B|$ and chrominance differential $\Delta C = |Cb_F - Cb_B| + |Cr_F - Cr_B|$. X_F is classified as foreground if $(\Delta Y/|\Sigma_{idx}|) > t_Y$ and $(\Delta C/|\Sigma_{idx}|) > t_C$, where t_Y and t_C are decision thresholds determined empirically as 0.05 and 0.03 respectively.

For a given CTU $\in S_t$, the number of pixels which correspond to the foreground say m, determines the value of adaptive threshold α. It is initially chosen as the value corresponding to 99% prediction interval for background clusters. As a priori information about the specific shape of each cluster is not available, we assume the normally distributed case for all $idx \in \{0, 1, ..., C-1\}$. Therefore, letting $\vec{\mu}_{idx} = (\mu_x, \mu_y)^T$ and $\Sigma_{idx} = \text{Diag}(\sigma_x^2, \sigma_y^2)$, α is obtained by equating the estimated background probability $P_B = 0.99$ with its analytical expression obtained over the 2D interval $\left[(\mu_x - \alpha\sigma_x, \mu_y - \alpha\sigma_y), (\mu_x + \alpha\sigma_x, \mu_y + \alpha\sigma_y)\right]$, i.e.,

$$\begin{aligned} P_B &= \int_{\mu_y-\alpha\sigma_y}^{\mu_y+\alpha\sigma_y} \int_{\mu_x+\alpha\sigma_x}^{\mu_x+\alpha\sigma_x} \frac{1}{2\pi\sigma_x\sigma_y} \exp\left[-\left(\frac{(x-\mu_x)^2}{2\sigma_x^2} + \frac{(y-\mu_y)^2}{2\sigma_y^2}\right)\right] dxdy \\ &= \text{erf}^2\left(\frac{\alpha}{\sqrt{2}}\right) = 0.99 \Rightarrow \alpha = 2.8. \end{aligned} \tag{5.22}$$

It is observed, that if a CTU location in the current frame is occupied (partially or fully) by parts of a foreground object, then the value of P_B for the collocated CTU patterns decreases abruptly in subsequent frames. Therefore, α is adapted based on the value of m in the current frame. For an $L \times L$ CTU, we ideally have $P_B = [1 - (m/L^2)]$, which is used to compute the value of α for the collocated CTU in the next frame as

$$\alpha = \sqrt{2}\,\text{erf}^{-1}\left(\sqrt{P_B}\right). \tag{5.23}$$

Theoretically, subsequent CTU patterns for $m=L^2$ will never be detected as foreground, while for $m=0$ any CTU pattern will be considered as foreground. This will lead to a deadlock situation with the CTU patterns being continuously detected as either foreground or background. Therefore, for all practical cases $m \in \{0, 1, ..., L^2\}$, we use the approximation $P_B \approx [0.99 - (0.98m/L^2)]$ such that P_B is always restricted to the values in [0.01, 0.99]. The adaptive thresholding process is found to be effective in detecting objects that move with a higher variation of speed in the field of view and those with intermittent motion.

In order to compute/update $\vec{\mu}_{idx}$ and on-the-fly, we consider a temporal sliding

window containing a maximum of K most recent samples of $\vec{F}_{t,idx}$ from the history of location *idx*. With every incoming frame t, a new CTU pattern enters the window. If $t > K$, this causes the least recent pattern $\vec{F}_{t-1,idx}$ to first exit the window (in first-in-first-out manner) to make room for the new pattern. Therefore, after sliding past n frames of a sequence, the window remains populated with a sequence of $\mathrm{V}_n = \min(K, n)$ CTU patterns given by $\{(x_{t,idx}, y_{t,idx})^T\}_{t=\max(1,n-K+1)}^{n}$, which determine the current values of $\vec{\mu}_{idx}$ and Σ_{idx}. Given the current state n of the window, let $E_n[x^a y^b]$ denote the expected value of $(x^a y^b)$, where the ordered pair $(a, b) \in \{(1,0), (0,1), (2,0), (0,2)\}$. Using the expected value notation, $\vec{\mu}_{idx}$ and (bias corrected) Σ_{idx} is given by

$$[\vec{\mu}_{idx}]_n = (E_n[x], \quad E_n[y])^T \tag{5.24}$$

and

$$[\Sigma_{idx}]_n = \frac{V_n}{V_n - 1}\begin{pmatrix} E_n[x^2] - (E_n[x])^2 & 0 \\ 0 & E_n[y^2] - (E_n[y])^2 \end{pmatrix} \tag{5.25}$$

respectively. The off-diagonal terms in (5.24), which indicate the covariance between practically uncorrelated variables x and y, are taken as zero. The nth update step of $E_n[x^a y^b]$ is defined by the recursion

$$E_n[x^a y^b] = \begin{cases} x_{1,idx}^a y_{1,idx}^b & \text{for } n = 1, \\ \frac{1}{n}\left((n-1)E_{n-1}[x^a y^b] + x_{n,idx}^a y_{n,idx}^b\right) & \text{for } 1 < n \le K, \\ E_{n-1}[x^a y^b] + \frac{1}{K}(x_{n,idx}^a y_{n,idx}^b - x_{n-K,idx}^a y_{n-K,idx}^b) & \text{for } n > K. \end{cases} \tag{5.26}$$

It may be noted that the overall update of $\vec{\mu}_{idx}$ and Σ_{idx} using (5.23), (5.24) requires no more than 11 additions (or subtractions) and 15 multiplications (or divisions). The value of Σ_{idx} obtained as described above is susceptible outliers, i.e., foreground regions appearing in initial frames. Therefore, instead of using the pattern $(x_{t,idx}, y_{t,idx})^T$ directly for computing Σ_{idx}, we use the temporal medoid of $(x_{t,idx}, y_{t,idx})^T$, $(x_{t-1,idx}, y_{t-1,idx})^T$, and $(x_{t-2,idx}, y_{t-2,idx})^T$ as its robust counterpart. This helps obtain a robust estimate of Σ_{idx} for a negligible overhead of six floating-point comparisons in the worst case.

5.4 Experimental results and Discussion

This section is divided into three sub-sections. At first, an experimental validation of the

statistically predicted values of CTU feature x against its respective counterparts computed directly from the decoded bitstream is provided. Later, qualitative as well as quantitative comparisons of the proposed method with those of the state-of-the-art on standard datasets are provided. Finally, the impact of the size of CTU blocks on performance is discussed.

5.4.1 Validation of the Predicted Value of CTU feature $x_{t,idx}$

In Section 5.2.1, we presented a statistical formulation of the CTU residual energy x based on the information parsed from RQT. The predicted values of x are validated experimentally against their actual counterparts computed from the decoded coefficients. Fig. 5.1 illustrates the comparison of the predicted and the actual values of $\{x_{t,idx}\}_{t=1}^{4000}; idx \in \{589,706\}$; plotted on x-axis and y-axis respectively for the *Fall* sequence. The original (uncompressed) sequence was encoded in HEVC at a constant bit-rate of 500 KB/s. It is observed that the magnitudes of the predicted values are slightly lower compared to those computed directly from the decoded coefficients. This comes as a direct consequence of the fact that prediction of $x_{t,idx}$ depends on B, whose lower-bound on the average bit-rate was modeled using the entropy measure. Furthermore, the predicted value and its actual counterpart are found to be linearly correlated; in fact, the correlation coefficient was found to be greater than 0.96 for all recorded cases. The discrepancy between the predicted and the actual values, however, do not affect segmentation decisions taken using (5.23), because the Mahalanobis distance is known to be invariant under

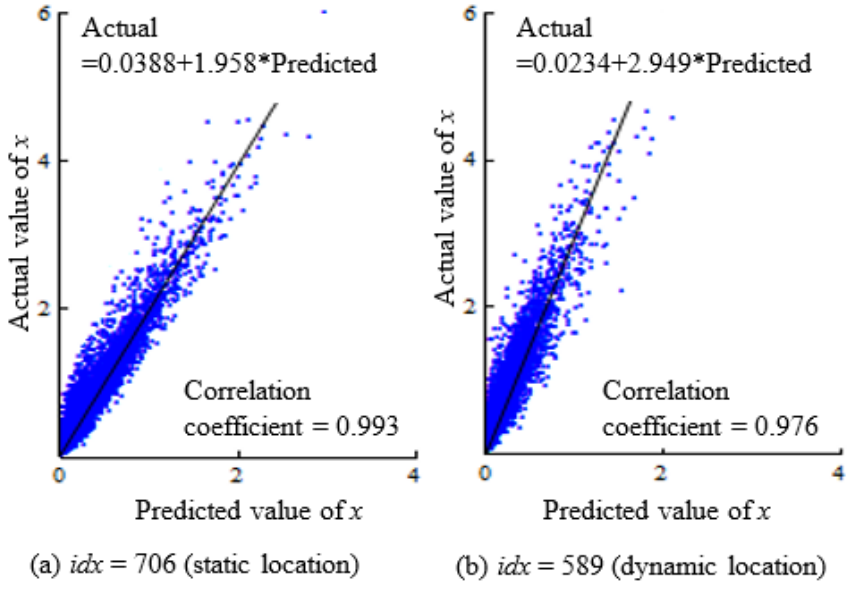


Fig. 5.1 Comparison of the predicted and the actual values of the proposed CTU feature x. Regression lines are obtained by the least square method.

arbitrary linear transformations of the feature space. Therefore, in order to support faster computation, the predicted value of x is used as a convenient and yet justified surrogate to its actual counterpart; as the latter, would otherwise have involved prohibitively high complexity due to coefficient level decoding.

5.4.2 Qualitative and Quantitative evaluation

Experiments were conducted on challenge data sets used for the Change Detection workshop (CD.net) [130] and the Background Models Challenge (BMC) [146]. The CD.net data set, unlike any other publicly available, is very challenging and comprehensive. It includes 31 real-world (non-synthetic) sequences captured in diverse

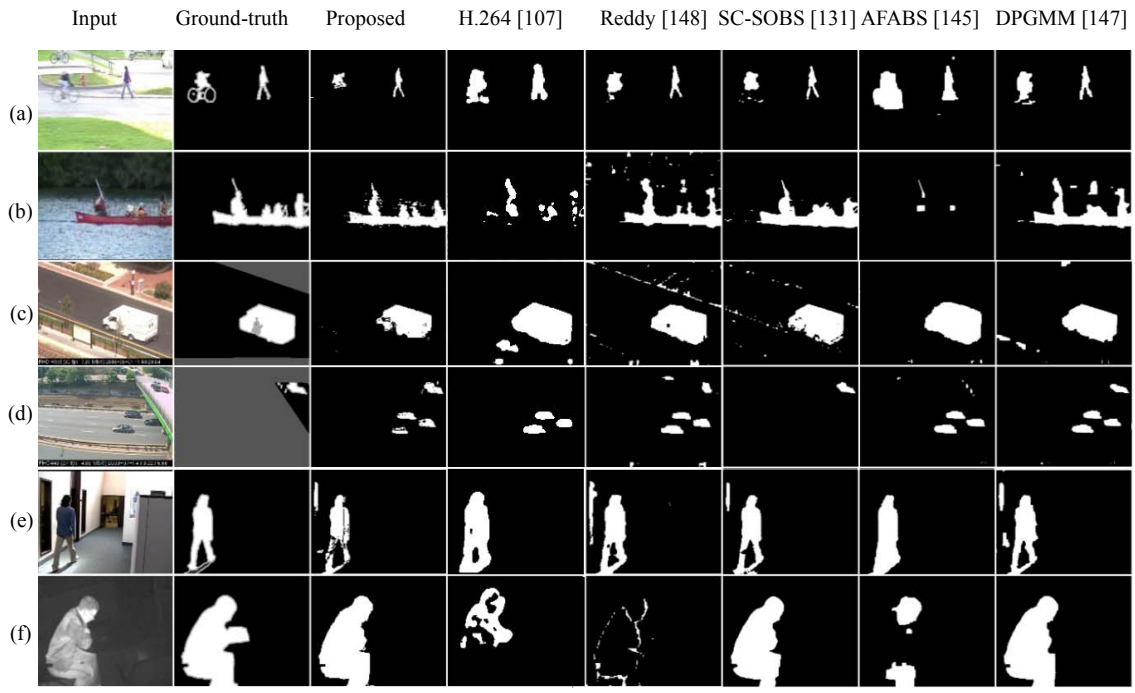


Fig. 5.2 Qualitative results are shown row-wise with two sequences selected from each category of the CD.net dataset [130]. Starting from the top, the first row (a) illustrates results from the *pedestrians* (frame# 471) sequence of the *baseline* category. Row (b) illustrates results from the *canoe* (frame# 993) sequence of the *dynamic background* category. Row (c) illustrates results of the *boulevard* (frame# 1197) from the category *camera jitter*. Row (d) depicts results from the *street light* (frame# 2185) sequence of the *intermittent object motion* category. Row (e) contains results from the *cubicle* (frame# 4987) sequence of the *shadow* category. Finally, row (f) illustrates results from the *library* (frame# 2735) sequence of the *thermal* category. The input frames, the ground-truth masks, and the output of each algorithm are arranged column-wise as indicated at the top of the figure.

environments. All sequences of the data set are accompanied by accurate ground-truth segmentation of change/motion areas for each video frame that are subject to evaluation. The BMC data, on the other hand, encapsulates both synthetic and real-world sequences

along with encrypted ground-truth masks for selected frames.

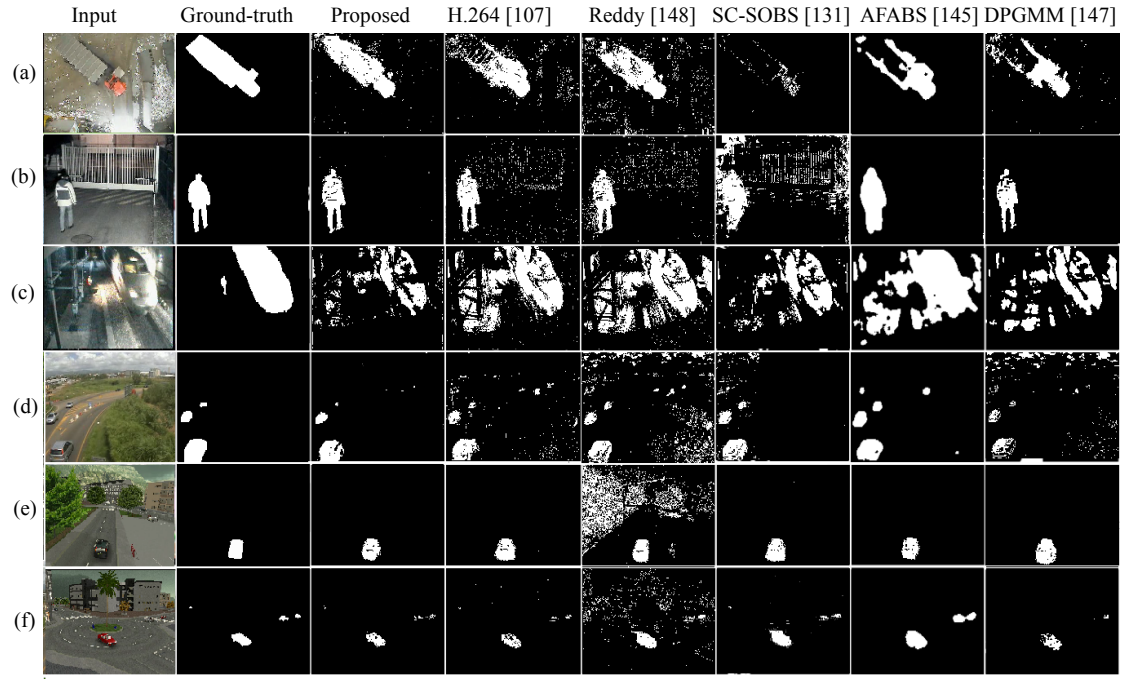


Fig. 5.3 Qualitative results on selected evaluation sequences of the BMC dataset [146]. Starting from the top, the first four rows, i.e., (a)-(d) correspond to real sequences *Video2* (frame# 1249), *Video4* (frame# 213), *Video7* (frame# 258), and *Video8* (frame# 130) respectively. Rows (e) and (f) illustrate results for synthetic sequences 112 (frame# 300) and 222 (frame# 645). The input frames, the ground-truth masks, and the output of each algorithm are arranged column-wise as indicated at the top of the figure.

All sequences were considered for evaluation, with the original sequences transcoded to HEVC format using FFmpeg [129]. The encoder configuration was set as follows: Main profile with YCbCr 4:2:0 sampling and CTU sizes 16×16, weighted prediction as applied for both uni-directional and bi-directional cases, rate control was enabled (at the CTU level) with a target bit rate of 512 MB/s for all sequences, and video frames streaming at 25 frames/second. It is important to mention here, that in most streaming video applications, a predetermined output bit rate is desired. These applications, referred to as constant bit rate (CBR) applications, use a rate control strategy ensure a target bit rate by carefully selecting a different QP for each CBs comprising a CTU. The proposed method was implemented in C (built on MingW 64-bit platform) as a software patch and integrated into the HEVC decoding module of FFmpeg.

For comparison, we chose a set of five state-of-the-art methods [147], [148], [107], [131], and [145]. Fig. 5.2 and Fig. 5.3 visually compare segmentation result of six sequences chosen randomly from the CD.net and BMC data sets respectively. As shown in the case of H.264 (previous chapter), please note that segmentation results were obtained using CIELUV as well as the native YCbCr color space for HEVC encoded videos as well.

The ground-truth and segmented frames contain black, white, and grayed-out portions, which respectively annotate the background, the foreground, and the regions that do not belong to the ROIs (which are, therefore, not evaluated). Based on similarity with the provided ground-truth data, clearly our algorithm performs qualitatively as well or better than the other techniques.

In order to compare the achieved results quantitatively with those of the state-of-the-art methods, we consider the average performance on both the data sets, as reported in Table-5.1. The metrics adopted for quantitative evaluation includes recall, precision, f-measure, and average processing speed as discussed in Chapter 3. The processing times were recorded for videos having a resolution of 720 × 420 on a PC powered by Intel Core i7-2600 3.40 GHz CPU with 16 GB RAM. To ensure an unbiased computing platform, the uses of graphics processing units (GPU) were disabled.

Table 5.1 Overall quantitative comparison on CD.net and BMC datasets

Method	Average Recall		Average Precision		Average F-Measure		Avg. speed (fps)	**Avg. Rank**
	CD.net	BMC	CD.net	BMC	CD.net	BMC		
Proposed ST-HBF(LUV)	0.8026 (2)	0.8009 (2)	**0.8408** (1)	**0.9292** (1)	**0.8201** (1)	**0.8619** (1)	**147.9** (3)	**1.5714**
Proposed ST-HBF(YCbCr)	0.7997 (4)	0.7905 (3)	**0.8339** (2)	**0.9228** (2)	**0.8164** (2)	**0.8515** (2)	**169.1** (1)	**2.2857**
DPGMM [147]	**0.8275** (1)	**0.8027** (1)	0.7928 (4)	0.8018 (4)	0.8098 (3)	0.8022 (3)	6.9 (6)	3.1428
Reddy et al. [148]	0.7935 (5)	0.7811 (4)	0.7294 (6)	0.7991 (5)	0.7601 (5)	0.7900 (4)	9.2 (4)	4.7143
SC-SOBS [131]	0.8016 (3)	0.7316 (6)	0.7316 (5)	0.7916 (6)	0.7650 (4)	0.7604 (5)	7.3 (5)	4.8571
H.264 (CBR) [107]	0.5899 (7)	0.6612 (7)	0.7986 (3)	0.8916 (3)	0.6786 (7)	0.7593 (6)	151.1 (2)	5.0000
AFABS [145]	0.7577 (6)	0.7391 (5)	0.7191 (7)	0.7794 (7)	0.7379 (6)	0.7587 (7)	5.2 (7)	6.4286

The evaluation is based on a average ranking system with the average rank computed over individual performance metrics and across both the data sets. The higher the scores of the metrics, the better is the performance and the rank. The overall quantitative results based on all sequences of the CD.net, and the BMC data set are summarized in Table-5.1. The performance scores in each evaluation category were converted to ordinal ranks, which are included in the parentheses alongside. The boldface values for each of the considered metrics indicate the best results achieved by the compared methods. Finally, for each method, the average rank has been obtained by computing the arithmetic mean achieved according to each single metric.

Prior to a formal evaluation, it is important to realize that compression essentially introduces visual artefacts. As a result, foreground segmentation directly using features of

compressed video is accurate to the extent that can only be obtained after transcoding. Despite the above fact, it is observed that the proposed method obtained better results when it came to the overall performance comparison in terms of segmentation accuracy and processing speed. Table-5.2, on the other hand, analyzes the segmentation performance (using F-measure) on various sequence categories. Our performance is best demonstrated in the baseline and the dynamic background category, which includes scenes depicting strong background motion: boats on shimmering water (result from canoe sequence shown in Fig. 4-b), cars and pedestrians passing through background containing a fountain, waving tree, etc.

Table 5.2 Quantitative evaluation on CD.net dataset (using average f-measure for each sequence)

Method	Baseline	Camera Jitter	Dynamic Background	Intermittent Object Motion	Shadow	Thermal	**Average Rank**
Proposed ST-HBF (Luv)	**0.9402** (1)	0.7151 (3)	**0.8283** (1)	0.5351 (3)	0.8644 (2)	0.7353 (2)	**2.0000**
Proposed ST-HBF (YCbCr)	**0.9342** (2)	0.7124 (3)	**0.8283** (1)	0.5351 (3)	0.8644 (2)	0.7353 (2)	**2.0000**
DPGMM [147]	0.9333 (3)	0.7477 (2)	0.8137 (2)	0.5418 (2)	0.8128 (3)	**0.8134** (1)	**2.0000**
Reddy et al. [148]	0.9206 (5)	**0.7784** (1)	0.6823 (4)	0.4955 (5)	**0.8671** (1)	0.5957 (6)	3.5000
SC-SOBS [131]	0.9286 (4)	0.7051 (4)	0.6686 (5)	**0.5918** (1)	0.7784 (6)	0.6923 (4)	3.8333
H.264 (CBR) [107]	0.8349 (6)	0.6737 (5)	0.7829 (3)	0.4679 (6)	0.7871 (5)	0.6328 (5)	4.8333
AFABS [145]	0.7294 (7)	0.6036 (6)	0.5764 (6)	0.5334 (4)	0.8103 (4)	0.7183 (3)	4.8333

The proposed method is found to be significantly faster than the normal streaming rate, i.e., 25-30 frames/second. It is also overwhelmingly faster compared to any of the related pixel-based methods. As discussed in sections 5.2 and 5.3, the computation required for each CTU in terms of the number of comparisons, additions/subtractions, and multiplications/divisions cost up to a constant factor. Consequently, the complexity of the proposed method evaluates to be of the order $(c1 \cdot \text{card}(S_t) + c2 \cdot (C - \text{card}(S_t)))$, where $\text{card}(S_t)$ (i.e., cardinality of S_t) is the number of selected CTUs requiring pixel level processing in the given frame t, C is the total number of CTUs per frame, and c1, c2 are constants. With $\text{card}(S_t) << C$ being the dominant factor, the running time scales linearly with $\text{card}(S_t)$, incurring only a negligible overhead in addition to the regular decoding cost claimed by each frame. Given today's limitation on network bandwidth and overwhelmingly high computational demands in real-time, the qualitative and quantitative

analyses significantly favor the proposed method in comparison to the state-of-the-art.

It is worth mentioning that the choice of the size $L \times L$ of each CTU (i.e., whether L = 16, 32, or 64) is entirely an encoding decision. For wider resolution videos, HEVC benefits from using larger CTU sizes in terms of bandwidth. To find the effect of CTU size on segmentation accuracy and processing speed, two sequences form the CD.net data set, i.e., *Fall* and *PETS2006*, were encoded under three different CTU sizes L = {16, 32, 64}. Experimentally, it is observed that the average processing speed increases significantly as L grows from 16 to 64. This is perfectly in line with the earlier discussion on computational complexity as the number of CTUs having a larger size decreases in case the input video resolution remains fixed. However, we observed a minor dip in segmentation accuracy (assessed in terms of F-measure) for larger CTU sizes. This is because the segmentation map obtained with large CTUs corresponds to very coarse approximation of the foreground-background boundary. Fig. 6 illustrates the variation of (a) average processing speed, and (b) F-measure obtained for CTU block sizes L = 16, 32, and 64.

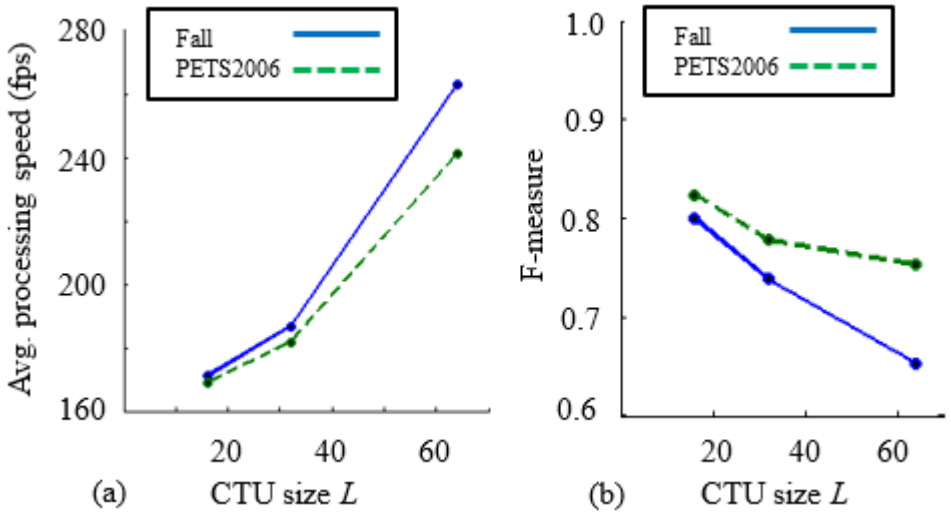


Fig. 5.4 Variation of (a) avg. processing speed and (b) F-measure against the CTU block size L = 16, 32, and 64.

5.5 Conclusion

In this chapter, a method for extracting foreground objects using novel CTU features of HEVC compressed video is proposed. The method exploits the fact that compressed HEVC video is essentially a source of highly de-correlated data having two features that sufficiently describe each CTU block. The proposed method delivers a segmentation performance comparable to those of the state-of-the-art methods while maintaining low

computational requirements that are affordable well within real-time constraints.

There are, however, room for improvements. Although the proposed method robustly handles gradual illumination changes and dynamic background, it is unable to cope with sudden illumination changes, moving cast shadows, and cases of "removed objects," i.e., stationary objects that were present in the scene during training phase and then move away. Using normalized color space for illumination invariance and an explicit model of the "removed objects" could further improve our method, basically by introducing pixel-level correspondence between frames.

So far, we have proposed background subtraction applications for popular video compression standards such as H.264 (Baseline, Main and High) and HEVC. To show the effectiveness of the proposed methods in real-life compressed raw data, we used H.264 High profile video captured by the surveillance camera at various traffic intersections in the city of Kolkata. The videos were applied for traffic intersection analysis, which is discussed in the next chapter.

****Major portion of this Chapter taken from the following publication**

B. Dey and M.K. Kundu, "Efficient Foreground Extraction from HEVC Compressed Video for Application to Real-Time Analysis of Surveillance 'Big' Data," IEEE Transactions on Image Processing, vol. 24, no. 11, pp. 3574-3585, Nov. 2015.

Chapter 6

An Application to Traffic Sequence Analysis

6.1 Introduction

With the modern socio-economic development, the number of vehicles in metropolitan cities is growing rapidly. Therefore, obtaining real-time traffic volume estimates have a very important significance in using the limited road space and traffic-infrastructure. In this chapter, an application for video-based traffic volume and direction estimation at road intersections is developed. In order to identify the vehicles from the remaining foreground objects, vehicle recognition is performed using the convolutional neural network (CNN). Compared with the usual hand-crafted features used for recognition, CNN can automatically learn features to capture complex visual variations by leveraging a large amount of labeled training data. The encoded video sequence is first detected for moving foreground regions or patches using the methods described in earlier chapters. The trained model is subsequently used to classify the detected patches as a vehicle or non-vehicle. The vehicles are tracked, and trajectory patterns are clustered using standard techniques. The number and direction of vehicles are noted, which are later compared with the manually observed values. All experiments were performed on real-life surveillance sequences recorded at four different traffic intersections in the city of Kolkata.

Vision-based techniques have been applied to vehicle detection systems for over a decade. One of the earliest works in this field is the Autoscope® [149] vehicle detection system that has been commercially deployed for traffic detections. There have been several recent investigations on vehicle classification using computer vision. Kamijo et al. [150] proposed a method for vision-based traffic monitoring and accident detection at intersections. In this method, the vehicle-detection and tracking are modeled as a labeling problem for each pixel in the image sequence. In order to detect vehicles from static images, Wu et al. [151] used a pattern classifier based on the PCA. Wavelet-transform has been used to extract texture features for PCA. In addition, Sun et al. [152] applied Gabor filters to extract textures features and subsequently verified each vehicle candidate using support vector machines (SVM). In [153], Lipton et al. used a classification metric to classify moving targets into vehicles, humans, and background clutter. Recently, Ohn-Bar and Trivedi [154] demonstrated the importance of learning appearance patterns of vehicle categories for identification and orientation estimation. A regression-analysis-based counting and classification of vehicles are presented in [155]. Despite these great efforts, most vehicle detection/counting applications are limited in a sense that do not discriminate moving vehicles from other moving objects, such as pedestrians, cycles, etc., in congested

traffic. This is because of the usage of a set of low-level hand-crafted features for vehicle detection. Of late, some deep learning architectures have shown promising results in many visual recognition tasks [156]-[158], including traffic-flow direction. Compared to other state of the art object recognition algorithms, deep learning networks - more specifically CNNs, do not require computation of specific had-crafted features. Put otherwise, the CNN is directly responsible for learning the features. The lack of dependence on prior knowledge and effort in designing domain-specific features is a major advantage.

6.2 Methodology

This section details the methods used for automatically obtaining information on traffic volumes as well as their directionality at road-intersections, i.e., intersection analysis from traffic video. To the best of my knowledge, it is the first time that the video-based intersection analysis approach has been attempted. Our objective is to develop automated statistics of the total number of motor-vehicles (excluding two-wheelers and other vehicles that usually do not attract toll) moving in or out of the connecting road-segments of a traffic intersection. For the current application, 1) only real-life closed-circuit television (CCTV) footages captured under daylight conditions were used; and 2) no discrimination was made among the various sub-categories of vehicles. A number of sample images of moving objects usually encountered in a congested traffic scene, i.e., pedestrians and vehicles are extracted manually and trained using a deep layered architecture of CNN. Using background segmentation algorithm described in earlier chapters, foreground patches corresponding to moving objects are obtained and classified with the trained model. The foreground patches classified as vehicles are tracked using a particle filter. The trajectories of the detected vehicles are then clustered using a standard trajectory clustering algorithm.

It may be noted that, detection and classification of vehicles are of paramount importance and formed the major contributory part of the current application. First, a general introduction of the CNN is given. This is followed by a description the architecture used for training and classification of vehicles. The testing phase which consists of background segmentation, tracking and trajectory clustering is discussed later.

6.2.1 Theoretical basis: Convolutional Neural Network

A CNN, as shown in Fig. 6.1, is comprised of one or more convolutional layers (often interspersed with a subsampling layer) and then optionally followed by one or more fully

connected layers as in a standard artificial neural network (ANN). The architecture of a CNN is designed to exploit the spatial correlation between the adjacent pixels of an input image. Therefore, CNNs work on 2 or 3-dimentional image data, so called maps, directly, unlike normal neural network which would concatenate these into vectors. A CNN consists of several layers. These layers, which can be of three types, are discussed as follows:

1. Convolutional: The input to a convolutional layer, say x, is a $M\times N\times K$ image map where M is the height, N is the width of the input, and K is the number of channels per pixel, e.g. an RGB image has K=3. The convolutional layer will have K' filters (or kernels), say w, of size $m\times n\times K$ where $m < M$ and $n < N$. The size of the filters gives rise to the locally connected structure which are each convolved with x to produce output feature map y of size $(M–m+1)\times(N–n+1)\times K$'. Each output feature map, in turn, may be subjected to convolutions to generate sub-feature maps for the next layer. Symbolically, the convolution operation to compute the lth output feature map is given as,

$$y^l_{pqr} = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\sum_{k=0}^{K-1} w^l_{ijkr} x^l_{i+p,j+q,k}, \tag{6.1}$$

   where the superscripts denote the indices of the feature maps or layers. Then, the convolutional layer applies its nonlinearity after adding a bias b^l as

$$x^{l+1}_{pqr} = f\left(y^l_{pqr} + b^l\right), \tag{6.2}$$

   where f is a non-linear activation function. Training process of CNN will learn w, as parameters of the convolutional layer. The choice of the activation function in the convolutional layer has a huge impact on the classification performance. There are several choices of the activation function, *viz.* hyperbolic tangent, sigmoid and Rectified Linear Unit (ReLU) [170].

2. Sub-sampling or pooling layer: Each convolutional layer in the network may optionally be followed by a sub-sampling layer. After obtaining the convolved features as described earlier, one may decide the size of the region, say $m\times n$ to pool the convolved features over. For each $m\times n$ regions, the maximum (or mean) of the activation is taken to obtain the pooled convolved features. An equivalent max-pooling is defined by

$$y_{ijk} = \max\{y_{pqk} : i \le p < i+m, j \le q < j+n\}, \tag{6.3}$$

3. Fully-Connected: Finally, after a series of convolutional and sub-sampling layers, the high-level decision making is processed via fully connected layers. A fully

connected layer is connected to all neurons in the previous layer (be it fully connected or convolutional) and each of them connects it to every single neuron it has. It will perform the same duties found in standard ANNs and attempt to produce class scores from the activations, to be used for classification. It is also suggested that ReLU may be used between these layers, as to improve performance.

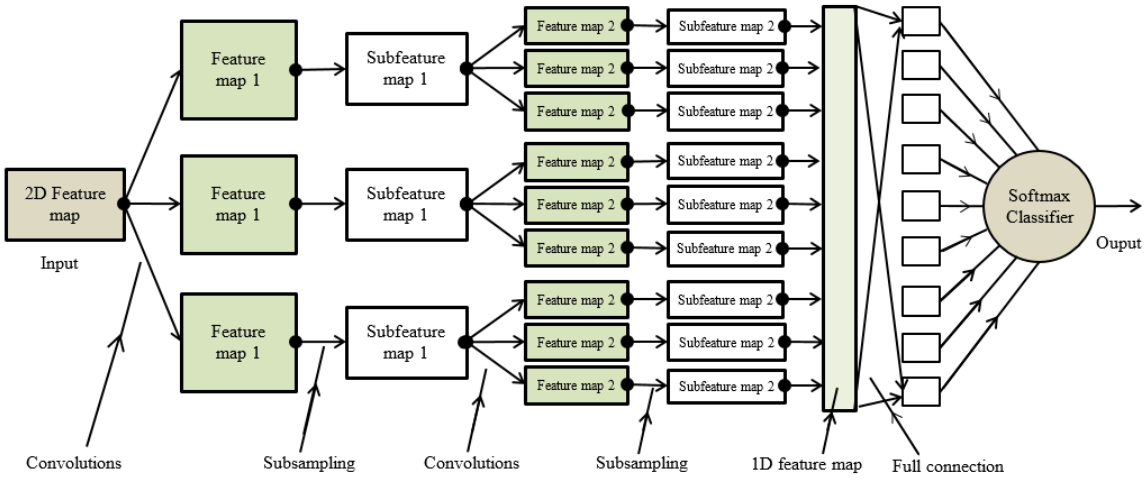


Fig. 6.1 Architecture of Convolutional neural networks.

During the training process, the weight parameters of a CNN $W = (w^1, w^2, \ldots, w^l)$ and bias vector $b = (b^1, b^2, \ldots, b^l)$ should be trained such that the overall CNN function $z = F(x; W, b)$ achieves a pre-defined goal. In most cases, the goal is to fit F over the entire distribution of the data. Let us consider input-output relations $(x_1, z_1), (x_2, z_2), \ldots, (x_n, z_n)$ where x_i is the input map and z_i the corresponding output values. Also, let $E^i = \ell(z_i, \hat{z}_i)$ represent the loss ℓ for predicting $\hat{z}_i$ instead of z_i. The empirical cost E of the CNN is obtained by computing the mean for all the examples

$$E(W,b) = \frac{1}{n}\sum_{i=1}^{n} E^i. \tag{6.4}$$

The simplest method is to minimize E using gradients. This is done by taking a single example (x_i, z_i) randomly from the training set in an iterative manner. An estimate of the gradient is then computed based on the error E^t for that example, and then weights and bias for iteration t are updated as

$$\left.\begin{aligned} W_{t+1} &= W_t - \eta \frac{\partial E^t}{\partial W} \\ b_{t+1} &= b_t - \eta \frac{\partial E^t}{\partial b} \end{aligned}\right\} \tag{6.5}$$

where $\eta \in \mathbb{R}+$ is the learning rate.

Typically, the last layer of a CNN is a fully-connected logistic regression layer, where each unit of the output reflects a class membership probability: Usually, to discriminate between k output classes, a fully connected layer with k neurons is added. This is the output layer, which takes as input the vectorized feature maps x of the layer below it as follows

$$p\left(z_i = j \mid x_i; W, b\right) = \frac{\exp\left(w_j^T x_j + b_j\right)}{\sum_{i=1}^{k} \exp\left(w_i^T x_i + b_i\right)} \tag{6.6}$$

where $p(z_i = j|x_i; W, b)$ is the probability of z_i being in class j given input x_i with CNN parameters W, b.

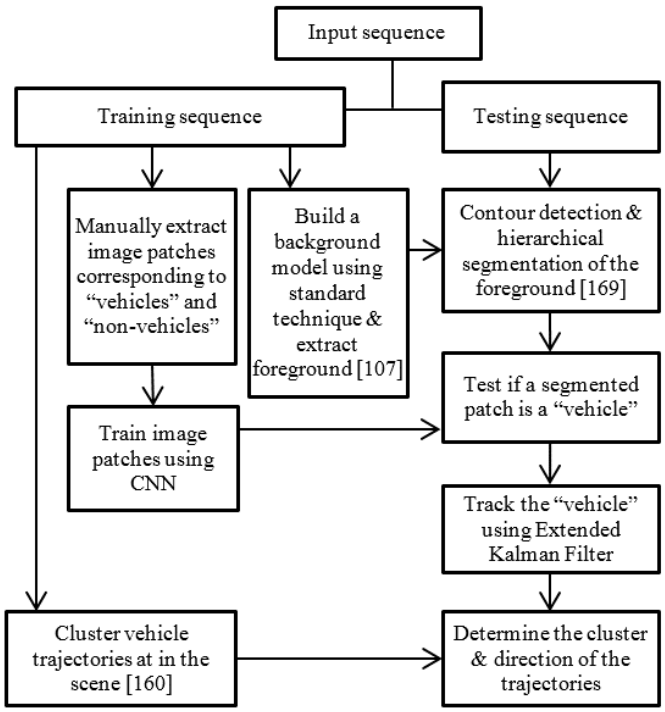


Fig. 6.2 Schematic representation of the proposed method.

6.2.2 Detailed Overview

Fig. 6.2 presents an overview of the proposed architecture. The overall process consists of training and testing phases. As mentioned earlier, sample traffic sequences from various road intersections are collected and split into two sub-sequences, which are used for

training and testing separately. From the training sequences, image patches defined by the minimum bounding rectangles corresponding to two classes of objects normally encountered in traffic, *i.e.*, "vehicle" and "non-vehicle" are manually extracted. For each category, 3000 images patches, which may be of different sizes, are obtained and rescaled to spatial dimensions of 64×64 for training with a CNN. We discuss the specific CNN architecture later. For the current application, motor driven vehicles having three or more wheels are considered as members of the "vehicle" class. Typically, these are the vehicles that attract toll and identification/counting of these vehicles is a prerequisite for proper video-based electronic toll collection and traffic regulation. Other moving objects such as pedestrians, bikes, cycles, manually-driven vans, etc. are considered as "non-vehicles." Fig. 6.3 shows illustrative examples from the two classes.

In order to extract and count the number of "vehicles" in a traffic intersection

(a) (b)

Fig. 6.3. (a) Some examples of manually extracted image patches which correspond to the vehicle class. (b) Some examples of manually extracted image patches which correspond to the non-vehicle class. These are objects encountered in a traffic scene, but are not considered a vehicle for the current application, viz. pedestrians, two-wheelers (bicycles / motor-bikes), and three-wheelers that are not driven by a motor engine.

video, it is first required to segment the regions of interest (ROIs) corresponding to moving vehicles. This requires foreground regions to be segmented from the background. Since the input videos are usually available in a compressed form, we apply a fast background modeling method [107] on the training sequence to obtain a background frame. A key issue in a congested traffic scene is that the foreground segment of the frame obtained by background subtraction often remains occluded due to other objects present in the same scene. In order to count the number of vehicles correctly it is required to segment

independently moving objects as accurately as possible. It is a familiar experience, that when an object is occluded by another, both the objects appear adjacent to the imaging plane and share a common boundary/contour. Therefore, the result of foreground segmentation is subjected to contour detection and hierarchical segmentation [169]. The image patches corresponding to minimum-bounding rectangles of the object contours thus obtained, are the probable candidates of the "vehicle" class. For each detected patch, a model trained with a CNN is applied to ensure that the detection is actually a vehicle. In order to classify a patch, we train a set of manually extracted images from the "vehicle" and "non-vehicle" classes with a CNN.

Table 6.1 The CNN Architecture

Layer index	Layer	Input Type	Filter Size	Filter Num	Stride	Pad	Output Size
0		Input	-	-	-	-	64×64×3
1	L1	Convolution	5×5×3	64	1	2	64×64×64
2		ReLU	-	-	-	-	64×64×64
3		Maxpool	2×2		2	0	32×32×64
4	L2	Convolution	5×5×64	128	1	2	32×32×128
5		ReLU	-	-	-	-	32×32×128
6		Maxpool	2×2	-	2	0	16×16×128
7	L3	Convolution	5×5×128	128	1	1	16×16×128
8		ReLU	-	-	-	-	16×16×128
9	L4	Convolution	5×5×128	128	1	1	16×16×128
10		ReLU	-	-	-	-	16×16×128
11	L5	Convolution	5×5×128	256	1	1	16×16×256
12		ReLU	-	-	-	-	16×16×256
13		Maxpool	2×2	-	2	0	8×8×256
14	L6	Fully connected	8×8×256	512	1	0	1×1×512
15		ReLU	-	-	-	-	1×1×512
16		Dropout (0.5)	-	-	-	-	1×1×512
17	L7	Fully connected	1×1×512	512	1	0	1×1×512
18		ReLU	-	-	-	-	1×1×512
19		Dropout (0.5)	-	-	-	-	1×1×512
20	L8	Fully connected	1×1×512	2	1	0	1×1×2
21		Softmax	-	-	-	-	1×1×2
22		Classification Output	-	-	-	-	1×1×2

Our specific architecture has been detailed in Table-6.1. It consists of five convolutional layers L1-L5, each with ReLU [170] activation function. Each of the layers L1, L2, and L5 are followed by a max-pooling layer. Before going into fully-connected layers, the pooling output of L5 is concatenated into one long vector. The input filter size of L6 is 8×8×256, which is the same as the output from the previous pooling layer. It is a fully-connected layer. Two more fully-connected layers L7 and L8 are appended. Dropout regularization [171], i.e., randomly setting units' activation to zero, is applied on layers L6 and L7. It may be noted that deeper learning networks generally improve upon the performance of shallow networks, but risk overfitting [156]. Recent works on CNN such as [158] deal with overfitting is case of the (very large) ImageNet dataset [172] for training

using the concept of dropout. Our training dataset is typically much smaller when compared with ImageNet; hence, it is compensated by setting a high dropout rate of 50%. The final layer L8 has two output units corresponding to two output classes (*i.e.*, vehicle & non-vehicle). The output class probabilities are predicted based on softmax regression as described in (6.6). For the current application, the cost function is formulated as follows. Recall that our CNN should compute for each image patch p a score $z = F(x; W, b)$. Therefore, it is desirable that the score be:

1. at least as large as 1 for any patch that is labelled as a member of the "non-vehicle" class *N*, i.e., $p \in N$; and
2. at most zero for any patch that is marked as a member of the "vehicle" class *V*, i.e., $p \in V$.

Based on the above requirements, we define and optimize the following objective function

$$E(W,b) = \frac{\lambda}{2}\|W\|^2 + \frac{1}{|\mathcal{V}|}\sum_{p\in\mathcal{V}} \max\{0, 1 - F(p;W,b)\} + \frac{1}{|\mathcal{N}|}\sum_{p\in\mathcal{N}} \max\{0, 1 - F(p;W,b)\}, \quad (6.7)$$

where $\lambda\|W\|^2/2$ is an additional weight decay term that penalizes large values of the parameters. Furthermore, with this weight decay parameter (for any $\lambda>0$), the cost function $E(W, b)$ is now strictly convex, and hence guaranteed to have a unique solution.

The network is trained by minimizing the objective function with respect to W and b. We do so by using a common approach called gradient descent with momentum. Given the current solution (W_t, b_t), it is updated to (W_{t+1}, b_{t+1}) by following the direction of fastest descent in order to reach the global minimum. It reduces the risk of getting stuck in a local minimum. The gradient is smoothed by considering momentum parameters $(\overline{W}_t, \overline{b}_t)$, yielding a faster convergence. The update equations are as follows

$$\left.\begin{aligned} \overline{W}_{t+1} &= \mu\overline{W}_t - \eta\frac{\partial E}{\partial W_t} \\ W_{t+1} &= W_t - \overline{W}_t \end{aligned}\right\} \quad (6.8)$$

and similarly, for the bias term. Here μ is the rate of momentum and η the learning rate. The detailed architecture of our CNN is summarized in Table-6.1. The training phase generates a network model, which is later used to classify segmented image patches as a "vehicle" or "non-vehicle" while processing a given video sequence.

As illustrated in Fig. 6.2, the image patches identified as containing objects of the "vehicle" class are tracked using the Extended Kalman filter (EKF) for tracking multiple targets. For each sequence, each of the road segments connected to the intersection is first identified and named with alphabets. The centroid of the "vehicles" tracked throughout the sequence are used as trajectories for clustering. In order to determine the number of vehicles entering/leaving the road segments of a traffic intersection and for subsequent evaluation, we demarcate and name each of them manually for identification (shown in Fig. 6.4) as point locations, A, B, …and so on, on the image plane. During the testing phase the number of "vehicles" belonging to a particular cluster and the direction are recorded. For a given vehicle trajectory, the direction of the vehicle is inferred based on the minimal Euclidean (or straight line) distance between the start/end locations of a given vehicle trajectory and the demarcated point locations. Let a trajectory T_j be represented as $T_j = \{(x_j^i, y_j^i); i = 1, \ldots, N_j\}$, where (x_j^i, y_j^i) is the estimated position of the jth target on the image plane, N_j is the number of trajectory points and j =1,…, J. J is the number of trajectories. Furthermore, let $(A_x^k, A_y^k), k = 1,2, \ldots, N$ be the coordinates of the demarcated points on the image plane. For a given vehicle trajectory T_j having the starting location at (x_j^1, y_j^1) and the ending location at $\left(x_j^{N_j}, y_j^{N_j}\right)$, the start and the end locations are assigned

(a) Chandni Chowk Crossing, Kolkata (4-way intersection)

(b) Rajabazar Crossing, Kolkata (4-way intersection)

(c) Kankurgachi Crossing, Kolkata (3-way intersection)

(d) Shyambazar Crossing, Kolkata (5-way intersection)

Fig. 6.4 Demarcation of the road segments at traffic intersections. The point locations are named as A, B, and so on, marked in white on the imaging plane

to two of the N demarcated locations as

$$\arg \min_{k=1,2,\ldots,N} \left(\left(x_j^1 - A_x^k \right)^2 + \left(y_j^1 - A_y^k \right)^2 \right) \tag{6.9}$$

and

$$\arg \min_{k=1,2,\ldots,N} \left(\left(x_j^{N_j} - A_x^k \right)^2 + \left(y_j^{N_j} - A_y^k \right)^2 \right) \tag{6.10}$$

respectively. The above formulation is based on the premise that the start and end locations of most trajectories are closest to the demarcated points in the image.

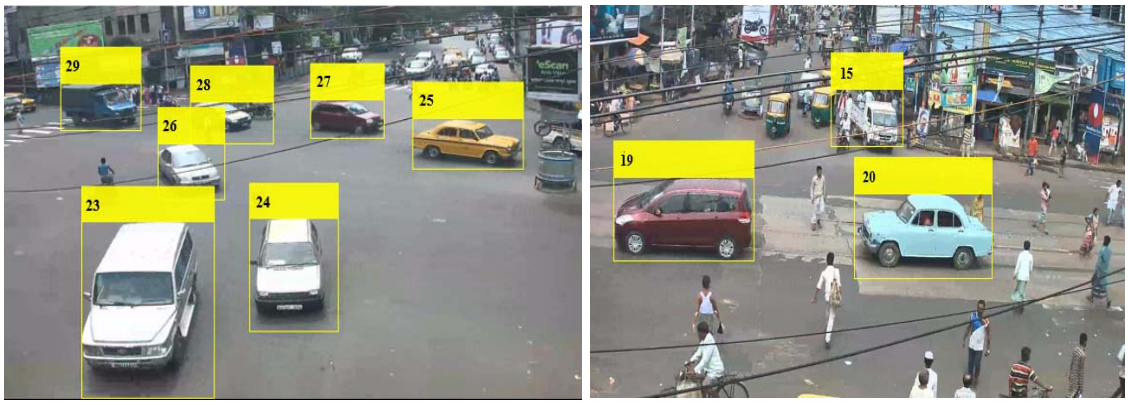
6.3 Experiments and Results

6.3.1 Data Description

For the current application, it was found that the number of existing data sets depicting intersection scenes along with ground-truth volume estimates for each road segment is severely limited. Moreover, sufficiently large number of training images for the non-vehicle class is not publicly available. Therefore, for our specific application, we obtained a real-life data set from the traffic-control authority managing busy traffic in the city of Kolkata, India. Our data set consists of four video sequences, each of duration 15 minutes, captured from far-field traffic intersections recorded with fixed CCD cameras. The video sequences were captured under daylight conditions at four different traffic intersections in Kolkata, viz. Chandni-Chowk, Rajabazar, Kankurghachi, and Shyambazar. The video sequences were obtained in the H.264 encoding format. There are a myriad of activities and interactions in the video data involving both pedestrian and vehicle movements. Please note that the current application uses vehicle detection for intersection analysis at congested places. Its primary objective is to identify motor-driven vehicles from pedestrians and manually-driven vehicles. It is unlike [155], [161], or any other related work, which is based on the premise that any foreground movement is associated with a vehicle traveling on a highway. A comparison of these methods would be unfair and hence not addressed.

6.3.2 Training Data and Augmentation

As mentioned earlier, each of the four sequences was divided into two parts; the duration if the first part is five minutes, which is used for training; the remaining part of the sequence is used for testing, i.e., traffic volume measurements at the intersections for the



(a) Chandni Chowk Crossing, Kolkata (4-way intersection) (b) Raja-bazar Crossing, Kolkata (4-way intersection)

(c) Kankurgachi Crossing, Kolkata (3-way intersection) (d) Shyambazar Crossing, Kolkata (5-way intersection)

Fig. 6.5 Screenshots of the segmented vehicles at traffic intersections. Vehicles are numbered in the order in which they were first detected in the scene.

next ten minutes. The training data, which consists of 2033 and 2177 images respectively from the vehicle and the non-vehicle category, was obtained manually by cropping minimally-bounded images for individual objects. The cropped images were resized to 64×64×3 for training. This is followed by the process of data augmentation, which is conventionally used with deep layered architectures in order to reduce overfitting. Data augmentation was performed by replicating each input image with horizontal (lateral) inversion. Please note that this process increases the number of training samples by a factor of two.

Table 6.2 Confusion Matrix

Predicted Class			
1(non-vehicle)	979	24	98.5%
2 (vehicle)	34	969	96.6%
	96.6%	97.6%	
	1 (non-vehicle)	2 (vehicle)	

Target Class

6.3.3 Experimental Results

The evaluation is done in two stages. First, the trained model is evaluated against a set of 2004 testing images obtained from the testing part of the sequences. The training module was implemented using Matlab® based on the MatConvNet [173] toolbox. The weights of the networks are initialized from a zero mean uniform distribution in the interval $[-\sqrt{6/(n_j + n_{j+1})}, \sqrt{6/(n_j + n_{j+1})}]$, where n_j and n_{j+1} denote the number of neurons in the jth and the (j+1)th layer. The process is known as normalized initialization as suggested in [174]. The stochastic gradient descent (SGD) method, is used as an optimization method for minimization of loss function. It may be noted, that SGD will work for all functions that have a gradient or first derivative. Usually, the system uses SGD for minimizing the error or loss function and updates the weight parameters based on (6.8). Changing momentum and weight decay parameters affect the time required for training. Our CNN architecture was trained using SGD with 50 examples per batch, μ = 0.9, and λ = 0.0005.

Table 6.3 Traffic volume estimates for Kankurgachi intersection, Kolkata

(i)	Ground truth volume (R_i)	Estimated Result (r_i)
AA	0	0
AB	0	0
AC	162	108
BA	123	89
BB	0	0
BC	53	47
CA	182	163
CB	0	0
CC	0	0

Table 6.4 Traffic volume estimates for Chandni-chowk 4-point intersection, Kolkata

(i)	Ground truth volume (R_i)	Estimated Result (r_i)
AA	0	0
AB	23	20
AC	70	63
AD	32	29
BA	25	0
BB	0	0
BC	28	19
BD	216	209
CA	67	62
CB	13	10
CC	0	0
CD	0	0
DA	2	0
DB	177	152
DC	4	2
DD	0	0

In the current CNN architecture, the ReLU activation function has been used for all the layers. We trained our network for 40 epochs with batch sizes of 50. The training took roughly eight hours in total on a 3.50GHz CPU (using parallelization with six cores). Images from the vehicle and the non-vehicle categories were used for testing. The classification result for the test images is illustrated in Table 6.2 as a confusion matrix. The confusion matrix is a popular method to assess the performance of a classification algorithm. As shown in Table 6.2, each row represents the numbers of members of a predicted class (output class), while each column represents their counterparts from the actual class (target class). The trained model, although could detect images of the "vehicle" and the "non-vehicle" class with high accuracy, some images were wrongly classified as belonging to the other category.

Table 6.5 Traffic volume estimates for Rajabazar 4-point intersection, Kolkata

(i)	Ground truth volume (R_i)	Estimated Result (r_i)
AA	0	0
AB	1	0
AC	58	50
AD	6	4
BA	25	21
BB	0	0
BC	3	2
BD	111	97
CA	90	83
CB	0	0
CC	0	0
CD	6	4
DA	0	0
DB	94	87
DC	5	2
DD	1	0

In the secondary stage of our evaluation, the traffic volume estimates at each road segment of an intersection are evaluated. We used the testing part (duration 10 minutes) of the sequences corresponding to each of the four traffic-intersections. The ground-truth data detailing the traffic volume statistics were noted manually. The ground truth data for Kankurgachi, Chandni-Chowk, Rajabazar, and Shyambazar has been compared with the estimated traffic volumes obtained by the proposed method in Table-6.3, Table-6.4, Table-6.5 and Table-6.6 respectively.

Fig. 6.5 shows the screenshots of segmented vehicles at each of the four intersections. In each figure, the location of tracked vehicles is detected and indicated as minimum bounding rectangles. The labels against the bounding boxes indicate the order in which the vehicles were detected. In order to quantify the accuracy of the proposed

method, the observed direction (i) of the vehicles, the actual traffic volume (R_i) or ground-truth, and the estimated volume along that direction (r_i) are noted. For a given intersection, the accuracy of traffic volume estimates was measured as

$$\text{Accuracy} = 1 - \frac{\sum_i |R_i - r_i|}{\sum_i R_i}. \tag{6.11}$$

The accuracy of the proposed method, based on the obtained results for Kankurgachi, Chandni-Chowk, Rajabazar, and Shyambazar is 0.78, 0.86, 0.88, and 0.87 respectively. Thus, the effectiveness of the proposed architecture for vision-based intersection analysis is promising and manifested.

Table 6.6 Traffic volume estimates for Shyambazar 5-point intersection, Kolkata

(i)	Ground truth volume (R_i)	Estimated Result (r_i)
AA	0	0
AB	1	0
AC	16	15
AD	8	5
AE	0	0
BA	0	0
BB	0	0
BC	8	6
BE	0	0
CA	0	0
CB	0	0
CC	0	0
CD	0	0
CE	0	0
DA	43	39
DB	87	80
DC	0	0
DE	0	0
EA	0	0
EB	3	1
EC	16	12
ED	0	0
DD	0	0

In this chapter, a deep layered architecture is proposed for recognition of vehicles from other objects in a traffic intersection video. The traffic volume estimates are used later to determine the traffic flow statistics from a given intersection video. Even with a limited set of training images the accuracy of vehicle detection of our method was 97.6% while that for traffic volume estimates at the intersections being 78%, in the worst case (Kankurgachi). It may be noted that the model is specifically designed to obtain real-time statistics of toll vehicles at road intersections from video. The current application, however, in case of highly overlapped areas where the view of one of the vehicles is obstructed by another vehicle. Another issue is to improve the detection results at night or

under very low-illumination conditions. The overall performance, in such cases, could be improved by using robust multi-layered segmentation technique, which could be interesting areas for future work.

Although we have shown the application using H.264 High profile, one may expect similar results from the same real-life data. This is because of the rest of the methods for foreground object extraction in H.264 and HEVC encoded video streams have been tested on the same set of benchmarking datasets. It was shown that the performance of the three methods are quite similar. This indicates the proposed techniques can work equally well even with the real-life dataset.

****Major portion of this Chapter taken from the following publication**

B. Dey and M.K. Kundu, "Turning Video into Traffic Data - An Application to Urban Intersection Analysis Using Convolution Neural Network," IET Image Processing, (communicated).

Chapter 7

Conclusion and Future Scope

7.1 Conclusion

This chapter summarizes the major contributions of findings and the research activities reported in the present thesis. It also provides certain pointers for future research in various fields of the video analytics using compressed information. It may here be noted, that the first stage of analysis originates from detecting motion activities or changes in the scene. Indeed, for many applications, the very fact that something is moving makes it of 'interest' while anything else can be ignored. In such cases, it is common for regions of interest or foreground to be differentiated from the remaining part of the scene or background. The thesis contributes towards extraction of block-based features from compressed video, which are used for foreground extraction. The last contributory chapter is an application of foreground extraction to traffic sequence analysis from a real-life video sequence captured from the different road-intersections in the city of Kolkata.

The major novel contributions of the present thesis can be summarized as follows:

- H.264 Baseline profile is the most widely used codec currently in use today. Although it is not the most bandwidth efficient codec, it is still widely used owing to its low-latency and low-computational requirement. A scheme for extraction macroblock features in H.264 Baseline has been proposed. This is used to extract foreground information. The proposed method has been tested on challenging benchmark sequences. It is found that although YCbCr is a widely-used color format for compressed video, but it lacks the capability of representing true information if there is a shadow or drastic intensity variation in the frames. To overcome this problem, we have tested CIELUV color space for the compressed video and it was found that the change in color format can reduce the above mentioned to a good extent, thereby improving the overall accuracy.
- It is found that segmentation performance on sequences deteriorates drastically with encoding bitrate. This is because the quantization parameters vary widely at lower bitrates. Therefore, enhanced block based features are proposed using the effective quantization step sizes for individual coefficients. Experiments have been demonstrated for low-bitrate videos encoded in H.264 Main/High profiles. Following the same argument as mentioned for H.264 Baseline, here also conversion of the color format from YCbCr to CIELUV achieved better result.
- HEVC is the successor standard to H.264, and has generated huge optimism given the struggle with shortage of bandwidth, spectrum, storage and imminent need to

take growing high-definition content for multi-platform delivery. New CTU block features are proposed which are used to predict CTU blocks containing potential foreground activity or motion. To the best of our knowledge, this is the first major attempt at moving object segmentation from HEVC encoded video. Following the same argument as mentioned previously the color format conversion from YCbCr to CIELUV is also found to deliver better performance as expected.

- An application for traffic video is developed. Usually, in a congested traffic intersection moving objects of all categories are encountered. To keep a statistic of the vehicles, a method is proposed that uses the extracted foreground patches to differentiate potential motor-vehicles from other moving objects (pedestrians, manually driven vehicles) in a traffic scene. The extracted patches are trained using a CNN architecture and later classified tracked. However, there are room for improvements, in cases of inaccurate segmentation in occlusion, illumination changes, trajectory clustering etc.

7.2 Future work

Video analytics is a rapidly expanding field and new challenges are arising every day. The work reported in this thesis is an attempt to solve the fundamental problem of reducing processing speed and computational cost of video analytics for real-time applications. Although the solutions proposed in this thesis have been found to be useful and efficient, they need further investigations to be widely applicable in real-life application domains. These may include theoretical analysis of performance, development of quantitative indices for evaluation, and study of sensitivity of the features to abrupt background motion, illumination change etc. Some of the possible future extensions of the work presented in thesis are pointed as follows:

1. The segmentation performance is comparable to the state-of-the-art in comparison to contemporary pixel-based approaches. However, additional features could be incorporated which might help increasing the segmentation accuracy.
2. Although the methods present in the thesis can robustly handle gradual illumination changes and dynamic background, it is unable to cope with sudden illumination changes, fast-moving cast shadows, and drastically changing background information. A more sophisticated model of the background may be developed of the proposed methods reported in this thesis. For example, the use of normalized

color space and an explicit model of the “moved objects” could further improve our method.

3. The proposed methods can be suitably modified to handle other real-life problem like occlusion by other moving objects, very low-intensity of the moving object due to dense shadow, effect of glittering and flash-light etc.
4. Although the methods were applied on traffic surveillance data obtained from a single-view camera, this could be extended to multi-view cameras to obtain precise information on tracked objects/vehicles.
5. In medical image domain, this method could be suitably modified to analysis of movement of different parts of human body like echocardiograph of the heart, movement of the fetus in ultrasonography images, etc. The contributions, could be extended with suitable modelling to the case of background perturbations caused by sensor noise, fog, rain, dust, snow etc. and taken as future work.

References

[1] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun, "3D Traffic Scene Understanding from Movable Platforms," *IEEE Transactions on Pattern Analysis Machine Intelligence*, vol. 36, no. 5, pp. 1012-900, 2014.

[2] T. Zhang, S. Liu, C. Xu, and H. Lu, "Mining semantic context information for intelligent video surveillance of traffic scenes," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 149–160, 2013.

[3] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik, "A real-time computer vision system for vehicle tracking and traffic surveillance," *Transportation Research Part C: Emerging Technologies*, vol. 6, no. 4, pp. 271-288, 1998.

[4] A. Smeulders, D.M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual Tracking: an Experimental Survey," *IEEE Transactions on Pattern Analysis Machine Intelligence*, vol. 36, no. 7, 2014.

[5] D. Ramanan, D. A. Forsyth, and A. Zisserman, "Tracking people by learning their appearance," *IEEE Transactions on Pattern Analysis Machine Intelligence*, vol. 29, no. 1, pp. 65–81, Jan. 2007.

[6] D. Ramanan, D. A. Forsyth, and K. Barnard, "Building models of animals from video," *IEEE Transactions on Pattern Analysis Machine Intelligence*, vol. 28, no. 8, pp. 1319–1334, 2006.

[7] W. Hu, T. Tan, L. Wang and S. Maybank, "A survey of visual surveillance of object motion and behaviors," IEEE Trans. Syst., Man, Cybern. C, vol. 34, no. 3, pp. 334-352, Aug. 2004.

[8] R. Kasturi, D. Goldgof, P. Soundararajan, V. Manohar, J. Garofolo, R. Bowers, M. Boonstra, V. Korzhova, and J. Zhang, "Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol," *IEEE Transactions on Pattern Analysis Machine Intelligence*, vol. 31, no. 2, pp. 319–336, Feb. 2009.

[9] Y. Fu, Y. Guo, Y. Zhu, F. Liu, C. Song, and Z. Zhou, "Multi-View Video Summarization," *IEEE Transactions on Multimedia*, vol. 12, no. 7, pp. 717-729, 2010.

[10] S. Ali and M. Shah, "Human action recognition in videos using kinematic features and multiple instance learning," *IEEE Transactions on Pattern Analysis Machine Intelligence*, vol. 32, no. 2, pp. 288–303, 2010.

[11] F. Lv and R. Nevatia, "Single view human action recognition using key pose matching and Viterbi path searching," *Proceedings on IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[12] J. Alon, V. Athitsos, Q. Yuan, and S. Sclaroff, "A unified framework for gesture recognition

and spatiotemporal gesture segmentation," *IEEE Transactions on Pattern Analysis Machine Intelligence*, vol. 31, no. 9, pp. 1685–1699, Sep. 2009.

[13] A.D. Wilson and A.F. Bobick, "Parametric Hidden Markov Models for Gesture Recognition," *IEEE Transactions on Pattern Analysis Machine Intelligence*, vol. 21, no. 9, pp. 884-900, 1999.

[14] M. Molinier, T. Häme, and H.I. Ahola, "3D-Connected Components Analysis for Traffic Monitoring in Image Sequences Acquired from a Helicopter," *Image Analysis*, pp. 141-150, 2005.

[15] Y. Chung, J. Wang, and S. Cheng, "Progressive background image generation," in *Proceedings of the of 15th IPPR Conf. on Computer Vision, Graphics and Image Processing*, CVGIP 2002, pp. 858–865, 2002.

[16] R. M. Colque, G. Cámara-Chávez, "Progressive background image generation of surveillance traffic videos based on a temporal histogram ruled by a reward/penalty function," In *Proceedings of the 24th SIBGRAPI Conference on Graphics, Patterns and Images*, 2011, pp. 297-304.

[17] B. Qin, J. Wang, J. Gao, T. Pang, and F. Su, "A traffic video background extraction algorithm based on image content sensitivity," in *Proceedings of the 1st International Conference on Swarm Intelligence*, 2010, pp. 603–610.

[18] F. El Baf, T. Bouwmans, and B. Vachon, "A fuzzy approach for background subtraction, in: International Conference on Image Processing," *Proceedings of the 17th International Conference on Image Processing*, 2008, pp. 2648–2651.

[19] M. Sigari, N. Mozayani, and H. Pourreza, "Fuzzy running average and fuzzy background subtraction: Concepts and application," *International Journal of Computer Science and Network Security*, vol. 8, no. 2, pp. 138–143, 2008.

[20] M. Sigari, "Fuzzy background modeling/subtraction and its application in vehicle detection," in *Proceedings of the World Congress on Engineering and Computer Science*, 2008.

[21] L. Maddalena and A. Petrosino, "A fuzzy spatial coherence-based approach to background/foreground separation for moving object detection," *Neural Computing and Applications*, vol. 19, no. 2, pp. 179–186, 2010.

[22] L. Maddalena and A. Petrosino, "Self-organizing and fuzzy modelling for parked vehicles detection," in *Proceedings of the Advanced Concepts for Intelligent Vision Systems*, 2009, pp. 422–433.

[23] Y. Zhang, Z. Liang, Z. Hou, H. Wang, and M. Tan, "An adaptive mixture Gaussian background model with online background reconstruction and adjustable foreground mergence time for motion segmentation," in *Proceedings of the IEEE International Conference Industrial Technology*, 2005, pp. 23–27.

[24] H. Wang and D. Suter, “A re-evaluation of mixture-of-Gaussian background modeling,” in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2005, pp. 1017–1020.

[25] F. Porikli, “Human body tracking by adaptive background models and mean-shift analysis,” in *Proceedings of the IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, March 2003.

[26] D. R. Magee, “Tracking multiple vehicles using foreground, background and motion models,” *Image and Vision Computing*, vol. 22, no. 2, pp. 143–155, 2004.

[27] C. Stauffer and E. Grimson, “Adaptive background mixture models for real-time tracking,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1999, pp. 246–252.

[28] X. Fang, W. Xiong, B. Hu, and L. Wang, “A Moving Object Detection Algorithm Based on Color Information,” in *Proceedings of the International Symposium on Instrumentation Science and Technology*, vol. 48, pp. 384-387, 2006.

[29] H. Bhaskar, L. Mihaylova, and A. Achim, “Video foreground detection based on symmetric alpha-stable mixture models,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 8, 2010.

[30] L. Li and W. Huang, “Statistical modeling of complex background for foreground object detection,” *IEEE Transactions on Image Processing*, vol. 13, no. 11, 2004, pp. 1459–1472.

[31] H. Zhang and D. Xu, “Fusing color and texture features for background model,” in *Third International Conference on Fuzzy Systems and Knowledge Discovery*, Lecture Notes in Computer Science, vol. 4223, 2006, pp. 887–893.

[32] F. El Baf, T. Bouwmans, and B. Vachon, “Foreground detection using the Choquet integral,” in *Ninth International Workshop on Image Analysis for Multimedia Interactive Services*, 2008, pp. 187–190.

[33] K. Toyama, J. Krumm, B. Brumiit, and B. Meyers, “Wallflower: principles and practice of background maintenance,” in *Proceedings of the IEEE International Conference on Computer Vision*, 1999, pp. 255–261.

[34] B. Lee and M. Hedley, “Background estimation for video surveillance,” in *Image and Vision Computing New Zealand*, 2002, pp. 315–320.

[35] N. McFarlane and C. Schofield, “Segmentation and tracking of piglets in images,” *Machine Vision and Applications*, vol. 8, no. 3, pp. 187–193.

[36] J. Zheng, Y. Wang, N. L. Nihan, M. E. Hallenbeck, “Extracting roadway background image: Mode-based approach,” *Transportation Research Record*, vol. 1944, pp. 82-88, 2006.

[37] C. Wren and A. Azarbayejani, “Pfinder: real-time tracking of the human body,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780–785, 1997.

[38] H. Kim, R. Sakamoto, I. Kitahara, T. Toriyama, and K. Kogure, “Robust foreground extraction technique using Gaussian family model and multiple thresholds,” in *Proceedings of the Asian Conference on Computer Vision*, 2007, pp. 758–768.

[39] F. Porikli and O. Tuzel, “Bayesian background modeling for foreground detection,” in *Proceedings of the ACM International Workshop on Video Surveillance and Sensor Networks*, 2005, pp. 55–58.

[40] M. Alvar, A. Rodriguez-Calvo, A. Sanchez-Miralles, and A. Arranz, “Mixture of merged Gaussian algorithm using RTDENN,” *Machine Vision and Applications*, vol. 25, no. 5, pp 1133–1144, 2014.

[41] M. Allili, N. Bouguila, and D. Ziou, A robust video foreground segmentation by using generalized Gaussian mixture modeling, in *Proceedings of the Fourth Canadian Conference on Computer and Robot Vision*, 2007, pp. 503–509.

[42] A. Elgammal, D. Harwood, and L. Davis, “Non-parametric model for background subtraction,” in *Proceedings of the European Conference on Computer Vision*, vol. 2, 2000, pp. 751–767.

[43] Y. Sheikh and M. Shah, “Bayesian object detection in dynamic scenes,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 74–79.

[44] Y. Sheikh and M. Shah, “Bayesian modeling of dynamic scenes for object detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 11, pp. 1778–1792, 2005.

[45] B. Han, D. Comaniciu, and L. Davis, “Sequential kernel density approximation through mode propagation: applications to background modeling,” in *Proceedings of the Asian Conference on Computer Vision*, 2004, pp. 1–7.

[46] H. Lin, T. Liu, and J. Chuang, “A probabilistic SVM approach for background scene initialization,” in *Proceedings of the International Conference on Image Processing*, 2002, pp. 893–896.

[47] J. Wang, G. Bebis, and R. Miller, “Robust video-based surveillance by integrating target detection with tracking,” in *IEEE International Conference on Computer Vision and Pattern Recognition Workshop*, 2006, pp. 137–144.

[48] A. Tavakkoli, M. Nicolescu, and G. Bebis, “Novelty detection approach for foreground region detection in videos with quasi-stationary backgrounds,” in *Proceedings of the International Symposium on Visual Computing*, 2006, pp. 40–49.

[49] H. Lin, T. Liu, J. Chuang, “Learning a scene background model via classification,” *IEEE*

Transactions on Signal Processing, vol. 57, no. 5, pp. 1641–1654, 2009.

[50] P. Blauensteiner and M. Kampel. Visual Surveillance of an Airport's Apron - An Overview of the AVITRACK Project. In *Workshop of the Austrian Association for Pattern Recognition*, vol. 179, pp. 213–220, 2004.

[51] H. Tan, B. Cheng, J. Feng, G. Feng, Y. Zhang, Tensor recovery via multi-linear augmented Lagrange multiplier method, in *Proceedings of the International Conference on Image and Graphics*, pp. 141–146, 2011.

[52] A. Tavakkoli, M. Nicolescu, G. Bebis, M. Nicolescu, "A support vector data description approach for background modeling in videos with quasi-stationary backgrounds," *International Journal of Artificial Intelligence Tools*, vol. 17, no. 4, 2008, pp. 635–658.

[53] J. Ma and J. Theiler, "Accurate on-line support vector regression," *Neural Computation*, vol. 15, no. 11, pp. 2683-2703, 2003.

[54] A. Tavakkoli, A. Ambardekar, M. Nicolescu, and S. Louis, "A genetic approach to training support vector data descriptors for background modeling in video data," In *Advances in Visual Computing*, pp. 318–327, 2007.

[55] A. Tavakkoli, M. Nicolescu, M. Nicolescu, and G. Bebis, Incremental SVDD training: Improving efficiency of background modeling in videos, in *Proceedings of the International Conference on Signal and Image Processing*, 2008.

[56] A. Tavakkoli, M. Nicolescu, M. Nicolescu, and G. Bebis, "Efficient background modeling through incremental support vector data description," in *Proceedings of the 19th IEEE International Conference on Pattern Recognition*, 2008, pp. 3958-3961.

[57] N. M. Oliver, B. Rosario, and A. P. Pentland, "A Bayesian computer vision system for modeling human interactions," *IEEE Transactions Pattern Analysis and Machine Intelligence*, vol. 22, pp. 831–843, Aug. 2000.

[58] Z. Xu, I. Gu, and P. Shi, "Recursive error-compensated dynamic eigen-background learning and adaptive background subtraction in video," *Optical Engineering*, vol. 47, no. 5, 2008.

[59] S. Kawabata, S. Hiura, and K. Sato, "*Real-time detection of anomalous objects in dynamic scene*," in Proceedings of the International Conference on Pattern Recognition, 2006, pp. 1171–1174.

[60] C. Quivy and I. Kumazawa, "Background images generation based on the nelder-mead simplex algorithm using the eigen-background model," in *International Conference on Image Analysis and Recognition*, 2011, pp. 21–29.

[61] J. D. Rymel, J.-P. Renno, D. Greenhill, J. Orwell, and G. A. Jones, "Adaptive eigen-backgrounds for object detection," in *Proceedings of the IEEE International Conference on Image Processing*, 2004, pp. 1847–1850.

[62] Y. Li, “On incremental and robust subspace learning,” *Pattern Recognition*, vol. 37, no. 7, pp. 1509–1518, 2004.

[63] D. Skocaj and A. Leonardis, “Weighted and robust incremental method for subspace learning,” in *Proceedings of the 9th IEEE International Conference on Computer Vision*, 2003, pp. 1494–1501.

[64] B. Han and R. Jain, “Real-time subspace-based background modeling using multi-channel data,” in *International Symposium on Visual Computing*, 2007, pp. 162–172.

[65] Y. Dong and G. DeSouza, “Adaptive learning of multi-subspace for foreground detection under illumination changes,” *Computer Vision and Image Understanding*, vol. 115, no. 1, 2011, pp. 31–49.

[66] M. Yamazaki, G. Xu, and Y. Chen, Detection of moving objects by independent component analysis, in *Proceedings of the Asian Conference on Computer Vision*, 2006, pp. 467–478.

[67] D. Tsai and C. Lai, “Independent component analysis-based background subtraction for indoor surveillance,” *IEEE Transactions on Image Processing*, vol. 8, no. 1, pp. 158–167, 2009.

[68] S. Bucak, B. Gunsel, and O. Gursoy, “Incremental non-negative matrix factorization for dynamic background modeling,” in *Proceedings of the 15th IEEE Conference on Signal Processing and Communications Applications*, 2007, pp. 1–4.

[69] X. Li, W. Hu, Z. Zhang, and X. Zhang, “Robust foreground segmentation based on two effective background models,” in *Proceedings of the 1st ACM international conference on Multimedia information retrieval*, 2008, pp. 223–228.

[70] M. G. Krishna, V. M. Aradhya, M. Ravishankar, and D. R. Babu, “LoPP: Locality pre-serving projections for moving object detection,” in *Proceedings of International Conference on Computer, Communication, Control and Information Technology*, 2012, pp. 624-628.

[71] D. E. Butler, V. B. Jr., and S. Sridharan, “Real-time adaptive foreground-background segmentation,” *EURASIP Journal on Applied Signal Processing*, vol. 14, pp. 2292–2304, 2005.

[72] K. Kim, T. Chalidabhongse, D. Harwood, and L. Davis, “Background modeling and subtraction by codebook construction,” in *Proceedings of the IEEE International Conference on Image Processing*, vol. 5, pp. 3061–3064, 2004.

[73] M. Xiao, C. Han, and X. Kang, “A background reconstruction for dynamic scenes,” in *Proceedings of the International Conference on Information Fusion*, 2006, pp. 1–6.

[74] D. Xiuman, S, Guoxia, and Y. Tao, “Moving target detection based on Genetic K-means Algorithm,” in *Proceedings of the International Conference on Communication Technology*, 2011, pp. 819–822.

[75] K. Kim, T. Chalidabhongse, D. Harwood, and L. Davis, “Real time foreground background segmentation using codebook model,” *Real time Imaging*, vol. 11, no. 3, 2005, pp. 167–256.

[76] M. Sigari and M. Fathy, “Real-time background modeling/subtraction using two-layer codebook model,” in *Proceedings of the International Multiconference on Engineering and Computer Science*, 2008, pp. 717-720.

[77] A. Doshi and M. Trivedi, “Hybrid cone-cylinder codebook model for foreground detection with shadow and highlight suppression,” in *Proceedings IEEE International Conference on Advanced Video and Signal based Surveillance*, 2006.

[78] H. Hu, L. Xu, and H. Zhao, “A spherical codebook in YUV color space for moving object detection,” *Sensor Letters*, vol. 10, no. 1, pp. 177–189, 2012.

[79] X. Deng, J. Bu, Z. Yang, C. Chen, Y. Liu, “A block-based background model for video surveillance,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2008, pp. 1013–1016.

[80] J. Xu, N. Jiang, and S. Goto, “Block-based codebook model with oriented-gradient feature for real-time foreground detection,” in *Proceedings of the 13th IEEE International Workshop on Multimedia Signal Processing*, 2011.

[81] J. Guo and C. Hsu, “Hierarchical method for foreground detection using codebook model,” in *Proceedings of the IEEE International Conference on Image Processing*, 2010, pp. 3441-3444.

[82] A. Zaharescu, M. Jamieson, Multi-scale multi-feature codebook-based background subtraction, in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2011, pp. 1753–1760

[83] M. Xiao, C. Han, X. Kang, “A background reconstruction for dynamic scenes,” in *Proceedings of the IEEE International Conference on Information Fusion*, 2006, pp. 1–6.

[84] M. Xiao and L. Zhang, “A background reconstruction algorithm based on modified basic sequential clustering,” in *Proceedings of the International Colloquium on Computing, Communication, Control, and Management*, vol. 1, 2008, pp. 47–51.

[85] M. Xiao and L. Zhang, “A background reconstruction algorithm based on two-threshold sequential clustering,” in *Proceedings of the International Colloquium on Computing, Communication, Control, and Management*, vol. 1, 2008, pp. 389–393.

[86] M. Benalia and S. Ait-Aoudia, “An improved basic sequential clustering algorithm for background construction and motion detection,” in *Proceedings of the 9th International Conference on Image Analysis and Recognition*, 2012, pp. 216-223.

[87] D. Culibrk, O. Marques, D. Socek, H. Kalva, and B. Furht, “A neural network approach to Bayesian background modeling for video object segmentation,” in *Proceedings of the*

International Conference on Computer Vision Theory and Applications, 2006, pp. 474–479.

[88] D. Culibrk, O. Marques, D. Socek, H. Kalva, and B. Furht, "Neural network approach to background modeling for video object segmentation," *IEEE Transactions on Neural Network*, vol. 18, no. 6, pp. 1614–1627, 2007.

[89] R. Luque, D. Lopez-Rodriguez, E. Merida-Casermeiro, and E. Palomo, "Video object segmentation with multivalued neural networks," in *Proceedings of the IEEE International Conference on Hybrid Intelligent Systems*, 2008, pp. 613–618.

[90] R. Luque, E. Dominguez, E. Palomo, and J. Munoz, "A neural network approach for video object segmentation in traffic surveillance," in *Proceedings of the International Conference on Image Analysis and Recognition*, 2008, pp. 151–158.

[91] R. Luque, D. Lopez-Rodriguez, E. Dominguez, and E. Palomo, "A dipolar competitive neural network for video segmentation," in *Proceedings of the Ibero-American Conference on Artificial Intelligence*, 2008, pp. 103–112.

[92] L. Maddalena and A. Petrosino, "A self-organizing approach to background subtraction for visual surveillance applications," *IEEE Transactions on Image Processing*, vol. 17, no. 7, 2008, pp. 1729–1736.

[93] L. Maddalena and A. Petrosino, "The SOBS algorithm: What are the limits?" in *Proceedings IEEE Computer Society Conference Computer Visual Pattern Recognition Workshops*, 2012, pp. 21–26.

[94] E. Palomo, E. Dominguez, R. Luque, and J. Munoz, "Image hierarchical segmentation based on a GHSOM," in *Proceedings of the International Conference on Neural Information Processing*, 2009, pp. 743–750.

[95] K. Karmann and A. Von Brand, "Moving object recognition using an adaptive background memory," *Time-Varying Image Processing and Moving Object Recognition*, ed. 2, Elsevier, 1990, pp. 289-307.

[96] T. Chang, T. Ghandi, and M. Trivedi, "Vision modules for a multi-sensory bridge monitoring approach," in *Proceedings of the 7th International IEEE Conference on Intelligent Transportation Systems*, 2004, pp. 971–976.

[97] M. Boninsegna and A. Bozzoli, "A tunable algorithm to update a reference image," *Signal Processing: Image Communication*, vol. 16, no. 4, pp. 1353–1365, 2000

[98] S. Messelodi, C. M. Modena, N. Segata, and M. Zanin, "A Kalman filter-based background updating algorithm robust to sharp illumination changes," *in Proceedings 13th International Conference on Image Analysis and Processing*, vol. 3617, Lect. Notes Comput. Sci., F. Roli and S. Vitulano, Eds., 2005, pp. 163–170.

[99] J. Zhong and S. Sclaroff, "Segmenting foreground objects from a dynamic textured background via a robust Kalman filter," in *Proceedings of the International Conference on*

Computer Vision, 2003, pp. 44–50.

[100] H. Yoshimura, Y. Iwai, and M. Yachida, "Object detection with adaptive background model and margined sign cross correlation," in *Proceedings of the International Conference on Pattern Recognition*, 2006, pp. 19–23.

[101] A. Yamamoto and Y. Iwai, "Real-time object detection with adaptive background model and margined sign correlation," in *Proceedings 9th Asian Conference on Computer Vision*, 2009, pp. 65-74.

[102] D. Gao, J. Zhou, and L. Xin, "A novel algorithm of adaptive background estimation," in *Proceedings of the International Conference on Image Processing*, 2001, pp. 395–398.

[103] T. Wang, G. Chen, and H. Zhou, "A novel background modelling approach for accurate and real-time motion segmentation," in *Proceedings of the International Conference on Signal Processing*, 2006.

[104] J. Ding, M. Li, K. Huang, and T. Tan, "Modeling complex scenes for accurate moving objects segmentation," in *Proceedings 10th Asian Conference on Computer Vision*, 2010, pp. 592–604.

[105] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[106] G.J. Sullivan, J.R. Ohm, W.J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, Dec 2012.

[107] B. Dey and M. K. Kundu, "Robust background subtraction for network surveillance in H. 264 streaming video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 10, pp. 1695-1703, 2013.

[108] B. Dey and M. K. Kundu, "Enhanced Macroblock Features for Dynamic Background Modeling in H.264/AVC Video Encoded at Low-Bitrate," *IEEE Transactions on Circuits and Systems for Video Technology*, (Accepted).

[109] B. Dey and M.K. Kundu, "Efficient Foreground Extraction from HEVC Compressed Video for Application to Real-Time Analysis of Surveillance 'Big' Data," *IEEE Transactions on Image Processing*, vol. 24, no. 11, pp. 3574-3585, Nov. 2015.

[110] B. Dey and M.K. Kundu, "Turning Video into Traffic Data - An Application to Urban Intersection Analysis Using Convolution Neural Network," *IET Image Processing*, (communicated).

[111] V. Thilak and C. D. Creusere, "Tracking of extended size targets in H.264 compressed video using the probabilistic data association filter," in *Proceedings of EUSIPCO*, 2004, pp. 281–284.

[112] W. Zeng, J. Du, W. Gao, and Q. M. Huang, "Robust moving object segmentation on H.264/AVC compressed video using the block-based MRF model," *Real-Time Imaging*, vol. 11, no. 4, pp. 290–299, Aug. 2005.

[113] Z. Liu, Z. Zhang, and L. Shen, "Moving object segmentation in the H.264 compressed domain," *Optical Engineering*, vol. 46, no. 1, p. 017003, Jan. 2007.

[114] C. Solana-Cipres, G. Fernandez-Escribano, L. Rodriguez-Benitez, J. Moreno-Garcia, and L. Jimenez-Linares, "Real-time moving object segmentation in H.264 compressed domain based on approximate reasoning," *International Journal Approximate Reasoning*, vol. 51, pp. 99–114, Sep. 2009.

[115] W. Fei and S. Zhu, "Mean shift clustering-based moving object segmentation in the H.264 compressed domain," *IET Image Processing*, vol. 4, no. 1, pp. 11–18, Feb. 2010.

[116] W. You, M. S. H. Sabirin, and M. Kim, "Moving object tracking in H.264/AVC bitstream," in *Proceedings International Workshop on Multimedia Content Analysis Mining*, Lecture Notes in Computer Science, vol. 4577. 2007, pp. 483–492.

[117] C. Poppe, S. D. Bruyne, T. Paridaens, P. Lambert, and R. V. D. Walle, "Moving Object Detection in the H.264/AVC compressed domain for video surveillance applications," *Journal of Visual Communication and Image Representation*, vol. 20, pp. 428–437, May 2009.

[118] C. Chen, J. Cai, W. Lin, and G. Shi, "Incremental low-rank and sparse decomposition for compressing videos captured by fixed cameras," *Journal of Visual Communication and Image Representation*, vol. 26, pp. 338–348, Jan. 2015.

[119] L. Zhao, X. Zhang, Y. Tian, R. Wang, and T. Huang, "A background proportion adaptive Lagrange multiplier selection method for surveillance video on HEVC," in *Proceedings of the IEEE International Conference on Multimedia Expo* (ICME), Jul. 2013, pp. 1–6.

[120] X. Guo, S. Li, and X. Cao, "Motion matters: A novel framework for compressing surveillance videos," in *Proceedings of the ACM International Conference on Multimedia*, Oct. 2013, pp. 549–552.

[121] C. Chen, J. Cai, W. Lin, and G. Shi, "Surveillance video coding via low-rank and sparse decomposition," in *Proceedings of the 20th ACM International Conference on Multimedia*, Oct. 2012, pp. 713–716.

[122] X. Zhang, T. Huang, Y. Tian, and W. Gao, "Background-modeling-based adaptive prediction for surveillance video coding," *IEEE Transactions on Image Processing*, vol. 23, no. 2, pp. 769–784, Feb. 2014.

[123] X. Zhang, Y. Tian, T. Huang, S. Dong, and W. Gao, "Optimizing the hierarchical prediction and coding in HEVC for surveillance and conference videos with background modeling," *IEEE Transactions on Image Processing*, vol. 23, no. 10, pp. 4511–4526, Oct. 2014.

[124] J. Nightingale, Q. Wang, C. Grecos, and S. R. Goma, "Deriving video content type from HEVC bitstream semantics," *Proceedings of the SPIE*, vol. 9139, p. 913902, May 2014.

[125] S. R. Smoot and L.A. Rowe, "Study of DCT Coefficient Distributions," in *Proceedings of SPIE Symposium on Electronic Imaging*, 1996, pp. 403–411.

[126] E. Y. Lam and J. W. Goodman, "A mathematical analysis of the DCT coefficient distributions for images," *IEEE Transactions on Image Processing*, vol. 9, no. 10, pp. 1661–1666, Oct. 2000.

[127] W. Wu and B. Song, "DC Coefficient Distributions for P-Frames," in *ETRI Journal*, vol. 33, no. 5, pp. 814–817, Oct. 2011.

[128] G. J. Sullivan and S. Sun, "On dead-zone plus uniform threshold scalar quantization," in *Proceedings of the SPIE Visual Communication and Image Processing*, vol. 5960, no. 2. Jul. 2005, pp. 1041–1052.

[129] F. Bellard. (2002, Apr. 26). FFmpeg [Online]. Available: http://ffmpeg.org

[130] N. Goyette, P.-M. Jodoin, F. Porikli, J. Konrad, and P. Ishwar, "Changedetection.net: A new change detection benchmark dataset," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, Jun. 2012, pp. 1–8.

[131] O. Barnich and M. Van Droogenbroeck, "ViBe: A universal background subtraction algorithm for video sequences," *IEEE Transactions on Image Processing*, vol. 20, no. 6, pp. 1709–1724, Jun. 2011.

[132] P. KaewTraKulPong and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection," in *Proceedings of the 2nd European Workshop on Advanced Video-Based Surveillance Systems*, 2001, pp. 149–158.

[133] M. Hofmann, P. Tiefenbacher, and G. Rigoll, "Background segmentation with feedback: The pixel-based adaptive segmenter," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2012, pp. 38–43.

[134] C. Benedek and T. Szirányi, "Study on Color Space Selection for Detecting Cast Shadows in Video Surveillance," *International Journal of Imaging Systems Technology*, vol. 17, no. 3, pp. 190-201, Oct. 2007.

[135] A. Hallapuro and M. Karczewicz, "Low Complexity Transform and Quantization," documents JVT-B038 and JVT-B039, Joint Video Team (JVT), Jan. 2002.

[136] S. Gordon, D. Marpe, and T. Wiegand, "Simplified Use of 8x8 Transforms – Updated Proposal & Results," Doc. JVT-K028, Mar. 2004.

[137] G. E. Forsythe, M. A. Malcolm, and C. B. Moler, *Computer Methods for Mathematical Computations*, Prentice-Hall, 1976.

[138] *Advanced Video Coding for Generic Audiovisual Services*, document Rec. ITU-T H.264, ITU-T, Apr. 2013.

[139] M. Paul, W. Lin, C. T. Lau, and B.-S. Lee, "Video coding using the most common frame in scene," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2010, pp. 734–737.

[140] M. Paul, W. Lin, C.-T. Lau, and B.-S. Lee, "A Long-Term Reference Frame for Hierarchical B-Picture-Based Video Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 10, pp. 1729–1742, Oct. 2014.

[141] Y.-M. Chen, I. V. Bajic, and P. Saeedi, "Moving region segmentation from compressed video using global motion estimation and Markov random fields," *IEEE Transactions on Multimedia*, vol. 13, no. 3, pp. 421–431, Jun. 2011.

[142] A. Fuldseth, G. Bjøntegaard, M. Budagavi, and V. Sze, "CE10: Core Transform Design for HEVC," document JCTVC-G495, Nov. 2011.

[143] N. S. Jayant and P. Noll, *Digital Coding of Waveforms*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1984.

[144] A. N. Akansu and R. A. Haddad, "Factorization of the coefficient variance matrix in orthogonal transforms," *IEEE Transactions on Signal Processing*, vol. 39, no. 3, pp. 714–718, Mar. 1991.

[145] P. Chiranjeevi and S. Sengupta, "Neighborhood supported model level fuzzy aggregation for moving object segmentation," *IEEE Transactions on Image Processing*, vol. 23, no. 2, pp. 645–657, Feb. 2014.

[146] A. Vacavant, T. Chateau, A. Wilhelm, and L. Lequièvre, "A benchmark dataset for foreground/background extraction," in *Proceedings of the Asian Conference on Computer Vision Workshops*, Background Models Challenge, vol. 7728. Nov. 2012, pp. 291–300.

[147] T.S.F. Haines and T. Xiang, "Background subtraction with Dirichlet process mixture models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 4, pp. 670–683, Apr. 2014.

[148] V. Reddy, C. Sanderson, and B. C. Lovell, "Improved foreground detection via block-based classifier cascade with probabilistic decision integration," *IEEE Transactions on Circuits Systems Video Technology*, vol. 23, no. 1, pp. 83–93, Jan. 2013.

[149] P.G. Michalopoulos, "Vehicle Detection Video Through Image Processing: The Autoscope System," *IEEE Transactions on Vehicular Technology*, vol. 40, No. 1, 1991, pp. 21-29.

[150] S. Kamijo, Y. Matsushita, K. Ikeuchi, and M. Sakauchi, "Traffic Monitoring and Accident Detection at Intersection," *IEEE Transactions on Intelligence Transportation Systems*, vol. 1, no. 2, pp. 108-118, Jun. 2000.

[151] J. Wu, X. Zhang, and J. Zhou, "Vehicle detection in static road images with PCA and wavelet-based classifier," in *Proceedings of the IEEE Intelligent Transportation Systems Conference*, Oakland, CA, Aug. 25–29, 2001, pp. 740–744.

[152] Z. Sun, G. Bebis, and R. Miller, "On-road vehicle detection using Gabor filters and support vector machines," *IEEE International Conference on Digital Signal Processing*, Santorini, Greece, Jul. 2002.

[153] A. J. Lipton, H. Fujiyoshi, and R. S. Patil, "Moving target classification and tracking from real-time video," *Proceedings of the IEEE Workshop on Applied Computer Vision*, 1998, pp. 8–14.

[154] E. Ohn-Bar and M.M. Trivedi, "Learning to detect vehicles by clustering appearance patterns," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2511-2521, 2015.

[155] M Liang, X Huang, C.-H. Chen, X. Chen, and A. Tokuta, "Counting and Classification of Highway Vehicles by Regression Analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol.16, no. 5, pp. 2878-2888, Oct. 2015.

[156] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Proceedings of Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[157] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Proceedings of International Conference on Learning Representations*, 2015.

[158] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic Flow Prediction With Big Data: A Deep Learning Approach," *IEEE Transactions on Intelligence Transportation Systems*, vol. 16, no. 2, pp. 865-872, Apr. 2015.

[159] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," Proc. British Machine Vision Conf. (BMVC), 2011.

[160] N. Anjum and A. Cavallaro, "Multi-feature object trajectory clustering for video analysis," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 11, pp. 1555-1564, Nov. 2008.

[161] L.-W. Tsai, J.-W. Hsieh, and K.-C. Fan, "Vehicle Detection Using Normalized Color and Edge Map," *IEEE Transactions on Image Processing*, vol. 16, no. 3, pp. 850–864, Mar. 2007.

[162] Y. Wang, Y. Zou, H. Shi, and H. Zhao, "Video image vehicle detection system for signaled traffic intersection," *Proceedings of the International Conference Hybrid Intell. Syst.*, 2009, vol. 1, pp. 222–227.

[163] Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Wang, "Traffic Flow Prediction With Big Data: A

Deep Learning Approach," *IEEE Transportation on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865-873, Apr. 2015.

[164] S. Gupte, O. Masoud, R.F.K. Martin, and N.P. Papanikolopoulos, "Detection and Classification of Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, No. 1, 2002, pp. 37-47.

[165] A. Faro, D. Giordano, and C. Spampinato, "Adaptive Background Modeling Integrated with Luminosity Sensors and Occlusion Processing for Reliable Vehicle Detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1398-1412, Dec. 2011.

[166] J.M. Milla, S.L. Toral, and F. Barrero, "An Enhanced Background Estimation Algorithm for Vehicle Detection in Urban Traffic Scenes Manuel Vargas," *IEEE Transactions on Vehicular Technology*, vol. 59, No. 8, pp. 3964-3709, Oct. 2010.

[167] S.-C. Chen, M.-L. Shyu, S. Peeta, and C. Zhang, "Spatiotemporal vehicle tracking: the use of unsupervised learning-based segmentation and object tracking," *IEEE Robotics & Automation Magazine*, Vol. 12, no. 1, pp. 50-58, Mar. 2005.

[168] Y.-. Ki and D.-Y. Lee, "A traffic accident recording and reporting model at intersections," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 2, pp. 188-194, Jun. 2007.

[169] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour Detection and Hierarchical Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 898-916, May 2011.

[170] Y. LeCun, L. Bottou, G. B. Orr, and K. Müller, "Efficient backprop," *Neural networks: Tricks of the trade*, pp. 9–48, 2012.

[171] G.E. Hinton, N.Srivastava, A. Krizhevsky, I. Sutskever, and R.R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," arXiv preprint arXiv:1207.0580, 2012.

[172] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2009.

[173] A. Vedaldi and K. Lenc, "MatConvNet - Convolutional Neural Networks for MATLAB," in *Proceedings of the ACM International Conference on Multimedia*, 2015.

[174] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the ACM International Conference on Multimedia*, 2015.
