Robust Background Subtraction for Network Surveillance in H.264 Streaming Video

Bhaskar Dey and Malay K. Kundu, *Senior Member, IEEE*

***Abstract*— The H.264/Advanced Video Coding (AVC) is the industry standard in network surveillance owing to improved video quality, low bandwidth and latency requirements. This paper presents a novel approach for background subtraction in H.264 encoded bitstreams. Temporal statistics of proposed feature vectors, representing macroblock units in each frame, are used to identify potential candidates containing moving objects. From the set of coarsely localized candidate macroblocks, all foreground pixels are detected by comparing them pixel wise with a background model.**

The basic difference of the current work compared to the related approaches is that, it allows each macroblock to have a different quantization parameter, in view of the requirements in variable as well as constant bitrate applications. Additionally, for pixel wise comparisons, a new color differencing technique of low complexity is proposed which enables us to obtain pixel-resolution segmentation incurring a negligible cost compared to those of classical pixel domain approaches. Results showing striking comparison against those of proven state-of-the-art pixel domain algorithms are presented over a diverse set of standardized surveillance sequences.

***Index Terms*— H.264/AVC, background subtraction, video surveillance, compressed domain algorithm.**

I. Introduction and Motivation

IN RECENT years, there has been a considerable interest in the use of network surveillance over internet protocol (IP surveillance) for a wide range of indoor and outdoor applications. This is driven by the advent of technology enabling replacement of analog closed-circuit television (CCTV) systems with network cameras coupled with increasing private and public security concerns. However, the limitations and deficiencies, together with the costs associated with human operators in monitoring the overwhelming multitude of streaming feeds, have created urgent demands for automated video surveillance solutions.

Background subtraction [1]-[3] is a fundamental process used in most video-based surveillance systems. Classical *pixel-based techniques* for background subtraction use low-level information about each pixel. However, most video transmitted over networks are encoded bitstreams such as those produced by MJPEG, MPEG-x, and H.26x-compliant encoders. Therefore, such algorithms necessitate prior decoding of each frame, involving substantially huge overhead in terms of computation as well as memory space. These algorithms have a good segmentation performance, but most of them cannot work in real-time because they have to process every pixel in each frame individually. Under the circumstances, it is desirable that new background subtraction techniques be able to process encoded data of considerably smaller size, directly for segmentation. Such techniques, collectively called *compressed domain algorithms*, avert significant computational effort by reusing aggregated information of blocks of pixels already available in coded form to localize moving objects in a given frame. This motivates the development of a background subtraction algorithm for bitstreams encoded in the latest video compression standards like H.264 [4].

The H.246/AVC standard brings new possibilities in the field of video surveillance [5]. It outperforms previous coding standards on its goals of supporting:

1) Significantly higher video resolution at the same bit rate;
2) Error resilience features so that transmission errors over various networks are tolerated;
3) Low latency capabilities. 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.

In this paper, a novel feature vector is proposed that effectively describes macroblock (MB) data in compressed video. The temporal statistics of feature vectors are used to select a set of potential MBs containing object motion. Subsequently, comparison of pixels constituting the selected MBs is used to eliminate those that are similar to the background. The proposed method is embedded into the decoding process and obtains pixel-level segmentation at the cost of a negligible processing overhead.

The subsequent sections are organized as follows. Related

Manuscript received September 4, 2012; revised January 10, 2013 and February 20, 2013; accepted February 26, 2013. This paper was recommended for acceptance by Associate Editor S. Takamura.

B. Dey is with the Center for Soft Computing Research, Indian Statistical Institute, Kolkata 700108, India (e-mail: bhaskar.dey09@gmail.com).

M. K. Kundu is with the Machine Intelligence Unit, Indian Statistical Institute, Kolkata-700108, India (e-mail: malay@isical.ac.in).

Digital Object Identifier XX.XXXX/TCSVT.2013.XXXXXX

work is presented in Section II. Section III presents the feature vector representation for MB data. Section IV introduces the proposed method. Experimental results are presented in Section V and Section VI concludes the paper.

II. Related Work

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 [6] 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*. [7] proposed an algorithm to segment moving objects from the sparse motion vector (MV) 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*. [8] use complex binary partition tree to segment normalized MV field into motion-homogeneous regions. The complexity of the algorithm increases drastically with a noisy MV field. Solana-Cipres *et al*. [9] incorporates fuzzy linguistic concepts that use MV and MB decision modes. Fei and Zhu [10] present a mean shift clustering based moving object segmentation approach using normalized MV field and partitioned block size. It fails to detect objects with slow or intermittent motion. W. You *et al*. [11] introduce 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 MV field. However, as MVs do not *necessarily* correspond to *true object motion* and end up wrongly detecting the dynamic components (eg. waving tree branches, fountain, ripples, camera jitter etc.) of the background. Poppe *et al*. [12] propose an alternative technique that relies on the total number of bits required to encode a MB. This technique fails to detect slow or intermittent object motion.

A major drawback of the related approaches is the assumption of a *fixed* quantization parameter (QP) for all MBs thereby limiting them to *variable bit rate* (VBR) applications with uncontrolled bitrate. 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 MB. Unlike the related approaches, our algorithm allows each MB to have a different QP in consideration of the stringent bandwidth requirements of a practical surveillance scenario.

Secondly, we perform filtering on the aggregate result of MB-level object segmentation obtained jointly from the MV field and the residual data rather than applying separate filtering schemes on either or both of the features in isolation. This enables us to reduce computation while relying more on the consensus between the MB features.

The performance of the related approaches are limited to coarse MB-level segmentation on sequences with static background. In contrast, we introduce a low-complexity pixel differencing technique to obtain pixel-level segmentation on sequences with highly dynamic background.

III. The Proposed Macroblock Feature Vector

A. H.264 preliminaries

A bitstream sequence encoded in H.264 consists of a sequence of *pictures* or *frames*, each of which is split into MB units covering a fixed-sized square area of 16×16 pixels. An *intra-coded* or I-macroblock is predicted from spatially neighboring samples of previously coded blocks in the same frame. On the other hand, most MBs are predicted using references to one or more previously coded block(s) of pixels in past frames or a combination of past and future frames. A bi-predictive or B-macroblock refers to past *and* future frames while a predictive or P-macroblock refers to past frames only. These are collectively called *inter-coded* MBs which require each referred block be indicated using a MV with the corresponding reference frame index. Additionally, the difference between the predicted and the target MB, i.e., the prediction error or *residual* is transformed, quantized, and finally entropy coded. Thus, the MVs as well as the residual component, bearing complementary information of the same MB, are the key components of its feature vector representation.

For network surveillance, the most commonly used encoding profile is the ‘baseline profile’. It uses less computational resources and has a very low latency, ideal for live surveillance video. It is achieved by using a simplified implementation of the H.264 standard based only on I-frames and P-frames. I-frames are coded using I-macroblocks only, and require more bits to encode compared to other frame types. P-frames, which are mainly coded using P-macroblocks, may contain I-macroblocks as well. A video sequence starts with an I-frame and mostly contains P-frames with I-frames at regular intervals. Therefore, the number of I-frames in a given sequence being negligible is not considered in the proposed method.

B. Feature #1: Mean of Absolute Transformed Difference (MATD)

MATD represents the information associated with the residual component of a MB. It is quantified as the arithmetic mean of the transform coefficients comprising a MB residual. We formulate MATD using the statistical properties of Discrete Cosine Transform (DCT) coefficients.

In literature, it is found to be most appropriate and convenient to model the distribution of DCT coefficients by Laplacian distributions [13]-[15]. A Laplacian probability density function (pdf) is given by:

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

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

In H.264 baseline encoding, 4×4 DCT (integer

implementation) operates on similar-sized blocks of residual data. The resulting blocks are subsequently *scaled and quantized* element-wise by a quantization matrix. The quantization scheme most commonly adopted is *uniform quantization with dead-zone* [16]. In this process, the value z of input coefficient is, in general terms, quantized as:

$$k = \lfloor (|z| + f)/Q \rfloor \mathrm{sgn}(z), \tag{2}$$

where $k \in \mathbb{Z}$ 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, and $f \in [0, Q/2]$ is the rounding offset that controls the width of the dead zone. Typically, f is set to be $Q/6$ for P-frames in an effort to approximate the distribution of coefficients in a quantization interval with a Laplacian pdf. The *decoder* process reconstructs the output approximation z_k of the input coefficient z as:

$$z_k = kQ \tag{3}$$

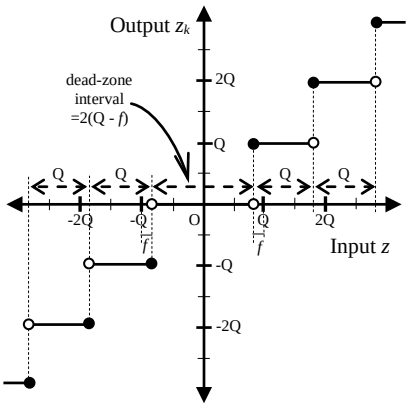

Fig. 1. Relation between the input coefficient z and the reconstructed output z_k for quantization step size Q and rounding offset f.

Fig. 1 illustrates the overall relationship between z and z_k. It may be verified that the value z of any input coefficient is mapped to kQ in the quantization process, where:

i) k=0 if $z \in (-Q + f, Q - f)$;

ii) k=−1, −2, −3,… if $z \in \left(\left(k - 1 + \frac{f}{Q}\right)Q, \left(k + \frac{f}{Q}\right)Q\right]$;

iii) k=1,2,3,… 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 (1) and substituting f=Q/6 for the limits of intervals as mentioned above, we have:

$$P(z_k) = \begin{cases} \int_{(k-\frac{5}{6})Q}^{(k+\frac{1}{6})Q} p(z)dz = \exp(-2r(1-3k))\sinh 3r; & \text{if } k < 0 \\ \int_{-\frac{5}{6}Q}^{\frac{5}{6}Q} p(z)dz = 1 - \exp(-5r); & \text{if } k = 0 \\ \int_{(k-\frac{1}{6})Q}^{(k+\frac{5}{6})Q} p(z)dz = \exp(-2r(1+3k))\sinh 3r; & \text{if } k > 0 \end{cases} \tag{4}$$

where,

$$r = Q/6b. \tag{5}$$

The standard does not specify Q directly for each coefficient individually but rather uses a quantization parameter QP, whose relationship to Q for a 4×4 block of transform coefficients is given by the quantization matrix

$$Q = \begin{bmatrix} q(QP,0) & q(QP,2) & q(QP,0) & q(QP,2) \\ q(QP,2) & q(QP,1) & q(QP,2) & q(QP,1) \\ q(QP,0) & q(QP,2) & q(QP,0) & q(QP,2) \\ q(QP,2) & q(QP,1) & q(QP,2) & q(QP,1) \end{bmatrix}, \tag{6}$$

where

$$q(QP,n) = q_0\left(\mathrm{mod}(QP,6), n\right) 2^{\lfloor QP/6 \rfloor}; \tag{7}$$

q_0 being a scalar multiplier as defined in Table-I [17]. It may be noted in (6), that $1/2$ 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. Therefore, r can take only three possible values $\frac{q(QP,0)}{6b}$, $\frac{q(QP,1)}{6b}$ or $\frac{q(QP,2)}{6b}$, for given values of QP and b.

TABLE I
SCALAR MULTIPLIER q_0

mod(QP,6)	$n=0$	$n=1$	$n=2$
0	0.6250	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

Given the fact that the quantized coefficients z_k of the residual 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 (4), the above expression simplifies to

$$H(r) = \frac{\exp(-2r)}{\sinh 3r \ln 4}\left(6r + (1-\exp(-6r))(2r - \ln\sinh 3r)\right) - (1-\exp(-5r))\log_2(1-\exp(-5r)). \tag{8}$$

Total bits B required in coding the residual data may be estimated as the product of H and the total number of residual transform coefficients. For sequences encoded in the popular YCbCr 4:2:0 format, a MB 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

$$\begin{aligned} B &= 384\left(\tfrac{1}{4}H\left(\tfrac{q(QP,0)}{6b}\right) + \tfrac{1}{4}H\left(\tfrac{q(QP,1)}{6b}\right) + \tfrac{1}{2}H\left(\tfrac{q(QP,2)}{6b}\right)\right) \\ &= 96H\left(\tfrac{q(QP,0)}{6b}\right) + 96H\left(\tfrac{q(QP,1)}{6b}\right) + 192H\left(\tfrac{q(QP,2)}{6b}\right). \end{aligned} \tag{9}$$

Table-I 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 (9) may be expressed in one unknown. For known values of B and QP (obtained from the MB header), we use numerical methods to evaluate $\frac{q(QP,0)}{6b}$ as it is difficult to derive its closed form solution by analytical means.

The MATD for the current MB is computed as the mean absolute value of $P(z_k)$.

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

Using (5) and (6), the above expression simplifies to (10).

$$MATD = \frac{q(QP,0)\exp\left(-\frac{q(QP,0)}{3b}\right)}{8\sinh\left(\frac{q(QP,0)}{2b}\right)} + \frac{q(QP,1)\exp\left(-\frac{q(QP,1)}{3b}\right)}{8\sinh\left(\frac{q(QP,1)}{2b}\right)} + \frac{q(QP,2)\exp\left(-\frac{q(QP,2)}{3b}\right)}{4\sinh\left(\frac{q(QP,2)}{2b}\right)}. \quad (10)$$

Substituting the value of $\frac{q(QP,0)}{6b}$ and its scalar multiples $\frac{q(QP,1)}{6b}$ and $\frac{q(QP,2)}{6b}$ in (10), we finally obtain MATD. 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*) to reduce repetitive run time costs. With minor memory overhead, this is a good tradeoff for higher execution speed.

An experimental validation comparing real MATD with predicted MATD is presented in Section V.

C. Feature #2: Sum of Normalized Motion Vector Magnitudes (SNMVM)

SNMVM represent the motion compensated information associated with the MVs of a MB. H.264 introduces two new coding characteristics in the motion compensation: variable partition size and multiple reference frames. Each inter coded MB may be predicted using a range of block sizes. Accordingly, the MB is split into one, two or four MB partitions using either:

a) one 16×16 partition (covering the whole MB),
b) two 8×16 partitions,
c) two 16×8 partitions or
d) four 8×8 partitions.

If 8×8 partition size is chosen, then each 8×8 block, heretofore a sub-MB, is split into one, two or four sub-MB partitions): one 8×8, two 4×8, two 8×4 or four 4×4 sub-MB partitions. Each partition and sub-MB partition of a P-macroblock has a MV (mvx_i, mvy_i) pointing to an area of the same size in a reference frame, which is used to predict the current (say *i*th) partition. Each partition in a given MB may be predicted from different reference frame(s). However, the sub-MB partitions within an 8×8 sub-MB share the same reference frame. Let us denote the reference index of the *i*th partition of a MB as f_i. A P-frame having frame number *t* is predicted from a list of previous frames where reference index 0 represents frame (*t*-1), reference index 1 represents frame (*t*-2) and so on. In order to obtain uniformity among the partitions we normalize them by f_i and assign fixed weights to each of them corresponding to the ratio of the MB area it represents. The weight w_i of the *i*th partition is defined as the fraction of the partition size contributing to the whole MB, i.e., 256. Assuming *p* partitions in the current MB, the computation of SNMVM is illustrated in Fig. 2. It may be noted that the value of SNMVM for an I-macroblock in P-frame *is taken as zero*.

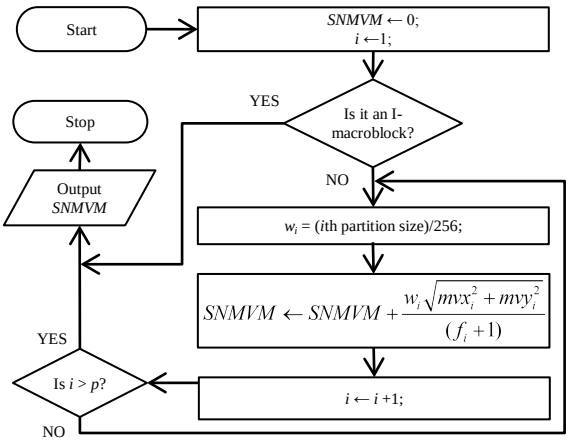

Fig. 2. Computation of SNMVM.

The computation involved in this step amounts to 4 multiplications/divisions and 4 additions for each partition, the number of partitions being no more than 16 for any given MB.

D. The macroblock feature vector

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

$$\vec{p}_{t,idx} = \left[x,\, y\right]^{\mathrm{T}}, \quad (11)$$

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

E. Initialization & incremental update of covariance

In the proposed approach, static and dynamic component present at different locations in scene backgrounds are modeled with temporally weighted local covariance for each location *idx*=0,1,…,*T*-1. Feature vectors computed for all locations are stored in *T* fist-in first-out (FIFO) buffers, each having a predefined capacity *M*. A new MB vector is *inserted* at the *rear*. However, if the buffer is full, the vector at the *front*, being least relevant in the temporal order of vectors in the buffer, is *removed* to make room for the new one. Corresponding to each position *i* in a buffer starting from *i*=1 at the *front* to *i*=*M* at the *rear*, predefined weights $\{W_i = i/M\}_{i=1}^{M}$ are assigned on the basis of temporal order.

Let us assume the buffer under consideration to be in a state in which all *n* vectors from the sequence $\left\{\left[x_i,\, y_i\right]^{\mathrm{T}}\right\}_{i=1}^{n}$ are inserted, with the *n*th sample of the sequence 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. Therefore, the required covariance matrix Σ_{idx} is expressed as in (12).

$$\Sigma_{idx} = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix} \quad (12)$$

Weighted variance σ_x^2 is defined as

$$\sigma_x^2 = \frac{\sum_{i=\max(1,n-M+1)}^{n} W_{M-n+i}\left(x_i - \bar{x}\right)^2}{\sum_{i=\max(1,n-M+1)}^{n} W_{M-n+i}} = \left(\overline{x^2} - \bar{x}^2\right) \quad (13)$$

where,

$$\bar{x} = \sum_{i=\max(1,n-M+1)}^{n} W_{M-n+i}x_i \Big/ \sum_{i=\max(1,n-M+1)}^{n} W_{M-n+i} \quad (14)$$

and

$$\overline{x^2} = \sum_{i=\max(1,n-M+1)}^{n} W_{M-n+i}x_i^2 \Big/ \sum_{i=\max(1,n-M+1)}^{n} W_{M-n+i}. \quad (15)$$

Similarly, we have

$$\sigma_y^2 = \left(\overline{y^2} - \bar{y}^2\right) \quad (16)$$

and

$$\sigma_{xy} = \left(\overline{xy} - \bar{x}\bar{y}\right) \quad (17)$$

where

$$\bar{y} = \sum_{i=\max(1,n-M+1)}^{n} W_{M-n+i}y_i \Big/ \sum_{i=\max(1,n-M+1)}^{n} W_{M-n+i}, \quad (18)$$

$$\overline{y^2} = \sum_{i=\max(1,n-M+1)}^{n} W_{M-n+i}y_i^2 \Big/ \sum_{i=\max(1,n-M+1)}^{n} W_{M-n+i}, \quad (19)$$

and

$$\overline{xy} = \sum_{i=\max(1,n-M+1)}^{n} W_{M-n+i}x_iy_i \Big/ \sum_{i=\max(1,n-M+1)}^{n} W_{M-n+i}. \quad (20)$$

Direct computation of Σ_{idx} using (13), (16) and (17) following every insertion would be computationally prohibitive and inefficient as most of the samples in the buffer remain unaltered between subsequent insertions. Hence, recursive counterparts of (14), (15), (18), (19), and (20) are formulated to facilitate online update of Σ_{idx}.

The recursive update of $\bar{x}$ is formulated 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*. If $n>M$, 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 of weights $[W]_n$ to obtain $\bar{x}$. Equations (21) and (22) summarize the initialization and recursive update process of Σ_{idx} for the nth insertion.

$$\begin{bmatrix} \overline{x^a y^b} \\ S_{x^a y^b} \end{bmatrix}_n = \begin{cases} \left[x_1^a y_1^b,\ x_1^a y_1^b \right]^{\mathrm{T}}; & \text{if } n = 1 \\ \left[\dfrac{\overline{x^a y^b} W - (S_{x^a y^b}/M) + x_n^a y_n^b}{W + W_{M-n+1}} \right]_{n-1}; & \text{if } 1 < n \leq M \\ \left[\dfrac{\overline{x^a y^b} W - (S_{x^a y^b}/M) + x_n^a y_n^b}{W} \right]_{n-1}; & \text{if } n > M. \end{cases} \quad (21)$$

where ordered pair $(a,b)\in\{(1,0), (2,0), (0,1), (0,2), (1,1)\}$. Similarly, the recursive counterparts of (15), (18), (19) and (20) may be obtained from (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 $\bar{x}, \overline{x^2}, \bar{y}, \overline{y^2}$, and $\overline{xy}$ are followed by an adjustment of the sum of weights $[W]_n$ as:

$$[W]_n = \begin{cases} 1; & \text{if } n = 1 \\ [W + W_{M-n+1}]_{n-1}; & \text{if } 1 < n \leq M \\ [W]_{n-1}; & \text{if } n > M. \end{cases} \quad (22)$$

The overall process of updating Σ_{idx} requires *no more* than 14 multiplications, 10 divisions and 21 additions.

IV. The Proposed Method

A. System overview

As highlighted in the introductory section, the proposed

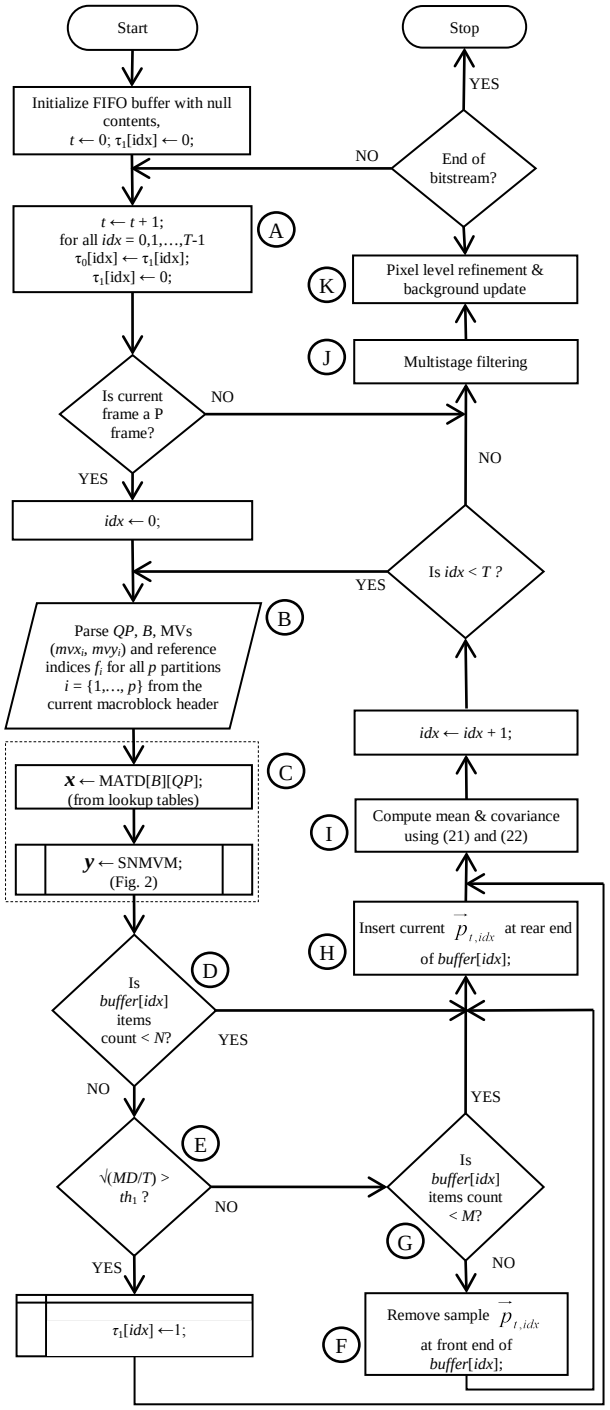

Fig. 3. Proposed approach in H.264 compressed bitstream.

method operates in two stages as follows:

i) performing a coarse motion classification, using temporal covariance of MBs, to determine a set of potential candidates containing moving objects; and
ii) performing pixel-wise comparison in order to eliminate background pixels from the set of candidate MBs.

The flowchart of the proposed method for segmentation of moving objects in H.264 bitstream is illustrated in Fig. 3. In order to compute temporal covariance for each MB in a frame, T identical FIFO buffers (indexed by idx) are used. The MB level segmentation of the current frame 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 and τ_1 is initialized to zero (block A). As usual, frame numbers are denoted using t. Parameters B, QP and MVs, which are parsed from the header data of the current MB (block B) are used to compute its feature vector as described in Section III (block C).

Using the first N frames, a median background frame is initialized (Section IV-D). At the same time, feature vectors $\forall\, idx \in \{0,1,\ldots,T-1\}$ are inserted (block I) into the corresponding buffers, i.e., $buffer[idx]$. Temporally weighted mean $\vec{\mu}_{idx}$ and covariance Σ_{idx} are computed online (block I) using (21) and (22). Buffered feature vectors for a given MB represent locally varying background model in the compressed domain. Once the total contents in a given buffer $buffer[idx]$ reaches N (block D), any input vector $\vec{p}_{t,idx}$ is considered for further insertion, based on the criteria as follows. Let

$$\tau_1[idx] = \begin{cases} 1 \text{ (foreground candidate);} & \text{if } \sqrt{MD/T} > th_1 \\ 0 \text{ (background);} & \text{otherwise} \end{cases} \quad (23)$$

where $MD = (\vec{p}_{t,idx} - \vec{\mu}_{idx})^{\mathrm{T}} \Sigma_{idx}^{-1} (\vec{p}_{t,idx} - \vec{\mu}_{idx})$ is the Mahalanobis distance and th_1=0.225 being a predetermined threshold. If $\tau_1[idx]$=0, $\vec{p}_{t,idx}$ is inserted into $buffer[idx]$. Otherwise, the current MB is selected as one of the probable candidates expected to contain part(s) of a moving object. Should the buffer be already full (block G), the vector at the *front* is removed (block F) prior to the insertion of $\vec{p}_{t,idx}$ (block H) at the *rear*. The coarse MB-level segmentation so obtained, is filtered (block J) using the procedure described in Section IV-B. Pixels constituting the set of filtered candidate MBs are used to obtain precise object segmentation (block K), details of which are presented in Section IV-D.

Fig. 4 shows the scatter plots of the buffered feature vectors corresponding to different MB locations at 330 and 591 (highlighted as white squares) for frame t=1896 from *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 modeled with a sparsely distributed scatter with high $|\Sigma_{330}|$.

B. Multistage filtering for noisy macroblocks

As mentioned in (23), the binary decision $\tau_t[idx]$ for idx=1,…,T-1 constituting the set of candidate MBs may erroneously include those that do not correspond to true object motion, i.e., false positives. Such noisy MBs tend to appear as flickering 'speckles' in the reconstructed output. To suppress such false positives, we use multi-staged filter banks in which the output of a 3×3×2 spatio-temporal median filter (STMF) is cascaded to a 3×3 Gaussian filter. Let us consider frames of width u MBs and height v MBs as shown in Fig. 5. The spatio-temporal window for the candidate MB at location idx in frame t is indicated with shaded blocks. The output of the STMF is computed as the median of $\tau_0[idx]$, $\tau_1[idx]$, $\tau_1[idx\text{-}1]$, $\tau_1[idx\text{+}1]$, $\tau_1[idx - u]$, and $\tau_1[idx + u]$.

The STMF often erroneously removes MBs containing boundaries of slow moving objects. In addition, the output of STMF may occasionally contain small 'holes' in the segmented regions that correspond to very large moving

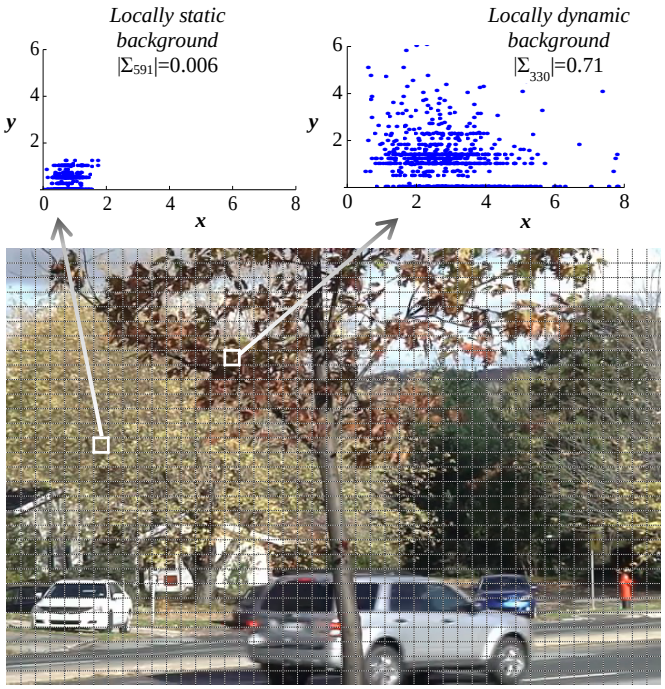

Fig. 4. Modeling background variance in the proposed feature space.

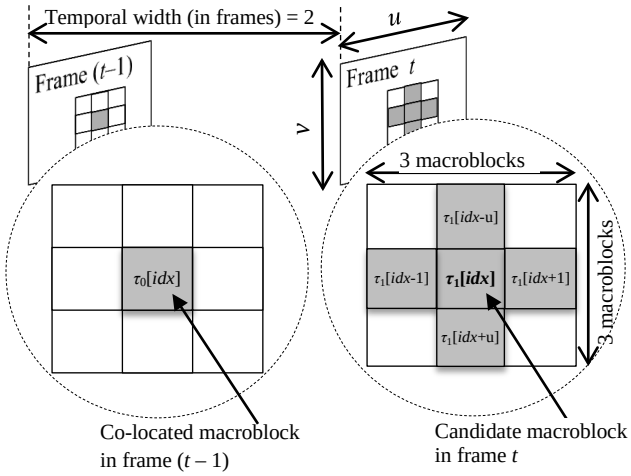

Fig. 5. Illustration of 3×3×2 support for the STMF (enlarged insets).

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 MB-level.

It may be noted that an *optimized* algorithm for computation of the above median (of 6 binary numbers) would require no more than 9 comparisons. The symmetric Gaussian filter additionally requires 3 multiplications and 8 additions.

C. Proposed pixel differencing technique

In the proposed method, a low-complexity color difference technique is used to ascertain if the colors of any two given pixels are either *similar* or *different*. Considering two pixels having YCbCr co-ordinates $X_1 \equiv (Y_1, Cb_1, Cr_1)$ and $X_2 \equiv (Y_2, Cb_2, Cr_2)$, let luminance differential $\Delta Y = |Y_1 - Y_2|$ and chrominance differential $\Delta C = |Cb_1 - Cb_2| + |Cr_1 - Cr_2|$. A pair of adaptive thresholds t_Y (for luminance) and t_C (for chrominance) are used, details of which are in Section-V. Colors X_1 and X_2 are considered similar if $\Delta Y < t_Y$ *and* $\Delta C < t_C$, X_1 and X_2 are, else different. This is in contrast to most existing techniques (in pixel domain) where the similarity of pixels is determined based on the Euclidean distance or straight-line between the given points in RGB color space. It is obvious that the proposed similarity measure involves fewer computation compared to those of Euclidean distance based techniques. The proposed technique exploits the principal advantage of YCbCr color space, i.e., *decoupling* of luminance (or brightness) and chrominance (or color) information which are processed independently of each other. As luminance is separated from chrominance, the space is less affected by visible shadows.

D. Background initialization and update process

In order to obtain accurate segmentation of moving objects, a background image is initialized and then updated in order to incorporate gradual changes in the scene. For initialization of the background image, the sequence of first N (say 100) frames are divided into 10 equal-sized groups, from each of which one frame is randomly picked for computation of the temporal median frame. The median frame is used as the background image corresponding to the frame $t=N+1$.

For frame $t > N$, pixels constituting the (filtered) set of candidate MBs in the current frame are compared with the corresponding pixels in the background frame. If the pixels are found to be *different* (as discussed in Section IV-C), the corresponding pixel in the background subtraction mask is labeled as *foreground*. The remaining pixels in the mask constitute the *background*. This produces pixel resolution segmentation of objects in the current frame. If the number of pixels labeled a foreground exceeds 25% of the total pixels in a given MB, the pixel wise comparison process is repeated for all pixels constituting the corresponding MB in the following frame. This enforces inter frame continuity of segmentation masks as MBs containing parts of *slow-moving objects* often go undetected.

MBs corresponding to which $\tau_1[idx]=0$ according to (23) *and* having $\sqrt{(MD/T)} < th_2$ (say 0.1), represent the background. Pixels constituting such MBs are used to update the corresponding pixels $B_t(x, y)$ in the existing background using (24). The number of such MBs in a given frame, say β, is practically very small.

$$B_{t+1}(x,y) = \alpha I_t(x,y) + (1-\alpha) B_t(x,y) \text{ for } t > N \quad (24)$$

where $I_t(x, y)$ is the pixel's intensity value for frame t; α=0.05 is a predefined learning rate that determines the tradeoff between stability and quick update.

V. Experimental Results

Following the discussion in Section III-B, we provide a statistical comparison of the actual MATD against the predicted MATD values of all MBs comprising selected frames at regular intervals from the *Traffic* sequence. The sequence was encoded in VBR as well as CBR modes as indicated in Fig. 7. As expected, 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 bit-rate. It is observed that the actual MATD and the predicted MATD values are *very highly correlated* (correlation coefficient ρ > 0.98). In (23), we used Mahalanobis distance, which is invariant under arbitrary non-singular linear transformations. This makes predicted MATD a qualified 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 MB decoding module of *libavcodec*, an open source audio/video codec library that is developed as a part of the FFmpeg [18] project. We evaluate our approach on the benchmark dataset provided for the Change Detection Challenge 2012 [19]. As the proposed method uses pixel level information in the final stage of motion classification, we consider the results of five state-of-the-art pixel domain techniques [20]-[24] for a fair comparison. All video sequences from *Baseline*, *Dynamic Background* and *Camera Jitter* categories of the dataset were encoded in two different ways: i) VBR with a fixed QP=25 for

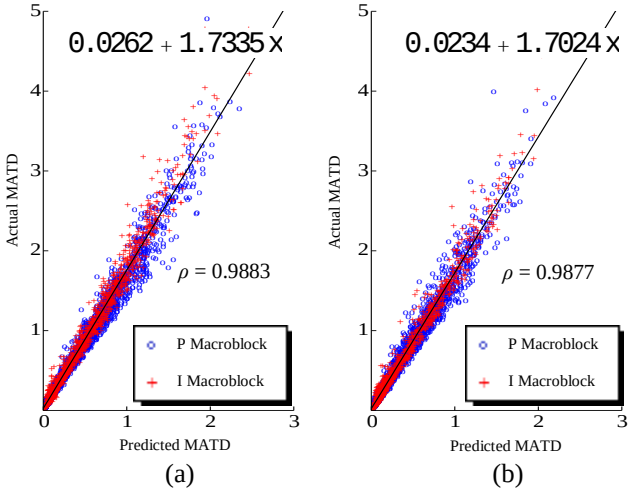

Fig. 6. Scatterplot of Actual MATD vs. Predicted MATD (with regression line). (a) VBR with fixed QP=25 and (b) CBR @1024kbps.

all MBs; and ii) CBR with a target bitrate of 1024kbps. The encoder configuration was set as follows: baseline profile, chroma sampling 4:2:0 progressive, GOP size 250, and the range of MVs (using hexagonal search) being ±16. The Decoded Picture Buffer (DPB) size was fixed at 3, *i.e.*, the number of distinct frames each P-frame can use *at the most* as references. The decoding frame rate was set to 25 frames/sec.

The results of the proposed method for a few selected frames are shown in Fig. 7, for visual/qualitative evaluation against the specified ground-truth masks. For quantitative evaluation, a set of seven performance metrics defined in [19] *viz*. recall, specificity, false-positive rate (FPR), false-negative rate (FNR), percentage of bad classification (PBC), F-measure and precision have been used together with processing speed to compute average rank. We used a total of four decision thresholds in our experiments out of which, *fixed* parameters th_1=0.225 and th_2=0.1 lead to excellent results in most situation while t_Y=18.0(1+4.5|Σ_{idx}|) and t_C=0.5(1+8.4|Σ_{idx}|) are *locally adaptive* and linearly depending on |Σ_{idx}| for a given MB (constants were empirically set based on optimal performance over a wide range of dynamic sequences).

An exhaustive comparison of the results of the proposed method with those of [20]-[24] (applied on individual images *prior to encoding*) is summarized in Table-II. Subscripts indicate rank in the corresponding performance category.

Prior to performance comparison, it is important to realize that every codec can give a *varying degree of quality* for a given set of source frames. Any degradation of visual data introduced by *lossy* compression will inevitably remain visible through any further processing of the content. Notwithstanding such practical constraints that are ignored in pixel domain approaches, our algorithm outperforms the current state-of-the-art techniques in most of the performance measures. The processing times for the VBR encoded videos are found to be comparatively longer than those of CBR encoded counterparts owing to higher average bitrates.

As discussed in Section-III, the computation involved per MB in each step of the proposed method cost up to constant factor. Consequently, the complexity of the overall process is $\mathcal{O}\left(T + c_1\beta + c_2\sum_{idx=0}^{T-1}\tau_{1,idx}\right)$, where $\sum_{idx=0}^{T-1}\tau_{1,idx}$ denotes the number of candidate MBs that require pixel level processing, T is the total number of MBs per frame and c_1, c_2 are constants. Thus, the proposed method delivers real-time performance incurring a negligible processing overhead $\kappa = \left(c_1\beta + c_2\sum_{idx=0}^{T-1}\tau_{1,idx}\right)$ to the standard decoding cost claimed by each frame. Table-II lists the average processing speed (frames/sec.) on sequences with 720x576 pixel resolution for each method on our platform (Intel® Core™ i7-2600 CPU @3.40 GHz with 16GB RAM). As shown, the proposed method achieves remarkably higher speed compared to those of the current state-of-the-art.

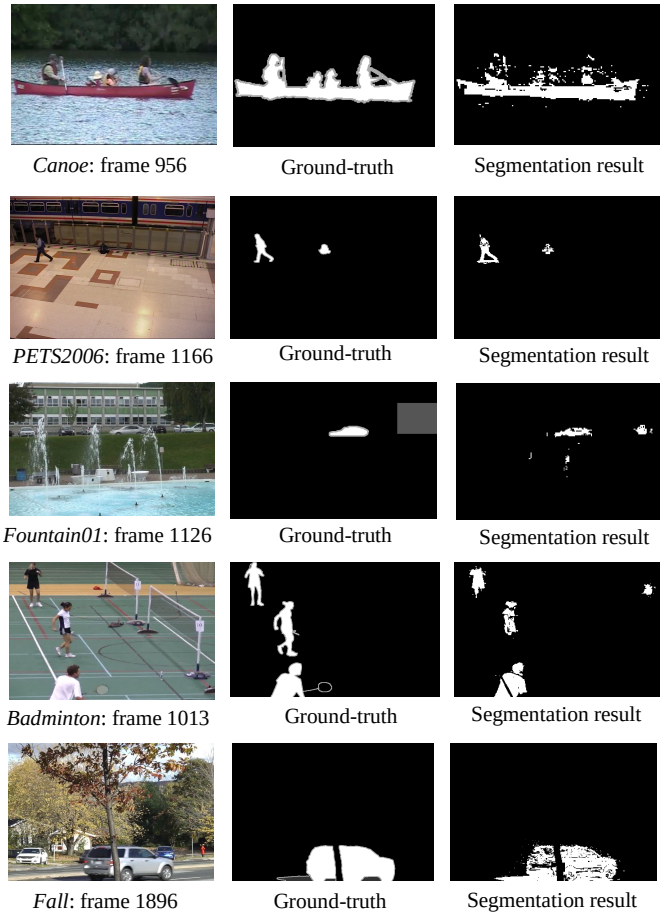

Fig. 7. Segmentation results of the proposed method for selected frames.

VI. Conclusion

This paper introduces a novel approach for background subtraction algorithm in H.264 (although the same algorithm should work for MPEG-2/4 codecs as well) compressed bitstream. The proposed methodology is not only built for real-time applications in consideration of practical bandwidth constraints but has proven to be robust to diverse set of real-world (non-synthetic) video sequences.

TABLE II
QUANTITATIVE EVALUATION (CATEGORIES: BASELINE, DYNAMIC BACKGROUND & CAMERA JITTER)

Method	Average Recall	Average Specificity	Average FPR	Average FNR	Average PWC	Average F-Measure	Average Precision	Avg. speed (in frames/sec.)	**Average Rank**
Proposed (VBR)	0.6272_5	0.9965_2	0.0035_2	0.3728_5	1.4664_2	0.7048_3	0.8533_2	409.6_2	**2.8750**
Proposed (CBR)	0.5858_6	0.9969_1	0.0031_1	0.4142_6	1.5506_3	0.6854_5	0.8604_1	451.2_1	**3.0000**
PBAS [24]	0.7974_3	0.9932_4	0.0067_4	0.2026_3	1.1711_1	0.7764_1	0.8284_3	4.0_7	3.2500
SC-SOBS [20]	0.8786_1	0.9862_5	0.0138_5	0.1214_1	1.6480_4	0.7691_2	0.7304_5	4.4_6	3.6250
GMM [22]	0.5747_7	0.9953_3	0.0047_3	0.4253_7	1.8339_5	0.6525_7	0.8043_4	25.8_4	5.0000
ViBe [21]	0.7513_4	0.9857_6	0.0143_6	0.2487_4	2.0605_6	0.6782_6	0.6641_6	126.8_3	5.1250
KDE [23]	0.8119_2	0.9798_7	0.0202_7	0.1881_2	2.4414_7	0.6925_4	0.6606_7	7.5_5	5.1250

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers and the associate editor for their valuable comments that significantly improved the quality of this paper.

REFERENCES

[1] Y. Benezeth, P.-M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger. "Comparative study of background subtraction algorithms," *J. Electron. Imaging*, vol. 19, no. 3, pp. 1-12, Jul. 2010.

[2] S. Elhabian, K. El-Sayed, and S. Ahmed, "Moving object detection in spatial domain using background removal techniques – State-of-art," *Recent Patents on Computer Science*, vol. 1, pp. 32–54, Jan. 2008.

[3] T. Bouwmans, F. El Baf, and B. Vachon, "Statistical background modeling for foreground detection: A survey," in *Handbook of Pattern Recognition and Computer Vision*, vol. 4, ch. 3, pp. 181–199. *World Scientific Publishing*, Jan. 2010.

[4] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560-576, Jul. 2003.

[5] "H.264 video compression standard - New possibilities within video surveillance," *White paper, Axis Communications Inc.*, Mar. 2008.

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

[7] 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.

[8] Z. Liu, Z. Zhang, L. Shen, "Moving object segmentation in the H.264 compressed domain," *Opt. Eng.*, vol. 46, no. 1, 017003 (Jan. 16, 2007); doi: 10.1117/1.2431374.

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

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

[11] W. You, M.S. H. Sabirin, and M. Kim, "Moving Object Tracking in H.264/AVC bitstream," *MCAM 2007*, LNCS, vol. 4577, pp.483-492, Springer, Heldelberg (2007).

[12] C. Poppe, S. D. Bruyne, T. Paridaens, P. Lambert, R. V. D. Walle, "Moving Object Detection in the H.264/AVC compressed domain for video surveillance applications," *J. Vis. Commun. Image Representation*, vol. 20, pp. 428-437, May 2009.

[13] S.R. Smoot and L.A. Rowe, "Study of DCT Coefficient Distributions," *Proc. SPIE Symp. Electronic Imaging*, pp. 403-411, Jan 1996.

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

[15] Wei Wu, and Bin Song, "DC Coefficient Distributions for P-Frames in H.264/AVC," *ETRI J.*, vol.33, no.5, pp.814-817, Oct. 2011.

[16] G.J. Sullivan and S. Sun, "On Dead-Zone Plus Uniform Threshold Scalar Quantization," *Proc. SPIE Vis. Commun. Image Process.*, vol. 5960, no. 2, pp. 1041-1052, 2005.

[17] I.E. Richardson, *The H.264 Advanced Video Compression Standard*. 2nd ed., UK: John Wiley & Sons Ltd., 2010, pp. 191.

[18] [Online]. Available: http://ffmpeg.org/

[19] N. Goyette, P.-M. Jodoin, F. Porikli, J. Konrad, and P. Ishwar, "Changedetection.net: A new change detection benchmark dataset," *IEEE Computer Soc. Conf. Computer Vis. Pattern Recog. Workshops*, pp. 1-8, 2012.

[20] L. Maddalena, A. Petrosino, "The SOBS algorithm: what are the limits?," *IEEE Computer Soc. Conf. Computer Vis. Pattern Recog. Workshops*, pp. 21-26, 2012.

[21] O. Barnich and M. Van Droogenbroeck. ViBe: A universal background subtraction algorithm for video sequences. *IEEE Trans. Image Process.*, vol 20, no. 6, pp. 1709–1724, June 2011.

[22] P. KaewTraKulPong and R. Bowden, "An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection," in *Proc. Second European Workshop Advanced Video Based Surveillance Systems*, pp. 149-158, 2001.

[23] A. Elgammal, D. Harwood, L. S. Davis, "Non-parametric Model for Background Subtraction," *Proc. Sixth European Conf. Computer Vision*, pp. 751-767, 2000.

[24] M. Hofmann, P.Tiefenbacher, G. Rigoll "Background Segmentation with Feedback: The Pixel-Based Adaptive Segmenter," *IEEE Computer Soc. Conf. Computer Vis. Pattern Recog. Workshops*, pp. 38-43, 2012.

Bhaskar Dey received his B.Tech. degree in Information Technology from University of Kalyani, Kalyani, India, in 2007, and the M.Tech. degree in the same discipline from University of Calcutta, Kolkata, India, in 2009.

He is currently pursuing his Ph.D. degree in the Center for Soft Computing Research, Indian Statistical Institute, Kolkata, India. His current research interests include compressed domain video and image analysis, machine vision, and pattern recognition.

Malay K. Kundu received his B. Tech., M. Tech. and Ph.D (Tech.) degrees in Radio physics and Electronics all are from the University of Calcutta. In 1982, he joined the Indian Statistical Institute, Calcutta, as a faculty member. Currently he is a full professor in the Machine Intelligence Unit of this Institute. He had been the head of the Machine Intelligence Unit during September 1993 to November 1995 and Professor In-charge (Chairman) of the Computer & Communication Sciences Division of the Institute during 2004 to 2006. He is the Co-Principal Investigator & acting in-charge of the Center for Soft Computing Research: A National Facility (funded by Department of Science and Technology, Govt. of India at Indian Statistical Institute, Kolkata).

His research interests include Image processing & analysis, Soft Computing, Content Based Image Retrieval, Digital watermarking, Wavelets, Genetic algorithms, Machine vision, Fractals, and VLSI design for digital imaging. He has contributed 3 book volumes, about 140 research papers in well-known and prestigious archival journals, international refereed conferences and in the edited monograph volumes. He is the holder of nine U.S patents, two International and two E.U patents.

Prof. Kundu received the prestigious VASVIK award for industrial research in the field of Electronic Sciences & Technology for the year 1999. He also received the Sir. J. C. Bose memorial award of the Institute of Electronics and Telecommunication Engineers (IETE), India in the year 1986. He is a Fellow of a the International Association for Pattern Recognition, USA (IAPR), Indian National Academy of Engineering (INAE), National Academy of Sciences, India and the Institute of Electronics and Telecommunication Engineers (IETE), India. He is a senior member of the IEEE, USA and the founding life member & Vice President of the Indian Unit for Pattern Recognition and Artificial Intelligence (IUPRAI), the Indian wing of the International Association for Pattern Recognition (IAPR).