Poisson Mixture Model for High Speed and Low-Power Background Subtraction

Muhammad Umar Karim Khan and Chong-Min Kyung

1 Introduction

Internet of Things (IoT) has been driving research in industry as well as academia for the past decade. An IoT system presents numerous challenges to designers of different expertise. One of the major tasks of designing an IoT system is to develop the sensors at the front-end of the system. These sensors are required to perform their respective tasks in real-time with limited energy and area. Inefficient implementation of sensors in IoT clusters with battery-operated nodes will result in limited operational time, as high energy consumption will quickly drain the battery. Although energy-efficient communication in IoT has been thoroughly researched, relatively little progress has been made in the design of energy-efficient sensors.

Vision sensors in the form of smart cameras are expected to be a core part of the IoT. This is because many real-life applications depend on visual understanding of a scene. Designing a smart camera for IoT that infers from the scene is a challenging proposition. First, computer vision algorithms have high computational complexity, making them inefficient for IoT. Second, accuracy is generally compromised in hardware implementations of computer vision algorithms for achieving higher speed, lower power, or smaller area.

Background subtraction (BS) is a core computer vision algorithm to segment moving objects from the dynamic or static background. BS algorithms aim to model the background that is robust against lighting changes, camera jitter, and dynamic

M. U. K. Khan (✉)
KAIST, Daejeon, South Korea
e-mail: umar@kaist.ac.kr

C.-M. Kyung
#310 IT Convergence Building (N1), Center for Integrated Smart Sensors, Yuseong-gu, Daejeon, Korea (Democratic People's Republic of)
e-mail: kyung@kaist.ac.kr

Y. Liu et al. (eds.), *Smart Sensors and Systems*,
https://doi.org/10.1007/978-3-030-42234-9_1

backgrounds [1]. Generally, BS provides a region of interest (the moving object) in the frame of a given video, which is either used to trigger an alarm or further analyzed to understand the scene. Computational gain is achieved by only analyzing the moving objects in the scene rather than the whole frame.

Researchers have proposed many BS schemes in the past. Almost all of these schemes target high accuracy of BS. Such an approach is useful in some applications but not for smart cameras. The reason is that the algorithms that solely target high accuracy are computationally complex, resulting in high power consumption, delay, and area. On the other hand, some researchers have proposed dedicated implementations of BS, which provide relatively high speed and low-power consumption. However, these implementations generally degrade the accuracy of the algorithms.

Previously, we have described the use of BS in surveillance systems, its implications and proposed a novel method. Khan et al. [2] presents a strategy for obtaining the ideal learning rate of GMM-based BS to minimize energy consumption. A dual-frame rate system was proposed in [3], which allows efficient use of memory in a blackbox system. We proposed an accurate, fast, and low-power BS scheme in [4].

Shot noise is the most dominant source of noise in camera system. Shot noise is modeled by a Poisson distribution. Therefore, in this paper, we use a Poisson distribution under the shot noise assumption to model the background pixel intensity. In fact, we use a Poisson mixture model (PMM) to model dynamic background. We use a relatively stable approach for online approximation of the parameters of the distribution. Resultantly, the proposed method provides competitive performance compared to common BS schemes.

The rest of the chapter is structured as follows. Section 2 describes some efficient implementations of BS from the literature. In Sect. 3, we present a brief review of EBSCam for BS. Section 4 describes the proposed method of BS. Experimental results are discussed in Sect. 5 and Sect. 6 concludes the chapter.

2 Previous Work

This section is divided into two parts. In the first part, we describe numerous BS algorithms. In the second part, efficient implementations of BS algorithms are presented.

2.1 Background Subtraction Algorithms

Previously, numerous surveys have been performed on BS schemes [5–7]. These BS schemes can be classified into region-based and pixel-based categories.

Region-based schemes make use of the fact that background pixels of the same object have similar intensities and variations over time. In [8], authors divide a frame into $N \times N$ blocks and each block is processed separately. Samples of the N^2 vectors are then used to train a principal component analysis (PCA) model. The PCA model is used for foreground classification. A similar technique is described in [9]. In [10], independent component analysis (ICA) of images from a training sequence is used for foreground classification. A hierarchical scheme for region-based BS is presented in [11]. In the first step, a block in the image is classified as background and the block is updated in the second step.

Pixel-based BS schemes have attracted more attention due to their simpler implementations. In these schemes, the background model is maintained for each pixel in the frame. The simplest of these methods are classified as frame differencing, where the background is modeled by the previous frame [12] or by the (weighted) average of the previous frames. Authors in [13] use the running average of most recent frames so that old information is discarded from the background model. This scheme requires storing some of the previous frames; thereby, larger memory space is consumed. In [14] and [15], a univariate Gaussian distribution is associated with each pixel in the frame, i.e., pixel intensities are assumed to be of Gaussian distribution and the parameters of the distribution (mean and variance) for each pixel are updated with every incoming frame. The mean and the variance of the background model for each pixel are then used to classify foreground pixels from the background.

Sigma-delta (Σ-Δ)-based BS [16, 17] is another scheme which is popular in embedded applications [18, 19]. Inspired from analog-to-digital converters, these methods use simple non-recursive approximates of the background image. The background pixel intensity is incremented, decremented, or unchanged if the pixel intensity is considered greater than, less than, or similar to the background pixel intensity, respectively.

Kernel density estimation (KDE) schemes [20, 21] accumulate the histogram of pixels separately in a given scene to model the background. Although claimed to be non-parametric, the kernel bandwidth of KDE-based schemes needs to be decided in advance. Using a small bandwidth produces rough density estimates whereas a large bandwidth produces over-smoothed ones.

Perhaps the most popular BS methods are the ones based on Gaussians mixture models (GMM). These methods, first introduced in [22], assume that background pixels follow a Gaussian distribution and model a background pixel with multiple Gaussian distributions to include multiple colors of the background. Numerous improvements have been suggested to improve foreground classification [23] as well as speed [24–27] of GMM-based methods. Notable variants of the original work are [28] and [29]. In [28], an adaptive learning rate is used to update the model. In [29], which is usually referred to as extended GMM or EGMM, author uses the Dirichlet prior with some of the update equations to determine the sufficient number of distributions to be associated with each pixel. The Flux Tensor with Split Gaussians (FTSG) scheme [30] uses separate models for the background and foreground. The method develops a tensor based on spatial and temporal information, and uses the tensor for BS.

Another pixel-based method for BS is the codebook scheme [31] and [32], which assigns a code word to each background pixel. A code word, extracted from a codebook, indicates the long-term background motion associated with a pixel. This method requires offline training and cannot add new code words to the codebook at runtime.

ViBe [33] is a technique enabling BS at a very high speed. The background model for each pixel includes some of the previous pixels at the given pixel location as well as from the neighboring locations. A pixel identified as background replaces one of the randomly selected background pixels for the corresponding and the neighboring pixel locations. The rate of update is controlled by a fixed parameter called the sampling factor. Despite its advantages, the BS performance of ViBe is unsatisfactory in challenging scenarios such as dark backgrounds, shadows, and under frequent background changes [34].

PBAS [35] is another BS scheme that only maintains background samples in the background model. PBAS has a similar set of parameters as ViBe. It operates on the three independent color channels separately, and combines their results for foreground classification. Another sample based scheme is SACON [36]. This method computes a sample consensus by considering a large number of previous pixels, similar to ViBe. Authors assign time out map values to pixels and objects, which indicate for how long a pixel has been observed in consecutive frames. These values are used to add static foreground objects to the background.

Recently, several BS schemes based on human perception have been proposed. In [37], authors assumed that human vision does not visualize the scene globally but is rather focused on key-points in a given scene. Their proposed method uses key-point detection and matching for maintaining the background model. Saliency has been used in [38] for developing a BS technique. In [34], authors consider how the human visual system perceives noticeable intensity deviation from the background. Authors in [39] present a method of BS for cell-phone cameras under view angle changes.

2.2 *Hardware Implementation of Background Subtraction Schemes*

Some implementations of BS algorithms for constrained environments have been proposed in the past. The BS algorithm presented in [40] has been implemented on a Spartan 3 FPGA in [41]. Details of the implementation are missing in their work. Furthermore, [40] assumes the background to be static, which is impractical. In [42], authors present a modification to the method proposed in [43]. They have achieved significant gains in memory consumption and execution time; however, they have not presented the BS performance results over a complete dataset. Authors in [44] implement the algorithm presented in [45] on the Spartan 3 FPGA. The implemented algorithm, however, is non-adaptive and applicable to static backgrounds only. Furthermore, the algorithm of [45] lacks quantitative evaluation. An implementation of single Gaussian-based BS on a Digilent FPGA is given in [46]. As [45], the implemented algorithm cannot model dynamic backgrounds. Another implementation of a BS scheme for static backgrounds, more specifically the selective running Gaussian filter-based BS, has been performed on a Virtex 4

FPGA in [47]. In [48], authors present a modified multi-modal *Σ*-*Δ* BS scheme. They have achieved very high speed with their implementation on a Spartan II FPGA. Like other methods discussed above, they have not evaluated the BS performance of their method on a standard dataset.

Many researchers have implemented comparatively better performing BS schemes as well. A SoC implementation of GMM is presented in [49], which consumes 421 mW. In [50], an FPGA implementation of GMM is presented, which is faster and requires less energy compared to previous implementations. The authors maneuver the update equations to simplify the hardware implementations. Other implementations of GMM include [51] and [52]. An implementation of the codebook algorithm for BS is presented in [53]. Similarly, FPGA implementations of ViBe and PBAS algorithms have presented in [54] and [55], respectively.

It should be noted that the above implementations do not exactly implement the original algorithms but use a different set of parameters or post-processing to adapt their methods for better hardware implementations; therefore, the BS performance of these implementations is expected to be different from the original algorithms.

3 Review of EBSCam

EBSCam is a BS scheme proposed in [4]. It uses a background model that is robust against the effect of noise. In this work, we use EBSCam to estimate the parameters of the PMM distribution for every pixel.

It is shown in [4] that the noise in the input samples results in the background model of each pixel to fluctuate. This results in BS scheme making classification errors, i.e., identifying background pixels as foreground and vice versa. These errors are typically defined in terms of false positive and false negatives. A false positive is said to occur if a pixel belonging to the background is identified to be part of a moving object. Similarly, a false negative is said to occur if a pixel belonging to the moving object is identified as to be part of the background. In [4], the probability of false positives and false negatives with GMM-based BS is shown to be

$$\mathrm{P}\left[FP\right] = 1 - \mathrm{erf}\left(\frac{\sqrt{T}\sigma_{BG}}{\sqrt{2\left(\sigma_{BG}^2 + s_{\mu,k}^2\left(\alpha_k,\sigma_{BG}\right) + \psi + \sqrt{T}s_{\sigma,k}^2\left(\alpha_k,\sigma_{BG}\right)\right)}}\right) \tag{1}$$

and

$$\mathrm{P}\left[FN\right] = 1 - \mathrm{erf}\left(\frac{\mathrm{E}\left[I_{FG}\right] - \left(\mathrm{E}\left[I_{BG}\right] + \sqrt{T}\sigma_{BG}\right)}{\sqrt{2\left(\sigma_{FG}^2 + s_{\mu,k}^2\left(\alpha_k,\sigma_{BG}\right) + \sqrt{T}s_{\sigma,k}^2\left(\alpha_k,\sigma_{BG}\right)\right)}}\right), \tag{2}$$

respectively. Note that here $s^2_{\mu,k}$ and $s^2_{\sigma,k}$ denote the variance in the estimated mean and standard deviation parameters of GMM, respectively. Kindly, refer to [4] for the definition of the rest of the symbols as these are not relevant here. From the above equations it is seen that the variance in the estimated parameters increases the error probabilities. EBSCam mitigates this variance in the estimated parameters to reduce errors in BS.

In EBSCam, the intensity of the background of each pixel is limited to have K different intensities at maximum. In other words, the background can have K different layers to allow the method to estimate dynamic backgrounds. For every pixel i, a set $\boldsymbol{E_i}$ of K elements is formed. Each element of $\boldsymbol{E_i}$ represents a single layer of the background.

The pixel intensity at i-th pixel is compared against the elements of the set $\boldsymbol{E_i}$ to populate $\boldsymbol{E_i}$. If the pixel intensity differs from all the elements of the set $\boldsymbol{E_i}$ by more than a constant D, then it is stored as a new element in the set $\boldsymbol{E_i}$. Let us define

$$R_{i,t} = \cup_{j=1}^{K} [E^{(j)}_{i,t-1} - D, E^{(j)}_{i,t-1} + D], \tag{3}$$

where $E^{(j)}_{i,t-1}$ is the j-th element of $\boldsymbol{E_i}$ at time $t-1$ and D is a global threshold, then the sampling frame (if the input intensity belongs to $S_{i,t}$ then it is included in $\boldsymbol{E_i}$) at time t is defined as

$$S_{i,t} = \boldsymbol{I} \setminus R_{i,t}. \tag{4}$$

Here $\boldsymbol{I}$ is the set of all possible values of background pixels and $\setminus$ denotes set subtraction.

It is seen from the above equations that the background model only changes when the input intensity differs from all the elements of $\boldsymbol{E_i}$ by more than D, otherwise, the background model does not fluctuate. In other words, triggering the update of the background model is thresholded by a step-size of D in EBSCam. Furthermore, pixel intensities can be used for estimating the background intensity as under the assumption of a normal distribution the mean and the mode are the same. In other words, the most frequently observed value of the background intensity is likely to be very close to the mean of the intensity, i.e.,

$$\arg\max_{I_{BG}} f_{I_{BG}}(I_{BG}) = \mathrm{E}[I_{BG}], \tag{5}$$

where $f_{I_{BG}}$ is the probability density function (PDF) of the background intensity.

The background model in EBSCam is not limited to the estimates $\boldsymbol{E_i}$ but also the credence of individual estimates, which is stored in a set $\boldsymbol{C_i}$. The credence gives the confidence in each estimate. In [4], it is shown that the credence should be incremented if an estimate is observed and should be decremented if the respective estimate is not observed. More precisely,

$$C^{(j)}_{i,t} = C^{(j)}_{i,t-1} + 1 \text{ if } I_{i,t} \in [E^{(j)}_{i,t-1} - D, E^{(j)}_{i,t-1} + D] \tag{6}$$

and

$$C_{i,t}^{(j)} = C_{i,t-1}^{(j)} - 1 \text{ if } (I_{i,t} \in [E_{i,t-1}^{(j)} - D, E_{i,t-1}^{(j)} + D]' \wedge C_{i,t-1}^{(j)} > C_{th}), \tag{7}$$

where C_{th} is a constant.

The background model in EBSCam is updated blindly. In blind updates, the input intensity at a pixel updates the background model regardless of the pixel intensity being part of the background or foreground. On the other hand, in non-blind updates only the background pixel intensities are used to update the background model. Whenever a pixel intensity that differs from all the elements of $\boldsymbol{E_i}$ by more than D is observed then it is included as a new estimate. The new estimate has a credence value of zero. Note that the new estimate replaces the estimate about which we are least confident, i.e., the estimate with the minimum credence.

The background model is used to classify pixels to either belonging to the background model (background pixels) or the foreground. The decision to classify a pixel to background or foreground is straightforward. First, the set of estimates about which we are confident enough are defined as

$$B_{i,t} = \cup_{j=1}^{K}(E_{i,t-1}^{(j)} | C_{i,t-1}^{(j)} > C_{th}), \tag{8}$$

i.e., if the credence of an estimate is greater than a threshold then we can consider the estimate to be valid. The pixel intensity is compared against the valid background estimates. If the pixel intensity matches the valid background estimates, then it is considered as part of the background, otherwise, it is considered to be part of the foreground. The foreground mask at a pixel i is obtained as

$$F_{i,t} = \begin{cases} 0 \text{ if } I_{i,t} \in [B_{i,t}^{(m)} - D, B_{i,t}^{(m)} + D] \text{ over any } m \\ 1 \qquad\qquad\qquad \text{otherwise} \end{cases} \tag{9}$$

where $B_{i,t}^{(m)}$ is the m-th element of $B_{i,t}$.

4 EBSCam with Poisson Mixture Model

In EBSCam, a fixed threshold is used for distinguishing the foreground pixels from the background. Although such an approach has been used by numerous authors, such as [33], it lacks theoretical foundation. The variation in the background pixel intensities varies spatially and temporally over video frames; therefore, an adaptive threshold should be used for distinguishing the foreground pixels from the background pixels.

Generally, the Poisson distribution is used to model the shot noise of image sensors. The probability that m photons have been absorbed at a pixel i is given by

$$(X_{i,t} = m) \sim g(m; \lambda_i) = \frac{\lambda_i^m e^{-\lambda_i}}{m!}, \tag{10}$$

where λ_i is a parameter denoting both the mean and the variance of the distribution. Assuming that the number of photons are such that the relationship between the observed intensity and the number of photons is linear, the observed intensity can be given by

$$I_{i,t} = aX_{i,t}, \tag{11}$$

where a is the gain factor. Thus, the mean and the variance of the observed intensity are given by

$$\mu_{i,t} = a\lambda_i \tag{12}$$

and

$$\sigma_{i,t}^2 = a^2\lambda_i = a\mu_{i,t}, \tag{13}$$

respectively.

The Poisson distribution can be used to model the noise or the variance of the background pixel intensities. In order to deal with dynamic backgrounds, we propose using a Poisson mixture model (PMM) for modeling the background intensities. Each subpopulation of the mixture model is representative of a layer of the background. Thus, the background pixel intensity can be modeled as

$$(I_{BG,i,t} = m) \sim \sum_{k=1}^{K} \psi_i^{(k)} g(m; \lambda_i^{(k)}). \tag{14}$$

The parameters of the distribution can be estimated as

$$\lambda_i^{(k)} = \frac{\sum_{t=1}^{T} 1\{I_{BG,i,t} \in k\} I_{BG,i,t}}{\sum_{t=1}^{T} 1\{I_{BG,i,t} \in k\}} \tag{15}$$

and

$$\psi_i^{(k)} = \frac{\sum_{t=1}^{T} 1\{I_{BG,i,t} \in k\}}{T}, \tag{16}$$

where T is the total number of frames. Generally, expectation maximization algorithm is used to estimate the above parameters.

The approach in (15) and (16) for estimating the parameters of the distribution of the background pixel intensities is not feasible for two reasons. First, it requires maintaining all the frames of the video. Second, EM is an iterative procedure and is constrained by speed requirements. Here, we propose using EBSCam for online approximation to estimate the parameters of the PMM. Since the mean and the mode of the Poisson distribution are the same, we can write

$$\lambda_{i,t}^{(k)} = E_{i,t}^{(k)}. \tag{17}$$

Rather than using the normalized values of $\psi_i^{(k)}$ for which $\sum_{k=1}^{K} \psi_i^{(k)} = 1$, we can use non-normalized values by replacing $\psi_i^{(k)}$ of (16) by

$$\phi_i^{(k)} = \sum_{n=1}^{T} 1\{I_{BG,i,n} \in k\}. \tag{18}$$

This approximation is plausible for the reason that the $\psi_i^{(k)}$ values are used for comparison between subpopulations. Since overall k division by T takes place as shown in (16), therefore, the scaling of $\psi_i^{(k)}$ by T will not have an effect on the result of comparison over all k. From (18),

$$\phi_{i,t}^{(k)} = \sum_{n=1}^{t} 1\{I_{BG,i,n} \in k\} = \phi_{i,t-1}^{(k)} + 1\{I_{BG,i,t} \in k\}. \tag{19}$$

In detail, the approximate weight of the k-th subpopulation is incremented if the input intensity matches the k-th subpopulation. Similarly, to get rid of old, unobserved background layers we decrement $\phi_{i,t}^{(k)}$ if $I_{BG,i,t}$ does not belong to the k-th subpopulation. In case a new background layer needs to be stored, the least observed background layer is to be removed. Based on this, the approximate weight $\phi_{i,t}^{(k)}$ can be replaced by $C_{i,t}^{(k)}$.

The decision whether the input intensity matches an estimate is modified as

$$\left(I_{i,t} - E_{i,t-1}^{(j)}\right)^2 < c^2 \left(\sigma_{i,t-1}^{(j)}\right)^2 \tag{20}$$

or

$$\left(I_{i,t} - E_{i,t-1}^{(j)}\right)^2 < c^2 E_{i,t-1}^{(j)} \tag{21}$$

or

$$\left|I_{i,t} - E_{i,t-1}^{(j)}\right|^2 < c\sqrt{E_{i,t-1}^{(j)}}, \tag{22}$$

where c is a constant. Thus, an adaptive threshold is used to distinguish foreground intensities from the background. The threshold is determined by the mean or the variance of PMM, which can be approximated by $\boldsymbol{E_i}$.

Algorithm 1 EBSCam-PMM

```
1: INPUT : $I_t \leftarrow$ video frame at time $t$, $P \leftarrow$ set of all pixels, $K \leftarrow$ cardinality of the estimates
2: OUTPUT : $F_t \leftarrow$ foreground mask at time $t$
3: All elements of $\boldsymbol{E_i}$ and $\boldsymbol{C_i}$ are set to zero for the first frame
4: for $i = 1$ to $P$ do
5:     $F_{i,t} = 1$;
6:     $match = 0$;
7:     for $j = 1$ to $K$ do
8:         $diff = |I_{i,t} - E_{i,t}^{(j)}|$;
9:         if $diff < c\sqrt{E_{i,t}^{(j)}}$ then                ▷ Estimate is observed
10:            $C_{i,t}^{(j)} = C_{i,t-1}^{(j)} + 1$;
11:            $match = 1$;
12:            if $C_{i,t-1}^{(j)} > C_{th}$ then
13:                $F_{i,t} = 0$;
14:            end if
15:        else                                                  ▷ Estimate is not observed
16:            if $C_{i,t-1}^{(j)} > C_{th}$ then
17:                $C_{i,t}^{(j)} = C_{i,t-1}^{(j)} - 1$;
18:            end if
19:        end if
20:    end for
21:    if $match == 0$ then
22:        Find $m$ such that $C_{i,t}^{(m)} < C_{i,t}^{(j)}$ over $j = 1 : K$
23:        $E_{i,t}^{(m)} = I_{i,t}$;                            ▷ Initializing the estimate
24:        $C_{i,t}^{(m)} = 0$;
25:    end if
26: end for
```

The update of $\boldsymbol{E_i}$ and foreground classification is performed similar to EBSCam. Since, the proposed method uses EBSCam to estimate the distribution parameters of PMM, we term the proposed scheme as EBSCam-PMM. A step-wise procedure is shown in Algorithm 1.

5 Parameter Selection

In EBCam-PMM, there are two parameters namely c and K. If a too large value of K is used, then it will cover more than the background model, resulting in false negatives. Similarly, a too small value of K will result in false positives, as background intensities will not be fully covered by the estimated mixture model. For c we propose using a value of 2, i.e., the intensity at a pixel is said to match a background estimate if it is within two standard deviations of the estimate. We choose $c = 2$ for a couple of reasons. First, two standard deviations sufficiently cover the distribution. Second, assuming integer values of the input intensity, we can write the condition in (22) as

$$\left|I_{i,t} - E_{i,t-1}^{(j)}\right| < \left\lfloor c\sqrt{E_{i,t-1}^{(j)}}\right\rfloor. \tag{23}$$

With a non-negative integer c, the above can be written as

$$\left|I_{i,t} - E_{i,t-1}^{(j)}\right| < c\left\lfloor \sqrt{E_{i,t-1}^{(j)}}\right\rfloor. \tag{24}$$

By using (24), we can replace the complicated square-root operation by simple LUTs. For example, for all values of $E_{i,t}^{(j)}$ between 170 and 224, the value of $\left\lfloor \sqrt{E_{i,t-1}^{(j)}}\right\rfloor$ is 14. Generally, for an n-bit wide input intensity, we require only $\lfloor 2^{\frac{n}{2}} \rfloor$ if-else conditions to compute $\left\lfloor \sqrt{E_{i,t-1}^{(j)}}\right\rfloor$.

We applied EBSCam-PMM to the dataset in [56] with different values of K. In Fig. 1, we show the percent of wrong classification (PWC) defined as

$$\text{PWC} = \frac{FP + FN}{TP + TN + FP + FN} \times 100. \tag{25}$$

Here TP are true positives and TN are true negatives. With EBSCam-PMM, optimal performance is achieved with $K = 5$ as seen in Fig. 1.

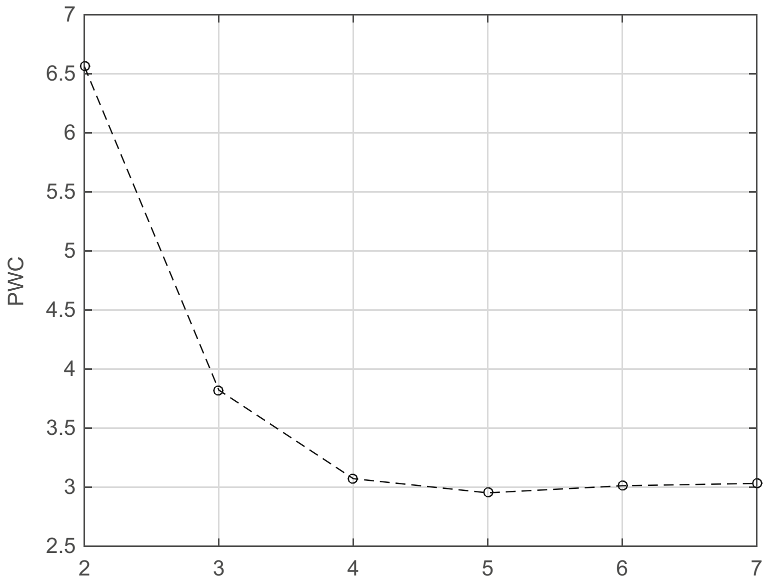

Fig. 1 The effect of changing K on background subtraction performance of EBSCam-PMM over CDNET-2014 dataset

6 Hardware Implementation of EBSCam-PMM

In this section, a dedicated implementation of EBSCam-PMM is described as dedicated implementations can attain much higher processing speed and much lower energy compared to a general purpose implementation.

The abstract diagram of the overall system is shown in Fig. 2. The scene is captured by the sensor array. Afterwards, the intensity values of the scene are passed to the image signal processor (ISP). The ISP performs multiple image processing tasks, which include providing the luminosity values which are used in BS. Note that this front-end configuration is not fixed and can be replaced by any system which provides luminosity values of the scene.

The EBSCam-PMM engine performs the task of identifying the foreground pixels from the background. The EBSCam-PMM engine is composed of the memory unit and the EBSCam-PMM circuit. The memory unit maintains the background model, which is used by the EBSCam-PMM circuit to classify a given pixel into foreground or background.

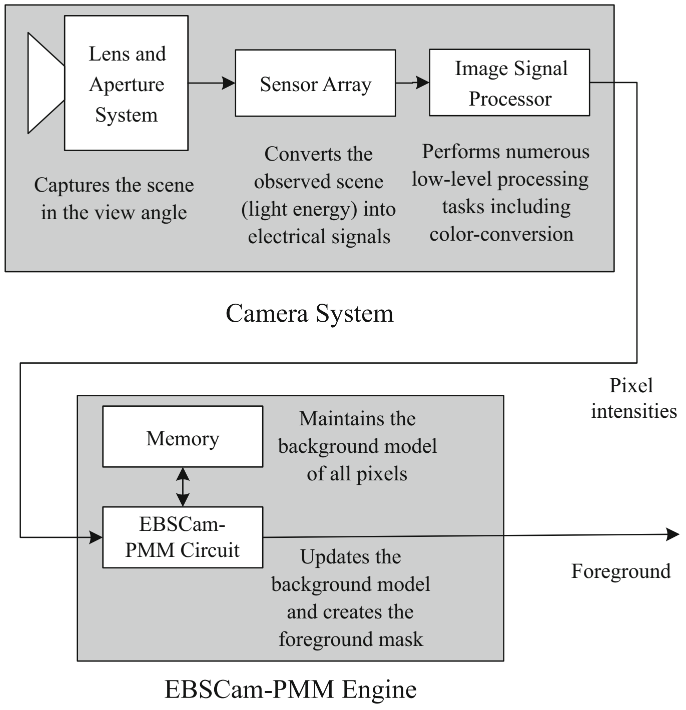
Lens and Aperture System
Sensor Array
Image Signal Processor
Captures the scene in the view angle
Converts the observed scene (light energy) into electrical signals
Performs numerous low-level processing tasks including color-conversion
Camera System
Pixel intensities
Memory
Maintains the background model of all pixels
EBSCam-PMM Circuit
Updates the background model and creates the foreground mask
Foreground
EBSCam-PMM Engine

Fig. 2 Abstract diagram of the overall system

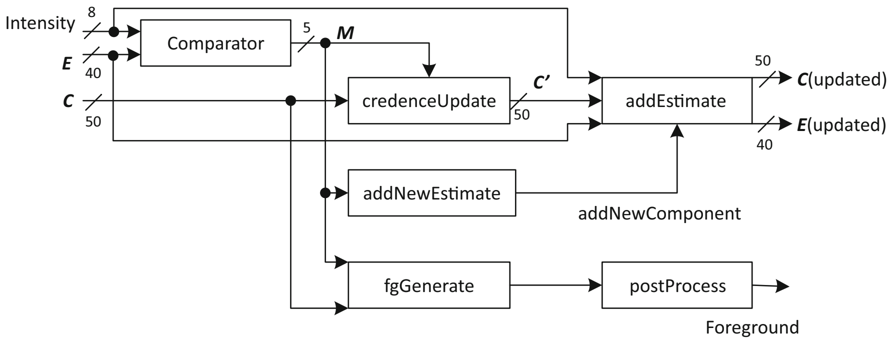

Fig. 3 Block diagram of EBSCam-PMM circuit. Bit-widths are based on FPGA implementation (Sect. 7)

The block diagram of the BS circuit is shown in Fig. 3. This implementation uses 8 and 10 bits for pixel intensities and $C_i^{(j)}$ for all j, respectively. In the graphical representation of each of the constituent modules of EBSCam-PMM circuit, we have excluded the pixel index i as the same circuit is used for all pixels. Similarly, we have excluded the time index from the notation in this section as it can be directly derived from the time index of input intensity. Also, we have not included clock and control signals in the figures to emphasize the main data-flow of the system.

The *comparator* module in the BS circuit compares all the elements of $\boldsymbol{E_i}$ with $I_{i,t}$ in parallel, and generates a K-bit wide output $\boldsymbol{M}$. $M^{(j)} = 1$ indicates that the pixel has matched $E_i^{(j)}$ and vice versa, which is determined by comparing $|I_{i,t} - E_i^{(j)}|$ with $c \left\lfloor \sqrt{E_{(i,t-1)}^{(j)}} \right\rfloor$. The rest of the hardware is the same for EBSCam and EBSCam-PMM. Note that multiple $M^{(j)}$ can be high at the same instant.

A new estimate needs to be added to the background model if $M^{(j)} = 0$ for $j = 1$ to K. The *addNewEstimate* module checks this condition by performing a logical-NOR of all the bits of $\boldsymbol{M}$.

Next, the *credenceUpdate* module updates $C_i^{(j)}$ values based on $M^{(j)}$, i.e., *credenceUpdate* module is an implementation of (6) and (7).

The *addEstimate* module is used to add a new estimate to the background model. A new estimate is added to the background model of a pixel if the output of *addNewEstimate* module is high. The module is further subdivided into two submodules.

The *replaceRequired* submodule determines the index of the estimate which needs to be replaced, and the *replaceEstimate* submodule generates the updated estimates and credence values. If required, the *addEstimate* module replaces the estimate with minimum credence value by the intensity of the pixel. Also, the credence value is initialized to zero. Note that if multiple $C_i^{(j)}$ are minimum at the same time, then the $C_i^{(j)}$ and $E_i^{(j)}$ with smallest j are initialized to zero and $I_{i,t}$, respectively.

The foreground pixel should be high if $M^{(j)} = 0$ and $C_i^{(j)} > C_{th}$ for all j. This task is implemented by the *fgGenerate* block. The output of the *fgGenerate* block is then passed to the *postProcess* block which applies a 7×7 median filter to its input.

7 Experimental Results

To analyze the performance of EBSCam-PMM, we present results of applying EBSCam-PMM to standard datasets. Also, in this section we discuss the FPGA implementation and results of EBSCam-PMM. The performance of the proposed method is compared against some state-of-the-art implementations as well.

7.1 Background Subtraction Performance

To analyze the accuracy of BS under different scene conditions, we have applied EBSCam-PMM to the CDNET-2014 dataset, which is the most thorough BS evaluation dataset available online. CDNET-2014 [56] is an extensive dataset of real-life videos. It includes 53 video sequences divided into 11 categories.

We have used a fixed set of parameters for evaluation of our method. In practice, the value of C_{th} should be varied with the frame rate of the video. However, here we have used a fixed value of C_{th} over the whole dataset. We have used a 7×7 median filter as a post-process. The PWC metric has been used to evaluate the performance, as it is a commonly used performance metric to evaluate and compare binary classifiers such as BS.

In Table 1, we present and compare the performance of EBSCam-PMM with GMM [22], EGMM [29], KDE [57], ViBe [33], PBAS [35], and EBSCam [4]. From Table 1, it is seen that EBSCam-PMM shows improved performance compared to GMM, EGMM, KDE, and ViBe and is only next to PBAS.

To compare the accuracy of dedicated implementations, we show the PWC results of FPGA implementations of ViBe and PBAS algorithms. The results are shown over the CDNET-2012 [58] dataset. It is seen that EBSCam-PMM outperforms most of the methods except for ViBe. However, it will be seen shortly that the hardware complexity of ViBe is much higher compared to the proposed method (Tables 2, 4 and 6).

Table 1 PWC comparison of different methods on CDNET-2014

Category	GMM [22]	EGMM [29]	KDE [57]	ViBe [33]	PBAS [35]	EBSCam [4]	EBSCam-PMM
B. Weather	0.79	0.77	0.72	1.11	0.60	1.19	1.36
Baseline	1.53	1.32	0.55	1.08	1.88	2.05	1.88
C. Jitter	4.22	4.41	5.13	15.34	2.77	3.21	2.53
D. Background	1.21	1.17	1.64	11.57	1.00	1.41	0.86
I. O. Motion	5.19	5.49	10.07	8.75	5.23	5.15	5.53
L. Frame-rate	1.29	1.36	1.31	5.46	1.17	1.64	1.87
N. Videos	4.92	4.72	5.27	4.23	2.03	2.62	2.15
PTZ	14.53	16.94	32.31	35.37	6.15	10.32	5.34
Shadow	2.19	2.19	1.68	1.84	2.00	2.72	2.43
Thermal	4.26	4.30	1.67	2.49	4.73	5.05	5.58
Turbulence	1.27	1.24	1.51	2.12	0.22	2.11	0.74
Overall	3.77	3.99	5.14	8.13	2.53	3.41	2.75

Table 2 PWC comparison of FPGA implementations on CDNET-2012

Category	GMM [50]	ViBe [54]	PBAS [55]	EBSCam [4]	EBSCam- PMM
Baseline	1.88	N.A.	N.A	2.05	1.88
C. Jitter	5.54	N.A.	N.A	3.21	2.53
D. Background	2.19	N.A.	N.A	1.41	0.86
I. O. Motion	6.40	N.A.	N.A	5.15	5.53
Shadow	2.74	N.A.	N.A	2.72	2.43
Thermal	4.58	N.A.	N.A	5.05	5.58
Overall	3.83	1.7	3.43	3.27	3.15

The actual foreground masks generated by different algorithms for a variety of scenes are shown in Table 3. False negatives are observed when the foreground and background intensities are very similar. However, it is seen that generally EBSCam-PMM shows lower number of false positives compared to other methods. It is also seen that PBAS cannot distinguish foreground from the background as seen with the *dynamic background* sequence.

7.2 *FPGA Implementation of EBSCam-PMM*

To analyze the area utilization, power consumption, and speed of EBSCam-PMM for embedded applications such as smart cameras, in this subsection we discuss the FPGA implementation. We also compare the FPGA implementation of the proposed method with other implementations of BS schemes.

Table 3 Foreground masks of CDNET-2014 using EBSCam-PMM

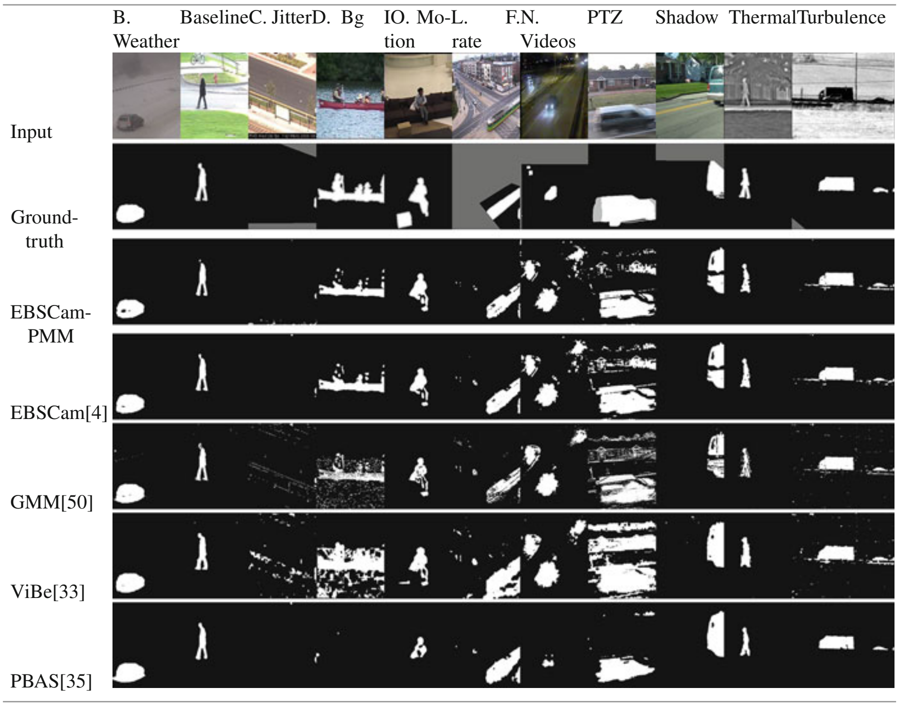

Table 4 Memory bandwidth comparison

Method	Bits per pixel	Bandwidth for 20 HD fps (Gbits/s)
EBSCam-PMM	90	7.26
GMM [50]	99	7.96
GMM [51]	108	8.65
GMM [59]	132	≥2.63 compressed 10.51 uncompressed
ViBe [54]	160	12.67
PBAS [55]	2048	78.09

For synthesis of the circuit, we used XST. ISE was used for place and route on the FPGA. To verify the functionality of the RTL, Xilinx ISE simulation (ISIM) was used. Power estimates were obtained using XPower analyzer.

Due to increasing video resolutions and frame rates, the number of pixels that need to be processed in digital videos is continuously increasing. Thus, the memory bandwidth becomes an important feature of hardware design. Algorithms which require lower memory bandwidths are more suitable to hardware implementations. The number of bits maintained per pixel of the frame in EBSCam-PMM is 90. It is seen in Table 4 that the memory bandwidth of EBSCam-PMM is lower compared to GMM. Compared to ViBe and PBAS, the memory bandwidth of EBSCam-PMM is very small, showing the utility of the proposed method compared to these implementations. In [59], authors have used compression to reduce the memory bandwidth of GMM-based BS. However, compression adds to the circuitry, resulting in increased power consumption, area, and delay.

The implementation results of EBSCam-PMM engine on an FPGA are shown in Table 5. The background model is stored in the internal SRAM of FPGA. From these results, it is seen that EBSCam-PMM requires low area, has high speed, and consumes low power. As an example, for SVGA sequences EBSCam-PMM consumes 1.192 W at 30 fps. Also, a frame rate of approximately 96 can be achieved with EBSCam-PMM at SVGA resolution.

To compare the performance of the proposed method against previous art, we show the implementation results of our method and other methods in Table 6. It is seen that EBSCam-PMM requires significantly lower power compared to GMM. This comes as no surprise as EBSCam-PMM uses parameters that do not fluctuate rapidly, resulting in lower switching activity and, thus, lower power consumption. The logic resources of the FPGA used by EBSCam-PMM are also lower and EBSCam-PMM can achieve higher speeds compared to GMM. The logic requirement and speed of EBSCam-PMM are significantly better than ViBe and PBAS. In fact, the power consumption of EBSCam-PMM is almost negligible compared to that of PBAS. Note that EBSCam-PMM only requires slightly more logic resources compared to EBSCam, with almost similar speed and power consumption.

Recently, [61] proposed a method to reduce the memory bandwidth of GMM. From Table 7, it is seen that their method achieves a lower memory bandwidth compared to EBSCam-PMM. However, it is seen that the speed and power consumption of our implementation are much lower compared to [61]. Also, note that their method can at best achieve the accuracy of GMM, whereas it is seen earlier that EBSCam-PMM outperforms GMM in BS accuracy.

8 Conclusion

This chapter presented a new background subtraction scheme based on Poisson mixture models called EBSCam-PMM. Since shot noise is the most dominant form of noise in natural images, it is natural to use a Poisson mixture model to model the background intensity. A sequential approach to estimating the parameters of the Poisson mixture model is also presented. The estimated parameters are more robust against noise in the samples. Resultantly, the proposed method shows superior performance compare to numerous common algorithms. Furthermore, an FPGA implementation of the proposed method is also presented. EBSCam-PMM requires low memory bandwidth and battery power while providing very high frame rates at high resolutions. These features of EBSCam-PMM make it a suitable candidate for smart cameras.

Table 5 EBSCam-PMM engine on FPGA

Resolution	FPGA	LUT	FF	Slice	DSP Slice	BRAM	Frequency (MHz)	Max. fps Max. fps	Power at 30 fps (W)	Dynamic Power at 30 fps (W)
SVGA	Virtex7 xc7vx690t	837/433200	106/866400	466/108300	0/2880	1350/1470	46.34	96	1.192	0.849
VGA	Virtex7 xc7vx485t	733/303600	106/607200	439/75900	0/2800	901/1030	54.32	176	0.461	0.237
QVGA	Artix7 xc7ac200t	715/134600	105/269200	395/33560	0/740	225/365	53.48	696	0.107	0.029

Table 6 EBSCam-PMM BS circuit and comparison with previous art

FPGA	Method	Circuit	LUT	FF	Slice	DSP Slice	BRAM	Frequency (MHz)	Energy per pixel (nJ)
Virtex6 xc6vlx75t	EBSCam-PMM	Prop.	451/46560	136/93120	209/11640	0/288	1/156	270.1	0.17
	EBSCam	Ref. [4]	446/46560	120/93120	206/11640	0/288	1/156	271.3	0.14
	GMM	Ref. [50]	788/46560	363/93120	349/11640	3/288	0/156	189.3	0.72
Virtex5 xc5vlx50	EBSCam-PMM	Prop.	537/28800	119/28800	257/7200	0/48	1/48	314	0.19
	EBSCam	Ref. [4]	531/28800	111/28800	253/7200	0/48	1/48	320	0.18
	GMM	Ref. [50]	724/28800	223/28800	323/7200	3/48	3/48	130.9	1.02
		Ref. [52]	1572/28800	0/28800	N. A.	N. A.	0/48	47	5.87
		Ref. [60]	1066/28800	0/28800	346/7200	10/48	0/48	50.5	4.60
Virtex7 xc7vlx240t	EBSCam-PMM	Prop.	448/150720	128/301440	218/37680	0/768	1/832	301	0.15
	EBSCam	Ref. [4]	445/150720	120/301440	213/37680	0/768	1/832	306	0.13
	ViBe	Ref. [54]	9278/150720	12571/301440	N. A.	13/768	172/832	140	N. A.
Virtex7 xc7vlx240t	EBSCam-PMM	Prop.	532/303600	121/607200	258/75900	0/2800	1/1030	281	0.11
	EBSCam	Ref. [4]	526/303600	120/607200	254/75900	0/2800	1/1030	284	0.09
	PBAS	Ref. [55]	36060/303600	39108/607200	14903/75900	12/2800	248/1030	116	206.3

Table 7 Comparison of CoSCS-MoG and EBSCam-PMM

Method	Design	Delay/frame (ms)	Bandwidth (Gbits/s) at 20 HD fps	Energy per frame (mJ)
CoSCS-MoG	Ref. [61]	65.3	1.68	125.96
EBSCam-PMM	Prop.	1.82×10^{-5}	7.26	3.4

References

1. McIvor AM (2000) Background subtraction techniques. Proc Image Vision Comput 4:3099–3104
2. Khan MUK, Kyung CM, Yahya KM (2013) Optimized learning rate for energy waste minimization in a background subtraction based surveillance system. In: IEEE int. symp. on circuits and sys. (ISCAS). IEEE, Piscataway, pp 2355–2360
3. Khan MUK, Khan A, Kyung C-M (2014) Dual frame rate motion detection for memory-and energy-constrained surveillance systems. In: IEEE int. conf. on adv. video and signal based surveillance (AVSS). IEEE, Piscataway, pp 81–86
4. Khan MUK, Khan A, Kyung C-M (2017) Ebscam: background subtraction for ubiquitous computing. IEEE Trans Very Large Scale Integra VLSI Syst 25(1):35–47
5. Piccardi M (2004) Background subtraction techniques: a review. In: 2004 IEEE international conference on systems, man and cybernetics, vol 4. IEEE, Piscataway, pp 3099–3104
6. Kristensen F, Hedberg H, Jiang H, Nilsson P, Öwall V (2008) An embedded real-time surveillance system: implementation and evaluation. J Signal Process Syst 52(1):75–94
7. Radke RJ, Andra S, Al-Kofahi O, Roysam B (2005) Image change detection algorithms: a systematic survey. IEEE Trans Image Process 14(3):294–307
8. Seki M, Wada T, Fujiwara H, Sumi K (2003) Background subtraction based on cooccurrence of image variations. In: Proceedings 2003 IEEE computer society conference on computer vision and pattern recognition, 2003, vol 2. IEEE, Piscataway, pp II–II
9. Power PW, Schoonees JA (2002) Understanding background mixture models for foreground segmentation. In: Proceedings image and vision computing, New Zealand, vol 2002, pp 10–11
10. Tsai D-M, Lai S-C (2009) Independent component analysis-based background subtraction for indoor surveillance. IEEE Trans Image Process 18(1):158–167
11. Lin H-H, Liu T-L, Chuang J-H (2009) Learning a scene background model via classification. IEEE Trans Signal Process 57(5):1641–1654
12. Sivabalakrishnan M, Manjula D (2009) An efficient foreground detection algorithm for visual surveillance system. Int J Comput Sci Netw Secur 9(5):221–227
13. Jacques J, Jung CR, Musse SR (2006) A background subtraction model adapted to illumination changes. In: Proc. IEEE int. conf. on image processing, pp 1817–1820
14. Davis JW, Sharma V (2004) Robust background-subtraction for person detection in thermal imagery. In: IEEE int. workshop on object tracking and classification beyond the visible spectrum
15. Wren CR, Azarbayejani A, Darrell T, Pentland AP (1997) Pfinder: real-time tracking of the human body. IEEE Trans Pattern Anal Mach Intell 19(7):780–785
16. Manzanera A, Richefeu JC (2007) A new motion detection algorithm based on $\sigma - \delta$ background estimation. Pattern Recogn Lett 28(3):320–328
17. Manzanera A (2007) σ-δ background subtraction and the Zipf law. In: Progress in pattern recognition, image analysis and applications, pp 42–51
18. Lacassagne L, Manzanera A, Denoulet J, Mérigot A (2009) High performance motion detection: some trends toward new embedded architectures for vision systems. J Real-Time Image Process 4(2):127–146
19. Lacassagne L, Manzanera A (2009) Motion detection: fast and robust algorithms for embedded systems. In: Proc. IEEE int. conf. on image processing

20. Elgammal A, Harwood D, Davis L (2000) Non-parametric model for background subtraction. In: Computer VisionECCV 2000, pp 751–767
21. Tavakkoli A, Nicolescu M, Bebis G, Nicolescu M (2009) Non-parametric statistical background modeling for efficient foreground region detection. Mach Vis Appl 20(6):395–409
22. Stauffer C, Grimson WEL (1999) Adaptive background mixture models for real-time tracking. In: IEEE conf. on comp. vis. and pattern recog., vol 2. IEEE, Piscataway
23. White B, Shah M (2007) Automatically tuning background subtraction parameters using particle swarm optimization. In: 2007 IEEE international conference on multimedia and expo. IEEE, Piscataway, pp 1826–1829
24. Atrey PK, Kumar V, Kumar A, Kankanhalli MS (2006) Experiential sampling based foreground/background segmentation for video surveillance. In: 2006 IEEE international conference on multimedia and expo. IEEE, Piscataway, pp 1809–1812
25. Park J, Tabb A, Kak AC (2006) Hierarchical data structure for real-time background subtraction. In: 2006 IEEE international conference on image processing. IEEE, Piscataway, 1849–1852
26. Porikli F, Tuzel O (2003) Human body tracking by adaptive background models and mean-shift analysis. In: IEEE international workshop on performance evaluation of tracking and surveillance, pp 1–9
27. Yang S-Y, Hsu C-T (2006) Background modeling from GMM likelihood combined with spatial and color coherency. In: 2006 IEEE international conference on image processing. IEEE, Piscataway, pp 2801–2804
28. Lee D-S (2005) Effective gaussian mixture learning for video background subtraction. IEEE Trans Pattern Anal Mach Intell 27(5):827–832
29. Zivkovic Z (2004) Improved adaptive gaussian mixture model for background subtraction. In: Proc. IEEE int. conf. on pattern recognition, vol 2, pp 28–31
30. Wang R, Bunyak F, Seetharaman G, Palaniappan K (2014) Static and moving object detection using flux tensor with split gaussian models. In: Proc. IEEE conf. on computer vision and pattern recognition Wkshps., pp 414–418
31. Kim K, Chalidabhongse TH, Harwood D, Davis L (2005) Real-time foreground–background segmentation using codebook model. Real-Time Imaging 11(3):172–185
32. Sigari MH, Fathy M (2008) Real-time background modeling/subtraction using two-layer codebook model. In: Proceedings of the international multiconference of engineers and computer scientists, vol 1
33. Barnich O, Van Droogenbroeck M (2011) Vibe: a universal background subtraction algorithm for video sequences. IEEE Trans Image Process 20(6):1709–1724
34. Haque M, Murshed M (2013) Perception-inspired background subtraction. IEEE Trans Circuits Syst Video Technol 23(12):2127–2140
35. Hofmann M, Tiefenbacher P, Rigoll G (2012) Background segmentation with feedback: the pixel-based adaptive segmenter. In: Proc. IEEE conf. on computer vision and pattern recognition wkshps. IEEE, Piscataway, pp 38–43
36. Wang H, Suter D (2006) Background subtraction based on a robust consensus method. In: 18th international conference on pattern recognition, 2006. ICPR 2006, vol 1. IEEE, Piscataway, pp 223–226
37. Guillot C, Taron M, Sayd P, Pham QC, Tilmant C, Lavest J-M (2010) Background subtraction adapted to PTZ cameras by keypoint density estimation. In: Proceedings of the British machine vision conference, p 34-1
38. Tavakoli HR, Rahtu E, Heikkilä J (2012) Temporal saliency for fast motion detection. In: Asian conference on computer vision. Springer, Berlin, pp 321–326
39. Hamid R, Sarma AD, DeCoste D, Sundaresan N (2015) Fast approximate matching of videos from hand-held cameras for robust background subtraction. In: 2015 IEEE winter conference on applications of computer vision (WACV). IEEE, Piscataway, pp 294–301
40. Foresti GL (1998) A real-time system for video surveillance of unattended outdoor environments. IEEE Trans Circuits Syst Video Technol 8(6):697–704

41. Kalpana Chowdary M, Suparshya Babu S, Susrutha Babu S, Khan H (2013) FPGA implementation of moving object detection in frames by using background subtraction algorithm. In: 2013 International conference on communications and signal processing (ICCSP). IEEE, Piscataway, pp 1032–1036
42. Lee Y, Kim Y-G, Lee C-H, Kyung C-M (2014) Memory-efficient background subtraction for battery-operated surveillance system. In: The 18th IEEE international symposium on consumer electronics (ISCE 2014). IEEE, Piscataway, pp 1–2
43. Chan W-K, Chien S-Y (2007) Real-time memory-efficient video object segmentation in dynamic background with multi-background registration technique. In: IEEE 9th workshop on multimedia signal processing, 2007. MMSP 2007. IEEE, Piscataway, pp 219–222
44. Gujrathi P, Priya RA, Malathi P (2014) Detecting moving object using background subtraction algorithm in FPGA. In: 2014 Fourth international conference on advances in computing and communications (ICACC). IEEE, Piscataway, pp 117–120
45. Menezes GGS, Silva-Filho AG (2010) Motion detection of vehicles based on FPGA. In: Programmable logic conference (SPL), 2010 VI Southern. IEEE, Piscataway, pp 151–154
46. Cherian S, Senthil Singh C, Manikandan M (2014) Implementation of real time moving object detection using background subtraction in FPGA. In: 2014 International conference on communications and signal processing (ICCSP). IEEE, Piscataway, pp 867–871
47. Hiraiwa J, Vargas E, Toral S (2010) An FPGA based embedded vision system for real-time motion segmentation. In: Proceedings of 17th international conference on systems, signals and image processing, Brazil
48. Abutaleb MM, Hamdy A, Abuelwafa ME, Saad EM (2009) FPGA-based object-extraction based on multimodal σ-δ background estimation. In: 2nd International conference on computer, control and communication, 2009. IC4 2009. IEEE, Piscataway, pp 1–7
49. Tabkhi H, Sabbagh M, Schirner G (2014) A power-efficient FPGA-based mixture-of-Gaussian (MoG) background subtraction for full-HD resolution. In: Proc. annual int. symp. on field-programmable custom computing machines, pp 241–241
50. Genovese M, Napoli E (2014) ASIC and FPGA implementation of the gaussian mixture model algorithm for real-time segmentation of high definition video. IEEE Trans Very Large Scale Integr VLSI Syst 22(3):537–547
51. Genovese M, Napoli E (2013) FPGA-based architecture for real time segmentation and denoising of HD video. J Real-Time Image Process 8(4):389–401
52. Genovese M, Napoli E, Petra N (2010) OpenCV compatible real time processor for background foreground identification. In: Proc. IEEE int. conf. on microelectronics, pp 467–470
53. Rodriguez-Gomez R, Fernandez-Sanchez EJ, Diaz J, Ros E (2015) Codebook hardware implementation on FPGA for background subtraction. J Real-Time Image Process 10(1):43–57
54. Kryjak T, Gorgon M (2013) Real-time implementation of the ViBe foreground object segmentation algorithm. In: Proc. IEEE federated conf. on computer science and information sys., pp 591–596
55. Kryjak T, Komorkiewicz M, Gorgon M (2013) Hardware implementation of the PBAS foreground detection method in FPGA. In: Proc. IEEE int. conf. mixed design of integrated circuits and sys., pp 479–484
56. Wang Y, Jodoin P-M, Porikli F, Konrad J, Benezeth Y, Ishwar P (2014) CDnet 2014: an expanded change detection benchmark dataset. In: Proc. IEEE conf. on computer vision and pattern recognition wkshps., pp 387–394
57. Elgammal A, Duraiswami R, Harwood D, Davis LS (2002) Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. Proc IEEE 90(7):1151–1163
58. Goyette N, Jodoin P-M, Porikli F, Konrad J, Ishwar P (2012) Changedetection. net: a new change detection benchmark dataset. In: Proc. IEEE conf. on computer vision and pattern recognition wkshps., pp 1–8

59. Jiang H, Ardö H, Öwall V (2009) A hardware architecture for real-time video segmentation utilizing memory reduction techniques. IEEE Trans Circuits Syst Video Technol 19(2):226–236
60. Ratnayake K, Amer A (2014) Embedded architecture for noise-adaptive video object detection using parameter-compressed background modeling. J Real-Time Image Process, pp 1–18
61. Shen Y, Hu W, Yang M, Liu J, Wei B, Lucey S, Chou CT (2016) Real-time and robust compressive background subtraction for embedded camera networks. IEEE Trans Mobile Comput 15(2):406–418