A Full-Process Optimization-Based Background Subtraction for Moving Object Detection on General-Purpose Embedded Devices

Shushang Li, Jing Wu, *Senior Member, IEEE*, Chengnian Long, *Senior Member, IEEE*, and Yi-Bing Lin, *Fellow, IEEE*

***Abstract*—Real-time computer vision tasks are emerging in consumer electronics with lightweight computing performance, which are an exquisite design art to balance the computational efficiency and accuracy. In this paper, we present the embedded background subtraction (EBGS) – an optimization algorithm for the entire process to increase computational efficiency and detection accuracy simultaneously. EBGS exploits a simple and efficient Additive Increase Multiplicative Decrease (AIMD) filter to improve the foreground detection accuracy without spending too much time. Moreover, the design combination between the contracted codebook background subtraction (BGS) model and a random model update is proposed to reduce the time consumption. Experiments demonstrate that EBGS can decrease the computing overhead for the three parts of BGS process simultaneously and achieve real-time performance and satisfactory detection accuracy under challenging environments.**

***Index Terms*—Consumer devices, background subtraction, real-time performance, AIMD filtering, adaptive update.**

I. Introduction

DUE TO the increasing awareness of security and home health care, the number of demands for video surveillance systems has increased rapidly. A large number of monitoring equipment are installed indoors and outdoors, such as roads, squares, residences, etc. Huge video flow is transmitted to the sever for processing and analysis [1] (moving object tracking and behavior analysis, etc.), which undoubtedly increases bandwidth and computing burden of the service. Besides, storing all the videos on the server for a long time is neither feasible nor easy to process. Smart consumer devices (such as smart phones, Internet protocol (IP) cameras, single-board computers (SBC), etc.) with built-in general purpose embedded microprocessor are an important part of consumer electronics, which have the characteristics of lightweight computing and multitasking. Using smart consumer devices to preprocess the captured video flow can greatly reduce the transmission bandwidth and the computational burden of the server. Motion detection frame selection based on background subtraction (BGS) is one of the important preprocessing steps in video surveillance applications [2], which can effectively reduce the number of uploaded video frames and the computing overhead of the server. However, the large scale application of smart consumer devices in video surveillance not only promotes the development of BGS, but also poses challenges to algorithms. On one hand, in the outdoors, smart consumer devices may only be powered by batteries, which limits the power consumption of the devices and the computational complexity of the algorithms. On the other hand, a BGS algorithm should be robust to whatever is in its visual field or whatever background change, which may require complex floating point operations. However, the state-of-the-art low power smart consumer devices cannot support complex floating point operations well even with GPU support [3]. Thus, how to design an efficient embedded BGS algorithm is still an open and interesting problem.

In general, the background subtraction consists of three parts: background modelling, post-processing and background update. In background modelling, a large number of studies have been proposed by using statistical models [4] or non-statistical models [5], [6], [7], [8]. Complex modelling approaches were introduced to increase accuracy and robustness, e.g., single Gaussian [3], related non-parametric estimation method [9], spatio-temporal scheme-based [10]. With the help of GPU computing advantage, a complex BGS algorithm [11] are proposed in order to realize better detection accuracy and efficiency. These complex models require high computing and memory due to pixel level operation, which is the main roadblock for real-time BGS on consumer devices (such as SBC and IP cameras), especially low-cost consumer devices. Thus, the recent research attention in embedded computer vision devotes to decreasing the computing consumption with satisfactory accuracy in background modeling process, e.g., the compressive method for MCBS [12], MoG [13] and LW [14].

However, most of the above approaches focus merely on the modelling process while neglect the efficiency of post-processing and model update. Actually, post-processing techniques, such as morphology algorithm and filtering, can significantly improve the foreground segmentation masks attained by a BGS algorithm [15]. Typical morphology

Manuscript received October 30, 2020; revised January 20, 2021, April 3, 2021, and April 26, 2021; accepted April 27, 2021. Date of publication May 4, 2021; date of current version May 25, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 62073215 and Grant 61873166. *(Corresponding authors: Jing Wu and Chengnian Long.)*

Shushang Li, Jing Wu, and Chengnian Long are with the Department of Automation and the Key Laboratory of System Control and Information Processing, Ministry of China, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: lsscumt@163.com; jingwu@sjtu.edu.cn; longcn@sjtu.edu.cn).

Yi-Bing Lin is with the College of Humanities and Sciences, China Medical School, Taichung 404, Taiwan (e-mail: liny@csie.nctu.edu.tw).

Digital Object Identifier 10.1109/TCE.2021.3077241

TABLE I
AVERAGE TIME PER FRAME OF DETECTING AND POST-PROCESSING FOR MEDIAN FILTER AND MORPHOLOGY

	ViBe	CS-MoG	GMM	codebook
modelling	8.0ms	12.4ms	16.8ms	20.6ms
median filter (3×3)	9.0ms	9.3ms	9.5ms	9.2ms
morphology (5×5)	21.8ms	18.6ms	20.2ms	18.0ms

algorithms, such as open and close operations, the computation complexity is $O(nN_s^2)$, where N_s is a side of the structure element, and n is the number of open and close operations. As shown in Table I, on a SBC, the time spent on the post-processing accounts for more than half of the entire process. To attain ideal foreground mask, multiple open and close are operated, which cause serious computational overhead for low-performance general-purpose embedded devices. Besides, update mechanism is beneficial for model to adapt to the change of background. Global update can better maintain the accuracy of the model, but it will generate redundant update and consume a lot of computing resources. To improve update efficiency, ViBe [16] proposed an partial update mechanism of random replacement of samples with fixed frequency, but it cannot effectively deal with dynamic background. In order to overcome the shortcomings of ViBe, SuBSENSE [17] adopted the method of increasing the number of samples, which increases the computational burden. Besides, WeSamBE [18] proposed a weight-based sample update method to improve the accuracy of model but with low efficiency. Because the degree of background change in different areas is different, the partial update method based on a fixed frequency cannot adapt to the background change well. Therefore, it is absolutely imperative to develop fast and efficient post-processing and update mechanism for BGS. Moreover, it is extremely important to consider the trade-off between computational efficiency and detection accuracy.

Based on the above discussion, we propose EBGS, a new embedded background subtraction scheme. With a streamlined design method, EBGS is a systematic organization for background modelling, post-processing, and update, which increases computational efficiency without reducing detection accuracy simultaneously. In summary, the contributions of this paper are shown as follows.

- We propose a system optimization design principle for EBGS, and a full-process optimization scheme is designed according to the design principle to optimize the process of foreground detection, post-processing of initial detection results and model update respectively, which reduces the computation burden without degrading detection accuracy.
- EBGS extends the AIMD filtering to verify its robustness and effectiveness, where we further formally prove that the high accuracy of edge detection with a Markov model for the AIMD method can be obtained.
- We propose a contracted codebook background model, the progressive object detection and block-based adaptive random update mechanism which can effectively reduce computational complexity and memory overhead in the detection and update process simultaneously.

This paper is organized as follows. In Section II, we briefly review some related work. In Section III, we propose the process of constructing the model of background subtraction and the randomly partial update. Experimental results including comparisons with other algorithms are discussed in Section IV. Section V concisely concludes the proposed method.

II. RELATED WORK

There are numerous BGS schemes have been proposed to improve detection precision and the real-time performance. The essential procedure of BGS is to create background model. In this section, we briefly review the recent related work.

Real-time BGS scheme on personal computers: Gaussian Mixture Model (GMM) [20], [21] and Imbalance Compensation Framework [26] are proposed, which can effectively handle background changes such as subtle illumination changes and swaying trees. Nevertheless, the previous model is difficult to deal with the influence from shadows and rapid variations of illumination [4]. As a solution, many modified methods based on GMM are proposed. Zivkovic [22] proposed an adaptive scheme to automatically compute the number of Gaussian models for each pixel, which improves the real-time performance of the algorithm. An improved GMM algorithm based on the literature [22] was employed to model the background to select the correct number of components for each pixel online [23]. Chen *et al.* [24] proposed a method that mixed pixel operation with block operation into a simple framework, which brought a much better robustness to background shaking. To solve the problem of brightness distortion, an algorithm based on brightness distortion and chromaticity distortion of RGB color space and combined with GMM to calculate the threshold of brightness distortion was proposed [25]. Kim *et al.* [27] employed the RGB color background modelling, the morphology and blob-labeling to attain more accurate foreground, but with low efficiency. However, due to intensive computational burden, they cannot run in real-time on low power terminal devices.

Real-time BGS scheme with hardware acceleration: To achieve effective BGS on embedded platforms, many researchers pay attention to special hardware, such as DSP, FPGA and GPU. There also exist some BGS work using GMM on ASIC and FGPA [28], [29]. These algorithms are specifically optimized according to the hardware architectures, which indeed improves the computational efficiency. However, these approaches are based on VHDL design flow, which spends long time in testing and parameter setting of the algorithms. Some researchers consider introducing GPU to relieve the high computational burden caused by complex BGS algorithms. Lightweight non-parametric spatio-temporal-based background modeling methods were proposed, which attains real-time and efficient result [30], [31]. It is hard for a long time running with the limited storage capacity and batteries. Relying on the powerful performance of GPU, BGS based on deep learning were proposed in [32], [33] and [34]. These models based on deep learning are computationally intensive, which consume a lot of computing resources, memory and manual annotation.

In general, such hardware-acceleration based BGS methods heavily depend on the hardware properties and often have poor applicability to current low power consumer devices. The cost

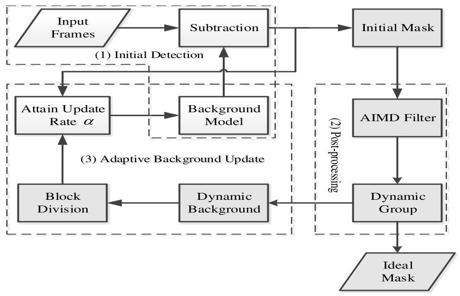


Fig. 1. Flow chart of the EBGS.

of embedded hardware is often higher than the general CPU, and it is difficult to port these typical algorithms to other platforms directly. Besides, it is impossible to perform multitask in such hardware without the assistance of additional processors.

On board BGS scheme on general purpose consumer devices: To reduce the computational overhead, a relatively low-cost BGS method was proposed [35], which uses background sets with image- and color-space reduction and even can be run on mobile devices. Compressive sensing method was introduced to improve the classical GMM [13]. However, this method can not attain a satisfactory detection precision due to the single model per pixel, which is difficult to model the pixel's states. Casares *et al.* [14] presented an objects detection algorithm employing an adaptive background update rate for the embedded smart cameras. Tessens *et al.* [36] addressed a powerful BGS scheme which constructed an approximation of the ideal foreground information by a reduced scan line version of the captured video frames. Both of the above algorithms deal with the entire image after the subtraction operation, which hinders the further improvement of the processing efficiency. ViBe [16] proposed a pixel-level background model and random update, which only costs a small amount of computing resources. However, due to the fixed update rate, ViBe cannot adapt to abrupt change of background. A low-power video surveillance system based on video compression was proposed, which uses motion estimation information to segment the image into foreground and background, but only a coarse-grained foreground can be obtained [37]. Codebook is a popular model for moving object detection [5], [38]. The basic codebook model can provide good BGS accuracy but its computational cost is still high. Two-level or multilayer codebook models [12], [39] were presented to improve the real-time performance at the expense of accuracy. In [39], background subtraction based on image block was employed to improve the detecting speed while neglect accuracy. To solve this problem, Guo *et al.* [12] proposed a novel multi-layer codebook model for extracting foreground information.

Although there have been lots of methods that optimize different segments of BGS to adapt to consumer devices, there is no widely approved method that performs perfectly in both efficiency and accuracy.

III. EBGS Design

EBGS is a full-process operation, which contains background modelling, foreground detection, post-processing and adaptive model update, As shown in Fig. 1.

A. Background Modelling and Foreground Detection

EBGS uses a simple and contracted codebook to structure background model. Each pixel has its own codebook $C = \{c_1, c_2, \dots, c_L\}$. Each codeword, $c_k, k = 1, \dots, L$ contains $\{R, G, B, w, \theta\}$, where R, G and B represent the intensity values of three color channels, g, w and θ represent the gray value, weight and the last updating time, respectively. L is a variate affected by the external environment. In the initial phase, the codebook for each pixel is created with its R, G, B and gray values of the first frame. The weight w is initialized to $1/S$ (S is the number of frames to be trained). Starting from the second frame, the parameters of each codeword in the codebook are compared with their corresponding pixel's *R, G, B* and gray values. If all differences between the parameters of a codeword c_k in the codebook and pixel values are less than the given thresholds, c_k will be updated. For example, at frame t, its weight $\omega_{k,t}$ is updated as:

$$\omega_{k,t} = \omega_{k,t-1} + \frac{1}{S}, \tag{1}$$

Meanwhile, other parameters are updated as:

$$\overline{\mu}_{k,t+1} = (1 - \lambda)\overline{\mu}_{k,t} + \lambda X_t, \tag{2}$$

where λ is the learning ratio and $\overline{\mu}_{k,t}$ denotes the parameters R, G, B and g of the codeword c_k. X_t denotes the four values of a pixel in frame t. Besides, $\theta_{k,t}$ is equal to t.

If any difference between pixel's four values and their corresponding parameters is larger than the threshold, a new codeword will be added to the codebook whose parameters are the R, G, B, g values of the current pixel and initialize its weight, respectively. After all training frames are processed, each codebook may contain many codewords with different weights. To reduce the number of unimportant codewords, we just keep part of the high-weight codewords during the detection stage. Thus the number of codewords in a codebook is determined as follows:

$$L = \arg\min_{n}\left[\left(\sum_{i=1}^{n} \omega_i\right) > 0.7\right]. \tag{3}$$

where 0.7 is the scale parameter that determines how many codewords should be maintained [12].

To reduce memory occupation and computation, the number of codewords in each codebook is limited to no more than 5. In the detection stage, if the background changes suddenly, EBGS can promptly absorb it into the background model according to the following principle. That is, if the foreground does not change greatly in space position within a certain time (such as 20 frames) will be regarded as background. Then a new codeword is added to the codebook if the number of its codewords no more than 4. Otherwise, the earliest updated codeword is replaced by a new codeword.

To reduce computation burden while ensuring accuracy, EBGS adopts the combination of gray-scale and color images to detect foreground region. That is to say, a simple background is detected using the gray-scale value, followed by the complex background (e.g., dynamic background) is detected using the color channels. Note that a large part of the background of the image is stationary. Therefore, when the gray-scale image uses a lower threshold T_L for foreground

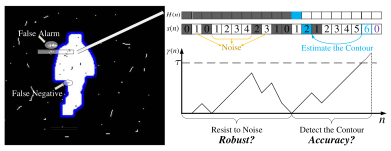


Fig. 2. Use AIMD filter for contour detection.

detection, the detection results are still acceptable. Besides, a higher threshold T_H is added for further detection.

When the gray-scale method does not distinguish foreground from background, we will exploit the color images to make further judgement. The criterion is shown as follows:

$$MAP = \begin{cases} 0, & \left|X_i - \overline{\mu}_i\right| \le T_H \ \ and \ \ i \in (R, G, B) \\ 1, & otherwise. \end{cases} \tag{4}$$

Note that when the three difference values between R, G, B and their corresponding parameters are not greater than T_H, MAP is equal to 0. Based on detecting a large number of different types of sequences, when the T_L and T_H are set to 10 and 17, respectively, the algorithm can obtain better detection results. Based on the above detection method, we can further simplify the update process in Eq. (2) as follows:

$$\overline{\mu}_{k,t+1} = \begin{cases} \overline{\mu}_{k,t} + 1, & if \ T_L < X - \overline{\mu}_{k,t} \le T_H \\ \overline{\mu}_{k,t} - 1, & if \ T_L < \overline{\mu}_{k,t} - X \le T_H \\ \overline{\mu}_{k,t}, & otherwise. \end{cases} \tag{5}$$

B. AIMD Filtering

After an image is processed by the above operation, the initial detection result is mapped onto a binary image. The initial detection results obtained by the simplified codebook model are not accurate enough, as shown on the left-hand side in Fig. 2. We introduce a fast spatial filtering method—Additive Increase Multiplicative Decrease (AIMD) filtering to remove these regions. Each pixel is only operated once with very little computation overhead. AIMD filtering starts from the upper left area and then from left to right, and from top to bottom.

Take detecting a left contour for example, for each line, assume that the accumulated value corresponding to the *n-th* pixel in binary image is $\gamma(n)$, then the accumulated value $\gamma(n+1)$ corresponding to the $(n+1)th$ pixel has the following relationship with its detected value $s(n+1)$ and $\gamma(n)$:

$$\gamma(n+1) = \begin{cases} \gamma(n) + 1, & s(n+1) = 1 \ and \ \gamma(n) < \tau \\ 0, & s(n+1) = 1 \ and \ \gamma(n) = \tau \\ \lfloor \frac{\gamma(n)}{2} \rfloor, & s(n+1) = 0 \end{cases} \tag{6}$$

where τ is a given threshold. If $s(n+1) = 1$ and $\gamma(n) < \tau$, the accumulated value $\gamma(n+1)$ is equal to $\gamma(n)+1$. If $s(n+1) = 0$, the accumulated value $\gamma(n+1)$ is equal to $\lfloor \frac{\gamma(n)}{2} \rfloor$. When $\gamma(n) = \tau$ and $s(n+1) = 1$, the contour is located. The details of AIMD filtering were described in [19], [40]. The right part of Fig. 2 shows the contour detection using AIMD filter. The white, black and blue squares denote the foreground, background and contour pixels, respectively. The values in the second row of the squares are the index values of AIMD filtering $\gamma(n)$. Detecting a series of consecutive foreground pixels and when $\gamma(n)$ is equal to τ, it is determined that a contour point has been detected.

TABLE II
SYMBOLS OF AIMD MODELLING

$H(n)$	The actual value of the nth pixel. Thus $H(n) \in$ {0,1}.
$s(n)$	The detection value of the nth pixel. Thus $s(n) \in$ {0,1}.
p	The false alarm rate. $\forall$ n, $\exists$ $\Pr(s(n) = 1 \vert H(n) = 0) = p, \Pr(s(n) = 0 \vert H(n) = 0) = 1 - p$.
q	The false negative rate, $\forall$ n, $\exists$ $\Pr(s(n) = 0 \vert H(n) = 1) = q, \Pr(s(n) = 1 \vert H(n) = 1) = 1 - q$.
$\gamma(n)$	AIMD index. It additively increases $(\gamma(n-1)+1)$ when a positive signal (s(n) = 1) is detected, otherwise it multiplicative decreases $\lfloor \frac{\gamma(n-1)}{2} \rfloor$.
τ	Threshold for confirming a series of adequate continuous positive detections of signal {s(n)}.

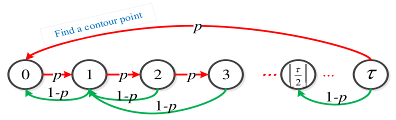


Fig. 3. The transition diagram of the finite Markov chain for the AIMD method.

1) Robustness of AIMD Filtering: The parameters used in the performance analysis are shown in Table II. In our method, we only detect the contour point of the object. We use the following metric $\kappa(\tau, p)$ to evaluate the probability of false contour detection.

$$\kappa(\tau, p) = \Pr(\gamma(n) = \tau | s(n+1) = 1). \tag{7}$$

Intuitively, $\kappa(\tau, p)$ is related to the noise level p and τ. In the following, we give a detailed theoretical analysis of the relationship between them. From Eq. (6), we notice that the next index $\gamma(n+1)$ is only dependent on its current value $\gamma(n)$, that is:

$$\begin{aligned} &\mathbf{Pr}(\gamma(n+1) = x | \gamma(0) = x_0, \gamma(1) = x_1, \ldots, \gamma(n) = x_n) \\ &\quad = \mathbf{Pr}(\gamma(n+1) = x | \gamma(n) = x_n). \end{aligned} \tag{8}$$

Eq. (8) shows that the AIMD method has no post-effect properties, which makes it be modelled by Markov Chain. Specifically, the entire process of the AIMD filtering can be modelled as a homogeneous finite element Markov chain, which contains a total of $\tau + 1$ states. As illustrated in Fig. 3, each state $x_i\{0, \ldots, \tau\}$ represents the current value of γ in AIMD. In the Markov chain, the probability from the state i to the next state j can be expressed as:

$$\mathbf{P}_{i,j} = \mathbf{Pr}(\gamma(n+1) = j | \gamma(n) = i). \tag{9}$$

The transfer probability matrix $\mathbf{P}_{(\tau+1)\times(\tau+1)}$ of the entire process can be obtained by Eq. (6) and Eq. (8), where the entries $\mathbf{P}_{i,j}$ satisfy the following relationship.

$$\mathbf{P}_{i,j} = \begin{cases} p, & j = i + 1 \ or \ i = \tau, j = 0 \\ 1 - p, & j = \lfloor \frac{i}{2} \rfloor \\ 0, & otherwise. \end{cases} \tag{10}$$

For example, given $\tau = 7$, the 8×8 transition matrix $\mathbf{P}_{8\times 8}$ could be written as:

$$\mathbf{P}_{8\times 8} = \begin{bmatrix} 1-p & p & 0 & 0 & 0 & 0 & 0 & 0 \\ 1-p & 0 & p & 0 & 0 & 0 & 0 & 0 \\ 0 & 1-p & 0 & p & 0 & 0 & 0 & 0 \\ 0 & 1-p & 0 & 0 & p & 0 & 0 & 0 \\ 0 & 0 & 1-p & 0 & 0 & p & 0 & 0 \\ 0 & 0 & 1-p & 0 & 0 & 0 & p & 0 \\ 0 & 0 & 0 & 1-p & 0 & 0 & 0 & p \\ p & 0 & 0 & 1-p & 0 & 0 & 0 & 0 \end{bmatrix}$$

Obviously, the Markov Chain is irreducible and aperiodic, which means that there is an unique stationary distribution vector. We denote $\overrightarrow{\pi} = [\pi_0, \pi_1, \ldots, \pi_\tau]$ as the stationary distribution vector, where π_j denotes the steady-state probability of state j [41]. Thus,

$$\overrightarrow{\pi}\mathbf{P}_{(\tau+1)\times(\tau+1)} = \overrightarrow{\pi}, \sum_{j=0}^{\tau} \pi_j = 1, \tag{11}$$

Combining Eq. (10) and Eq. (11), we refer that the entire entries of $\overrightarrow{\pi}$ obey the following relation.

$$\pi_j = \begin{cases} p\pi_\tau + (1-p)(\pi_0 + \pi_1), & j = 0 \\ p\pi_{j-1} + (1-p)\big(\pi_{2j} + \pi_{2j+1}\big), & 0 < j \le \lfloor \frac{\tau}{2} \rfloor \\ p\pi_{j-1}, & \lfloor \frac{\tau}{2} \rfloor < j \le \tau. \end{cases} \tag{12}$$

Obviously, the last half entries (π_j, $\lfloor \frac{\pi}{2} \rfloor < j \le \tau$) are well organized. We pay attention to the relationship of the first half entries.

Lemma 1: The first half entries ($\pi_j, 0 \le j \le \lfloor \frac{\pi}{2} \rfloor$) in $\overrightarrow{\pi}$ obey the following relation.

$$p\pi_j = p\pi_\tau + (1+p)\sum_{k=j+1}^{2j+1} \pi_k, 0 \le j \le \left\lfloor \frac{\tau}{2} \right\rfloor. \tag{13}$$

Compared to Eq. (12), Eq. (13) suggests that each entry π_j (especially for the first half entries) can be represented by other entries in a better way. Nevertheless, the correlation among entries make the relationship between entry π_τ and p, τ obscure. Next, we further determine the range of $\kappa(\tau, p)$.

Lemma 2: The first half entries ($0 \le j \le \lfloor \frac{\tau}{2} \rfloor$) in $\overrightarrow{\pi}$ is constrained by the following inequality.

$$\left(\frac{1 - p^{j+1} + p^{\tau-j}}{p}\right)\pi_{j+1} \le \pi_j \le \frac{\pi_{j+1}}{p + p^{j+2} - p^{j+4}}. \tag{14}$$

According to Eq. (6), Eq. (7) and Eq. (11), the false detection probability of contour point $\kappa(\tau, p)$ can be written as:

$$\kappa(\tau, p) = p\pi_\tau. \tag{15}$$

Theorem 1: Assume that the false detection probability of the contours is $\kappa(\tau, p)$, the upper and lower bounds of $\kappa(\tau, p)$ are as follows:

$$\frac{p^{\tau+1}(1-p)}{1-p^{\tau+1}} < \kappa(\tau, p) < \frac{p^{\tau+1}(1-p)(1-2p)}{(1-p)^{\lfloor \frac{\tau}{2} \rfloor + 2} + 2p^{\tau+2} - p^{\lfloor \frac{\tau}{2} \rfloor + 2} - p^{\tau+1}}. \tag{16}$$

From **Theorem 1** we can deduce that the probability of false detection is very small when $p \ll 1$.

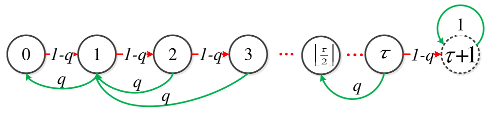
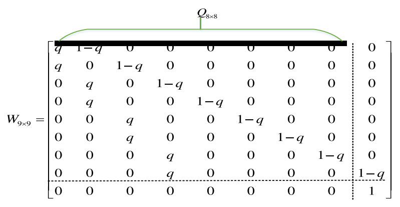


Fig. 4. Absorbing Markov chain.



$$W_{9\times 9} = \left[\begin{array}{cccccccc|c} \overbrace{q \quad 1-q \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0}^{Q_{8\times 8}} & & & & & & & & 0 \\ q & 0 & 1-q & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & q & 0 & 1-q & 0 & 0 & 0 & 0 & 0 \\ 0 & q & 0 & 0 & 1-q & 0 & 0 & 0 & 0 \\ 0 & 0 & q & 0 & 0 & 1-q & 0 & 0 & 0 \\ 0 & 0 & q & 0 & 0 & 0 & 1-q & 0 & 0 \\ 0 & 0 & 0 & q & 0 & 0 & 0 & 1-q & 0 \\ 0 & 0 & 0 & q & 0 & 0 & 0 & 0 & 1-q \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

Fig. 5. Transition matrix of absorbing Markov chain.

2) Accuracy Analysis for Detecting Contour: A contour is located only after a series of consecutive foreground or background pixels are scanned. The location of the a contour point x_{cp} can be attained as follows:

$$x_{cp} = x_d - D, \tag{17}$$

where x_d is the location where a contour point is confirmed to be detected. D is the distance between x_d and x_{cp}, whose value is related to the noise level (q and p) and the threshold τ. Take the example of locating a left contour point, its estimated position is related to q and τ. Eq. (17) can be rewritten as:

$$\widetilde{x_{cp}} = x_d - D(\tau, q). \tag{18}$$

where $\widetilde{x_{cp}}$ is the estimated position of a contour point. In the following, we will use a absorbing Markov Chain to estimate $D(\tau, q)$.

Fig. 4 shows a absorbing Markov Chain for random variable D. This Markov Chain is similar to the one in Fig. 3, but the difference is as: i) a virtual state $\tau + 1$ is added, which is an absorbing state indicating that a contour point is detected; ii) other states are the transient states and would finally evolve to state $\tau + 1$; iii) the transition probability for any two states is based on q.

According to Fig. 4, the transition matrix W for transient states could be conducted. The entire entries of W obey the following recurrence equations:

$$W_{ij} = \begin{cases} q, & j = \lfloor \frac{i}{2} \rfloor \\ 1-q, & j = i+1 \\ 1, & i = j = \tau + 1 \\ 0, & otherwise \end{cases} \tag{19}$$

An example of W under the situation of $\tau = 7$ is shown in Fig. 5. $M_{8\times 8} = (\mathrm{I} - \mathrm{Q}_{8\times 8})^{-1}$ is the fundamental matrix for $W_{9\times 9}$, where I is a identity matrix.

Let ξ_i be the expected time from transition state i to absorbing state. Given that the chain starts in transition state i, and let $\vec{\xi}$ be the column vector whose ith entry is ξ_i. Then

$$\overrightarrow{\xi} = ME \tag{20}$$

where $\overrightarrow{\xi} \triangleq [\xi_0, \xi_1, \ldots, \xi_\tau]^T$, $M = (\mathrm{I} - \mathrm{Q})^{-1}$ and $E \triangleq [1, \ldots, 1]^T$. Obviously, we only need to attain ξ_0 which is the time from state 0 to the absorbing state. Therefore, $D(\tau, q)$ is

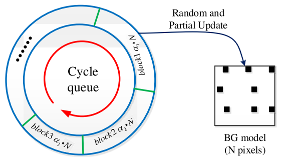

Random and
Partial Update
Cycle
queue
block1 $\alpha_1 \cdot N$
block2 $\alpha_2 \cdot N$
block3 $\alpha_3 \cdot N$
BG model
(N pixels)

Fig. 6. A cycle-queue based scheme for random and partial update at time t.

equal to ξ_0. To obtain $\overrightarrow{\xi}$, we propose an efficient recurrence method for calculating M.

Theorem 2: The entire entries of fundamental matrix M are given as follows:

$$M_{i,j} = \begin{cases} \frac{M_{i,j+1} - \mathrm{I}(i,j+1) - q(M_{i,2j+2} + M_{i,2j+3})}{1-q}, & 0 \le j < \lfloor \frac{\tau}{2} \rfloor \\ \frac{M_{i,j+1} - \mathrm{I}(i,j+1)}{1-q}, & \lfloor \frac{\tau}{2} \rfloor \le j < \tau \\ \frac{1}{1-q}, & j = \tau. \end{cases} \tag{21}$$

The proofs of **Lemma 1**, **Lemma 2**, **Theorem 1** and **Theorem 2** are given in the Appendix. In [19], they show the ROC curve obtained by adjusting the parameters τ and the threshold of background subtraction. Considering the impact of more complex background on the detection, we set τ to 7. In practice, p and q cannot be obtained directly, we propose an indirect estimation method to get them. In this paper, we assume that p is equal to q, and calculate the $M_{i,j}$ by the weighted average false alarm rate $\bar{p}_m$. In the subsection of "Adaptive Background Model Update" shows the detail of calculating $\bar{p}_m$.

After AIMD filtering, the left and right contour points in each line of the foreground can be located. The foreground may contain two or more targets, and each target has its own contour points. To further improve the detection accuracy and remove false contour points, we employ Dynamic Group method [40] to divide the foreground into different moving objects.

C. Update Background Model

In this subsection, we introduce a stochastic partial update scheme to increase update efficiency, which requires less computation than the traditional stochastic schemes.

1) Randomness of Circular-Queue Based Scheme: The interference produced by the external environment in different regions is different. Therefore, it is necessary to divide the image into small blocks (e.g., 16×16) to update their models with different ratios. The right-hand side of Fig. 6 shows an image block, where a small black square denotes a pixel whose model should be updated. We pre-generate a group of random numbers and store them in a reusable container, whose values are treated as the position indices of pixels, as shown on the left-hand side in Fig. 6. In the subsequent process, according to the update ratio α, part of the random numbers in the circular queue are selected to have their corresponding models updated.

When the background model is updated, the left-to-right and up-to-down update are sequentially performed. Meanwhile, the corresponding ending position of the current block in the circular-queue is taken as the starting position of the next block in the circular-queue. Besides, the ending position of the last (bottom right) image block of the frame $t-1$ is considered as the starting position of the first (upper left) image block of frame t in the circular-queue. Assume that each image is divided into K image blocks, the starting position of the m-th block in frame t in the queue can be written as:

$$Start_m(t) = \left[\sum_{i=1}^{t-1} \sum_{j=0}^{K} \alpha(i,j) * B + \sum_{j=0}^{m-1} \alpha(t,j) * B \right] \% C_q. \tag{22}$$

where C_q denotes the size of the circular-queue, % denotes remainder operator, B is a constant, which is equal to the number of pixels in a image block. In this case, only a small circular-queue with the size of C_q (such as $C_q = B$) can meet the demand of randomness.

2) Adaptive Background Model Update: Based on above discussion, we divide the image into many blocks with the size of 16 × 16, and each block has its own update ratio α. The block-based background model update can better adapt to the background change.

We propose a simple linear model to illustrate the effect of update ratio $\alpha_m(t)$ at frame t on the false alarm rate $p_m(t)$ of the m-th block.

$$p_m(t) = k\alpha_m(t), \tag{23}$$

In practice, $p_m(t)$ can not be measured directly. Intuitively, $p_m(t)$ is related to the difference between the current frame pixel parameters and its background model parameters. $p_m(t)$ increases as the difference value increases. Suppose the r-th pixel of the m-th block has a difference value of x, then the false alarm rate is $p_{m,r}(x)$. The relationship between x and $p_{m,r}(x)$ can be described by a Gaussian function. Therefore, we make use of a Gaussian function to fit the relationship between $p_{m,r}(x)$ and x, as:

$$p_{m,r}(x) = 0.1 \cdot e^{-\frac{(x-17)^2}{2 \cdot 10^2}}. \tag{24}$$

Since $\alpha \in [0, 1]$, α attains the maximum only when x is equal to 17, that is $p_{m,r}(x) = 0.1$. The value of k can be deduced from Eq.(23), that is, $k = 0.1$.

$$p_m(t) = 0.1\alpha_m(t). \tag{25}$$

Each block has its own update ratio, and the false alarm ratio is obtained by weighted averaging the false alarm ratio of each pixel in the block. To avoid aggravating computation burden, 10 pixels corresponding to the false alarm rates are selected randomly for the weighted average for each image block. In order to prevent $\alpha_m(t)$ from being too small and affecting the detection result, a lower bound 0.05 is set.

IV. Experimental Evaluation

In this section, we evaluate the effectiveness of the EBGS scheme. All experiments except for WeSamBE were performed on a SBC:a 900MHz quad-core CPU, and 1GB RAM. The EBGS and its comparison algorithms are implemented in C++ with OpenCV libraries. In order to obtain higher accuracy, WeSamBE uses a variety of post-processing method, whose experimental results have to be obtained on a PC.

TABLE III
THE OBJECTIVE EVALUATION DETECTION RESULTS ON CHALLENGING WITH SNOW

Post-processing	No			Yes		
	Re	Pr	F	Re	Pr	F
GMM	0.612	0.849	0.711	0.690	0.880	0.774
CS-MoG	0.795	0.570	0.664	0.809	0.809	0.809
ViBe	0.797	0.830	0.813	0.862	0.832	0.842
codebook	0.796	0.452	0.577	0.791	0.857	0.823
WeSamBE	–	–	–	**0.898**	**0.920**	**0.909**
EBGS	–	–	–	**0.933**	**0.902**	**0.918**

TABLE IV
AVERAGE COMPUTATIONAL EFFICIENCY (*FPS*) WITH THE CHALLENGE OF SNOWING

Post-processing	GMM	CS-MoG	ViBe	codebook	WeSamBE	EBGS
No	35.5	50.5	92.6	40.6	–	–
Yes	14.2	16.3	18.8	15.5	<1	80.7

A. Datasets and Metrics

We collect a number of indoor and outdoor video sets for all experiments. These datasets can be downloaded from [42], [43], [44], [45]. These sequences are of challenging for the state-of-the-art detection schemes. In the evaluation experiments, we compared EBGS with five algorithms, namely, GMM [22], codebook [5], CS-MoG [13], ViBe [16] and WeSamBE [18]. "Recall (*Re*)", "Precision (*Pr*)" and "F-measure (*F*)" are employed as the quantitative metric for performance evaluation, which means that the higher the value is, the better the detection result is. "Recall (*Re*)" and "Precision (*Pr*)" are defined as the proportion of the predicted positive sample in all positive samples and the proportion of the predicted positive samples in all the predicted samples. "F-measure (*F*)" is the harmonic mean of the Precision and Recall. In this paper, we regard foreground pixels as positive samples. The three indexes are calculated as follows:

$$Re = \frac{T_P}{T_P + F_N}, Pr = \frac{T_P}{T_P + F_P}, F = \frac{2 \times Re \times Pr}{Re + Pr}. \quad (26)$$

where T_P, F_N and F_P denote the number of detected true foreground, undetected foreground and false detected foreground, respectively.

B. Performance Comparison

We used challenging video sequences to verify the robustness and accuracy of the algorithms, such as dynamic background, lighting changes, ghost, bootstrapping and camera jitter. According to the literature [15], median filtering and morphological processing are the most practical post-processing methods, and we also give the related test results. The results are given in the form of binary images and tables. We adopt a median window with a window size of 3 × 3, followed by open and close operations using a rectangular window with a size of 5. Since the EBGS is a full-process optimization algorithm, we only provide the post-processed detection results. Besides, WeSamBE includes a lot of complex post-processing, we only provide the post-processed detection results as the reference upper limit value of detection accuracy. In order to better demonstrate the objective evaluation, we bolded the two highest indicators obtained after adding post-processing, as shown in Tables III, V, and VII.

Experiments on challenging sequences with dynamic background: Fig. 7 shows the detection results of six algorithms under two dynamic backgrounds (snow and fountain), and the resolution of two videos are 288×352 and 288×432. As shown in Fig. 7(a), the detection results attained by the codebook and CS-MoG algorithms contain part of misjudged background. Although with a varity of post-processing, WeSamBE cannot get ideal foreground. As shown in Fig. 7(b), when the captured images contain serious and irregular changes dynamic background, the detection result will be seriously affected. Due to the severely dynamic background, when the initial detection attained by CS-MoG is not accurate, the further detection based on pixel-level model is also not accurate. The single gaussian model for each pixel cannot adapt to the severe irregular changes of dynamic background. Thanks to the fast post-processing of the AIMD filtering, EBGS attains more accurate contour points and clean up irregular noise from the background. Tables III and V show the objective evaluation results of the six algorithms in terms of three indicators. As shown in Table III, compared with EBGS, the smaller Recall value of WeSamBE method indicates that it obtained larger false negative and result in a smaller F-measure value despite a higher Precision value. Compared with EBGS, each model of WeSamBE contains more samples and can better adapt to serious dynamic background changes, so it can have a larger Recall value and F-measure value, as shown in Table V. When CS-MoG, codebook and GMM are used to deal with video sequences containing fountains, the obtained foreground contains a lot of false foregrounds due to the inaccuracy of their models, resulting in very low Precision and F-measure value. With the help of post-processing, the Recall, Precision and F-measure value are obviously larger, especially for CS-MoG and codebook. The post-processing based on the combination of median filtering and morphology can effectively remove false foreground so that the detected foreground becomes more pure. The detection performance of EBGS and WeSamBE is very close, and their robustness is better than other algorithms.

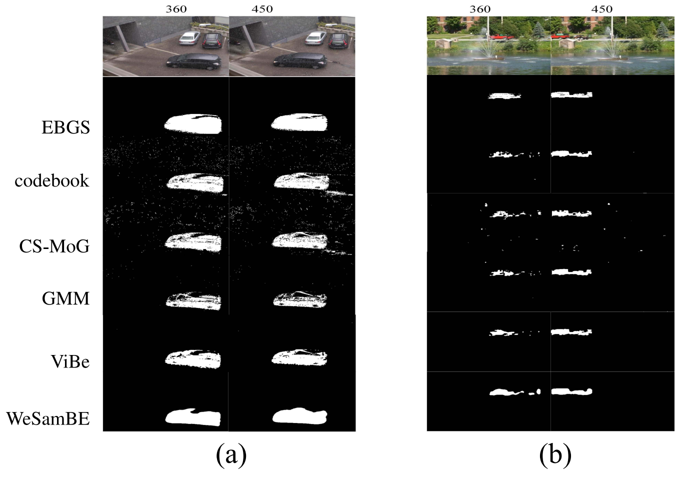


Fig. 7. The detection results on challenging with dynamic background.

TABLE V
THE OBJECTIVE EVALUATION DETECTION RESULTS ON CHALLENGING WITH FOUNTAIN

Post-processing	No			Yes		
	Re	Pr	F	Re	Pr	F
GMM	0.460	0.623	0.529	0.551	0.928	0.692
CS-MoG	0.640	0.169	0.267	0.564	0.505	0.533
ViBe	0.463	0.909	0.614	0.565	0.917	0.699
codebook	0.699	0.137	0.229	0.624	**0.991**	0.706
WeSamBE	–	–	–	**0.778**	**0.959**	**0.858**
EBGS	–	–	–	**0.751**	0.938	**0.833**

TABLE VI
AVERAGE COMPUTATIONAL EFFICIENCY (*FPS*) WITH THE CHALLENGE OF FOUNTAIN

Post-processing	GMM	CS-MoG	ViBe	codebook	WeSamBE	EBGS
No	29.2	43.4	78.4	33.0	–	–
Yes	11.6	13.2	15.5	11.9	<1	55.0

Tables IV and VI show the average computational efficiency under conditions of snowing and fountain.

Experiments on challenging with ghost: To test the detection performance of the above six algorithms, a video sequence with the resolution of 288×352 is used. After the foreground target (a person) enters the camera's acquisition area, he stays still for a period of time (as shown in frame 250) and then leaves (as shown in frame 452). Fig. 8(a) shows two frames and their detection results. When the foreground becomes motionless, EBGS can quickly absorb it into the background model according to the principle that it is regarded as background if the foreground does not change greatly in space position within a certain time (such as 20 frames). When the foreground object is converted from a stationary state to a moving state, it can be promptly detected by EBGS. However, other algorithms were not robust to the influence of the ghost. WeSamBE, codebook and ViBe always misjudged static people as foreground. Besides, owning to the similarity of color between the foreground (white T-shirt) and the background wall, part of foreground is misjudged as the background by the three algorithms. Although the BGS based on GMM and CS-MoG can handle this kind of ghost, there are still some false positive, such as the outline of the human body. The method based on a large-size image block (CS-MoG) can obtain purer background than the method based on a pixel point (GMM). Even in the case of *removing the frames including still object*, the Recall value of EBGS is better than the four comparison algorithms performed on SBC, as shown in Table VII. Besides, the F-measure attained by EBGS exceeds 0.88, which is very close to WeSamBE. WeSamBE obtains a better F-measure based on the premise of removing video sequences containing still object. The Precision value obtained by WeSamBE is only slightly higher than that of EBGS, even not higher than 0.01. After removing the frames including still object GMM achieves a higher Precision value, but its Recall value is still very low. That is, too many foreground pixels are not detected by GMM to be used in high-level applications. Except for WeSamBE, even after adding post-processing, the Precision value of other algorithms is still 0.1 lower than that of EBGS. Table VIII shows the average computational efficiency (FPS) of processing the above video.

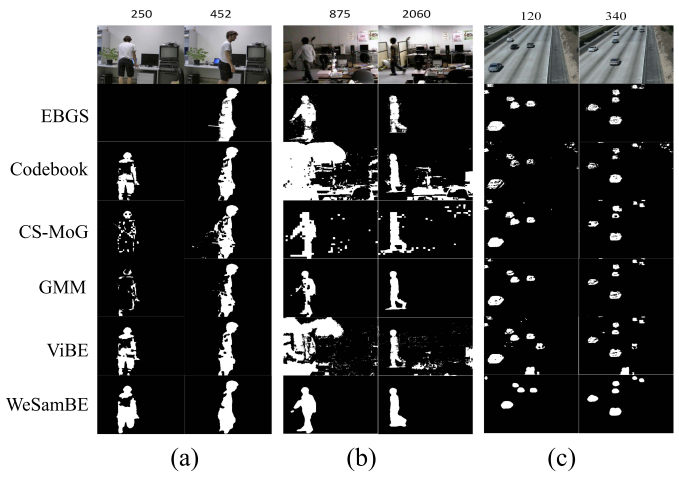


Fig. 8. The detection results on challenging with ghost, light and camera jitter.

TABLE VII
THE OBJECTIVE EVALUATION DETECTION RESULTS ON CHALLENGING WITH THE TRANSITION BETWEEN FOREGROUND AND BACKGROUND

Post-processing	No			Yes		
	Re	Pr	F	Re	Pr	F
GMM	0.544	0.972	0.698	0.576	**0.974**	0.724
CS-MoG	0.664	0.909	0.767	0.706	0.927	0.801
ViBe	0.685	0.972	0.803	0.728	0.964	0.830
codebook	0.719	0.929	0.811	0.722	0.973	0.830
WeSamBE	–	–	–	**0.845**	**0.939**	**0.887**
EBGS	–	–	–	**0.838**	0.929	**0.881**

TABLE VIII
AVERAGE COMPUTATIONAL EFFICIENCY (*FPS*) WITH THE CHALLENGE OF STATE TRANSITION BETWEEN FOREGROUND AND BACKGROUND

Post-processing	GMM	CS-MoG	ViBe	codebook	WeSamBE	EBGS
No	35.0	51.0	89.0	37.4	–	–
Yes	14.2	16.2	18.6	13.6	<1	80.8

Experiments on challenging sequences with light switch on/off: Most of the algorithms are able to adapt to the slow illumination changes. However, many algorithms are difficult to deal with sudden changes in illumination, such as light switch on/off in room. Therefore, it will have a serious impact on the detection of the foreground. Fig. 8(b) shows the results of six algorithms under illumination mutations, and the resolution of the video is 240×320. It is seen that EBGS can finely adapt sudden changes in illumination. Thanks to the fast update/regeneration principle of codewords, EBGS rapidly update or regenerate correct codewords for these pseudo-foreground in a short period of time. WeSamBE is able to attain better foreground through complex post processing but with low efficiency. Even after post-processing the foreground obtained by GMM still has a lot of missed pixels, such as frame 875. Apart from the above three algorithms, other algorithms are relatively sensitive to sudden changes in illumination, especially for codebook and ViBe, which can hardly attain foreground. Table IX shows the average computational efficiency of the six algorithms under the conditions of light switch on/off.

TABLE IX
AVERAGE COMPUTATIONAL EFFICIENCY (*FPS*) WITH THE CHALLENGE OF LIGHT SWITCH ON/OFF

Post-processing	GMM	CS-MoG	ViBe	codebook	WeSamBE	EBGS
No	45.6	51.6	103.2	35.2	–	–
Yes	18.3	16.1	20.7	14.9	<1	89.7

TABLE X
AVERAGE COMPUTATIONAL EFFICIENCY (*FPS*) WITH THE CHALLENGE OF CAMERA JITTER

Post-processing	GMM	CS-MoG	ViBe	codebook	WeSamBE	EBGS
No	46.3	61.3	81.8	36.3	–	–
Yes	18.2	19.8	22.0	15.8	<1	90.2

Experiments on challenging sequences with bootstrapping and camera jitter: Fig. 8(c) shows the detection results in the case of bootstrapping and camera jitter, and the resolution of the video is 240 × 320. Since the initial frame contains the foreground, ViBe attains some false foreground in subsequent detection. Due to the inaccurate initial detection model and the blind random update principle, ViBe cannot achieve fast and accurate model maintenance. Although variety of post-processing are added to WeSamBE, some small foreground is misjudged, such as the frame 120 of Fig. 8(c). In the case of a slight shake of the camera, the foreground detected by the codebook differs significantly from the true foreground, and is still very poor even after post-processing. Compared with other four detection methods, EBGS algorithm obtains the best detection effect. Thanks to the fast update principle, EBGS attains better foreground/background separation. Since the ground truth corresponding to the video sequence was not provided, we do not evaluate the three performance indicators. Table X shows the average computational efficiency of the six algorithms.

Performance evaluation of EBGS with different post-processing: To further compare the performance of the typical post-processing and the lightweight post-processing employed by us, we use the above sequences perform performance evaluation, as shown in Table XI. The weighted average values are added into the last row of Table XI. The performance indicators for the left and right sides are obtained by EBGS using traditional post-processing and a combination of AIMD filtering and Dynamic Group, respectively. As shown in Table XI, when the test sequences do not contain serious challenges, the performance of the two post-processing methods is similar. The Precision value obtained by typical post-processing is slightly but no more than 0.02. However, traditional post-processing cannot remove false foregrounds when subjected to severe dynamic background effects. Severe dynamic background, such as fountain, large area of false foreground will be detected as foreground by contracted codebook. Morphological operations cannot remove such large image blocks, resulting in a very low Precision value. The Dynamic Group can effectively and quickly absorb this kind of false foreground into background model and obtain a higher Precision value. The last row of Table XI shows that the mean of Recall value achieved by typical post-process is only higher than 0.004 than that of “AIMD filtering + Dynamic Group”, but the Precision and F-measure of typical post-process is lower than “AIMD filtering + Dynamic Group” by more than 0.15. Therefore, the performance of “AIMD filtering + Dynamic Group” is significantly better than traditional post-processing methods. As can be seen from aforementioned tables on computational efficiency, computational efficiency will be significantly decreased after the addition of the typical post-processing. The traditional post-processing method is computationally intensive and cannot guarantee real-time performance on low-performance consumer devices, let alone process high-resolution images using multiple post-processing.

TABLE XI
THE OBJECTIVE EVALUATION DETECTION RESULTS OF EBGS WITH DIFFERENT POST-PROCESSING

Post-processing	Typical			AIMD + Dynamic Group		
	Re	Pr	F	Re	Pr	F
Ghost	0.833	**0.948**	**0.887**	**0.838**	0.929	0.881
Snow	**0.940**	0.783	0.854	0.933	**0.902**	**0.918**
Fountain	**0.762**	0.289	0.417	0.751	**0.938**	**0.833**
Average	**0.845**	0.673	0.720	0.841	**0.923**	**0.877**

C. Computational Efficiency Analysis

This subsection provides computational efficiency analysis of the challenging background changes in embedded platforms. Tables IV, VI, VIII, IX, and X indicate that without post-processing, when being employed to process the low-resolution images, except for WeSamBE, the other five comparison algorithms can meet requirement performance. Especially for ViBe, using simple integer operations, when background subtraction are performed on single channel gray-scale image, which can process 90 frames per second or even faster. When background subtraction are performed on color images, the processing speed of ViBe will be severely reduced. Because the computational complexity of ViBe based on color images is three times that based on gray-scale images. In the case of using the same number of color channels, EBGS will achieve higher detection efficiency. Each codebook contains multiple codewords, and a lot of time has to be spent on traversing and maintaining the codewords, which increases the consumption of computing resource and memory. Compared with other embedded algorithms, GMM and codebook have the lower computational efficiency. When GMM is used for foreground detection, each pixel will perform multiple floating-point and exponential operations, which increases the computational overhead. Compared with GMM, CS-MoG uses a combination of block-level and pixel-level models for foreground detection, which effectively reduces the computational complexity.

Background subtraction is the preliminary work of target detection, tracking and other high-level applications. When the obtained detection results contain some false negative and false positive, the detection results are not conducive to subsequent operations. To improve detection accuracy, it is necessary in order to turn to post-processing (such as filtering and morphological processing). For example, morphology algorithms, such as dilation and erosion operations, the computation complexity is $O(nN_s^2)$, where N_s is a side of the structure element, and n is the number of dilation and erosion operations. However, when the archetypal post-processing is added, the computational efficiency seriously drops, and the requirement for real-time processing cannot be convinced. As shown in above tables, the processing speed of these algorithms except EBGS is almost lower than 20 FPS when post-processing is added. The time spent on post-processing accounts for more than two-thirds. That is, post-processing process becomes a major bottleneck for the real-time BGS algorithm.

To obtain a satisfactory detection result, it is necessary to perform morphological treatment such as open and close operations more than once. Nevertheless, existing low-power consumer devices can not provide so high-density computation to realize real-time processing. When the added post-processing operation is constant, assuming the image has a resolution of N_{w*h}, the amount of calculation for processing each image has a linear relationship with its resolution N_{w*h}. As the resolution of the image increases, the computational complexity of the entire processing process will further increase.

EBGS is a full-process computing optimization, which employs contracted codebook, AIMD filtering and random update to realize the trade-off between accuracy and efficiency. The detection accuracy obtained by EBGS is close to that of WeSamBE, and even higher than WeSamBE in some cases. However, WeSamBE has higher computational complexity, and its efficiency is lower than 1 FPS even if it runs on a PC. AIMD filtering is very simple and does not occupy too much computing resource. Each pixel in a binary image only executed once to add 1 or halve the cumulative value. Therefore, it is possible for EBGS to implement speedy processing of the above-mentioned videos. Even when used to process the image with a resolution of 480 × 720, the processing speed can still reach a speed of 20 PFS. On a single-core SBC, the video with a resolution of 240 × 320 can be processed more than 25 FPS.

V. Conclusion

We propose EBGS, a full process computing optimization for entire background subtraction, which can be applied to consumer electronics such as home and business surveillance systems containing smart consumer devices. EBGS is computationally efficient on consumer devices with low computing and storage capabilities. A full-process optimization framework for foreground detection, post processing and background model update is constructed. EBGS uses a simplified codebook that can achieve fast initial foreground extraction. With the help of AIMD filtering and dynamic groups the initial detected foreground can further be refined. In the background model update, EBGS proposes a random partial update strategy based on a micro-sized circular-queue. This scheme achieves fast update without sacrificing the accuracy of the model, which greatly reduces the time consumption of background model update. Finally, a large number of experiments show that EBGS can achieve a balance between detection accuracy and computational efficiency in some complex background environments.

Appendix

A. Proof of Lemma 1

We prove by induction on j. According to Eq. (12), Eq. (13) will hold when $j = 0$. Suppose that the hypothesis Eq. (13) holds for $j = n, n < \lfloor \frac{\tau}{2} \rfloor$.

$$p\pi_n = p\pi_\tau + (1-p)\sum_{k=n+1}^{2n+1} \pi_k, 0 \le n \le \left\lfloor \frac{\tau}{2} \right\rfloor. \quad (27)$$

As stated in Eq. (12): $\pi_{n+1} = p\pi_n + (1-p)(\pi_{2n+2} + \pi_{2n+3}), 0 \le n < \lfloor \frac{\tau}{2} \rfloor$. Then in the inductive step, we prove that when $j = n + 1$, we have

$$\begin{aligned} p\pi_{n+1} &= p(p\pi_n + (1-p)(\pi_{2n+2} + \pi_{2n+3})) \\ &= p\left(p\pi_\tau + (1-p)\left(\sum_{k=n+2}^{2n+3} \pi_k + \pi_{n+1}\right)\right) \\ &= p\pi_\tau + (1+p)\sum_{k=n+2}^{2n+3}. \end{aligned} \quad (28)$$

■

B. Proof of Lemma 2

From the second and third rows in Eq. (12), it can be easily inferred that for all entries of $\overrightarrow{\pi}$, $\pi_{j+1} \ge p\pi_j$. we can get:

$$\pi_{j+k} > p^k \pi_j, 0 \le j \le \tau - k. \quad (29)$$

Substituting Eq. (29) into the third row of Eq. (12), we can obtain:

$$\begin{aligned} \pi_{j+1} \ &\ge \ p\pi_j + (1-p)\left(p^{j+2}\pi_j + p^{j+3}\pi_j\right), 0 \le j < \left\lfloor \frac{\tau}{2} \right\rfloor \\ &\Rightarrow \pi_j \le \frac{1}{p + p^{j+2} - p^{j+4}} \pi_{j+1}, 0 \le j < \left\lfloor \frac{\tau}{2} \right\rfloor. \end{aligned} \quad (30)$$

Furthermore, substituting Eq. (29) into Eq. (27), we have:

$$p\pi_j \ge p^{\tau - j}\pi_{j+1} + (1-p)\sum_{k=0}^{j+1} p^k \pi_{j+1}, 0 \le j \le \left\lfloor \frac{\tau}{2} \right\rfloor.$$

Then,

$$\pi_j \ge \frac{1 - p^{j+1} + p^{\tau - j}}{p} \pi_{j+1}, 0 \le j \le \left\lfloor \frac{\tau}{2} \right\rfloor. \quad (31)$$

Combine Eq. (30) and Eq. (31), ***Lemma 2*** is proved. ■

C. Proof of Theorem 1

Since $\overrightarrow{\pi}$ is a stationary distribution vector of Markov Chain, the sum of its entries equals to 1. We separate the sum into two parts, as follows:

$$\sum_{j=0}^{\tau} \pi_j = \sum_{j=0}^{\lfloor \frac{\tau}{2} \rfloor - 1} \pi_j + \sum_{j=\lfloor \frac{\tau}{2} \rfloor}^{\tau} \pi_j = 1. \quad (32)$$

For the second part of Eq. (32), we find that it is the sum of a geometric progression, whose common ration is defined in the third row of Eq. (12). Consequently, the sum of third part could be induced as follows:

$$\sum_{j=\lfloor\frac{\tau}{2}\rfloor}^{\tau} \pi_j = \sum_{j=\lfloor\frac{\tau}{2}\rfloor}^{\tau} \frac{\pi_\tau}{p^{\tau-j}} = \left[\frac{1-p^{\tau-\lfloor\frac{\tau}{2}\rfloor+1}}{p^{\tau-\lfloor\frac{\tau}{2}\rfloor}(1-p)}\right]\pi_\tau. \quad (33)$$

In a practical situation, p is usually very small in spite of its mathematical constraints $p \ll 1$. For example, if we choose $p = 0.07, p^{\tau-j}$ is at least 14 times smaller than p^{j+1} in the lower bound. So we could slightly simplify the condition of ***Lemma 2:***

$$1-p^{j+1} < 1-p^{j+1}+p^{\tau-j}.$$

Since $p \ll 1$, we could further simplify our inequality through

$$1-p < 1-p^{j+1}. \quad (34)$$

The similar situation also holds in the upper bound, that is:

$$p < p + p^{j+2} + p^{j+4}. \quad (35)$$

For the first part of Eq. (32), since $\pi_{\lfloor\frac{\tau}{2}\rfloor} = \pi_\tau / p^{\tau-\lfloor\frac{\tau}{2}\rfloor}$, when $0 \le j \le \lfloor\frac{\tau}{2}\rfloor$ substitute Eq. (34) and Eq. (35) into Eq. (14), to yield

$$\sum_{j=0}^{\lfloor\frac{\tau}{2}\rfloor-1}\left[\frac{(1-p)^{\lfloor\frac{\tau}{2}\rfloor-j}}{p^{\pi-j}}\right]\pi_\tau < \sum_{j=0}^{\lfloor\frac{\tau}{2}\rfloor-1}\pi_j < \sum_{j=0}^{\lfloor\frac{\tau}{2}\rfloor-1}\frac{\pi_\tau}{p^{\pi-j}}, \Rightarrow$$

$$\frac{(1-p)^{\lfloor\frac{\tau}{2}\rfloor+1}-p^{\lfloor\frac{\tau}{2}\rfloor}+p^{\lfloor\frac{\tau}{2}\rfloor+1}}{p^\tau(1-2p)}\pi_\tau < \sum_{j=0}^{\lfloor\frac{\tau}{2}\rfloor-1}\pi_j < \left[\frac{1-p^{\lfloor\frac{\tau}{2}\rfloor}}{p^\tau(1-p)}\right]\pi_\tau. \quad (36)$$

As the sum of all entries equals to 1, by Eq. (33) and Eq. (36) we have:

$$\frac{p^{\tau+1}(1-p)}{1-p^{\tau+1}} < p\pi_\tau < \frac{p^{\tau+1}(1-p)(1-2p)}{(1-p)^{\lfloor\frac{\tau}{2}\rfloor+2}+2p^{\tau+2}-p^{\lfloor\frac{\tau}{2}\rfloor+2}-p^{\tau+1}}.$$

■

D. Proof of Theorem 2

We use recurrence method to calculate every entry of M.

First, we calculate the entries of the last column. The entries of matrix $(\mathrm{I}-Q)$ can be written as:

$$(\mathrm{I}-Q)_{i,j} = \begin{cases} 1-q, & i=0, j=0 \\ 1, & i=j, j>0 \\ q-1, & i=j-1, j>0 \\ -q, & \lfloor\frac{i}{2}\rfloor = j, 0 < j \le \lfloor\frac{\tau}{2}\rfloor \\ 0, & \textit{otherwise}. \end{cases} \quad (37)$$

According to the definition of fundamental matrix, we have $M * (\mathrm{I}-Q) = \mathrm{I}$, then:

$$\sum_j\sum_k M_{i,k} * (\mathrm{I}-Q)_{k,j} = \sum_j \mathrm{I}_{i,j} = 1, \quad (38)$$

According to Eq. (37), we get $\sum_j (\mathrm{I}-Q)_{i,j} = 0, i \ne \tau$. Apply it into Eq. (38), we have:

$$\sum_j\sum_k M_{i,k} * (\mathrm{I}-Q)_{k,j} = \sum_k M_{i,k}\sum_j(\mathrm{I}-Q)_{k,j}$$

$$= M_{i\tau}\sum_j(\mathrm{I}-Q)_{\tau,j} = 1.$$

According to Eq. (37), we can easily infer that the sum of last row in $(\mathrm{I}-Q)$ is equal to $1-q$, which means $M_{i,\tau} = \frac{1}{1-q}$ for all i.

When $\lfloor\frac{\tau}{2}\rfloor \le j < \tau$, according to $M * (\mathrm{I}-Q) = \mathrm{I}$ and Eq. (37), we have:$M_{i,j}(q-1)+M_{i,j+1} = \mathrm{I}(i, j+1)$.

We can infer that $M_{i,j} = \frac{M_{i,j+1}-\mathrm{I}(i,j+1)}{1-q}$.

Similarly, when $0 \le j < \lfloor\frac{\tau}{2}\rfloor$, we have:

$$M_{i,j}(q-1)+M_{i,j+1}-q\big(M_{i,2j+2}+M_{i,2j+3}\big) = \mathrm{I}(i, j+1),$$

$$\text{Thus, } M_{i,j} = \frac{M_{i,j+1}-\mathrm{I}(i,j+1)-q\big(M_{i,2j+2}+M_{i,2j+3}\big)}{1-q}.$$

■

References

[1] W. Lao, J. Han, and P. H. N. de With, "Automatic video-based human motion analyzer for consumer surveillance system," *IEEE Trans. Consum. Electron.*, vol. 55, no. 2, pp. 591–598, May 2009.

[2] T. T. Zin, P. Tin, H. Hama, and T. Toriu, "Unattended object intelligent analyzer for consumer video surveillance," *IEEE Trans. Consum. Electron.*, vol. 57, no. 2, pp. 549–557, May 2011.

[3] J. Clemons, A. Pelligrini, S. Savarese, and T. M. Austin, "EVA: An efficient vision architecture for mobile systems," in *Proc. Int. Conf. Compilers Architect. Synth. Embedded Syst.*, Sep.–Oct. 2013, pp. 1–10.

[4] C. Stauffer and W. E. L. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 747–757, Aug. 2000.

[5] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis, "Real-time foreground-background segmentation using codebook model," *Real Time Imag.*, vol. 11, no. 3, pp. 172–185, Jun. 2005.

[6] I. Haritaoglu, D. Harwoodand, and L. S. Davis, "W4: Real-time surveillance of people and their activities," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 809–830, Aug. 2000.

[7] X. Zhang, C. Zhu, S. Wang, Y. Liu, and M. Ye, "A Bayesian approach to camouflaged moving object detection," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 9, pp. 2001–2013, Sep. 2017.

[8] C. Cuevas, R. Martinez, n. D. Berj, and N. García, "Detection of stationary foreground objects using multiple nonparametric background-foreground models on a finite state machine," *IEEE Trans. Image Process.*, vol. 26, no. 3, pp. 1127–1142, Mar. 2017.

[9] M. Narayana, A. Hanson, and E. Learned-Miller, "Background modeling using adaptive pixelwise kernel variances in a hybrid feature space," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 2104–2111.

[10] R. Li, S. Yu, and X. Yang, "Efficient spatio-temporal segmentation for extracting moving objects in video sequences," *IEEE Trans. Consum. Electron.*, vol. 53, no. 3, pp. 1161–1167, Aug. 2007.

[11] T. S. F. Haines and T. Xiang, "Background subtraction with dirichletprocess mixture models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 4, pp. 670–683, Apr. 2014.

[12] J.-M. Guo, C.-H. Hsia, Y.-F. Liu, M.-H. Shih, C.-H. Chang, and J.-Y. Wu, "Fast background subtraction based on a multilayer codebook model for moving object detection," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 10, pp. 1809–1821, Oct. 2013.

[13] Y. Shen *et al.*, "Real-time and robust compressive background subtraction for embedded camera networks," *IEEE Trans. Mobile Comput.*, vol. 15, no. 2, pp. 406–418, Feb. 2016.

[14] M. Casares, S. Velipasalar, and A. Pinto, "Light-weight salient foreground detection for embedded smart cameras," *J. Comput. Vis. Image Understand.*, vol. 114, no. 11, pp. 1223–1237, Apr. 2010.

[15] D. H. Parks and S. S. Fels, "Evaluation of background subtraction algorithms with post-processing," in *Proc. IEEE Int. Conf. Adv. Video Signal Based Surveillence*, Dec. 2008, pp. 192–199.

[16] 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, Jun. 2011.

[17] P. L. St-Charles, G. A. Bilodeau, and R. Bergevin, "SuBSENSE: A universal change detection method with local adaptive sensitivity," *IEEE Trans. Image Process.*, vol. 24, no. 1, pp. 359–373, Jan. 2015.

[18] S. Jiang and X. Lu, “WeSamBE: A weight-sample-based method for background subtraction,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 9, pp. 2105–2115, Sep. 2018.

[19] Q. Wang, P. Zhou, J. Wu, and C. Long, “RaFFD: Resource-aware fast foreground detection in embedded smart cameras,” in *Proc. IEEE Global Commun. Conf.*, Apr. 2012, pp. 481–486.

[20] H. Yong, D. Meng, W. Zuo, and L. Zhang, “Robust online matrix factorization for dynamic background subtraction,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 7, pp. 1726–1740, Jul. 2018.

[21] M. Chen, X. Wei, Q. Yang, Q. Li, G. Wang, and M.-H. Yang, “Spatiotemporal GMM for background subtraction with superpixel hierarchy,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 6, pp. 1518–1525, Jun. 2018.

[22] Z. Zivkovic, “Improved adaptive Gaussian mixture model for background subtraction,” in *Proc. IEEE Int. Conf. Pattern Recognit.*, vol. 2, Aug. 2004, pp. 28–31.

[23] C. R. del-Blanco, F. Jaureguizar, and N. Garcia, “An efficient multiple object detection and tracking framework for automatic counting and video surveillance applications,” *IEEE Trans. Consum. Electron.*, vol. 58, no. 3, pp. 857–862, Aug. 2012.

[24] Y. T. Chen, C. S. Chen, C. R. Huang, and Y. P. Hung, “Efficient hierarchical method for background subtraction,” *Pattern Recognit.*, vol. 40, no. 10, pp. 2706–2715, Oct. 2007.

[25] R. Zhang, S. Zhang, and S. Yu, “Moving objects detection method based on brightness distortion and chromaticity distortion,” *IEEE Trans. Consum. Electron.*, vol. 53, no. 3, pp. 1177–1185, Aug. 2007.

[26] X. Zhang, C. Zhu, H. G. Wu, Z. Liu, and Y. Y. Xu, “An imbalance compensation framework for background subtraction,” *IEEE Trans. Multimedia*, vol. 19, no. 11, pp. 2425–2438, Nov. 2017.

[27] J. Kim, D. Yeom, and Y. Joo, “Fast and robust algorithm of tracking multiple moving objects for intelligent video surveillance systems,” *IEEE Trans. Consum. Electron.*, vol. 57, no. 3, pp. 1165–1170, Aug. 2011.

[28] M. Genovese and E. Napoli, “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.*, vol. 22, no. 3, pp. 537–547, Mar. 2014.

[29] M. Genovese and E. Napoli, “FPGA-based architecture for real-time segmentation and denoising of HD video,” *J. Real Time Image Process.*, vol. 8, pp. 389–401, Dec. 2013.

[30] C. Cuevas, D. Berjon, F. Moran, and N. Garcia, “Moving object detection for real-time augmented reality applications in a GPGPU,” *IEEE Trans. Consum. Electron.*, vol. 58, no. 1, pp. 117–125, Feb. 2012.

[31] D. Berjon, C. Cuevas, F. Moran, and N. Garcia, “GPU-based implementation of an optimized nonparametric background modeling for real-time moving object detection,” *IEEE Trans. Consum. Electron.*, vol. 59, no. 2, pp. 361–369, May 2013.

[32] M. Babaee, D. T. Dinh, and G. Rigoll, “A deep convolutional neural network for video sequence background subtraction,” *Pattern Recognit.*, vol. 76, pp. 635–649, Apr. 2018.

[33] P. W. Patil and S. Murala, “MSFgNet: A novel compact end-to-end deep network for moving object detection,” *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 11, pp. 4066–4077, Nov. 2019.

[34] T. Akilan, Q. M. J. Wu, and W. Zhang, “Video foreground extraction using multi-View receptive field and encoder–decoder DCNN for traffic and surveillance applications,” *IEEE Trans. Veh. Technol.*, vol. 68, no. 10, pp. 9478–9493, Oct. 2019.

[35] H. Lee, H. Kim, and J.-I. Kim, “Background subtraction using background sets with image-and color-space reduction,” *IEEE Trans. Multimedia*, vol. 18, no. 10, pp. 2093–2103, Oct. 2016.

[36] L. Tessens, M. Morbee, W. Philips, R. Kleihorst, and H. Aghajan, “Efficient approximate foreground detection for low-resource devices,” in *Proc. ACM/IEEE Int. Conf. Distrib. Smart Cameras*, Sep. 2009, pp. 1–8.

[37] H. Kim and H. J. Lee, “A low-power surveillance video coding system with early background subtraction and adaptive frame memory compression,” *IEEE Trans. Consum. Electron.*, vol. 63, no. 4, pp. 359–367, Nov. 2017.

[38] M. Wu and X. Peng, “Spatio-temporal context for codebook-based dynamic background subtraction,” *AEU Int. J. Electron. Commun.*, vol. 64, no. 8, pp. 739–747, Aug. 2010.

[39] J. M. Guo, C. H. Hsia, Y. F. Liu, M. H. Shih, and C. H. Chang, “Hierarchical method for foreground detection using codebook model,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 6, pp. 804–815, Mar. 2011.

[40] Q. Wang, P. Zhou, J. Wu, and C. Long, “P-FAD: Real-time face detection schemeon embedded smart cameras,” *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 3, no. 2, pp. 210–222, Apr. 2013.

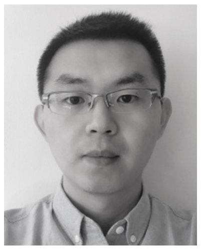
[41] R. D. Yates and D. J. Goodman, *Probability and Stochastic Processes*. Hoboken, NJ, USA: Wiley, 1999.

[42] C. Cuevas and N. García, “Efficient moving object detection for lightweight applications on smart cameras,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 1, pp. 1–14, Jan. 2013.

[43] *CDW 2012 Change Detection Dataset*. Accessed: Apr. 2021. [Online]. Available: http://www.changedetection.net

[44] *Scene Background Initialization (SBI) Dataset 2015*. Accessed: Apr. 2021. [Online]. Available: https://sbmi2015.na.icar.cnr.it/SBIdataset.html

[45] *Limu Dataset for Detection of Moving Objects*. Accessed: Apr. 2021. [Online]. Available: https://limu.ait.kyushu-u.ac.jp/dataset/en/



Shushang Li received the M.S. degree from the China University of Mining and Technology, Beijing, China, in 2013. He is currently pursuing the Ph.D. degree with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China. His current research interests include image forensics, embedded computer vision, and deep learning.

Jing Wu (Senior Member, IEEE) received the B.S. degree in electrical engineering from Nanchang University, Nanchang, China, in 2000, the M.S. degree in electrical engineering from Yanshan University, Hebei, China, in 2002, and the Ph.D. degree in electrical engineering from the University of Alberta, Edmonton, AB, Canada, in 2008. Since 2011, she has been with Shanghai Jiao Tong University, Shanghai, China, and is currently an Associate Professor. She is a Registered Professional Engineer in Alberta, Canada. Her current research interests include robust model predictive control, security control, and stability analysis, and estimations for cyber-physical systems.

Chengnian Long (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees in control theory and engineering from Yanshan University, Hebei, China, in 1999, 2001, and 2004, respectively. He was a Research Associate with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, and a Killam Postdoctoral Fellow with the University of Alberta, Canada. Since 2009, he has been with Shanghai Jiao Tong University, and is currently a Full Professor since 2011. His current research interests include artificial intelligence of things, blockchain technology, deep learning, and cyber-physical system security.

Yi-Bing Lin (Fellow, IEEE) is the Winbond Chair Professor with National Yang Ming Chiao Tung University, and the Chair Professor of National Cheng Kung University and China Medical University. He has authored the books *Wireless and Mobile Network Architecture* (Wiley, 2001), *Wireless and Mobile All-IP Networks* (John Wiley, 2005), and *Charging for Mobile All-IP Telecommunications* (Wiley, 2008). He received numerous research awards, including the 2005 NSC Distinguished Researcher, the 2006 Academic Award of Ministry of Education, and the 2008 Award for Outstanding contributions in Science and Technology, Executive Yuen, the 2011 National Chair Award, and the TWAS Prize in Engineering Sciences, 2011 (The Academy of Sciences for the Developing World). He is an AAAS Fellow, ACM Fellow, and IET Fellow.