Available online at www.sciencedirect.com

ScienceDirect

CAAI Transactions on Intelligence Technology 1 (2016) 43–60
http://www.journals.elsevier.com/caai-transactions-on-intelligence-technology/

Original article

Background modeling methods in video analysis: A review and comparative evaluation

Yong Xu [a,b,*], Jixiang Dong [a], Bob Zhang [c], Daoyun Xu [d]

[a] *Bio-Computing Research Center, Shenzhen Graduate School, Harbin Institute of Technology, 518055 Shenzhen, China*
[b] *Shenzhen Key Laboratory of Network Oriented Intelligent Computation, Shenzhen, China*
[c] *Department of Computer and Information Science, University of Macau, Avenida da Universidade, Taipa, Macau, China*
[d] *The College of Computer Science and Technology, Guizhou University, Guiyang 550025, Guizhou, China*

Available online 4 June 2016

Abstract

Foreground detection methods can be applied to efficiently distinguish foreground objects including moving or static objects from background which is very important in the application of video analysis, especially video surveillance. An excellent background model can obtain a good foreground detection results. A lot of background modeling methods had been proposed, but few comprehensive evaluations of them are available. These methods suffer from various challenges such as illumination changes and dynamic background. This paper first analyzed advantages and disadvantages of various background modeling methods in video analysis applications and then compared their performance in terms of quality and the computational cost. The Change detection.Net (CDnet2014) dataset and another video dataset with different environmental conditions (indoor, outdoor, snow) were used to test each method. The experimental results sufficiently demonstrated the strengths and drawbacks of traditional and recently proposed state-of-the-art background modeling methods. This work is helpful for both researchers and engineering practitioners. Codes of background modeling methods evaluated in this paper are available at www.yongxu.org/lunwen.html.

Keywords: Background modeling; Video analysis; Comprehensive evaluation

1. Introduction

Foreground detection based on video streams is the first step in computer vision applications, including real-time tracking [1,2] and event analysis [3–6]. Many researchers in the field of image and video semantics analysis pay attention to intelligent video surveillance in residential areas, junctions, shopping malls, subways, and airports which are closely associated with foreground detection [7–9]. Background modeling is an efficient way to obtain foreground objects. Though background modeling methods for foreground detection have been studied for several decades, each method has its own strength and weakness in detecting objects of interest from video streams [10,11]. Therefore a comprehensive evaluation is needed to help researchers and practitioners choose suitable methods under different scenarios.

Over the past few decades, a large number of background modeling methods have been proposed to identify foreground objects in a video. They generally share the same following scheme [2,12]: they utilize the first frame or previous frames to build a background model, and then compare the current frame with the background model to detect foreground objects, and finally they update the background model. Various background modeling methods can be categorized into pixel-based, region-based, and hybrid methods. Background modeling methods can also be categorized into parametric and nonparametric methods. One of the most famous pixel-based parametric methods is the Gaussian model. Wren et al. [13] first proposed modeling the background at each pixel

* Corresponding author. Bio-Computing Research Center, Shenzhen Graduate School, Harbin Institute of Technology, 518055 Shenzhen, China.
E-mail address: yongxu@ymail.com (Y. Xu).
Peer review under responsibility of Chongqing University of Technology.

http://dx.doi.org/10.1016/j.trit.2016.03.005

location with a Gaussian distribution [14,15]. However a single Gaussian function is not able to quickly deal with an actual dynamic background owing to a low updating rate of the background model [14]. In order to eliminate the influence of the background texture caused by waves on the water or trees shaken by the wind [15], Stauffer and Grimson [16,17] proposed the Gaussian mixture model (GMM) which models every pixel with a mixture of K Gaussians functions. After that, an improvement to GMM was proposed by using the online EM-based algorithm to initialize the parameter in the background model, which is time consuming. Zivkovic also [18,43] proposed an adaptive GMM (AGMM) to efficiently update parameters in GMM, and Lee [19] used a new adaptive learning rate to improve the convergence rate without changing the stability of GMM [9]. To improve the accuracy and reduce the computational time, Shimada et al. [20] used a dynamic Gaussian component to control the Gaussian mixture model. In addition, Oliver et al. [22] proposed a Bayesian method to model the background based on the prior knowledge and evidence from the data. Chien et al. [63] proposed a threshold decision method to detect foreground objects. They assumed the camera noise to be the zero-mean Gaussian distribution which is the only factor affecting the threshold. However, this assumption is hard to satisfy in practice.

Unlike parametric background modeling methods, a nonparametric algorithm based on self-organization through artificial neural networks (SOBS) was proposed by Maddalena et al. [30]. Kim et al. [28,29] proposed a codebook method to model the background which initializes codewords of codebooks to store background states. Wang et al. [55] proposed a method computing sample consensus (SACON) of the background samples to estimate a statistical model of the background, per pixel. SACON exploits both color and motion information to detect foreground objects. Barnich et al. [23,24] proposed a pixel-based nonparametric algorithm named Vibe to detect the foreground using a novel random selection strategy. The performance of Vibe is superior to many other state-of-the-art methods and it can represent exact background changes in recent frames [25]. The Vibe method was further studied by Van Droogenbroeck and Paquot [26] and they considered additional constraints to enhance the performance of Vibe. Another pixel-based nonparametric adaptive segmenter (PBAS) method was proposed by Hofmann et al. [27]. PBAS makes the foreground decision by applying a history of recently observed pixel values as the background model. Although, pixel-based background modeling methods can effectively obtain detailed shapes of foreground objects, they are easily affected by noise, illumination changes, and dynamic backgrounds.

Differing from pixel-based methods, region-based methods take advantage of inter-pixel relations to segment the images into regions and identify foreground objects from image regions. Elgammal et al. [21,41] presented a novel method by building a nonparametric background model based on kernel density estimation (KDE). Seki et al. [64] applied co-occurrence of image variations to model background changes in image regions. A heuristic block matching algorithm was proposed by Russell et al. [65] to distinguish foreground object from the background. They compared each image region of incoming frames with typical examples of a fixed-size database of backgrounds. In order to solve the dynamic background modeling in outdoor swimming pool environments, Eng et al. [66] used random homogeneous region movements and pre-filtering of image regions in the ClELab color space to detect foregrounds. In addition to methods featured by color, texture or descriptor-based methods also received much attention among region-based methods. Heikkila et al. [67] employed a discriminative texture feature called local binary pattern (LBP) [77] for modeling the background. They built LBP histograms based on partially overlapping regions for the background, and compared them with LBP histograms of each region of incoming frames via histogram intersection. Liu et al. [68] proposed a binary descriptor-based background modeling method to extract foreground objects under illumination changes. In addition, Huang et al. [69] modeled the background as samples of binary descriptors which can replace parametric distributions. In contrast to pixel-based methods, region-based methods can reduce the effects of noise, however, they can only obtain rough shapes of foreground objects.

Hybrid methods, which integrate both pixel-based and region-based methods, can achieve better background representation and deal with illumination changes and dynamic backgrounds [70]. The Wallflower system proposed by Toyama et al. [1] obtains the background model using pixel-level, region-level, and frame-level information. It applies the Wiener filter to predict background values at the pixel level, fills homogeneous regions of foreground objects at the region level, and handles sudden or global changes of a video sequence at the frame level. Huang et al. [71] integrated pixel-based RGB colors with optical-flow motions to model the background. Though hybrid methods can efficiently retrieve foreground objects from the background, their computational complexity is relatively high. Thus, Tsai et al. [72] proposed to embed hybrid algorithms in hardware to implement foreground detection. Some representative background modeling methods are classified in Table 1.

Table 1
Classification of representative background modeling methods.

Background modeling methods				
Category	Pixel-based methods		Region-based methods	Hybrid methods
Parametric	GMM [16]	AGMM [18,43]	Russell [65]	Huang [71]
	Oliver [22]		Heikkila [67]	Tsai [72]
Nonparametric	Vibe [23,24]	Schick [73]	KDE [21,41]	Cristani [75]
	SACON [55]	CodeBook [28,29]	Seki [64]	Chen [74]
	SOBS [30]	PBAS [27]	Liu [68]	Toyama [1]

Numerous surveys and comparison studies have been published to evaluate these kind of methods [2,10,31−37]. Unfortunately, many papers do not contain recent popular and state-of-the-art background modeling methods. Toyama et al. [1] and Panahi et al. [36] provided a comparison for various pixel-based background modeling methods. Their evaluation compared background modeling methods and described the challenges that background modeling methods need to deal with. Cheung et al. [81,82] evaluated six background modeling methods using traffic monitoring videos under different weather conditions. This comparison was only conducted on four grayscale test datasets. Parks et al. [37] and Brutzer et al. [10] compared the performance of several background modeling methods and examined the effects of post-processing. Recently, Sobral et al. [2] presented a paper entitled "a comprehensive review of background subtraction algorithms with synthetic and real videos". This paper compared 29 methods which tested by a C++ MFC Application using BGSlibrary (a background subtraction library) [38] and without state-of-the-art methods such as Vibe.

In this paper, we used the Change detection.Net (CDnet2014) [39] dataset and another video dataset [83] to conduct comparative experiments of background modeling methods. CDnet2014 dataset consists of 53 video sequences representing 11 video categories including outdoor and indoor environments with cars, pedestrians, and other objects. Most frames in the videos were annotated for obtaining the ground truth foreground, background, and shadow region boundaries. Another video dataset [83] contains four video sequences. This dataset includes the challenges of large size of occlusion and every 10-th frame in each sequence is labeled the ground truths. This paper objectively evaluated the performance of background modeling methods from two aspects. For the objective comparison of methods, recall, specificity, false positive rate (FPR), false negative rate (FNR), percentage wrong classification (PWC), precision, and F-measure evaluation metrics are used to present the accuracy of detection. To compute the accuracy of detection, we compared the foreground detection mask with the ground truth. For a background modeling method, it is also important to evaluate the speed and memory requirement, so corresponding evaluation was also performed. The remainder of this paper is organized as follows: Section 2 analyzes the challenges which background modeling methods have to deal with. Section 3 briefly reviews representative methods for background modeling. Section 4 shows the experiment protocol. Section 5 shows the experimental results. The paper is concluded in Section 6. The structure of this paper is summarized in Fig. 1.

2. Challenges of background modeling methods

Though background modeling methods are significant, there are various challenges in the video sequences owing to complex application environments [10]. Many publications have exemplified different background modeling challenges, for example, Bouwmans et al. [60] stated several typical challenges to evaluate the background modeling methods.

2.1. Illumination change

Background modeling methods should take into account their robustness against illumination changes. In particular, many computer vision systems are used in outdoor scenes, and the light intensity usually varies, and therefore a robust

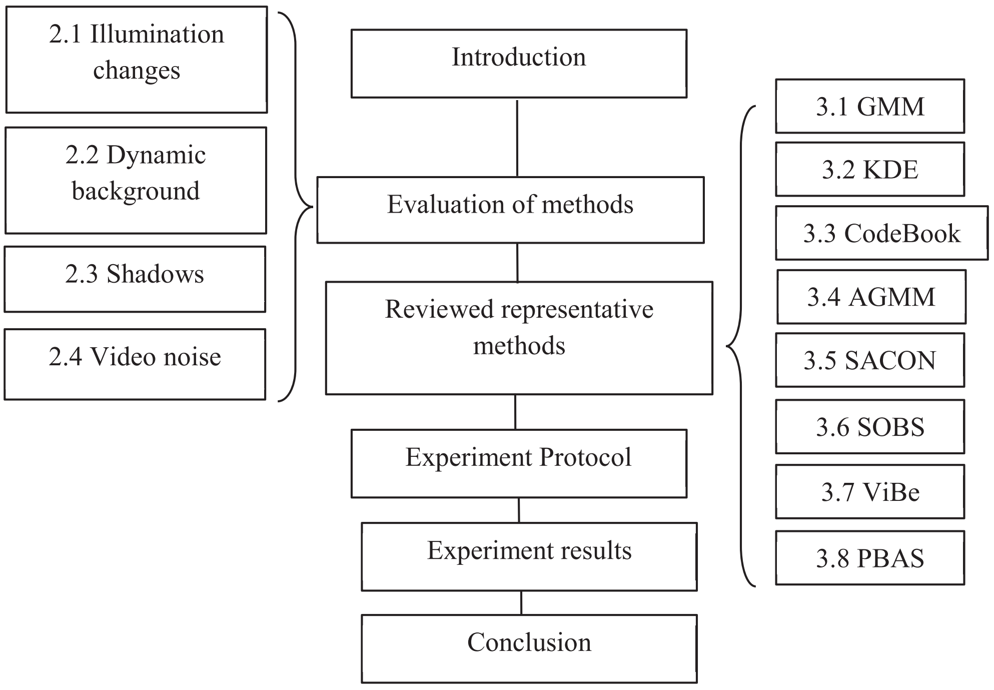


Fig. 1. Structure of this paper.

background modeling method must be able to adapt to gradual changes of the light in the environment.

Illumination changes are not only just progressive, but sudden once-off illumination changes also often occur in scenes. For example, a sudden switch of light or sunlight blocked by clouds strongly affect the appearance of the background model. Therefore, the ability to quickly adapt to sudden illumination changes is a key to evaluate the background modeling methods.

The GMM algorithm is sensitive to illumination changes. It defines a pixel value within 2.5 standard deviations of a distribution. This threshold can ensure that performance of the algorithm is only slightly perturbed by illumination changes which is extremely useful when different regions have different lighting. In order to minimize the effects of brightness changes, SOBS uses the HSV color space and SACON uses a normalized color space. The main reason is that RGB color space is sensitive to changes of illumination. In addition, a long term background model and a short term background model are combined in KDE to quickly adapt to changes in the scene to achieve sensitive detection and low false positive rates. CodeBook uses a color model to perform a separate evaluation of color distortion and brightness distortion. The motivation of this model is that background pixel values lie along the principal axis of the codeword along with the low and high boundaries of brightness, since the variation is mainly due to brightness.

2.2. Dynamic background

Though we may assume that the background contains only static objects, in normal circumstances some parts of a background always contain some movement such as waving trees, water ripples, traffic lights, and flashing displays. An excellent background modeling method should effectively identify periodical or irregular movement of objects.

To deal with the dynamic background, GMM and SOBS assume that the intensity values of a pixel are modeled by multimodal distributions. In contrast to a single unimodal model, multimodal models usually have high time complexity. Vibe and SACON are sample-based methods. They use the difference between new pixels and the sample to discriminate a moving background. SACON uses a selective strategy to update the background model. It can not only efficiently cope with lighting changes but can deal with objects appearing or fading in the background. It can incorporate the moved/inserted background object into the background model. For example, if a background object is moved to a new place, or a new background object is inserted into the background scene, this method can adaptively add the corresponding pixels of the background object to the background model [55].

Among region-based methods, texture or descriptor-based methods model background changes using descriptors provide reliable information to represent dynamic backgrounds and illumination changes. A discriminative texture called local binary pattern (LBP) [77] was used by Heikkila et al. [67] to model backgrounds. Zhu et al. [78] proposed a corner-based background modeling method to detect foreground objects in dynamic scenes. Harris corners [79] are detected and represented by SIFT-like descriptors [80] without multiscale implementation and orientation alignment. Then, each background model is constructed by the frequency of corner occurrence, means of the descriptors, and the correlation matrix between previous and current frames.

2.3. Shadows

Foreground objects often have shaded areas owing to the influence of light changing which usually affect the separation of foreground objects and the performance of subsequent modules of a background modeling algorithm. Hence, a robust algorithm should contain a strategy to remove shadows of foreground regions or ignore these irrelevant shadows.

The idea that a cast shadow certainly darkens the background while a moving object can darken it or not is used in the SOBS algorithm to detect shadows [61]. Specifically, in a shadowed area, there is significant illumination variation, but only a small color variation. If a pixel belongs to the background model but has been darkened by a shadow, then it belongs to the shadow cast by an object in the scene. KDE and SACON separate color information from lightness information and use the chromaticity coordinates to suppress shadows. However, using the chromaticity coordinates has the disadvantage of losing lightness information which is related to the difference in whiteness, blackness, and grayness between different objects [62].

2.4. Video noise

In the applications of computer vision, noise is always inevitable and video signals are generally affected by noise [10]. For example, camera shaking, lens aging, sensor noise, or compression artifacts can cause image degradation. Background modeling methods should consider the effects of these factors and should be able to cope with degraded signals affected by different types of noise.

Vibe and SACON only need to compare the current pixel with a small number of close background samples rather than most of samples in the background model, so they can weaken the effects of noise in the model. Two factors explain why Vibe has a high resilience to noise. The first factor is that the design of the Vibe method allows the pixel model of Vibe to be exclusively comprised of observed pixel values. The pixel models of Vibe automatically adapt to noise, as they are constructed from noisy pixel values. The use of the pure conservative update scheme in Vibe is the other factor. By relying on pixel values exclusively classified as background, the model update scheme of Vibe prevents the inclusion of any outlier in the pixel models [24].

3. Representative methods

In this section, the paper reviewed the background modeling algorithms to be compared and evaluated. These algorithms range from the most classical pixel-based parametric methods such as GMM [16,17] to sophisticated nonparametric methods [23,24,27,30], as well as region-based methods such as KDE [21,41]. These methods aim to improve the accuracy of classification and achieve maximum speed and low memory consumption under possible conditions [14].

3.1. Gaussian mixture model (GMM)

In order to model a background which is dynamic or contains animated texture (such as waves on the water or trees shaken by the wind) [15,31], Stauffer et al. [16] proposed to model each pixel with a mixture of K Gaussians distributions. The probability of input pixel value x from current frame at time t being a background pixel is represented by the following mixture of Gaussians

$$P(x_t) = \sum_{i=1}^{K} \omega_{i,t} \cdot \eta\left(x_t, \mu_{i,t}, \sum_{i,t}\right) \tag{1}$$

where K is the number of Gaussian distributions, $\eta(x_t,\mu_{i,t},\sum_{i,t})$ is the *i-th* Gaussian probability density function, $\omega_{i,t}$ is its weight at time t, $\mu_{i,t}$ is the mean value of the *i-th* Gaussian in the mixture at time t, and $\sum_{i,t}$ is the covariance matrix of the *i-th* Gaussian in the mixture at time t. In practical cases, the covariance matrix can be assumed to be a diagonal matrix and K is set between 3 and 5.

A new pixel x_t, is checked against the exiting K Gaussian distributions, until a match is found. A match is defined as a pixel value within 2.5 standard deviations of a distribution. If none of K distributions match the current pixel value, the least probable distribution is replaced with a distribution with the current value as its mean value, an initially high variance, and low prior weight. The prior weights of the K distributions at time t, $\omega_{i,t}$ are updated as,

$$\omega_{i,t} = (1-\alpha)\omega_{i,t-1} + \alpha \cdot M_{k,t} \tag{2}$$

where α is a learning rate and $M_{k,t}$ is 1 for the model which matched, otherwise, $M_{k,t}$ is 0.

The parameters $\mu_{i,t}$ and standard deviation $\sigma_{i,t}$ for unmatched distributions remain the same. When they match the new observation, they are updated as follows

$$\mu_{i,t} = (1-\rho)\mu_{i,t-1} + \rho \cdot x_{i,t} \tag{3}$$

$$\sigma_t^2 = (1-\rho)\sigma_{t-1}^2 + \rho(x_t-\mu_t)^T(x_t-\mu_t) \tag{4}$$

where ρ is a learning rate defined as $\rho = \alpha\eta(x_t|\mu_k,\sigma_k)$.

In order to recognize a pixel in a frame as a foreground or background pixel, Stauffer et al. [16] used the following way to estimate the background model: first, the K Gaussians distributions are ordered by the value of ω/σ. Then only the first B distributions are chosen as the background model, distributions should satisfy

$$B = \arg\min_b \left(\sum_{k=1}^{b} \omega_k > T \right) \tag{5}$$

where T is an assigned threshold. If pixel values cannot match the background model distributions, they will be labeled “in motion” [15].

The traditional GMM has several advantages. It does not need to store a set of input data in the running process. GMM uses the mean value and covariance to measure the pixel. This means that each pixel has its own unique threshold without the constraint of a unified global threshold. The multimodality of GMM allows it to deal with multimodal backgrounds caused by waving trees and gradual illumination changes.

However, GMM also has some disadvantages. The number of Gaussians must be predetermined, and it is suggested to set K to 3, 4 or 5 [16]. When the background model is defined, the parameters of the GMM must be initialized. In addition, the results depend on the distribution law which can be a non-Gaussian distribution and it only slowly recovers from failures. A series of training frames without moving objects to train the GMM background model is another limitation of GMM and this step require enough memory.

3.2. Kernel density estimator (KDE)

Background model methods can be divided into parametric or non-parametric methods. Parametric methods estimate the background by building the parameter estimation probability distribution based on the color distribution of images [9]. In non-parametric methods, a kernel function is introduced to model the distribution. For this kind of method, Elgammal et al. [21] introduced Kernel density estimation on N recent sample of intensity values $\{x_1, x_2\ldots, x_N\}$ to model the background distribution.

In this method, the probability of the intensity of each pixel x_t at time t can be estimated by

$$\Pr(x_t) = \frac{1}{N}\sum_{i=1}^{N}\prod_{j=1}^{d}\frac{1}{\sqrt{2\pi\sigma_j^2}}e^{-\frac{1}{2}\frac{\left(x_{t_j}-x_{i_j}\right)^2}{\sigma_j^2}} \tag{6}$$

where N is the number of samples, d is the number of channels, and σ represents the kernel function bandwidth for each color channel estimated by

$$\sigma = \frac{m}{0.68\sqrt{2}} \tag{7}$$

where m is the median absolute deviation over the sample for consecutive values of the pixel [21]. Mittal and Paragio [40] also used a more sophisticated method based on variable bandwidth kernels to determine σ.

If $Pr(x_i) < T$, then pixel x_i will be classified as a foreground pixel, otherwise, it will be considered as a background pixel. Threshold T is a global threshold over all images and it can be adjusted to achieve a desired percentage of false positives [21].

In the update of the background model, the paper presented a way to generate two background models (a long term model and a short term model) to achieve better update decisions and make the model quickly adapt to changes in the scene to support sensitive detection and low false positive rates [21]. The short term model consists of the most recent N background sample values and the sample uses a selective-update strategy to perform updating. In contrast to the short term model, the long term model is updated by using a blind strategy.

One of the strengths of the KDE method is the ability to circumvent a part of the delicate parameter estimation step due to the fact that it relies on pixel values observed in the past [21,40–44]. Another advantage is that KDE can deal with multimodal backgrounds particularly in a background with fast changes by directly including newly observed values in the pixel model.

KDE also has disadvantages. For example, during the entire foreground detection process, KDE has to keep N frames in memory which is time consuming when N is large. To solve these problems, researchers proposed different improvements. For example, the number of training samples can be decreased by determining a proper size of the frame buffer [45]. Furthermore, adopting a recursive strategy maintaining the background to reduce the computation time is also feasible [46,47]. However, the key improvement is to change the kernel function, for example, Zivkovic [43] used a variable kernel bandwidth in a rectangular kernel function. Ianasi et al. [45] and Tanaka et al. [49] also used a rectangular kernel function but with a constant kernel bandwidth. Tavakkoli et al. [48] selected automatically kernel bandwidth in a Gaussian kernel function and Ramezani et al. [50] used the Cauchy kernel function.

3.3. CodeBook

Kim et al. [28,29] proposed a method called codebook which uses a quantization/clustering technique inspired by Kohonen [51,52] to obtain a multimodel background model from long observation sequences. For each pixel, the method builds a codebook to store one or several codewords based on training sequences. The codewords are a series of key color values and the number depends on the background variation [53].

Each codeword consists of an RGB vector $v_i = (\overline{R}_i, \overline{G}_i, \overline{B}_i)$ and a 6-tuple $aux_i = <\check{I}_i, \hat{I}_i, f_i, \lambda_i p_i, q_i>$. The tuple aux_i contains intensity (brightness) values and temporal variables, $\check{I}$ and $\hat{I}$ are the min and max brightness, respectively, of all pixels assigned to this codeword, f is the frequency at which the codeword has occurred, λ is the maximum negative run-length (MNRL) defined as the largest time interval during which the codeword is not updated or recurred, p and q, respectively, are the first and last access time that the codeword has occurred.

The method uses the following criteria to create or update the codewords in the codebook during the initial training time. Color distortion δ is first calculated by

$$p^2 = \|x_t\|^2 \cos^2\theta = \frac{<x_t, v_t>^2}{\|v_t\|^2},$$

$$\delta = \text{colordist}(x_t, v_t) = \sqrt{\|x_t\|^2 - p^2} \tag{8}$$

p^2 is the autocorrelation of R, G, and B colors of the input pixel, and δ is color distortion and its measure can be interpreted as a brightness-weighted version in the normalized color space which is equivalent to geometrically normalizing a codeword vector to the brightness of an input pixel.

The value of brightness changes can vary in a certain range that has two bounds: $I_{low} = \alpha\hat{I}$ and $I_{hi} = \min\left\{\beta\hat{I}, \frac{\check{I}}{\alpha}\right\}$ where $\alpha < 1$ and $\beta > 1$.

After the initial training time, the scene can change. For example, cars might enter or leave a parking area on a street surveillance application, and the system can detect the foreground. Kim et al. [28] also introduced another structure similar to codewords called cache book to avoid detecting false background or foreground pixels because codebook cannot be adapted to changes in the scene [53].

GMM with just a few Gaussians cannot accurately model quickly varying backgrounds. It depends on the learning rate to adapt to background changes, and thus for a low learning rate, it is difficult to detect a sudden change in the background. For a high learning rate, slowly moving object will be absorbed into the background. To solve these problems, KDE was exploited [21]. It can quickly adapt to changes in the background and detect the foreground with high sensitivity. However, KDE cannot be used when long-time periods are needed to sufficiently sample the background.

The CodeBook algorithm was intended to sample values over a long time, but there are no parametric assumptions. It can capture background motion over a long period of time under limited memory which allows it to encode dynamic backgrounds. In addition, it can efficiently cope with local and global illumination changes. In contrast to GMM, it properly deals with moving foreground objects in the scene during the initial training period. However, there are some problems that codebook cannot cope with. For example, if the color of foreground pixels is similar to that of background pixels, it will incorrectly segment the foreground. Though it can tune the parameters to partially overcome this problem, it simultaneously reduces the global performance in other situations.

3.4. Adaptive Gaussian mixture model (AGMM)

Basic GMM uses a fixed number of components, but Zivkovic designed an improved algorithm based on the result from [54] to adaptively adjust the parameters and number of components of GMM. The improved algorithm can automatically adapt to the scene by choosing the number of components for each pixel [18]. Given a new data sample x_t at time t, the method updates recursively parameters $\omega_{i,x,t}$, $\mu_{i,x,t}$, and $\sigma_{i,x,t}$ as follows

$$\omega_{i,x,t} = \omega_{i,x,t} + \alpha(o_{i,x,t} - \omega_{i,x,t}) \tag{9}$$

$$\mu_{i,x,t} = \mu_{i,x,t} + o_{i,x,t}(\alpha/\omega_{i,x,t})\delta_{i,x,t} \tag{10}$$

$$\sigma^2_{i,x,t} = \sigma^2_{i,x,t} + o_{i,x,t}(\alpha/\omega_{i,x,t})\left(\delta^T_{i,x,t}\delta_{i,x,t} - \sigma^2_{i,x,t}\right) \tag{11}$$

where $\delta_{i,x,t} = x_t - \mu_{i,x,t}$, constant $\alpha = 1/P$, P is a reasonable selected time adaptation period, and α defines an exponentially decaying envelope which can limit the influence of previous data. For a new sample ownership $o_{i,x,t}$ is set to 1 for the "close" component with largest $\omega_{i,x,t}$ and the other $o_{i,x,t}$ is set to 0. For example, a sample is "close" to a component if the Mahalanobis distance from the component is less than three. The squared distance from the *i-th* components is calculated as $D^2_i(x_t) = \delta^T_i\delta_i/\sigma^2_i$.

If the components are sorted to have descending weight $\omega_{i,x,t}$, Eq. (5) is updated as follows

$$B = \arg\min_b\left(\sum_{i=1}^{b}\omega_i > (1 - c_f)\right) \tag{12}$$

where c_f is a measure of the maximum portion of the data that can belong to foreground objects without influencing the background model. For example, if a new object comes into a scene and remains static for some time, it will be temporally presented as an additional cluster. Weight ω_{B+1} of the new cluster will constantly increase because the "old" background is occluded. If the object remains static for a long time, its weight as an object will become larger than c_f and it will be classified as a part of the background.

3.5. Consensus-based method (SACON)

Wang et al. [55] proposed a method that computes the sample consensus of the background samples and estimates a statistical model of the background. The method builds a cache of N background samples at each pixel at time t, $\{x_i(m)|i = 1, ..., N, N < t\}$, where $x_t(m)$ is an observation of a pixel m at time t. Each observation $x_t(m) = (x_t^{C_1}(m), ..., x_t^{C_k}(m))$ has k channels. For each sample in the cache, the sample consensus in background modeling is defined as

$$\Gamma^c_i(m,t) = \begin{cases} 1 & \text{if}\left|x^c_i(m) - x^c_t(m)\right| \le T_r \\ 0 & \text{otherwise}\end{cases} \tag{13}$$

T_r is a selective threshold of the error tolerance and different pixels have different thresholds. T_r is proportional to the sample standard variance and the standard variance is overestimated when the data is a multimodal distribution. Therefore, the algorithm usually sets a global value of T_r for all pixels and simultaneously sets T_r to $\eta\sigma_i$. σ_i is the standard variance at each of the *i-th* image pixel and η is usually set as 2.5 or 3. Thus, the algorithm sets T_r for *i-th* image pixel: $T_{ri} = min(T_1, \eta\delta_i)$ and T_1 is a constant. Pixel m is consistent with sample i in channel c at time t when $\Gamma^c_i(m,t)$ is equal to 1. The algorithm simply judges whether previous samples "agree" with the current sample or not via

$$B_t(m) = \begin{cases} 1 & \sum\limits_{i=1}^{N}\Gamma^c_i(m,t) \ge T_n \;\; \forall c\in\{C_1, \cdots, C_k\} \\ 0 & otherwise\end{cases} \tag{14}$$

where $B_t(m)$ is a binary value with 1 for a background pixel and 0 for a foreground pixel, T_n is a threshold of the number of data points that are within error tolerance T_r of a mode [56] and T_n is influenced by sample size N. If N is large, T_n should also be large. Thus, T_n can be effectively set to $\tau T_r N$ where τ is a constant and is empirically determined.

The RGB color space is sensitive to changes of illuminations. In order to address this problem, the background model uses normalized color space (r, g, I) as the feature space. In addition, the algorithm removes shadows according to different values of l and uses the following modified version of Eq. (13)

$$\Gamma^c_i(m,t) = \{\begin{cases} 1 & if\left|x^c_i(m) - x^c_t(m)\right| \le T \;\forall c\in\{r,g\} \\ & and\; \beta \le x^c_t/x^c_b \;\le \gamma \;\; c\in\{\mathrm{I}\} \\ 0 & otherwise\end{cases} \tag{15}$$

When intensity I of the pixel is lower than a threshold *Itd*, and r and g values are not reliable and are sensitive to illuminations. In this case, the algorithm uses only intensity I as follows

$$x = \begin{cases} (r,g,I) & if\;\; I \ge Itd \\ (I) & otherwise\end{cases} \tag{16}$$

The algorithm uses a selective update strategy to update the background samples at both the pixel level and blob level. The algorithm creates a *TOM* value for each pixel and updates it at the pixel level and blob level, respectively. The *TOM* value is updated at the pixel level as follows

$$TOM_t(m) = \begin{cases} TOM_{t-1}(m) + 1 & if\;\; B_t(m) = 0 \\ 0 & otherwise\end{cases} \tag{17}$$

When the value of *TOM* at a pixel is larger than threshold T_{TM}, then the pixel is classified as a background pixel. Once the value of the pixel is assigned to the background, the value of *TOM* will be set to zero. In fact, *TOM* is used to record how many frames a pixel has been continuously classified as a foreground pixel.

When either the center of the object or the number of pixels of the object changes compared by a large amount with the values of the nearest blob in the previous frame, the object is considered as "moving foreground", otherwise, it is "static foreground". If pixels belong to the stationary foreground blob (a large connected region), the *TOM* value is updated at the blob level as follows

$$\underset{m'\in\Omega}{TOM}_t(m')\begin{cases} \underset{m'\in\Omega}{TOM}_{t-1}(m') + 1 & if\;\; \Omega\;\; is\;\; static \\ 0 & otherwise\end{cases} \tag{18}$$

If the *TOM* value of an object is higher than T_{TM}, this object is judged as a static object and all its pixels are used as the background samples. If a blob is judged to be moving, the *TOM* values of all pixels are set to zero. If a blob is judged to be static, the *TOM* value of all pixels of that blob are increased by 1. If the *TOM* value of an object is higher than T_{TM}, all its pixels are added to the background samples and this object is classified as background because of remaining stationary for a long time.

Wang and Suter [55] proposed to keep a cache (or history) of the last observed background samples for each pixel and classified a new pixel value as background if it matches most of the values stored in the pixel model. This method avoids the issues related to deviations from an arbitrarily assumed density model, but when a first-in-first-out scheme is used to update the background model, the method fails to avoid issues. Finally, to deal with lighting changes and objects appearing or fading in the background, two additional strategies (pixel-level and blob-level) are used to handle entire objects. When a part of an object is moving and other parts are static, the static part will be continuously classified as the foreground. Then, it will be updated as background at the pixel-level, but the moving regions will be judged as the foreground or background which is not updated at the pixel-level. Finally, an entire object will be divided into the foreground and the background. However, updating the background at the blob level can ensure the integrity of the object.

3.6. A self-organizing background subtraction method (SOBS)

Maddalena and Petrosino [30] proposed a background modeling method named SOBS based on self-organization achieved by artificial neural networks. The proposed method can adapt to the dynamic background, gradual illumination changes, and camouflage and achieve robust detection for different types of videos captured by stationary cameras [30].

For each pixel, the method builds a neural map consisting of $n \times n$ weight vector and the model of every pixel can be represented as $C = (c_1, c_2, \dots c_n)$. The whole set of weight vectors acts as a background model. For an incoming pixel p_t of the *t-th* sequence frame I_t, if a best matching weight vector c_m in the current pixel model C for p_t can be found, and p_t is considered as a background pixel. Otherwise, if p_t is the shadow, it will still be considered as the background and should be not used to update the corresponding weight vectors. If p_t is not the background and shadow, it should belong to the foreground.

Weight vector c_m gives the best match for incoming pixel p_t if its distance from p_t is the smallest among all weight vectors in model C of p_t and the distance is not greater than a fixed threshold. This is formulated by

$$d(c_m, p_t) = \min_{i=1,\cdots,n^2} d(c_i, p_t) \le \varepsilon \tag{19}$$

Threshold ε can distinguish a foreground pixel from a background pixel and it is a small constant defined as

$$\varepsilon = \begin{cases} \varepsilon_1 & if\ \ 0 \le t \le K \\ \varepsilon_2 & if\ \ t > K \end{cases} \tag{20}$$

ε_1 should be greater than ε_2, because high values for ε_1 allow, within the first K sequence frames, to obtain a (possibly rough) background model including several observed pixel intensity variations, ε_2 should be set to a lower value for obtaining a more accurate background model.

The algorithm initializes each neuron of pixels using the first image of the video sequence and achieves self-organization of neurons via the weighted average method. The background model will be updated based on the conservative and regional principle only when the pixels are considered as background pixels and the current pixel model will be spread to neighboring models after it is updated.

When the best match c_m in model C of current sample p_t is determined, c_m is regarded as the background model at position $(\bar{x}, \bar{y})$. Then weight vectors A_t in the $n \times n$ neighborhood of $(\bar{x}, \bar{y})$ are updated via

$$A_t(i,j) = \left(1 - \alpha_{i,j}(t)\right) A_{t-1}(i,j) + \alpha_{i,j}(t) p_t(x,y) \tag{21}$$

In Eq. (21), $\alpha_{i,j}(t) = \alpha(t)\omega_{i,j}$; $\omega_{i,j}$ is the Gaussian weights in the $n \times n$ neighborhood. $\alpha(t)$ is the learning factor and defined as

$$\alpha(t) = \begin{cases} \alpha_1 - t\dfrac{\alpha_1 - \alpha_2}{K} & if\ \ 0 \le t \le K \\ \alpha_2 & if\ \ t > K \end{cases} \tag{22}$$

where α_1 and α_2 are predefined constants such that $\alpha_2 \le \alpha_1$. In order to ensure that the value of $\alpha_{i,j}(t)$ is in the range of [0,1], α_1 and α_2 are set to

$$\begin{aligned} \alpha_1 &= \frac{c_1}{\max\{\omega_{i,j}\}} \\ \alpha_2 &= \frac{c_2}{\max\{\omega_{i,j}\}} \end{aligned} \tag{23}$$

where c_1 and c_2 are predefined constants such that $0 \le c_2 \le c_1 \le 1$. If the best match c_m is not found, p_t is classified as the foreground and the background model is not updated.

SOBS is based on the idea of building an image sequence neural background model by learning image sequence variations [57], thus it does not require prior knowledge about the involved patterns. The network behaves as a competitive neural network that implements a winner-take-all function, with an associated strategy that modifies the local synaptic plasticity of neurons, allowing learning them to be spatially restricted to the local neighborhood of the most active neurons. Therefore, the neural background model will adapt to scene changes and can capture the most persistent features of the image sequence.

However, for each color pixel, SOBS builds a neuronal map consisting of $n \times n$ weight vectors. Thus, each incoming sample must be measured between weight vectors and the minimum value should be determined, which is time consuming. If there is a strategy to efficiently update the background model, this problem may be solved.

3.7. A universal background subtraction algorithm (Vibe)

Barnich and Droogenbroeck [23] used a random strategy that was first used in the field of background modeling to

Table 2
Overview of the methods evaluated in this paper.

Method	Year	Classification	Features
GMM [16]	1999	Parametric, pixel	Color
KDE [21,41]	2000	Nonparametric, pixel	Color
CodeBook [28,29]	2004	Nonparametric, pixel	Color, luminance
AGMM [18,43]	2006	Parametric, pixel	Color
SACON [55]	2007	Nonparametric, pixel/region	Color, motion
SOBS [30]	2008	Nonparametric, pixel	Color
Vibe [23,24]	2011	Nonparametric, pixel	Color
PBAS [27]	2012	Nonparametric, pixel	Color

select values to build a sample-based estimation of the background. The algorithm needs to build a cache for every pixel to preserve a collection of N background sample values $M(x) = \{v_1, v_2, \ldots v_N\}$ from previous frames. The sample in M is randomly selected. An incoming pixel is classified based on its corresponding model $M(x)$. The algorithm defines a sphere $S_R(v(x))$ of radius R centered on $v(x)$ and determines the closest sample of value $v(x)$ from the set of samples. Pixel value $v(x)$ is then considered as the background if the cardinality, denoted by #, of the set intersection of this sphere and the collection of model samples $M(x)$ is greater than or equal to a given threshold $\#_{min}$ [24].

The algorithm utilizes the first frame of the video to initialize the background model based on an assumption the same as [59], i.e. neighboring pixels share a similar temporal distribution. The authors used $t = 0$ to index the first frame and $N_G(x)$ is a spatial neighborhood of pixel location x, therefore

$$M^0(x) = \left\{v^0(y|y \in N_G(x))\right\} \tag{24}$$

where location y is randomly chosen by using a uniform strategy.

In the processing of updating the background model, a non-recursive and conservative update policy is utilized. A pixel value will randomly replace the samples in $M(x)$ if it is classified as the background. In contrast to the first-in-first-out strategy, this strategy guarantees an exponential monotonic decay for the probability of previous sample values remaining in the pixel model. In addition, a random time strategy used in the algorithm also extends the time windows covered by the background pixel models. When a pixel value has been classified as background, this pixel has one chance in p to be selected to update its pixel model owing to a time subsampling factor adopted. Unfortunately, it is possible for a background sample to be incorrectly classified as foreground which prevents its background pixel model from being updated. To address this problem, Vibe uses the spatial consistency via a background sample propagation scheme. According to this scheme, when a pixel has been updated, the algorithm uses this value $v(x)$ to update the neighborhood pixel samples in $M(y \in N_c(x))$.

Vibe applies observed color values of pixels of background training sequences as samples of observed backgrounds. Thus, compared with other methods, Vibe achieves superior performance because using samples as background models can successfully represent background changes [25]. However, Vibe has the disadvantages that it only uses color values of pixels to build the background but color values are usually sensitive to noise and illumination changes [58]. Thus, the foreground detection performance of Vibe is easily affected by noise and illumination changes. In addition, Vibe uses fixed parameters to identify whether a pixel belongs to the background. Thus, for different videos especially videos containing dynamic backgrounds, Vibe needs to manually adjust parameters to adapt to background changes.

3.8. Pixel-based adaptive segmenter (PBAS)

Hofmann et al. [27] combined Vibe with SACON to create a novel pixel-based adaptive segmenter (PBAS) method for foreground segmentation. The PBAS method uses a history of N image values as the background model and uses a random update rule similar to Vibe. In contrast to Vibe, PBAS does not treat parameter values as fixed parameters, but instead as adaptive state variables, which can dynamically change over time for each pixel.

The background model $B(x_i)$ of PBAS is defined by an array of N recently observed pixel values

$$B(x_i) = \{B_1(x_i), \ldots, B_k(x_i), \ldots, B_N(x_i)\} \tag{25}$$

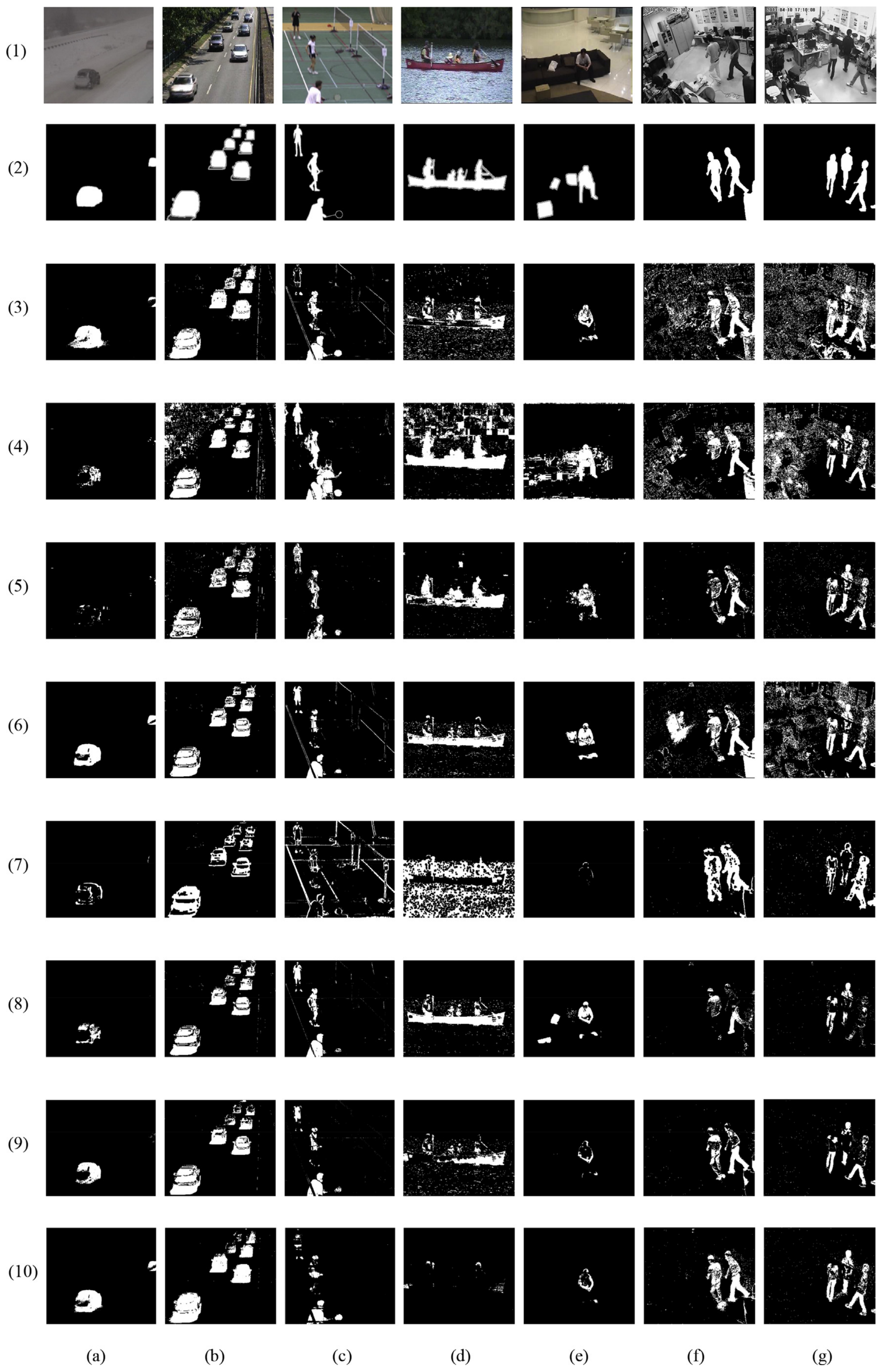
Pixel x_i is classified as background, if its pixel value $I(x_i)$ is closer to at least $\#_{min}$ of the N background values in terms of a certain decision threshold $R(x_i)$. Thus the background is defined as

$$B(x_i) = \begin{cases} 0 & \#\{dist(I(x_i), B_K(x_i)) < R(x_i)\} \geq \#_{\min} \\ 1 & otherwise \end{cases} \tag{26}$$

where $R(x_i)$ is separately defined for each pixel and can be dynamically changed. The minimum number $\#_{min}$ is a fixed global parameter. $R(x_i)$ needs to be able to automatically adapt

Table 3
Parameters used in evaluation.

Method	Parameters
GMM [16]	$K = 3$, $\alpha = 0.01$
KDE [21,41]	$N = 50$, $\alpha = 0.3$, $LF = 10$
CodeBook [28,29]	$LF = 30$,
AGMM [18,43]	$K = 4$, $\alpha = 0.013$
SACON [55]	$T_{TM} = 100$, $LF = 50$
SOBS [30]	$\varepsilon_1 = 0.15$, $\varepsilon_2 = 0.01$, $c_1 = 1$, $c_2 = 0.02$
ViBe [23,24]	$N = 20$, $R = 20$, $\#_{min} = 2$
PBAS [27]	$N = 20$, $\#_{min} = 2$, $R = 18$, $R_{inc/dec} = 0.05$, $R_{scale} = 5$, $T_{inc} = 1$, $T_{dec} = 0.05$, $T_{lower} = 2$, $T_{upper} = 200_r$


(1)
(2)
(3)
(4)
(5)
(6)
(7)
(8)
(9)
(10)
(a)
(b)
(c)
(d)
(e)
(f)
(g)

to areas with highly dynamic background. Thus, $R(x_i)$ is dynamically adapted as follows

$$R(x_i) = \begin{cases} R(x_i)\cdot\left(1 - R_{inc/dec}\right) & if\ R(x_i) > \bar{d}_{\min}(x_i)\cdot R_{scale} \\ R(x_i)\cdot\left(1 + R_{inc/dec}\right) & otherwise \end{cases} \tag{27}$$

here, $R_{inc/dec}$, R_{scale} are fixed parameters and $\bar{d}_{\min}(x_i)$ is the average of observed minimal distance $d_{min}(x_i)$.

When pixel x_i is classified as a background pixel, the background will be updated and a neighboring pixel y_i which might be a foreground pixel can be updated as well. This means that foreground pixels at the boundary will gradually be included into the background model. However, it depends on the update parameter $T(x_i)$. $T(x_i)$ is updated as follows

$$T(x_i) = \begin{cases} T(x_i) + \dfrac{T_{inc}}{\bar{d}_{\min}(x_i)} & if\ B(x_i) = 0 \\ T(x_i) - \dfrac{T_{dec}}{\bar{d}_{\min}(x_i)} & if\ B(x_i) = 1 \end{cases} \tag{28}$$

here, T_{inc}, T_{dec} are fixed parameters, T_{inc} means that parameter $T(x_i)$ increases and T_{dec} means that parameter $T(x_i)$ decreases. Furthermore, an upper and lower boundary $T_{lower} < T < T_{upper}$ were defined in paper [27], such that values cannot go outside a specified boundary. In the case of a highly dynamic background, $T(x_i)$ stays constant or only slightly changes which means that an erroneously detected foreground will not remain for a long time. However, in case of a static background, the classification is quite solid, hence $T(x_i)$ rapidly increases, validating the background model, and enabling the background model to be less updated.

4. Experimental protocol

4.1. Evaluation dataset

For performance evaluation, we used the CDnet2014 benchmark data set proposed in [76] and another video dataset proposed in [83]. CDnet2014 dataset contains 11 video categories with 4–6 videos sequences in each category. Most frames in videos were annotated for obtaining the ground truth data. For example, the category Dynamic Background contains six videos and the ground truth data of three videos are available, which present outdoor scenes with strong and dynamic background motions, and camera jitter aims to evaluate the robustness of the methods when videos are captured by vibrating cameras. However, this category only contains two videos which can be used to test methods because the ground truth data of others cannot be obtained. Dataset in [83] contains four video sequences, named Fighting, Walking I, Walking II and Pets 2006 S7. They are typical indoor visual data in surveillance scenarios including the challenges of large size of occlusion. In this paper, we chose the video sequence containing the ground truth data to evaluate the methods described in Section 3.

4.2. Performance measure

We evaluated the performance of background modeling methods at the pixel-level. Thus, we considered foreground detection as binary classification of each pixel. The correctness of this classification is expressed by the framework of the CDnet 2014 challenges [76]. The framework implements the following seven different measure metrics: recall, specificity, false positive rate (FPR), false negative rate (FNR), percentage of wrong classification (PWC), precision, and F-measure. Let *tp* denote the number of true positives and *tn* denote the number of true negative, respectively. Let *fp* and *fn* represent the numbers of false positives and false negatives, respectively. The performance metrics are expressed as:

$$recall = \frac{tp}{tp+fn}, \qquad specificity = \frac{tn}{tn+fp}$$

$$fpr = \frac{fp}{fp+tn}, \qquad fnr = \frac{fn}{tp+fn}$$

$$precision = \frac{tp}{tp+fp}, \quad pwc = 100\times\frac{fn+fp}{tp+fn+fp+tn}$$

$$f-measure = \frac{2\times precision\times recall}{precision+recall}$$

If positive is regarded as foreground and negative is regarded as background, then *tp* gives the number of correctly detected foreground pixels, and *tn* gives the number of correctly identified background pixels. In contrast, *fn* is the number of pixels that are falsely marked as background whereas *fp* is the number of pixels that are falsely detected as foreground. The method can trivially optimize one of them by ignoring the other one with precision and recall conflict to each other. In Section 5, the values of the seven measure metrics are given in details.

4.3. Experimental framework

In this paper, we tested a wide range of background modeling methods including traditional method such as GMM and recent methods such as Vibe and PBAS. The details for methods were introduced in Section 3. Table 2 shows overview of the methods which we evaluated in this paper.

For the GMM and AGMM method, we used the implementation based on code written by Zivkovic for his enhanced

Fig. 2. Foreground detection results of the bad weather, baseline, camera jitter, dynamic background, and intermittent object motion video from the CDnet2014 dataset, fighting, and walking I video from the dataset in [83]. (1) original frame. (2) ground truth. (3) GMM. (4) KDE. (5) CodeBook. (6) AGMM. (7) SACON. (8) SOBS. (9) Vibe. (10) PBAS.

GMM. The programs of Vibe and PBAS methods were provided by the authors of Vibe and PBAS. We used the best parameters of two methods suggested by the authors. For the KDE method, we used the implementation available in the BGSlibrary. We adjusted the parameters by the suggestion in the BGSlibrary. Because the codes of SACON, SOBS, and CodeBook methods are not available, we finished the code by ourselves, and selected the optimal parameters following the paper of the authors.

The selection of appropriate parameters is critical to the evaluation of background modeling methods. The various parameters settings used for each background modeling method are presented in Table 3. Default parameters, *LF* presents learning frames, α is learning rate, and other parameters can be found in Section 3.

5. Experimental results

We evaluated the eight state-of-the-art background modeling methods discussed in Section 3. We tested these different background modeling methods and did not use any pre-processing and post-processing schemes. We also present the amount of memory as well as the computational time used by each method.

First, we present the experimental results of evaluated methods on the CDnet2014 video dataset and another video dataset in Figs. 2 and 3. The scenarios used to evaluate different methods contain bad weather, baseline, camera jitter, dynamic background, intermittent motion object, low frame rate, PTZ, shadow, thermal, turbulence, fighting, walking I, walking II, and Pets 2006 S7. The thermal video was captured by a far-infrared camera. There are several videos for each scenario. These critical situations have different spatial and temporal properties. We selected one typical frame from each video to represent each video. Fig. 2 (a)–(e) and Fig. 3 (a)–(e) are different videos which were selected from 10 categories in the CDnet2014 dataset. Fig. 2 (f), (g) and Fig. 3 (f), (g) are four videos selected from the dataset proposed in [83]. Fig. 2 (1) and Fig. 3 (1) show the original frame of the video and Fig. 2 (2) and Fig. 3 (2) are the results of the ground truth data. Fig. 2 (3)–(10) and Fig. 3 (3)–(10) are the foreground detection results of state-of-the-art background modeling methods. Table 4 presents seven evaluation metrics of the eight background modeling methods in the video dataset proposed in [83]. Table 5 presents seven evaluation metrics of the eight background modeling methods in the CDnet2014 dataset. The advantage of different methods can be confirmed by the recall, precision, F-measure and other metrics. Which method obtained the higher recall, precision, and F-measure score and the lower PWC, FPR, and FNR was easily found. For each evaluation metric, we give the results of the eight background modeling methods in different scenes via Figs. 4 to 17.

Bad weather: results are depicted in Fig. 2 (a). They indicate that GMM, AGMM, Vibe, and PBAS outperform other methods. It can also be seen from Figs. 9 and 10 that their Precision and F-measure scores are higher than others.

Note that CodeBook and SACON nearly fail to detect the foreground objects in bad weather videos. Though, their precision score is very high, the recall is very low, and thus the F-measure is lower than 25%. Fig. 10 shows the F-measure score of each method.

Baseline: these videos contain a noise-free static background. Fig. 2 (b) shows foreground detection results of every method. The results indicate that all methods successfully detected the foreground objects. It can also be seen that the F-measure score of each method in Fig. 10 is very high, greater than 55%.

Camera jitter: as can be seen from Fig. 2 (c), SACON less effectively deals with camera jitter than the other methods. Fig. 10 also indicates the F-measure score of SACON is lower than 25%, but the scores of other methods are greater than 50%. In addition, SOBS is the best of all methods and its F-measure score is greater than 60%. This shows that the SOBS method can obtain satisfactory foreground detection mask for a camera jitter video.

Dynamic background: by using some dynamic background videos, we reviewed the capability of each background modeling method to cope with uninteresting movements that have to be deemed as background. Segmentation masks in Fig. 2 (d) illustrates that CodeBook, AGMM, SOBS, and PBAS methods are more effective than the other methods when dealing with dynamic backgrounds. Fig. 10 shows that their F-measure score is greater than 50%. SACON shows the worst representation of dynamic backgrounds which hardly detects foreground objects.

Intermittent object motion: Fig. 2 (e) shows the foreground segmentation results of videos with intermittent object motion. In contrast to other methods, AGMM and SOBS can obtain the stable shape of the man. Other background modeling methods such as SACON regard parts or the whole man as the background.

Low frame rate: the quality of all background modeling methods adaption to low frame rate is depicted in Fig. 3 (a). The foreground masks of all methods look close to the ground truth data. Their F-measure scores are greater than 50% (showed in Fig. 10). The most robust methods to low frame rate are GMM, SOBS, Vibe, and PBAS whose F-measure scores are greater than 70%.

PTZ: the PTZ videos were captured by a rotating and zooming camera. Fig. 3 (b) illustrates that hardly any method is able to cope with these videos. Almost all methods fail to correctly detect foreground objects owing to camera rotation or zooming. Fig. 10 shows their F-measure scores are lower than 20%. In addition, the FNR scores of AGMM and PBAS are also greater than 60%. Hence, in order to strengthen the results of detection, the background model needs to be able to quickly respond after the camera rotates or zooms.

Shadow: the methods vary in their capability of classifying shadow pixels as backgrounds. Fig. 10 shows that the SOBS obtains the highest F-measure score of 79%. Additionally, the F-measure scores of AGMM and PBAS are greater than 60%. Fig. 3 (c) also shows the results of SOBS, AGMM, and PBAS. Their foreground masks look better than these of other

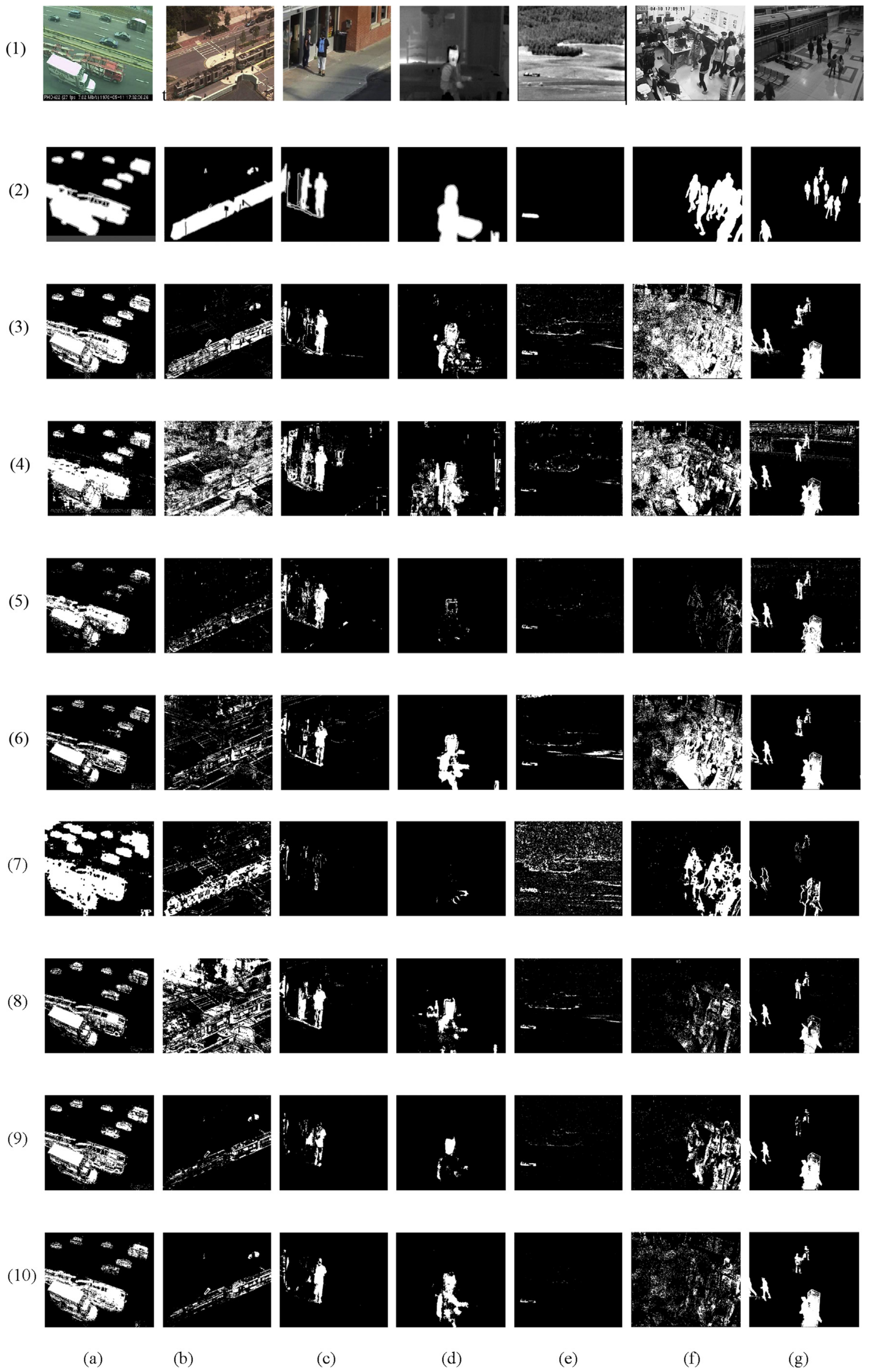


Fig. 3. Foreground detection results of the low frame rate, PTZ, shadow, thermal, and turbulence video from the CDnet2014 dataset, walking II and Pets 2006 S7 from the dataset in [83]. (1) original frame. (2) ground truth. (3) GMM. (4) KDE. (5) CodeBook. (6) AGMM. (7) SACON. (8) SOBS. (9) Vibe. (10) PBAS.

Table 4
Average evaluation metrics of video dataset in [83].

Methods	Recall	Specificity	FPR	FNR	PWC	Precision	F-measure
GMM	0.7569	0.8807	0.1193	0.2432	1.5098	0.4036	0.5196
KDE	0.7169	0.9092	0.0908	0.2831	3.7577	0.3700	0.4654
CodeBook	0.3541	0.9893	0.0108	0.6460	4.2732	0.6688	0.4368
AGMM	0.7368	0.9141	0.0860	0.2632	4.4957	0.5527	0.6151
SACON	0.2095	0.9907	0.0093	0.7906	2.1029	0.6652	0.3044
SOBS	0.4056	0.9927	0.0074	0.5944	3.4970	0.7682	0.5051
ViBe	0.4553	0.9879	0.0122	0.5447	5.5844	0.7793	0.5741
PBAS	0.4848	0.9822	0.0178	0.5145	3.5966	0.6670	0.5472

Table 5
Average evaluation metrics of Change Detection Challenge 2014 dataset.

Methods	Recall	Specificity	FPR	FNR	PWC	Precision	F-measure
GMM	0.6226	0.9687	0.0313	0.3775	3.3599	0.4986	0.4760
KDE	0.6866	0.9195	0.0806	0.3134	4.2119	0.3764	0.4075
CodeBook	0.3855	0.9888	0.0112	0.6145	3.1316	0.6119	0.4105
AGMM	0.5603	0.9719	0.0274	0.4397	2.4064	0.6342	0.5389
SACON	0.3822	0.9449	0.0551	0.6178	5.1920	0.4626	0.2396
SOBS	0.6011	0.9607	0.0393	0.3989	5.1545	0.6666	0.5640
Vibe	0.4651	0.9868	0.0132	0.5349	3.0410	0.6530	0.4718
PBAS	0.5079	0.9920	0.0080	0.4921	2.4110	0.7090	0.5505

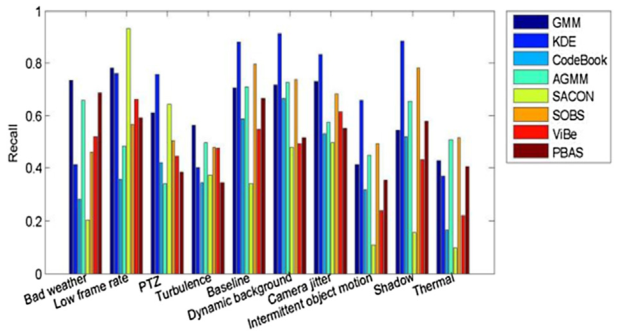


Fig. 4. The recall of different methods.

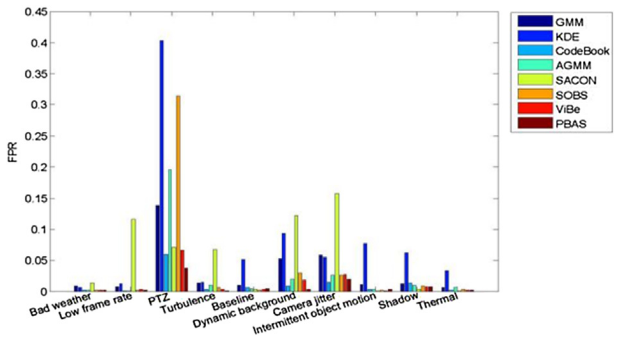


Fig. 6. The FPR of different methods.

methods and are closer to the ground truth data. SACON has the worst detection results in Fig. 3 (c).

Thermal: compared with other methods, AGMM, SOBS, and PBAS can better handle the thermal video captured by a far-infrared camera. As shown in Fig. 3 (d), their results are closer to the ground truth data. Fig. 10 also indicates the F-measure scores are greater than 50%. In addition, SOBS has the best results of detecting foreground objects in thermal videos. SACON has the worst detection results whose F-measure score is lower than 20%.

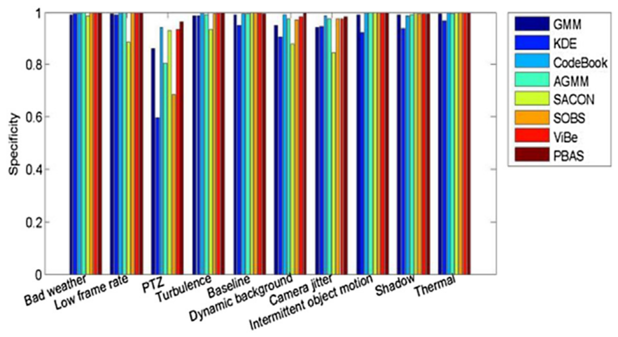


Fig. 5. The specificity of different methods.

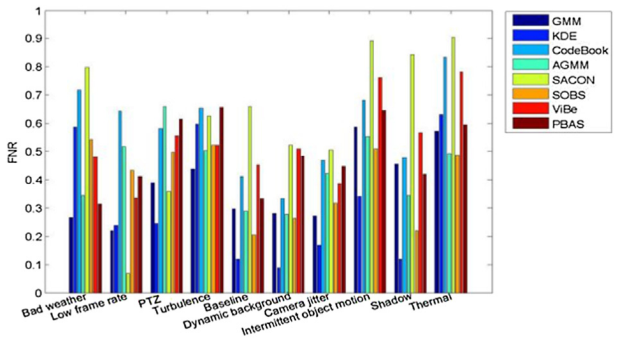


Fig. 7. The FNR of different methods.

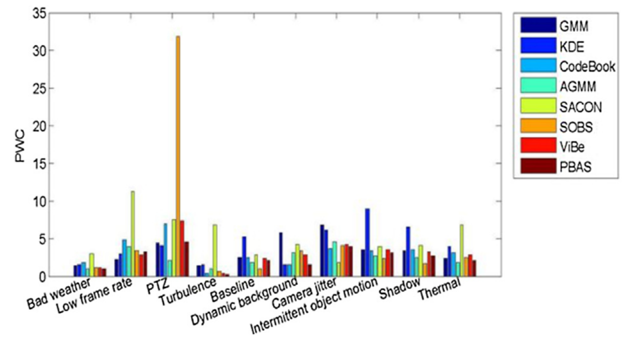


Fig. 8. The PWC of different methods.

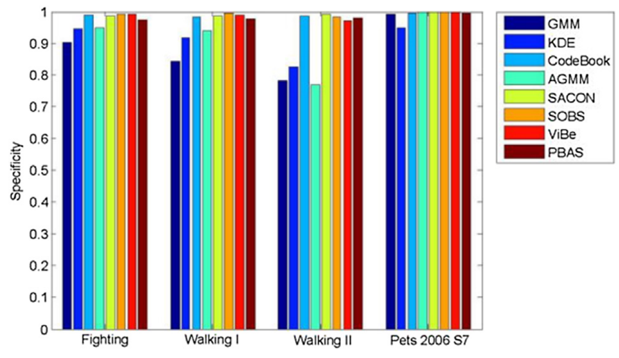


Fig. 12. The specificity of different methods.

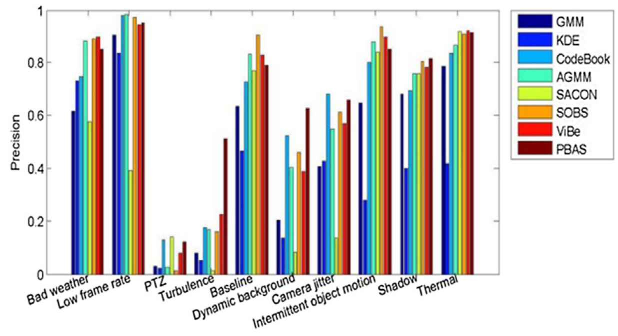


Fig. 9. The precision of different methods.

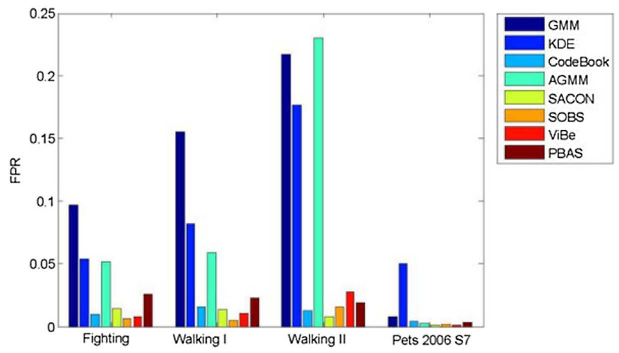


Fig. 13. The FPR of different methods.

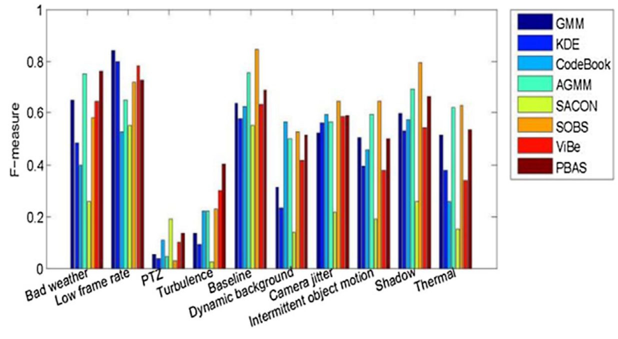


Fig. 10. The F-measure of different methods.

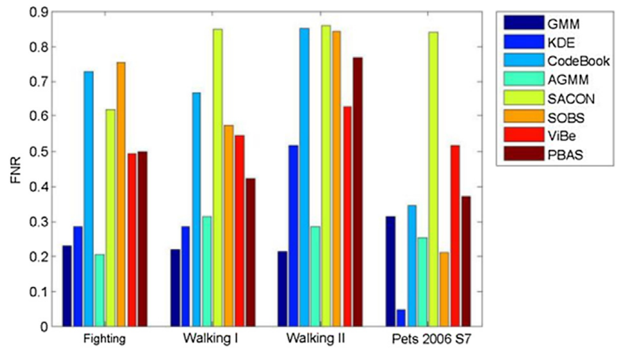


Fig. 14. The FNR of different methods.

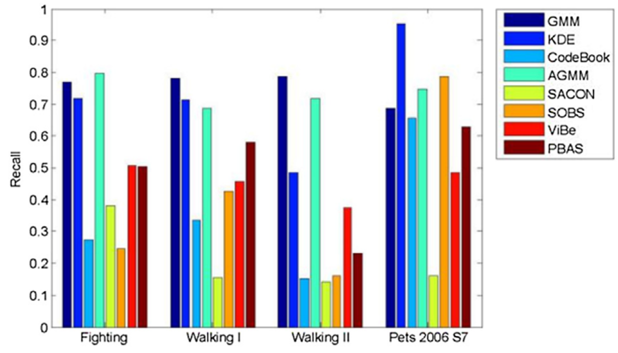


Fig. 11. The recall of different methods.

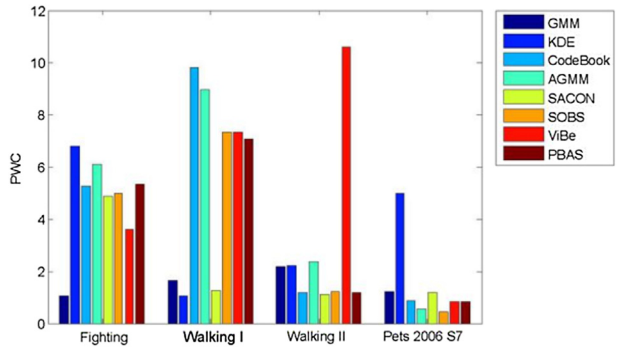


Fig. 15. The PWC of different methods.

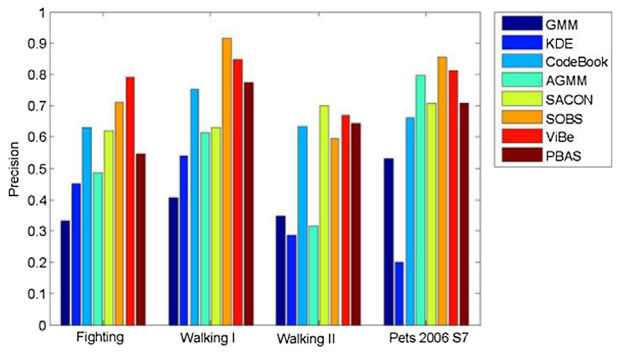


Fig. 16. The precision of different methods.

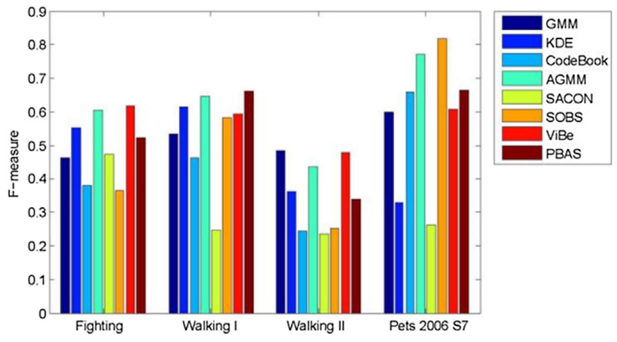


Fig. 17. The F-measure of different methods.

Turbulence: none of the evaluated background modeling methods except for SOBS and PBAS is able to satisfactorily handle turbulence video. As Fig. 3 (e) shows, they absorb the foreground objects into the background or classify the background as foreground pixels. The detection results of SOBS and PBAS have better performance than others. Their F-measure scores are greater than 63% and 53% respectively as shown in Fig. 10.

Video dataset in [83]**:** this dataset contains four indoor video sequences including the large size of occlusion. We run the evaluated methods on this video dataset. Fig. 2 (f), (g) and Fig. 3 (f), (g) show the results of foreground detection. As figures show, SACON and PBAS outperform other methods. Fig. 16 shows they have a higher precision than others. Their precision scores are greater than 66% and the F-measure score of PBAS is greater than 54%.

When selecting a background modeling method, it is important to evaluate its memory consumption and execution time. In our experiments, each method was implemented by C++ with the OpenCV library on an Intel Core i3-3320 computer with a 3.3 GHz CPU and 4 GB memory. We used three videos of different sizes: 320 × 240, 540 × 360, and 720 × 480 to estimate the memory and time comsumption. All videos are 25 fps. Table 6 shows average frames per second of each method and Table 7 shows total memory requirements of every method in a whole project on our computing platform.

6. Conclusion

We evaluated the performance of eight background modeling methods and presented the challenges and comprehensive experimental results. These methods were tested based on their capability of correctly detecting motion as well as their time and memory requirements on various kinds of videos (e.g., indoor/outdoor environments, moving backgrounds, thermal). We used seven evaluation metrics to compare the relative accuracy of the methods. An overall evaluation was presented in Tables 4 and 5. Though each tested method has different advantages and disadvantages, we found that AGMM, SOBS, Vibe, and PBAS were the most promising methods. But it was notable that sophisticated methods do not always produce more precise results, even though they show good performance for many challenges. Considering the future development of background modeling methods, we believe that background modeling methods were able to deal with complex situations, such as “moved objects”, “casted shadows”, and “illumination changes”, and even the best methods of our study are not completely able to handle these challenges. Most background modeling methods can work well on static background, but adapt them to dynamic

Table 6
The comparison of average frames per second (fps).

Size	GMM	KDE	CodeBook	AGMM	SACON	SOBS	Vibe	PBAS
320 × 240	47.9	51.1	53.1	61.8	22.2	9.1	61.8	20.7
540 × 360	21.1	35.1	19.6	53.3	7.8	4.1	52.5	14.6
720 × 480	14.7	9.5	15.2	29.2	4.2	2.3	31.5	8.1

Table 7
Memory requirements (MB).

Size	GMM	KDE	CodeBook	AGMM	SACON	SOBS	Vibe	PBAS
320 × 240	12.3	24.8	231.5	11.4	19.4	15.5	10.8	31.7
540 × 360	25.8	56.7	419.8	23.6	42.3	33.6	20.6	58.8
720 × 480	42.8	96.9	529.4	38.9	71.2	56.3	32.9	89.7

scenes or moving backgrounds is a real challenge in the background modeling field.

Acknowledgment

The work reported in this paper is partly supported by NSFC under grants Nos. 61370163, as well as the Shenzhen Municipal Science and Technology Innovation Council under grant No. JCYJ20140904154630436. Thanks to Dr. Edward C. Mignot, Shandong University for linguistic advive.

References

[1] K. Toyama, J. Krumm, B. Brumitt, B. Meyers, Wallflower: principles and practice of background maintenance, in: International Conference on Computer Vision (ICCV), 1999, pp. 255–261.
[2] A. Sobral, A. Vacavant, Comput. Vis. Image Underst. (2014) 4–21.
[3] X. Wang, K.T. Ma, G.-W. Ng, W.E.L. Grimson, Trajectory analysis and semantic region modeling using a nonparametric Bayesian model, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2008, pp. 1–8.
[4] S. Calderara, R. Cucchiara, A. Prati, A distributed outdoor video surveillance system for detection of abnormal people trajectories, in: Proc. 1st ACM/IEEE Int. Conf. Distrib. Smart Cameras, Sep. 2007, pp. 364–371.
[5] C.-R. Huang, P.-C.J. Chung, D.-K. Yang, H.-C. Chen, G.-J. Huang, IEEE Trans. Circuits Syst. Video Technol. 24 (8) (Aug. 2014) 1417–1429.
[6] C.-R. Huang, H.-C. Chen, P.-C. Chung, Online surveillance video synopsis, in: Proc. IEEE Int. Symp. Circuits Syst, May 2012, pp. 1843–1846.
[7] K.N. Plataniotis, C.S. Regazzoni, IEEE Signal Process. Mag. 22 (2) (2005) 12–15.
[8] S. Ferrando, G. Gera, C. Regazzoni, Classification of unattended and stolen objects in video-surveillance system, in: Proc. of AVSS, 2006, pp. 21–27.
[9] M. Hedayati, Wan Mimi Diyana Wan Zaki, Aini Hussain, J. Comput. Inf. Syst. (2012) pp.493–505.
[10] S. Brutzer, B. Hoferlin, G. Heidemann, Evaluation of background subtraction techniques for video surveillance, in: Computer Vision and Pattern Recognition (CVPR), 2011, pp. pp.1937–1944.
[11] Mustafa Karaman, Lutz Goldmann, Da Yu, Thomas Sikora, Comparison of static background segmentation methods, in: Proc. SPIE, Visual Communications and Image Processing, 2005.
[12] T. Bouwmans, Background subtraction for visual surveillance: a fuzzy approach, in: Handbook on Soft Computing for Video Surveillance, Taylor and Francis Group, 2012.
[13] C. Wren, A. Azarhayejani, T. Darrell, A.P. Pentland, IEEE Trans. Pattern Anal. Mach Intell. 19 (7) (1997) 780–785.
[14] M. Piccardi, IEEE Int. Conf. Syst. Man Cybern 4 (Oct. 2004) 3099–3104. The Hague, The Netherlands.
[15] Y. Benezeth, P.M. Jodoin, B. Emile, H. Laurent, C. Rosenberge, Review and evaluation of commonly-implemented background subtraction algorithms, in: Proc. IEEE Int. Conf. Pattern Recognit, Dec.2008, pp. 1–4.
[16] C. Stauffer, E. Grimson, Adaptive background mixture models for real-time tracking, in: Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit., vol. 2, 1999, pp. 246–252.
[17] C. Stauffer, E. Grimson, IEEE Trans. Pattern Anal. Mach. Intell. 22 (8) (Aug. 2000) 747–757.
[18] Z. Zivkovic, Improved adaptive Gaussian mixture model for background subtraction, in: Proc. 17th Int. Conf. Pattern Recognit., vol. 2, Aug. 2004, pp. 28–31.
[19] D.S. Lee, Pattern Anal Mach. Intell. IEEE Trans. 27 (5) (May 2005) 827–832.
[20] A. Shimada, D. Arita, R.I. Taniguchi, Dynamic control of adaptive mixture-of-Gaussians background model, in: AVSS '06 Proceedings of the IEEE International Conference on Video and Signal Based Surveillance, IEEE Computer Society Washington, DC, USA, 2006.
[21] A. Elgammal, D. Hanvood, L.S. Davis, Non-parametric model for background subtraction, in: Proc. ECCV 2000, June 2000, pp. 751–767.
[22] N.M. Oliver, B. Rosario, A.P. Pentland, IEEE Trans. Pattern Anal. Mach. Zntell. 22 (8) (2000) 831–843.
[23] O. Barnich, M. Van Droogenbroeck, Vibe: a powerful random technique to estimate the background in video sequences, in: IEEE Int. Conf. on Acoustics, Speech and Signal Processing, 2009, pp. 945–948.
[24] O. Barnich, M. Van Droogenbroeck, IEEE Trans. Image Process. 20 (6) (2011) 1709–1724.
[25] Y. Min-Hsiang, H. Chun-Rong, L. Wan-Chen, L. Shu-Zhe, C. Kun-Ta, IEEE Trans. Circuits Syst. Video Technol. 25 (4) (2015) 595–608.
[26] M. Van Droogenbroeck, O. Paquot, Background subtraction: experiments and improvements for ViBe, in: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops, Jun. 2012, pp. 32–37.
[27] M. Hofmann, P. Tiefenbacher, G. Rigoll, Background segmentation with feedback: the pixel-based adaptive segmenter, in: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops, Jun. 2012, pp. 38–43.
[28] K. Kim, T. Chalidabhongse, D. Harwood, L. Davis, Background modeling and subtraction by codebook construction, in: Int. Conf. on Image Processing, 2004, pp. 3061–3064.
[29] K. Kim, T. Chalidabhongse, D. Harwood, L. Davis, Real-Time Imaging 11 (3) (2005) 172–185.
[30] L. Maddalena, A. Petrosino, IEEE Trans. Image Process. 17 (7) (2008) 1168–1177.
[31] Y. Benezeth, P.M. Jodoin, B. Emile, H. Laurent, C. Rosenberge, J. Electron Imaging (July 23, 2010).
[32] A. McIvor, Background subtraction techniques, in: Proc. Image Vis. Comput., Auckland, New Zealand, Nov. 2000.
[33] R. Radke, S. Andra, O. Al-Kofahi, B. Roysam, IEEE Trans. Image Process 14 (Mar. 2005) 294–307.
[34] S. Elhabian, K. El-Sayed, S. Ahmed, Recent Pat. comput. Sci. 1 (Jan. 2008) 32–54.
[35] T. Bouwmans, F. El Baf, B. Vachon, Statistical background modeling for foreground detection: a survey, in: Handbook of Pattern Recognition and Computer Vision, vol. 4, World Scientific, Singapore, Jan. 2010, pp. 181–199 ch. 3.
[36] S. Panahi, S. Sheikhi, S. Hadadan, N. Gheissari, Evaluation of background subtraction methods, in: Proc. Int. Conf. on Digital Image Computing: Techniques and Applications, IEEE, Piscataway, NJ, 2008, pp. 357–364.
[37] D. Parks, S. Fels, Evaluation of background subtraction algorithms with post-processing, in: Proc. of IEEE Int. Conf. on Advanced Video and Signal Based Surveillance, 2008, pp. 192–199.
[38] A. Sobral, BGSLibrary: an opencv c++ background subtraction library, in: IX Workshop de Viso Computacional (WVC'2013). Rio de Janeiro, Brazil, 2013. Software available at: http://code.google.com/p/bgslibrary/.
[39] http://www.changedetection.net.
[40] A. Mittal, N. Paragios, Motion-based background subtraction using adaptive kernel density estimation, in: Proc. Int. Conf. On Computer Vision and Pattern Recognition, IEEE, Piscataway, NJ, 2004, pp. 302–309.
[41] A. Elgammal, R. Duraiswami, D. Harwood, L. Davis, Proc. IEEE 90 (7) (Jul. 2002) 1151–1163.
[42] Y. Sheikh, M. Shah, IEEE Trans. Pattern Anal. Mach. Intell. 27 (11) (Nov. 2005) 1778–1792.
[43] Z. Zivkovic, F. van der Heijden, Pattern Recognit. Lett. 27 (May 2006) 773–780.
[44] A. Tavakkoli, M. Nicolescu, G. Bebis, M. Nicolescu, Mach. Vis. Appl. 20 (Oct. 2008) 395–409.
[45] C. Ianasi, V. Gui, C. Toma, D. Pescaru, Facta Univ. Ser. Elec. Energ. 18 (1) (Apr. 2005) 127–144.
[46] A. Tavakkoli, M. Nicolescu, G. Bebis, An adaptive recursive learning technique for robust foreground object detection, in: ECCV 2006, May 2006.

[47] A. Tavakkoli, M. Nicolescu, G. Bebis, Robust recursive learning for foreground region detection in videos with quasi-stationary backgrounds, in: ICPR 2006, Aug. 2006.
[48] A. Tavakkoli, M. Nicolescu, G. Bebis, Automatic statistical object detection for visual surveillance, in: IEEE Southwest Symposium on Image Analysis and Interpretation, Denver, Colorado, SSIAI 2006, March 2006.
[49] T. Tanaka, A. Shimada, D. Arita, R. Taniguchi, Non-parametric background and shadow modeling for object detection, in: ACCV 2007, Nov. 2007, pp. 159−168.
[50] R. Ramezani, P. Angelov, X. Zhou, A fast approach to novelty detection in video streams using recursive density estimation, in: International IEEE Symposium on Intelligent Systems, vol. 2, Sept. 2008, pp. 142−147.
[51] T. Kohonen, Learning vector quantization, Neural Networks vol. 1, 1988, pp. 3−16.
[52] B.D. Ripley, Pattern Recognition and Neural Networks, Cambridge University Press, 1996.
[53] A. Ilyas, M. Scuturici, S. Miguet, Real time foreground-background segmentation using a modified codebook model, in: Advanced Video and Signal Based Surveillance, 2009.
[54] Z. Zivkovic, F. van der Heijden, IEEE Trans. PAMI 26 (5) (2004).
[55] H. Wang, D. Suter, Pattern Recognit. (2007) 1091−1105.
[56] H. Wang, D. Suter, Background Subtraction Based on a Robust Consensus Method, Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06), pp. 223−226.
[57] L. Maddalena, A. Petrosino, The SOBS algorithm: what are the limits ?, in: CVPRW 2012, June 2012, pp. 21−26.
[58] K.E.A. Sande, T. Gevers, C.G.M. Snoek, IEEE Trans. Pattern Anal. Mach. Intell. 32 (9) (Sep. 2010) 1582−1596.
[59] P.-M. Jodoin, M. Mignotte, J. Konrad, IEEE Trans. Circuits Syst. Video Technol. 17 (12) (Dec. 2007) 1758−1763.
[60] T. Bouwmans, F. El Baf, B. Vachon, Recent Pat. Comput. Sci. 1 (3) (2008) 219−237.
[61] R. Cucchiara, M. Piccardi, A. Prati, IEEE Trans. Pattern Anal. Mach. Intell. 25 (10) (Oct. 2003) 1−6.
[62] E.L. Hall, Computer Image Processing and Recognition, Academic Press, 1979.
[63] S.-Y. Chien, W.-K. Chan, Y.-H. Tseng, H.-Y. Chen, IEEE Trans. Circuits Syst. Video Technol. 23 (6) (Jun. 2013) 921−934.
[64] M. Seki, T. Wada, H. Fujiwara, K. Sumi, Background subtraction based on cooccurrence of image variations, in: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit, Jun. 2003, pp. II-65−II-72.
[65] D. Russell, S. Gong, A highly efficient block-based dynamic background model, in: Proc. IEEE Conf. Adv. Video Signal Based Surveill, Sep. 2005, pp. 417−422.
[66] H.-L. Eng, J. Wang, A.H.K.S. Wah, W.-Y. Yau, IEEE Trans. Image Process. 15 (6) (Jun. 2006) 1583−1600.
[67] M. Heikkilä, M. Pietikäinen, IEEE Trans. Pattern Anal. Mach. Intell. 28 (4) (Apr. 2006) 657−662.
[68] W.-C. Liu, S.-Z. Lin, M.-H. Yang, C.-R. Huang, Real-time binary descriptor based background modeling, in: Proc. IAPR Asian Conf. Pattern Recognit, Nov. 2013, pp. 722−726.
[69] G.-H. Huang, C.-R. Huang, Binary invariant cross color descriptor using galaxy sampling, in: Proc. 21st Int. Conf. Pattern Recognit, Nov. 2012, pp. 2610−2613.
[70] Y. Nonaka, A. Shimada, H. Nagahara, R. Taniguchi, Evaluation report of integrated background modeling based on spatio-temporal features, in: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops, Jun. 2012, pp. 9−14.
[71] S.-S. Huang, L.-C. Fu, P.-Y. Hsiao, IEEE Trans. Circuits Syst. Video Technol. 19 (4) (Apr. 2009) 522−532.
[72] T.H. Tsai, C.-Y. Lin, S.-Y. Li, IEEE Trans. Circuits Syst. Video Technol. 23 (1) (Jan. 2013) 15−29.
[73] A. Schick, M. Bauml, R. Stiefelhagen, Improving foreground segmentations with probabilistic superpixel Markov random fields, in: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops, Jun. 2012, pp. 27−31.
[74] Y.-T. Chen, C.-S. Chen, C.-R. Huang, Y.-P. Hung, Pattern Recognit. 40 (10) (Oct. 2007) 2706−2715.
[75] M. Cristani, M. Bicego, V. Murino, Integrated region- and pixel-based approach to background modeling, in: Proc. Workshop Motion Video Comput, Dec. 2002, pp. 3−8.
[76] Y. Wang, P.-M. Jodoin, F. Porikli, J. Konrad, Y. Benezeth, P. Ishwar, Cdnet 2014: an expanded change detection benchmark dataset, in: Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on. IEEE, 2014, pp. 393−400.
[77] T. Ojala, M. Pietikäinen, T. Mäenpää, IEEE Trans. Pattern Anal. Mach. Intell. 24 (7) (Jul. 2002) 971−987.
[78] Q. Zhu, S. Avidan, K.-T. Cheng, Learning a sparse, corner-based representation for time-varying background modeling, in: Proc. 10th IEEE Int. Conf. Comput. Vis., vol. 1, Oct. 2005, pp. 678−685.
[79] C. Harris, M. Stephens, A combined corner and edge detector, in: Proc. 4th Alvey Vis. Conf, 1988, pp. 147−151.
[80] D.G. Lowe, Int. J. Comput. Vis. 60 (2) (Nov. 2004) 91−110.
[81] S.-C.S. Cheung, C. Kamath, Robust techniques for background subtraction in urban traffic video, in: SPIE04, 5308, 2004, pp. 881−892.
[82] S.-C.S. Cheung, C. Kamath, Robust background subtraction with foreground validation for urban traffic video, in: JASP05, 14, 2005, pp. 1337−2340.
[83] Jiajun Wen, Yong Xu, Jinhui Tang, Yinwei Zhan, Zhihui Lai, Xiaotang Guo. Joint video frame set division and low-rank decomposition for background subtraction, IEEE Trans. Circuits Syst. Video Technology.