Journal of Electronic Imaging 19(3), 033003 (Jul–Sep 2010)

Comparative study of background subtraction algorithms

Yannick Benezeth
Orange Labs France Telecom R&D
4, rue du Clos Courtel
Cesson Sévigné Cedex 35512, France

Pierre-Marc Jodoin
Université de Sherbrooke
Centre de Recherche MOIVRE
2500 Boulevard de l'Université
Sherbrooke, Quebec J1K 2R1, Canada

Bruno Emile
Hélène Laurent
Université d'Orléans
Institut PRISME
88 Boulevard Lahitolle
18020 Bourges, France

Christophe Rosenberger
Université de Caen—CNRS
GREYC, ENSICAEN
6 Boulevard Maréchal Juin
14000 Caen, France

Abstract. *We present a comparative study of several state-of-the-art background subtraction methods. Approaches ranging from simple background subtraction with global thresholding to more sophisticated statistical methods have been implemented and tested on different videos with ground truth. The goal is to provide a solid analytic ground to underscore the strengths and weaknesses of the most widely implemented motion detection methods. The methods are compared based on their robustness to different types of video, their memory requirements, and the computational effort they require. The impact of a Markovian prior as well as some postprocessing operators are also evaluated. Most of the videos used come from state-of-the-art benchmark databases and represent different challenges such as poor SNR, multimodal background motion, and camera jitter. Overall, we not only help to better understand for which type of videos each method best suits but also estimate how, sophisticated methods are better compared to basic background subtraction methods.* © *2010 SPIE and IS&T.* [DOI: 10.1117/1.3456695]

1 Introduction

For various computer vision applications, background subtraction (BS) is a "quick and dirty" way of localizing moving objects in a video shot by a static camera. In this perspective, motion detection is often the first step of a multistage computer vision system[1–4] (car tracking, person recognition, wild-life monitoring, etc.). For this reason, it is usually required to be as fast and as simple as possible. Consequently, most BS methods label "in motion" every pixel at time t whose color is significantly different from the ones in the background.[5] This solution has proven successful whenever the camera is rigorously static with a fixed noise-free background (see Ref. 6 for some examples).

But detecting motion through BS is not always as easy as it may first appear. Indeed, some videos with poor SNRs, caused by a low-quality camera, compression artifacts, or a noisy environment, are likely to generate numerous false positives. False positives can also be induced by illumination changes (gradual or sudden), an animated background (waves on the water, trees shaken by the wind), or camera jitter to name a few. On the other hand, false negatives can also occur when a moving object is made of colors similar to the ones in the background (the so-called camouflage effect). With such scenarios, a simple interframe difference with a global threshold reveals itself as a weak solution. To cope with these challenges, numerous background models and distance measures bound up to different optimization schemes have been proposed in the past decade. These methods are (at least in theory) more robust to noise and background instability than the basic BS approaches. But

Paper 09200R received Oct. 15, 2009; revised manuscript received Apr. 26, 2010; accepted for publication May 12, 2010; published online Jul. 23, 2010.

1017-9909/2010/19(3)/033003/12/$25.00 © 2010 SPIE and IS&T.

are they really? And if they are, how much better are they? Are they suitable for real-time applications? Can they be implemented on a lightweight architecture?

In this paper, we compare some of the most implemented BS methods on various real, synthetic, and semi-synthetic video sequences representing different challenges. The goal of this study is threefold:

1. evaluate how better sophisticated methods are compared to simple BS methods
2. compare the processing power and the amount of memory required by each method at runtime
3. determine for which type of video each method suits best

As BS is widely used in computer vision, numerous surveys and comparative studies have been published. While some of those papers contain descriptive evaluations of motion detection methods,[7] others provide quantitative evaluations based on preannotated video sequences. It is the case with Toyama *et al.*[8] and Panahi *et al.*,[9] who conducted comparative studies for various pixel-based BS algorithms. In both papers, the BS methods are defined using a single set of parameters and then executed on various video sequences. The BS methods are then-compared based on the overall number of false negatives (FNs) and false positives (FPs) they produced in each video sequence. Although FNs and FPs are typical quality measures, they are nonetheless strongly dependent: when FNs decrease, FPs always increase in return and vice versa. Thus, a single couple {FN, FP} is not sufficient to compare BS methods together as a method with large FNs and low FPs is not necessarily better or worse than one with low FNs and large FPs. Moreover, the FN and FP values given in those surveys were obtained with a predefined threshold per method, which leaves us to think that performances could be increased by further tweaking the thresholds. In a similar way, Herrero and Bescòs[10] use a single couple {Precision, Recall} to compare BS algorithms. This approach however suffers from the same limitations as the ones based on {FN, FP}.

Chalidabhongse *et al.*[11] proposed a different way to compare BS methods based on a so-called analysis of disturbances. In a first stage, the FP rate in a learning video sequence is fixed after adjusting some *ad hoc* thresholds. Then, the background of each video sequence is corrupted by a vector of disturbance in all directions of the *RGB* space. Such corruption simulates foreground moving objects. The ability of a BS algorithm to detect low-contrast targets against the background is measured as a function of contrast. The main advantage of this method is its ability to use all kinds of videos for quality assessment, even those without ground truth. Unfortunately though, this method is not devoid of drawbacks. While the pixel distribution of a foreground pixel is usually unimodal, the analysis of disturbances method with multimodal backgrounds involves that the simulated foreground moving objects are a combination of the multimodal distribution and the disturbance. Also, this method allows neither the evaluation of region-based methods nor the benefits of postprocessing tools. Moreover, a few methods are compared in their paper. We extend the comparison to seven in this paper.

Other surveys evaluate BS methods in the context of target detection such as car and pedestrian localization.[12,13] These surveys focus on object-based motion detection methods for which connected foreground pixels are grouped together into moving blobs. Then the position of these blobs is used for the evaluation. The main problem with object-based approaches and connected component analysis is their fundamental inability to deal with occlusion. Indeed, when a moving object partially occludes another one, both are connected together into one large moving blob. In this case, the moving objects can only be separated via a high-level postprocessing stage. This is a typical cause for large FP and FN rates. Moreover, the postprocessing stage in Ref. 13 used to clean up the motion mask is not the same for every method, so the comparison may be biased in favor of some methods.

In this paper, we chose to conduct a comparative study whose focus is significantly different from the ones published so far. The key points of our study are the following:

1. The interdependence between FPs and FNs is considered so the evaluation is fair for every method.
2. The video data set is split into groups of videos containing similar features and representing similar levels of difficulty.
3. The study focuses on frequently implemented pixel-based motion detection methods.
4. Each BS method is evaluated with and without the same spatial aggregation, be it a low-pass filter or a Markovian prior.

The paper is organized as follows. Section 2 describes seven commonly implemented motion detection methods in detail. The protocol used for the comparison, including the video data set, is introduced in Sec. 3, while results and conclusion are presented in Secs. 4 and 5.

2 Background Subtraction Algorithms

Although different, most BS techniques share a common denominator: they make the assumption that the observed video sequence I is made of a static background B in front of which moving objects are observed. With the assumption that every moving object is made of a color (or a color distribution) different from the one observed in B, numerous BS methods can be summarized by the following formula:

$$\mathcal{X}_t(s) = \begin{cases} 1 & \text{if } d(\mathbf{I}_{s,t}, \mathbf{B}_s) > \tau \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

where τ is a threshold, χ_t is the motion label field at time t (also called motion mask), d is the distance between $\mathbf{I}_{s,t}$, the color at time t and pixel s, and $\mathbf{B}_s$, the background model at pixel s. The main difference between several BS methods is how B is modeled and which distance metric d they use. In the following subsection, various BS techniques are presented as well as their respective distance measure.

2.1 *Basic Motion Detection (Basic)*

The easiest way to model the background B is through a single gray-scale/color image void of moving objects. This image can be a picture taken in absence of motion or esti-

mated via a temporal median filter.[5,14,15] To cope with illumination changes and background modifications, it can be iteratively updated as follows:

$$\mathbf{B}_{s,t+1} = (1-\alpha)\mathbf{B}_{s,t} + \alpha \cdot \mathbf{I}_{s,t} \tag{2}$$

where α is a constant whose value ranges between 0 and 1. With this simple background model, pixels corresponding to foreground moving objects can be detected by thresholding any of those distance functions:

$$d_0 = |I_{s,t} - B_{s,t}|, \tag{3}$$

$$d_1 = |I^R_{s,t} - B^R_{s,t}| + |I^G_{s,t} - B^G_{s,t}| + |I^B_{s,t} - B^B_{s,t}|, \tag{4}$$

$$d_2 = (I^R_{s,t} - B^R_{s,t})^2 + (I^G_{s,t} - B^G_{s,t})^2 + (I^B_{s,t} - B^B_{s,t})^2, \tag{5}$$

$$d_\infty = \max\{|I^R_{s,t} - B^R_{s,t}|, |I^G_{s,t} - B^G_{s,t}|, |I^B_{s,t} - B^B_{s,t}|\}, \tag{6}$$

where R, G, and B stand for the red, green, and blue channels and d_0 is a measure operating on gray-scale images.

Note that it is also possible[16] to use the previous frame I_{t-1} as background image B. With this configuration though, motion detection becomes an interframe change detection process, which is robust to illumation changes but suffers from a severe aperture problem since only parts of the moving objects are detected.

2.2 *One Gaussian (1-G)*

Modeling B with a single image, as in Sec. 2.1, requires a rigorously fixed background void of noise and artifacts. Since this requirement cannot be satisfied in every real-life scenario, many authors model each background pixel with a probability density function (pdf) learned over a series of training frames. In this case, the BS problem becomes a pdf-thresholding issue for which a pixel with low probability is likely to correspond to a foreground moving object. For instance, to account for noise, Wren *et al.*[17] model every background pixel with a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_{s,t}, \boldsymbol{\Sigma}_{s,t})$, where $\boldsymbol{\mu}_{s,t}$ and $\boldsymbol{\Sigma}_{s,t}$ stand for the average background color and covariance matrix at pixel s and time t. In this context, the distance metric can be the following log likelihood:

$$d_G = \frac{1}{2}\log[(2\pi)^3|\boldsymbol{\Sigma}_{s,t}|] + \frac{1}{2}(\mathbf{I}_{s,t} - \boldsymbol{\mu}_{s,t})\boldsymbol{\Sigma}^{-1}_{s,t}(\mathbf{I}_{s,t} - \boldsymbol{\mu}_{s,t})^T, \tag{7}$$

or a Mahalanobis distance:

$$d_M = |\mathbf{I}_{s,t} - \boldsymbol{\mu}_{s,t}|\boldsymbol{\Sigma}^{-1}_{s,t}|\mathbf{I}_{s,t} - \boldsymbol{\mu}_{s,t}|^T. \tag{8}$$

Since the covariance matrix contains large values in noisy areas and low values in more stable areas, $\boldsymbol{\Sigma}$ makes the threshold locally dependent on the amount of noise. In other words, the noisier a pixel is, the larger the temporal gradient $|\mathbf{I}_{s,t} - \boldsymbol{\mu}_{s,t}|$ must be to get the pixel labeled in motion. This makes the method significantly more flexible than the basic motion detection one.

Since the illumination often changes in time, the mean and covariance of each pixel can also be iteratively updated following this procedure:

$$\boldsymbol{\mu}_{s,t+1} = (1-\alpha)\cdot\boldsymbol{\mu}_{s,t} + \alpha\cdot\mathbf{I}_{s,t}, \tag{9}$$

$$\boldsymbol{\Sigma}_{s,t+1} = (1-\alpha)\cdot\boldsymbol{\Sigma}_{s,t} + \alpha\cdot(\mathbf{I}_{s,t} - \boldsymbol{\mu}_{s,t})(\mathbf{I}_{s,t} - \boldsymbol{\mu}_{s,t})^T. \tag{10}$$

Note that even if $\boldsymbol{\Sigma}$ is by definition a 3×3 matrix, it can be assumed to be diagonal to reduce memory and processing costs. Other adaptation schemes have been proposed, some working at the pixel level, others at the blob level,[18] and others being robust to shadows.[19]

2.3 *Minimum, Maximum and Maximum Interframe Difference (MinMax)*

Another method whose goal is to locally adapt to noise is the W^4 system by Haritaoglu *et al.*[20] Here, every background pixel s comes with a minimum m_s, a maximum M_s, and a maximum of consecutive frames difference D_s observed over a training sequence. The MinMax method labels "static" every pixel s whose value $I_{s,t}$ satisfies the following criteria:

$$|M_s - I_{s,t}| < \tau d_\mu \quad \text{or} \quad |m_s - I_{s,t}| < \tau d_\mu, \tag{11}$$

where τ is a user-defined threshold, and d_μ is the median of the largest interframe absolute difference over the entire image. Similarly to the 1-G method, a pixel in a noisy area requires a larger variation to be labeled in motion than a pixel in a stable area. In this case though, each background pixel is associated with three extremum values instead of a mean vector and a covariance matrix. The original algorithm operates only on gray-scale videos, which results in a loss of information compared to color video sequences. The authors mention that the background can be updated following a pixel-based and an object-based method.

2.4 *Gaussian Mixture Model (GMM)*

To account for backgrounds made of animated textures (such as waves on the water or trees shaken by the wind), some authors proposed the use of multimodal pdfs. Stauffer and Grimson's method,[21] for example, models every pixel with a mixture of K Gaussians. For this method, the probability of occurrence of a color at a given pixel s is given by

$$P(\mathbf{I}_{s,t}) = \sum_{i=1}^{K} \omega_{i,s,t}\mathcal{N}(\boldsymbol{\mu}_{i,s,t}, \boldsymbol{\Sigma}_{i,s,t}), \tag{12}$$

where $\mathcal{N}(\boldsymbol{\mu}_{i,s,t}, \boldsymbol{\Sigma}_{i,s,t})$ is the i'th Gaussian model, and $\omega_{i,s,t}$ is its weight. Note that for computational purposes, as suggested by Stauffer and Grimson, the covariance matrix $\boldsymbol{\Sigma}_{i,s,t}$ can be assumed to be diagonal, $\boldsymbol{\Sigma} = \sigma^2\mathbf{Id}$. In their method, parameters of the matched component (i.e., the nearest Gaussian for which $\mathbf{I}_{s,t}$ is within 2.5 standard deviations of its mean) are updated as follows:

$$\omega_{i,s,t} = (1-\alpha)\omega_{i,s,t-1} + \alpha, \tag{13}$$

$$\boldsymbol{\mu}_{i,s,t} = (1-\rho)\cdot\boldsymbol{\mu}_{i,s,t-1} + \rho\cdot\mathbf{I}_{s,t}, \tag{14}$$

$$\sigma_{i,s,t}^2 = (1-\rho)\cdot\sigma_{i,s,t-1}^2 + \rho\cdot d_2(\mathbf{I}_{s,t},\boldsymbol{\mu}_{i,s,t}), \tag{15}$$

where α is a user-defined learning rate, ρ is a second learning rate defined as $\rho=\alpha\mathcal{N}(\boldsymbol{\mu}_{i,s,t},\boldsymbol{\Sigma}_{i,s,t})$, and d_2 is the distance defined in Eq. (5). Parameters μ and σ of unmatched distributions remain the same, while their weight is reduced as follows: $\omega_{i,s,t}=(1-\alpha)\omega_{i,s,t-1}$ to achieve decay. Whenever no component matches a color $\mathbf{I}_{s,t}$, the one with the lowest weight is replaced by a Gaussian with mean $\mathbf{I}_{s,t}$, a large initial variance σ_0^2, and a small weight ω_0. Once every Gaussian has been updated, the K weights $\omega_{i,s,t}$ are normalized so they sum up to 1. Then, the K distributions are ordered based on a fitness value $\omega_{i,s,t}/\sigma_{i,s,t}$, and only the H most reliable ones are chosen as part of the background:

$$H = \underset{h}{\operatorname{argmin}}\left(\sum_{i=1}^{h}\omega_i > \tau\right), \tag{16}$$

where τ is a threshold. Then, those pixels whose color $\mathbf{I}_{s,t}$ is located at more than 2.5 standard deviations away from every H distributions are labeled "in motion."

Many authors have proposed improvements of this method. For example, in Refs. 22 and 23, new updating algorithms used to learn mixture models are presented.

2.5 *Kernel Density Estimation (KDE)*

An unstructured approach can also be used to model a multimodal pdf. In this perspective, Elgammal *et al.*[24] proposed a Parzen-window estimate at each background pixel:

$$P(I_{s,t}) = \frac{1}{N}\sum_{i=t-N}^{t-1} K(I_{s,t}-I_{s,i}), \tag{17}$$

where K is a kernel (typically a Gaussian), and N is the number of previous frames used to estimate $P(.)$. When dealing with color video frames, products of 1-D kernels can be used:

$$P(\mathbf{I}_{s,t}) = \frac{1}{N}\sum_{i=t-N}^{t-1}\prod_{j=\{R,G,B\}} K\left(\frac{I_{s,t}^j - I_{s,i}^j}{\sigma_j}\right). \tag{18}$$

A pixel is labeled as foreground if it is unlikely to come from this distribution, i.e., when $P(\mathbf{I}_{s,t})$ is smaller than a predefined threshold. Note that σ_j can be fixed or preestimated following Elgammal *et al.*'s method.[24] More sophisticated methods can also be envisaged such as Mittal and Paragios's,[25] which is based on "variable bandwidth kernels."

2.6 *Codebook* (CB_{RGB})

Another approach whose goal is to cope with multimodal backgrounds is the so-called codebook method by Kim *et al.*[26] Based on a training sequence, the method assigns to each background pixel a series of key color values (called code words) stored in a codebook. These code words describe which color a pixel is likely to take over a certain period of time. For instance, a pixel in a stable area may be summarized by only one code word, whereas a pixel located over a tree shaken by the wind could be, for example, summarized by three values: green for the foliage, blue for the sky, and brown for the bark. With the assumption that shadows correspond to brightness shifts and real foreground moving objects to chroma shifts, the original version of the method was designed to eliminate FPs caused by illumination changes. This is done by performing a separate evaluation of color distortion:

$$\left[I_{s,t}^{R^2}+I_{s,t}^{G^2}+I_{s,t}^{B^2} - \frac{(\mu_{i,s}^R\cdot I_{s,t}^R + \mu_{i,s}^G\cdot I_{s,t}^G + \mu_{i,s}^B\cdot I_{s,t}^B)^2}{\mu_{i,s}^{R^2}+\mu_{i,s}^{G^2}+\mu_{i,s}^{B^2}}\right]^{1/2} < \tau \tag{19}$$

and brightness distortion:

$$\alpha_{i,s} \leqslant I_{s,t}^{R^2}+I_{s,t}^{G^2}+I_{s,t}^{B^2} \leqslant \beta_{i,s}, \tag{20}$$

where $\mu_{i,s}^R$, $\mu_{i,s}^G$, $\mu_{i,s}^B$, $\alpha_{i,s}$, and $\beta_{i,s}$ are parameters of the i'th code word of pixel s, and τ is a threshold. Whenever a pixel s satisfies Eqs. (19) and (20), it indicates that the pixel matches the i'th code word and thus is labeled "static."

However, we empirically observed that such a chroma shift assumption is far too restrictive for some urban scenes and produces a large number of FNs. For instance, when monitoring traffic scenes, only color moving objects are correctly picked up while dark gray cars are falsely associated to shadows and white vehicles to sudden increase of intensity. Since this drawback over penalizes the codebook approach on some of our video sequences, we made a slight modification to the original method.

In our implementation, each code word is an *RGB* Gaussian distribution. Based on an N-frame long training sequence and $C_s=\{c_{1,s},\ldots,c_{L,s}\}$, a codebook associated to pixel s made of L code words, each code word $c_{i,s}$ is a Gaussian defined by a mean $\boldsymbol{\mu}_{i,s}$ and a covariance matrix $\boldsymbol{\Sigma}_{i,s}$ (which is assumed to be diagonal). Those Gaussian parameters as well as their number are estimated during a training phase. During that phase, the codebook of each pixel is initialized with its color at time 0, i.e., $\boldsymbol{\mu}_{1,s}=\mathbf{I}_{s,0}$ and $\boldsymbol{\Sigma}_{1,s}=\sigma_0^2\cdot\mathbf{Id}$, where σ_0^2 is a constant and $\mathbf{Id}$ is the identity matrix. Then, each new color $\mathbf{I}_{s,t}$ is compared with the preestimated code words $c_{i,s}$ [cf. Eq. (8)] and, for each match, the associated codebook's parameters are updated following Eqs. (9) and (10). Whenever $\mathbf{I}_{s,t}$ has no match in the codebook, a new code word $c_{j,s}$ is created and initialized as follows: $\boldsymbol{\mu}_{j,s}=\mathbf{I}_{s,t}$ and $\boldsymbol{\Sigma}_{j,s}=\sigma_0^2\cdot\mathbf{Id}$. During the detection phase, each pixel is classified based on its code word as follows:

$$\mathcal{X}_t(s) = \begin{cases} 1 & \text{if } d_M(\mathbf{I}_{s,t},c_{i,s}) > \tau \ \ \forall\, i \\ 0 & \text{otherwise,} \end{cases} \tag{21}$$

where τ is a threshold, and d_M is the Mahalanobis distance.

2.7 *Eigen Backgrounds (Eigen)*

A non-pixel-level method was proposed by Oliver *et al.*[27] that uses an eigenspace to model the background. The key element of this method lies in its ability of learning the background model from unconstraint video sequences, even when they contain moving foreground objects. While previous approaches use pixel-level statistics, Eigen takes into account neighboring statistics. It thus has a more global definition on background, which, hopefully, makes it more robust to unstable backgrounds.

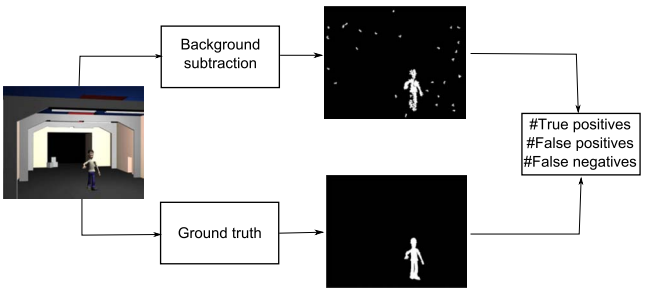

Fig. 1 Overview of the experimental framework.

Let $\{I_i\}_{i=1:N}$ be a column representation of the N-frames-long training sequence. The mean μ can be simply calculated with $\mu=(1/N)\Sigma_{i=1}^{N} I_i$ and then subtracted with each input image to build a zero-mean vector $\{X_i\}_{i=1:N}$, where $X_i=I_i-\mu$. Then, the covariance matrix $\mathbf{\Sigma}$ is built with $\mathbf{\Sigma}=E[\mathbf{X}\mathbf{X}^T]$, with $\mathbf{X}=[X_1,\ldots,X_n]$. According to the Karhunen-Loeve transform, we can compute the eigenvector matrix ϕ which diagonalizes the covariance matrix $\mathbf{\Sigma}$:

$$D=\phi\mathbf{\Sigma}\phi^T, \tag{22}$$

where D is the corresponding diagonal matrix. Following a principal component analysis (PCA), a new rectangular matrix ϕ_M is made out of the M eigenvectors with the largest eigenvalues. Once the eigenbackground ϕ_M and the mean μ have been computed, the column representation of each input image I_t is first projected onto the M-dimensional subspace:

$$\mathbf{B}_t=\phi_M(I_t-\mu), \tag{23}$$

and then reconstructed as follows:

$$I_t'=\phi_M^T\mathbf{B}_t+\mu. \tag{24}$$

Finally, foreground moving pixels are detected by computing the distance between the input image I_t and the reconstructed one I_t':

$$\mathcal{X}_t(s)=\begin{cases}1 & \text{if } d_2(I_t,I_t')>\tau\\ 0 & \text{otherwise,}\end{cases} \tag{25}$$

where τ is a threshold, and d_2 is the Euclidian distance. Note that the Eigen decomposition [Eq. (22)] can be quickly computed with a single value decomposition but, unfortunately, ϕ_M is hard to keep up to date as the video streams in. Solutions based on incremental PCA (e.g., Refs. 28 and 29) have been proposed to cope with this drawback.

3 Experimental Protocol

In this section, details on the experimental framework and the video data set are given.

3.1 *Experimental Framework*

The motion detection methods introduced in the previous section are evaluated following the framework shown in Fig. 1 (also see Table 1). Since our goal is to evaluate the ability of each method to correctly detect motion, a ground truth is available for all videos constituting the database enabling the evaluation of the number of true positives (TPs), FPs, and FNs. Those values are combined into a (precision/recall) couple defined as

$$\text{precision}=\frac{\text{TP}}{\text{TP}+\text{FP}} \quad \text{recall}=\frac{\text{TP}}{\text{TP}+\text{FN}}. \tag{26}$$

By definition, a good algorithm is one producing simultaneously a small number of FPs and FNs, i.e., both a high precision and recall value. Since a threshold τ for a method produces a single precision/recall couple per video, 15 different thresholds are used to produce curves. In this way, the comparison between methods is made easier as we do not have to find the best threshold for each method over each video.

We implemented most of the algorithms except for KDE, Eigen, and GMM for which we used OpenCV or C++ code available on line.[24,30] The various settings used for each method are presented in Table 2.

Table 1 Background subtraction methods evaluated in this comparative study.

Methods	Description
Basic	The model is a color image void of moving objects, described in Sec. 2.1.
1-G	The model of each pixel is a Gaussian distribution, described in Sec. 2.2.
MinMax	The model is composed of a Min, a Max, and a Max of interframes difference, described in Sec. 2.3.
GMM	The model of each pixel is a mixture of Gaussians, described in Sec. 2.4.
KDE	The model of each pixel is a nonparametric distribution, described in Sec. 2.5.
CB_{RGB}	The model is composed of a set of Gaussian distributions, described in Sec. 2.6.
Eigen	The model is nonpixel based using an eigenspace, described in Sec. 2.7.

3.2 *Video Data Set*

To gauge performances, BS methods were tested on a wide range of real, synthetic, and semisynthetic video sequences (one example corresponding to each camera viewpoint is presented in Fig. 2). Our data set is composed of 29 video sequences (15 real, 10 semisynthetic, and 4 synthetic) containing between 100 and 3000 frames of size 320×240. We created some synthetic and semisynthetic videos, others were downloaded from the PETS2001 data set,[31] the IBM data set,[32] and the VSSN 2006 competition.[33] The semisynthetic videos are made of synthetic foreground objects (people and cars) moving over a real background. The whole video data set represents both indoor (20 videos) and outdoor scenes (9 videos). Moreover, six videos contain

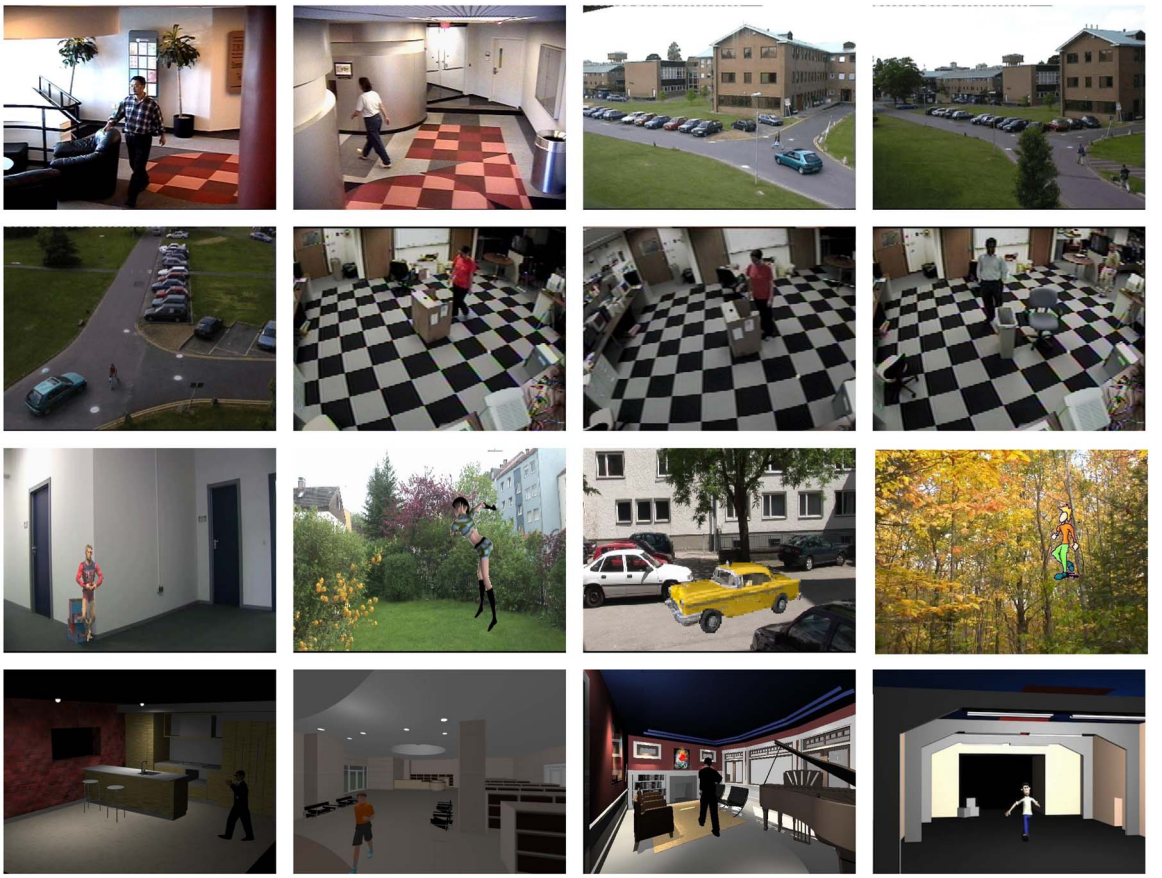

Fig. 2 Snapshots of each camera viewpoint of the video data set.

animated background textures caused by trees and bushes shaken by the wind. While ground truths are easily obtained for synthetic and semisynthetic video sequences, they are available only on some reference images (manually annotated or provided with the data set) for real videos. We therefore use the precise ground truth (in pixels) of each frame for 14 videos and we use the bounding box (about one frame per second) for the other 15 videos.

4 Experimental Results

We tested the BS algorithms described in Sec. 2 on groups of videos illustrating different scenarios and thus different challenges. This section presents benchmarks obtained for each method on videos showing static, noisy, and multimodal backgrounds. We also describe the amount of memory as well as the computational load required by each tech-

Table 2 Parameters used in the evaluation.

Algorithm	Parameters
Basic	distance d_2, $\alpha=10^{-3}$
1-G	distance d_M, $\alpha=10^{-3}$ covariance matrix is diagonal
GMM	$K=3$, $\alpha=10^{-2}$ covariance matrix is diagonal
KDE	$N=100$
CB_{RGB}	distance d_M, $\alpha=10^{-3}$ covariance matrix is diagonal
Eigen	$N=100$, $M=20$

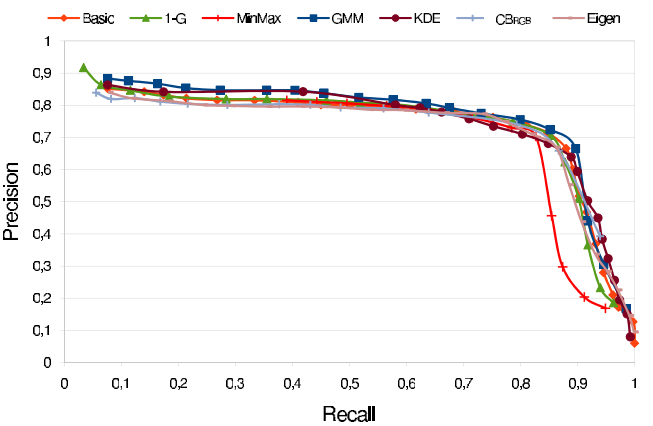

Fig. 3 Test 1: precision/recall curves for noise-free videos with static backgrounds.

(a) (b) (c)

Fig. 4 (a) Input video with a static background and a large SNR, (b) motion mask with Basic, and (c) motion mask with GMM.

nique at runtime. The effect of spatial aggregation such as postprocessing filters and a Markovian prior is also presented.

4.1 *Evaluation of Background Models*

4.1.1 *Videos with a noise-free static background*

The first test aims at evaluating every BS method under ideal conditions, i.e., videos with large SNRs with rigorously static background. Here, a total of 15 videos were used for testing. Results are presented in Fig. 3.

As can be seen from these precision/recall curves, the MinMax method is slightly less effective than the others, mostly because it exclusively works on gray-scale data, thus ignoring color. The other methods globally produce the same results, more or less with a few isolated pixels. Interestingly, the complexity of certain methods such as GMM or KDE does not bring any advantage regarding precision. This clearly suggests that simple methods such as Basic are as efficient as more sophisticated ones when dealing with videos shot in good conditions. This is clearly illustrated in Fig. 4 in which results obtained with Basic and GMM show barely any differences. In fact, every method, be it simple or not, fails to detect regions of the moving objects whose color is similar to the background (look at the hip of the pedestrian in Fig. 4). This is a typical camouflage effect with which no BS method is capable of coping.

4.1.2 *Videos with a multimodal background*

This test aims at evaluating the robustness of each method on videos exhibiting an animated background. Here six video sequences with strong background motion caused by trees and shrubs shaken by the wind are used. Results are presented in Fig. 5.

In this case, the results are significantly more heterogenous than they were in test 1. As one would expect, the simple Basic and MinMax methods are strongly penalized by this test as their global and nonadaptative thresholds do not suit animated backgrounds. Note that the gray-scale nature of MinMax again penalizes it, as it appears to be the weakest method. Surprisingly though, Eigen underperformed on this test, thus suggesting that PCA might not be as efficient as it might first appear on those kinds of videos. On the other hand, results obtained with the 1-G method are surprisingly good despite its unimodal nature. This can be explained by the fact that the 1-G threshold is locally weighted by a covariance matrix, which compensates for background instabilities. Thanks to their multimodal shape, the KDE, GMM, and CB_{RGB} methods produced the most accurate results. Figure 6 presents seven motion masks so the reader can visualize the differences between the BS methods. The difference between Basic, MinMax, and the other methods is clear.

4.1.3 *Noisy videos*

The third test aims at evaluating the influence of noise. Here, 15 videos corrupted with additive Gaussian noise were used for testing. This type of distortion often occurs when working with low-quality cameras or when filming dark areas. Results are presented in Fig. 7.

The MinMax method does not seem to be well suited to noisy videos either. This is explained by the fact that the MinMax threshold (which is global) depends on the maximum interframe difference (which is large for noisy videos) and thus is prone to generate FPs. As for the Basic method, its fixed global threshold significantly penalizes its performances. Statistical methods such as 1-G, GMM, KDE, or CB_{RGB} all gave better results, especially GMM.

4.2 *Evaluation of the Distance Metrics*

As mentioned in Secs. 2.1 and 2.2, various distance metrics can be implemented together with a simple BS method. To evaluate how good those distances are, we tested it on 12 videos containing animated backgrounds and low SNRs. Results are presented in Fig. 8.

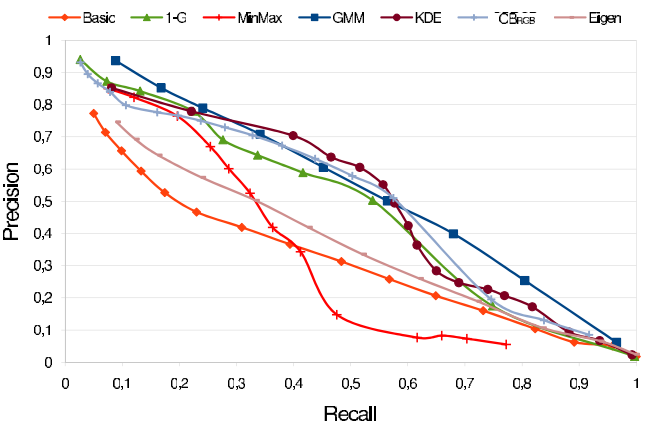

Fig. 5 Test 2: precision/recall curves for videos with multimodal backgrounds.

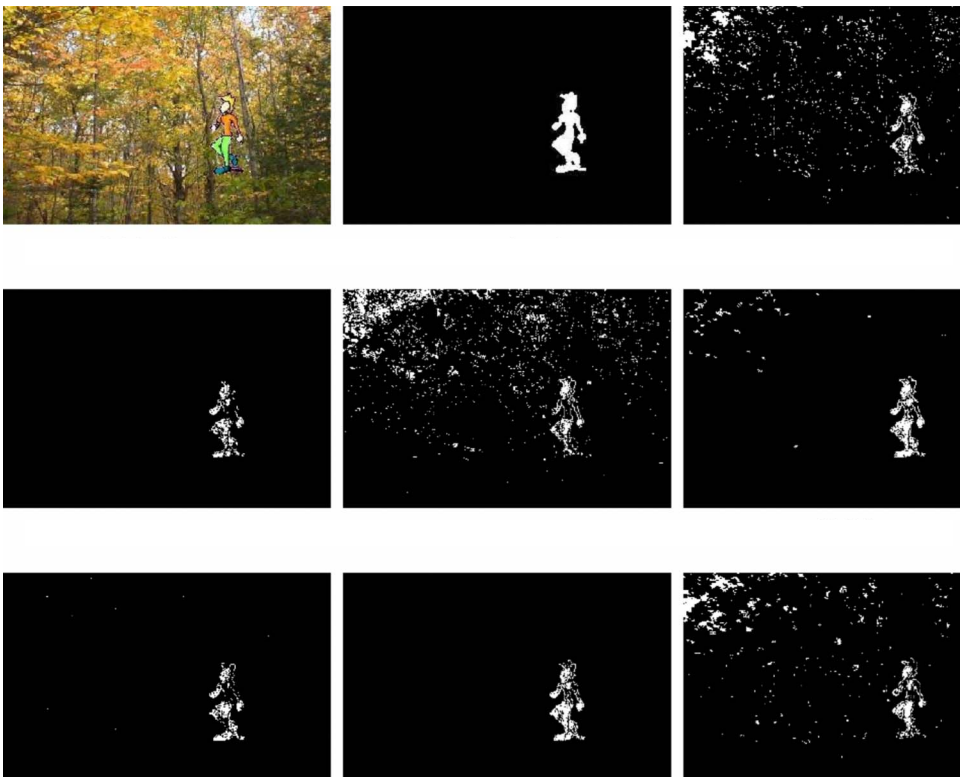

Fig. 6 Motion masks obtained with a video containing a multimodal background.

Out of these results, it turns out that d_0 is slightly less effective than the other ones, while d_M and d_G clearly step out. This can be explained by the fact that d_0 works only on gray-scale data and thus ignores chromacity. This leads d_0 to be more sensitive to camouflage effects (many objects with different color have the same grayscale). For d_M and d_G, their ability of locally adapting to noise makes them more robust to instabilities.

4.3 *Influence of Spatial Aggregation*

Regions in the motion mask are often assumed to be of compact shape with smooth boundaries, small isolated regions being associated to noisy specks. One typical way to improve the motion mask is through a spatial aggregation stage. A stage that would eliminate as many isolated false positives and false negatives as possible by enforcing smoothness across the motion mask. In this comparative study, we evaluate three typical ways of smoothing out the label field:

1. Median filter[34]
2. morphologic filter[34]
3. Markovian prior[35]

In our experiments, we used different window sizes for the median filter and different combination of dilatation and

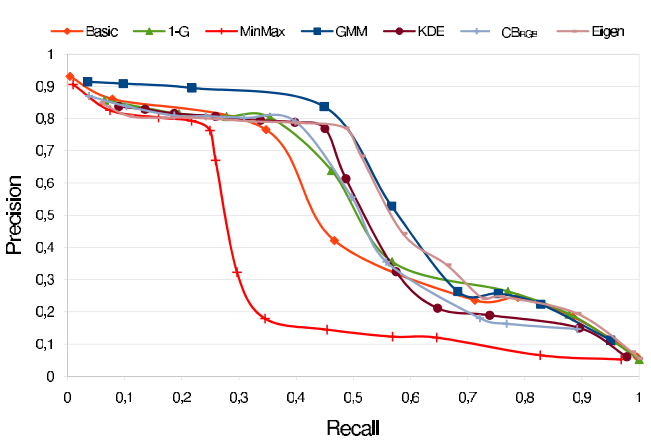

Fig. 7 Precision/recall curve for noisy videos.

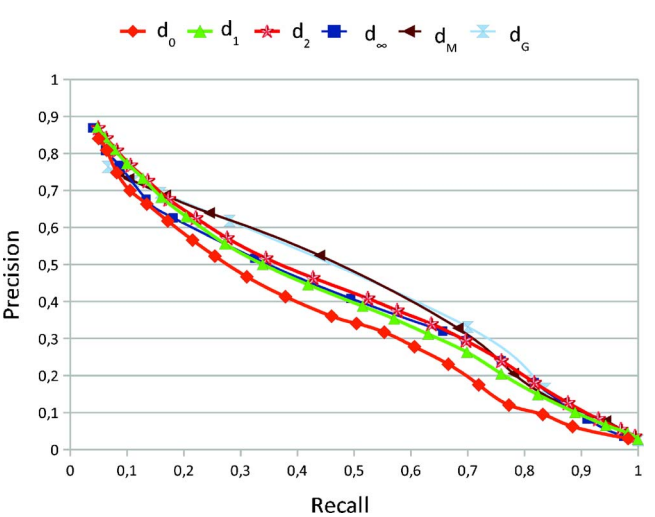

Fig. 8 Precision/recall curves. Influence of the distance metric.

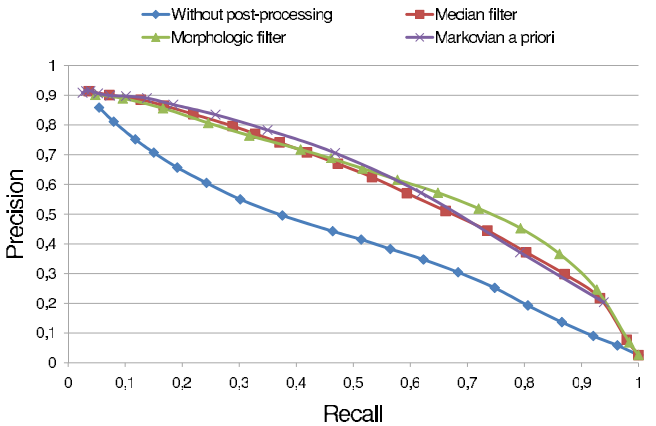

Fig. 9 Precision/recall curves; postprocessing evaluation.

erosion filters. The results presented here were obtained with a 5×5 median filter, while the morphologic operation is defined as follows: close[open$(\chi,W),W]$, where χ is the motion mask, and W is a 5×5 structuring element. As for the Markovian prior, we implemented the Ising potential function:

$$L(\mathcal{X}_\eta,x)=\beta\sum_{s\in\eta}[1-\delta(\mathcal{X}_s,x)], \tag{27}$$

where β is a user-defined constant, η is a second-order neighborhood, $x=\{0,1\}$, and $\delta(.,.)$ is the Kronecker delta. As mentioned by Aach and Kaup,[35] when using a Markovian prior, motion detection can be formulated as a likelihood-ratio test, leading to the following formula:

$$\mathcal{X}_t(s)=\begin{cases}1 & \text{if } d(I_{s,t},B_s)>\tau'\\ 0 & \text{otherwise},\end{cases} \tag{28}$$

where

$$\tau'=\tau-\beta[L(\mathcal{X}_\eta,0)-L(\mathcal{X}_\eta,1)], \tag{29}$$

and τ is a user-defined threshold. Since adding a Markovian prior leads to a maximum *a posteriori* formulation, Eq. (28) can only be solved through an optimization scheme. As in Ref. 35, we chose the ICM (iterated conditional modes) optimizer as it is the fastest and easiest way of solving this equation.

Due to space limitations, we present only the results of spatial aggregation applied to the Basic algorithm. Results obtained with other BS methods lead to similar conclusions. The curves in Fig. 9 were obtained with 12 challenging videos containing animated background and additive noise.

Unsurprisingly, spatial aggregation strongly increases the performance of the algorithm. However, the difference between the three spatial aggregation techniques is not clear as they all seem to produce similar results. This being said, since the Markovian prior uses an iterative optimization scheme (ICM), its computational load is significantly heavier than the other two postprocessing filters. This suggests that, all things considered, the Markovian prior is a weaker solution than the simple post-processing filtering. Motion fields are presented in Fig. 10, illustrating the benefits of postprocessing.

4.4 *Performance on Every Video*

In this section, results obtained with the entire dataset (videos with static background, multimodal backgrounds, or altered videos) are presented. The motion label field of every method has also been filtered out with a morphological filter similar to the one used in the previous section. Results are presented in Fig. 11.

Among these curves, we notice that the global variation between the methods is reduced compared to those in Figs. 5 and 7. This can be explained by a combination of two factors. First, some of the videos used for testing have a large SNR with a rigorously fixed background. Since all BS methods perform well on such videos, the variation between methods can only be smaller on that data set. Second, the postprocessing stage reduces the number of false positives and false negatives that simple methods produce on noisy or multimodal videos. Thus, by combining these two factors, the gap between simple methods (MinMax, Basic, and 1-G) and more sophisticated ones (GMM, KDE, and CB_{RGB}) narrows.

The curves in Fig. 11 also suggest that MinMax, Basic, and Eigen globally underperform while 1-G, GMM, CB_{RGB}, and KDE show more robustness. This observation is very interesting as the simple 1-G method performs al-

(a) (b) (c)

Fig. 10 Benefits of spatial aggregation; (a) input video, (b) motion mask, and (c) motion mask after median filtering.

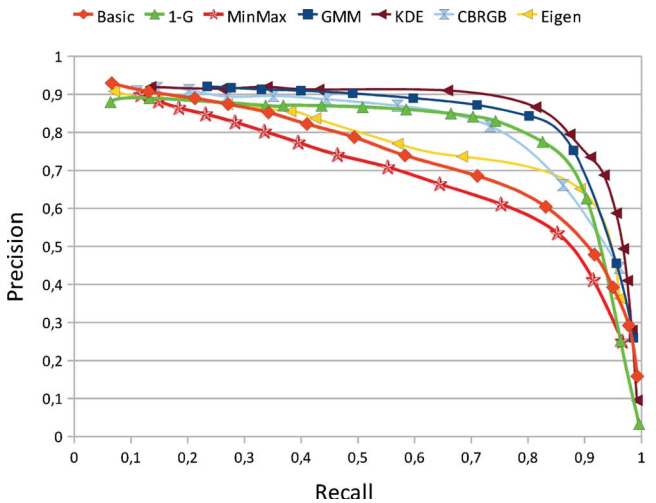

Fig. 11 Precision/recall curves. Evaluation over the whole video data set after postprocessing by a morphological filtering.

most as well as more complex methods such as GMM and KDE. The 1-G method thus seems to be a good compromise between simplicity and robustness.

4.5 *Computation Time and Memory Requirements*

Background subtraction of often the first stage of more global computer vision systems. For real-time applications, the BS algorithm must be light, fast, and efficient so the system can process a streaming video at a rate of 30 frames/s. In that case, computation time and memory requirements are critical information that one must keep in mind before choosing which BS method to implemente. Thus, the performances presented in Figs. 3–11 do not necessarily comply with the immediate requirements of real-time applications, as some methods may be slow and heavy.

In this section, we give both the relative processing time and the minimum memory requirement for each method. To do so, the methods were compiled in release mode with Visual C++. Since the videos do not have the same size (both in space and time), we computed the average relative computational time of each method (see Table 3). We did so by dividing (for each video) the computation time of each algorithm by the computation time of the reference algorithm (Basic). Although processing time strongly depends on how a method is implemented, we believe that some global conclusions can be drawn out of those results.

Table 3 Average relative computation time.

Algorithm	Relative time
Basic	1
1-G	1.32
MinMax	1.47
GMM	4.91
KDE	13.80
Eigen	11.98

Table 4 Memory requirements.

Method	Number of floats per pixel
Basic	3
1-G	6
MinMax	3
GMM	$K\times 5$
KDE	$N\times 3+3$
CB_{RGB}	$L\times 6$
Eigen	$M\times 3+3$

As one can see from Table 3, simple methods such as Basic, 1-G, and MinMax significantly outperform GMM, KDE, and Eigen. Indeed, the latter methods are between 5 and 14 times slower than the Basic one. This clearly suggests that more complicated methods (especially KDE and Eigen) are not *a priori* suited for real-time applications, unless precision is a diving factor.

Also, note that we did not provide processing time for CB_{RGB}. This is because the speed of CB_{RGB} strongly depends on the complexity of the video. For example, CB_{RGB} is as fast as 1-G on videos with static background, but its processing capabilities drop to those of GMM on videos with a multimodal background.

Other key information is the minimum amount of memory used by each algorithm. Considering that data are stored in floating point values, Table 4 presents the minimum number of floats per pixel each method requires to model the background. In Table 4, L, K, M, and N are, respectively, the number of code words (between 1 and 4), the number of Gaussians in the mixture (between 3 and 5), the number of eigenvectors kept (typically 20), and the number of frames in the buffer (between 100 and 200). Considering these numbers, KDE and Eigen are clearly penalized and could hardly be implemented on an embedded system such as an IP (Internet Protocol) camera DSP (digital signal processor) chip, whose local memory is very limited.

5 Synthesis of Results and Conclusion

We presented a comparative study of seven highly implemented BS methods. Some of these methods are simple (Basic, 1-G, MinMax), while others are significantly more sophisticated (CB_{RGB}, GMM, KDE, Eigen). These methods are compared based on their CPU/memory requirements as well as their capability of correctly detecting motion on all kinds of videos (e.g., indoor environments, moving backgrounds, etc.). Since the videos in our database come with ground truth, we used precision/recall curves to compare the relative accuracy of the algorithms. Three commonly implemented postprocessing methods were also tested.

Table 5 Summary of presented results, where the number of stars is proportional to the efficiency of the method.

	Basic	1-G	MinMax	GMM	KDE	CB_{RGB}	Eigen
Static background	***	***	**	***	***	***	***
Multimodel background	*	**	*	***	***	***	*
Noisy background	*	***	*	***	***	***	***
Computation time	***	***	***	**	*	—	*
Memory requirement	***	***	***	**	*	**	*

From the results reported in Sec. 4, an overall summary has been put together and presented in Table 5. This table highlights the five main conclusions of this study.

1. As some authors already mentioned,[13] methods working on gray-scale videos such as MinMax and d_0 are clearly less precise than others working with colors.
2. Sophisticated methods do not always produce more precise results. This is especially true when dealing with videos having a large SNR and little background motion. Methods such as CB_{RGB}, GMM, and KDE perform better only when the background is unstable or when the level of noise gets significantly large.
3. Methods such as GMM, KDE, and Eigen are not *a priori* well suited for real-time applications as far as their CPU and/or memory requirement is concerned.
4. There is very little difference between the three spatial aggregation techniques we implemented. However, due to its iterative nature, the Markovian prior is slower than the other two filters and thus appears as a weaker solution.
5. The use of spatial aggregation narrows the difference between every technique.

Out of these results, one may wonder if there is a win-win situation, i.e., whether there is a method that has light memory and low CPU usage while being robust to instabilities. The answer to this question is obviously no, as some methods perform well on some aspect and poorly on others. This being said, the 1-G method seems to offer the best compromise between speed, simplicity, and efficiency. This is especially true when a postprocessing filter is implemented.

References

1. V. Cheng and N. Kehtarnavaz, "A smart camera application: Dsp-based people detection and tracking," *J. Electron. Imaging* **9**(3), 336–346 (2000).
2. T. Inaguma, H. Saji, and H. Nakatani, "Hand motion tracking based on a constraint of three-dimensional continuity," *J. Electron. Imaging* **14**(1), 013021 (2005).
3. D. Makris and T. Ellis, "Path detection in video surveillance," *Image Vis. Comput.* **20**, 895–903 (2002).
4. D. Makris and T. Ellis, "Learning semantic scene models from observing activity in visual surveillance," *IEEE Trans. Syst., Man, Cybern., Part B: Cybern.* **35**, 397–408 (2005).
5. Q. Zhou and J. Aggarwal, "Tracking and classifying moving objects from video," in *Proc. Performance Evaluation of Tracking Systems Workshop*, IEEE, Piscataway, NJ (2001).
6. S. C. S. Cheung and C. Kamath, "Robust techniques for background subtraction in urban traffic video," in *Visual Communications and Image Processing 2004*, *Proc. SPIE* **5308**, 881–892 (2004).
7. W. Hu, T. Tan, L. Wang, and S. Maybank, "A survey on visual surveillance of object motion and behaviors," *IEEE Trans. Syst. Man Cybern., Part C Appl. Rev.* **34**(3), 334–352 (2004).
8. K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: principles and practice of background maintenance," in *Proc. Int. Conf. on Computer Vision*, Vol. 1, pp. 255–261, IEEE, Piscataway, NJ (1999).
9. S. Panahi, S. Sheikhi, S. Hadadan, and N. Gheissari, "Evaluation of background subtraction methods," in *Proc. Int. Conf. on Digital Image Computing: Techniques and Applications*, pp. 357–364, IEEE, Piscataway, NJ (2008).
10. S. Herrero and J. Bescós, "Background subtraction techniques: systematic evaluation and comparative analysis," in *Proc. Int. Conf. Advanced Concepts for Intelligent Vision Systems*, *Lect. Notes Comput. Sci.* **5807**, 33–42 (2009).
11. T. H. Chalidabhongse, K. Kim, D. Harwood, and L. Davis, "A perturbation method for evaluating background subtraction algorithms," in *Proc. Int. Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, IEEE, Piscataway, NJ (2003).
12. Y. Benezeth, B. Emile, and C. Rosenberger, "Comparative study on foreground detection algorithms for human detection," in *Proc. Int. Conf. on Image and Graphics*, pp. 661–666, IEEE, Piscataway, NJ (2007).
13. D. Hall, J. Nascimento, P. Ribeiro, E. Andrade, P. Moreno, S. Pesnel, T. List, R. Emonet, R. B. Fisher, J. S. Victor, and J. L. Crowley, "Comparison of target detection algorithms using adaptive background models," in *Proc. Int. Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pp. 113–120, IEEE, Piscataway, NJ (2005).
14. R. Cucchiara, C. Grana, A. Prati, and R. Vezzani, "Probabilistic posture classification for human-behavior analysis," *IEEE Trans. Syst. Man Cybern., Part A. Syst. Humans* **35**, 42–54 (2005).
15. J. Heikkila and O. Silven, "A real-time system for monitoring of cyclists and pedestrians," in *Proc. Workshop on Visual Surveillance*, pp. 74–81, IEEE, Piscataway, NJ (2004).
16. S. Ghidary, Y. Nakata, T. Takamori, and M. Hattori, "Human detection and localization at indoor environment by homerobot," in *Proc. Int. Conf. on Systems, Man, and Cybernetics*, Vol. 2, pp. 1360–1365, IEEE, Piscataway, NJ (2000).
17. C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: real-time tracking of the human body," *IEEE Trans. Pattern Anal. Mach. Intell.* **19**, 780–785 (1997).
18. S. C. S. Cheung and C. Kamath, "Robust background subtraction with foreground validation for urban traffic video," *EURASIP J. Appl. Signal Process.* **2005**, 2330–2340 (2005).
19. R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting moving objects, ghosts and shadows in video streams," *IEEE Trans. Pattern Anal. Mach. Intell.* **25**(10), 1337–1342 (2003).
20. I. Haritaoglu, D. Harwood, and L. S. Davis, "W 4: real-time surveillance of people and their activities," *IEEE Trans. Pattern Anal. Mach. Intell.* **22**, 809–830 (2000).
21. C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Proc. Int. Conf. on Computer Vision and Pattern Recognition*, Vol. 2, IEEE, Piscataway, NJ (1999).
22. P. KaewTraKulPong and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection," in *Proc. Workshop on Advanced Video-Based Surveillance Systems Conf.*, Kluwer Academic Publishers, Dordrecht, The Netherlands (2001).

23. Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction," in *Proc. Int. Conf. Pattern Recognition*, pp. 28–31, IEEE, Piscataway, NJ (2004).
24. A. Elgammal, D. Harwood, and L. Davis, "Non-parametric model for background subtraction," in *Proc. Eur. Conf. on Computer Vision*, *Lect. Notes Comput. Sci.* **1843**, 751–767 (2000).
25. A. Mittal and N. Paragios, "Motion-based background subtraction using adaptive kernel density estimation," in *Proc. Int. Conf. on Computer Vision and Pattern Recognition*, pp. 302–309, IEEE, Piscataway, NJ (2004).
26. K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis, "Real-time foreground-background segmentation using codebook model," *Real-Time Imag.* **11**, 172–185 (2005).
27. N. M. Oliver, B. Rosario, and A. P. Pentland, "A Bayesian computer vision system for modeling human interactions," *IEEE Trans. Pattern Anal. Mach. Intell.* **22**, 831–843 (2000).
28. R. Li, Y. Chen, and X. Zhang, "Fast robust eigen-background updating for foreground detection," in *Proc. Int. Conf. on Image Processing*, pp. 1833–1836, IEEE, Piscataway, NJ (2006).
29. J. Rymel, J. Renno, D. Greenhill, J. Orwell, and G. A. Jones, "Adaptive eigen-backgrounds for object detection," in *Proc. Int. Conf. on Image Processing*, pp. 1847–1850, IEEE, Piscataway, NJ (2004).
30. http://dparks.wikidot.com.
31. J. H. Piater and J. L. Crowley, "Multi-modal tracking of interacting targets using Gaussian approximations," in *Proc. 2nd Int. Workshop on Performance Evaluation of Tracking and Surveillance (PETS'2001)*, pp. 141–177, IEEE, Piscataway, NJ (2001).
32. L. Brown, A. Senior, Y. Tian, J. Vonnel, A. Hampapur, C. Shu, H. Merkl, and M. Lu, "Performance evaluation of surveillance systems under varying conditions," in *Proc. Performance Evaluation of Tracking Systems Workshop*, pp. 1–8, IEEE, Piscataway, NJ (2005).
33. http://imagelab.ing.unimore.it/vssn06.
34. A. C. Bovik, *Handbook of Image and Video Processing*, Academic Press, Orlando, FL (2005).
35. T. Aach and A. Kaup, "Bayesian algorithms for adaptive change detection in image sequences using markov random fields," *Signal Process. Image Commun.* **7**, 147–160 (1995).

Yannick Benezeth received his PhD degree in computer science from the University of Orléans, France in 2009. He also graduated as an engineer from the ENSI de Bourges and received his MS degree from the University of Versailles-Saint-Quentin-en-Yvelines, France, in 2006. Since November 2009, he has been a research engineer with the Orange Labs-France Telecom research center in Rennes, France. His research interests include computer vision, object detection, activity recognition, and video analysis techniques for TV stream structuring.

Pierre-Marc Jodoin graduated as an engineer in computer science from the Ecole Polytechnique of Montreal, Canada, in 2000. He then received his MSc and PhD degrees with honor in computer science from the University of Montreal, Canada, in 2003 and 2007. He is currently an associate professor in the Computer Science Department of the University of Sherbrooke. He is also a member of MOIVRE (Modlisation en Imagerie, Vision et Researux de neurones), a research group on computer vision and image analysis. His current research interests include statistical methods for video analysis and surveillance, segmentation methods applied to medical imaging, 3-D reconstruction, and image quality assessment.

Bruno Emile is an assistant professor at IUT of Châteauroux, France. He received his PhD degree from the University of Nice in 1996. He belongs to the PRISME Institut of the University of Orléans in the Image and Signal for System research unit. His research interests concern object detection.

Hélène Laurent is an assistant professor at ENSI of Bourges, France. She received her PhD degree from the University of Nantes in 1998. She belongs to the PRISME Institut of the University of Orléans in the ISS (Images and Signals for Systems) research unit. Her research interests concern evaluation of image processing algorithms.

Christophe Rosenberger is a full professor at ENSICAEN, France. He received PhD degree from the University of Rennes I in 1999. He has been with the GREYC Laboratory in the Computer Security Research Unit since 2007. His research interests include evaluation of image processing and biometrics.