Comparative Study of Background Subtraction Algorithms

Y. Benezeth[1] P.-M. Jodoin[2] B. Emile[3] H. Laurent[3] C. Rosenberger[4]

[1] *Orange Labs France Telecom R&D*
4, rue du Clos Courtel
Cesson Sévigné Cedex - 35512 , France

[2] *MOIVRE, Université de Sherbrooke,*
2500 bd. de l'Université,
Sherbrooke, J1K 2R1, Canada

[3] *Institut PRISME, Université d'Orléans,*
88 bd. Lahitolle, 18020 Bourges, France

[4] *GREYC, ENSICAEN,*
Université de Caen - CNRS,
6 bd. Maréchal Juin, 14000 Caen, France

In this paper, 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 of this study 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 requirement, and the computational effort they require. The impact of a Markovian prior as well as some post-processing operators are also evaluated. Most of the videos used in this study

come from state-of-the-art benchmark databases and represent different challenges such as poor signal-to-noise ratio, multimodal background motion and camera jitter. Overall, this study not only helps better understand to which type of videos each method suits best but also estimate how better sophisticated methods are, compared to basic background subtraction methods.

I. 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 multi-stage computer vision system [8, 20, 24, 25] (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 [34]. This solution has proven successful whenever the camera is rigorously static with a fixed noise-free background (see [9] for some examples).

But detecting motion through background subtraction is not always as easy as it may first appear. Indeed, some videos with poor signal-to-noise ratio 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 global threshold reveals itself as a weak solution. In order to cope with those challenges, numerous background models and distance measures bound up to different optimization schemes have been proposed in the past decade. Those methods are (at least in theory) more robust to noise and background instability than the basic background subtraction approaches. But 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 light weight architecture?

In this paper, we compare some of the most implemented background subtraction methods on various real, synthetic and semi-synthetic video sequences representing different chal-

lenges. The goal of this study is threefold:

1. evaluate how better sophisticated methods are compared to simple background subtraction methods;

2. compare the processing power and the amount of memory required by each method at runtime;

3. determine to which type of video each method suits best.

As background subtraction is widely used in computer vision, numerous surveys and comparative studies have been published over the years. While some of those papers contain descriptive evaluations of motion detection methods [19], others provide quantitative evaluations based on pre-annotated video sequences. It is the case with Toyama *et al.* [32] and Panahi *et al.* [28] which 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 (FN) and false positives (FP) they produced in each video sequence. Although FN and FP are typical quality measures, they are nonetheless strongly dependent: when FN decreases, FP always increases in return and vice-versa. Thus, a single couple $\{FN, FP\}$ is not sufficient to compare BS methods together as a method with large FN and low FP is not necessarily better or worse than one with low FN and large FP. 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 [18] 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.* [7] 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 background is mesured as a function of contrast. The main advantage of this method is its ability of using all kinds of videos for quality assessment, even those without

ground truth. Unfortunately though, this method is not void 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 post-processing tools. Moreover, a few methods are compared in their article. We extended the comparison to seven in this paper.

Other surveys evaluate BS methods in the context of target detection such as car and pedestrian localization [4, 15]. 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 of dealing 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 post-processing stage. This is a typical cause for large FP and FN rates. Moreover, the post-processing stage in [15] 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 than the ones published so far. The keypoints of our study are the following:

1. the inter-dependence between FP and FN is considered so the evaluation is fair for every method,

2. the video dataset is splited into groups of videos containing similar features and representing similar level 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. In section II, seven commonly-implemented motion detection methods are described in details. The protocol used for the comparison, including the video dataset, is introduced in section III while results and conclusion are presented in sections IV and V.

II. 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:

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

where τ is a threshold, $\mathcal{X}_t$ is the motion label field at time t (also called motion mask), d is the distance between $\boldsymbol{I}_{s,t}$, the color at time t and pixel s, and $\boldsymbol{B}_s$, 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.

A. Basic Motion Detection *(Basic)*

The easiest way to model the background B is through a single grayscale/color image void of moving objects. This image can be a picture taken in absence of motion or estimated via a temporal median filter [12, 17, 34]. In order to cope with illumination changes and background modifications, it can be iteratively updated as follows:

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

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 grayscale images.

Note that it is also possible to use the previous frame I_{t-1} as background image B [14]. With this configuration though, motion detection becomes an *inter-frame 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.

B. One Gaussian *(1-G)*

Modeling B with a single image as in Section II A. 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, in order to account for noise, Wren *et al.* [33] 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:

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

or a Mahalanobis distance:

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

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 $|\boldsymbol{I}_{s,t} - \boldsymbol{\mu}_{s,t}|$ has to 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:

$$\boldsymbol{\mu}_{s,t+1} = (1-\alpha).\boldsymbol{\mu}_{s,t} + \alpha.\boldsymbol{I}_{s,t} \quad (9)$$

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

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 [10], and others being robust to shadows [11].

C. Minimum, Maximum and Maximum Inter-Frame Difference *(MinMax)*

Another method whose goal is to locally adapt to noise is the W^4 system by Haritaoglu *et. al.* [16]. 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:

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

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 needs a larger variation to be labeled in motion than a pixel in a stable area. In this case though, each background pixel is associated to three extremum values instead of a mean vector and a covariance matrix. The original algorithm only operates on grayscale 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.

D. 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 [31], 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 :

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

where $\mathcal{N}(\boldsymbol{\mu}_{i,s,t}, \boldsymbol{\Sigma}_{i,s,t})$ is the i^{th} Gaussian model and $\omega_{i,s,t}$ 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, $\boldsymbol{\Sigma} = \sigma^2\text{Id}$. In their method, parameters of the matched component (*i.e.* the nearest Gaussian for which $\boldsymbol{I}_{s,t}$ is within 2.5 standard deviations of its mean) are updated as follows :

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

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

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

where α is an 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 equation 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 $\boldsymbol{I}_{s,t}$, the one with the lowest weight is replaced by a Gaussian with mean $\boldsymbol{I}_{s,t}$, 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 :

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

where τ is a threshold. Then, those pixels whose color $\boldsymbol{I}_{s,t}$ 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 [21] and [35], new updating algorithms used to learn mixture models are presented.

E. Kernel Density Estimation *(KDE)*

An unstructured approach can also be used to model a multimodal PDF. In this perspective, Elgammal *et al.* [13] 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 one-dimensional kernels can be used:

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

A pixel is labeled as foreground if it is unlikely to come from this distribution, *i.e.* when $P(\boldsymbol{I}_{s,t})$ is smaller than a predefined threshold. Note that σ_j can be fixed or pre-estimated following Elgammal *et al.*'s method [13]. More sophisticated methods can also be envisaged such as Mittal and Paragios's [26] which is based on "Variable Bandwidth Kernels".

F. Codebook *(CB_{RGB})*

Another approach whose goal is to cope with multimodal backgrounds is the so-called *codebook* method by Kim *et al.* [22]. Based on a training sequence, the method assigns to each background pixel a series of key color values (called codewords) stored in a codebook. Those codewords 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 codeword 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 has been designed to eliminate false positives

caused by illumination changes. This is done by performing a separate evaluation of color distortion:

$$\sqrt{I_{s,t}^{R^2} + I_{s,t}^{G^2} + I_{s,t}^{B^2} - \frac{(\mu_{i,s}^{R}.I_{s,t}^{R} + \mu_{i,s}^{G}.I_{s,t}^{G} + \mu_{i,s}^{B}.I_{s,t}^{B})^2}{\mu_{i,s}^{R^2} + \mu_{i,s}^{G^2} + \mu_{i,s}^{B^2}}} < \tau \tag{19}$$

and brightness distortion:

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

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} codeword of pixel s and τ is a threshold. Whenever a pixel s satisfies equations 19 and 20, it indicates that the pixel matches the i^{th} codeword and thus is labeled "static".

However, we empirically observed that such chroma shift assumption is far too restrictive for some urban scenes and produces a large number of false negatives. 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 codeword is a RGB Gaussian distribution. Based on a N-frame long training sequence and $C_s = \{c_{1,s}, \ldots, c_{L,s}\}$, a codebook associated to pixel s made of L codewords, each codeword $c_{i,s}$ 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.* $\boldsymbol{\mu}_{1,s} = \boldsymbol{I}_{s,0}$ and $\boldsymbol{\Sigma}_{1,s} = \sigma_0^2.\text{Id}$, where σ_0^2 is a constant and Id is the identity matrix. Then, each new color $\boldsymbol{I}_{s,t}$ is compared with the pre-estimated codewords $c_{i,s}$ (cf. equation 8) and, for each match, the associated codebook's parameters are updated following equations 9 and 10. Whenever $\boldsymbol{I}_{s,t}$ has no match in the codebook, a new codeword $c_{j,s}$ is created and initialized as follows: $\boldsymbol{\mu}_{j,s} = \boldsymbol{I}_{s,t}$ and $\boldsymbol{\Sigma}_{j,s} = \sigma_0^2.\text{Id}$. During the detection phase, each pixel is classified based on its codeword as follows:

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

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

G. Eigen Backgrounds *(Eigen)*

A non pixel-level method has been proposed by Oliver *et al.* [27] which 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.

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 $\mu = \frac{1}{N}\sum_{i=1}^{N} I_i$ 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 $\mathbf{\Sigma} = E[\boldsymbol{X}\boldsymbol{X}^T]$, with $\boldsymbol{X} = [X_1, \ldots, X_n]$. According to the Karhunen-Loeve Transform, we can compute the eigenvector matrix $\boldsymbol{\phi}$ which diagonalizes the covariance matrix $\mathbf{\Sigma}$:

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

where D is the corresponding diagonal matrix. Following a Principal Component Analysis (PCA), a new rectangular matrix $\boldsymbol{\phi}_M$ is made out of the M eigenvectors with the largest eigenvalues. Once the eigenbackground $\boldsymbol{\phi}_M$ and the mean μ have been computed, the column representation of each input image I_t is first projected onto the $M-$dimensional subspace :

$$\boldsymbol{B}_t = \boldsymbol{\phi}_M(I_t - \mu) \tag{23}$$

and then reconstructed as follows :

$$I'_t = \boldsymbol{\phi}_M^T\boldsymbol{B}_t + \mu. \tag{24}$$

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

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

where τ is a threshold and d_2 is the Euclidian distance. Note that the Eigen decomposition (equation 22) can be quickly computed with a Single Value Decomposition but, unfortu-

nately, ϕ_M is hard to keep up-to-date as the video streams in. Solutions based on incremental PCA (*e.g.* [23, 30]) have been proposed to cope with this drawback.

III. EXPERIMENTAL PROTOCOL

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

A. Experimental framework

The motion detection methods introduced in the previous section are evaluated following the framework in shown Figure 12. 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 allowing the evaluation of true positives (TP), false positives (FP) and false negatives (FN) numbers. Those values are combined into a (Precision/Recall) couple defined as:

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Recall} = \frac{TP}{TP + FN}. \tag{26}$$

By definition, a good algorithm is one producing simultaneously a small number of false positives and false negatives, *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 [1, 13]. The various settings used for each method are presented in Table II.

B. Video Dataset

In order to gauge performances, BS methods have been tested on a wide range of real, synthetic and semi-synthetic video sequences (one example corresponding to each camera viewpoint is presented in Figure 13). Our dataset is composed of 29 video sequences (15 real, 10 semi-synthetic and 4 synthetic) containing between 100 and 3000 frames of size 320×240. We created some synthetic and semi-synthetic videos, others were downloaded from the PETS2001 dataset [29], the IBM dataset [6] and the VSSN 2006 competition

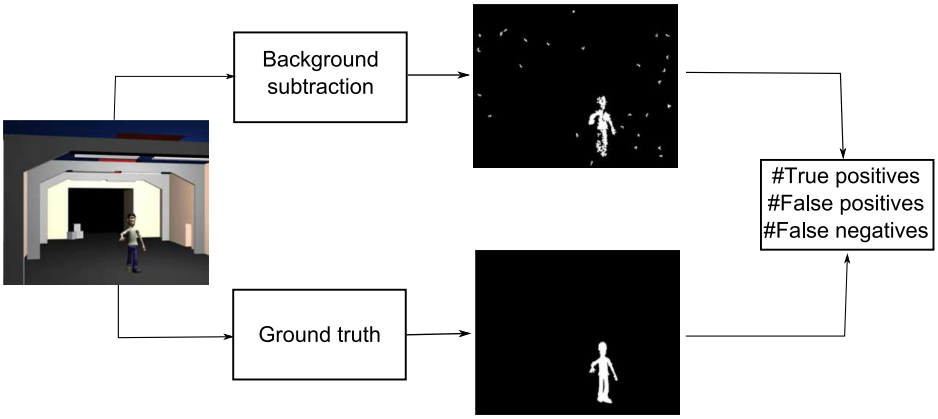

FIG. 1: Overview of the experimental framework.

TABLE I: Background subtraction methods evaluated in this comparative study.

Methods	**Description**
Basic	The model is a color image void of moving objects, described in II A.
1-G	The model of each pixel is a gaussian distribution, described in II B.
MinMax	The model is composed of a Min, a Max and a Max of interframes difference, described in II C.
GMM	The model of each pixel is a mixture of gaussians, described in II D.
KDE	The model of each pixel is a non-parametric distribution, described in II E.
CB_{RGB}	The model is composed of a set of gaussian distributions, described in II F.
Eigen	The model is non-pixel based using an eigenspace, described in II G.

[2]. The semi-synthetic videos are made of synthetic foreground objects (people and cars) moving over a real background. The whole video dataset represents both indoor (20 videos) and outdoor scenes (9 videos). Moreover, 6 videos contain animated background textures caused by trees and bushes shaken by the wind. While ground truths are easily obtained for synthetic and semi-synthetic video sequences, they are only available on some reference images (manually annotated or provided with the dataset) 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.

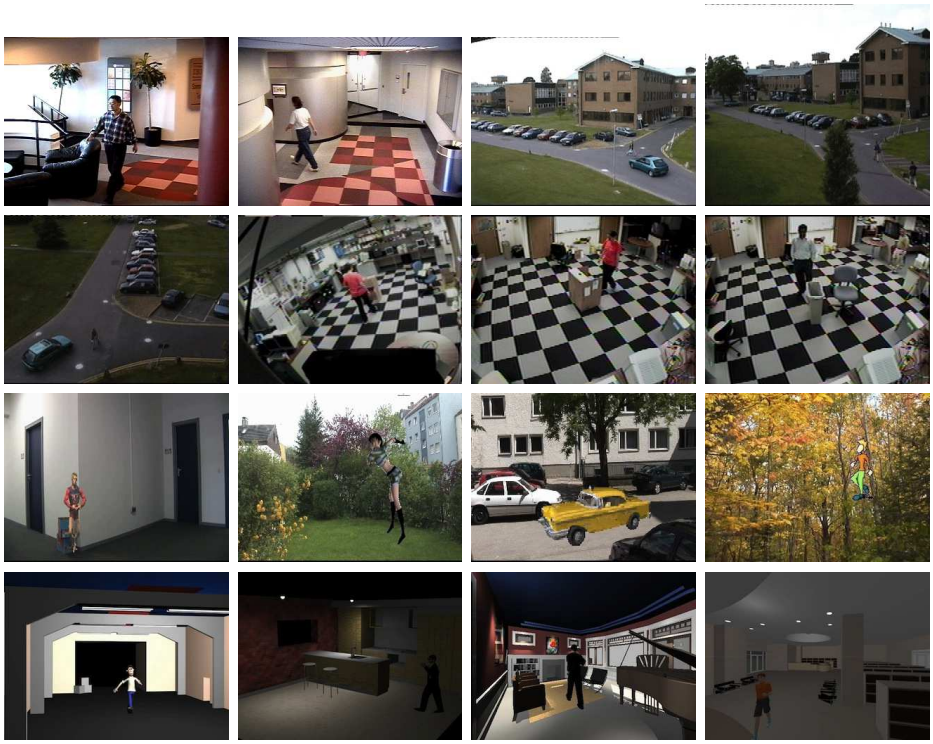

FIG. 2: Snapshots of each camera viewpoint of the video dataset.

TABLE II: 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	$N = 100$, $M = 20$

IV. EXPERIMENTAL RESULTS

We tested the BS algorithms described in section II on groups of videos illustrating different scenarios and thus different challenges. This section presents benchmarks obtained

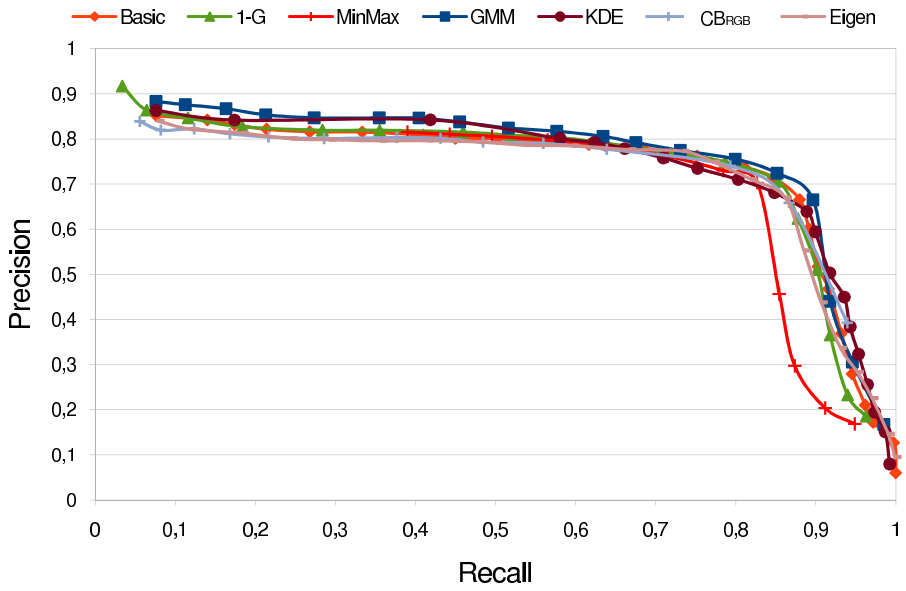

FIG. 3: Test 1: Precision/Recall curves for noise-free videos with static backgrounds.

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 technique at runtime. The effect of spatial aggregation such as post-processing filters and a Markovian prior is also presented.

A. Evaluation of Background Models

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 signal-to-noise ratio with rigorously static background. Here, a total of 15 videos have been used for testing. Results are presented in Figure 14.

As can be seen from those Precision / Recall curves, the *MinMax* method is slightly less effective than the others, mostly because it exclusively works on grayscale data, thus ignoring color. The other methods globally produce the same results, more of less 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

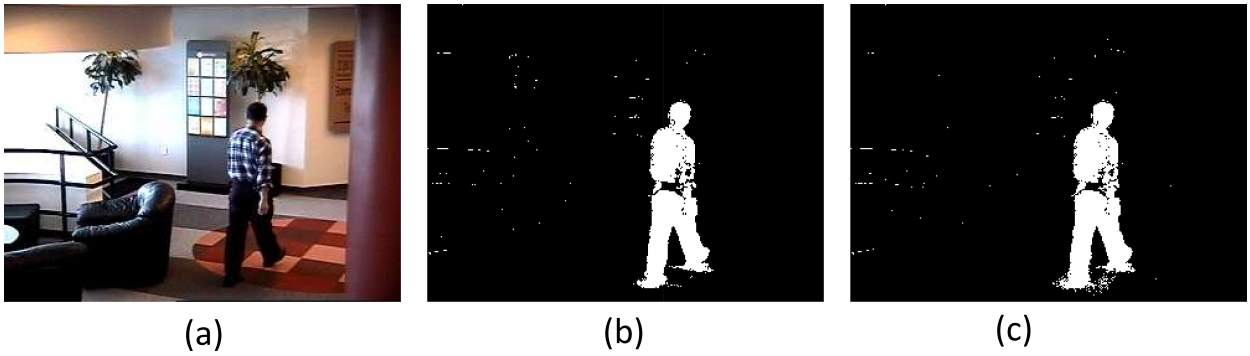

FIG. 4: (a) Input video with static background and large signal-to-noise ratio (b) motion mask with *Basic* (c) motion mask with *GMM*.

good conditions. This is clearly illustrated in Figure 15 in which results obtained with *Basic* and *GMM* show barely no differences. In fact, every method, be it simple or not, fails at detecting regions of the moving objects whose color is similar to the background (look at the hip of the pedestrian in Figure 15). This is a typical *camouflage* effect for which no BS method is capable of coping with.

2. Videos with a Multimodal Background

This test aims at evaluating the robustness of each method on videos exhibiting an animated background. Here, 6 video sequences with strong background motion caused by trees and shrubs shaken by the wind are used. Results are presented in Figure 16.

In this case, 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 non-adaptative threshold does not suit animated backgrounds. Note that the grayscale 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. In Figure 17, 7 motion masks are presented so the reader can visualize the differences between the BS

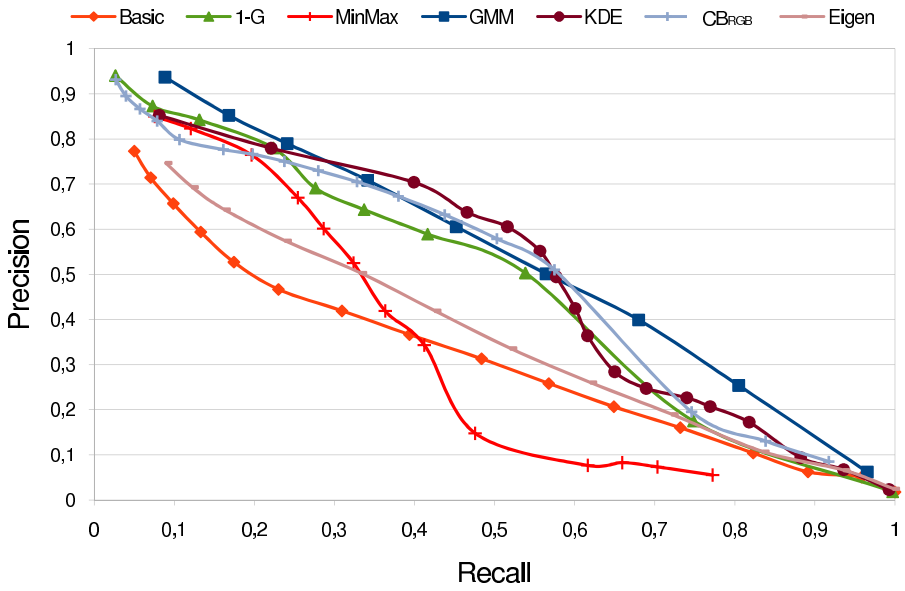

FIG. 5: Test 2: Precision/Recall curves for videos with multimodal backgrounds.

methods. The difference between *Basic*, *MinMax* and the other methods is clear.

3. Noisy Videos

The third test aims at evaluating the influence of noise. Here, 15 videos corrupted with additive Gaussian noise are used for testing. This type of distortion often occurs when working with low quality cameras or when filming dark areas. Results are presented in Figure 18.

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 false positives. 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*.

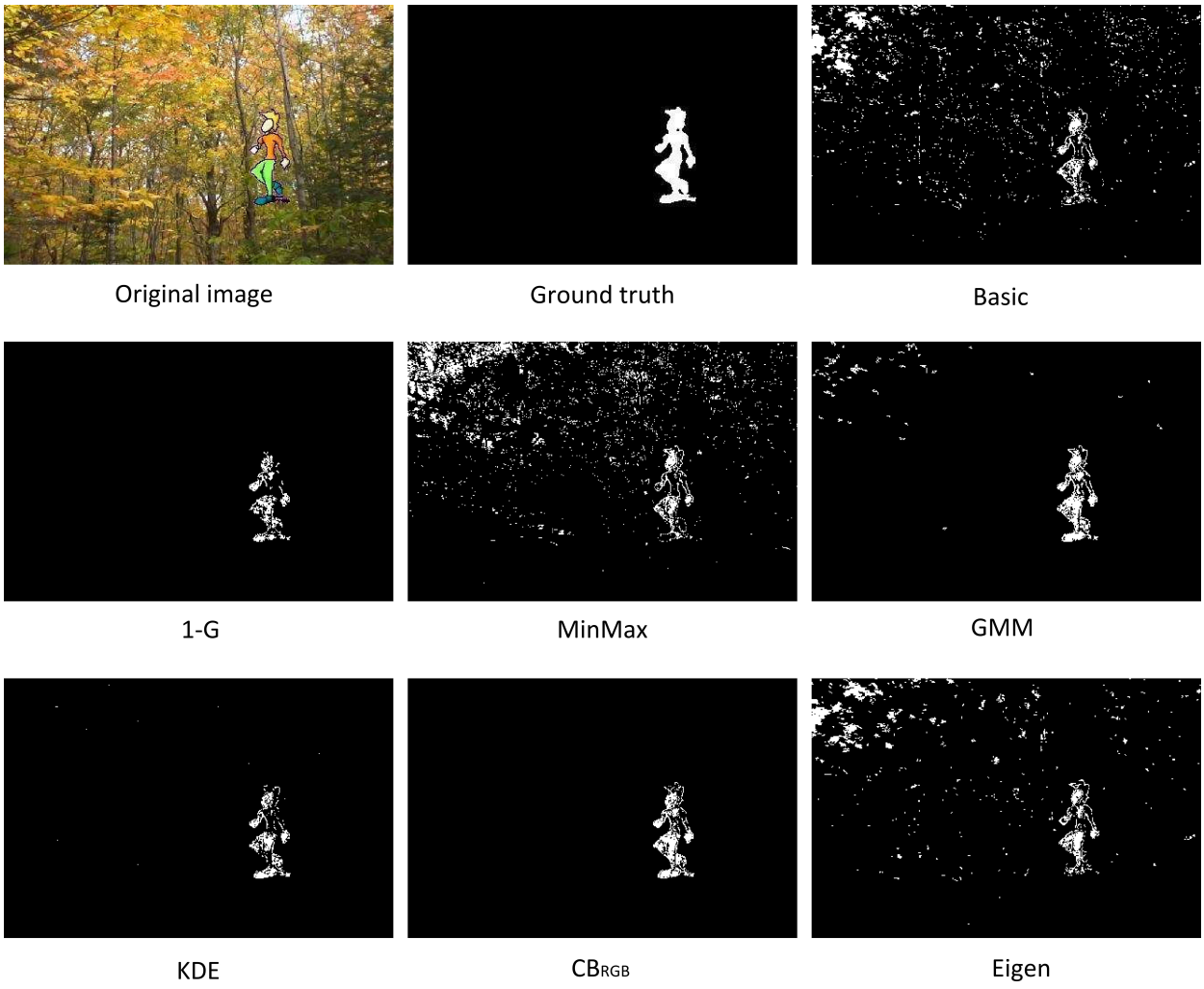

FIG. 6: Motion masks obtained with a video containing a multimodal background.

B. Evaluation of the Distance Metrics

As mentioned in section II A and II B, 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 signal-to-noise ratio. Results are presented in Figure 19.

Out of those results, it comes 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 only works on grayscale 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.

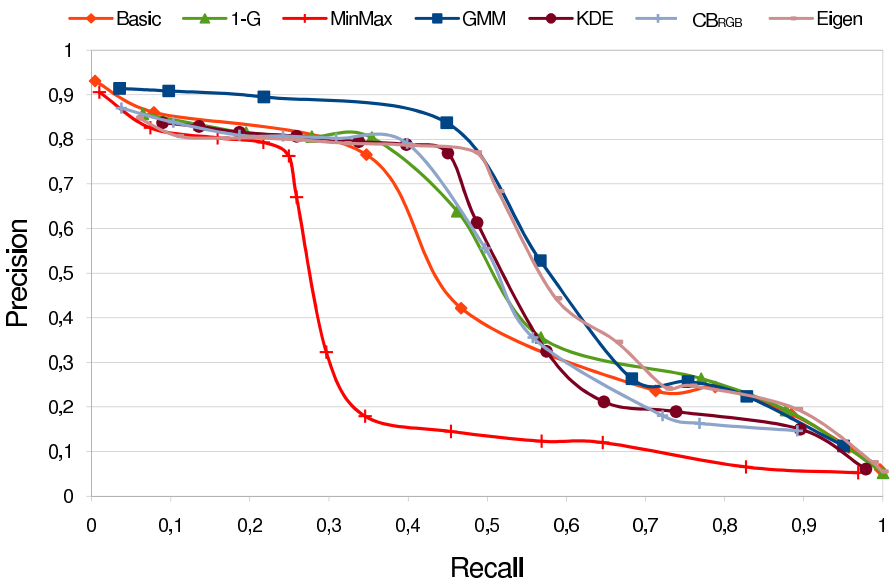

FIG. 7: Precision/Recall curve for noisy videos.

C. 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 :

- median filter [5]
- morphologic filter [5]
- Markovian prior. [3]

In our experiments, we used different window sizes for the median filter and different combination of dilatation and erosion filters. The results here presented were obtained with a 5×5 median filter, while the morphologic operation is defined as follows : $close(open(\mathcal{X}, W), W)$ where $\mathcal{X}$ is the motion mask and W is a 5×5 structuring element. As for the Markovian prior, we implemented the Ising potential function:

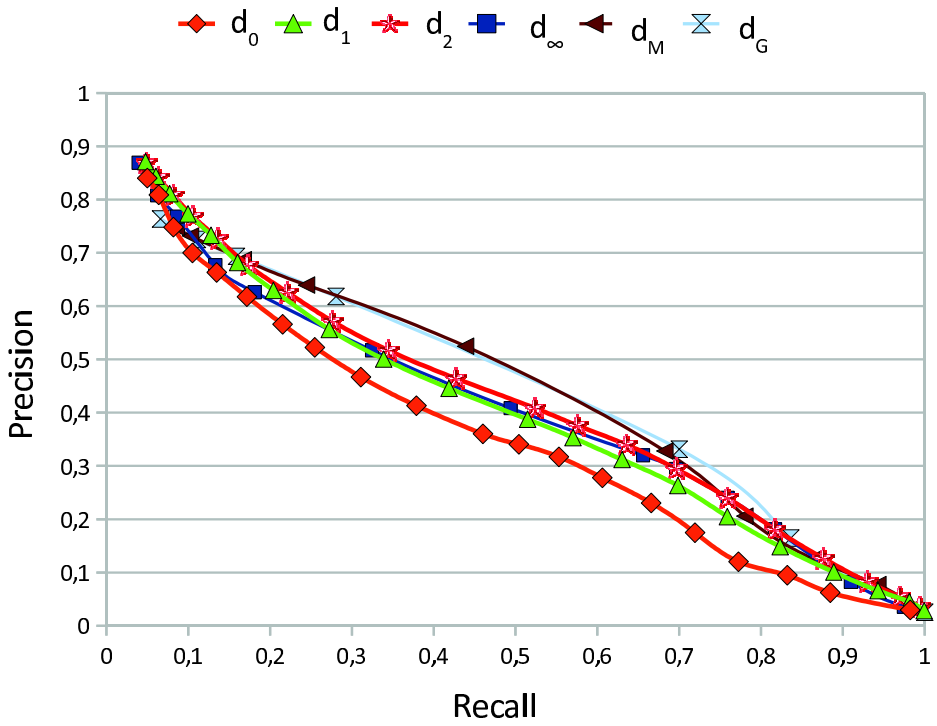

FIG. 8: Precision/Recall curves. Influence of the distance metric.

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

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 [3], when using a Markovian prior, motion detection can be formulated as a likelihood-ratio test leading to the following formula:

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

where

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

τ being a user-defined threshold. Since adding a Markovian prior leads to a Maximum a Posteriori formulation, equation 28 can only be solved through an optimization scheme. As in [3], we chose the ICM optimizer as it is the fastest and easiest way of solving this equation.

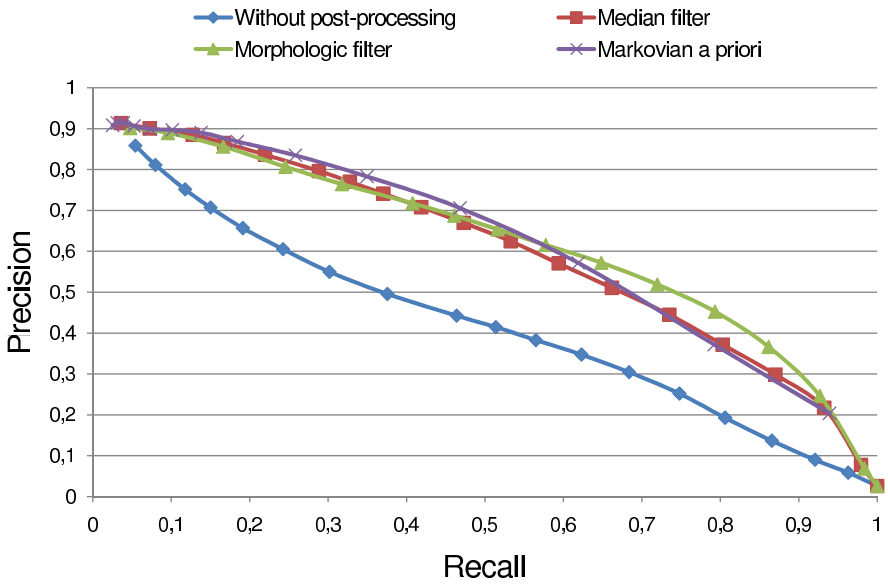

FIG. 9: Precision/Recall curves. Post-processing evaluation.

Due to space limitation, we only present the results of spatial aggregation applied to the *Basic* algorithm. Results obtained with other BS methods lead to similar conclusions. The curves in Figure 20 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 post-processing 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 Figure 21 illustrating the benefits of post-processing.

D. 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 Figure 22.

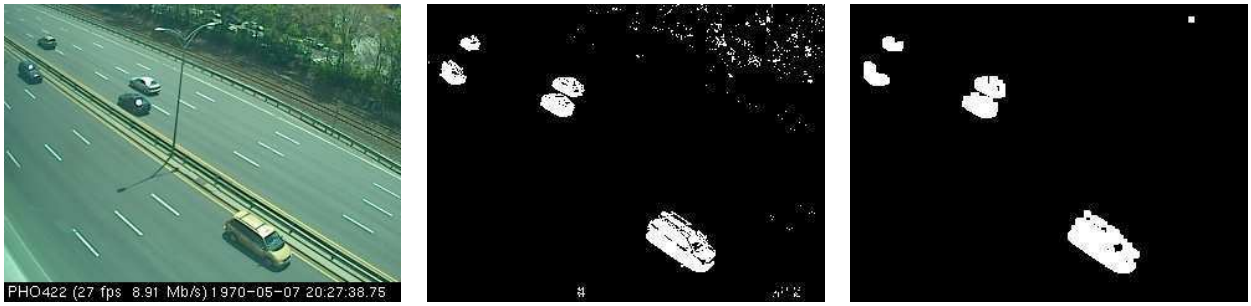

FIG. 10: Benefits of spatial aggregation (a) Input video (b) Motion mask (c) Motion mask after median filtering.

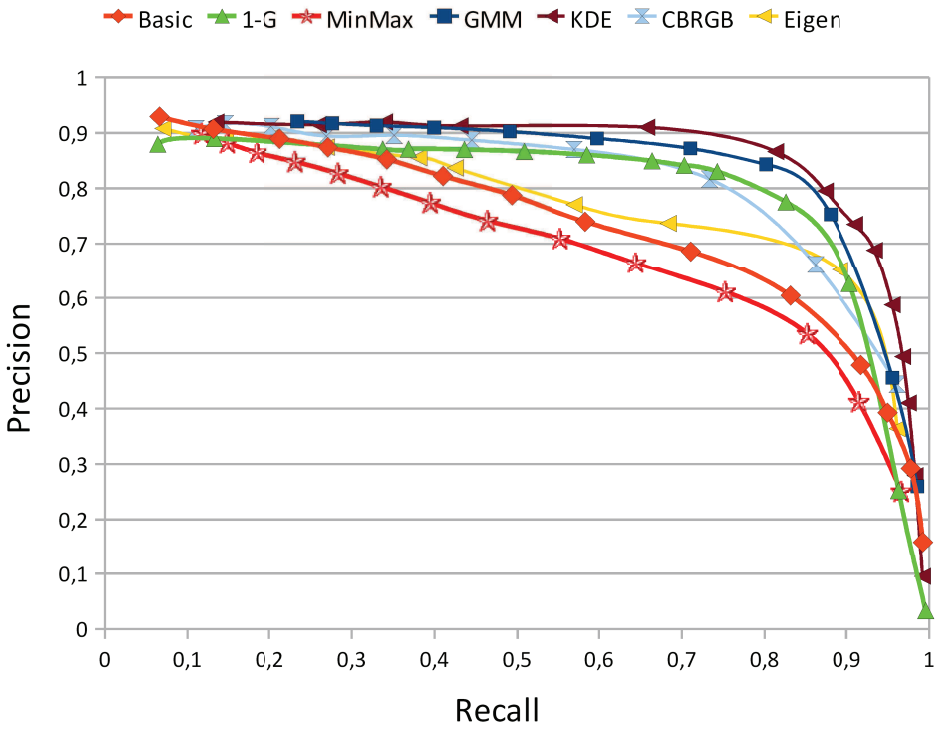

FIG. 11: Precision/Recall curves. Evaluation over the whole video dataset after post-processing by a morphological filtering.

Out of these curves, we notice that the global variation between the methods is reduced compared to those in Figure 16 and 18. This can be explained by a combination of two factors. First, some of the videos used for testing have a large signal-to-noise ratio with a rigorously fixed background. Since all BS methods perform well on such videos, the variation between methods can only be smaller on that dataset. Second, the post-processing 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 down.

The curves in Figure 22 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 almost as well as more complex methods such as *GMM* and *KDE*. The method *1-G* thus seems to be a good compromise between simplicity and robustness.

E. Computation Time and Memory Requirements

Background subtraction is often the first stage of more global computer vision systems. For real-time applications, the BS algorithm needs to be light, fast and efficient so the system can process a streaming video at a rate of 30 frames per second. In that case, computation time and memory requirements are critical information that one has to keep in mind before choosing which BS method to implemente. Thus, the performances presented in Figure 14 to 22 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 III). 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.

As one can see from table III, 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.

TABLE III: 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

Also, the reader shall notice that we did not provide processing times 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 down to those of *GMM* on videos with a multimodal background.

Another key information is the minimum amount of memory used by each algorithm. Considering that data are stored in floating point values, we present in table IV the minimum number of floats per pixel each method needs to model the background.

TABLE IV: Memory Requirement.

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

where L, K, M and N are respectively the number of codewords (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 those numbers, *KDE* and *Eigen* are clearly penalized and could hardly be implemented on a embedded system such as an IP camera DSP chip whose local memory is very limited.

TABLE V: Summary of presented results. The number of stars is proportional to the efficiency of the method

	Basic	**1-G**	**MinMax**	**GMM**	**KDE**	**CB$_{RGB}$**	**Eigen**
Static background	***	***	**	***	***	***	***
Multimodal background	*	**	*	***	***	***	*
Noisy background	*	***	*	***	***	***	***
Computation time	***	***	***	**	*	-	*
Memory requirement	***	***	***	**	*	**	*

V. SYNTHESIS OF RESULTS AND CONCLUSION.

In this paper, we presented a comparative study of 7 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*). Those 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 post-processing methods have also been tested.

From the results reported in section IV, an overall summary has been put together and presented in table V. This table highlights the five main conclusions of this study.

1. As some authors already mentioned [15], methods working on grayscale 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 signal-to-noise ratio 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 down the difference between every technique.

Out of those results, one may wonder if there is a win-win situation, *i.e* if there is a method which is 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 bad 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 post-processing filter is implemented.

[1] http://dparks.wikidot.com.

[2] http://imagelab.ing.unimore.it/vssn06.

[3] T. Aach and A. Kaup. Bayesian algorithms for adaptive change detection in image sequences using markov random fields. *Signal Processing: Image Communication*, 7:147–160, 1995.

[4] Y. Benezeth, B. Emile, and C. Rosenberger. Comparative study on foreground detection algorithms for human detection. *International Conference on Image and Graphics*, pages 661–666, 2007.

[5] A.C. Bovik. *Handbook of Image and Video Processing.* Academic Press, Inc. Orlando, FL, USA, 2005.

[6] 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. *Performance Evaluation of Tracking Systems Workshop*, pages 1–8, 2005.

[7] T.H. Chalidabhongse, K. Kim, D. Harwood, and L. Davis. A perturbation method for evaluating background subtraction algorithms. *International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2003.

[8] V. Cheng and N. Kehtarnavaz. A smart camera application: Dsp-based people detection and tracking. *Journal of Electronic Imaging*, 9(3):336–346, 2000.

[9] S.C.S. Cheung and C. Kamath. Robust techniques for background subtraction in urban traffic video. *Visual Communications and Image Processing*, 5308:881–892, 2004.

[10] S.C.S. Cheung and C. Kamath. Robust background subtraction with foreground validation for urban traffic video. *EURASIP Journal on Applied Signal Processing*, 14:2330–2340, 2005.

[11] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. Detecting moving objects, ghosts and shadows in video streams. *Transactions on Pattern Analysis and Machine Intelligence*, pages 1337–1342, 2003.

[12] R. Cucchiara, C. Grana, A. Prati, and R. Vezzani. Probabilistic posture classification for human-behavior analysis. *in Transactions on Systems, Man and Cybernetics, Part A*, 35:42–54, 2005.

[13] A Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. *European Conference on Computer Vision*, pages 751–767, 2000.

[14] S. Ghidary, Y. Nakata, T. Takamori, and M. Hattori. Human detection and localization at indoor environment by homerobot. *International Conference on Systems, Man, and Cybernetics*, 2:1360–1365, 2000.

[15] 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. *International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 113– 120, 2005.

[16] I. Haritaoglu, D. Harwood, and L.S. Davis. W 4: real-time surveillance of people and their activities. *Pattern Analysis and Machine Intelligence*, 22:809–830, 2000.

[17] J. Heikkila and O. Silven. A real-time system for monitoring of cyclists and pedestrians. *Workshop on Visual Surveillance*, pages 74–81, 2004.

[18] S. Herrero and J. Bescòs. Background subtraction techniques: systematic evaluation and comparative analysis. *International conferenve on Advanced Concepts for Intelligent Vision Systems*, pages 33–42, 2009.

[19] W. Hu, T. Tan, L. Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviors. *Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 34(3):334–352, 2004.

[20] T. Inaguma, H. Saji, and H. Nakatani. Hand motion tracking based on a constraint of three-dimensional continuity. *Journal of Electronic Imaging*, 14(1), 2005.

[21] P. KaewTraKulPong and R. Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. *Workshop on Advanced Video-based Surveillance Systems conference*, 2001.

[22] K. Kim, T.H. Chalidabhongse, D. Harwood, and L. Davis. Real-time foreground-background segmentation using codebook model. *Real-Time Imaging*, 11:172–185, 2005.

[23] R. Li, Y. Chen, and X. Zhang. Fast robust eigen-background updating for foreground detection. *International Conference on Image Processing*, pages 1833–1836, 2006.

[24] D. Makris and T. Ellis. Path detection in video surveillance. *in Image and Vision Computing*, 20:895–903, 2002.

[25] D. Makris and T. Ellis. Learning semantic scene models from observing activity in visual surveillance. *Transactions on Systems, Man and Cybernetics, Part B*, 35:397–408, 2005.

[26] A. Mittal and N. Paragios. Motion-based background subtraction using adaptive kernel density

estimation. *Proceedings of the international conference on Computer Vision and Pattern Recognition*, 2004.

[27] N.M. Oliver, B. Rosario, and A.P. Pentland. A bayesian computer vision system for modeling human interactions. *Pattern Analysis and Machine Intelligence*, 22:831–843, 2000.

[28] S. Panahi, S. Sheikhi, S. Hadadan, and N. Gheissari. Evaluation of background subtraction methods. *International conference on Digital Image Computing: Techniques and Applications*, pages 357–364, 2008.

[29] PETS'2001. 2nd international workshop on performance evaluation of tracking and surveillance. 2001.

[30] J. Rymel, J. Renno, D. Greenhill, J. Orwell, and G.A. Jones. Adaptive eigen-backgrounds for object detection. *International Conference on Image Processing*, pages 1847–1850, 2004.

[31] C. Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. *International conference on Computer Vision and Pattern Recognition*, 2, 1999.

[32] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: principles and practice of background maintenance. *International Conference on Computer Vision*, 1:255–261, 1999.

[33] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *Pattern Analysis and Machine Intelligence*, 1997.

[34] Q. Zhou and J. Aggarwal. Tracking and classifying moving objects from video. *Performance Evaluation of Tracking Systems Workshop*, 2001.

[35] Z. Zivkovic. Improved adaptive gaussian mixture model for background subtraction. *International Conference on Pattern Recognition*, 2004.

Yannick Benezeth received his Ph.D. degree in computer science from the University of Orléans, France, in 2009. He also graduated as an engineer from the ENSI de Bourges and obtained a MS degree from the University of Versailles-Saint-Quentin-en-Yvelines, France, in 2006. Since November 2009, he is a research engineer in the Orange Labs - France Telecom research center in Rennes, France. His research interests include computer vision, objects 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 an MSc and PhD degree with honour in computer science from the University of Montreal, Canada in 2003 and 2007. He is currently associate professor in the computer science department of the University of Sherbrooke. He is also a member of MOIVRE (MOdlisation en Imagerie, Vision et REseaux 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, 3D reconstruction, and image quality assessment.

Bruno Emile is an assistant professor at IUT of Châteauroux (France). He obtained his Phd 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 obtained her Phd 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 obtained his Ph.D. degree from the University of Rennes I in 1999. He has been working at the GREYC Laboratory in the Computer Security Research Unit since 2007. His research interests include evaluation of image processing and biometrics.