ViBe: A Universal Background Subtraction Algorithm for Video Sequences

Olivier Barnich and Marc Van Droogenbroeck, *Member, IEEE*

***Abstract*—This paper presents a technique for motion detection that incorporates several innovative mechanisms. For example, our proposed technique stores, for each pixel, a set of values taken in the past at the same location or in the neighborhood. It then compares this set to the current pixel value in order to determine whether that pixel belongs to the background, and adapts the model by choosing randomly which values to substitute from the background model. This approach differs from those based upon the classical belief that the oldest values should be replaced first. Finally, when the pixel is found to be part of the background, its value is propagated into the background model of a neighboring pixel. We describe our method in full details (including pseudo-code and the parameter values used) and compare it to other background subtraction techniques. Efficiency figures show that our method outperforms recent and proven state-of-the-art methods in terms of both computation speed and detection rate. We also analyze the performance of a downscaled version of our algorithm to the absolute minimum of one comparison and one byte of memory per pixel. It appears that even such a simplified version of our algorithm performs better than mainstream techniques.**

***Index Terms*—Background subtraction, computer vision, image motion analysis, image segmentation, learning (artificial intelligence), pixel classification, real-time systems, surveillance, vision and scene understanding, video signal processing.**

I. Introduction

THE number of cameras available worldwide has increased dramatically over the last decade. But this growth has resulted in a huge augmentation of data, meaning that the data are impossible either to store or to handle manually. In order to detect, segment, and track objects automatically in videos, several approaches are possible. Simple motion detection algorithms compare a static background frame with the current frame of a video scene, pixel by pixel. This is the basic principle of background subtraction, which can be formulated as a technique that builds a model of a background and compares this model with the current frame in order to detect zones where a significant difference occurs. The purpose of a background subtraction algorithm is, therefore, to distinguish moving objects (hereafter referred to as the *foreground*) from static, or slow moving, parts of the scene (called *background*). Note that when a static object starts moving, a background subtraction algorithm detects the object in motion as well as a hole left behind in the background (referred to as a *ghost*). Clearly a ghost is irrelevant for motion interpretation and has to be discarded. An alternative definition for the background is that it corresponds to a reference frame with values visible most of the time, that is with the highest appearance probability, but this kind of framework is not straightforward to use in practice.

While a static background model might be appropriate for analyzing short video sequences in a constrained indoor environment, the model is ineffective for most practical situations; a more sophisticated model is, therefore, required. Moreover, the detection of motion is often only a first step in the process of understanding the scene. For example, zones where motion is detected might be filtered and characterized for the detection of unattended bags, gait recognition, face detection, people counting, traffic surveillance, etc. The diversity of scene backgrounds and applications explains why countless papers discuss issues related to background subtraction.

In this paper, we present a universal method for background subtraction. This method has been briefly described in [1] and in a patent [2]. In Section II, we extensively review the literature of background subtraction algorithms. This review presents the major frameworks developed for background subtraction and highlights their respective advantages. We have implemented some of these algorithms in order to compare them with our method. Section III describes our technique and details our major innovations: the background model, the initialization process, and the update mechanism. Section IV discusses experimental results including comparisons with other state-of-the-art algorithms and computational performance. We also present a simplified version of our algorithm which requires only one comparison and one byte of memory per pixel; this is the absolute minimum in terms of comparisons and memory for any background subtraction technique. We show that, even in its simplified form, our algorithm performs better than more sophisticated techniques. Section V concludes the paper.

II. Review of Background Subtraction Algorithms

The problem tackled by background subtraction techniques involves the comparison of an observed image with an estimated image that does not contain any object of interest; this is referred to as the background model (or background image) [3]. This comparison process, called *foreground detection*, divides the observed image into two complementary sets of pixels that cover the entire image: 1) the foreground that contains the objects of interest, and 2) the background, its complementary set.

Manuscript received June 01, 2010; revised August 20, 2010, November 22, 2010; accepted December 01, 2010. Date of publication December 23, 2010; date of current version May 18, 2011. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Mark (Hong-Yuan) Liao.

O. Barnich is with EVS Broadcast Equipment, Liege Science Park, B-4102 Seraing, Belgium (e-mail: olivier.barnich@gmail.com).

M. Van Droogenbroeck is with the Faculty of Applied Sciences, University of Liège, Liège, B-4000 Belgium (e-mail: m.vandroogenbroeck@ulg.ac.be).

Digital Object Identifier 10.1109/TIP.2010.2101613

As stated in [4], it is difficult to specify a gold-standard definition of what a background subtraction technique should detect as a foreground region, as the definition of foreground objects relates to the application level.

Many background subtraction techniques have been proposed with as many models and segmentation strategies, and several surveys are devoted to this topic (see for example [3]–[9]). Some algorithms focus on specific requirements that an ideal background subtraction technique could or should fulfill. According to [7], a background subtraction technique must adapt to gradual or fast illumination changes (changing time of day, clouds, etc), motion changes (camera oscillations), high frequency background objects (e.g., tree leave or branches), and changes in the background geometry (e.g., parked cars). Some applications require background subtraction algorithms to be embedded in the camera, so that the computational load becomes the major concern. For the surveillance of outdoor scenes, robustness against noise and adaptivity to illumination changes are also essential.

Most techniques described in the literature operate on each pixel independently. These techniques relegate entirely to post-processing algorithms the task of adding some form of spatial consistency to their results. Since perturbations often affect individual pixels, this results in local misclassifications. By contrast, the method described by Seiki *et al.* in [10] is based upon the assumption that neighboring blocks of background pixels should follow similar variations over time. While this assumption holds most of the time, especially for pixels belonging to the same background object, it becomes problematic for neighboring pixels located at the border of multiple background objects. Despite this inconvenience, pixels are aggregated into blocks and each $N \times N$ block is processed as an N^2-component vector. A few samples are then collected over time and used to train a principal component analysis (PCA) model for each block. A block of a new video frame is classified as background if its observed image pattern is close to its reconstructions using PCA projection coefficients of 8-neighboring blocks. Such a technique is also described in [11], but it lacks an update mechanism to adapt the block models over time. In [12], the authors focus on the PCA reconstruction error. While the PCA model is also trained with time samples, the resulting model accounts for the whole image. Individual pixels are classified as background or foreground using simple image difference thresholding between the current image and the back-projection in the image space of its PCA coefficients. As for other PCA-based methods, the initialization process and the update mechanism are not described.

A similar approach, the independent component analysis (ICA) of serialized images from a training sequence, is described in [13] in the training of an ICA model. The resulting demixing vector is then computed and compared to that of a new image in order to separate the foreground from a reference background image. The method is said to be highly robust to indoor illumination changes.

A two-level mechanism based upon a classifier is introduced in [14]. A classifier first determines whether an image block belongs to the background. Appropriate blockwise updates of the background image are then carried out in the second stage, depending upon the results of the classification. Classification algorithms are also the basis of other algorithms, as in the one provided in [15], where the background model learns its motion patterns by self organization through artificial neural networks.

Algorithms based upon the framework of compressive sensing perform background subtraction by learning and adapting a low-dimensional compressed representation of the background [16]. The major advantage of this approach lies in the fact that compressive sensing estimates object silhouettes without any auxiliary image reconstruction. On the other hand, objects in the foreground need to occupy only a small portion of the camera view in order to be detected correctly.

Background subtraction is considered to be a sparse error recovery problem in [17]. These authors assumed that each color channel in the video can be independently modeled as the linear combination of the same color channel from other video frames. Consequently, the method they proposed is able to accurately compensate for global changes in the illumination sources without altering the general structure of the frame composition by finding appropriate scalings for each color channel separately.

Background estimation is formulated in [18] as an optimal labeling problem in which each pixel of the background image is labeled with a frame number, indicating which color from the past must be copied. The author's proposed algorithm produces a background image, which is constructed by copying areas from the input frames. Impressive results are shown for static backgrounds but the method is not designed to cope with objects moving slowly in the background, as its outcome is a single static background frame.

The authors of [19] were inspired by the biological mechanism of motion-based perceptual grouping. They propose a spatio-temporal saliency algorithm applicable to scenes with highly dynamic backgrounds, which can be used to perform background subtraction. Comparisons of their algorithm with other state-of-the-art techniques show that their algorithm reduces the average error rate, but at a cost of a prohibitive processing time (several seconds per frame), which makes it unsuitable for real-time applications.

Pixel-based background subtraction techniques compensate for the lack of spatial consistency by a constant updating of their model parameters. The simplest techniques in this category are the use of a static background frame (which has recently been used in [20]), the (weighted) running average [21], first-order low-pass filtering [22], temporal median filtering [23], [24], and the modeling of each pixel with a Gaussian [25]–[27].

Probabilistic methods predict the short-term evolution of a background frame with a Wiener [28] or a Kalman [29] filter. In [28], a frame-level component is added to the pixel-level operations. Its purpose is to detect sudden and global changes in the image and to adapt the background frame accordingly. Median and Gaussian models can be combined to allow inliers (with respect to the median) to have more weight than outliers during the Gaussian modeling, as in [30] or [31]. A method for properly initializing a Gaussian background model from a video sequence in which moving objects are present is proposed in [32].

The W^4 model presented in [33] is a rather simple but nevertheless effective method. It uses three values to represent each

pixel in the background image: the minimum and maximum intensity values, and the maximum intensity difference between consecutive images of the training sequence. The authors of [34] bring a small improvement to the W^4 model together with the incorporation of a technique for shadow detection and removal.

Methods based upon Σ-Δ (sigma-delta) motion detection filters [35]–[37] are popular for embedded processing [38], [39]. As in the case of analog-to-digital converters, a Σ-Δ motion detection filter consists of a simple nonlinear recursive approximation of the background image, which is based upon comparison and on an elementary increment/decrement (usually -1, 0, and 1 are the only possible updating values). The Σ-Δ motion detection filter is, therefore, well suited to many embedded systems that lack a floating point unit.

All these unimodal techniques can lead to satisfactory results in controlled environments while remaining fast, easy to implement, and simple. However, more sophisticated methods are necessary when dealing with videos captured in complex environments where moving background, camera egomotion, and high sensor noise are encountered [5].

Over the years, increasingly complex pixel-level algorithms have been proposed. Among these, by far the most popular is the Gaussian Mixture Model (GMM) [40], [41]. First presented in [40], this model consists of modeling the distribution of the values observed over time at each pixel by a weighted mixture of Gaussians. This background pixel model is able to cope with the multimodal nature of many practical situations and leads to good results when repetitive background motions, such as tree leaves or branches, are encountered. Since its introduction, the model has gained vastly in popularity among the computer vision community [4], [7], [11], [42]–[44], and it is still raising a lot of interest as authors continue to revisit the method and propose enhanced algorithms [45]–[50]. In [51], a particle swarm optimization method is proposed to automatically determine the parameters of the GMM algorithm. The authors of [52] combine a GMM model with a region-based algorithm based upon color histograms and texture information. In their experiments, the authors' method outperform the original GMM algorithm. However, the authors' technique has a considerable computational cost as they only manage to process seven frames of 640×480 pixels per second with an Intel Xeon 5150 processor.

The downside of the GMM algorithm resides in its strong assumptions that the background is more frequently visible than the foreground and that its variance is significantly lower. None of this is valid for every time window. Furthermore, if high- and low-frequency changes are present in the background, its sensitivity cannot be accurately tuned and the model may adapt to the targets themselves or miss the detection of some high speed targets, as detailed in [53]. Also, the estimation of the parameters of the model (especially the variance) can become problematic in real-world noisy environments. This often leaves one with no other choice than to use a fixed variance in a hardware implementation. Finally, it should be noted that the statistical relevance of a Gaussian model is debatable as some authors claim that natural images exhibit non-Gaussian statistics [54].

To avoid the difficult question of finding an appropriate shape for the probability density function, some authors have turned their attention to nonparametric methods to model background distributions. One of the strengths of nonparametric kernel density estimation methods [53], [55]–[59] is their ability to circumvent a part of the delicate parameter estimation step due to the fact that they rely on pixel values observed in the past. For each pixel, these methods build a histogram of background values by accumulating a set of real values sampled from the pixel's recent history. These methods then estimate the probability density function with this histogram to determine whether or not a pixel value of the current frame belongs to the background. Nonparametric kernel density estimation methods can provide fast responses to high-frequency events in the background by directly including newly observed values in the pixel model. However, the ability of these methods to successfully handle concomitant events evolving at various speeds is questionable since they update their pixel models in a first-in first-out manner. This has led some authors to represent background values with two series of values or models: a short term model and a long term model [53], [60]. While this can be a convenient solution for some situations, it leaves open the question of how to determine the proper time interval. In practical terms, handling two models increases the difficulty of fine-tuning the values of the underlying parameters. Our method incorporates a smoother lifespan policy for the sampled values, and as explained in Section IV, it improves the overall detection quality significantly.

In the codebook algorithm [61], [62], each pixel is represented by a codebook, which is a compressed form of background model for a long image sequence. Each codebook is composed of codewords comprising colors transformed by an innovative color distortion metric. An improved codebook incorporating the spatial and temporal context of each pixel has been proposed in [63]. Codebooks are believed to be able to capture background motion over a long period of time with a limited amount of memory. Therefore, codebooks are learned from a typically long training sequence and a codebook update mechanism is described in [62] allowing the algorithm to evolve with the lighting conditions once the training phase is over. However, one should note that the proposed codebook update mechanism does not allow the creation of new codewords, and this can be problematic if permanent structural changes occur in the background (for example, in the case of newly freed parking spots in urban outdoor scenes).

Instead of choosing a particular form of background density model, the authors of [64], [65] use the notion of "consensus." They keep a cache of a given number of last observed background values for each pixel and classify a new value as background if it matches most of the values stored in the pixel's model. One might expect that such an approach would avoid the issues related to deviations from an arbitrarily assumed density model, but since values of pixel models are replaced according to a first-in first-out update policy, they are also prone to the problems discussed previously, for example, the problem of slow and fast motions in the background, unless a large number of pixel samples are stored. The authors state that a cache of 20 samples is the minimum required for the method to be useful, but they also noticed no significant further improvement for caches with more than 60 samples. Consequently, the training period for their algorithm must comprise at least 20 frames. Finally, to cope with lighting changes and objects appearing or

fading in the background, two additional mechanisms (one at the pixel level, a second at the blob level) are added to the consensus algorithm to handle entire objects.

The method proposed in this paper operates differently in handling new or fading objects in the background, without the need to take account of them explicitly. In addition to being faster, our method exhibits an interesting asymmetry in that a ghost (a region of the background discovered once a static object starts moving) is added to the background model more quickly than an object that stops moving. Another major contribution of this paper resides in the proposed update policy. The underlying idea is to gather samples from the past and to update the sample values by ignoring when they were added to the models. This policy ensures a smooth exponential decaying lifespan for the sample values of the pixel models and allows our technique to deal with concomitant events evolving at various speeds with a unique model of a reasonable size for each pixel.

III. Description of a Universal Background Subtraction Technique: ViBe

Background subtraction techniques have to deal with at least three considerations in order to be successful in real applications: 1) what is the model and how does it behave? 2) how is the model initialized? and 3) how is the model updated over time? Answers to these questions are given in the three subsections of this section. Most papers describe the intrinsic model and the updating mechanism. Only a minority of papers discuss initialization, which is critical when a fast response is expected, as in the case inside a digital camera. In addition, there is often a lack of coherence between the model and the update mechanism. For example, some techniques compare the current value of a pixel p to that of a model b with a given tolerance T. They consider that there is a good match if the absolute difference between p and b is lower than T. To be adaptive over time, T is adjusted with respect to the statistical variance of p. But the statistical variance is estimated by a temporal average. Therefore, the adjustment speed is dependent upon the acquisition framerate and on the number of background pixels. This is inappropriate in some cases, as in the case of remote IP cameras whose framerate is determined by the available bandwidth.

We detail in the following a background subtraction technique, called visual background extractor (ViBe). For convenience, we present a complete version of our algorithm in a C-like code in Appendix A.

A. Pixel Model and Classification Process

To some extent, there is no way around the determination, for a given color space, of a probability density function (pdf) for every background pixel or at least the determination of statistical parameters, such as the mean or the variance. Note that with a Gaussian model, there is no distinction to be made as the knowledge of the mean and variance is sufficient to determine the pdf. While the classical approaches to background subtraction and most mainstream techniques rely on pdfs or statistical parameters, the question of their statistical significance is rarely discussed, if not simply ignored. In fact, there is no imperative to compute the pdf as long as the goal of reaching a relevant background segmentation is achieved. An alternative is to consider that one should enhance statistical significance over time, and one way to proceed is to build a model with real observed pixel values. The underlying assumption is that this makes more sense from a stochastic point of view, as already observed values should have a higher probability of being observed again than would values not yet encountered.

Like the authors of [65], we do not opt for a particular form for the pdf, as deviations from the assumed pdf model are ubiquitous. Furthermore, the evaluation of the pdf is a global process and the shape of a pdf is sensitive to outliers. In addition, the estimation of the pdf raises the nonobvious question regarding the number of samples to be considered; the problem of selecting a representative number of samples is intrinsic to all the estimation processes.

If we see the problem of background subtraction as a classification problem, we want to classify a new pixel value with respect to its immediate neighborhood in the chosen color space, so as to avoid the effect of any outliers. This motivates us to model each background pixel with a set of samples instead of with an explicit pixel model. Consequently no estimation of the pdf of the background pixel is performed, and so the current value of the pixel is compared to its closest samples within the collection of samples. This is an important difference in comparison with existing algorithms, in particular with those of consensus-based techniques. A new value is compared to background samples and should be close to some of the sample values instead of the majority of all values. The underlying idea is that it is more reliable to estimate the statistical distribution of a background pixel with a small number of close values than with a large number of samples. This is somewhat similar to ignoring the extremities of the pdf, or to considering only the central part of the underlying pdf by thresholding it. On the other hand, if one trusts the values of the model, it is crucial to select background pixel samples carefully. The classification of pixels in the background, therefore, needs to be conservative, in the sense that only background pixels should populate the background models.

Formally, let us denote by $v(x)$ the value in a given Euclidean color space taken by the pixel located at x in the image, and by v_i a background sample value with an index i. Each background pixel x is modeled by a collection of N background sample values

$$\mathcal{M}(x) = \{v_1, v_2, \ldots, v_N\} \tag{1}$$

taken in previous frames. For now, we ignore the notion of time; this is discussed later.

To classify a pixel value $v(x)$ according to its corresponding model $\mathcal{M}(x)$, we compare it to the *closest* values within the set of samples by defining a sphere $S_R(v(x))$ of radius R centered on $v(x)$. The pixel value $v(x)$ is then classified as background if the cardinality, denoted $\sharp$, of the set intersection of this sphere and the collection of model samples $\mathcal{M}(x)$ is larger than or equal to a given threshold $\sharp_{\min}$. More formally, we compare $\sharp_{\min}$ to

$$\sharp\{S_R(v(x)) \cap \{v_1, v_2, \ldots, v_N\}\}. \tag{2}$$

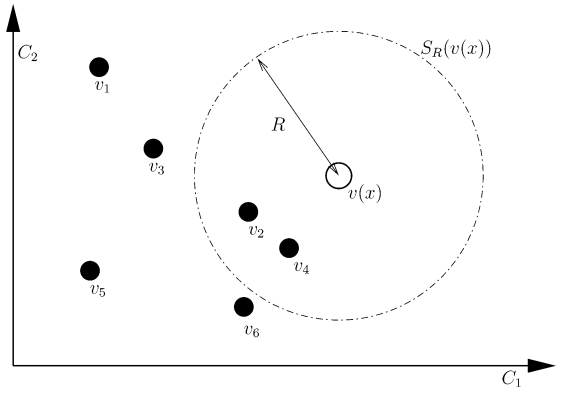

Fig. 1. Comparison of a pixel value with a set of samples in a 2-D Euclidean color space (C_1, C_2). To classify $v(x)$, we count the number of samples of $\mathcal{M}(x)$ intersecting the sphere of radius R centered on $v(x)$.

According to (2), the classification of a pixel value $v(x)$ involves the computation of N distances between $v(x)$ and model samples, and of N comparisons with a thresholded Euclidean distance R. This process is illustrated in Fig. 1. Note that, as we are only interested in finding a few matches, the segmentation process of a pixel can be stopped once $\sharp_{\min}$ matches have been found.

As can easily be seen, the accuracy of our model is determined by two parameters only: the radius R of the sphere and the minimal cardinality $\sharp_{\min}$. Experiments have shown that a unique radius R of 20 (for monochromatic images) and a cardinality of 2 are appropriate (see Section IV-A for a thorough discussion on parameter values). There is no need to adapt these parameters during the background subtraction nor do we need to change them for different pixel locations within the image. Note that since the number of samples N and $\sharp_{\min}$ are chosen to be fixed and since they impact on the same decision, the sensitivity of the model can be adjusted using the following ratio:

$$\frac{\sharp_{\min}}{N} \tag{3}$$

but in all our comparative tests we kept these values unchanged.

So far we have detailed the nature of the model. In the coming sections, we explain how to initialize the model from a single frame and how to update it over time.

B. Background Model Initialization From a Single Frame

Many popular techniques described in the literature, such as [53], [62], and [65], need a sequence of several dozens of frames to initialize their models. Such an approach makes sense from a statistical point of view as it seems imperative to gather a significant amount of data in order to estimate the temporal distribution of the background pixels. But one may wish to segment the foreground of a sequence that is even shorter than the typical initialization sequence required by some background subtraction algorithms. Furthermore, many applications require the ability to provide an *uninterrupted* foreground detection, even in the presence of sudden light changes, which cannot be handled properly by the regular update mechanism of the algorithm. A possible solution to both these issues is to provide a specific model update process that tunes the pixel models to new lighting conditions. But the use of such a dedicated update process is at best delicate, since a sudden illumination may completely alter the chromatic properties of the background.

A more convenient solution is to provide a technique that will initialize the background model from a single frame. Given such a technique, the response to sudden illumination changes is straightforward: the existing background model is discarded and a new model is initialized instantaneously. Furthermore, being able to provide a reliable foreground segmentation as early on as the second frame of a sequence has obvious benefits for short sequences in video-surveillance or for devices that embed a motion detection algorithm.

Since there is no temporal information in a single frame, we use the same assumption as the authors of [66], which is that neighboring pixels share a similar temporal distribution. This justifies the fact that we populate the pixel models with values found in the spatial neighborhood of each pixel. More precisely, we fill them with values randomly taken in their neighborhood in the first frame. The size of the neighborhood needs to be chosen so that it is large enough to comprise a sufficient number of different samples, while keeping in mind that the statistical correlation between values at different locations decreases as the size of the neighborhood increases. From our experiments, selecting samples randomly in the 8-connected neighborhood of each pixel has proved to be satisfactory for images of 640×480 pixels.

Formally, we assume that $t = 0$ indexes the first frame and that $N_G(x)$ is a spatial neighborhood of a pixel location x, therefore

$$\mathcal{M}^0(x) = \{v^0(y \,|\, y \in N_G(x))\} \tag{4}$$

where locations y are chosen randomly according to a uniform law. Note that it is possible for a given $v^0(y)$ to be selected several times (for example if the size of the neighborhood is smaller than the cardinality of $\mathcal{M}^0(x)$) or to not be selected at all. However, this is not an issue if one acknowledges that values in the neighborhood are excellent sample candidates.

This strategy has proved to be successful. The only drawback is that the presence of a moving object in the first frame will introduce an artifact commonly called a *ghost*. According to [24], a ghost is "a set of connected points, detected as in motion but not corresponding to any real moving object." In this particular case, the ghost is caused by the unfortunate initialization of pixel models with samples coming from the moving object. In subsequent frames, the object moves and uncovers the real background, which will be learned progressively through the regular model update process, making the ghost fade over time. Fortunately, as shown in Section IV-C, our update process ensures both a fast model recovery in the presence of a ghost and a slow incorporation of real moving objects into the background model.

C. Updating the Background Model Over Time

In this section, we describe how to continuously update the background model with each new frame. This a crucial step if we want to achieve accurate results over time: the update process must be able to adapt to lighting changes and to handle new objects that appear in a scene.

1) General Discussions on an Update Mechanism: The classification step of our algorithm compares the current pixel value $v^t(x)$ directly to the samples contained in the background model of the previous frame, $\mathcal{M}^{t-1}(x)$ at time $t-1$. Consequently, the question regarding *which* samples have to be memorized by the model and for *how long* is essential. One can see the model as a background memory or background history, as it is often referred to in the literature. The classical approach to the updating of the background history is to discard and replace old values after a number of frames or after a given period of time (typically about a few seconds); the oldest values are substituted by the new ones. Despite the rationale behind it, this substitution principle is not so obvious, as there is no reason to remove a valid value if it corresponds to a background value.

The question of including or not foreground pixel values in the model is one that is always raised for a background subtraction method based upon samples; otherwise the model will not adapt to changing conditions. It boils down to a choice between a conservative and a blind update scheme. Note that kernel-based pdf estimation techniques have a softer approach to updating. They are able to smooth the appearance of a new value by giving it a weight prior to inclusion.

A conservative update policy never includes a sample belonging to a foreground region in the background model. In practice, a pixel sample can be included in the background model only if it has been classified as a background sample. Such a policy seems, at first sight, to be the obvious choice. It actually guarantees a sharp detection of the moving objects, given that they do not share similar colors with the background. Unfortunately, it also leads to deadlock situations and everlasting ghosts: a background sample incorrectly classified as foreground prevents its background pixel model from being updated. This can keep indefinitely the background pixel model from being updated and could cause a permanent misclassification. Unfortunately, many practical scenarios lead to such situations. For example, the location freed by a previously parked car cannot be included in the background model with a purely conservative update scheme, unless a dedicated update mechanism handles such situations.

Blind update is not sensitive to deadlocks: samples are added to the background model whether they have been classified as background or not. The principal drawback of this method is a poor detection of slow moving targets, which are progressively included in the background model. A possible solution consists of using pixel models of a large size, which cover long time windows. But this comes at the price of both an increased memory usage and a higher computational cost. Furthermore, with a first-in first-out model update policy such as those employed in [53] or [65], 300 samples cover a time window of only 10 s [at 30 frames per second (FPS)]. A pixel covered by a slowly moving object for more than 10 s would still be included in the background model.

Strictly speaking, temporal information is not available when the background is masked. But background subtraction is a spatio-temporal process. In order to improve the technique, we could assume, as we proposed in Section III-B, that neighboring pixels are expected to have a similar temporal distribution. According to this hypothesis, the best strategy is, therefore, to adopt a conservative update scheme and to exploit spatial information in order to inject information regarding the background evolution into the background pixel models masked locally by the foreground. This process is common in inpainting, where objects are removed and values are taken in the neighborhood to fill holes [67]. In Section III-C.4, we provide a simple but effective method for exploiting spatial information, which enables us to counter most of the drawbacks of a purely conservative update scheme.

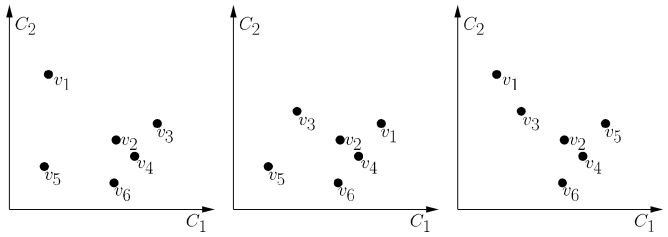

Fig. 2. Three of the six possible outcomes of the updating of a pixel model of size $N = 6$. We assume that values occupy the same color space as in Fig. 1 and that we have decided to update the model. This figure shows three possible models after the update. The decision process for selecting one particular model is random (with equal probabilities).

Our update method incorporates three important components: 1) a memoryless update policy, which ensures a smooth decaying lifespan for the samples stored in the background pixel models, 2) a random time subsampling to extend the time windows covered by the background pixel models, and 3) a mechanism that propagates background pixel samples spatially to ensure spatial consistency and to allow the adaptation of the background pixel models that are masked by the foreground. These components are described, together with our reasons for using them, in the following three subsections.

2) A Memoryless Update Policy: Many sample-based methods use first-in first-out policies to update their models. In order to deal properly with wide ranges of events in the scene background, Wang *et al.* [65] propose the inclusion of large numbers of samples in pixel models. But as stated earlier, this may still not be sufficient for high framerates. Other authors [53], [58] incorporate two temporal submodels to handle both fast and slow modifications. This approach proved to be effective. However, it increases the parametrization problem, in that it makes necessary to determine a greater number of parameter values in order to achieve a practical implementation.

From a theoretical point of view, we believe that it is more appropriate to ensure a monotonic decay of the probability of a sample value to remain inside the set of samples. A pixel model should contain samples from the recent past of the pixel but older samples should not necessarily be discarded.

We propose a method that offers an exponential monotonic decay for the remaining lifespan of the samples. The method improves the time relevance of the estimation by allowing a few old samples to remain in the pixel model. Remember that this approach is combined with a conservative update policy, so that foreground values should never be included in the models.

The technique, illustrated in Fig. 2, is simple but effective: instead of systematically removing the oldest sample from the pixel model, we choose the sample to be discarded randomly according to a uniform probability density function.

The new value then replaces the selected sample. This random strategy contradicts the idea that older values should be replaced first, which is not true for a conservative update policy. A conservative update policy is also necessary for the stability of the process. Indeed, the random update policy produces a nondeterministic background subtraction algorithm (to our knowledge, this is the first background subtraction algorithm to have that property). Only a conservative update policy ensures that the models do not diverge over time. Despite this, there may be slight differences, imperceptible in our experience, between the results of the same sequence processed by our background subtraction algorithm at different times.

Mathematically, the probability of a sample present in the model at time t being preserved after the update of the pixel model is given by $(N-1)/(N)$. Assuming time continuity and the absence of memory in the selection procedure, we can derive a similar probability, denoted $P(t,\, t+dt)$ hereafter, for any further time $t+dt$. This probability is equal to

$$P(t,\, t+dt) = \left(\frac{N-1}{N}\right)^{(t+dt)-t} \tag{5}$$

which can be rewritten as

$$P(t,\, t+dt) = e^{-\ln(\frac{N}{N-1})dt}. \tag{6}$$

This expression shows that the expected remaining lifespan of any sample value of the model decays exponentially. It appears that the probability of a sample being preserved for the interval $(t,\, t+dt)$, assuming that it was included in the model prior to time t, is independent of t. In other words, the past has no effect on the future. This property, called the *memoryless* property, is known to be applicable to an exponential density (see [68]). This is a remarkable and, to our knowledge, unique property in the field of background subtraction. It completely frees us to define a time period for keeping a sample in the history of a pixel and, to some extent, allows the update mechanism to adapt to an arbitrary framerate.

3) Time Subsampling: We have shown how the use of a random replacement policy allow our pixel model to cover a large (theoretically infinite) time window with a limited number of samples. In order to further extend the size of the time window covered by a pixel model of a fixed size, we resort to random time subsampling. The idea is that in many practical situations, it is not necessary to update each background pixel model for each new frame. By making the background update less frequent, we artificially extend the expected lifespan of the background samples. But in the presence of periodic or pseudo-periodic background motions, the use of fixed subsampling intervals might prevent the background model from properly adapting to these motions. This motivates us to use a *random* subsampling policy. In practice, when a pixel value has been classified as belonging to the background, a random process determines whether this value is used to update the corresponding pixel model.

In all our tests, we adopted a time subsampling factor, denoted ϕ, of 16: a background pixel value has one chance in 16 of being selected to update its pixel model. But one may wish to tune this parameter to adjust the length of the time window covered by the pixel model.

4) Spatial Consistency Through Background Samples Propagation: Since we use a conservative update scheme, we have to provide a way of updating the background pixel models that are hidden by the foreground. A popular way of doing this is to use what the authors of the W^4 algorithm [33] call a "*detection support map*" which counts the number of consecutive times that a pixel has been classified as foreground. If this number reaches a given threshold for a particular pixel location, the current pixel value at that location is inserted into the background model. A variant consists of including, in the background, groups of connected foreground pixels that have been found static for a long time, as in [69]. Some authors, like those of the W^4 algorithm and those of the SACON model [64], [65], use a combination of a pixel-level and an object-level background update.

The strength of using a conservative update comes from the fact that pixel values classified as foreground are never included in any background pixel model. While convenient, the support map related methods only delay the inclusion of foreground pixels. Furthermore, since these methods rely on a binary decision, it takes time to recover from a improper inclusion of a genuine foreground object in the background model. A progressive inclusion of foreground samples in the background pixel models is more appropriate.

As stated earlier, we have a different approach. We consider that neighboring background pixels share a similar temporal distribution and that a new background sample of a pixel should also update the models of neighboring pixels. According to this policy, background models hidden by the foreground will be updated with background samples *from neighboring pixel locations* from time to time. This allows a spatial diffusion of information regarding the background evolution that relies on samples classified *exclusively* as background. Our background model is, thus, able to adapt to a changing illumination and to structural evolutions (added or removed background objects) while relying on a *strict* conservative update scheme.

More precisely, let us consider the 4- or 8-connected spatial neighborhood of a pixel x, that is $N_G(x)$, and assume that it has been decided to update the set of samples $\mathcal{M}(x)$ by inserting $v(x)$. We then also use this value $v(x)$ to update the set of samples $\mathcal{M}(y \in N_G(x))$ from one of the pixels in the neighborhood, chosen at random according to a uniform law.

Since pixel models contain many samples, irrelevant information that could accidentally be inserted into the neighborhood model does not affect the accuracy of the detection. Furthermore, the erroneous diffusion of irrelevant information is blocked by the need to match an observed value before it can propagate further. This natural limitation inhibits the diffusion of error.

Note that neither the selection policy nor the spatial propagation method is deterministic. As stated earlier, if the algorithm is run over the same video sequence again, the results will always differ slightly (see Fig. 2). Although unusual, the strategy of allowing a random process to determine which samples are to be discarded proves to be very powerful. This is different from known strategies that introduce a fading factor or that use a long term and a short term history of values.

This concludes the description of our algorithm. ViBe makes no assumption regarding the video stream framerate or color space, nor regarding the scene content, the background itself, or its variability over time. Therefore, we refer to it as a universal method.

IV. Experimental Results

In this section, we determine optimal values for the parameters of ViBe, and compare its results with those of seven other algorithms: two simple methods and five state-of-the art techniques. We also describe some advantageous and intrinsic properties of ViBe, and finally, we illustrate the suitability of ViBe for embedded systems.

For the sake of comparison, we have produced manually ground-truth segmentation maps for subsets of frames taken from two test sequences. The first sequence (called "house") was captured outdoor on a windy day. The second sequence ("pets") was extracted from the PETS2001 public data-set (data-set 3, camera 2, testing). Both sequences are challenging as they feature background motion, moving trees and bushes, and changing illumination conditions. The "pets" sequence is first used below to determine objective values for some of the parameters of ViBe. We then compare ViBe with seven existing algorithms on both sequences.

Many metrics can be used to assess the output of a background subtraction algorithm given a series of ground-truth segmentation maps. These metrics usually involve the following quantities: the number of true positives (TP), which counts the number of correctly detected foreground pixels; the number of false positives (FP), which counts the number of background pixels incorrectly classified as foreground; the number of true negatives (TN), which counts the number of correctly classified background pixels; and the number of false negatives (FN), which accounts for the number of foreground pixels incorrectly classified as background.

The difficulty of assessing background subtraction algorithms originates from the lack of a standardized evaluation framework; some frameworks have been proposed by various authors but mainly with the aim of pointing out the advantages of their own method. According to [6], the metric most widely used in computer vision to assess the performance of a binary classifier is the percentage of correct classification (PCC), which combines all four values

$$\mathrm{PCC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}. \tag{7}$$

This metric was adopted for our comparative tests. Note that the PCC percentage needs to be as high as possible, in order to minimize errors.

A. Determination of Our Own Parameters

From previous discussions, it appears that ViBe has the following parameters:

- the radius R of the sphere used to compare a new pixel value to pixel samples [see (2)];
- the time subsampling factor ϕ;
- the number N of samples stored in each pixel model;
- and the number $\sharp_{\min}$ of close pixel samples needed to classify a new pixel value as background [see (2)].

In our experience, the use of a radius $R = 20$ and a time subsampling factor $\phi = 16$ leads to excellent results in every situation. Note that the use of $R = 20$ is an educated choice, which corresponds to a perceptible difference in color.

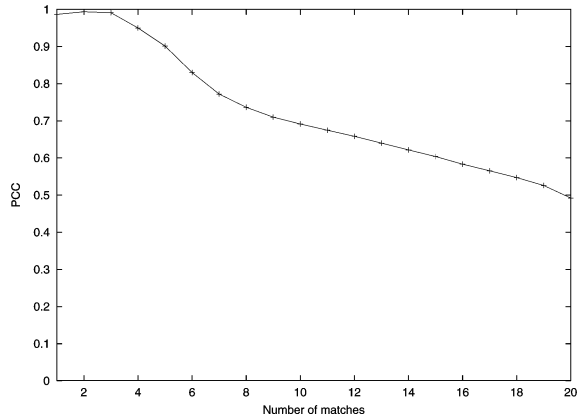

Fig. 3. PCCs for $\sharp_{\min}$ ranging from 1 to 20. The other parameters of ViBe were set to $N = 20$, $R = 20$, and $\phi = 16$.

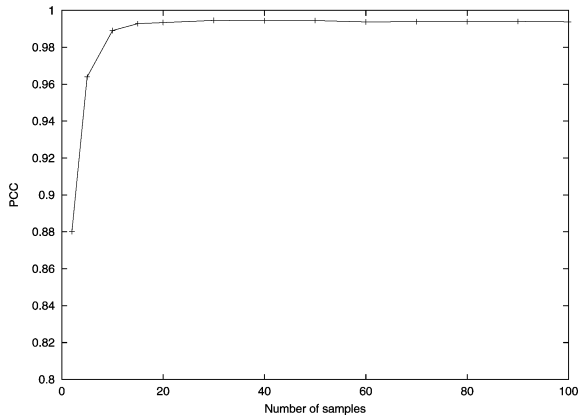

Fig. 4. PCCs given the number of samples collected in a background model.

To determine an optimal value for $\sharp_{\min}$, we compute the evolution of the PCC of ViBe on the "pets" sequence for $\sharp_{\min}$ ranging from 1 to 20. The other parameters were fixed to $N = 20$, $R = 20$, and $\phi = 16$. Fig. 3 shows that the best PCCs are obtained for $\sharp_{\min} = 2$ and $\sharp_{\min} = 3$.

Since a rise in $\sharp_{\min}$ is likely to increase the computational cost of ViBe, we set the optimal value of $\sharp_{\min}$ to $\sharp_{\min} = 2$. Note that in our experience, the use of $\sharp_{\min} = 1$ can lead to excellent results in scenes with a stable background.

Once the value of 2 has been selected for $\sharp_{\min}$, we study the influence of the parameter N on the performance of ViBe. Fig. 4 shows percentages obtained on the "pets" sequence for N ranging from 2 to 50 (R and ϕ were set to 20 and 16). We observe that higher values of N provide a better performance. However, they tend to saturate for values higher than 20. Since as for $\sharp_{\min}$, large N values induce a greater computational cost, we select N at the beginning of the plateau, that is $N = 20$.

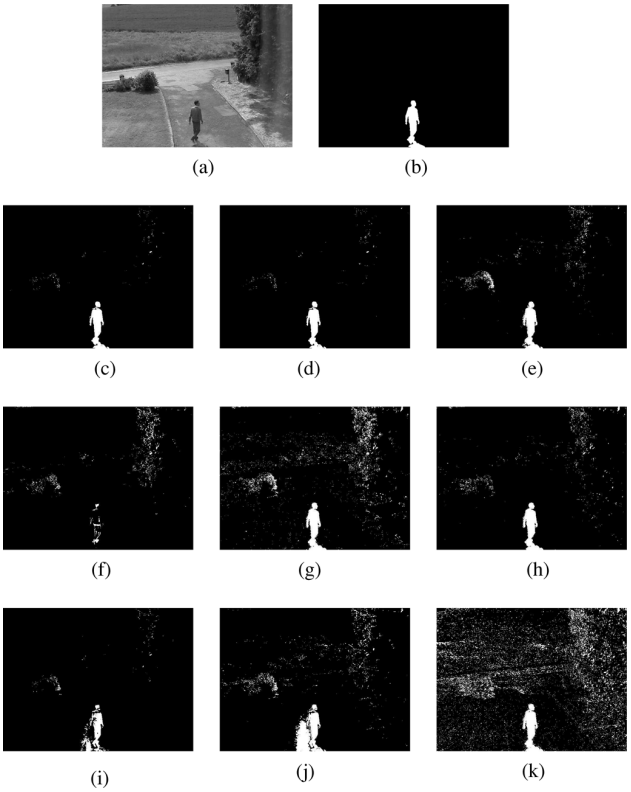

Fig. 5. Comparative background/foreground segmentation maps of nine background subtraction techniques for one frame taken from the "house" sequence. The segmentation maps of ViBe are the closest to the ground-truth reference. (a) Input image. (b) Ground-truth. (c) ViBe (RGB). (d) ViBe (gray). (e) Bayesian histogram. (f) Codebook. (g) EGMM [Zivkovic]. (h) GMM [Li *et al.*]. (i) Gaussian model. (j) First-order filter. (k) Sigma-Delta Z.

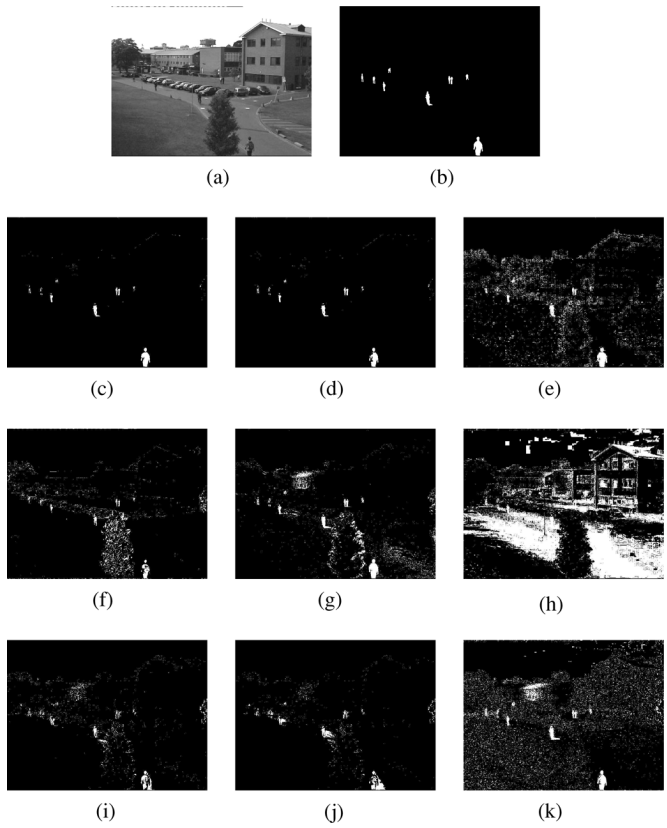

Fig. 6. Comparative background/foreground segmentation maps of nine background subtraction techniques for one frame taken from the "pets" sequence. Here too, the segmentation maps of ViBe are the closest to the ground-truth reference. (a) Input image. (b) Ground-truth. (c) ViBe (RGB). (d) ViBe (gray). (e) Bayesian histogram. (f) Codebook. (g) EGMM [Zivkovic]. (h) GMM [Li *et al.*]. (i) Gaussian model. (j) First order filter. (k) Sigma-Delta Z.

B. Comparison With Other Techniques

We now compare the results of ViBe with those of five state-of-the-art background subtraction algorithms and two basic methods: 1) the Gaussian mixture model proposed in [47] (hereafter referred to as GMM); 2) the Gaussian mixture model of [50] (referred to as EGMM); 3) the Bayesian algorithm based upon histograms introduced in [70]; 4) the codebook algorithm [62]; 5) the zipfian Σ-Δ estimator of [37]; 6) a single Gaussian model with an adaptive variance (named "Gaussian model" hereafter); and 7) the first-order low-pass filter (that is $B_t = \alpha I_t + (1-\alpha)B_{t-1}$, where I_t and B_t are respectively the input and background images at time t), which is used as a baseline.

The first-order low-pass filter was tested using a fading factor α of 0.05 and a detection threshold T of 20. A similar fading factor α of 0.05 was used for the Gaussian model. The GMM of [70] and the Bayesian algorithm of [47] were tested using their implementations available in Intel's IPP image processing library. For the EGMM algorithm of [50], we used the implementation available on the author's website[1]. The authors of the zipfian Σ-Δ filter were kind enough to provide us with their code to test their method. We implemented the codebook algorithm ourselves and used the following parameters: 50 training frames, $\lambda = 34$, $\epsilon_1 = 0.2$, $\epsilon_2 = 50$, $\alpha = 0.4$, and $\beta = 1.2$. ViBe was tested with the default values proposed in this paper: $N = 20, R = 20, \sharp_{\min} = 2$, and $\phi = 16$. Most of the algorithms were tested using the RGB color space; the codebook uses its own color space, and the $\Sigma - \Delta$ filter implementation works on grayscale image. In addition, we implemented a grayscale version of ViBe.

Figs. 5 and 6 show examples of foreground detection for one typical frame of each sequence. Foreground and background pixels are shown in white and black respectively.

Visually, the results of ViBe look better and are the closest to ground-truth references. This is confirmed by the PCC scores; the PCC scores of the nine comparison algorithms for both sequences are shown in Fig. 7.

We also compared the computation times of these nine algorithms with a profiling tool, and expressed the computation times in terms of achievable framerates. Fig. 8 shows their average processing speed on our platform (2.67 GHz Core i7 CPU, 6 GB of RAM, C implementation).

We did not optimize the code of the algorithms explicitly, except in the case of the $\Sigma - \Delta$ algorithm, which was optimized by its authors, the algorithms of the IPP library (GMM and Bayesian histogram), which are optimized for Intel processors, and ViBe to some extent. To speed up operations involving random numbers in ViBe, we used a buffer prefilled with random numbers.

[1]http://staff.science.uva.nl/~zivkovic/DOWNLOAD.html

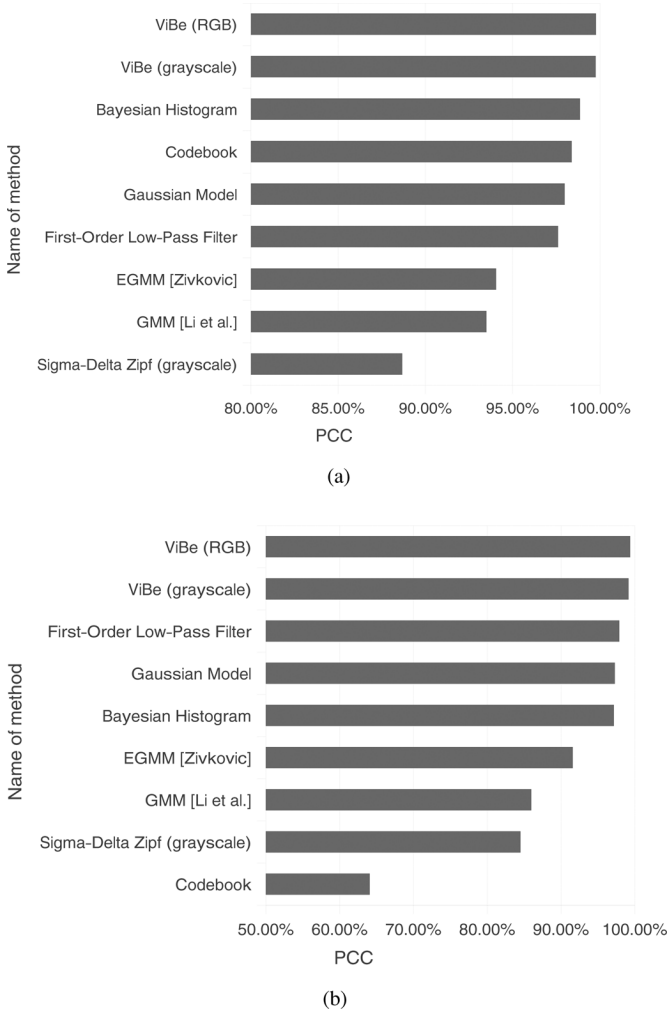

Fig. 7. PCCs of nine background subtraction techniques. ViBe has the highest PCCs. (a) Results for the first sequence ("house"). (b) Results for the second sequence ("pets").

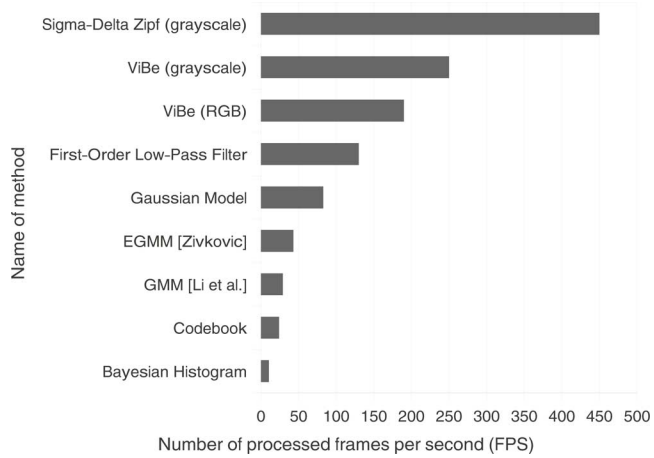

Fig. 8. Processing speed, expressed in terms of FPS, of nine background subtraction techniques for 640×480 pixels wide images.

We see that ViBe clearly outperforms the seven other techniques: its PCCs are the highest for both sequences and its processing speed is as high as a framerate of 200 FPS, that is five times more than algorithms optimized by Intel. Compare these figures to those obtained by the algorithm proposed by Chiu *et al.* [71] recently; they claim to segment 320×240 images at a framerate of around 40 FPS. A simple rescaling to the size of our images lowers this value to 10 FPS.

The only method faster than ViBe is the zipfian Σ-Δ estimator, whose PCC is 12 to 15% smaller than that of ViBe. The authors of the zipfian sigma-delta algorithm provided us with postprocessed segmentation maps of the "house" sequence which exhibit an improved PCC but a the cost of a lower processing speed. One can wonder how it is possible that ViBe runs faster than simpler techniques such as the first-order filter model. We discuss this question in Appendix B.

In terms of PCC scores, only the Bayesian algorithm of [70] based upon histograms competes with ViBe. However, it is more than 20 times slower than ViBe. As shown in Figs. 5 and 6, the grayscale and the color versions of ViBe manage to combine both a very small rate of FP and a sharp detection of the foreground pixels. The low FP rate of ViBe eliminates the need for any postprocessing, which further alleviates the total computational cost of the foreground detection.

Next, we concentrate on the specific strengths of ViBe: fast ghost suppression, intrinsic resilience to camera shake, noise resilience, and suitability for embedding.

C. Faster Ghost Suppression

A background model has to adapt to modifications of the background caused by changing lighting conditions but also to those caused by the addition, removal, or displacement of some of its parts. These events are the principal cause of the appearance of ghosts: regions of connected foreground points that do not correspond to any real object.

When using a detection support map or a related technique to detect and suppress ghosts, it is very hard, if not impossible, to distinguish ghosts from foreground objects that are currently static. As a result, real foreground objects are included in the background model if they remain static for too long. This is a correct behavior since a static foreground object must eventually become part of the background after a given time. It would be better if ghosts were included in the background model more rapidly than real objects, but this is impossible since they cannot be distinguished using a detection support map.

Our spatial update mechanism speeds up the inclusion of ghosts in the background model so that the process is faster than the inclusion of real static foreground objects. This can be achieved because the borders of the foreground objects often exhibit colors that differ noticeably from those of the samples stored in the surrounding background pixel models. When a foreground object stops moving, the information propagation technique described in Section III-C.4 updates the pixel models located at its borders with samples coming from surrounding background pixels. But these samples are irrelevant: their colors do not match at all those of the borders of the object. In subsequent frames, the object remains in the foreground, since background samples cannot diffuse inside the foreground object via its borders.

By contrast, a ghost area often shares similar colors with the surrounding background. When background samples from the area surrounding the ghost try to diffuse inside the ghost, they are

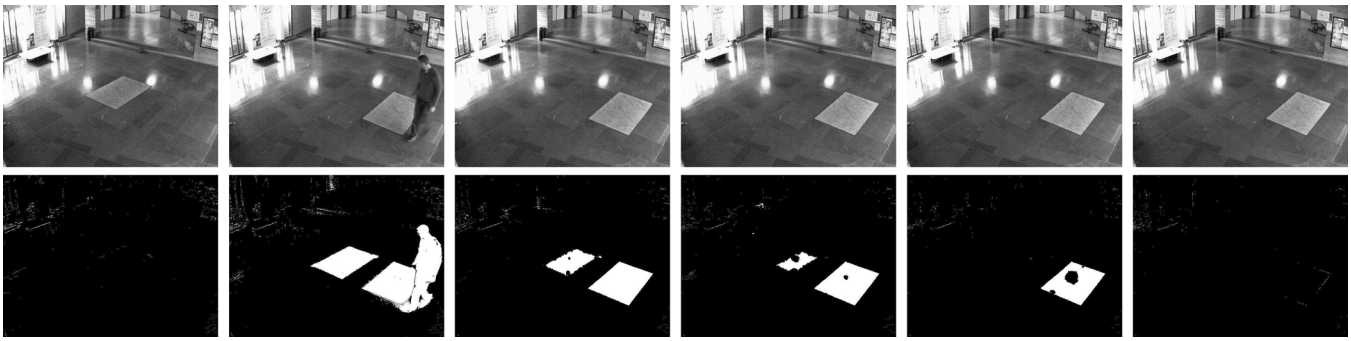

Fig. 9. Fast suppression of a ghost. In this scene, an object (a carpet) is moved, leaving a ghost behind it in the background, and is detected as being part of the foreground. It can be seen that the ghost is absorbed into the background model much faster than the foreground region corresponding to the real physical object.

likely to match the actual color of the image at the locations where they are diffused. As a result, the ghost is progressively eroded until it disappears entirely. Fig. 9 illustrates this discussion.

The speed of this process depends upon the texture of the background: the faster ghost suppressions are obtained with backgrounds void of edges. Furthermore, if the color of the removed object is close to that of the uncovered background area, the absorption of the ghost is faster. When needed, the speed of the ghost suppression process can be tuned by adapting the time subsampling factor ϕ. For example, in the sequence displayed in Fig. 9, if we assume a framerate of 30 FPS, the ghost fades out after 2 s for a time subsampling factor ϕ equal to 1. However, if we set ϕ to 64, it takes 2 min for ViBe to suppress the ghost completely. For the sake of comparison, the Bayesian histogram algorithm suppresses the same ghost area in 5 s.

One may ask how static foreground objects will ultimately be included in the background model. The responsibility for the absorption of foreground pixels into the background lies with the noise inevitably present in the video sequence. Due to the noise, some pixels of the foreground object end up in the background, and then serve as background seeds. Consequently, their models are corrupted with foreground samples. These samples later diffuse into their neighboring models, as a result of the spatial propagation mechanism of the background samples, and allow a slow inclusion of foreground objects in the background.

D. Resistance to Camera Displacements

In many situations, small displacements of the camera are encountered. These small displacements are typically due to vibrations or wind and, with many other techniques, they cause significant numbers of false foreground detections.

Another obvious benefit of the spatial consistency of our background model is an increased robustness against such small camera movements (see Fig. 10). Since samples are shared between neighboring pixel models, small displacements of the camera introduce very few erroneous foreground detections.

ViBe also has the capability of dealing with large displacements of the camera, at the price of a modification of the base algorithm. Since our model is purely pixel-based, we can make it able to handle moving cameras by allowing pixel models to follow the corresponding physical pixels according to the movements of the camera. The movements of the camera can be estimated either using embedded motion sensors or directly from the video stream using an algorithmic technique. This concept is illustrated in Figs. 11 and 12. The first series shows images taken from an old DARPA challenge. The camera pans the scene from left to right and the objective is to follow the car. Fig. 12 shows a similar scenario acquired with a Pan-Tilt Zoom video-surveillance camera; the aim here is to track the person.

Fig. 10. Background/foreground segmentation maps for a slightly moving camera. If spatial propagation is deactivated, the camera motions produce false positives in high-frequency areas (image in the center), while the activation of spatial propagation avoids a significant proportion of false positives (right-hand image).

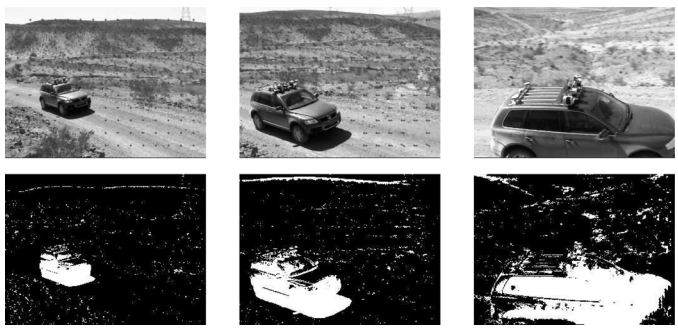

Fig. 11. Background/foreground segmentation maps for a sequence taken with a moving camera (from the DARPA challenge)

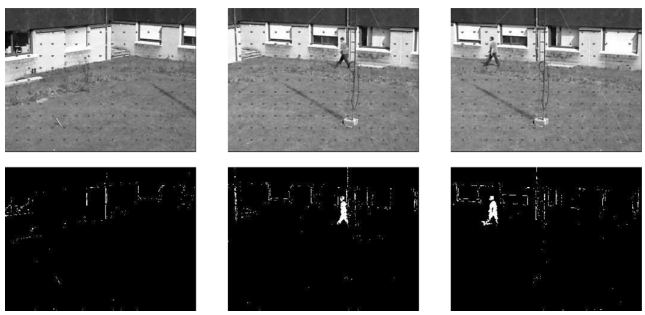

Fig. 12. Background/foreground segmentation maps for a sequence taken with a moving camera (surveillance camera).

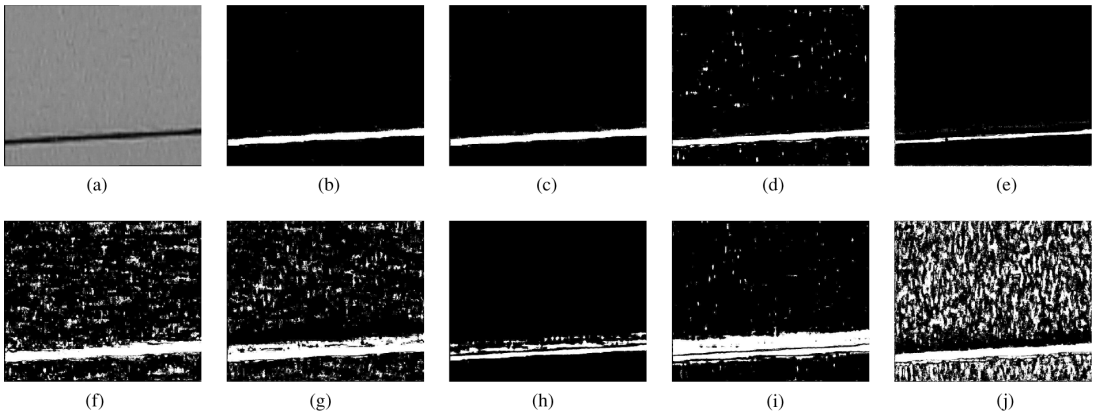

Fig. 13. Background/foreground segmentation maps for one frame taken from the noisy "cable" sequence. (a) Input image. (b) ViBe (RGB). (c) ViBe (grayscale). (d) GMM [Li *et al.*]. (e) Codebook. (f) Bayesian histogram. (g) EGMM [Zivkovic]. (h) Gaussian model. (i) First order filter. (j) Sigma-Delta Zipf.

To produce the images of Figs. 11 and 12, the displacement vector between two consecutive frames is estimated for a subset of background points located on a regularly spaced grid using Lucas and Kanade's optical flow estimator [72]. The global displacement vector of the camera is computed by averaging these pixel-wise displacement vectors. The pixel models are then relocated according to the displacement of the camera inside a larger mosaic reference image. The background model of pixels that correspond to areas seen for the first time is initialized instantaneously using the technique described in Section III-B. It can be seen that, even with such a simple technique, the results displayed in Figs. 11 and 12 are promising.

E. Resilience to Noise

To demonstrate the resilience of ViBe to noise, we compared it to seven other techniques on a difficult noisy sequence (called "cable"). This sequence shows an oscillating electrical cable filmed at a large distance with a $40\times$ optical zoom. As can be seen in Fig. 13(a), the difficult acquisition conditions result in a significant level of noise in the pixel values. Background/foreground segmentation maps displayed in Fig. 13 demonstrate that ViBe is the only technique that manages to combine a low rate of FP with both a precise and accurate detection of the foreground pixels.

Two factors must be credited for ViBe's high resilience to noise. The first originates from our design, allowing the pixel models of ViBe to comprise *exclusively* observed pixel values. The pixel models of ViBe adapt to noise automatically, as they are constructed from noisy pixel values. The second factor is the pure conservative update scheme used by ViBe (see Section III-C). By relying on pixel values classified exclusively as background, the model update policy of ViBe prevents the inclusion of any outlier in the pixel models. As a result, these two factors ensure a continuous adaptation to the noise present in the video sequence while maintaining coherent pixel models.

F. Downscaled Version and Embedded Implementation

Since ViBe has a low computational cost (see Fig. 8) and relies exclusively on integer computations, it is particularly well suited to an embedded implementation. Furthermore, the computational cost of ViBe can be further reduced by using low values for N and $\sharp_{\min}$. Appendix B provides some implementation details and compares the complexity of ViBe with respect to the complexity of the first-order filter model.

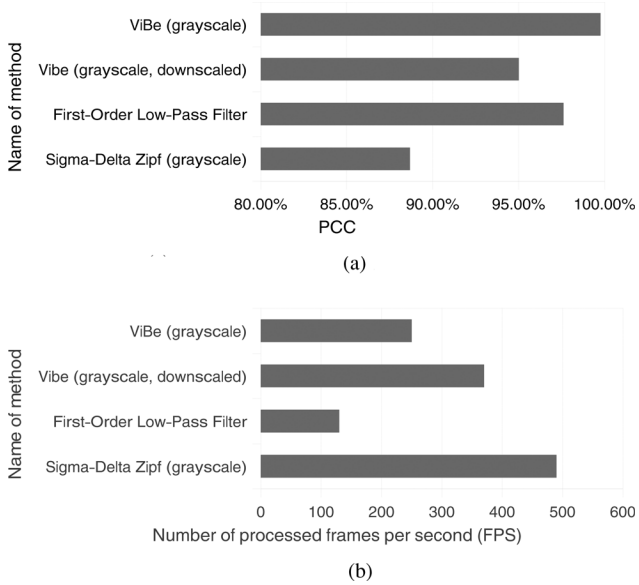

Fig. 14. PCCs and processing speeds of fast techniques, expressed in FPS, including a downscaled version of ViBe which requires only one comparison and one byte of memory per pixel. (a) PCC. (b) FPS for images of 640×480 pixels.

In Fig. 14, we give the PCC scores and framerates for a downscaled version of ViBe, which uses the absolute minimum of one comparison and one byte of memory per pixel. We also give the PCC scores and framerates for the full version of ViBe and for the two faster techniques from our tests in Section IV-B. One can see, on the left hand side of the graph in Fig. 14, that the downscaled version of ViBe maintains a high PCC. Note that its PCC is higher than that of the two GMM-based techniques tested in Section IV-B [see Fig. 7(a)]. In terms of processing speed or framerate, the zipfian Σ-Δ filter method of [37] is the only one to be faster than the downscaled version of ViBe. However, a postprocessing step of the segmentation map is necessary to increase the low PCC score of the zipfian Σ-Δ method, and the computational cost induced by this postprocessing process reduces the framerate significantly.

Fig. 15. Embedded implementation of ViBe in a Canon camera.

To illustrate the low computational cost of ViBe and its simplicity, we embedded our algorithm in a digital camera. The porting work of ViBe on a *Canon PowerShot SD870 IS* was performed with a modified version of the open source alternative firmware CHDK[2]. Parameters of ViBe were set to $N = 5$ and $\sharp_{\min} = 1$. Despite the camera's low speed ARM processor, we managed to process six frames of 320×240 pixels wide images per second on average. The result is shown in Fig. 15.

V. Conclusion

In this paper, we introduced a universal sample-based background subtraction algorithm, called ViBe, which combines three innovative techniques.

First, we proposed a classification model that is based upon a small number of correspondences between a candidate value and the corresponding background pixel model. Second, we explained how ViBe can be initialized with a single frame. This frees us from the need to wait for several seconds to initialize the background model, an advantage for image processing solutions embedded in digital cameras and for short sequences. Finally, we presented our last innovation: an original update mechanism. Instead of keeping samples in the pixel models for a fixed amount of time, we ignore the insertion time of a pixel in the model and select a value to be replaced randomly. This results in a smooth decaying lifespan for the pixel samples, and enables an appropriate behavior of the technique for wider ranges of background evolution rates while reducing the required number of samples needing to be stored for each pixel model. Furthermore, we also ensure the spatial consistency of the background model by allowing samples to diffuse between neighboring pixel models. We observe that the spatial process is responsible for a better resilience to camera motions, but that it also frees us from the need to postprocess segmentation maps in order to obtain spatially coherent results. To be effective, the spatial propagation technique and update mechanism are combined with a strictly conservative update scheme: no foreground pixel value should ever be included in any background model.

After a description of our algorithm, we determined optimal values for all the parameters of the method. Using this set of parameter values, we then compared the classification scores and processing speeds of ViBe with those of seven other background subtraction algorithms on two sequences. ViBe is shown to outperform all of these algorithms while being faster than six of them. Finally, we discussed the performance of a downscaled version of ViBe, which can process more than 350 FPS on our platform. This downscaled version was embedded in a digital camera to prove its suitability for low speed platforms. Interestingly, we found that a version of ViBe downscaled to the absolute minimum amount of resources for any background subtraction algorithm (i.e., one byte of memory and one comparison with a memorized value per pixel) performed better than the state-of-the-art algorithms in terms of the PCC criterion. ViBe might well be a new milestone for the large family of background subtraction algorithms.

Please note that programs and object-code are available at http://www.motiondetection.org.

[2]http://chdk.wikia.com

Appendix A
C-Like Source Code for Our Algorithm

Pseudo-code of ViBe for grayscale images, comprising default values for all the parameters of the method, is given hereafter.

```
// fixed parameters for ViBe
// number of samples per pixel
int N = 20;
// radius of the sphere
int R = 20;
// number of close samples for being
// part of the background (bg)
int #_min = 2;
// amount of random subsampling
int phi = 16;
// data
int width, height;
// current image
byte image[width][height];
// background model
byte samples[width][height][N];
// background/foreground segmentation map
byte segMap[width][height];
// background and foreground identifiers
byte background = 0;
byte foreground = 255;
// for each pixel
for (int x = 0; x < width; x++){
    for (int y = 0; y < height; y++){
        // 1. Compare pixel to background model
        int count = 0, index = 0, dist = 0;
```

```
while ((count < ♯min)&&(index < N)){
    // Euclidean distance computation
    dist = EuclidDist(image[x][y],
        samples[x][y][index]);
    if (dist < R){
        count++;
    }
    index++;
}
// 2. Classify pixel and update model
if (count >= ♯min){
        // store that image[x][y] ∈ background
        segMap[x][y] = background;
        // 3. Update current pixel model
        // get random number between 0 and φ − 1
        int rand = getRandomNumber(0, φ − 1);
        if (rand == 0){ // random subsampling
            // replace randomly chosen sample
            rand = getRandomNumber(0, N − 1);
            samples[x][y][rand] = image[x][y];
        }
        // 4. Update neighboring pixel model
        rand = getRandomNumber(0, φ − 1);
        if (rand == 0){ // random subsampling
            // choose neighboring pixel randomly
            int x_NG, y_NG;
            x_NG = getRandomNeighbrXCoordinate(x);
            y_NG = getRandomNeighbrYCoordinate(y);
            // replace randomly chosen sample
            rand = getRandomNumber(0, N − 1);
            samples[x_NG][y_NG][rand] = image[x][y];
        }
    }
    else{//count < ♯min
        // store that image[x][y] ∈ foreground
        segMap[x][y] = foreground;
    }
  }
}
```

APPENDIX B
IMPLEMENTATION DETAILS, AND COMPLEXITY ANALYSIS OF VIBE AND THE FIRST-ORDER MODEL

As computation times of hardware or software operations might depend upon the processor or the compiler, it is hard to provide an exact analysis of the computation times. Instead, we present the steps involved for the computation of ViBe and the first-order filter model and evaluate the number of operations involved.

For ViBe, the evaluation runs as follows:

- Segmentation step:
  Remember that we compare a new pixel value to background samples to find two matches ($\sharp_{\min} = 2$). Once two matches have been found, we step over to the next pixel and ignore the remaining background samples. Operations involved during the segmentation step are:
  — comparison of the current pixel value with the values of the background model. Most of the time, the two first values of the background model of a pixel are close to the new pixel value. Therefore, we consider 2,5 (byte) comparisons on average per pixel (this is an experimentally estimated value).
  — 1,5 (byte) comparisons of the counter to check if there are at least two matching values in the model; we only need to compare the counter value after the comparison between the current pixel value and the second value of the background model.
- Update step:
  — 1 pixel substitution per 16 background pixels (the update factor, ϕ, is equal to 16). Because we have to choose the value to substitute and access the appropriate memory block in the model, we perform an addition on memory addresses. Then we perform a similar operation, for a pixel in the neighborhood (first we locate which pixel in the neighborhood to select, then which value to substitute).
  In total, we evaluate the cost of the update step as three additions on memory addresses per 16 background pixels.
- Summary (average per pixel, assuming that most pixels belong to the background):
  — 4 subtractions on bytes.
  — $(3)/(16)$ addition on memory addresses.

For the first-order model, we have:

- Segmentation step:
  — 1 pixel comparison between an integer and a double number.
- Update step:
  — two multiplications and one addition on doubles, to perform $B_t = \alpha I_t + (1 - \alpha)B_{t-1}$.
- Summary (per pixel):
  — two multiplications and two additions on doubles

From this comparison, it appears that, once the random numbers are precalculated, the number of operations for ViBe is similar to that of the first-order filter model. However, if processors deal with "integer" (single byte) numbers faster than "double" numbers or if an addition is computed in less time than a multiplication, ViBe is faster than the first-order model.

ACKNOWLEDGMENT

The authors would like to thank A. Manzanera, who provided the code for his algorithms, and to Z. Zivkovic for publishing his code on the Internet.

REFERENCES

[1] O. Barnich and M. Van Droogenbroeck, "ViBe: A powerful random technique to estimate the background in video sequences," in *Proc. Int. Conf. Acoust., Speech Signal Process.*, Apr. 2009, pp. 945–948.

[2] M. Van Droogenbroeck and O. Barnich, Visual Background Extractor p. 36, Jan. 2009, World Intellectual Property Organization, WO 2009/007198.

[3] A. McIvor, "Background subtraction techniques," in *Proc. Image Vis. Comput.*, Auckland, New Zealand, Nov. 2000.

[4] R. Radke, S. Andra, O. Al-Kofahi, and B. Roysam, "Image change detection algorithms: A systematic survey," *IEEE Trans. Image Process.*, vol. 14, pp. 294–307, Mar. 2005.

[5] Y. Benezeth, P. Jodoin, B. Emile, H. Laurent, and C. Rosenberger, "Review and evaluation of commonly-implemented background subtraction algorithms," in *Proc. IEEE Int. Conf. Pattern Recognit.*, Dec. 2008, pp. 1–4.

[6] S. Elhabian, K. El-Sayed, and S. Ahmed, "Moving object detection in spatial domain using background removal techniques—State-of-art," *Recent Pat. Comput. Sci.*, vol. 1, pp. 32–54, Jan. 2008.

[7] M. Piccardi, "Background subtraction techniques: A review," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, The Hague, The Netherlands, Oct. 2004, vol. 4, pp. 3099–3104.

[8] D. Parks and S. Fels, "Evaluation of background subtraction algorithms with post-processing," in *Proc. IEEE Int. Conf. Adv. Video Signal Based Surveillance*, Santa Fe, New Mexico, Sep. 2008, pp. 192–199.

[9] T. Bouwmans, F. El Baf, and B. Vachon, "Statistical background modeling for foreground detection: A survey," in *Handbook of Pattern Recognition and Computer Vision (Volume 4)*. Singapore: World Scientific, Jan. 2010, ch. 3, pp. 181–199.

[10] M. Seki, T. Wada, H. Fujiwara, and K. Sumi, "Background subtraction based on cooccurrence of image variations," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Los Alamitos, CA, Jun. 2003, vol. 2, pp. 65–72.

[11] P. Power and J. Schoonees, "Understanding background mixture models for foreground segmentation," in *Proc. Image Vis. Comput.*, Auckland, New Zealand, Nov. 2002, pp. 267–271.

[12] N. Oliver, B. Rosario, and A. Pentland, "A Bayesian computer vision system for modeling human interactions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 831–843, Aug. 2000.

[13] D.-M. Tsai and S.-C. Lai, "Independent component analysis-based background subtraction for indoor surveillance," *IEEE Trans. Image Process.*, vol. 18, no. 1, pp. 158–167, Jan. 2009.

[14] H.-H. Lin, T.-L. Liu, and J.-C. Chuang, "Learning a scene background model via classification," *IEEE Signal Process. Mag.*, vol. 57, no. 5, pp. 1641–1654, May 2009.

[15] L. Maddalena and A. Petrosino, "A self-organizing approach to background subtraction for visual surveillance applications," *IEEE Trans. Image Process.*, vol. 17, no. 7, pp. 1168–1177, Jul. 2008.

[16] V. Cevher, A. Sankaranarayanan, M. Duarte, D. Reddy, R. Baraniuk, and R. Chellappa, "Compressive sensing for background subtraction," in *Proc. Eur. Conf. Comput. Vis.*, Oct. 2008, pp. 155–168.

[17] M. Dikmen and T. Huang, "Robust estimation of foreground in surveillance videos by sparse error estimation," in *Proc. IEEE Int. Conf. Pattern Recognit.*, Tampa, FL, Dec. 2008, pp. 1–4.

[18] S. Cohen, "Background estimation as a labeling problem," in *Proc. Int. Conf. Comput. Vis.*, Beijing, China, Oct. 2005, vol. 2, pp. 1034–1041.

[19] V. Mahadevan and N. Vasconcelos, "Spatiotemporal saliency in dynamic scenes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 1, pp. 171–177, Jan. 2010.

[20] M. Sivabalakrishnan and D. Manjula, "An efficient foreground detection algorithm for visual surveillance system," *Int. J. Comput. Sci. Network Sec.*, vol. 9, pp. 221–227, May 2009.

[21] A. Cavallaro and T. Ebrahimi, "Video object extraction based on adaptive background and statistical change detection," in *Proc. Vis. Commun. Image Process.*, Jan. 2001, pp. 465–475.

[22] A. El Maadi and X. Maldague, "Outdoor infrared video surveillance: A novel dynamic technique for the subtraction of a changing background of IR images," *Infrared Phys. Technol.*, vol. 49, pp. 261–265, Jan. 2007.

[23] R. Abbott and L. Williams, "Multiple target tracking with lazy background subtraction and connected components analysis," *Mach. Vis. Appl.*, vol. 20, pp. 93–101, Feb. 2009.

[24] B. Shoushtarian and H. Bez, "A practical adaptive approach for dynamic background subtraction using an invariant colour model and object tracking," *Pattern Recognit. Lett.*, vol. 26, pp. 5–26, Jan. 2005.

[25] J. Cezar, C. Rosito, and S. Musse, "A background subtraction model adapted to illumination changes," in *Proc. IEEE Int. Conf. Image Process.*, Oct. 2006, pp. 1817–1820.

[26] J. Davis and V. Sharma, "Robust background-subtraction for person detection in thermal imagery," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Washington, DC, Jun. 2004, vol. 8, p. 128.

[27] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 780–785, Jul. 1997.

[28] K. Toyama, J. Krumm, B. Brumitt, and M. Meyers, "Wallflower: Principles and practice of background maintenance," in *Proc. Int. Conf. Comput. Vis.*, Kerkyra, Greece, Sep. 1999, pp. 255–261.

[29] D. Koller, J. Weber, and J. Malik, "Robust multiple car tracking with occlusion reasoning," in *Proc. Eur. Conf. Comput. Vis.*, Stockholm, Sweden, May 1994, pp. 189–196.

[30] J. Davis and V. Sharma, "Background-subtraction in thermal imagery using contour saliency," *Int. J. Comput. Vis.*, vol. 71, pp. 161–181, Feb. 2007.

[31] C. Jung, "Efficient background subtraction and shadow removal for monochromatic video sequences," *IEEE Trans. Multimedia*, vol. 11, no. 3, pp. 571–577, Apr. 2009.

[32] D. Gutchess, M. Trajkovics, E. Cohen-Solal, D. Lyons, and A. Jain, "A background model initialization algorithm for video surveillance," in *Proc. Int. Conf. Comput. Vis.*, Vancouver, BC, Jul. 2001, vol. 1, pp. 733–740.

[33] I. Haritaoglu, D. Harwood, and L. Davis, "W^4: Real-time surveillance of people and their activities," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 809–830, Aug. 2000.

[34] J. Jacques, C. Jung, and S. Musse, "Background subtraction and shadow detection in grayscale video sequences," in *Proc. Brazilian Symp. Comput. Graph. Image Process.*, Natal, Brazil, Oct. 2005, pp. 189–196.

[35] A. Manzanera and J. Richefeu, "A robust and computationally efficient motion detection algorithm based on sigma-delta background estimation," in *Proc. Indian Conf. Comput. Vis., Graph. Image Process.*, Kolkata, India, Dec. 2004, pp. 46–51.

[36] A. Manzanera and J. Richefeu, "A new motion detection algorithm based on $\Sigma - \Delta$ background estimation," *Pattern Recognit. Lett.*, vol. 28, pp. 320–328, Feb. 2007.

[37] A. Manzanera, "$\Sigma - \Delta$ background subtraction and the Zipf law," in *Proc. Progr. Pattern Recognit., Image Anal. Appl.*, Nov. 2007, pp. 42–51.

[38] L. Lacassagne, A. Manzanera, J. Denoulet, and A. Mérigot, "High performance motion detection: Some trends toward new embedded architectures for vision systems," *J. Real-Time Image Process.*, vol. 4, pp. 127–146, Jun. 2009.

[39] L. Lacassagne, A. Manzanera, and A. Dupret, "Motion detection: Fast and robust algorithms for embedded systems," in *Proc. IEEE Int. Conf. Image Process.*, Cairo, Egypt, Nov. 2009, pp. 3265–3268.

[40] C. Stauffer and E. Grimson, "Adaptive background mixture models for real-time tracking," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Ft. Collins, CO, Jun. 1999, vol. 2, pp. 246–252.

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

[42] B. Lei and L. Xu, "Real-time outdoor video surveillance with robust foreground extraction and object tracking via multi-state transition management," *Pattern Recognit. Lett.*, vol. 27, pp. 1816–1825, Nov. 2006.

[43] Y. Wang, T. Tan, K. Loe, and J. Wu, "A probabilistic approach for foreground and shadow segmentation in monocular image sequences," *Pattern Recognit.*, vol. 38, pp. 1937–1946, Nov. 2005.

[44] Y. Wang, K. Loe, and J. Wu, "A dynamic conditional random field model for foreground and shadow segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 2, pp. 279–289, Feb. 2006.

[45] O. Barnich, S. Jodogne, and M. Van Droogenbroeck, "Robust analysis of silhouettes by morphological size distributions," in *Advanced Concepts for Intelligent Vision Systems (ACIVS 2006), Vol. 4179 of Lecture Notes on Computer Science*. New York: Springer-Verlag, Sep. 2006, pp. 734–745.

[46] J.-S. Hu and T.-M. Su, "Robust background subtraction with shadow and highlight removal for indoor surveillance," *EURASIP J. Appl. Signal Process.*, vol. 2007, pp. 108–108, Jan. 2007.
[47] P. KaewTraKulPong and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection," in *Proc. Eur. Workshop Adv. Video Based Surveillance Syst.*, London, U.K., Sep. 2001.
[48] D. Lee, "Effective Gaussian mixture learning for video background subtraction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 5, pp. 827–832, May 2005.
[49] Q. Zang and R. Klette, "Robust background subtraction and maintenance," in *Proc. IEEE Int. Conf. Pattern Recognit.*, Washington, DC, Aug. 2004, vol. 2, pp. 90–93.
[50] Z. Zivkovic, "Improved adaptive gausian mixture model for background subtraction," in *Proc. IEEE Int. Conf. Pattern Recognit.*, Cambridge, U.K., Aug. 2004, vol. 2, pp. 28–31.
[51] B. White and M. Shah, "Automatically tuning background subtraction parameters using particle swarm optimization," in *Proc. IEEE Int. Conf. Multimedia Expo*, Beijing, China, Jul. 2007, pp. 1826–1829.
[52] P. Varcheie, M. Sills-Lavoie, and G.-A. Bilodeau, "A multiscale region-based motion detection and background subtraction algorithm," *Sensors*, vol. 10, pp. 1041–1061, Jan. 2010.
[53] A. Elgammal, D. Harwood, and L. Davis, "Non-parametric model for background subtraction," in *Proc. 6th Eur. Conf. Comput. Vis.*, London, U.K., Jun.–Jul. 2000, pp. 751–767.
[54] A. Srivastava, A. Lee, E. Simoncelli, and S.-C. Zhu, "On advances in statistical modeling of natural images," *J. Math. Imag. Vis.*, vol. 18, pp. 17–33, Jan. 2003.
[55] A. Elgammal, R. Duraiswami, D. Harwood, and L. Davis, "Background and foreground modeling using nonparametric kernel density estimation for visual surveillance," *Proc. IEEE*, vol. 90, no. 7, pp. 1151–1163, Jul. 2002.
[56] A. Mittal and N. Paragios, "Motion-based background subtraction using adaptive kernel density estimation," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Los Alamitos, CA, Jun.–Jul. 2004, vol. 2, pp. 302–309.
[57] Y. Sheikh and M. Shah, "Bayesian modeling of dynamic scenes for object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 11, pp. 1778–1792, Nov. 2005.
[58] Z. Zivkovic and F. van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognit. Lett.*, vol. 27, pp. 773–780, May 2006.
[59] A. Tavakkoli, M. Nicolescu, G. Bebis, and M. Nicolescu, "Non-parametric statistical background modeling for efficient foreground region detection," *Mach. Vis. Appl.*, vol. 20, pp. 395–409, Oct. 2008.
[60] E. Monari and C. Pasqual, "Fusion of background estimation approaches for motion detection in non-static backgrounds," in *Proc. IEEE Int. Conf. Adv. Video Signal Based Surveillance*, London, U.K., Sep. 2007, pp. 347–352.
[61] K. Kim, T. Chalidabhongse, D. Harwood, and L. Davis, "Background modeling and subtraction by codebook construction," in *Proc. IEEE Int. Conf. Image Process.*, Singapore, Oct. 2004, vol. 5, pp. 3061–3064.
[62] K. Kim, T. Chalidabhongse, D. Harwood, and L. Davis, "Real-time foreground-background segmentation using codebook model," *Real-Time Imag.*, vol. 11, Special Issue on Video Object Processing, pp. 172–185, Jun. 2005.
[63] M. Wu and X. Peng, "Spatio-temporal context for codebook-based dynamic background subtraction," *Int. J. Electron. Commun.*, vol. 64, no. 8, pp. 739–747, 2010.
[64] H. Wang and D. Suter, "Background subtraction based on a robust consensus method," in *Proc. IEEE Int. Conf. Pattern Recognit.*, Washington, DC, Aug. 2006, pp. 223–226.
[65] H. Wang and D. Suter, "A consensus-based method for tracking: Modelling background scenario and foreground appearance," *Pattern Recognit.*, vol. 40, pp. 1091–1105, Mar. 2007.
[66] P.-M. Jodoin, M. Mignotte, and J. Konrad, "Statistical background subtraction using spatial cues," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 12, pp. 1758–1763, Dec. 2007.
[67] A. Criminisi, P. Pérez, and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," *IEEE Trans. Image Process.*, vol. 13, no. 9, pp. 1200–1212, Sep. 2004.
[68] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*. New York: McGraw-Hill, 1984.
[69] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting moving objects, ghosts, and shadows in video streams," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 10, pp. 1337–1342, Oct. 2003.
[70] L. Li, W. Huang, I. Gu, and Q. Tian, "Foreground object detection from videos containing complex background," in *Proc. ACM Int. Conf. Multimedia*, Berkeley, CA, Nov. 2003, pp. 2–10.
[71] C.-C. Chiu, M.-Y. Ku, and L.-W. Liang, "A robust object segmentation system using a probability-based background extraction algorithm," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 4, pp. 518–528, Apr. 2010.
[72] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. Int. Joint Conf. Artif. Intell.*, Vancouver, BC, Apr. 1981, pp. 674–679.

Olivier Barnich was born in Liège, Belgium. He received the electrical engineering degree and Ph.D. degree from the University of Liège, Belgium, in 2004 and 2010, respectively.

He is currently working for EVS Broadcast Equipment, Seraing, Belgium, a company that delivers production and playout systems for the broadcast of sports, news and entertainment. His research interests are mainly focused on motion detection, human detection and recognition, and machine learning enabled computer vision.

Marc Van Droogenbroeck (M'99) received the degree in electrical engineering and the Ph.D. degree from the University of Louvain (UCL, Belgium) in 1990 and 1994, respectively.

Since 1998, M. Van Droogenbroeck has been a member of the Faculty of Applied Sciences at the University of Liège, Belgium, where he is currently a Professor. During his Ph.D. he spent two years at the Center of Mathematical Morphology (CMM) of the School of Mines of Paris. In April 1994, he joined the New Development Department, Belgacom. He was appointed as the Head of the Belgian Delegation in the ISO/MPEG Committee and as a representative to the World Wide Web Consortium for two years. In 2003, he was a visiting Scientist at the CSIRO, Australia. His current interests are image processing, computer vision, mathematical morphology, fast algorithms, and video surveillance.